ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Навчально-науковий інститут Інформаційних технологій

Кафедра інженерії програмного забезпечення

Пояснювальна записка

до бакалаврської роботи
на ступінь вищої освіти бакалавр

на тему: «**РОЗРОБКА КРОСПЛАТФОРМНОГО КРИПТОВАЛЮТНОГО ГАМАНЦЯ НА C++**»

Виконав: студент 4 курсу, групи ПД-44
спеціальності
121 Інженерія програмного забезпечення
(шифр і назва спеціальності)
Івлєв Р.В.
(прізвище та ініціали)
Керівник Золотухіна О.А.
(прізвище та ініціали)
Рецензент _____
(прізвище та ініціали)

Київ – 2022

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Навчально-науковий інститут Інформаційних технологій

Кафедра Інженерії програмного забезпечення
Ступінь вищої освіти - «Бакалавр»
Напрям підготовки -121 - Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри
Телекомунікаційних технологій
Негоденко О.В.
“ ” 2022 року

М

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Івлєв Ростислав Володимирович
(прізвище, ім‘я, по батькові)

1.Тема роботи: «Розробка кросплатформного криптовалютного гаманця на С++»

Керівник роботи Золотухіна Оксана Анатоліївна, кандидат технічних наук, доцент ,
(прізвище, ім‘я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від “ 16 ” лютого 2022 року № 22.

2.Строк подання студентом роботи 03.06.2022
3.Вхідні дані до роботи:
Науково-технічна література з питань, пов’язаних з технологією розподіленого реєстру;
4.Зміст розрахунково-пояснювальної записки(перелік питань, які потрібно розробити):
4.1.Криптова люти та технологія розподіленого реєстру.
4.2. Аналіз існуючих криптовалютних гаманців.
4.3. Опис використаних технології.
4.4. Опис проектування системи.
5.Перелік графічного матеріалу:
5.1.Алгоритм створення гаманця
5.2.Діаграма класів
5.3.Діаграма послідовності створення транзакції
5.4.Висновки

6.Дата видачі завдання 11.04.2022

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури	11.04 - 16.04	
2	Вивчення та аналіз задачі	17.04 - 18.04	
3	Аналіз аналогів	19.04 - 24.04	
4	Дослідження програмних засобів	25.04 - 27.04	
5	Розробка структури бібліотеки	28.04 – 5.05	
6	Програмна реалізація системи	6.05 – 13.05	
7	Написання вступу, реферату, висновку	14.05 – 23.05	
8	Розробка обов‘язкових демонстраційних матеріалів	24.05 – 28.05	
9	Попередній захист роботи	29.05– 03.06	
10	Подання роботи в деканат	06.06	

Студент _____ Івлєв Р.В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Золотухіна О.А
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Текстова частина бакалаврської роботи 43 с., рис. 18, 19 джерел.

Об’єкт дослідження – процес контролю криптовалютних гаманців.

Предмет дослідження – програмне забезпечення для контролю криптовалютних гаманців, що забезпечують роботу з кількома DLT мережами.

Мета роботи – спрощення процесу контролю кросплатформних криптовалютних гаманців та підвищення їх рівня безпеки.

Проведено дослідження теоретичної частини роботи криптовалют та криптовалютних гаманців, виконано аналіз існуючих аналогів додатків. Встановлено їх переваги та недоліки. В результаті аналізу було визначено основні проблеми додатків аналогів та потреби користувачів. Обґрунтовано вибір мов програмування C++ та Kotlin, середовищ розробки Visual Studio, Android Studio, бібліотеки криптографічних алгоритмів Trezor-crypto та утиліти CMake. Розроблено варіанти використання та в результаті аналізу архітектурних рішень було сформульовано пропозицій до розробки з використанням цієї бібліотеки для забезпечення високого рівня безпеки. Розроблено логіку, алгоритм та базовий функціонал додатку.

Галузь використання – бібліотеку може використовувати будь-яка людина для розробки критовалютного гаманця.

КРИПТОВАЛЮТА, DLT МЕРЕЖИ, БЛОКЧЕЙН, C+, KOTLIN

ЗМІСТ

ВСТУП

Сьогодні криптовалюта все більше використовується в світі, та все більше з’являється шляхів її використання. Але більшість криптовалютних гаманців обмежені в тому чи іншому функціоналі. А створення нових гаманців спотикається о бібліотеки які вже були створення, найчастіше вони спрямовані на роботу з одним розподіленим реєстром, а ті які підтримують їх кілька, часто громіздкі що ускладнює чи зовсім не дозволяє самостійно додавати підтримку DLT мереж.

Функціональність та різновиди криптовалютних гаманців дуже добре описані на сайті криптовалюти [7]. З статі про безпеку гаманців від Cossack Labs [12], займається створенням рішення для захисту даних, стає зрозуміло, що більшість гаманців знаходяться в зоні ризику.

Основною інформаційною базою послужили роботи Светлин Накова [5], Мишеля Раушса [3], Юлії Потапенко [12], Сід Логана [13] та фактичні дані Kotlin [15], Mozilla [19].

Об’єктом дослідження є процес контролю криптовалютних гаманців.

Предметом дослідження є програмне забезпечення для контролю криптовалютних гаманців, що забезпечують роботу з кількома DLT мережами.

Метою роботи є спрощення процесу контролю кросплатформних криптовалютних гаманців та підвищення їх рівня безпеки.

Для досягнення поставленої мети виконано наступні завдання:

- проаналізовано поняття криптовалют, технологій розподіленого реєстру, криптовалютних гаманців;
- порівняно існуючих криптовалютних гаманців, виділено їх основні переваги та недоліки;
- обґрунтувано обрання технологій і середовища розробки;
- розроблено варіанти використання програмного продукту;
- побудувано структурну модель використання;

- проаналізовано архітектурні шаблони та змодельовано програмне забезпечення;
- розроблено та протестовано програмне забезпечення.

Методика дослідження: було вивчено джерела, на їх основі окреслено поняття криптовалют, розподіленого реєстру, криптовалютних гаманців та технології на яких вони базуються або які вони використовують. Також було порівняно між собою існуючі додатки такі, як Electrum, MetaMask. Exodus та Trust Wallet. На основі зібраної інформації було сформульовані вимоги до розроблюваного продукту та обраний стек технологій.

Виходячи з піклування безпеки було рішено створювати бібліотеку під Windows та портувати її на інші платформи, щоб зніжити кількість залежностей. Також бібліотека була побудовано так, щоб можна було легко додавати підтримку нових DLT мереж, а використання бібліотеки Trezor-crypto надає підтримку багатьох криптографічних алгоритмів. Це дозволяє створювати криптовалютні гаманці швидко та легко при цьому не хвилюватися за дані які використовує додаток.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Криптовалюта. Технологія розподіленого реєстру

Криптовалюта це одна з видів цифрової валюти, вартість якої визначається торговою ціною по відношенню до реальної валюти такої як долар США [1]. Вона використовує криптографію не тільки для безпеки, а також для запобігання подвійного використання й шахрайства.

На даний час всі криптовалюти використовують технологію розподіленого реєстру (Distributed Ledger Technology або DLT) – це технологія зберігання даних, де інформація спільно використовується та зберігається одночасно на різних пристроях, які знаходяться в різних місцях, утворювачі мережу, в якої нема єдиного контролю.

Технологія розподіленого реєстру не має однієї стандартизованої реалізацій через це існують багато концепції з яких з'являються цілі типи DLT мереж. На рис. 1.1 наведено існуючі типи розподіленої книги.

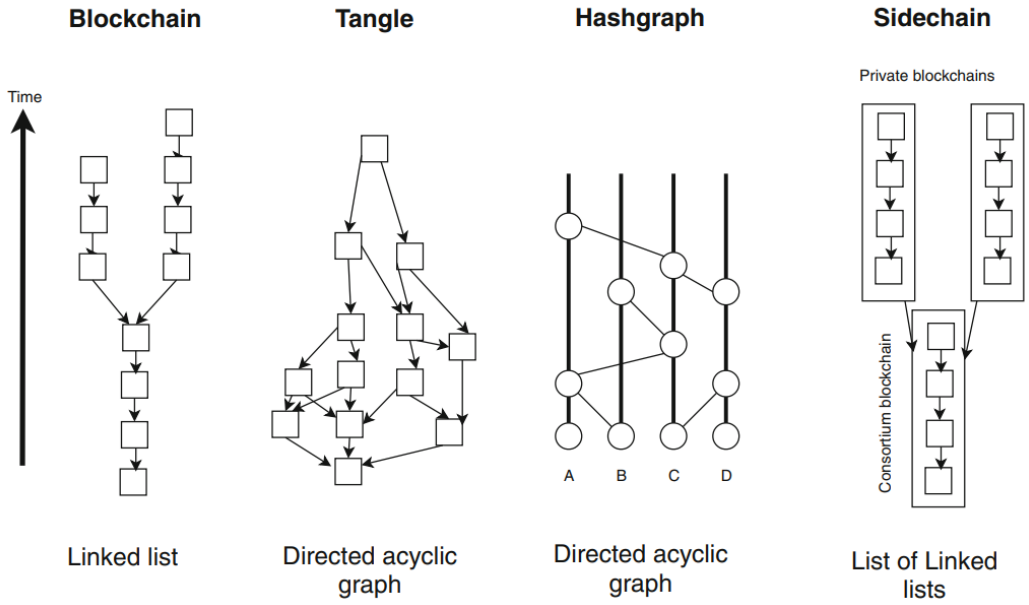


Рисунок 1.1 - Існуючи типи розподіленої книги [2]

Кожна DLT мережа складається з вузлів, вони можуть бути різного типу та мати різню функціональність в залежності від розподіленого реєстру. Але наступні функцій обов'язково присутні хоча б у одного типу вузлів у кожній DLT мережі: реєстр, опрацювання запитів, механізм консенсусу.

Консенсус в DLT мережах це механізм перевірки транзакцій, при якому позитивне рішення ґрунтується на відсутності заперечення. Найбільш відомі протоколи консенсусу це Proof of Work (PoW) та Proof of Stake (PoS).

Реєстр зберігає всі транзакції які були підтверджені та виконані. Перед тим як транзакція потрапляє до реєстру вона повинна пройти наступні кроки [3]:

- Користувач створює транзакція відправляє її до мережі. Транзакція – це запит на будь яку маніпуляцію з реєстром, структура транзакції залежить від DLT мережі.
- Коли транзакція потрапляє до одного з вузлів та відповідає структурі вона попадає в лог (мемпул) – це набір не підтверджених транзакції який знаходиться на рівні вузла.
- Вузол випадково вибирає транзакції з свого лога та утворює запис кандидат.
- Запис кандидат підтверджується відповідно до протоколу консенсусу який використовується у DLT мережі, після чого становиться підтвердженою записом кандидатом яка поширюється між іншими вузлами.
- Коли інші вузли отримають поширений запис кандидат, вони теж підтверджують його відповідно до протоколу консенсусу, після чого цей запис додається до журналу. Журнал – це копія реєстру яка зберігається на вузлі. Дані в журналі можуть відрізнятися від вузла до вузла.
- Останній крок - це злиття синхронізованих журналів які утворюють реєстр.

DLT мережі можна класифікувати за відкритістю контролю на такі типи [4]:

- Загальнодоступні - любий користувач може запустити власний вузол без потреби отримання дозволу від власника.

- Приватні - відміно від загальнодоступного потребує отримання дозволу, що дозволяє обмежити доступ к читанню та здійснюванню транзакцій.
- Гібридні - поєднують в собі особливості попередніх типів що дозволяє власнику обмежити які дані будуть публічні, а які приватні.

Для того щоб користувач міг взаємодіяти з DLT мережею, йому потрібно мати адресу. Адреса складається з цифр та літер в залежності від розподіленого реєстру може мати різний вид та розмір, вона генерується з публічного ключа.

Публічний (відкритий) ключ та приватний (закритий) ключ утворюють ключову пару, алгоритм шифрування який використовує її називають асиметричним або криптографією з відкритим ключем. Принцип асиметричного шифрування полягає в тому, щоб за допомоги одного ключа шифрувати дані, а іншим ключем дешифрувати.

Приватний ключ уявляє собою велике випадкове число, найчастіше він має розмір 128, 192 або 256 біт. Для того щоб приватний ключ був менше розміром його представляють в шістнадцятковій, Base58 або Base64 формі.

Публічний ключ математично зв’язаний з приватним ключем. Процес створення публічного ключа залежить від криптосистеми яку використовують. Основна особливість публічного ключа в тому що, з нього неможливо отримати приватний ключа, а в деяких криптосистемах ключі неможливо отримати один з одного, наприклад як в RSA.

Асиметричне шифрування в DLT мережах використовується також для електронного цифрового підпису (ЕЦП) що допомагає ідентифікувати користувача та перевірити цілісність транзакції. В процесі цифрового підпису окрім криптографії з відкритим ключем бере участь криптографічна хеш-функція, вона перетворює дані, будь-якого розміру, в хеш значення, фіксованого розміру. Процес електронного цифрового підписування даних наведень на рис. 1.2.

Деякі DLT мережі окрім даних можуть містити програми які виконуються при попередньо визначених умовах, та називаються смарт контрактами. Смарт контракт зберігається в розподіленому реєстрі та має власну адресу. Коли смарт контрактом можна користуватися за допомогою графічного інтерфейсу це

називають децентралізованим додатком (dApp), зазвичай використовує більше одного контракту.

