ДВНЗ «ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ»

Факультет комп'ютерно-інформаційних технологій та автоматизації
Кафедра прикладної математики та інформатики

«До захисту допущено»
в. о. завідувача кафедри ПМІ
_____ Наталія Маслова
(підпис) (ініціали, прізвище)
«____»_____2022 р.

Випускна кваліфікаційна робота

магістра
(освітньо-кваліфікаційний рівень)

на тему: «Аналіз особливостей застосування технології блокчейн Stellar при проєктуванні автоматизованої системи керування криптоактивами»

Виконала: студентка 2 курсу, групи ІПЗм-21а
(шифр групи)

спеціальності 121 Інженерія програмного забезпечення
(шифр і назва напряму підготовки)

Гусєва А. А. _____
(прізвище та ініціали) (підпис)

Керівник доц. кафедри ПМІ, к.т.н., доц. Маслова Н. О. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Нормоконтроль

к.т.н., доц., доц.каф.ПМІ

_____ Ірина Назарова
(підпис)

Засвідчую, що у цій випускній кваліфікаційній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Луцьк – 2022

ДВНЗ «Донецький національний технічний університет»

Факультет комп'ютерно-інформаційних технологій та автоматизації
Кафедра прикладної математики та інформатики
Освітній ступінь магістр
Спеціальність 121 Інженерія програмного забезпечення
(шифр і назва)

ЗАТВЕРДЖУЮ
в. о. завідувача кафедри
_____ /Н. О. Маслова/
«____»_____2022 року

З А В Д А Н Н Я
НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ
Гусєвій Анні Андріївні
(прізвище, ім'я, по батькові)

1. Тема роботи Аналіз особливостей застосування технології блокчейн Stellar при проєктуванні автоматизованої системи керування криптоактивами
Спецчастина
Керівник роботи Маслова Н. О., к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом від « 28 » вересня 20 22 року № 446
2. Строк подання студентом роботи 6 грудня 2022 року
3. Вихідні дані до роботи результати переддипломної практики
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1) аналіз технологій блокчейну для вирішення проблем керування криптоактивами; 2) проєктування автоматизованої системи керування криптоактивами; 3) програмна реалізація автоматизованої системи керування криптоактивами; 4) контрольна перевірка роботи автоматизованої системи керування криптоактивами

5. Перелік графічного матеріалу
Рисунків, діаграм та графіків - 62 , таблиць - 7, в цілому в кількості, достатній для демонстрації матеріалів випускної кваліфікаційної роботи

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 3 вересня 2022 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз блокчейну та проблематики керування криптоактивами. Дослідження аналогів вирішення проблеми та Stellar Network. Постановка задачі на розробку	03.10.22 – 09.10.22	
2	Проєктування функціональної, логічної, математичної моделі, торгівельної стратегії, поведінки в процесі життєвого циклу системи	10.10.22 – 20.10.22	
3	Програмна реалізація автоматизованої системи керування криптоактивами	21.10.22 – 11.11.22	
4	Перевірка функціоналу, опис послідовності роботи та ефективності розробленої системи	31.10.22 – 27.11.22	
5	Оформлення пояснювальної записки та опис створеної системи. Проходження нормоконтролю	28.11.22 – 05.12.22	
6	Перевірка пояснювальної записки антиплагіатною системою. Оформлення презентації до роботи, графічного матеріалу та рецензування роботи	06.12.22 – 19.12.22	
7	Захист роботи	20.12.22	

Студент _____ Гусєва А. А.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Маслова Н.О.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Гусєва А. А. Аналіз особливостей застосування технології блокчейн Stellar при проєктуванні автоматизованої системи керування криптоактивами / Випускна кваліфікаційна робота на здобуття освітнього ступеня «магістр» за спеціальністю 121 «Інженерія програмного забезпечення». – ДВНЗ ДонНТУ, Луцьк, 2022.

Об’єктом дослідження виступає мережа Stellar та засоби для розробки системи управління криптоактивами у блокчейні.

Предмет дослідження – технології блокчейн та Stellar Network, методи та засоби мережі для роботи із криптоактивами та угодами користувачів.

Метою роботи є дослідження особливостей та засобів блокчейн-мережі Stellar для розробки автоматизованої системи, що забезпечить ефективне керування криптоактивами користувача.

Методи досліджень базуються на сучасних положення про блокчейн та мережу Stellar, теорії про роботу з API та розробку автоматизованих систем.

Наукова новизна полягає у створенні ефективної системи для автоматизації криптоактивів користувача на основі мережі Stellar.

Практичне значення полягає у тому, що користувачеві надається автоматизована система для забезпечення ефективної роботи із своїми активами та можливості збільшення основного прибутку, за допомогою засобів мережі Stellar.

Ключові слова: Stellar Network, Python, Stellar SDK, Horizon API, блокчейн, автоматизована система

Список публікацій здобувача:

1. Гусєва А. А. Засоби для створення автоматизованого управління криптовалютним портфелем на основі технології Blockchain з застосуванням Stellar Network / А. А. Гусєва, Н. О. Маслова // Матеріали III Міжнародної студентської наукової конференції «Теоретичне та практичне застосування результатів сучасної науки» (м. Біла Церква, 07.10.2022) – С. 177-178
2. Гусєва А. А. Застосування мережі Stellar при управлінні криптоактивами // А. А. Гусєва, Н. О. Маслова // Наукові праці Донецького національного технічного університету. Серія: «Інформатика, кібернетика та обчислювальна техніка» – 2022, №35. – С.

ABSTRACT

Anna Husieva. Analysis of the peculiarities of the use of Stellar blockchain technology in the design of an automated crypto-asset management system / Graduation thesis for the degree of "master" in the specialty "121 Software engineering". – SHEI DonNTU, Lutsk, 2022.

The object of the study is the Stellar Network and tools for developing a cryptocurrency management system in the blockchain.

The subject of research is blockchain and Stellar Network technologies, network methods and tools for working with cryptoassets and user agreements.

The purpose of the work is to study the features and means of the Stellar Network for the development of an automated system that will ensure the effective management of the user's cryptoassets.

Research methods are based on modern blockchain and Stellar Network theory, API work theory, and automated systems development.

The scientific novelty is the creation of an effective system for automating the user's crypto-assets based on the Stellar Network.

The practical meaning is that the user is provided with an automated system to ensure the efficient operation of their assets and the possibility of increasing the main profit, with the help of the Stellar Network.

Keywords: Stellar Network, Python, Stellar SDK, Horizon API, blockchain, automated system

List of publications:

1. Husieva A. A. Zasoby dlya stvorennya avtomatyzovanoho upravlinnya kryptovalyutnym portfelem na osnovi tekhnolohiyi Blockchain z zastosuvannyam Stellar Network / A. A. Husieva, N. O. Maslova // Materialy III Mizhnarodnoyi students′koyi naukovoyi konferentsiyi «Teoretychne ta praktychne zastosuvannya rezul′tativ suchasnoyi nauky» (m. Bila Tserkva, 07.10.2022) – C. 177-178
2. Husieva A. A. The use of the Stellar network in the management of cryptoassets // A. A. Husieva, N. O. Maslova // Scientific works of the Donetsk National Technical University. Series: "Informatics, cybernetics and computing" – 2022, №35. – C.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ТБ – технологія блокчейн

АСКК – автоматизована система керування криптоактивами

SN – мережа Stellar (Stellar Network)

ПК – персональний комп’ютер

PrK – приватний ключ (private key)

PbK – публічний ключ (public key)

СМ – соціальна мережа

БД – база даних

ЖО – журнал обліку

ТС – торгівельна стратегія

API – Application Programming Interface

xlm – грошова одиниця SN

БТ – бот-творець «BotFather»

ВСТУП

Протягом століть людство відстежувало та фіксувало різноманітну інформацію використовуючи реєстри у вигляді хронологічних списків. Протягом часу інформація набувала цифрового вигляду та з’являлось все більше способів її відстеження, обробки та синхронізації. Таким чином з’явилось вирішення проблем збереження інформації – бази даних, за допомогою яких інформація стала загальнодоступною для використання.

Оскільки світ став більш зв’язним, а взаємодія між людьми більш цифровою, було розроблено більш стійку, масштабну, незалежну від тих чи інших зовнішніх чинників – технологію блокчейн.

Дана робота присвячена дослідженню, проєктуванню та розробці системи для автоматизації, прискорення та простого керування криптоактивами користувачів на основі блокчейну Stellar.

Метою роботи є дослідження особливостей блокчейну-мережі Stellar як засобу для розробки системи, що забезпечить ефективне керування криптоактивами користувача за рахунок автоматизації основних функцій.

До об’єкта дослідження належать мережа Stellar, процеси та засоби для розробки АСКК у блокчейні.

У якості предмету дослідження виступають технології, методи та засоби блокчейн та Stellar Network.

Методами досліджень обрано аналіз блокчейну та поведінки крипторинку, аналіз особливостей роботи SN та засобів для розробки систем на базі даної мережі.

Практичне застосування такої системи полягає у можливості автоматизувати основні робочі процеси з транзакціями, угодами та активами SN у межах блокчейну. Така можливість забезпечить ефективну обробку даних та збільшить основний прибуток користувача.

Для здійснення роботи передбачається виконання наступних завдань:

– проаналізувати технології блокчейну та мережі Stellar;

– дослідити можливі існуючі аналоги розроблюваної АСКК;

– визначити засоби для реалізації АСКК;

– сформувати задачі на розробку АСКК;

– виконати проєктування АСКК у вигляді моделі функціонування, логічної моделі та поведінки системи;

– розробити та описати торгівельну стратегію та її математичну модель;

– реалізувати програмну частину та інтерфейс АСКК;

– виконати тестування роботи розробленої АСКК;

– провести аналіз ефективності роботи АСКК на основі експерименту з декількома користувачами.

В результаті роботи, загальний обсяг роботи складає _ сторінок, що містять: 4 розділи, матеріали у вигляді 62 рисунків, 7 таблиць, 50 джерел інформації, загалом обсяг роботи – 110 сторінок.

1 БЛОКЧЕЙН ТА STELLAR NETWORK ДЛЯ КЕРУВАННЯ КРИПТОАКТИВАМИ

1.1 Аналіз технологій блокчейну для вирішення проблем керування криптоактивами

З появою можливості переказу коштів в електронному форматі виникла проблема подвійних витрат, через що користувачі почали турбуватися за свої кошти. Одним з варіантів рішення даної проблеми стало створення технології блокчейн (ТБ), що має децентралізовану систему без втручання сторонніх осіб із прозорою обліковою книгою, що зветься Bitcoin [1-5] (Btc).

ТБ вважають механізмом для доказу усіх транзакцій мережі, а Btc став новим видом грошей, що вміщує в собі обмін криптографічно захищеними файлами одного рангу з відкритим ключем. Таким чином користувачі почали активно використовувати можливостями ТБ для збереження будь якої важливої інформації [6-8]. Така перспектива технології дала поштовх у створенні імітаторів Btc, що звуться альтернативними валютами (або «альткоіни»). Вони мають той самий загальний принцип дії, але по різному оптимізовані та налаштовані [9-10].

На сьогоднішній час було створено безліч «коінів» (на основі «алькоінів») та варіантів популяризації кожного з них, аби заохотити користувачів інвестувати у них. Початок кругообігу «коінів» та «альткоінів» між користувачами призвів до створення цінового курсу кожного з них [11-14]. Через купівлю та продаж криптоактивів курс певного «коіну»»/«альткоіну» зростав чи падав. Така поведінка курсу змусила користувачів уважніше вивчати поведінку обраних криптоактивів та знаходити можливості для заробітку на таких стрибках [15-17]. Але зі збільшенням криптоактивів на користувацькому рахунку, збільшувалась навантаженість відслідковування кожного з них.

Для того щоб вирішити проблему із відслідковуванням та управлінням «коінами» та «альткоінами» на користувацькому рахунку виникла необхідність у створення автоматизованої системи керування криптоактивами (АСКК), яка буде врахувати можливості:

– обміну криптоактивів між користувачами;
– додаткового збільшення заробітку на ціновому курсі;
– купівлі нових криптоактивів.

Для створення такої системи необхідно враховувати особливості блокчейну та мережі, у якій буде функціонувати АСКК. Для такої системи необхідно обрати мережу яка:

– налічує у собі функціонал необхідний для створення системи;
– надає можливість для перегляду свого алгоритму роботи;
– зрозуміла у використанні;
– має швидку обробку запиту;
– можливість обміну валюти в одній атомарній транзакції.

Для вирішення таких завдань для АСКК було доцільно обрати мережу Stellar (SN) для зниження часу відслідковування стану криптоактивів та збільшення заробітку на них.

1.2 Дослідження існуючих аналогів вирішення проблем

У п.1.1 було зазначено, що розроблювана АСКК буде направлена на мережу SN. Оскільки дана мережа існує вже вісім років [18], доречно було виконати огляд вже розроблених та доступних до використання аналогів.

При виконанні пошуку в мережі Інтернет, було визначено, що АСКК має невелику кількість аналогів, але одиниці з них завершено до кінця. В результаті було знайдено дані засоби: «StellarBot», «Stellar Tipping bot», «Stellarium bot», «Crypto-watcher» та «Stellar Portal».

«StellarBot». Розробник зазначає, що це «Робот для створення ринку Stellar». Дана система розроблена для версій персонального комп’ютера (ПК). Написаний засобами: JavaScript, Vue та HTML [19].

Функціонал «StellarBot»:

– авторизація;

– додавання та видалення «ліній довіри» та активів;

– редагування суми активів;

– обмін, налаштування, зняття обмінної пари;

– перевірка журналу обліку (ЖО).

Інтерфейс «StellarBot». Увесь функціонал представлено на одній вкладці, яку розбито на три секції: «Wallet Information», «Robot Exchange Pair» та «Order». Перші дві області мають поля для налаштувань. Третя область не піддається редагуванню. Також присутній перемикач для включення та виключення «робота». Для ознайомлення інтерфейс представлено у загальному вигляді на рис. 1.1.