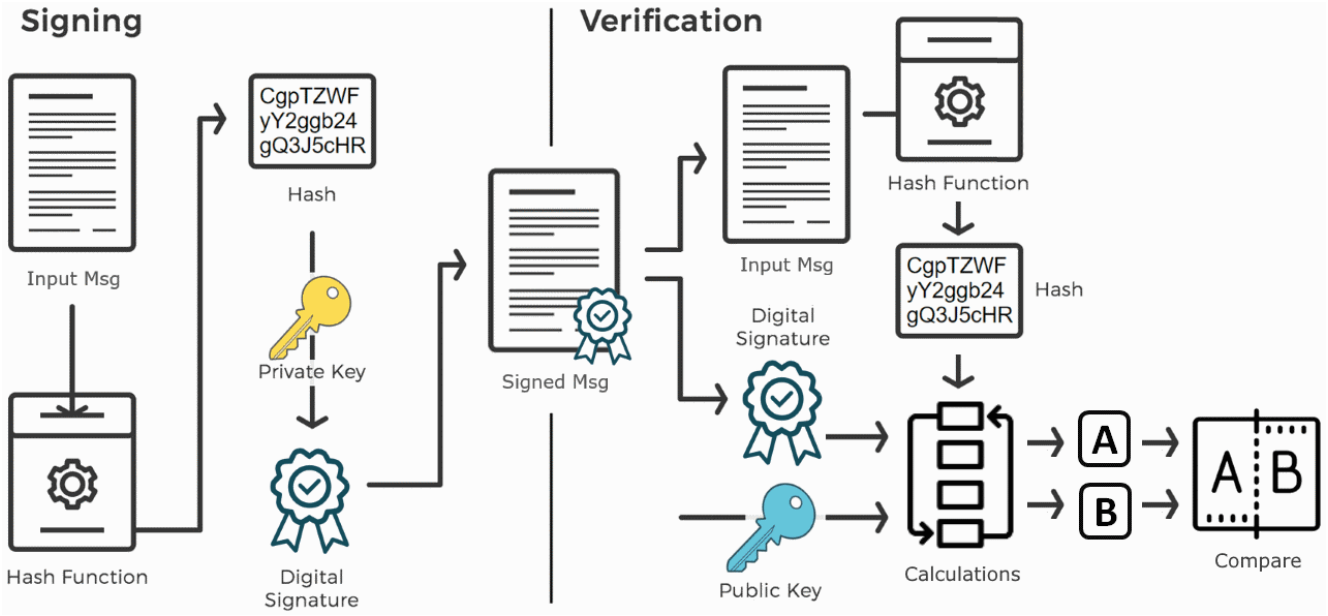


Рисунок 1.2 - Процес електронного цифрового підписування даних [5]

Криптовалюту можна поділити на два типи:

- Коїни – це криптовалюта яка є частиною розподіленого реєстру, в більшості DLT мережах потрібні для проведення транзакції.
- Токени – це криптовалюта яка створена на вже існуючому розподіленому реєстрі за допомогою смарт контракту [6].

1.2 Криптовалютний гаманець

Криптовалютний гаманець це програма що дозволяє користувачеві створювати, зберігати та використовувати ключові пари, створювати та підписувати транзакцій, відправляти транзакцій до DLT мережі.

Гаманці можна класифікують за функціональністю на:

- Гаманець повного обслуговування (Full-Service Wallets), головною перевагою цього типу це простота використання тому що, всі функції в одній програмі, але це також й недолік через те що приватний ключ

зберігається на пристрою який завжди підключень к інтернету завжди є ризик його викрадення [7].

- Гаманець тільки для підписання (Signing-Only Wallets) дозволяє тільки генерувати ключову пару та підписувати транзакції, а всі інші функції виконую інша програма [7].

Також всі гаманці діляться на «гарячі» та «холодні». Відмінність між ним тільки в тому, що «гарячі» гаманці потребують інтернет для роботи, а «холодні» ні. Зазвичай «гарячі» гаманці використовують для частої роботи з криптовалютою та для зберігання їх невеликої кількості, а «холодні» навпаки.

«Гарячі» гаманці можна поділити на: мобільні та комп'ютерні, які часто об'єднають та називають програмними, онлайн (веб). А «Холодні» ділять на апаратні та паперові.

Апаратні гаманці найбезпечніші з усіх видів, найчастіше зустрічаються в форматі USB й більшість з них - це гаманці тільки для підписування.

Переваги:

- Найбезпечніші криптовалютний гаманець.

Недоліки:

- Треба купляти.

Паперові гаманці це шматок бумаги яка містить приватний ключ, публічний ключ та QR код. У QR код закодована ключова пара, що дозволяє використовувати її для будь яких транзакції.

Переваги:

- Не потребує підключення до інтернету чи комп'ютера

Недоліки:

- Для проведення транзакції треба більше часу.

Мобільні гаманці одні з самих зручних, вони дозволяють користуватися криптовалютою з любого місця, де є доступ до інтернету.

Переваги:

- Зручні та прості у використанні криптовалютний гаманець.
- Мають змогу зчитувати QR коди.

Недоліки:

- Втрата телефона означає втрату гаманця.
- Дуже вразливі до шкідливого ПО.

Комп’ютерні криптовалютні гаманці безпечніші за мобільні.

Переваги:

- Надають повний контроль над приватними ключами.

Недоліки:

- Використовують інтернет.

Веб гаманці найзручніші у використанні, але самі небезпечні.

Переваги:

- Найпростіші в використанні.
- Є автоматизація транзакції.

Недоліки:

- Найнебезпечніший криптовалютний гаманець.

Мультипідписні гаманці потребують кілька приватних ключів, кількість залежить від рівня безпеки, для доступу к криптовалюті та підписування транзакції. Вони дозволяють створювати гаманці для сумісного використання.

Ієрархічно детерміновані (Hierarchical deterministic або HD) гаманці використовують запропоновану в BIP32 модель деревоподібної структури та розширена в BIP44 стандарті. В BIP32 стандарті описано як для гаманців, які підтримують кілька криптовалют, генерувати з однієї пари ключів всі інші.

1.3 Огляд та аналіз існуючих рішень

Сьогодні існує дуже багато різноманітних реалізації криптовалютних гаманців, вони відрізняються функціональністю, кількістю підтримуваних коїнів та рівним безпеки.

Electrum – це криптовалютний гаманець націлений на роботу тільки з BitCoin. Він був створений ще у 2011 році [8] та сумісний з Windows, Linux, OSX, а у 2016 році була додана сумісність з Android. Він дозволяє створювати та

використовувати гаманець з мультипідписом, також користувач має можливість встановити двофакторну аутентифікацію. Electrum має багато серверів та вони децентралізовані, що дозволяє йому працювати без затримки. Підтримка інтеграції всі відомих апаратних гаманців дозволяє зберігати приватний ключ в більш безпечному місці. А можливість використовувати TOR дає можливість підвищити конфіденційність користувача. На рис. 1.3 зображено інтерфейс Electrum.

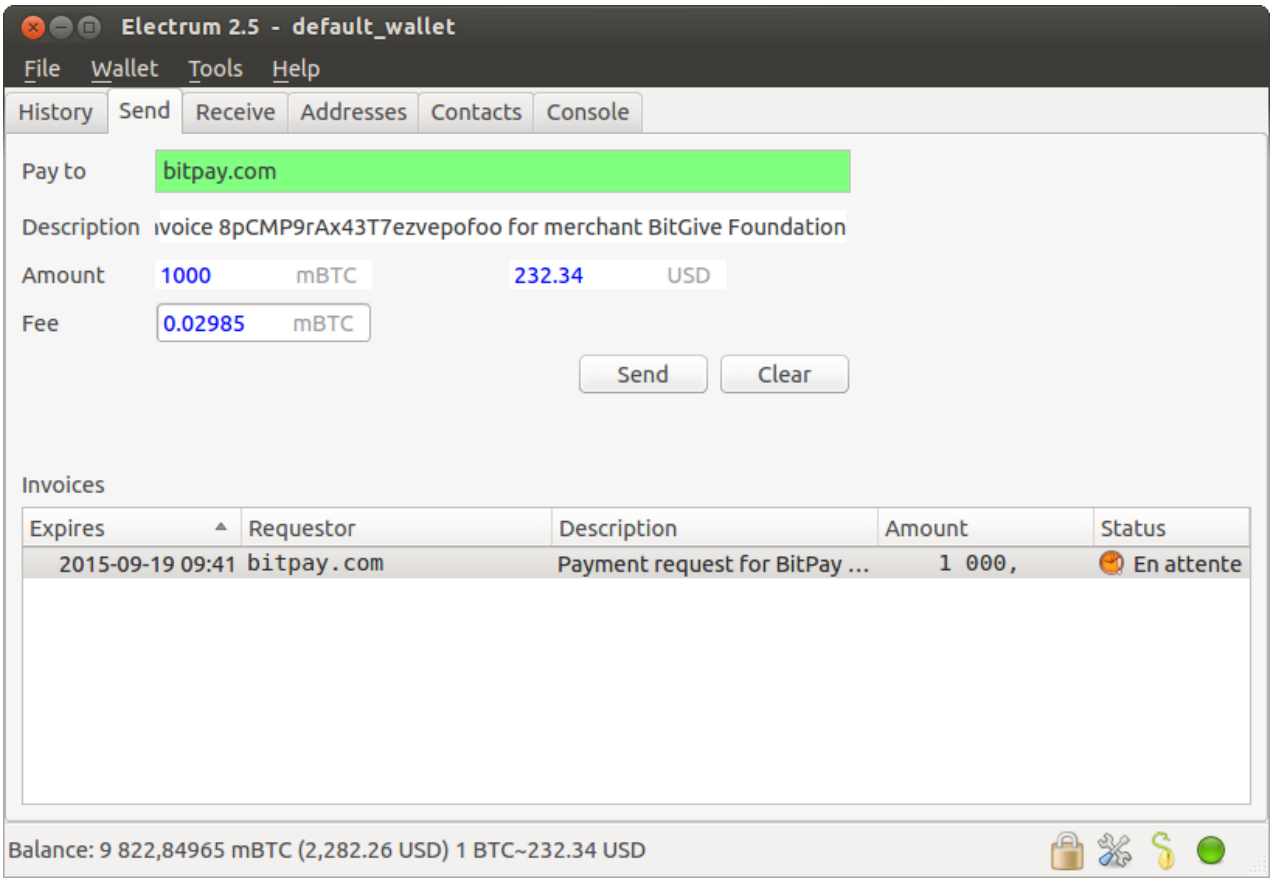


Рисунок 1.3 – Інтерфейс Electrum

MetaMask у 2016 році випущений як розширення для браузера, виключно для Chrome та Firefox, а з 2020 року став доступний для використання на iOS та Android пристроях. Цей гаманець з коїнів підтримує тільки Ethereum, а з токенів всі що були побудовані за ERC20 та ERC721 стандартом [9]. Можливість змінювати мережі дозволяє використовувати всі блокчейни зроблені на основі Ethereum та тестувати децентралізовані додатки. Також MetaMask підтримує

інтеграцію апаратних гаманців, але тільки марки Trezor та Ledger. Підтримка більшістю децентралізованими додатками робить його зручним у використанні через що він став дуже популярним. Також він має вбудований обмінник який шукає найкращу ціну обміну. На рис. 1.4 зображено інтерфейс MetaMask.

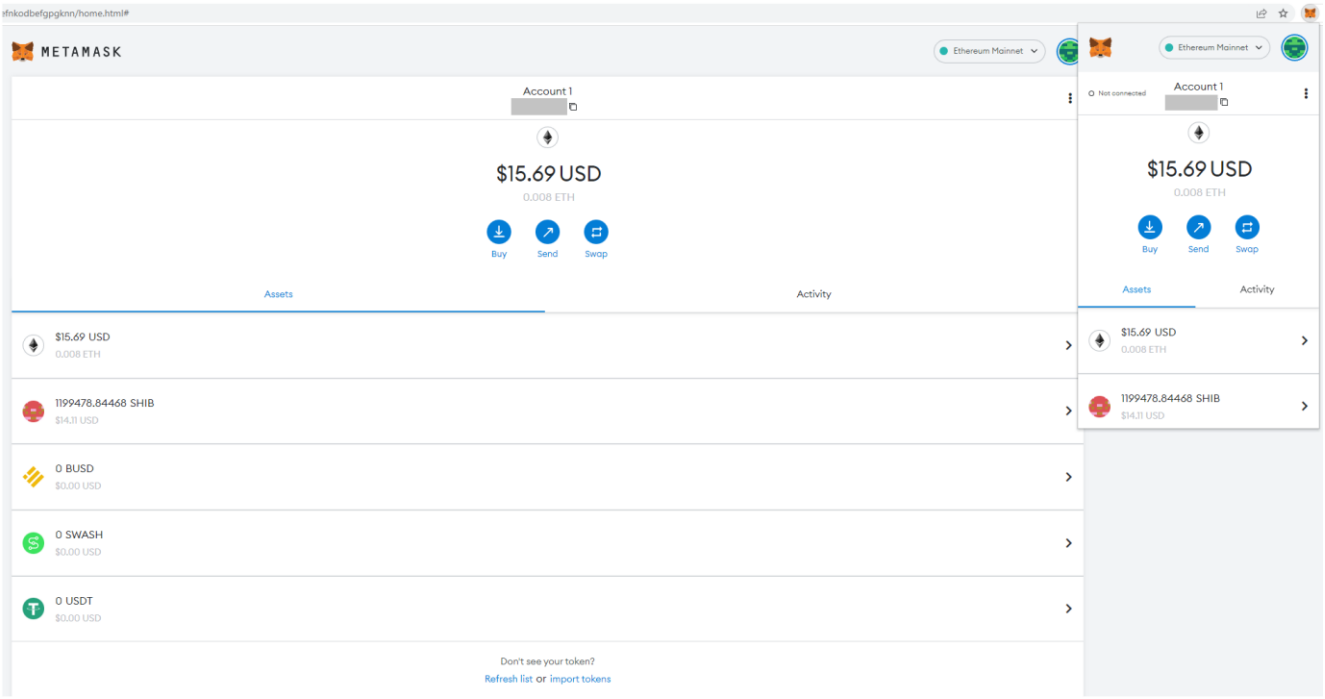


Рисунок 1.4 - Інтерфейс MetaMask

Зараз один з найпопулярніших комп'ютерних гаманців є Exodus. Він з'явився у 2016 році та на сьогоднішній день сумісний з Windows, MacOS, iOS, Linux, Android [10]. Він підтримує інтеграцію апаратних гаманців марки Trezor. На даний момент здатний працювати з більш ніж 150-тю коїнами та будь-якими ERC20 токенами, але треба дивитися які коїни підтримуються платформою, на якій використовується гаманець, та також немає можливості змінювати мережі. В Exodus вбудований ShapeShift обмінник, що дає змогу обмінювати криптовалюту не виходячи з додатку. Також він має додатки які можна встановити з гаманця, які дають користувачеві можливість мати пасивний заробіток:

- Earn Crypto Rewards. Він дозволяє стакувати криптовалюту безпосередньо з гаманця.