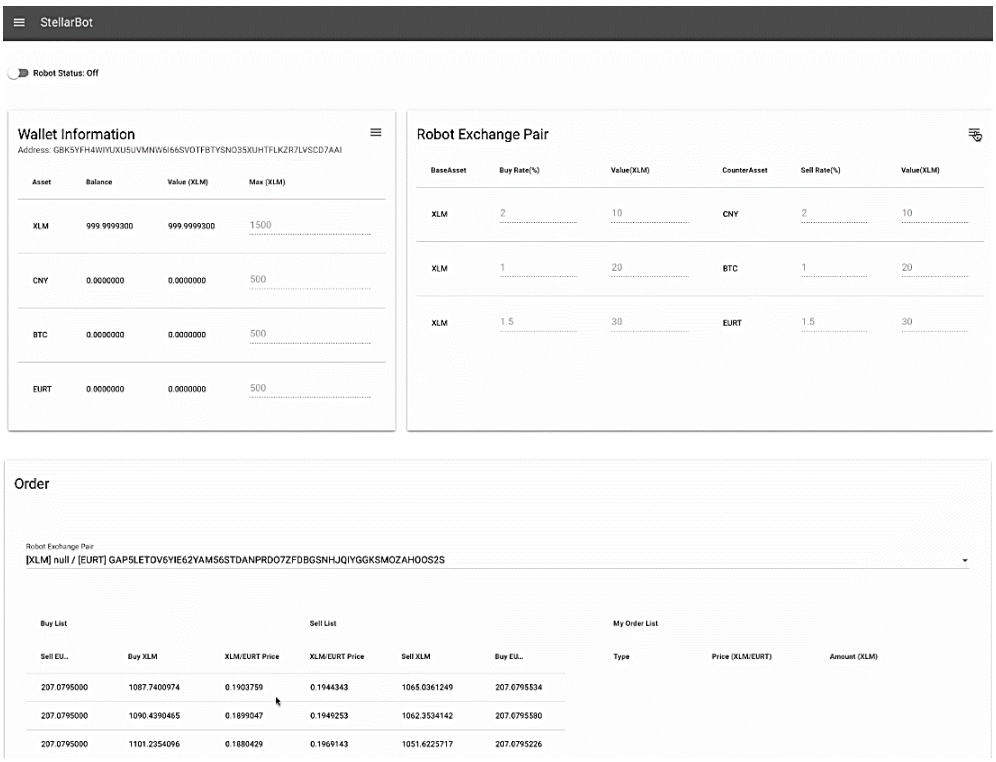

Рисунок 1.1 – Інтерфейс «StellarBot» у загальному вигляду

Переваги:

– автор надає інструкцію користування «StellarBot»;

– можливість роботи із обмінними парами;

– ненавантажений інтерфейс;

– можливість перегляду ЖО.

Недоліки:

– «StellarBot» має тільки ПК версію;

– має лише англійську версію перекладу;

– «секретний ключ» (PrK) користувацького гаманця для авторизації зберігається у браузері.

«Stellar Tipping bot». За описом розробника, даний засіб передбачено як засіб поширення інформації про SN та способу відправлення внутрішньої валюти SN (xlm). Написаний засобами JavaScript та Shell. Розроблено для використання у соціальних мережах (СМ) «Reddit» [20] та «Twitter» [21].

Функціонал «Stellar Tipping bot»:

– відправлення криптоактиву (xlm);

– надання інформації про баланс гаманця;

– поповнення рахунку;

– налаштування анонімності відправлення.

Інтерфейс «Stellar Tipping bot». Для відправлення коштів є один загальний інтерфейс, що має декілька полів вводу та кнопку відправлення (рис. 1.2). Для СМ «Reddit» представлено інтерфейс із двох полів, де зазначається «subject» та «message» (рис. 1.3). Для СМ «Twitter» зв’язок користувача виконується за допомогою відправлення повідомлення.

Переваги:

– майже миттєвий депозит і зняття коштів;

– має інструкцію користувача для кожної платформи;

– для контакту з користувачем використовує соціальні мережі;

– «Stellar Tipping bot» не запитує PrK.

Недоліки:

– обмежений функціонал;

– має лише англійський інтерфейс перекладу;

– необхідно точно знати формулювання запитів.

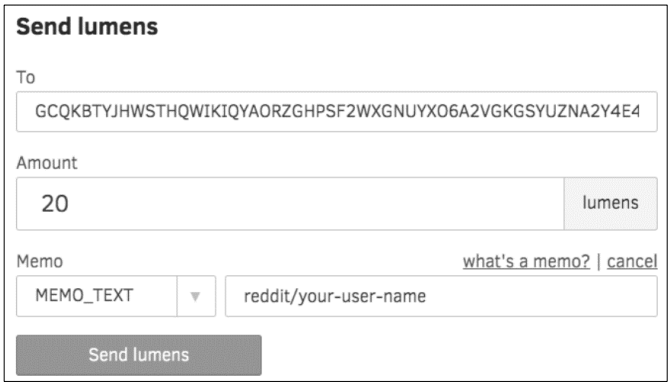

Рисунок 1.2 – Інтерфейс «Stellar Tipping bot» у загальному вигляду

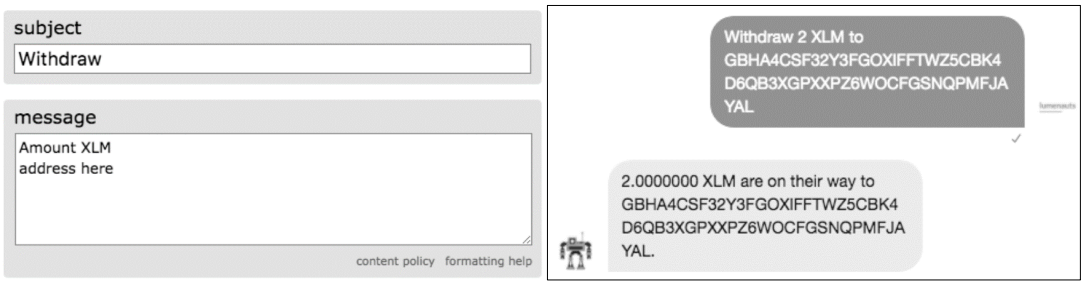

Рисунок 1.3 – Інтерфейс «Stellar Tipping bot» для «Reddit» та «Twitter» відповідно

«Stellarium bot». За позначенням розробника, дана програма призначена для відправлення повідомлень користувачеві щодо подій у його гаманці. Написана засобами Go для СМ «Telegram» [22].

Функціонал «Stellarium bot». Налічує шість команд:

– «/start stellar_address» – запуск бота;

– «/stop» – зупинка бота;

– «/info» – відображення інформації про обліковий запис;

– «/qr» – відображення QR-коду «stellar address»;

– «/help» – довідка про команди;

– «/donate» – надсилання адресу для «подяки».

Інтерфейс «Stellarium bot». Має стандартний інтерфейс бота «Telegram». Для запуску або зупинки необхідно відправляти відповідні команді через поле

вводу. Результати спостережень надсилаються у вигляді звичайних текстових повідомлень. Екранні форми роботи «Stellarium bot» на рис. 1.4.

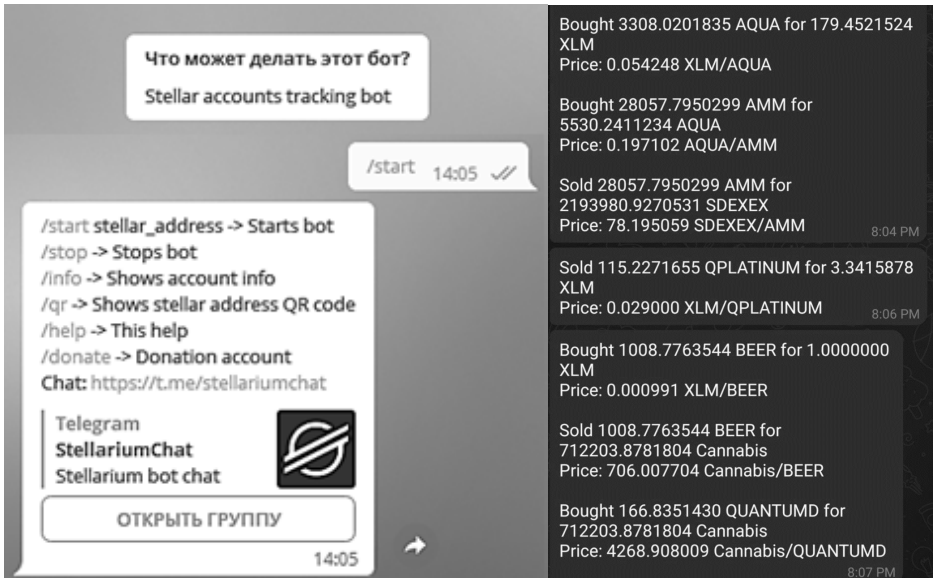

Рисунок 1.4 – Інтерфейс «Stellarium bot» у загальному вигляді

Переваги:

– простий функціонал;

– відстеження дій в гаманці у реальному часі;

– зв’язок із ботом виконується з будь-яких платформ.

Оскільки функціонал бота невеликий та примітивний, він має лише один, але дуже вагомий недолік – невідомо яким чином буде захищено PrK користувача.

«Crypto-watcher». Розробник зазначає, що даний продукт показує курс криптоактивів в доларах США та євро. Реалізовано засобами Python [23].

Функціонал «Crypto-watcher»:

– має кнопку «Help» для надання інформаційної довідки;

– має кнопку «Menu» для створення особистого списку криптоактивів;

– виконує відображення курсу обраних криптоактивів.

Інтерфейс «Crypto-watcher»:

– має мінімум зайвих функцій;

– надає можливість зміни кольорової теми системи;

– перегляд криптоактивів відображено за допомогою таблиці із чотирьома стовпцями: номер активу, назва, ціна USD, ціна EUR, ціна GBP.

На рис. 1.5 представлено загальний вигляд застосунку «Crypto-watcher».

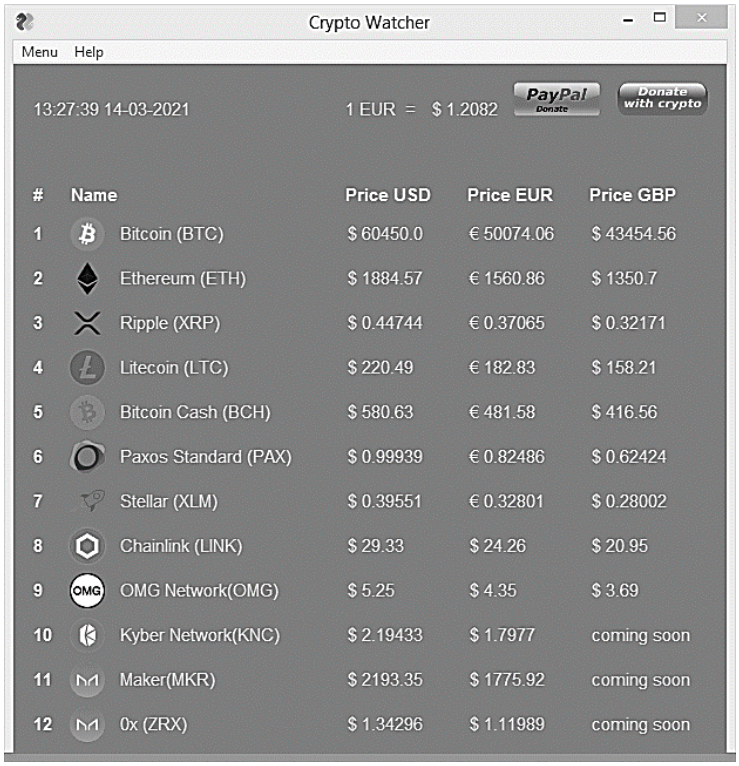

Рисунок 1.5 – Інтерфейс «Crypto-watcher» у загальному вигляді

Переваги:

– зрозумілий інтерфейс;

– можливість вибору бажаних до відслідковування криптоактивів;

– відстеження курсу криптоактивів у реальному часі.

Недоліки:

– відсутня інструкція для користувача;

– відсутній вибору бажаної валюти для відслідковування курсу;

– деякі кольорові теми зливаються із іконками криптоактивів.

«Stellar Portal». Розробник описує програму як веб-додаток для повного контролю обліковим записом у SN. Розроблений засобами JavaScript, HTML та CSS [24].

Функціонал «Stellar Portal»:

– перегляд ЖО;

– перемикач для роботи у тестовій або публічній мережі;

– створення, редагування та видалення угод у ЖО;

– надсилання платежів;

– перегляд особистого рахунку.

Інтерфейс «Stellar Portal» представлено у вигляді сторінки, поділеної на чотири секції: «Balance», «Offers», «Operations» та «Order Book». Секції мають: поля для вводу, перемикачі, форми відображення інформації, кнопки та випадаючі списки. На рис. 1.6 показано інтерфейс описаного додатку.

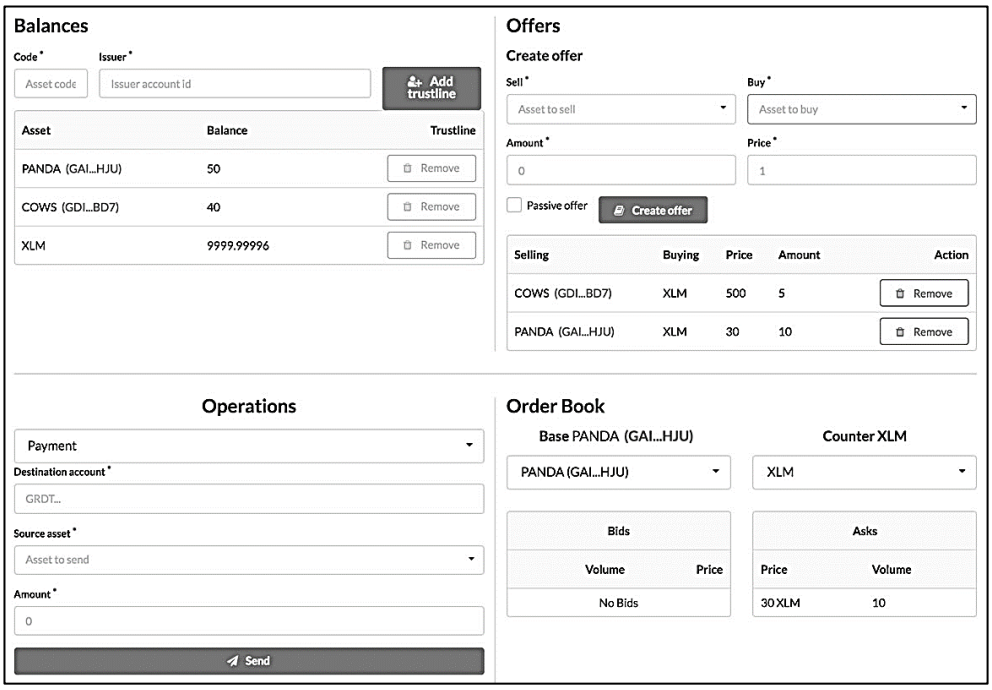

Рисунок 1.6 – Інтерфейс «Stellar Portal» у загальному вигляді

Переваги:

– надання основних функцій для роботи із особистим гаманцем;

– простий інтерфейс;

– надання можливості роботи у тестовій мережі.

Недоліки:

– відсутня інструкція користувача;

– тільки англійська локалізація інтерфейсу;

– розроблена тільки для версій ПК;

– невідомі умови безпеки даних.

Виконавши огляд аналогів у вільному доступі, отримано висновки щодо проєктування та розробки АСКК виключивши можливі недоліки, наявні в її аналогів.

1.3 Stellar Network та засоби реалізації системи

Як було зазначено раніше, Stellar має відкриту децентралізовану мережу в загальному доступі. Основні задачі SN направлені на транскордонні перекази, створення доступних та ефективних платіжних рішень [25-27].

Книга обліку Stellar виконує перевірку та оновлення своїх даних кожні п’ять секунд. Така швидкість отримана в результаті роботи алгоритму «Stellar Consensus Protocol» (SCP), що виконує синхронізацію даних із використанням методу «Proof-of-Agreement» (PoA) [28-31].

У свою чергу PoA найбільш ефективний, порівняно зі старими блокчейнами, оскільки не вимагає грубого рішення складних задач, великих потужностей та безперервного охолодження обладнання.

SCP – це протокол консенсусу, що базується на візантійському кворумі з відкритим членством. У якості кворумів виступають результати різних рішень щодо конфігурацій вузлів. Самі вузли розпізнають лише кворум до якого вони належать, після валідації та вивчення конфігурацій решти членів кворуму [32]. Процес валідації Stellar можна представити у вигляді архітектури, що складається з: Stellar core (ядро), SQL database (БД), Horizon (це API для взаємодії зі SN), federation server (ФС). Дану архітектуру представлено на рис. 1.7.

АСКК виступає клієнтом, що надає запити до SN за допомогою Horizon. Дане API є провідним мостом до Stellar Core між клієнтом-програмою (у нашому випадку АСКК). Виклик API напряму збільшує складність виконуваного проєкту, оскільки Horizon має стиль архітектури – «RESTful API» [33-34]. Виходячи з цього, SN надає можливість використання SDK

майже будь-якою мовою програмування, що надає широкий вибір інструментів реалізації.

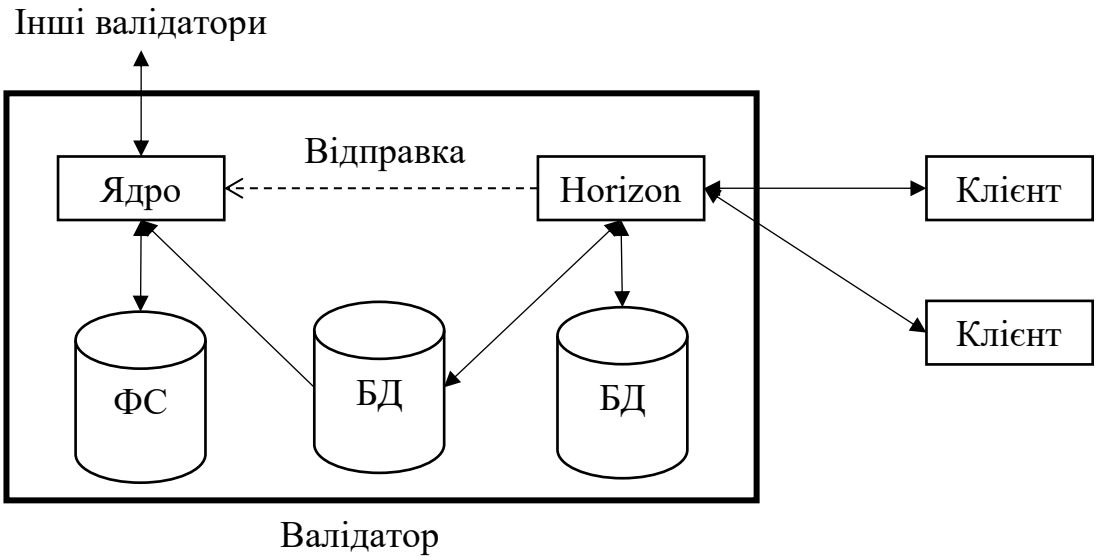

Рисунок 1.7 – Архітектура процесу валідації Stellar

Для реалізації АСКК була обрана мова Python, яка є актуальною, оскільки надає можливості:

– реалізації системи для будь-якої платформи;

– оптимізації, якості та швидкості обробки різних об’ємів даних;

– великий об’єм документації із розробки;

– роботи із ТБ [33, 35-36].

Так як АСКК передбачена для комфорту та швидкості користування, необхідно забезпечити доступ до неї за допомогою різних платформ. Для цього треба взяти до уваги, що АСКК:

– не повинна забирати багато пам’яті на пристрої;

– інтерфейс не повинен мати зайвих елементів та бути інтуїтивно зрозумілим;

– не повинна потребувати великих потужностей (або не потребувати їх взагалі);

– повинна бути поруч із її користувачем (для забезпечення контролю її дій).

Для реалізації таких вимог буде доречно реалізувати АСКК у вигляді «бота» в СМ, оскільки від користувача потребуватиметься лише відправлення запиту до системи. У якості СМ було обрано «Telegram», обґрунтовано це тим, що дана СМ:

– представлена для усіх сучасних платформ;

– має власний API для розробки «ботів»

– не потребує потужностей пристрою для оброки інформації;

– не потребує зайвого місця у пам’яті при використанні АСКК;

– миттєве сповіщення користувача від системи, про результат виконання запиту;

– завжди у доступному для користувача місці [33, 37-38].

Виконавши огляд ТБ, існуючих аналогів, SN та засобів для реалізації системи, необхідно сформувати задачі на розробку АСКК.

1.4 Постановка задач на розробку

Призначення, що має АСКК:

– відслідковування подій в ЖО користувача;

– зміна параметрів «угод» у ЖО користувача за параметрами;

– передавати користувачеві повідомлення про результати відслідковування та зміни у ЖО;

– шукати вигідні активи за наданим «доменом» від користувача;

– надання інформації про баланс криптоактивів;

– можливість переказу користувацьких коштів за наданою адресою отримувача.

Цілі, що має АСКК:

– збільшення заробітку користувача на власних криптоактивах;

– надання доступу до управління користувацьким ЖО із будь якого пристрою;

– зменшення можливості виникнення помилки користувача при роботі з ЖО;

– збільшення продуктивності користувача за рахунок звільненого часу від особистого керування ЖО;

– звільнення користувача від рутинної перевірки та редагування «угод».

Об’єкти автоматизації розробки що має АСКК:

– зміна параметрів угод на покупку/продаж криптоактиву;

– пошук нових вигідних угод;

– відправка криптоактиву;

– авторизація користувача;

– відслідковування подій в ЖО.

Вимоги щодо функціоналу та структури АСКК.

У якості вимог до функціоналу та структури системи представлено наступні пункти:

– АСКК повинна виконувати свій функціонал для власників особистого гаманця у SN;

– АСКК повинна виконувати свій функціонал для користувачів СМ «Telegram»;

– передбачення команди авторизації користувача із використанням особистого ключа від особистого гаманця;

– наявність опису торгівельної стратегії (ТС) (налаштована у системі за замовченням);

– наявність команди запуску АСКК;

– наявність команди зупинки АСКК;

– наявність команди пошуку криптоактивів за заданим «доменом»;

– наявність команди відслідковування подій у ЖО;

– наявність команди постійного редагування угод (з продажу або покупки) у ЖО;

– наявність команди для редагування ТС для редагування угод (з продажу або покупки) та пошуку вигідних для купівлі криптоактивів;

– наявність команди перегляду балансу користувача;

– наявність команди переводу криптоактиву за заданою адресою;

– АСКК повинна повідомляти користувача у вигляді текстового повідомлення стосовно результатів своєї роботи: результат зміни параметрів угоди, результат пошуку вигідного криптоактиву, отримання події у ЖО користувача, результат переводу криптоактиву.

Вимоги щодо чисельності користувачів передбачають врахування того факту, що користувач одночасно може управляти лише одним гаманцем.

Вимоги щодо підготовленості користувачів АСКК:

– навички користування обраним технічним засобом або браузером, достатніх для встановлення або відкриття веб-версії СМ «Telegram»;

– навички користування СМ «Telegram», достатніх для реєстрації/авторизації акаунту, користування пошуковою строчкою, ведення чату та налаштування повідомлень СМ;

– розуміння побудови особистої ТС, достатніх для використання стратегії за замовчуванням або налаштування особистої

– знання ТБ та SN, достатніх для створення особистого гаманця та розуміння яким чином виконуються операції у блокчейні.

Вимоги щодо режиму роботи АСКК:

– АСКК повинна почати роботу з моменту виклику користувачем команди запуску;

– виконання АСКК повинно бути безперервним до моменту, виклику користувачем команди зупинки;

– обмежене використання інших команд (окрім команди зупинки) під час відслідковування угод у ЖО;

– обмежене використання інших команд (окрім команди зупинки) під час відслідковування подій у ЖО;

– обмежене використання інших команд (окрім команди зупинки) під час пошуку вигідних криптоактивів.

Вимоги щодо застосування БД – даних АСКК про параметри ТС користувача повинні зберігатись засобами БД SQLite.

Вимоги щодо безпеки АСКК:

– користувач повинен проходити процес авторизації перед початком роботи у АСКК;

– АСКК не повинна зберігати PrK користувача до БД;

– дані про налаштовані параметри АСКК закріплюються за відповідним ідентифікатором користувача.

Вимоги до математичного забезпечення – редагування угод у ЖО користувача повинні виконуватись відповідно до зазначеної ТС, що буде зазначена в р. 2 п. 2.3.

Програмне забезпечення повинно передбачити виконання двох вимог. Першою та основною вимогою є доступ до мережі Інтернет (для надсилання запитів системі). Другою вимогою є наявність браузера або мінімальної версії системи для пристрою, обраного користувачем. Перелік пристроїв та їх версій наведено на табл. 1.1.

Таблиця 1.1 – Мінімальні вимоги до версій ОС для обраних засобів

Вид засобу	Назва ОС	Версія
Мобільний пристрій	Android	Android 4.0 Ice Cream Sandwich та вище
	iOS	iOS 9.0 та вище
Розумний годинник	iOS	iOS 5.0 та вище
ПК	macOS	macOS 10.11 та вище
	Windows	Windows 7 та вище
	Linux	Linux Mint, Ubuntu та Debian

Технічне забезпечення вимагає можливість вільного вибору користувача серед зазначених технічних засобів:

– ПК: ноутбук, стаціонарний комп’ютер;

– мобільний пристрій: телефон, планшет, розумний годинник.

Основна вимога до лінгвістичного забезпечення – створення АСКК засобами Python.

1.5 Висновки за розділом

При роботі із першим розділом було отримано такі результати роботи:

– виконаний аналіз сучасних проблем, які виникають при керуванні власними криптоактивами, знайдено вирішення за допомогою ТБ;

– виконаний пошук аналогів АСКК, проаналізовано їх функціонал та інтерфейс, визначено недоліки та переваги аналогів;

– описано переваги та особливості роботи із SN;

– визначено засоби для реалізації АСКК;

– описано вимоги до розроблюваної АСКК та сформовано задачі на розробку.

На основі сформованої інформації в першому розіділі, далі пранується виконати проєктування АСКК для подальшої розробки.

2 ПРОЄКТУВАННЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ КЕРУВАННЯ КРИПТОАКТИВАМИ

2.1 Модель функціонування

При роботі з АСКК користувач повинен виконувати дії в деякій послідовності. Відображення усіх можливих сценаріїв варіацій використання системи АСКК можливе у вигляді діаграми «use case» [39-43].

Головний актор даної моделі є «Користувач», що має особистий рахунок у SN та користується СМ «Telegram». Для користування системою, йому доступні такі функції: «Управління системою», «Авторизація» та «Робота з командним меню».

Функція «Управління системою» це функція, що надається за замовчуванням СМ «Telegram» для комфортного користування «діалогом» із системою розробника. Дана функція розширюється за допомогою трьох функцій:

– «Управління роботою системи»;

– «Пошук повідомлень»;

– «Налаштування звуку повідомлень».

«Управління роботою системи» передбачає процес включення та зупинки діалогу із системою. Для цього функція має два розширення: «Включення системи» та «Зупинка системи».

«Пошук повідомлень» надає можливість користувачеві шукати повідомлення (власні та АСКК) за ключовими словами або фразами.

«Налаштування звуку повідомлень» відповідає за налаштування звуку повідомлень АСКК у «діалозі» та надає дві розширені функції: «Включення звуку повідомлень» та «Вимкнення звуку повідомлень».

Функція «Авторизація» включає у себе функцію «Введення PrK особистого рахунку» що відповідають за авторизацію користувача у АСКК за допомогою PrK.