- Compound Finance. Він використовує протокол, з такою ж назвою, побудованим на Ethereum та дозволяє позичати свої DAI токени, та кожний раз коли добивається блок Ethereum користувач отримує процент. Всі позичені токени користувач може повернути в любий час.

Ще можна встановити FTX біржу, та торгувати безпосередньо з Exodus гаманця. На рис. 1.5 зображено інтерфейс Exodus.

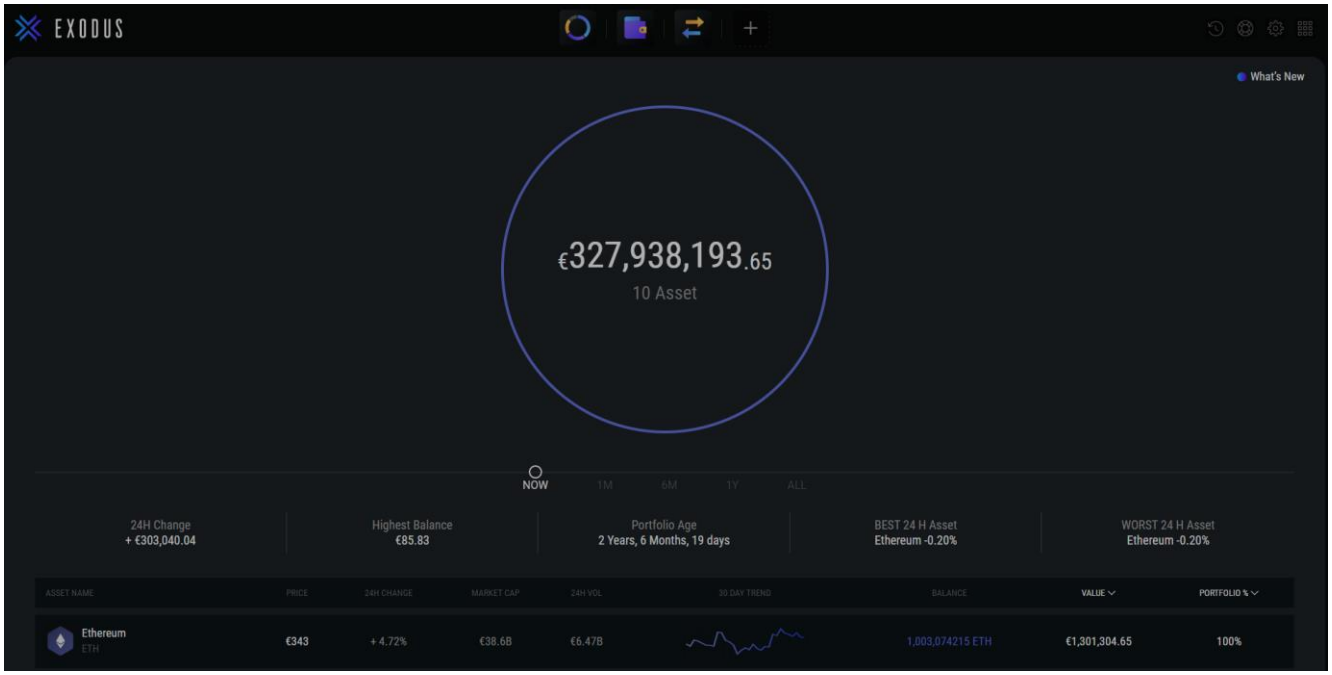


Рисунок 1.5 – Інтерфейс Exodus

У 2017 році був випущений Trust Wallet – це криптовалютний гаманець який доступний на Android та iOS. Він підтримує 60 коїнів та всі ERC20, ERC721, BEP2 токени, але немає можливості змінювати мережі [11]. Також має вбудовану купівлю великої кількості криптовалют, на даний час Trust Wallet підтримує 6 платіжних мереж. Вбудований обмінник криптовалют, який використовує 1inch Network, дозволяє обмінювати як ERC20, BEP2 токени та Polygon активи. Trust Wallet має можливість стакувати криптовалюту, що дозволяє користувачеві мати пасивний заробіток за допомогою своїх активів. Також має вбудований браузер децентралізованих додатків. На рис. 1.6 зображено інтерфейс Trust Wallet.

Trust Wallet має своє ядро, що дозволяє користувачам бути впевненим за те що їх дані будуть захищені від атаки на ланцюги поставок (Supply chain attacks) – це кібератака як використовує вразливості в бібліотеках які використовуються для розробки додатку, чим більше таких залежностей тим більша загроза. Сучасні криптовалютні гаманці використовують багато різноманітних залежностей які взаємодіють з криптографічними бібліотеками, що надає їм доступ до важливої інформації. Ця велика кількість залежностей також мають свої залежності, що робить гаманці дуже вразливими до такого типу атак [12].

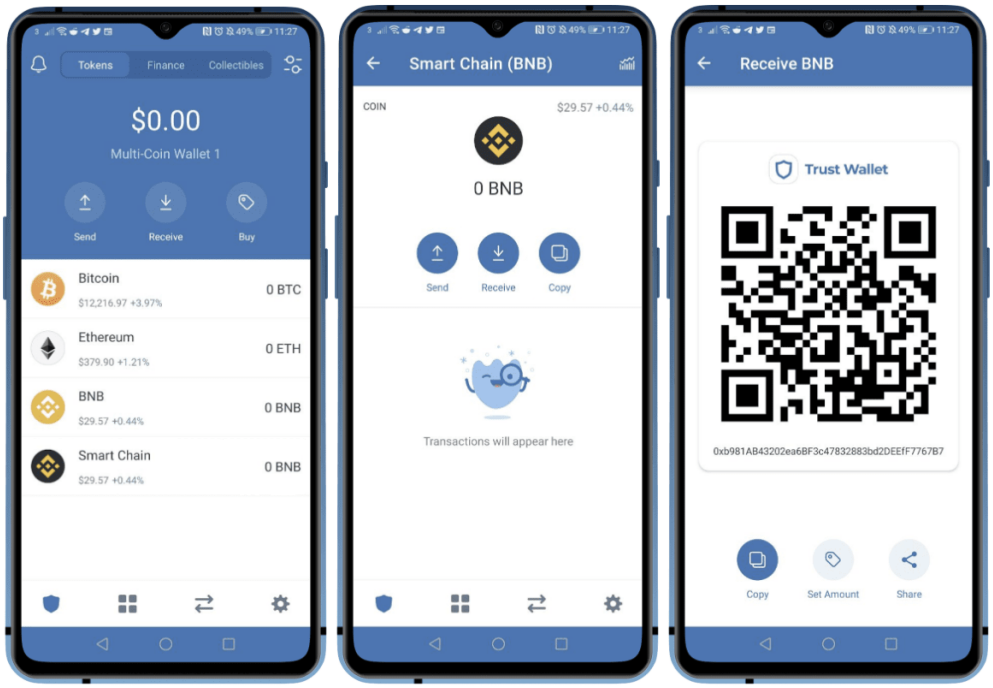


Рисунок 1.5 – Інтерфейс Trust Wallet

Всі кличеві переваги та недоліки розглянутих аналогів зведені в таблиці 1.1.

Таблиця 1.1 - Недоліки та переваги аналогів

Назва	Переваги	Недоліки
Electrum	1. Підтримка мультипідпису. 2. Має двофакторну аутентифікацію.	1. Підтримує тільки BitCoin.

Продовження таблиці 1.1 - Недоліки та переваги аналогів

MetaMask	1. Простий у використанні. 2. Дозволяє додавати мережі.	1. Підтримує тільки Ethereum
Exodus	1. Підтримує більше 150 коїнів. 2. Підтримує Windows, Android, iOS, MacOS, Linux.	1. Висока комісія за транзакцію 2. Нема можливості додавати мережі.
Trust Wallet	1. Має власне ядро. 2. Підтримує 60 коїнів. 3. Підтримує Android, iOS.	1. Нема можливості додавати мережі.

1.4 Постановка технічного завдання

Об’єктом дослідження є процес контролю криптовалютних гаманців.

Предметом дослідження є програмне забезпечення для контролю криптовалютних гаманців, що забезпечують роботу з кількома DLT мережами.

Метою роботи є спрощення процесу контролю кросплатформних криптовалютних гаманців та підвищення їх рівня безпеки.

Для досягнення поставленої мети необхідно виконати наступні задачі:

1) проаналізувати поняття криптовалют, технологій розподіленого реєстру, криптовалютних гаманців;
2) порівняти існуючих криптовалютних гаманців, виділити їх основні переваги та недоліки;
3) обрати технологій і середовища розробки;
4) обґрунтувати обрання технологій і середовища розробки;
5) розробити варіанти використання програмного продукту;
6) побудувати структурну модель використання;
7) проаналізувати архітектурні шаблони;
8) змоделювати програмне забезпечення;

9) розробити програмне забезпечення;

10) провести тестування програмного забезпечення.

Після аналізу процесу контролю криптовалютних гаманців та огляд існуючих рішень можна сформулювати наступні вимоги до розроблювальної бібліотеки:

- бібліотека повинна основуватися на ієрархічної детермінованої моделі(BIP32 та BIP44);
- бібліотека повинна виконувати всі основні функції криптовалютного гаманця (генерування ключової пари, створення транзакції, підписування транзакції);
- бібліотека повинна забезпечувати просте додовання підтримки нової DLT мережі;
- бібліотека повинна бути портована на Android.

2 ЗАСОБИ ТА ТЕХНОЛОГІЇ РОЗРОБКИ

2.1 Вибір мови програмування

При розробці кросплатформних додатків використовують один з наступних двох основних підходів [13]:

- використовувати готові інструментарій такі як Qt, React Native, Electron та інші;
- створювати бібліотеку з бізнес логікою та портувати її на інші платформи.

Для розробки було обрано другий підхід тому, що він дозволяє використовувати рідне API платформу та надає більший рівень безпеки ніж перший.

Тому як основну мову програмування було обрано C++ тому, що вона в тому чи іншому вигляді підтримується великою кількістю платформ так, як від початку створювався як розширення мови C. Також його можна назвати рідною мовою Windows та Linux.

На даний час C++ має достатньо великий функціонал доповнюється новітніми стандартами, до C++17 [14] нічого не видалялося, що достатньо довго надавало гарну сумісність з кодом який був написаний з використанням старого стандарту.

Головною відмінністю від C – це наявність класів, в першій версій мови багато функціоналу було орієнтовано на роботу з ними. У 1992 році у мову було додану невід’ємну його частину, а сам стандартну бібліотеку шаблонів (STL) яка на даний час містить велику частину мови [14].

Першій стандарт ISO/IEC 14882:1998 (C++98) доповнював як функціонал самої мови та і STL, достатньо довго не виходило нових стандартів, але в 2003 році вийшов ISO/IEC 14882:2003 (C++03) яка виправляла велику кількість дефектів які були виявлені в попередньому стандарті [14]. Після цього достатньо

регулярно виходять нові стандарт, на даний час останній стандарт який вийшов та використовується – це ISO/IEC 14882:2020 (C++20).

Для розробки додатків під Android використовують мову програмування Kotlin створену JetBrains у 2011 році. Вона повністю сумісні з мовою Java ти JVM (Java virtual machine) та є кроссплатформною. У 2017 році на Google I/O, Google анонсував повну підтримку Kotlin, також в цей рок Kotlin був доданий у Android Studio як альтернативний компілятор, а вже у 2019 року Google назвала це найкращим вибором для розробки додатків на Android [15].

Kotlin – це функціональна мова, синтаксис якої дуже схожий до Java через що його так швидко стало використовувати достатньо велика кількість розробників. Але це не суцільна заміна Java, його можна назвати великим розширенням Java. На рис. 2.1 представлено приклад синтаксису Kotlin.

```kotlin
import photomover.cli.web.StartWebOptions

fun main(arguments: Array<String>) {
  val commands = mapOf(
      "organize" to wrapOperation(::organize, OrganizeOptions()),
      "upload" to wrapOperation(::upload, UploadOptions()),
      "startWeb" to wrapOperation(::start, StartWebOptions()))
  val args = if (arguments.size == 0) {
    array("startWeb")
  } else {
    arguments
  }
  val command = commands[args[0]]
  if (command == null) {
    println("Command '${args[0]}' is not supported.")
    println("Please specify one of following: ${commands.keySet()}")
  } else {
    println("Starting ${args[0]}")
    command(args.drop(1))
  }
}

fun wrapOperation<T>(
    operation: (options: T) -> Unit, defaultOptions: T): (args: List<String>) -> Unit {
  return { args ->
    val parser = CmdLineParser(defaultOptions)
    try {
      parser.parseArgument(args)
    } catch (ex: CmdLineException) {
      println(ex.getMessage())
      println(parser.printUsage(System.err))
    }
    println("args: $defaultOptions")
    operation(defaultOptions)
  }
}
```

Рисунок 2.1 - Приклад синтаксису Kotlin

Сумісність з JVM дозволяє використовувати JNI (Java Native Interface) який надає можливість співпрацювати з C/C++. Для того щоб викликати C++ код з Java або Kotlin цей код треба «обгорнути» в C.

2.2 Вибір програмних засобів

Основним критерієм при виборі середи розробки, окрім підтримку потрібних мов, була популярність у розробників для відповідної платформи, тому як основним інтегрованим середовищем розробки була обрана Visual Studio. Перша версія була випущена у 1997 році та на даний час останній реліз був у 2022 році. Вона підтримує достатньо виклику кількість мов програмування, а саме C, C++, C#, F#, Python, Ruby та ще 30 інших [16].

Visual Studio представлена в 3 виданнях, які відрізняються функціоналом та ціною [16]:

- Community – це безплатна версія Visual Studio, вона має той самий функціонал що і Professional, окрім урізаного CodeLens. Дозволяє створювати власні безкоштовні або платні додатки, а для компанії має обмеження, в організації повинно бути менше 250 комп'ютерів на яких вона встановлена та річний дохід повинен бути менше 1 мільйона доларів США.
- Professional, платна версія, має повний CodeLens та відсутні обмеження для компанії. Також дозволяє користуватися базовим планом Azure DevOps.
- Enterprise має більш розширений функціонал у порівняні з попередніми версіями та к базовому плану Azure DevOps дається план тестування.

Visual Studio має дуже великий та різноманітний функціонал, який дозволяє не тільки писати та займатися дебагом коду, а також має інструментарій який дозволяє займатися рефакторингом, займатися дизайном класів, даних, інтерфейсу та багато інших. В ній вбудований маркет з розширеннями. На рис. 2.2 зображено інтерфейс Visual Studio 2022.

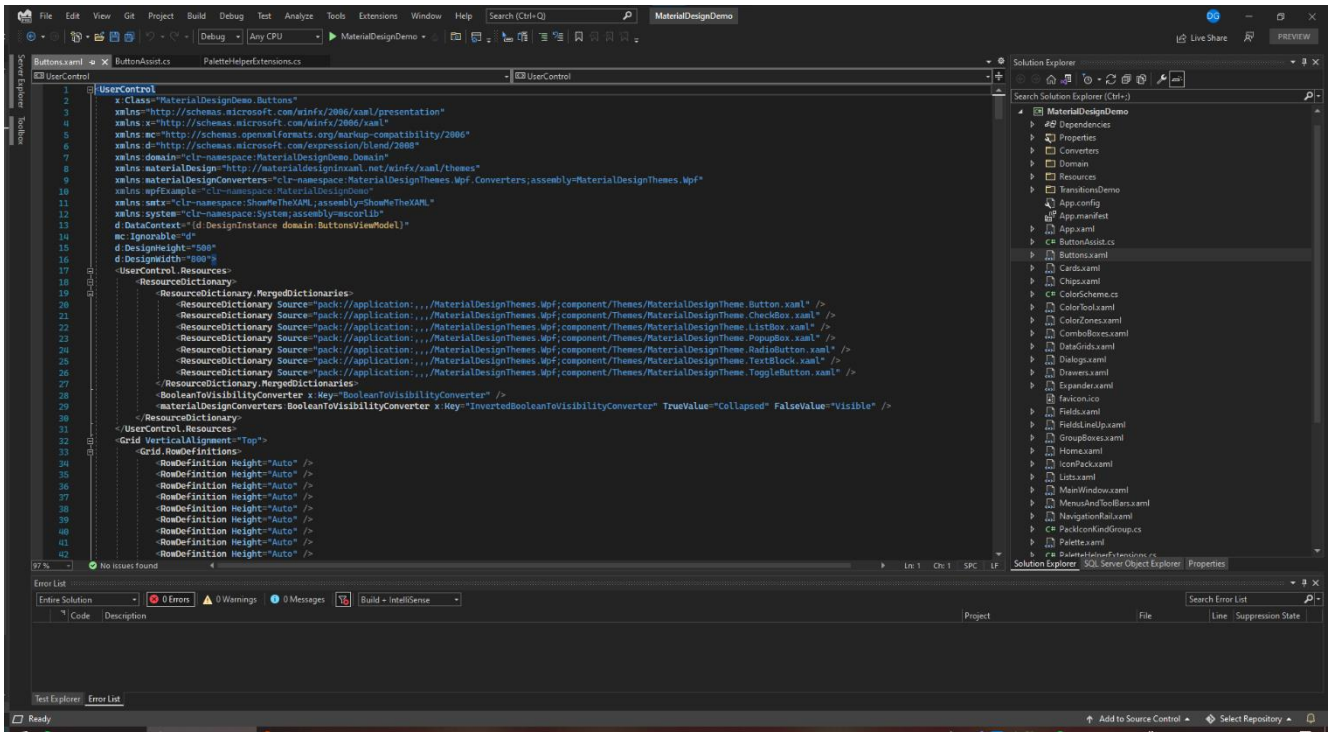


Рисунок 2.2 - Інтерфейс Visual Studio 2022

Використовуючи такий же критерії було обрано Android Studio – це інтегрованим середовищем розробки створена Google на основі IntelliJ IDEA, підтримує Windows, MacOS, Linux [17]. Android Studio безкоштовна для індивідуального та для корпоративного використання.

Вона надає інструментарії для розробки додатків під Android, як смартфонів та планшетів, так і Android TV, Android Wear, Android Auto, використовуючи Kotlin, Java, C++. В ній надано UI макети, що прискорює початок розробки. Окрім описаних раніше також в Android Studio має наступні особливості:

- Редактор графічного інтерфейсу за допомогою Drag-and-Drop.
- Система збірки основана на Gradle.
- Вбудований швидкий емулятор, який окрім смартфонів дозволяє емулювати смарт годинники.
- Інтегрований GitHub.
- Вбудовані інструментарій та фреймворки для тестування.
- Вбудований аналізатор коду Lint.

- Дозволяє вносити зміни в коді додатку який працює без потреби його перезапуску.

На рис. 2.3 зображено інтерфейс Android Studio.

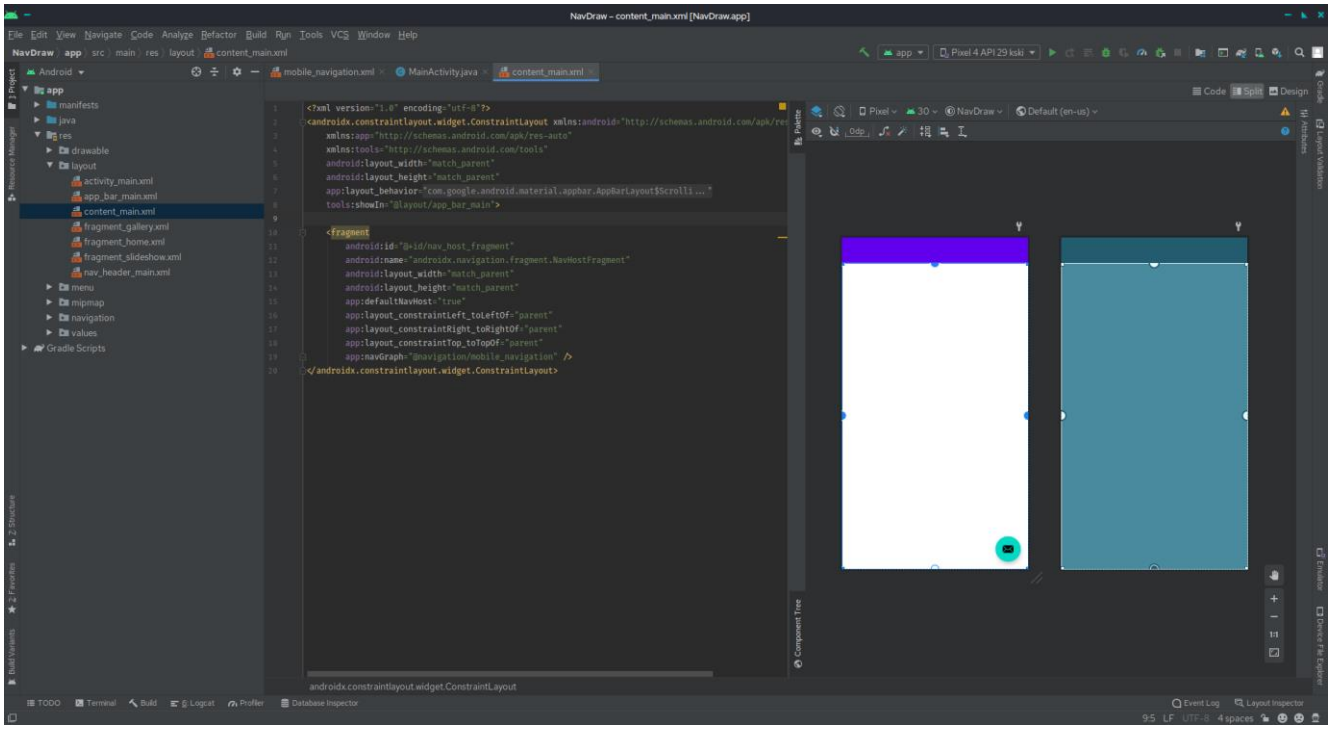


Рисунок 2.3 - Інтерфейс Android Studio

Для забезпечення в бібліотеці підтримку більшості криптографічних алгоритмів використовується бібліотека Trezor-crypto, її створили та підтримує компанія Trezor. Ця компанія розробляє та продають апаратні криптовалютні гаманці з 2012 року, на даний час в продажі представлені такі їх гаманці Trezor Model T та Trezor Model One. Вони мають репутацію як одні з найбезпечніших гаманців.

Для збірки проекту було обрано CMake – це утиліта для автоматизації збірки додатків, статичних та динамічних бібліотек з програмного коду [18]. Вона не є системою збірки вона генерує файли для інших систем збірки, для таких як Makefile, Ninja, також вона може генерувати файли для створення проекту в Visual Studio, Xcode та для інших IDE [18]. CMake була створений у 2000 році та

на даний час остання її версія яка доступна на офіційному сайті для Windows, MacOS, Linux – це 3.23.3.

Вона підтримує кілька мов програмування, а саме C++, C#, Cuda та інші. Для збірки проекту, CMake має власну скриптову мову, яка підтримує змінні, методи маніпулювання рядками, масиви, оголошення функцій і включення модулів. За допомогою цієї мови вказуються шлях до програмних файлів та налаштування збірки в спеціалізованому файлі з назвою CMakeLists.txt [18], який знаходиться у кожному проекті. На рис. 2.4 зображений приклад наповнення CMakeLists.txt.

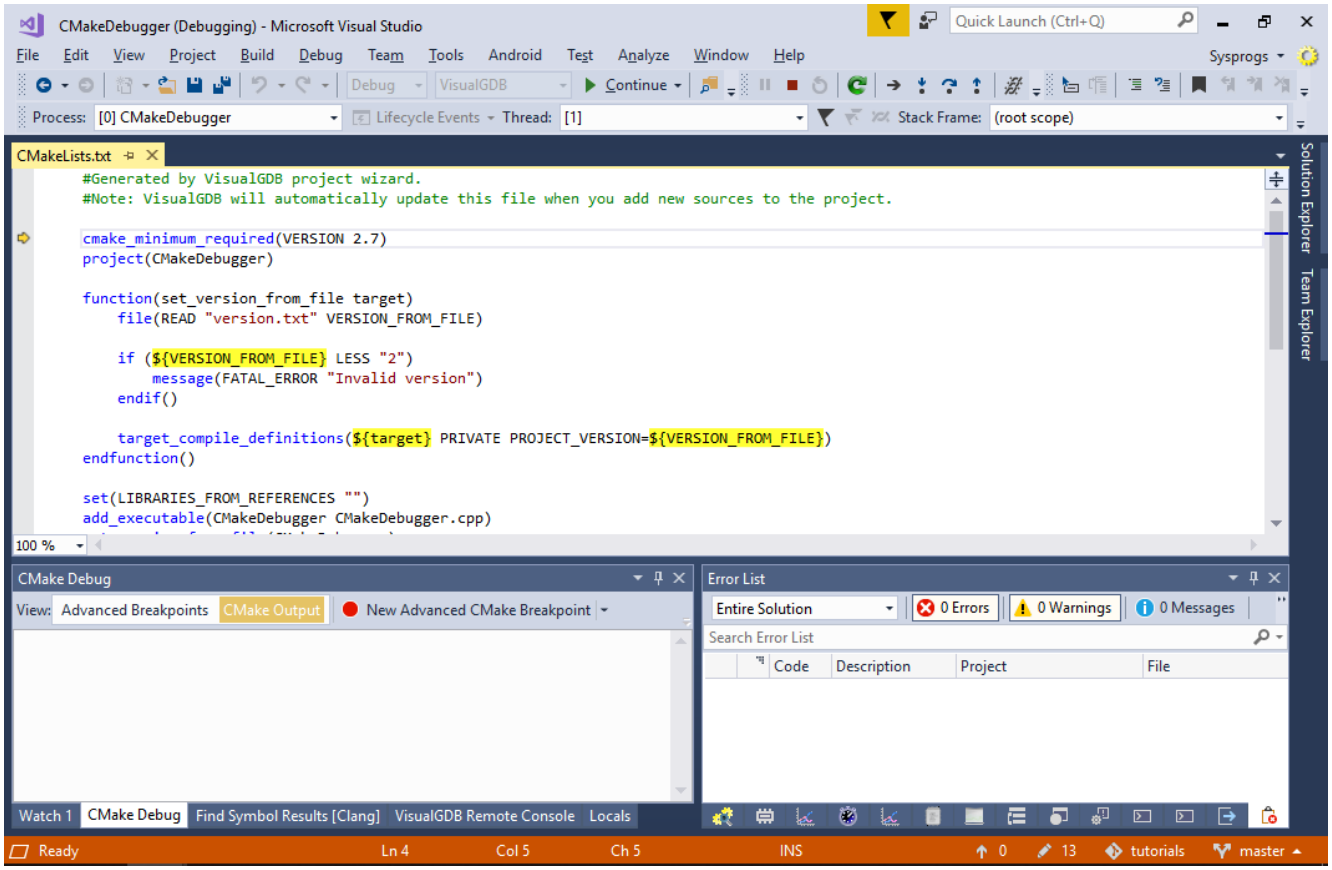


Рисунок 2.4 - Приклад наповнення CMakeLists.txt

CMake генерує кеш файл які можна редагувати в графічному редакторі. Це файли які створюються після роботи CMake, в ньому зберігається всі включені файли, виконувані файли та бібліотеки, а також можуть бути додані необов’язкові файли, а кеш дозволяє відредагувати цей список та згенерувати нові файли збірки.