Функція «Робота з командним меню» надає користувачу можливість управління системи за допомогою командного меню. Для цього, вона розширюється до шести основних функцій:

– «Отримання балансу рахунку»;
– «Переказ коштів»;
– «Пошук за доменом»;
– «Налаштування ТС»;
– «Управління процесом обробки»;
– «Налаштування режиму роботи».

«Отримання балансу рахунку» відповідає за можливість користувача отримати інформації щодо балансу особистого рахунку SN.

«Переказ коштів» надає можливість переказу власних криптоактивів на особистий рахунок іншого користувача. Для цього включені такі три функції як: «Заповнення адреси користувача», «Заповнення суми переказу» та «Заповнення назви активу».

«Пошук за доменом» виконує пошук вигідних криптоактивів за наданим користувачем доменом. Має включену функцію «Введення бажаного домену».

Функція «Налаштування ТС» передбачена для можливості користувача змінити параметри ТС, за допомогою якої працює АСКК. Налаштування умов розширюється двома функціями: «Налаштування умов для редагування угод» та «Налаштування вимог для пошуку за доменом».

«Налаштування умов для редагування угод» відповідає за налаштування ТС при редагування угод ЖО користувача. Вона включає чотири функції:

– «Вибір виду угоди»;
– «Заповнення коеф. різниці к-сті активу угоди»;
– «Заповнення коеф. прибутку угоди»;
– «Заповнення коеф. різниці вартості активу угоди».

«Налаштування вимог для пошуку за доменом» відповідає за налаштування умов ТС при виконанні пошуку вигідних криптоактивів за

доменом, наданим користувачем. Включає у себе дві функції: «Заповнення коеф. прибутку активу домену» та «Заповнення загальної вартості закупки».

«Управління процесом обробки» відповідає за процес обробки інформації ЖО системою (початок та завершення), що регулює користувач самостійно. Розширюється двома функціями: «Запуск процесу обробки ЖО» та «Зупинка процесу обробки ЖО».

«Налаштування режиму роботи» дає можливість перемикання режимів роботи системи. Має дві розширені функції: «Вибір режиму відстеження подій» та «Вибір режиму редагування угод».

На рис. 2.1 представлено спроєктовану модель функціонування майбутньої розробки.

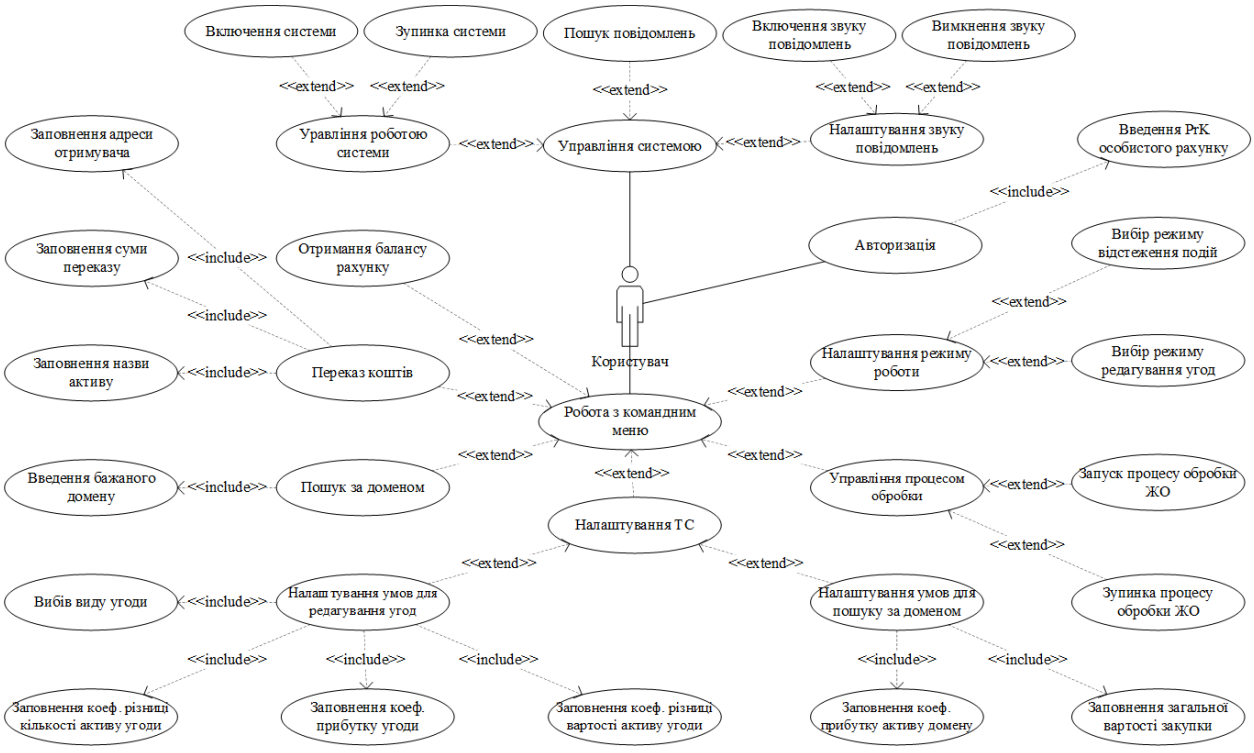

Рисунок 2.1 – Модель функціонування АСКК [32]

Виконавши проєктування функціональної моделі АСКК, необхідно виконати проєктування її логічної моделі.

2.2 Логічна модель

Для представлення внутрішньої структури АСКК було використано діаграму класів [39, 44-47], що містить у собі десять основних класів із таким ім’ям: «Процес_обробки», «Переказ», «Режим_роботи», «Авторизування», «Командне_меню», «Баланс», «Налаштування_ТС», «Пошук_за_доменом», «ТС_редагування_угод», «ТС_пошуку_домена».

На рис. 2.2 представлено логічну модель розроблюваної АСКК.

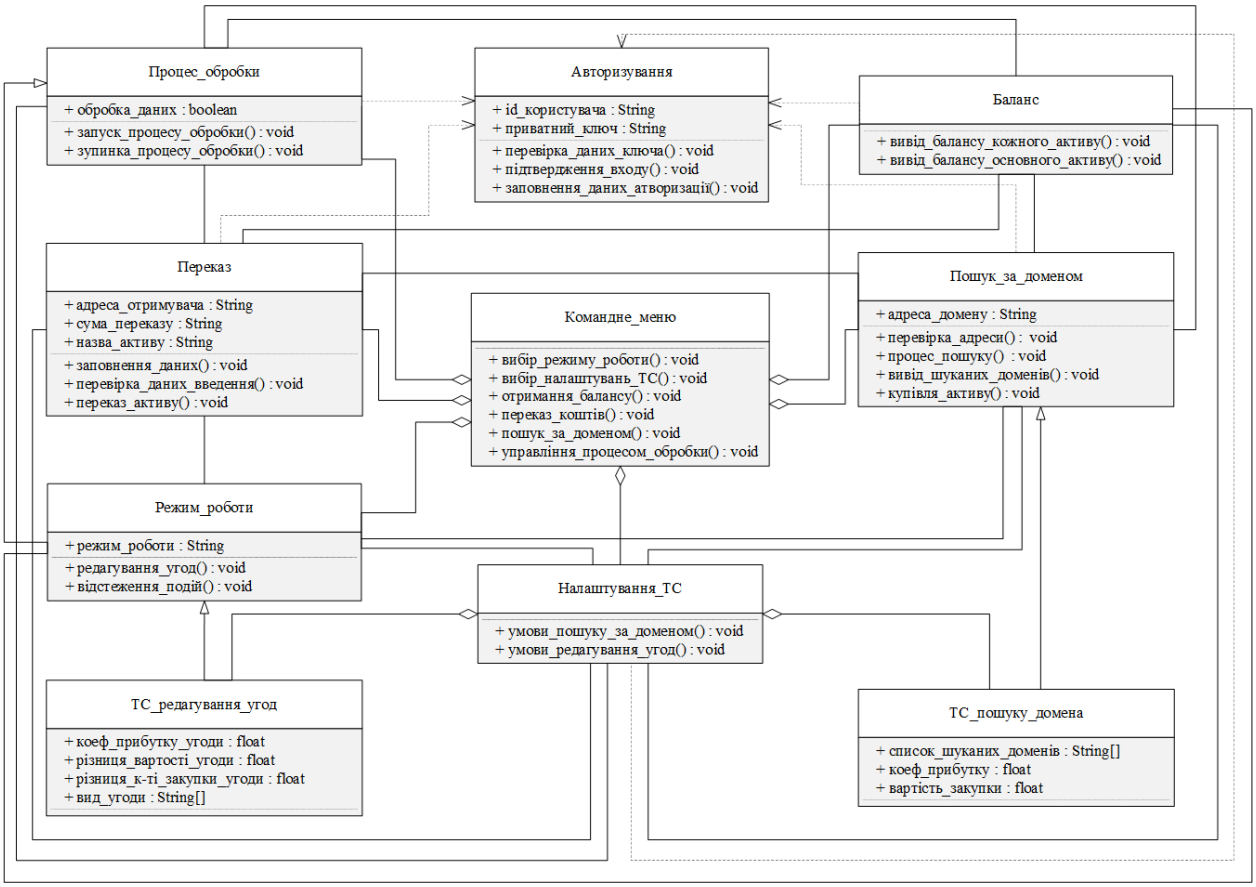

Рисунок 2.2 – Логічна модель АСКК [32]

Клас «Процес_обробки» відповідає за процес запуску або зупинки обробки інформації ЖО користувача. Має один атрибут «обробка_даних», де тип даних – «boolean», доступ – публічний. Операції класу:

– «запуск_процесу_обробки()» (тип повертаючих даних – «void», доступ – публічний);

– «зупинка_процесу_обробки()» (тип повертаючих даних – «void», доступ – публічний).

Даний клас має зв’язок із класами у вигляді відношень:

– узагальнення – клас «Режим_роботи»;

– асоціації – класи «Баланс», «Налаштування_ТС», «Переказ», «Пошук_за_доменом», «Режим_роботи».

Клас «Переказ» відповідає за переказ криптоактиву користувача отримувачу. Має три атрибути та операції. До атрибутів входять:

– «адреса_отримувача» (тип даних – «String», доступ – публічний);

– «сума_переказу» (тип даних – «String», доступ – публічний);

– «назва_активу» (тип даних – «String», доступ – публічний).

До операцій класу входять:

– «заповнення_даних()» (тип повертаючих даних – «void», доступ – публічний);

– «перевірка_даних()» (тип повертаючих даних – «void», доступ – публічний);

– «переказ_активу()» (тип повертаючих даних – «void», доступ – публічний).

Даний клас має зв’язок у вигляді відношення асоціації із класами: «Баланс», «Налаштування_ТС», «Процес_обробки», «Пошук_за_доменом», «Режим_роботи».

Клас «Режим_роботи» відповідає за встановлення одного із режимів обробки інформації ЖО. Має один атрибут – «режим_роботи» (тип даних – «String», доступ – публічний) та дві операції:

– «редагування_угод()» (тип повертаючих даних – «void», доступ – публічний);

– «відстеження_подій()» (тип повертаючих даних – «void», доступ – публічний).

Даний клас має зв’язок із класами у вигляді відношень:

– узагальнення – клас «ТС_редагування_угод»;

– асоціації – класи «Баланс», «Налаштування_ТС», «Переказ», «Пошук_за_доменом», «Процес_обробки».

Клас «Авторизування» відповідає за процес проходження авторизації користувача. Він має два атрибути та три операції. До атрибутів входять:

– «id_користувача» (тип даних – «String», доступ – публічний);

– «приватний_ключ» (тип даних – «String», доступ – публічний).

До операцій входять:

– «заповнення_даних_авторизації()» (тип повертаючих даних – «void», доступ – публічний);

– «перевірка_даних_ключа()» (тип повертаючих даних – «void», доступ – публічний);

– «підтвердження_входу()» (тип повертаючих даних – «void», доступ – публічний).

Даний клас має зв’язок у вигляді відношення залежності із класами: «Баланс», «Налаштування_ТС», «Процес_обробки», «Пошук_за_доменом», «Переказ».

Клас «Командне_меню» виступає як зв’язок користувача із наданими функціями АСКК. Операції класу:

– «вибір_режиму_роботи()» (тип повертаючих даних – «void», доступ – публічний);

– «вибір_налаштувань()» (тип повертаючих даних – «void», доступ – публічний);

– «отримання_балансу()» (тип повертаючих даних – «void», доступ – публічний);

– «переказ_коштів()» (тип повертаючих даних – «void», доступ – публічний);

– «пошук_за_доменом()» (тип повертаючих даних – «void», доступ – публічний);

– «управління_процесом_обробки()» (тип повертаючих даних – «void», доступ – публічний).

Даний клас має зв’язок у вигляді відношення агрегації із класами: «Баланс», «Налаштування_ТС», «Процес_обробки», «Пошук_за_доменом», «Переказ», «Режим_роботи».

Клас «Налаштування_ТС» виконує процес налаштування користувачем ТС для обробки ЖО. Операціями класу виступають:

– «умови_пошуку_за_доменом()» (тип повертаючих даних – «void», доступ – публічний);

– «умови_редагування_угод()» (тип повертаючих даних – «void», доступ – публічний).

Даний клас має зв’язок із класами у вигляді відношень:

– агрегації – класи «ТС_редагування_угод» та «ТС_пошуку_домена»;

– асоціації – класи «Баланс», «Процес_роботи», «Переказ», «Пошук_за_доменом», «Режим_роботи».

Клас «Пошук_за_доменом» виконує пошук та купівлю вигідних криптоактивів за заданим користувачем доменом. Має атрибут «адреса домену» (тип даних – «String», доступ – публічний) та операції:

– «перевіка_адреси()» (тип повертаючих даних – «void», доступ – публічний);

– «процес_пошуку()» (тип повертаючих даних – «void», доступ – публічний);

– «вивід_шуканих_доменів()» (тип повертаючих даних – «void», доступ – публічний);

– «купівля_активу()» (тип повертаючих даних – «void», доступ – публічний).