2.3 Розробка варіантів використання програмного продукту

Бібліотеку буде використовувати тільки користувач який взаємодіє з нею через клієнт – це додаток, який відображає інформацію у зручному форматі користувачеві, та дає можливість контролювати створені криптовалютні гаманці та створювати нові. Це може бути мобільний або комп’ютерний додаток.

Варіанти використання які передбачені у бібліотеки представлені в таблицях 2.1-2.6

Таблиця 2.1 – Варіант використання UC001

Назва	Створення гаманця.
Опис	Користувач може створювати нові гаманці.
Учасники	Клієнт, користувач.
Передумови	
Постумови	Для клієнта згенерована основну ключова пара та похідні ключові пари для кожної підтримуваної криптовалютиє.
Основний сценарій	- Клієнт використовує функції генерування мнемонічної фрази. - Бібліотека генерує мнемонічну фразу. - Бібліотека повертає мнемонічну фразу клієнту. - Клієнт надає можливість користувача ввести додаткове слово. - Користувач вводить додаткове слово. - Клієнт використовує функцію створення гаманця використовуючи згенеровану мнемонічну фразу та додаткове слово. - Бібліотека генерує основну ключову пару та похідні. - Бібліотека повертає сутність гаманця до клієнту.

Продовження таблиці 2.1 – Варіант використання UC001

Розширення сценаріїв	Якщо користувач не вводить додаткове слово то гаманець створюється тільки з мнемонічної фрази.

Таблиця 2.2 – Варіант використання UC002

Назва	Підписування транзакції.
Опис	Користувач може створювати та підписувати транзакції.
Учасники	Клієнт, користувач.
Передумови	В клієнті вже було створено гаманець.
Постумови	Бібліотека повертає клієнту підписану транзакцію.
Основний сценарій	- Користувач вводить необхідні дані для транзакції. - Клієнт використовує функцію створення транзакції. - Бібліотека валідує адреси. - Бібліотека створює транзакцію та повертає. - Клієнт використовує функцію для підписання транзакції. - Клієнт отримає підписану транзакцію.
Розширення сценаріїв	Клієнт може підписати транзакцію яка була створена не функцією бібліотеки.

Таблиця 2.3 – Варіант використання UC003

Назва	Додавання мережі.
Опис	Користувач може додавати мережі через які клієнт буде взаємодіяти з розподіленими реєстрами.
Учасники	Клієнт, користувач.
Передумови	В клієнті вже було створено гаманець.
Постумови	Бібліотека має базу DLT мереж які будуть використовуватися для поширення транзакції.
Основний сценарій	- Користувач вводить дані мережі.

Продовження таблиці 2.3 – Варіант використання UC003

Основний сценарій	- Клієнт використовує функцію додавання мережі. - Бібліотека додає нову мережу до списку.
Розширення сценаріїв	

Таблиця 2.4 – Варіант використання UC004

Назва	Додавання адреси.
Опис	Користувач може додати адресу в вже існуючому гаманці.
Учасники	Клієнт, користувач.
Передумови	В клієнті вже було створено гаманець.
Постумови	В клієнті додано нову адресу.
Основний сценарій	- Клієнт використовує функцію додавання адреси. - Бібліотека генерую похідну ключову пару. - Бібліотека генерує адресу з створеної похідної ключової пари.
Розширення сценаріїв	

Таблиця 2.5 – Варіант використання UC005

Назва	Отримання адреси.
Опис	Користувач може подивитися публічний ключ певного аккаунта певного коїну.
Учасники	Клієнт, користувач.
Передумови	В клієнті вже було створено гаманець.
Постумови	Користувач отримає адресу.
Основний сценарій	- Користувач обирає коїн та аккаунт. - Клієнт використовує функцію отримання публічного ключа з обраним шляхом - Бібліотека повертає клієнту публічний ключ

Продовження таблиці 2.5 – Варіант використання UC005

Основний сценарій	відповідно до шляху. - Клієнт виводить адресу.
Розширення сценаріїв	Якщо в функцію передано шлях тільки до типу коїну повертається адресу для першого акаунта.

Таблиця 2.6 – Варіант використання UC006

Назва	Отримання мнемонічної фрази.
Опис	Користувач може подивитися мнемонічну фразу.
Учасники	Клієнт, користувач.
Передумови	В клієнті вже було створено гаманець.
Постумови	Користувач отримає мнемонічної фрази.
Основний сценарій	- Клієнт виконує функцію отримання мнемонічної фрази. - Бібліотека повертає клієнту мнемонічну фразу. - Клієнт виводить мнемонічну фразу.
Розширення сценаріїв	

Проаналізовавши створені основні варіанти використання було побудована структурну схему використання, вона зображена на рис. 2.5. Ця схема буде взята як основа для розробки архітектури бібліотеки.

Аналіз варіантів використання передбачає подальший аналіз та створення специфікації для функціонального та нефункціональних вимог. Такий поділ базується на їх відмінностях в природі.

Основні функціональних вимог – це перелік функціоналу, який має бути присутній в додатку, обмеження даних та опис його поводження при їх виконанні.

Після аналізу вимог до системи контролю криптовалютних гаманців були утворені наступні функціональні вимоги до бібліотеки криптовалютного гаманця та представлено в таблицях 2.7-2.13.

Рисунок 2.5 - Діаграма варіантів використання

Таблиця 2.7 – Опис функціональної вимоги REQ001

Номер	REQ001
Назва	Створення гаманця
Опис	Після створення гаманця, бібліотека повертає клієнту його сутність та він має зберегти основний приватний ключ на пристрій.

Таблиця 2.8 – Опис функціональної вимоги REQ002

Номер	REQ002
Назва	Підписування транзакції.
Опис	Бібліотека повертає підписану транзакцію, у вигляді строки байтів, клієнту яку він повинен поширити з DLT мережею.

Таблиця 2.9 – Опис функціональної вимоги REQ003

Номер	REQ003
Назва	Додавання мережі.

Продовження таблиці 2.9 – Опис функціональної вимоги REQ003

Опис	Бібліотека поповнює список доступних мереж, вхідними дані повинні бути у вигляді строки.

Таблиця 2.10 – Опис функціональної вимоги REQ004

Номер	REQ004
Назва	Додавання адреси.
Опис	Бібліотека додає нову адресу до списку адрес відповідного коїну.

Таблиця 2.12 – Опис функціональної вимоги REQ005

Номер	REQ005
Назва	Отримання публічного ключа.
Опис	Бібліотека повертає публічний ключ клієнту, у вигляді строки байтів.

Таблиця 2.13 – Опис функціональної вимоги REQ006

Номер	REQ006
Назва	Отримання мнемонічної фрази.
Опис	Бібліотека повертає мнемонічну фразу клієнту, у вигляді строки.

Нефункціональні вимоги – це вимоги які вказують якою система повинна бути та задають критерії для оцінки якості.

Для отримання повного списку вимог були сформуванні нефункціональні вимоги та представлені в таблицях 2.14-2.18.

Таблиця 2.14 – Опис нефункціональних вимоги NFR001

Номер	NFR001

Продовження таблиці 2.14 – Опис нефункціональних вимоги NFR001

Назва	Підтримка мови програмування C++
Опис	Бібліотека повинна мати підтримку мови програмування C++.

Таблиця 2.15 – Опис нефункціональних вимоги NFR002

Номер	NFR002
Назва	Підтримка мови програмування Kotlin
Опис	Бібліотека повинна мати підтримку мови програмування Kotlin.

Таблиця 2.16 – Опис нефункціональних вимоги NFR003

Номер	NFR003
Назва	Підтримка кілька DLT мереж
Опис	Бібліотека повинна мат можливість підтримувати кілька DLT мереж.

Таблиця 2.17 – Опис нефункціональних вимоги NFR004

Номер	NFR004
Назва	Переносимість
Опис	Бібліотека повинна працювати без змін при використанні інших середовищ розробки.

Таблиця 2.18 – Опис нефункціональних вимоги NFR005

Номер	NFR005
Назва	Швидкодія
Опис	Бібліотека повинна мати малий час виконання функції,

2.4 Аналіз архітектурних рішень

Існує багато різноманітних архітектурних шаблонів, але для взаємодії між бібліотекою та додатком підійдуть є MVC (Model-View-Controller), MVP (Model-View-Presenter) і MVVM (Model-View-ViewModel). Ці три шаблона достатньо часто використовуються для створення додатків для Android та iOS.

Приклад структури архітектури Model-View-Controller можна побачити на рис. 2.6.

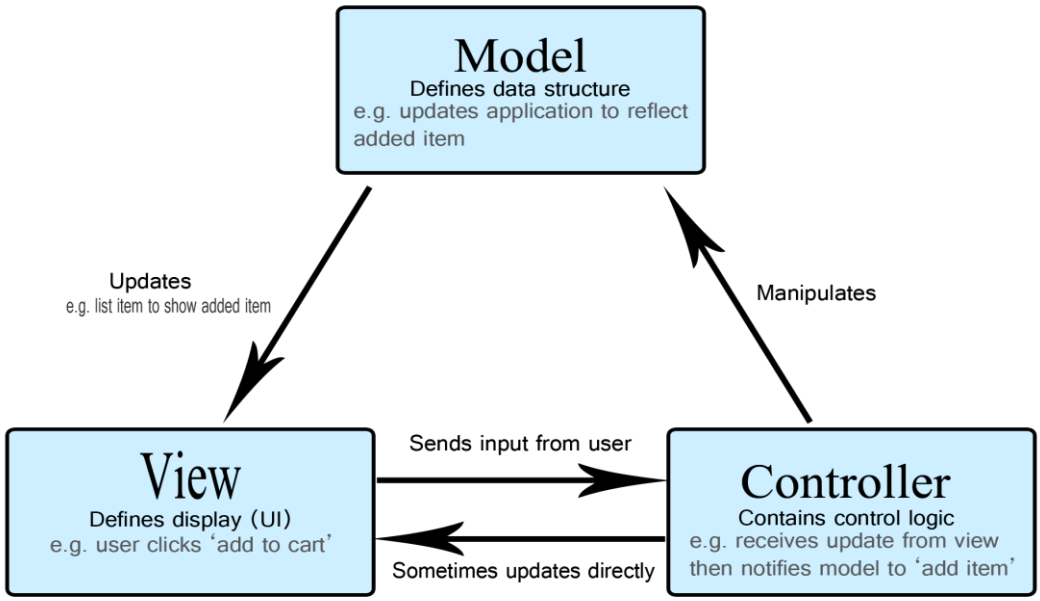


Рисунок 2.6 – Архітектура MVC [19]

MVC описує три основних компонента:

- Модель (Model) визначає які дані необхідні для роботи застосунку, та їх оновлення.
- Представлення (View) визначає як додаток повинен виводити дані, тобто відповідає за графічний інтерфейс.
- Контролер (Controller) відповідає за обновлення модель та надання відповіді на запити користувач.

Архітектура MVC дозволяє багаторазово використовувати розроблені компоненти додатку. Змінювання перегляду не потребує робити зміни у моделі або контролері, це стосується й контролера і моделі. Але безпосередній зв'язок між моделлю та представленням сприяє появі циклу залежностей, що ускладнює використання однієї моделі з різними представленнями.

MVP зберігає переваги MVC та виправляє її проблеми. Модель не так щільно зв’язана з представленням, що дозволяє використовувати модель повторно без обмежень. Відміно від MVC модель тут не тільки зберігає та надає дані, а також оброблює їх, що дозволяє використовувати її окремо. Структури архітектури Model-View-Presenter зображена на рис. 2.7.

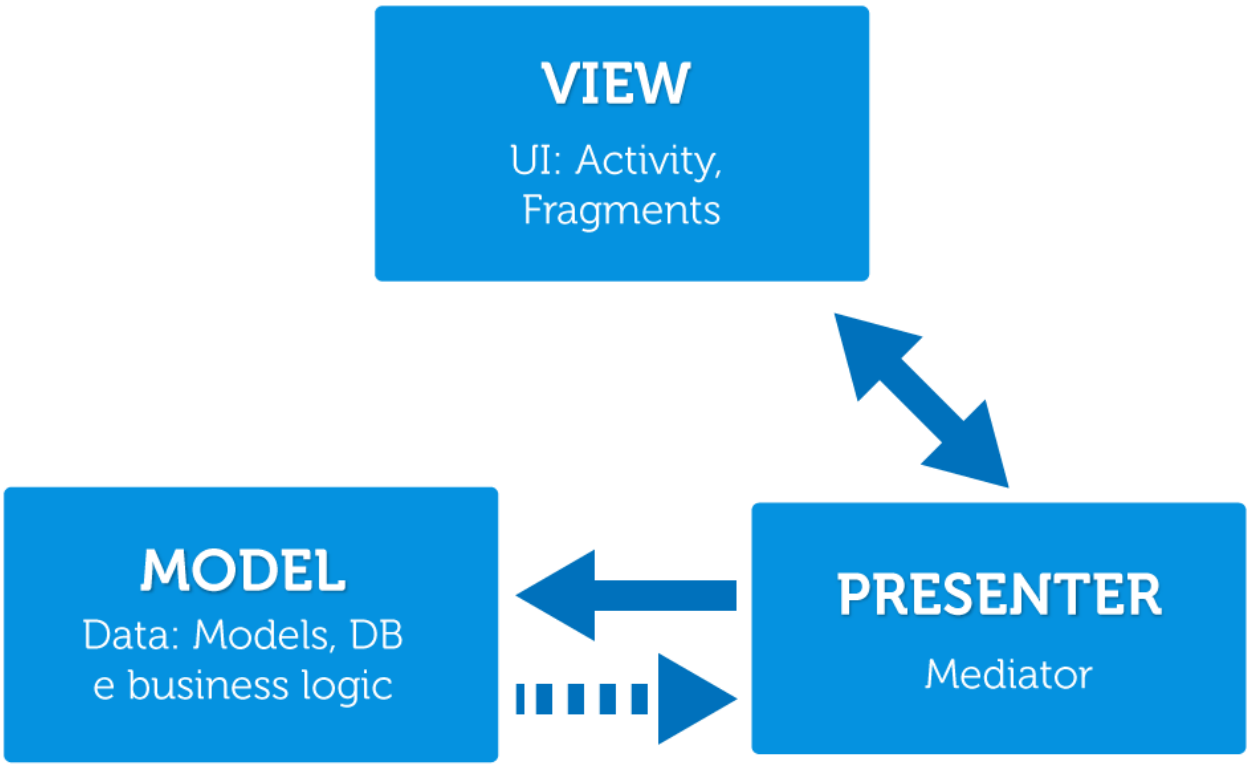


Рисунок 2.7 – Архітектура MVP

Представлення виконує ті самі функції як в MVC. Пред’явник (Presenter) виступає посередником між моделлю та представленням. Він оброблює всі запити з представлення та отримує дані, які необхідні представленню для створення інтерфейсу користувача, з моделі.