Даний клас має зв’язок із класами у вигляді відношень:

– узагальнення – клас «ТС_пошуку_домена»;

– асоціації – класи «Баланс», «Налаштування_ТС», «Процес_обробки», «Переказ», «Режим_роботи».

Клас «ТС_редагування_угод» виступає як допоміжний клас для редагування угод ЖС за ТС. Має такі аргументи:

– «коеф_прибутку_угоди» (тип даних – «float», доступ – публічний);

– «різниця_вартості_угоди» (тип даних – «float», доступ – публічний).

– «різниця_к-ті_закупки_угоди» (тип даних – «float», доступ – публічний);

– «вид_угоди» (тип даних – «String[]», доступ – публічний).

Клас «ТС_пошуку_домена» виступає як допоміжний клас для пошуку криптоактивів за доменом ТС. Має такі аргументи:

– «список_шуканих_доменів» (тип даних – «String[]», доступ – публічний);

– «коеф_прибутку» (тип даних – «float», доступ – публічний).

– «вартість_закупки» (тип даних – «float», доступ – публічний);

Клас «Баланс» відповідає за отримання та відображення балансу особистого рахунку користувача. Має операції:

– «вивід_балансу_кожного_активу()» (тип повертаючих даних – «void», доступ – публічний);

– «вивід_балансу_основного_активу()» (тип повертаючих даних – «void», доступ – публічний).

Даний клас має зв’язок у вигляді відношення асоціації із класами: «Баланс», «Налаштування_ТС», «Процес_обробки», «Пошук_за_доменом», «Режим_роботи», «Переказ».

2.3 Торгівельна стратегія та математичні моделі

Оскільки АСКК направлена на збільшення заробітку користувача, було розроблено загальну ТС, яка буде задовольняти основні потреби при роботі із ринком криптоактивів.

Угоди користувача мають два стани – купівля та продаж. Для кожного стану слід передбачити окремі особливості ТС, що мають в своїй основі чіткі математичні моделі.

Також при роботі із угодами, в ЖО окремих криптоактивів, не слід виключати можливість існування інших систем, що мають те саме завдання – заробіток. Цю особливість було включено до ТС.

Узагальнена ТС. Редагування угод відбувається за принципом збільшення або зменшення вартості криптоактиву (за одну одиницю) відповідно до стану купівлі чи продажу. Враховуючи це, аби запобігти можливим втратам користувача було встановлено:

– узагальнену одиницю зміни вартості, за допомогою якої буде відредаговано угоду;

– граничні умови редагування угоди (точки зупинки);

– умови постійної вигоди при формуванні угод.

Враховуючи узагальнену ТС буро розроблено окремі ТС купівлі та продажу та їх математичні моделі, із урахуванням можливості редагування коефіцієнтів користувачем.

ТС купівлі криптоактиву. Зміна ціни для угод на купівлю активу повинна відбуватися за формулою:

$$price_{buy} + \frac{1}{10^{11}}, \tag{2.1}$$

де $price_{buy}$ – поточне значення вартості активу в угоді на купівлю (xlm).

Для забезпечення вигідної угоди купівлі, АСКК буде виконувати перевірку, що задовольняє даній формулі:

$$\frac{price_{sell}}{price_{buy}} \geq prof_{buy}, \tag{2.2}$$

де $price_{sell}$ – поточне значення вартості активу в угоді на продаж (xlm);
$prof_{buy}$ – значення бажаного загального значення вигоди від купівлі активу.

Оскільки постійна реверсивна зміна ціни не є ефективним вирішенням розумного заробітку, передбачено умови, за яких система буде враховувати

значення загального об’єму та загальної вартості угоди активу на купівлю, що стоїть перед угодою користувача.

Значення загальної вартості при купівлі активу визначається за формулою:

$$total_{buy} = price_{buy} * amount_{buy}\,, \tag{2.3}$$

де $total_{buy}$ – значення загальної вартості активу, що купується (xlm);

$amount_{buy}$ – значення загального об’єму активу, що купується.

Угоду на купівлю активу буде відредаговано, якщо загальна вартість ціни активу (див. формулу (2.3)) користувача задовольняє наступній умові:

$$total_{buy} > \frac{total_{buy}}{100} * ttl_{buy}\,, \tag{2.4}$$

де ttl_{buy} – значення користувача для визначення загальної вартості угоди на купівлю (за умовою значення $ttl_{buy} = 0.2$).

Значення загального об’єму активу в угоді (на купівлю) користувача повинно задовольняти такій умові:

$$amount_{buy} > \frac{amount_{buy}}{100} * amnt_{buy}\,, \tag{2.5}$$

де $amnt_{buy}$ – значення користувача для визначення загального об’єму активу угоди на купівлю (за умовою значення $amnt_{buy} = 0.0002$).

ТС продаж криптоактиву. Зміна ціни для угод на продаж активу повинна відбуватися за формулою:

$$price_{sell} - \frac{1}{10^{11}}\,, \tag{2.6}$$

де $price_{sell}$ – поточне значення вартості активу в угоді на продаж (xlm).

Для забезпечення вигідної угоди продажу, АСКК буде виконувати перевірку, що задовольняє даній формулі:

$$\frac{price_{sell}}{price_{buy}} \geq prof_{sell}\,, \tag{2.7}$$

де $prof_{sell}$ – значення бажаного загального значення вигоди від купівлі активу.

Значення загальної вартості активу при продажі визначається за формулою:

$$total_{sell} = price_{buy} * amount_{sell}\,, \tag{2.8}$$

де $total_{sell}$ – значення загальної вартості активу, що продається (xlm);
$amount_{sell}$ – значення загального об'єму активу, що продається.

Угоду на продаж активу буде відредаговано, якщо загальна вартість ціни активу (див. формулу(2.8)) користувача задовольняє наступній умові:

$$total_{sell} > \frac{total_{sell}}{100} * ttl_{sell}\,, \tag{2.9}$$

де ttl_{sell} – значення користувача для визначення загальної вартості угоди на продаж (за умовою значення $ttl_{sell} = 0.1$).

Значення загального об'єму активу в угоді (на продаж) користувача повинно задовольняти такій умові:

$$amount_{sell} > \frac{amount_{sell}}{100} * amnt_{sell}\,, \tag{2.10}$$

де $amnt_{sell}$ – значення користувача для визначення загального об'єму активу угоди на продаж (за умовою значення $amnt_{buy} = 0.0001$).

При виконанні програмою усіх зазначених вище умов ТС, АСКК зможе виконувати редагування угоди із мінімізацією ризиків для користувача.

2.4 Поведінка автоматизованої системи в процесі життєвого циклу

Для відображення поведінки АСКК у процесі всього ЖЦ, було спроєктовано діаграму станів [39, 48-49].

На рис. 2.3 зображено спроєктовану діаграму станів АСКК.

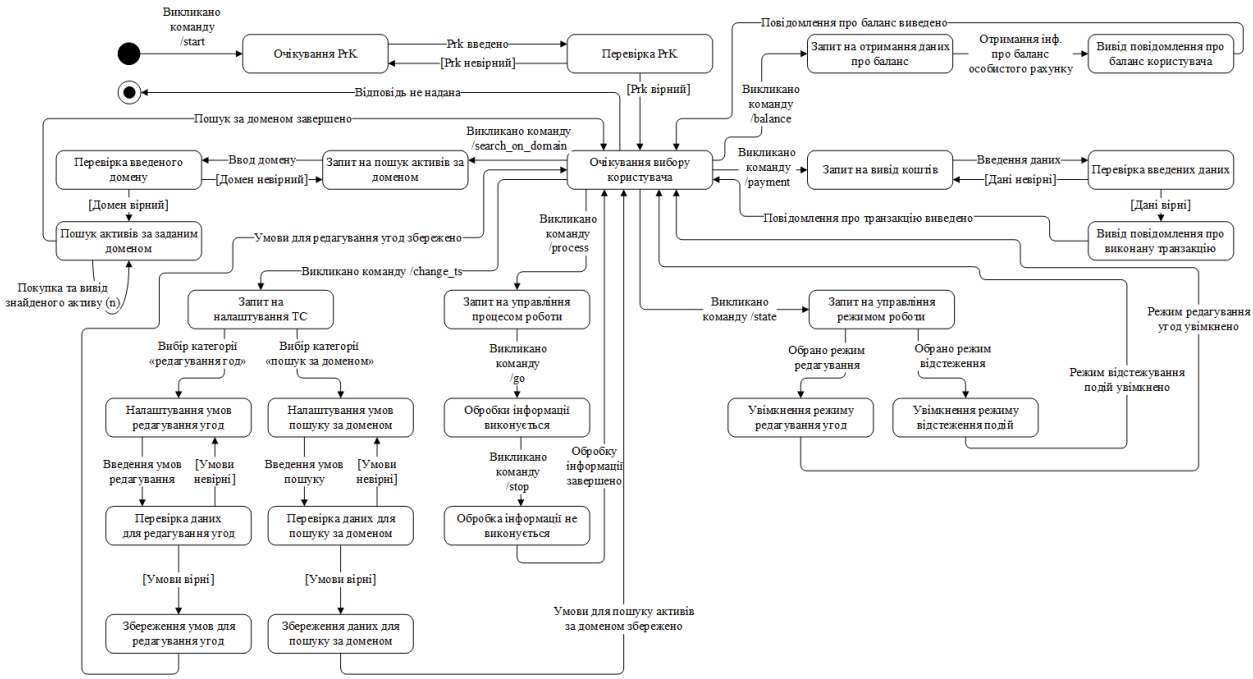

Рисунок 2.3 – Діаграма станів АСКК

Діаграма побудована у вигляді двадцяти чотирьох станів та двох псевдостанів (початок і кінець) зв‘язаних між собою переходами різного виду. До основних станів входять: «Очікування PrK», «Перевірка PrK», «Очікування вибору користувача», «Запит на тримання даних про баланс», «Вивід повідомлення про баланс користувача», «Запит на вивід коштів», «Перевірка введених даних», «Вивід повідомлення про виконану транзакцію», «Запит на управління режимом роботи», «Увімкнення режиму редагування угод», «Увімкнення режиму відстеження подій», «Запит на пошук активів за доменом», «Перевірка введеного домену», «Пошук активів за заданим

доменом», «Запит на налаштування ТС», «Налаштування умов редагування угод», «Налаштування умов пошуку за доменом», «Перевірка даних для редагування угод», «Перевірка даних для пошуку за доменом», «Збереження умов для редагування угод», «Збереження умов для пошуку за доменом», «Запит на управління процесом роботи», «Обробка інформації виконується», «Обробка інформації не виконується».

Далі описано процес переходів між станами системи:

– між псевдостаном початку роботи АСКК до стану «Очікування PrK» виконано перехід «Викликано команду /start»;

– від стану «Очікування PrK» до стану «Перевірка PrK» виконано перехід «PrK введено»;

– якщо виконано сторожову умову «[PrK невірний], то виконується нетригерний перехід назад до стану «Очікування PrK» від стану «Перевірка PrK»;

– якщо виконано сторожову умову «[PrK вірний], то виконується нетригерний перехід вперед до стану «Очікування вибору користувача» від стану «Перевірка PrK»;

– від стану «Очікування вибору користувача» може бути виконано перехід «Викликано команду /balance» до стану «Запит на отримання даних про баланс»;

– від стану «Запит на отримання даних про баланс» виконано перехід «Отримання інф. про баланс рахунку» до стану «Вивід повідомлення про баланс користувача»;

– після завершення стану «Вивід повідомлення про баланс користувача», виконується перехід «Повідомлення про баланс виведено» до попереднього стану «Очікування вибору користувача»;

– від стану «Очікування вибору користувача» може бути виконано до стану «Запит на вивід коштів» за допомогою переходу «Викликано команду /payment»;

– від стану «Запит на вивід коштів» виконано перехід «Введення даних» до стану «Перевірка введених даних»;

– якщо виконано сторожову умову «[Дані невірний], то виконується нетригерний перехід назад до стану «Запит на вивід коштів» від стану «Перевірка введених даних»;

– якщо виконано сторожову умову «[Дані вірний], то виконується нетригерний перехід вперед до стану «Вивід повідомлення про виконану транзакцію» від стану «Перевірка введених даних»;

– після завершення стану «Вивід повідомлення про виконану транзакцію», виконується перехід «Повідомлення про транзакцію виведено» до попереднього стану «Очікування вибору користувача»;

– від стану «Очікування вибору користувача» може бути виконано до стану «Запит на управління режимом роботи» за допомогою переходу «Викликано команду /state»;

– щоб перейти від стану «Запит на управління режимом роботи» до стану «Увімкнення режиму редагування угод» необхідно виконати перехід «Обрано режим редагування»;

– після завершення стану «Увімкнення режиму редагування угод», виконується перехід «Режим редагування угод увімкнено» до попереднього стану «Очікування вибору користувача»;

– щоб перейти від стану «Запит на управління режимом роботи» до стану «Увімкнення режиму відстеження подій» необхідно виконати перехід «Обрано режим відстеження»;

– після завершення стану «Увімкнення режиму відстеження подій», виконується перехід «Режим відстеження подій увімкнено» до попереднього стану «Очікування вибору користувача»;

– від стану «Очікування вибору користувача» може бути виконано до стану «Запит на пошук активів за доменом» за допомогою переходу «Викликано команду /search_on_domain»;

– від стану «Запит на пошук активів за доменом» виконано перехід «Введення домену» до стану «Перевірка введеного домену»;

– якщо виконано сторожову умову «[Домен невірний], то виконується нетригерний перехід назад до стану «Запит на пошук активів за доменом» від стану «Перевірка введеного домену»;

– якщо виконано сторожову умову «[Домен вірний], то виконується нетригерний перехід вперед до стану «Пошук активів за заданим доменом» від стану «Перевірка введеного домену»;

– стан «Пошук активів за заданим доменом» виконує перехід «Пошук та вивід знайденого активу (n)» у себе (n-раз) доки не буде виконано перевірку останнього активу в домені;

– після завершення стану «Пошук активів за заданим доменом», виконується перехід «Пошук за доменом завершено» до попереднього стану «Очікування вибору користувача»;

– від стану «Очікування вибору користувача» може бути виконано до стану «Запит на налаштування ТС» за допомогою переходу «Викликано команду /change_ts»;

– щоб перейти від стану «Запит на налаштування ТС» до стану «Налаштування умов редагування угод» необхідно виконати перехід «Вибір категорії “редагування угод”»;

– від стану «Налаштування умов редагування угод» виконано перехід «Введення умов редагування» до стану «Перевірка даних для редагування угод»;

– якщо виконано сторожову умову «[Умови невірні], то виконується нетригерний перехід назад до стану «Налаштування умов редагування угод» від стану «Перевірка даних для редагування угод»;

– якщо виконано сторожову умову «[Умови вірні], то виконується нетригерний перехід вперед до стану «Збереження умов для редагування угод» від стану «Перевірка даних для редагування угод»;

– після завершення стану «Збереження умов для редагування угод», виконується перехід «Умови для редагування угод збережено» до попереднього стану «Очікування вибору користувача»;

– щоб перейти від стану «Запит на налаштування ТС» до стану «Налаштування умов пошуку за доменом» необхідно виконати перехід «Вибір категорії “пошук за доменом”»;

– від стану «Налаштування умов пошуку за доменом» виконано перехід «Введення умов пошуку» до стану «Перевірка даних для пошуку за доменом»;

– якщо виконано сторожову умову «[Умови невірні], то виконується нетригерний перехід назад до стану «Налаштування умов пошуку за доменом» від стану «Перевірка даних для пошуку за доменом»;

– якщо виконано сторожову умову «[Умови вірні], то виконується нетригерний перехід вперед до стану «Збереження умов для пошуку за доменом» від стану «Перевірка даних для пошуку за доменом»;

– після завершення стану «Збереження умов для пошуку за доменом», виконується перехід «Умови для пошуку активів за доменом збережено» до попереднього стану «Очікування вибору користувача»;

– від стану «Очікування вибору користувача» може бути виконано до стану «Запит на управління процесом роботи» за допомогою переходу «Викликано команду /process»;

– для переходу від стану «Запит на управління процесом роботи» до стану «Обробка інформації виконується» виконується перехід «Викликано команду /go»;

– для переходу від стану «Обробка інформації виконується» до стану «Обробка інформації не виконується» виконується перехід «Викликано команду /stop»;

– після завершення стану «Обробка інформації не виконується», виконується перехід «Обробку інформації завершено» до попереднього стану «Очікування вибору користувача».

Виконавши частину роботи із проєктування системи, переходимо до розробки майбутньої АСКК.

2.5 Висновки за розділом

В результаті роботи над даним розділом було спроєктовано, відображено та описано:

- модель функціонування АСКК;
- логічну модель АСКК;
- поведінку АСКК в процесі ЖЦ;
- математичну модель ТС АСКК.

Дані складові проєктування, сформували чітку картину розробки АСКК. За допомогою них буде розроблено програмний код та засоби управління АСКК користувачем СМ «Telegram».

Спроєктовані моделі розділу пришвидшать процес розробки АСКК та зменшать можливість виникнення помилок при створенні зв'зків між класами, функціями та елементами інтерфейсу.

Наступним етапом роботи планується програмна реалізація АСКК.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ КЕРУВАННЯ КРИПТОАКТИВАМИ

3.1 Підключення до Telegram

Для функціонування системи на основі СМ Telegram необхідно створити окремий спеціальний акаунт – бот. За допомогою діалогу в чаті з ботом, користувач зможе виконувати зв’язок із розробленою системою.

Створення спеціального акаунту потребує певну послідовність дій. Спочатку необхідно виконати пошук користувача «BotFather» (БТ), що слугує засобом для створення бот-акаунтів (рис. 3.1).

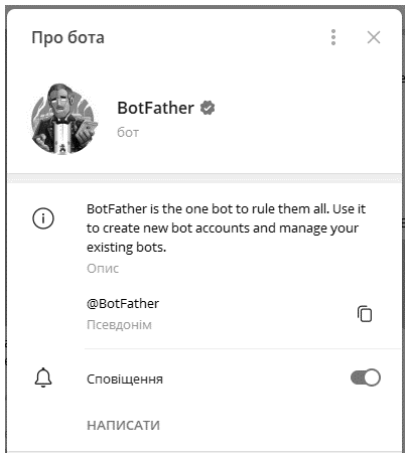

Рисунок 3.1 – Основна інформація про БТ

Далі необхідно перейти до чату з даним користувачем та розпочати діалог із ним, надіславши команду «/start». БТ надішле відповідь у вигляді переліку команд для роботи із ним.

Для створення особистого боту (ОБ), необхідна команда «/newbot». Після відправлення цієї команди, необхідно відповісти на початковий перелік питань БТ. Перше питання – бажана назва ОБ (Stellar Jobber). Отримавши відповідь, БТ запитає наступне – бажаний псевдонім (StellarJobber_bot) створюваного ОБ (рис. 3.2).

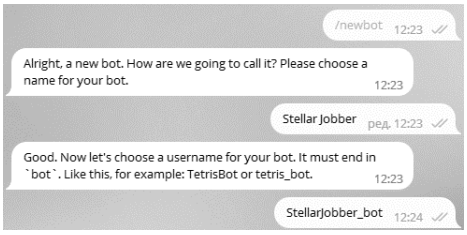

Рисунок 3.2 – Початковий діалог створення ОБ

Після завершення процесу створення БТ надішле повідомлення із спеціальним секретним токеном доступу Stellar Jobber. На рис. 3.3 показано загальний вигляд отриманого повідомлення із замальованим токеном.

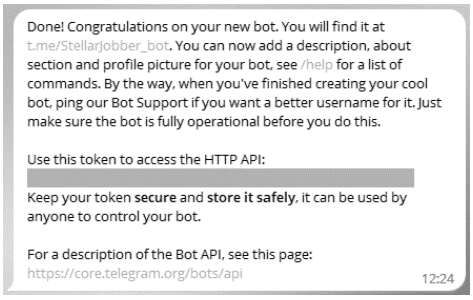

Рисунок 3.3 – Загальний вигляд повідомлення із особистим токеном

Отриманий токен використовується у програмному коді системи для зв’язку із Telegram. Зв’язок виконується за допомогою спеціальної бібліотеки «telebot» та її функції «TeleBot(‘token’)», де «token» – секретний токен Stellar Jobber. Для виконання зв’язуючої ролі між програмою та Stellar Jobber, створюється змінна, що зберігатиме дану функцію (рис. 3.4).

```
import telebot
bot = telebot.TeleBot("[redacted]")
```

Рисунок 3.4 – Бібліотека та змінна для зв’язку між програмою та Stellar Jobber

Далі було виконано наповнення основної інформації в Stellar Jobber за допомогою команд БТ:

– «/setdescription» – додає опис для Stellar Jobber;

– «/setuserpic» – додає зображення для Stellar Jobber;

– «/setabouttext» – додає інформацію про Stellar Jobber.

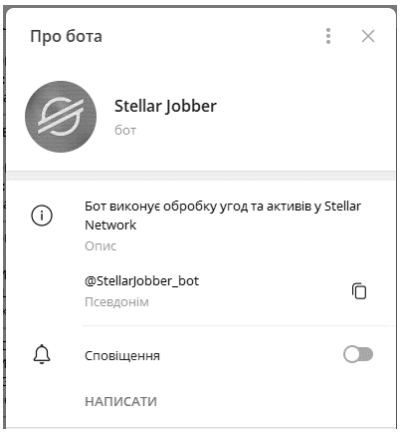

Рисунок 3.5 – Загальний вигляд вікна «Про бота»

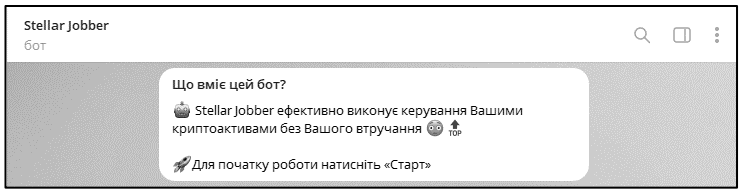

Рисунок 3.6 – Загальний вигляд початкового чату бота

Створивши ОБ Stellar Jobber та підключив до нього програмний код було виконано опис основних реалізованих класів та функцій.

3.2 Основні класи та функції

На основі сформованих задач із п. 1.4 було розроблено АСКК, що побудовано на:

– класах: «User», «StellarAccount», «OfferAccount», «StatusProgram», «MessageTemplate», «MarkupInterface»;

– функціях: «main_markup_menu», «start_save_keys», «go», «stop», «search_on_domain», «callback», «take_private_key», «take_domain_address», «add_new_order_from_domen», «domain_asset_check», «check_top_order», «connect_Stellar», «sell_orders_check», «update_sell_offer», «buy_orders_check», «chek_my_top_orders», «get_balance», «payment_asset», «change_sell_order»,

«get_ts_edit_buy_amount», «get_ts_edit_sell_amount», «update_buy_offer», «get_ts_edit_buy_profit», «get_ts_edit_sell_profit», «get_payment_asset», «get_payment_public_key», «get_ts_domain_profit», «change_buy_order», «get_payment_total», «get_ts_domain_total».

Далі було виконано опис вищезазначених складових програмного коду.

Збереження даних про користувача, під час виконання обробки даних програмним кодом, та застосування іншими класами та функціями, які потребують даних користувача, виконує клас «User» (дод. Б, п. Б. 1).

Клас «StellarAccount» (дод. Б, п. Б. 2) виконує завдання збереження даних про акаунт користувача в SN. До цього класу входять функції: «start_save_keys», «take_private_key».

Функція «start_save_keys» (дод. Б, п. Б. 3) відповідає за контакт із користувачем на початку роботи системи для отримання даних для авторизації гаманця SN.

Функція «take_private_key» (дод. Б, п. Б. 4) відповідає за сам процес проходження авторизації. Визначає правильність введених даних після чого зберігає (тимчасово) отримані дані до змінних класу «StellarAccount» для їх подальшого застосування програмою при обробці даних.

Інформація про угоди користувача: номер угоди, продавець, назва активу, код активу, та інші дані, зберігається у класі «OfferAccount» (дод. Б, п. Б. 5)

Основним класом, що відповідає за перемикання між класами, функціями та процесами обробки АСКК виступає «StatusProgram». До класу входить такий набір функцій: «go», «stop», «search_on_domain», «take_domain_address», «add_new_order_from_domen», «domain_asset_check», «check_top_order», «connect_Stellar», «sell_orders_check», «update_sell_offer», «buy_orders_check», «chek_my_top_orders», «get_balance», «payment_asset», «get_ts_edit_buy_amount», «get_ts_edit_sell_amount», «get_ts_edit_buy_profit», «get_ts_edit_sell_profit», «get_payment_asset», «get_payment_public_key»,

«get_ts_domain_profit», «change_buy_order», «get_payment_total», «get_ts_domain_total», «change_sell_order», «update_buy_offer».

Функція «go» (дод. Б, п. Б. 6) відповідає за початок обробки даних на гаманці користувача SN, залежно від заданих користувачем параметрів в АСКК.

«stop» (дод. Б, п. Б. 7) – функція, що зупиняє процес оброки даних.

Для виконання «пошуку за адресою домена» та отримання самої адреси слугує функція «search_on_domain» (дод. Б, п. Б. 8).

За перевірку наданого користувачем домену та передачі його на подальшу обробку, відповідає функція «take_domain_address» (дод. Б, п. Б. 9).

Створення нових угод на купівлю активів за наданим користувачем доменом, проведення перевірки цих активів (та угод на них) на наявність у гаманці користувача, відповідає функція «add_new_order_from_domen» (дод. Б, п. Б. 10).

«domain_asset_check» (дод. Б, п. Б. 11) – функція що виконує перевірку маркетплейсу актива, ти самим визначає його вигоду на подальшу купівлю.

Функція є однією із основних «connect_Stellar» (дод. Б, п. Б. 12) та відповідає за:

– отримання даних із гаманця користувача;

– сортування та перевірку угод за їх статусом;

– створення нових вигідних угод із наявних активів.

«check_top_order» (дод. Б, п. Б. 13) це функція, що визначає найвищу ціну маркетплейсу актива, в залежності від статусу угоди, що перевіряється.

За саму перевірку активу за статусом «продаж» відповідає функція «sell_orders_check» (дод. Б, п. Б. 14). Окрім цього дана функція також оновлює угоду, якщо було збільшено баланс перевіряємого активу у процесі існування даної угоди.

«update_sell_offer» (дод. Б, п. Б. 15) – це функція що виконує операцію зміни даних угоди продажу в SN, за наданими параметрами.

Функція «buy_orders_check» (дод. Б, п. Б. 16) виконує перевірку угоди активу за статусом «купівля».

Функція «chek_my_top_orders» (дод. Б, п. Б. 17) визначає наявність різниці (ціни та балансу) між угодою користувача та «топових» угод на маркетплейсі активу.

За процес зміни параметрів угоди та повідомлення про це користувача, відповідає функція «change_sell_order» (дод. Б, п. Б. 18).

Операція зміни даних угоди на купівлю у SN виконується за допомогою функції «update_buy_offer» (дод. Б, п. Б. 19).

Функція «change_buy_order» (дод. Б, п. Б. 20) відповідає за процес зміни даних угоди на купівлю активу. Повідомляє користувача про результат зміни угоди.

«get_balance» (дод. Б, п. Б. 21) – функція відповідає за отримання інформації про баланс гаманця користувача, в залежності від заданих параметрів.

За перевірку адреси отримувача (функція «Надіслати кошти») відповідає функція «get_payment_public_key» (дод. Б, п. Б. 22).

За отримання даних про суму відправлення (функція «Надіслати кошти») відповідає функція «get_payment_asset» (дод. Б, п. Б. 23).

Функція «get_payment_total» (дод. Б, п. Б. 24) запитує текстове повідомлення відправлення (функція «Надіслати кошти»).