Архітектуру MVVM також містить три компонента. Модель повність аналогічна моделі з MVC, містить та оновлює дані які потрібні для роботи додатка. Представлення – це графічний інтерфейс, так саме як в MVC. Модель представлення (ViewModel) зв’язує модель та представлення, але також має логіку для отримання даних з моделі, які потім будуть передані до представлення, та логіку по оновленню даних в моделі. Взаємодія між представленням та моделлю представленням відбувається за допомогою команд. Структури архітектури Model-View-ViewModel зображена на рис. 2.8.

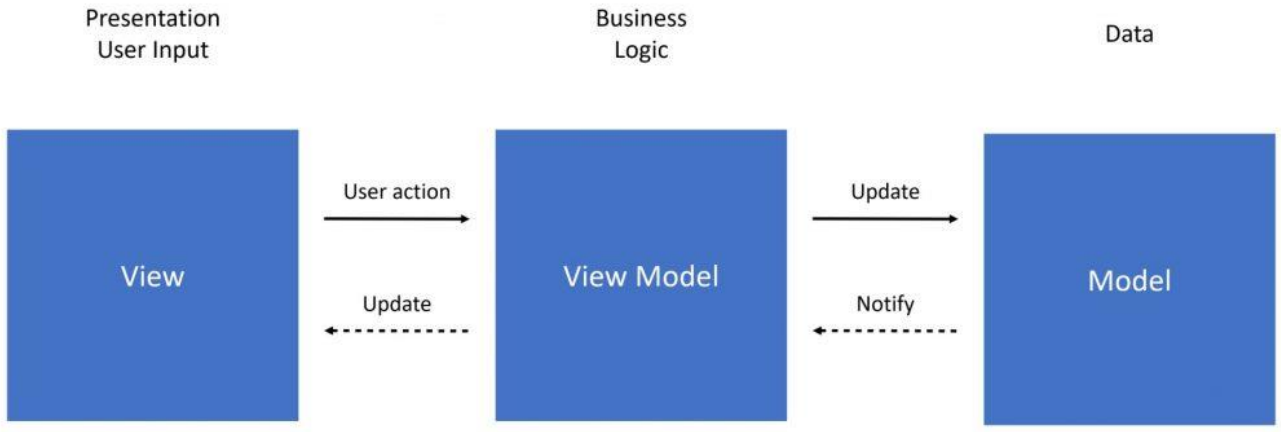


Рисунок 2.8 – Архітектура MVVM

Зведені результати аналізу основних відмінностей архітектурних шаблонів MVC, MVP, MVVM представлені в таблиці 2.19

Таблиця 2.19 – Результати аналізу відмінності між архітектурами

Назва	MVC	MVP	MVVM
Третій компонент	Контролер	Пред’явник	Модель представлення
Компонент відповідальний за ввод даних	Контролер	Представлення	Представлення
Зв'язок між представленням та третім компонентом	Один-до-багатьох	Один-до-одного	Багато-до-одного

Продовження таблиці 2.19 – Результати аналізу відмінності між архітектурами

Взаємодія між представленням та третім компонентом	-	+	+
Взаємодія між представленням та моделлю	+	-	-

Згідно з результатами були сформульовані наступні пропозиції до розробки криптовалютного гаманця з використанням даної бібліотеки для забезпечення максимального рівня безпеки:

- використовувать архітектуру MVVM;
- бібліотека повинна використовуватися тільки в моделі представлення;
- реалізація мережевих модулів повинна знаходиться тільки в моделі представлення;
- всі дані які відправляються до моделі повинні бути серіалізовані.

3 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Моделювання програмного забезпечення

На рис. 3.1 представлено алгоритм створення гаманця.

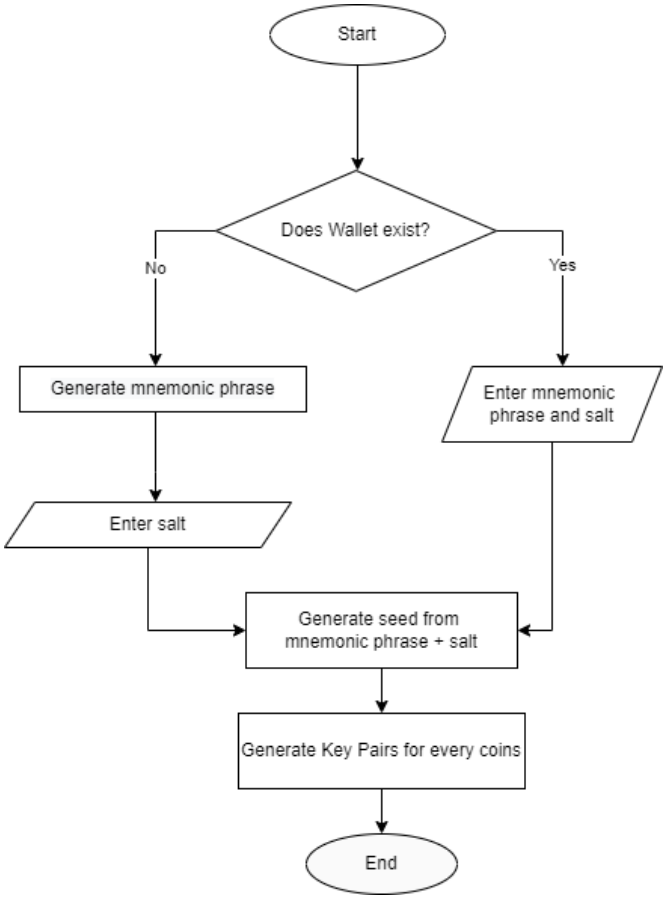


Рисунок 3.1 – Алгоритм створення гаманця

Послідовний опис алгоритму створення гаманця:

1) Якщо криптовалютний гаманець вже існує - то користувачеві потрібно ввести його мнемонічну та додаткову фразу.
2) Інакше система генерує мнемонічну фразу.
3) Користувач вводить додаткову фразу.
4) Система генерує основну ключову пару з мнемонічну та додаткову фрази.
5) Система генерує похідні ключові пари для кожного коїну.

Створення криптовалютного гаманця це перший процес з яким зустрічається користувач, в результаті якого утворюється ключові пари. Мнемонічна фраза допомагає відтворювати вже створені гаманці, її розмір варіюється від 12 до 24 слів, її ідея була описана в BIP39, також там були надані списки слів на 10 мовах. Додаткова фраза, теж описана в BIP39, вона допомагає підвищити унікальність основної ключової пари. Якщо використовувати одну мнемонічну фразу з різними додатковими фразами будуть створюватися різні ключові пари, що означає створення різних гаманців.

На рис. 3.2 представлено алгоритм отримання адреси.

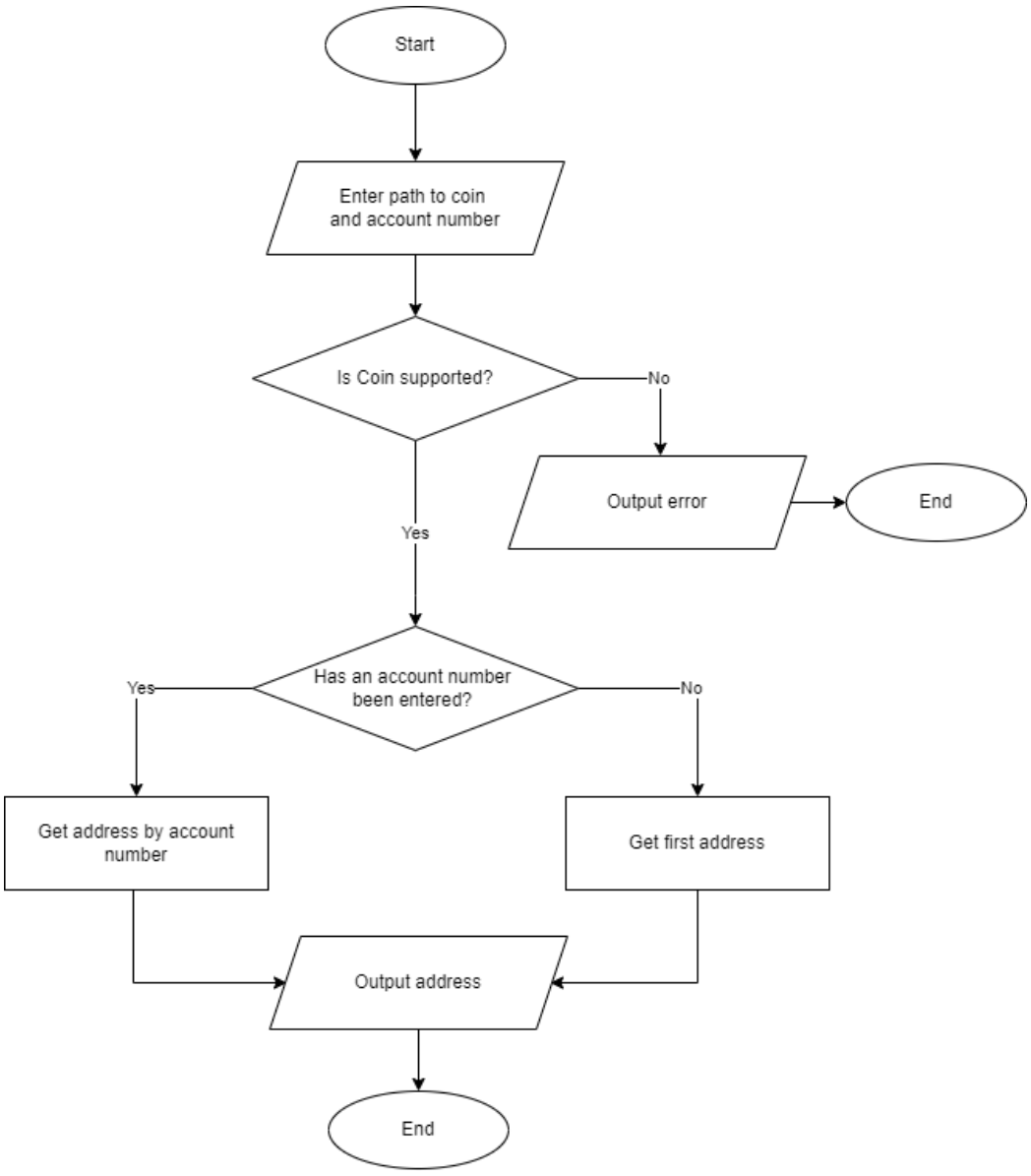


Рисунок 3.2 – Алгоритм отримання адреси

Послідовний опис алгоритму отримання адреси:

1) Введення шляху до коїну та номер аккаунта.
2) Якщо коїн не підтримується системою, то повертається помилка та закінчується процес отримання адреси.
3) Якщо був введений номер акаунта, то одержується адреса за номером акаунта.
4) Інакше одержується адреса першого акаунта.
5) Виводиться отримана адреса.

Для отримання адреси використовується шлях, ідея якого описується в BIP32, а в BIP44 отримав вигляд який зараз використовується. Шлях складається з 5 рівнів:

1) Призначення, вказує який стандарт використовується.
2) Тип коїну, вказує яка криптовалюти використовуються. Це константа, в SLIP0044 резервує номера для великої кількості коїнів.
3) Акаунт вказує який акаунт використовуються.
4) Зміна, 0 для зовнішніх мереж та 1 для внутрішніх
5) Індекс, використовується в отриманні BIP32 як додатковий індекс, починається з 0 та послідовно зростає.

Адреса достатньо часто потрібна для користувачів гаманців, вона використовується для отримання криптовалюти та у побудові транзакції.

Бібліотека надає інструментарії для створення та підписування транзакції, а поширення їх DLT мережею займається додаток ,який використовую бібліотеку, це можна пробачати на діаграмі послідовності створення транзакції які зображена на рис. 3.3.

Послідовний опис створення транзакції:

- Користувач створює нову транзакцію.
- Додаток викликає функцію buildTransaction.
- Wallet валідує адреси.
- Wallet викликає Sign.

- Signer повертає підписану транзакцію в вигляді порядку байтів.
- Wallet повертає побудовану та підписану транзакцію до додатку.
- Додаток викликає функцію CoinNode.Network.
- Wallet повертає URL вузла DLT мережі.
- Додаток відправляє транзакцію до DLT мережі.
- Користувач отримає результат відправки транзакції.

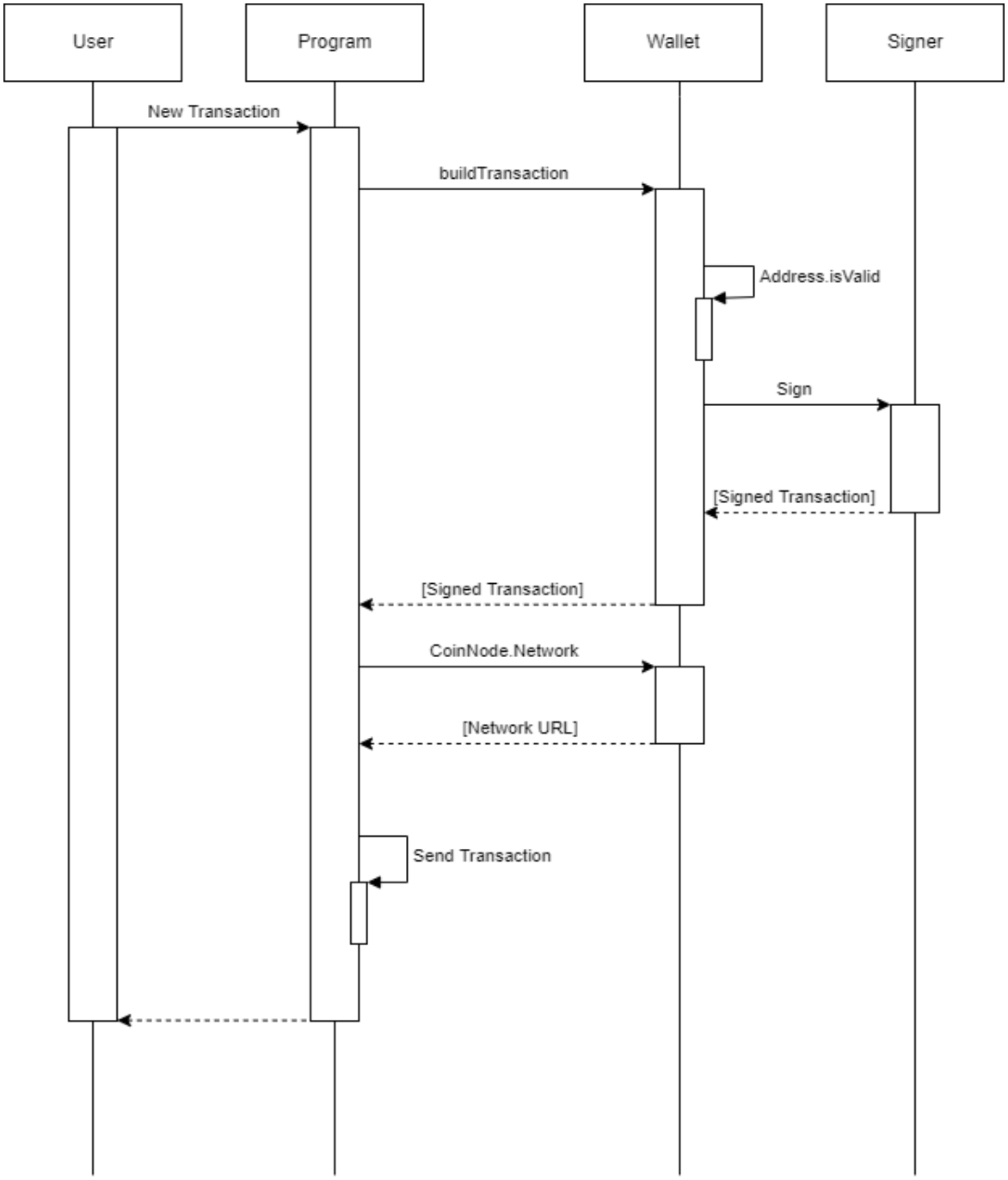


Рисунок 3.3 – Діаграма послідовності створення транзакції

3.2 Структура класів

Діаграма класів – UML діаграма, за допомогою якої модулюють об’єктно-орієнтованих систем. Її називають «статичною діаграмою» тому, що вона зображає класи разом з методами і атрибутами та статичний зв’язки між класами.

Діаграма класів зображена на рис 3.4 описує основні класи бібліотеки.

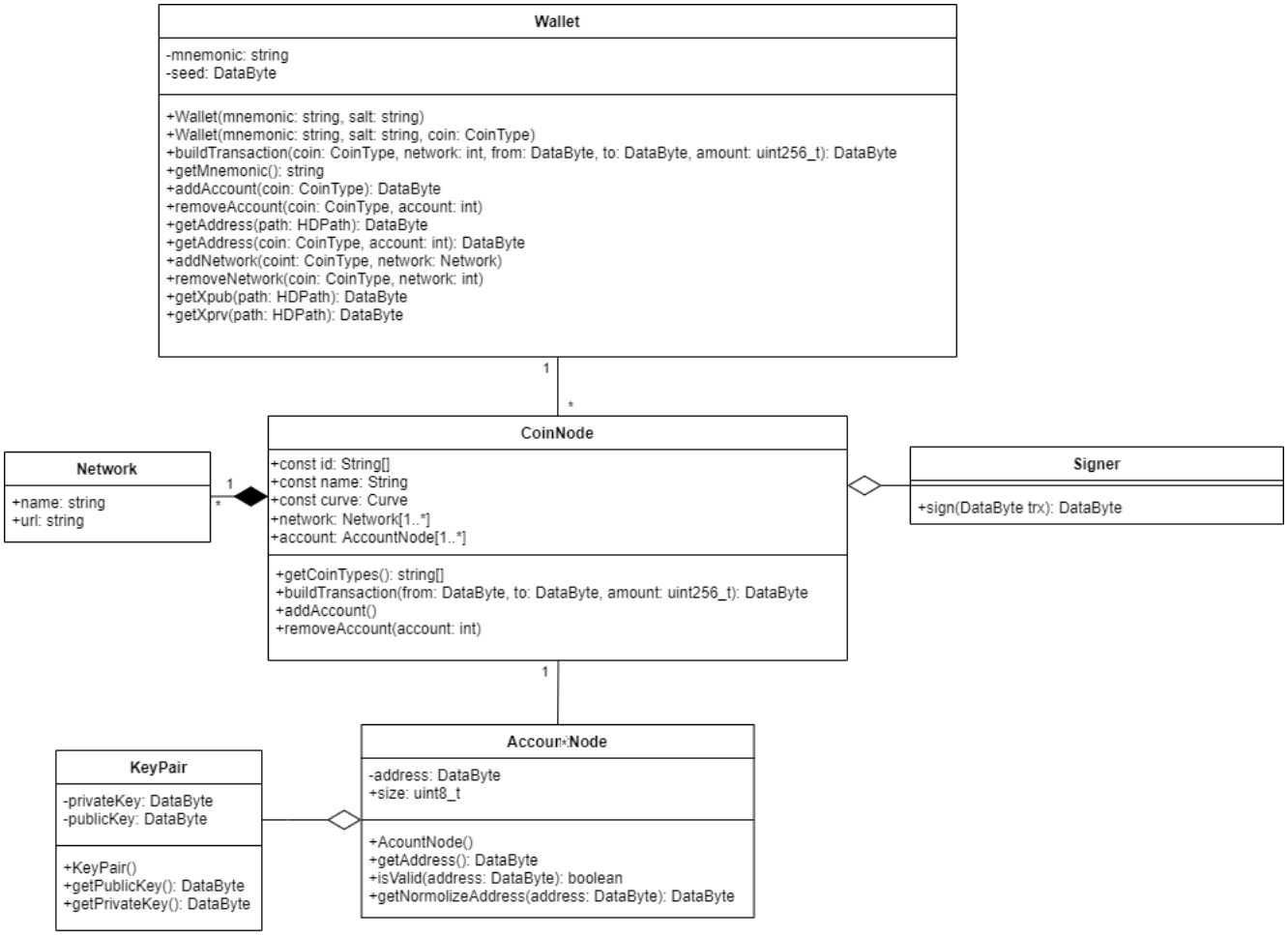


Рисунок 3.4 – Діаграма класів

Як видно з діаграми класів, бібліотека складається з 3 основних класів, а саме Wallet, CoinNode, AccountNode.

Wallet – це основний клас, він допомагає взаємодіяти з усіма криптовалютамі які підтримує бібліотека, в ній реалізовані універсальні метод для створення транзакції, функції додавання та видалення для акаунтів та мереж.

CoinNode цей клас використовується як батьківський клас для створення основних класів криптовалют. В ньому реалізовані методи потрібні для підтримки DLT мережі.

AccountNode клас який використовується при додаванні підтримки DLT мережі. Основний функціонал - це генерування ключової пари, створення адресу та його валідація відповідно до вимог коїну.

В таблиці 3.1 представлені основні класи і методи.

Таблиця 3.1 – Основні класи і методи

Клас	Метод	Параметри	Дія
Wallet	Wallet	string mnemonic, string salt	Створює новий гаманець.
Wallet	Wallet	string mnemonic, string salt, CoinType coin	Створює новий гаманець з підтримкою відповідного коїну.
Wallet	buildTransaction	CoinType coin, int network, DataByte from, DataByte to, uint256_t amount	Створює та підписує транзакцію, відповідно до вимог обраної DLT мережі.
Wallet	getMnemonic		Повертає мнемонічну фразу, яку було використано для створення цього гаманця.

Продовження таблиці 3.1 – Основні класи і методи

Wallet	addAccount	CoinType coin	Створює новий акаунт для відповідного коїна.
Wallet	removeAccount	CoinType coin, int account	Видаляє певний акаунт у відповідного коїну.
Wallet	getAddress	HDPath path	Повертає адресу яка відповідає шляху.
Wallet	getAddress	CoinType coin, int account	Повертає адресу належить певному коїну та має певний номер.
Wallet	addNetwork	CoinType coin, Network network	Додає мережу у список відповідного коїну.
Wallet	removeNetwork	CoinType coin, int network	Видаляє з списку певну мережу у обраного коїну
Wallet	getXpub	HDPath path	Повертає розширений публічний ключ відповідно до шляху.
Wallet	getXprv	HDPath path	Повертає розширений

Продовження таблиці 3.1 – Основні класи і методи

Wallet	getXprv	HDPath path	приватний ключ відповідно до шляху.
CoinNode	getCoinTypes		Повертає список підтримуваних коїнів.
CoinNode	buildTransaction	DataByte from, DataByte to, uint256_t amount	Створює транзакцію та поодписує відповідно до стандарту коїні.
CoinNode	addAccount		Створює новий акаунт.
CoinNode	removeAccount	Int account	Видаляє акаунт відповідно до номер.
Signer	sign	DataByte data	Підписує дані.
AccountNode	AccountNode		Генерує ключову пару та адресу, відповідно до вимог DLT мережі.
AccountNode	getAddress		Повертає адресу акаунта.

Продовження таблиці 3.1 – Основні класи і методи

AccountNode	isValid	DataByte address	Проводить валідацію адреси відповідно до норми певного розподіленого реєстру.
AccountNode	isEqual	DataByte address	Перевіряє чи схожа отримана адреса з отриманою.
KeyPair	getPublicKey		отриману адресу.
KeyPair	getPublicKey		Повертає публічний ключ.

4 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 План тестування

На основі функціональних вимог було розроблено план тестування в якому описано необхідні ресурси та попередні налаштування для тестування, визначені цілі тестування, та методики його проведення. Окрім цього було визначено які результати та функціонал має бути протестована. План тестування представлений в таблиці 4.1.

Таблиця 4.1 – План тестування

№	Розділ	Елементи
1	Вступ	У цій таблиці описується тестування бібліотеки криптовалютного гаманця.
2	Тестові елементи	- створення гаманця; - контроль гаманця; - створення транзакції; - контроль списку мереж; - контроль списку адрес;
3	Функціонал, що підлягає тестуванню	- створення гаманця; - отримання мнемонічної фрази - створення транзакції; - підписування транзакції; - додавання мережі; - видалення мережі; - отримання мережі; - додавання адреси - видалення адреси;

Продовження таблиці 4.1 – План тестування

	Функціонал, що підлягає тестуванню	- отримання адреси; - отримання публічного ключа
4	Тестові підходи	Тестування проводить за допомогою підходу gray box.
5	Критерій успішно/не успішно	- всі вказані вимоги мають покриватись позитивними тестами, і хоча б одним негативним тестом - всі блокуючі дефекти мають бути виправлені;
6	Критерій призупинення/відновлення	- гаманець не створюється/створюється;
7	Вимоги до середи	Необхідна комп'ютер, на якому має бути встановлена ОС Windows.
8	Вимоги до навичок персоналу	Персона повинна мати основні знання для написання коду та опит використання однієї з мов загального призначення.
9	Задачі тестування	- пошук дефектів, що можуть викликати відмову застосунку; - формування сценаріїв для перевірки виконання функціоналу; - підготовка звіту про роботу.
10	Супровід тестування	- тестовий план; - тестові сценарії; - специфікація формату запитів; - журнал помилок.
11	Розклад тестування	Для виконання кожного тесту виділяється одна година.

Продовження таблиці 4.1 – План тестування

12	Відповідальні	Автор плану тестування.
13	Підтвердження	Для переходу на наступний етап, необхідно рішення відповідального за тестовий план
14	Ризики	При зменшенні часу на тестування, перевірити додаток в повному обсязі не вдасться. Буде виконано або часткову перевірку деяких пунктів, або виключення пунктів з плану тестування, що знизить загальну якість тестування застосунку. Критичні дефекти можуть бути не знайдені і невиправлені.

4.2 Процес тестування

Щоб контролювати рівень якості програмного забезпечення на етапі розробки використовують модульне тестування. Тести, в цьому методі, можна писати як до розробки, цей підхід називають Test Driven Development, так і після, в залежності від потреб та вподобань розробник. Для цього проекту деякі тести були написані до розробки, а деякі після.

Для мов програмування C++ було обрано GTest – це фреймворк для тестування програмного забезпечення, створений Google. Для його використання треба щоб кодова база та компілятор були сумісні з стандартом C++11 або новіше. З платформ він підтримує Windows, MacOS, Linux, а з компіляторів gcc, clang, MSVC.

Щоб проводити модульне тестування на мові Kotlin було обрано JUnit, на даний момент остання версія JUnit 5, вона була націлений на адаптацію стилю програмування Java 8 або новіше. Він складається з 3 модулів JUnit Platform, JUnit Jupiter, JUnit Vintage.