«payment_asset» (дод. Б, п. Б. 25) – функція «Надіслати кошти». Виконує ряд запитів про адресу отримувача, суму та текст відправлення. Повідомляє про результат обробки транзакції.

Функція «get_ts_edit_domain_profit» (дод. Б, п. Б. 26) відповідає за редагування параметрів ТС на вигоду при виконання функції «пошук активів за доменом».

Функція «get_ts_edit_domain_total» (дод. Б, п. Б. 27) відповідає за редагування параметрів ТС, а саме об’єму купівлі активу при виконанні функції «пошук активів за доменом».

Функція «get_ts_edit_buy_profit» (дод. Б, п. Б. 28) відповідає за редагування параметру вигоди ТС функції «редагування угод» зі статусом «купівля».

Функція «get_ts_edit_buy_amount» (дод. Б, п. Б. 29) відповідає за редагування параметрів різниці об’єму ТС функції «редагування угод» зі статусом «купівля».

Функція «get_ts_edit_sell_profit» (дод. Б, п. Б. 30) відповідає за редагування параметру вигоди ТС функції «редагування угод» зі статусом «продаж».

Функція «get_ts_edit_sell_amount» (дод. Б, п.Б. 31) відповідає за редагування параметрів різниці об’єму ТС функції «редагування угод» зі статусом «продаж».

Клас «MessageTemplate» відповідає за збереження змінних-шаблонів при відправленні текстових повідомлень користувачу про результати обробки даних АСКК.

«MarkupInterface» (дод. Б, п. Б. 32) – це клас, що містить у собі складові інтерфейсу АСКК при роботі із користувачем. Має такі функції:

– «main_markup_men» (дод. Б, п. Б. 33) відповідає за відображення «головного меню» користувачу АСКК

– «callback» (дод. Б, п. Б. 34) відповідає за функціональність елементів «головного меню» при взаємодії з користувачем.

Таким чином було розроблено програмний код АСКК. Далі необхідно виконати опис реалізованих елементів взаємодії користувача із АСКК – інтерфейсу.

3.3 Зв’язок користувача з автоматизованою системою керування криптоактивами

Використовуючи засоби СМ Telegram, до ОБ Stellar Jobber було додано спеціальні елементі користувацького інтерфейсу для більш нагального та зручного використання.

Діалог користувача з Stellar Jobber починається із відображення блоку «Що вміє цей бот?» (рис. 3.7) із коротким описом основного завдання даного боту.

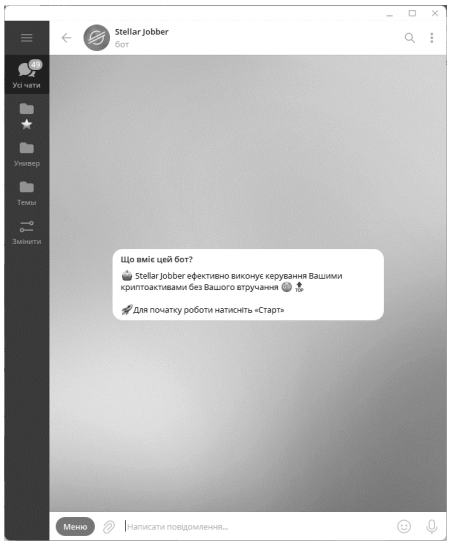

Рисунок 3.7 – Блок «Що вміє цей бот?»

Головне меню управління функціоналом Stellar Jobber складається із блоку тексту та кнопок. До тексту входить інформація про:

– адресу гаманця користувача;

– режим роботи Stellar Jobber;

– параметри ТС для редагування угод (на продаж та купівлю);

– параметри ТС для пошуку активів за доменом.

Блок кнопок створено за допомогою методу «InlineKeyboardMarkup» що складається з елементів методів «InlineKeyboardButton». Кожна кнопка відповідає за крему функцію Stellar Jobber. До блоку входять такі функції:

– «Особистий гаманець»;

– «Пошук за доменом»;

– «Зміна параметрів ТС»;

– «Обробка даних»;

– «Режим роботи»;

– «Вихід із акаунту».

На рис. 3.8 представлено загальний вигляд головного меню.

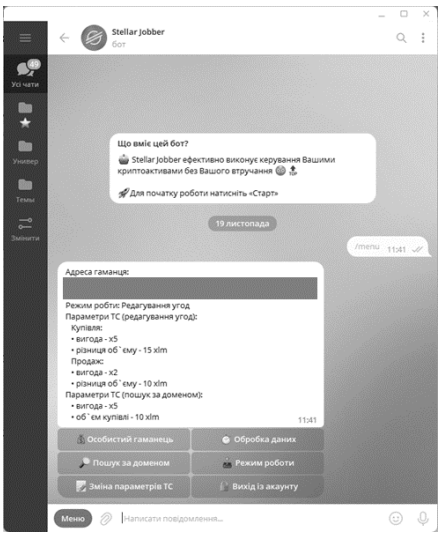

Рисунок 3.8 – «Головне меню» Stellar Jobber

Функція «Особистий гаманець» складається із трьох кнопок:

– функція «Баланс рахунку»;

– функція «Надіслати кошти»;

– кнопка повернення «До головного меню».

«Баланс рахунку» передбачає отримання відповіді на питання для здійснення роботи. Після цього надсилає результат виконання функції.

«Надіслати кошти» передбачає послідовне відправлення питань та отримання відповідних відповідей для отримання інформації про транзакцію. Після формування транзакції, відправляє її шаблон у вигляді текстового блоку:

– адреса отримувача;

– назва активу;

– сума активу;

– текст повідомлення.

Та кнопки регулювання подальшої роботи:

– «Так» (для підтвердження даних);

– «Редагувати» (для зміни необхідних даних).

На рис. 3.9 представлено загальний вигляд функції «Особистий гаманець» та її підфункцій: «Баланс рахунку» та «Надіслати кошти».

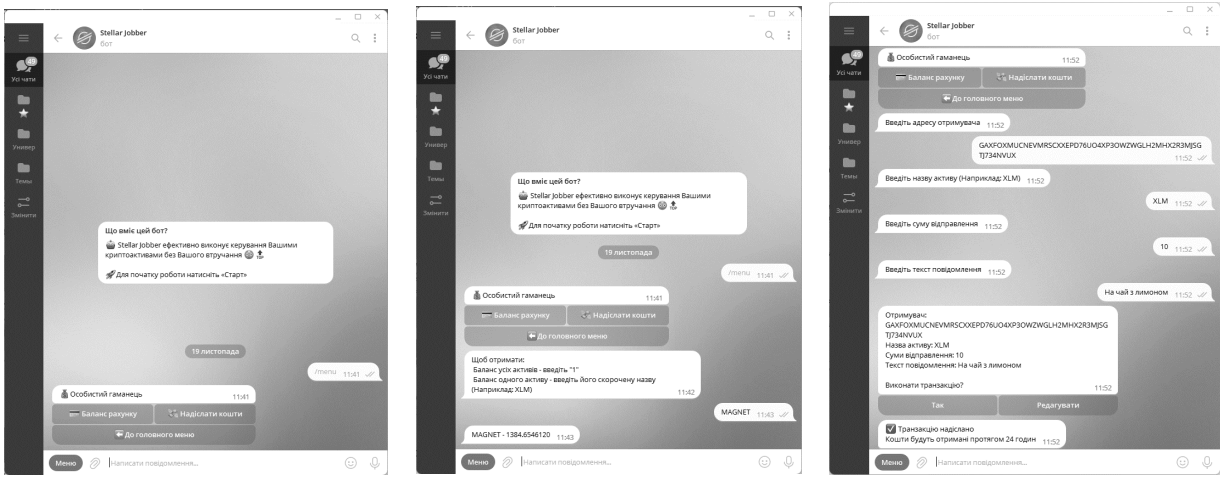

Рисунок 3.9 – Функції «Особистий гаманець», «Баланс рахунку» та «Надіслати кошти»

«Пошук за доменом» передбачає отримання адреси домена, у якому відбуватиметься пошук. Результат пошуку відображається у вигляді повідомлень. Пошук починається із повідомлення «пошук монет…». Далі будуть отримані повідомлення про створені угоди із знайденими активами. Наприкінці пошуку, користувач отримує повідомлення «Пошук активів завершено». На рис. 3.10 представлено процес роботи функції «Пошук за доменом».

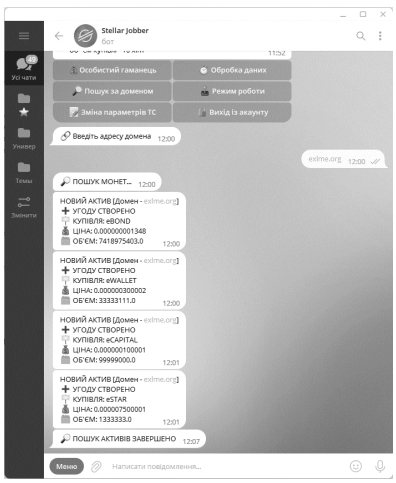

Рисунок 3.10 – Функція «Пошук за доменом»

«Режим роботи» налічує у собі три кнопки:

– «Редагування угод»;

– «Відстеження подій»;

– «До головного меню».

Вибір однієї із кнопок («Редагування угод» чи «Відстеження подій») буде відображено надалі у «Головне меню». На рис. 3.11 показано процес відображення роботи функції «Режим роботи».

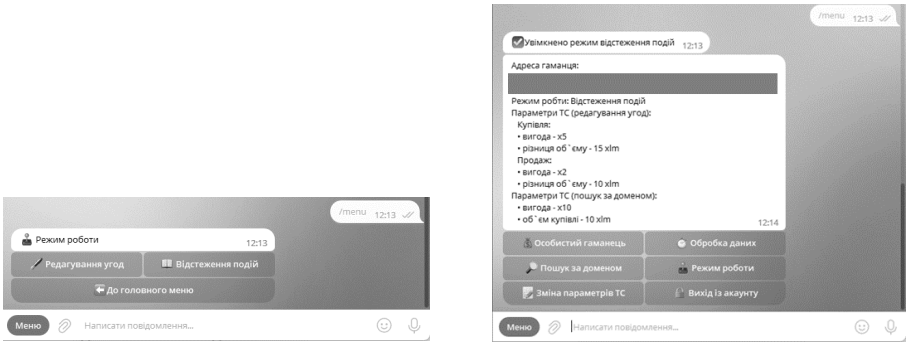

Рисунок 3.11 – Функція «Режим роботи»

«Зміна параметрів ТС» має три кнопки:

– «Параметри для пошуку за доменом»;

– «Параметри для редагування угод»;

– «До головного меню».

При натисканні на кнопку «Параметри для пошуку за доменом» чи «Параметри для редагування угод» буде отримано відповідний текстовий блок із інформацією про поточні параметри ТС та відповідні кнопки для редагування.

Для «Параметри для редагування угод» буде відображено такий текстовий блок:

– «Купівля»: вигода та різниця об’єму;

– «Продаж»: вигода та різниця об’єму.

Для «Параметри для редагування угод» буде відображено кнопки «Купівля» та «Продаж». Після натискання однієї із цих кнопок, буде

відображено повідомлення із обраним статусом угоди та вибором параметрів для редагування. Результат редагування буде відображено в усіх відповідних текстовий блоках. На рис. 3.12 показано процес відображення блоків функції «Зміна параметрів ТС».

Для «Параметри для пошуку за доменом» буде відображено текстовий блок із інформацією про вигоду та об’єм купівлі. Має аналогічний інтерфейс функції «Параметри для редагування угод».

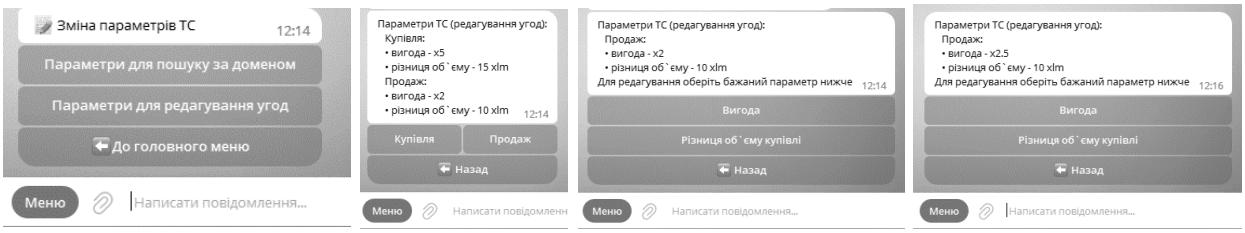

Рисунок 3.12 – Функція «Зміна параметрів ТС»

«Обробка даних» має дві кнопки: «Почати обробку» та «До головного меню». При натисканні на кнопку «Почати обробку» у користувача з’являється кнопка «Зупинити редагування» під полем вводу. Під час роботи користувач отримує повідомлення про результати обробки. При натисканні на кнопку «Зупинити редагування» обробка буде завершено, но підтвердить відповідне повідомлення. На рис. 3.13 представлено процес відображення функції «Обробка даних».

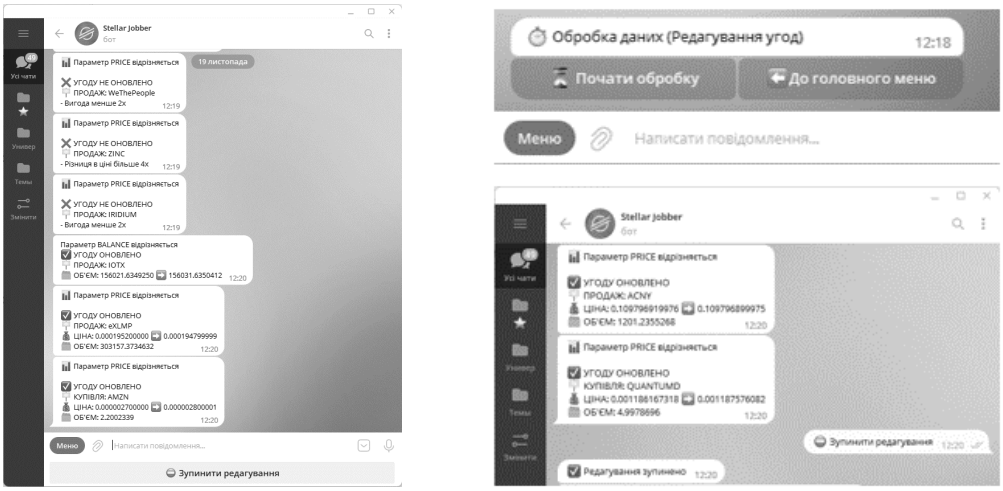

Рисунок 3.13 – Функція «Обробка даних»

Кнопка Авторизуватися слугує для початку процесу входу. Після його успішного проведення з’являється «Головне меню». На рис. 3.14 показано функцію «Авторизація».