Для прикладу розглянемо тестування класу AccountNode наведеного в таблиці 4.2. В процесі тестування перевіряються відкриті методи репозиторію, кожному тесту відповідає тестовий метод. Поетапно описуються дії необхідні для виконання тесту, очікуваний та отриманий результат, а також висновок.

Таблиця 4.2 - Процес тестування

№	Процес	Назва	Етапи	Очікуваний результат	Реальний результат	Висновок
1	Перевірка чи відповідає адреса вимогам DLT мережі	isValidTest	отримує адресу для перевірки.	Повертає позитивну відповідь.	Повернула позитивну відповідь.	Успіх
2	Перевірка однаковість отриманої адреси з власною	isEqualTest	отримує адресу для зрівняння.	Повертає негативну відповідь.	Повернула негативну відповідь.	Успіх

ВИСНОВКИ

1. Проведено аналіз криптовалют. Визначено, що всі вони базуються на технології розподіленого реєстру, які використовують криптографічні алгоритм як для забезпечення безпечного використання, так для запобіганню шахрайству. Також було проаналізовано класифікацію криптографічних гаманців за функціоналом та платформами.

2. Проаналізовані існуючі додатки-аналоги, виділено їх переваги та недоліки. Результати зведені у порівняльну таблицю.

3. Обрано та обґрунтовано вибір мов програмування C++ та Kotlin з використання середовищ розробки Visual Studio та Android Studio відповідно, також було обґрунтоване використання бібліотеки Trezor-crypto та утиліти для збірки CMake.

4. Сформульовані функціональні вимоги до бібліотеки, на їх основі розроблена модель варіантів використання. Визначено нефункціональні вимоги до бібліотеки.

5. Проаналізовані архітектурні шаблони MVC, MVP, MVVM за рядом критеріїв та сформульовані пропозицій вибору шаблону, шляхи імплементації бібліотеки в додаток та отримання необхідного рівня безпеки.

6. За допомогою блок-схем було описано процес створення гаманця та процес отримання адреси, а діаграмою послідовності описано процес створення транзакції. Було виділені основні класи, розроблена діаграма класів.

7. Розроблено програмне забезпечення програмної системи, яке відповідає вимогам та задачам, що було висунуто.

8. Розроблено тест план та створенні тест кейси, за допомогою яких було проведено тестування. За результатами тестування продукт повністю відповідає усім вимогам, що було висунуто до системи при проектуванні.

ПЕРЕЛІК ПОСИЛАНЬ

1.Heston A. Cryptocurrencies: How to Safely Create Stable and Long-term Passive Income by Investing in Cryptocurrencies / Anthony Heston. – North Charleston: CreateSpace Independent Publishing Platform, 2017. – 94 с.

2.On the Move to Meaningful Internet Systems. OTM 2018 Conferences - Confederated International Conferences: CoopIS, C&TC, and ODBASE 2018, Valletta, Malta, October 22-26, 2018, Proceedings, Part II / [H. Panetto, C. Debruyne, H. Proper та ін.]., 2018. – 636 с. – (Lecture Notes in Computer Science; т. 11230).

3.DISTRIBUTED LEDGER TECHNOLOGY SYSTEMS A Conceptual Framework [Електронний ресурс] / [M. Rauchs, A. Glidden, B. Gordon та ін.] // Cambridge Centre for Alternative Finance. – 2018. – Режим доступу до ресурсу: https://www.jbs.cam.ac.uk/wp-content/uploads/2020/08/2018-10-26-conceptualising-dlt-systems.pdf

4.Distributed ledger technology overview, concepts, ecosystem [Електронний ресурс] // TELECOMMUNICATION STANDARDIZATION SECTOR OF ITU. – 2019. – Режим доступу до ресурсу: https://www.itu.int/en/ITU-T/focusgroups/dlt/Documents/d12.pdf.

5.Nakov S. Practical Cryptography for Developers / Svetlin Nakov., 2018. – 300 с.

6.Buterin V. Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform. [Електронний ресурс] / Vitalik Buterin. – 2014. – Режим доступу до ресурсу: https://ethereum.org/669c9e2e2027310b6b3cdce6e1c52962/Ethereum_Whitepaper_-_Buterin_2014.pdf.

7.Bitcoin Developer [Електронний ресурс] – Режим доступу до ресурсу: https://developer.bitcoin.org/index.html.

8.Electrum [Електронний ресурс] – Режим доступу до ресурсу: https://electrum.org/.

9.MetaMask [Електронний ресурс] – Режим доступу до ресурсу: https://metamask.io/.

10.Exodus [Електронний ресурс] – Режим доступу до ресурсу: https://www.exodus.com/.

11.Trust Wallet [Електронний ресурс] – Режим доступу до ресурсу: https://trustwallet.com/.

12.Potapenko J. Crypto wallets security as seen by security engineers [Електронний ресурс] / J. Potapenko, A. Hil, A. Voitova. – 2021. – Режим доступу до ресурсу: https://www.cossacklabs.com/blog/crypto-wallets-security/#crypto-wallets-supply-chain-risks.

13.Syd Logan. Cross-Platform Development in C++: Building Mac OS X, Linux, and Windows Applications / Addison-Wesley Professional., 2008 – 576 s.

14.C++ reference [Електронний ресурс] – Режим доступу до ресурсу: https://en.cppreference.com/.

15.Kotlin [Електронний ресурс] – Режим доступу до ресурсу: https://kotlinlang.org/.

16.Visual Studio IDE documentation [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/visualstudio/ide/?view=vs-2022.

17.Android Studio [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/studio.

18.CMake [Електронний ресурс] – Режим доступу до ресурсу: https://cmake.org/.

19.MDN Web Docs Glossary: Definitions of Web-related terms [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Glossary.

Додаток А.

Демонстраційні матеріали

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО- НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

РОЗРОБКА КРОСПЛАТФОРМНОГО КРИПТОВАЛЮТНОГО ГАМАНЦЯ НА C++

Виконав студент 4 курсу
групи ПД-44
Івлєв Ростислав Володимирович
Керівникроботи
К.т.н, доц., доцент кафедри ІПЗ Золотухіна Оксана Анатоліївна

Київ – 2022

АНАЛОГИ

Назва	Переваги	Недоліки
Electrum	1. Підтримка мультипідпису. 2. Має двофакторну аутентифікацію.	1. Підтримує тільки BitCoin.
MetaMask	1. Простий у використанні. 2. Дозволяє додавати мережі.	1. Підтримує тільки Ethereum.
Exodus	1. Підтримує більше 150 коїнів. 2. Підтримує Windows, Android, iOS, MacOS, Linux.	1. Висока комісія за транзакцію 2. Нема можливості додавати мережі.
Trust Wallet	1. Має власне ядро. 2. Підтримує 60 коїнів. 3. Підтримує Android, iOS.	1. Нема можливості додавати мережі.

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи - спрощення процесу контролю кросплатформних криптовалютних гаманців та підвищення їх рівня безпеки.

Об'єкт дослідження - процес контролю криптовалютних гаманців.

Предмет дослідження - програмне забезпечення для контролю криптовалютних гаманців, що забезпечують роботу з кількома DLT мережами.

3

ТЕХНІЧНІ ЗАВДАННЯ

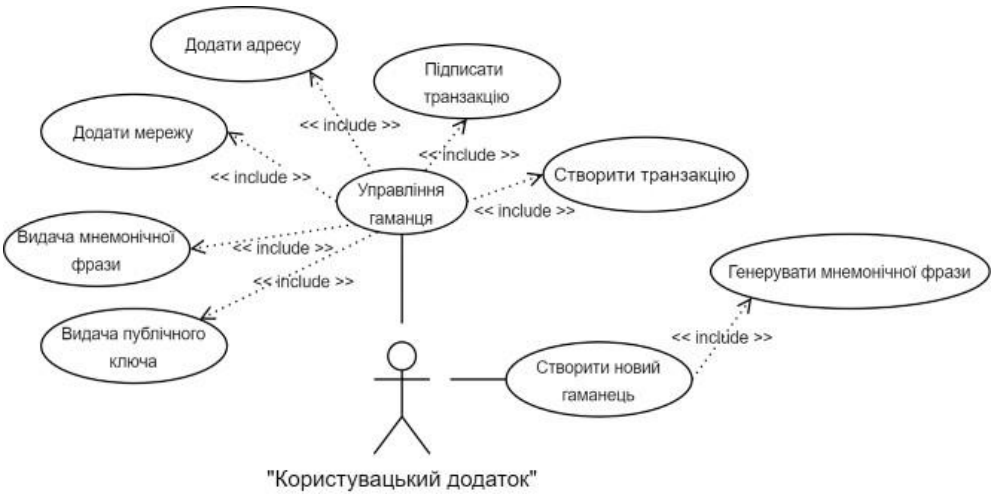
1. Бібліотека повинна основуватися на ієрархічної детермінованої моделі (BIP-32 та BIP-44).
2. Бібліотека повинна виконувати всі основні функції криптовалютного гаманця (генерування ключової пари, створення транзакції, підписування транзакції).
3. Бібліотека повинна забезпечувати просте додовання підтримки нової DLT мережі.
4. Бібліотека повинна бути портована на Android.

4

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

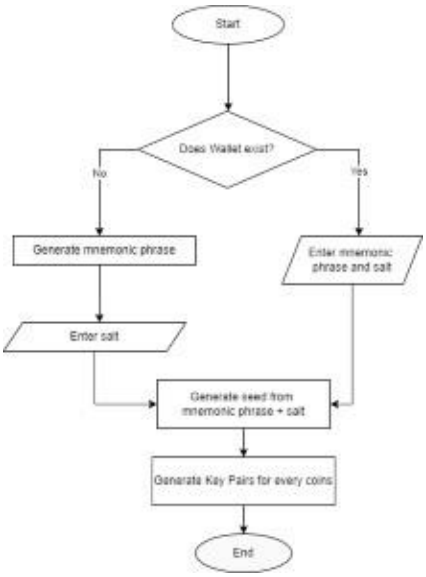
5

ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ



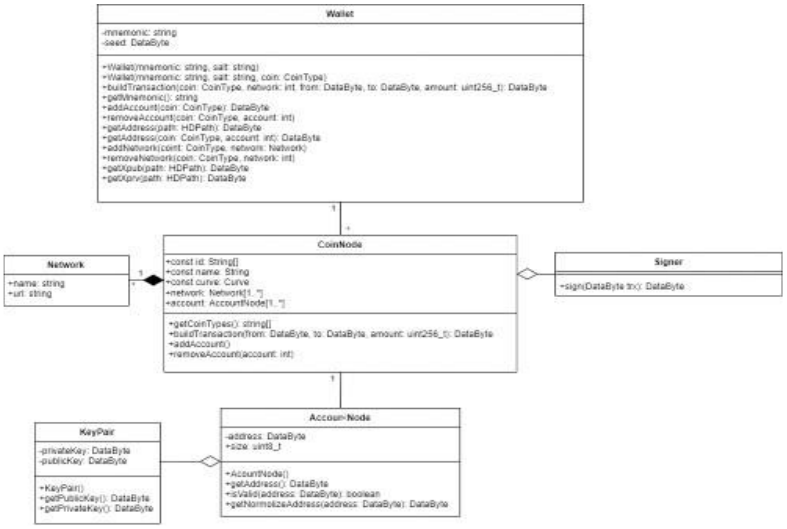
6

АЛГОРИТМ СТВОРЕННЯ ГАМАНЦЯ



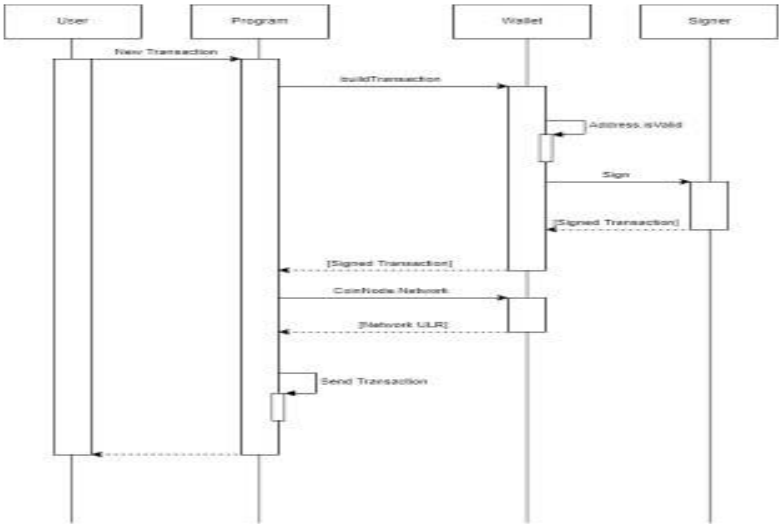
7

ДІАГРАМА КЛАСІВ



8

ДІАГРАМА ПОСЛІДОВНОСТІ СТВОРЕННЯ ТРАНЗАКЦІЇ



9

ПРАКТИЧНЕ ВИКОРИСТАННЯ

10

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

Івлєв Р.В. Підходи до розробкикросплатформнихдодатків мовоюC++ / Івлєв Р.В., Дзядович О.С. // Застосуванняпрограмногозабезпеченняв ІКТ: Матеріали науковотехнічної конференції Збірник тез. 20.4.2022, ДУТ, м. Київ– К.: ДУТ, 2022. – С.25

Івлєв Р.В. Технологія розподіленого реєстру/ Івлєв Р.В., Золотухіна О. А // Сучасні інфокомунікаційнітехнології: Матеріали чотирнадцятої науковотехнічної конференції студентів та молодихвчених. Збірник тез. 19.5.2022, ДУТ, м. Київ– К.: ДУТ, 2022. – С.33

11

ВИСНОВКИ

1. Проведено аналіз аналогів.
2. Розглянуті особливості розробки криптовалютних гаманців.
3. Обрано технологій та середовища розробки.
4. Спроектована та реалізована бібліотека з підтримкою блокчейну Ethereum.
5. Бібліотека перенесена на Android.

12

ДЯКУЮ ЗА УВАГУ!