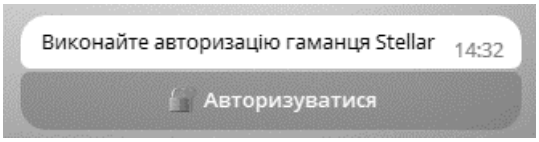

Рисунок 3.14 – Функція «Авторизація»

Розроблений інтерфейс Stellar Jobber було представлено на прикладі комп’ютерної версії додатку Telegram. На рис. 3.15 буде показано фрагменти інтерфейсу для мобільної версії.

Рисунок 3.15 – Загальний вигляд Stellar Jobber мобільної версії

Виконавши розробку Stellar Jobber, необхідно подбати про стабільну роботи системи. Для цього було прийнято рішення про встановлення АСКК на віддалене середовище.

3.4 Налаштування віддаленого середовища

По звершенню розробки АСКК, виконавши перевірку на її працездатність та відсутність помилок при роботі, було прийнято рішення про встановлення АСКК на віддалене середовище (ВС).

Робота АСКК на ВС забезпечить постійну та стабільну роботу Stellar Jobber для користувачів. Для цього необхідно було обрати ВС серед доступних у мережі Інтернет.

Наразі доступно безліч ВС із різними перевагами та недоліками. Для цієї роботи було обрано ВС – «pythonanywhere» [50]. Критеріями вибору були такі параметри:

– є безкоштовна версія;

– можливість управління проєктом в межах ВС;

– робота із проєктами на мові Python;

– можливість використання командної строки;

– робота із проєктами GitHub.

Спочатку було створено директорію для зберігання Stellar Jobber. Для цього у вкладці Files, у зоні Directories було створено файл «StellarJobberBot» (рис. 3.16).

Рисунок 3.16 – Створення директорії Stellar Jobber

Далі було виконано завантаження файлів Stellar Jobber за допомогою кнопки «Upload a file». На рис. 3.17 показано директорію «StellarJobberBot» після завантаження файлів.

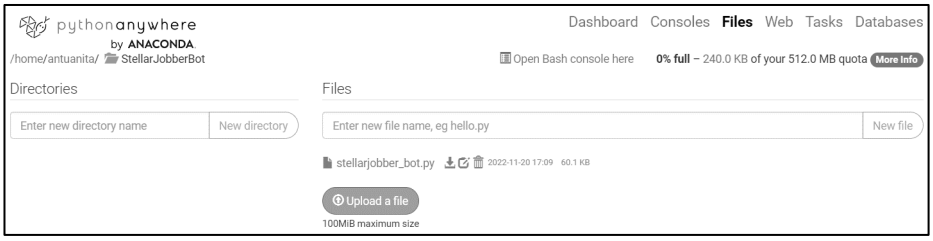

Рисунок 3.17 – Завантажений файлу в директорії

Далі за допомогою влаштованої bash-консолі встановлюємо бібліотеку «pytelegrambotapi» (рис. 3.18).

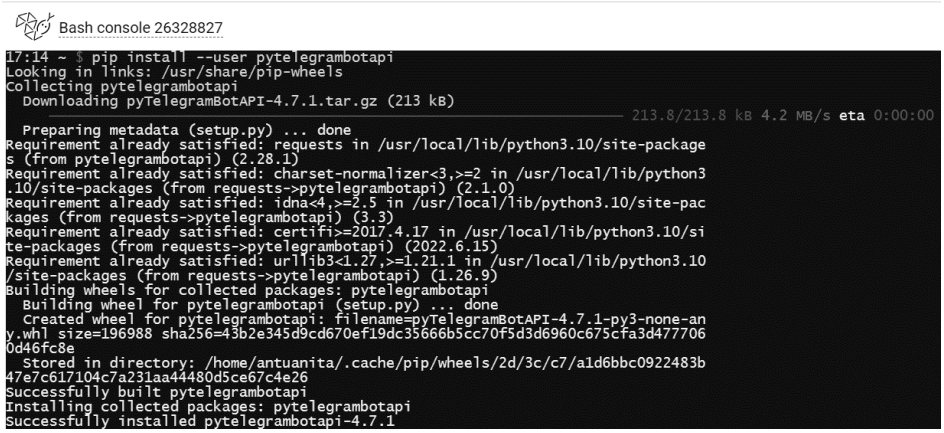

Рисунок 3.18 – Встановлення бібліотеки «pytelegrambotapi»

Наступним кроком стало створення Web app із версією Python 3.8 для керування Stellar Jobber у середовищі «pythonanywhere». Для створеного Web app було прописано адреси для директорій, де будуть зберігатися необхідні файли АСКК (рис. 3.19).

Рисунок 3.19 – Директорії файлів

Для остаточного запуску Stellar Jobber необхідно повернутися до вкладки Console (де раніше було встановлено бібліотеку). Далі було виконано наступні команди: «cd StellarJobber» та «python stellajobber_bot» послідовно.

Рисунок 3.20 – Процес запуску Stellar Jobber

Після проведення даних маніпуляцій, було виконано перевірку на працездатність Stellar Jobber через ВС «pythonanywhere» (рис. 3.21).

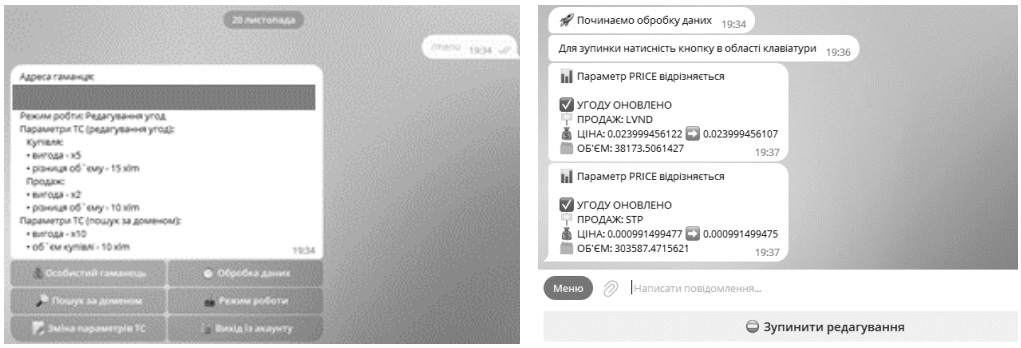

Рисунок 3.21 – Перевірка працездатності Stellar Jobber

Згідно отриманих результатів, маємо, що Stellar Jobber працює вправно. Функціонал виконується згідно поставленим задачам.

3.5 Висновки за розділом

Даний розділ присвячено опису процесу розробки АСКК. У ньому було освідчено таку інформацію як:

– процес налаштування ОБ в Telegram та підключення до нього програмного коду;

– опис розроблених класів та функцій АСКК;

– опис та екрані форми інтерфейсу, за допомогою якого виконується зв’язок користувача із АСКК;

– процес встановлення АСКК на середовище «pythonanywhere».

4 КОНТРОЛЬНА ПЕРЕВІРКА РОБОТИ АВТОМАТИЗОВАНОЇ СИСТЕМИ КЕРУВАННЯ КРИПТОАКТИВАМИ

4.1 Перевірка функціоналу розробленої автоматизованої системи керування криптоактивами

Для оприлюднення АСКК та надання до неї доступу користувачам, необхідно переконатися у справності роботи АСКК та відповідності наступним пунктам:

– АСКК виконана згідно поставленим на початку роботи задачам;

– АСКК безпомилково виконує заданий до неї функціонал, користувачем;

– зміни ЖО успішно надходять до SN.

Поставлені цілі АСКК було успішно досягнуто, а саме:

– користувача звільнено від самостійної обробки угод ЖО;

– доступ до АСКК виконується із будь-якого пристрою та ОС;

– за рахунок автономної роботи АСКК, користувач отримав більше вільного (від управління ЖО) часу;

– робота АСКК забезпечила безпомилкову обробку ЖО користувача.

Усі вимог до функціоналу та структури системи було виконано:

– АСКК виконує обробку даних у SN за допомогою Horizon API та SDK мовою Python;

– зв’язок користувача із АСКК виконується в СМ «Telegram»;

– вхід до акаунту користувача у АСКК виконується за допомогою приватного ключа, який закріплено за його особистим гаманцем SN;

– параметри усіх ТС (р.2 п. 2.3) представлено на головному меню АСКК та підлягають редагуванню зі сторони користувача, для створення своїх особистих налаштувань;

– процес обробки угод та відслідковування подій в ЖО керується за допомогою команд запуску та зупинки;

– виконано функцію пошуку вигідних активів за доменом, що регулюється окремою ТС (р.2 п. 2.3);

– процес редагування угод виконується безперервно, доки користувач самостійно не виконає процес зупинки;

– команди: редагування угод, редагування ТС («редагування угод» та «пошук за доменом»), перегляд балансу та переказ коштів, було додано до функціоналу АСКК;

– АСКК, за допомогою текстового повідомлення, результує про: зміну параметрів угоди в ЖО, події в ЖО, результат переказу активу отримувачу, результат пошуку вигідного активу в домені.

Як результат АСКК виконує своє призначення – відслідковує та регулює угоди користувача в ЖО; виконує пошук вигідних активів, перевірку балансу, переказ коштів; повідомляє користувача про виконані дії. Усі дії виконуються автономно за допомагаю засобів Horizon API, SDK мовою Python та СМ «Telegram». Можливість одночасного управління надається тільки одному гаманцю користувача.

Наступною перевіркою є безпомилкове виконання функцій АСКК при роботі користувача.

Авторизація. При запуску АСКК «Stellar Jobber», була натиснута кнопка «Старт». Результат:

– отримано повідомлення про необхідність виконання авторизації гаманця Stellar;

– будь-які команди блоку «Меню» недоступні;

– перевірка на відповідність до шаблону приватного ключа виконана;

– після завершення авторизації користувач отримав повідомлення у вигляді головного меню АСКК.

Особистий гаманець. Після переходу до головного меню, натиснута кнопка «Особистий гаманець». Результат: отримано доступ до вибору команд «Баланс рахунку» та «Надіслати кошти» та кнопки «До головного меню».

До головного меню. Дана кнопка присутня в кожній функції головного меню. Результат роботи кнопки: повернення повідомлення головного меню виконується швидко та безпомилково.

Баланс рахунку. Виконано перехід до функції. Результат:

– отримано повідомлення про введення даних;

– перевірка введених даних за шаблоном та для одного активу виконується вірно;

– баланс одного/всіх активів користувача відповідає дійсності.

Надіслати кошти. Перехід до функції виконано, в результаті маємо:

– послідовне отримання повідомлень про введення даних (адреса отримувача, назву активу, суму відправлення, текст повідомлення);

– отримано повідомлення-підтвердження із введеною інформацією;

– редагування даних виконується вірно;

– транзакція виконується успішно.

Пошук за доменом. Результат перевірки:

– перевірка на правильність введеної адреси домена успішна;

– пошук за доменом виконується;

– знайдений вигідний актив додано до ЖО користувача;

– отримано повідомлення про додавання нової угоди;

– після перевірки усіх активів домену отримано повідомлення про завершення пошуку.

Зміна параметрів ТС. Після переходу до головного меню, натиснута кнопка «Зміна параметрів ТС». Результат: отримано доступ до вибору команд «Параметри для пошуку за доменом» та «Параметри для редагування угод» та кнопки «До головного меню».

Параметри для пошуку за доменом. Результат:

– отримано доступ до редагування параметрів: вигода та об’єм купівлі;

– при натисканні обраного параметру отримано повідомлення про необхідні дані вводу;

– змінені параметри успішно відображено в усіх інших повідомленнях та змінено у системі.

Параметри для редагування угод. Результат: отримано доступ до редагування угод за параметрами: «Купівля» та «Продаж». Перевірка кожного параметру виконана та дала відмінний результат, аналогічний перевірці функції «Параметри для пошуку за доменом».

Режим роботи. Результат перевірки:

– перехід до даної функції надає два параметри: «Редагування угод» та «Відстеження подій»;

– вибір відповідного параметру успішно змінює режим роботи АСКК;

– змінений параметр успішно відображено в усіх інших повідомленнях.

Обробка даних. Перевірку виконано за двома режимами роботи АСКК: «Редагування угод» та «Відстеження подій». За режимом «Редагування угод» отримано такі результати:

– з’явилася кнопка «Зупинити редагування»

– отримано повідомлення про початок обробки;

– отримано повідомлення про додавання нового активу;

– додано новий актив;

– отримано повідомлення про зміну параметрів угоди;

– угоду відредаговано;

– натискання кнопки «Зупинити редагування» завершило виконання редагування угод, кнопка зникла, з’явилося відповідне повідомлення про зупинку та отримано повідомлення головного меню.

За режимом «Відстеження подій» отримано такі результати:

– з’явилася кнопка «Зупинити відстеження»

– отримано повідомлення про початок відстеження;

– отримано повідомлення про подію в ЖО (виконано продаж активу);

– натискання кнопки «Зупинити відстеження» завершило виконання відстеження подій, кнопка зникла, з’явилося відповідне повідомлення про завершення, з’явилося повідомлення «Головне меню».

Вихід із акаунту. Для цього було виконано перехід до головного меню Stellar Jobber та натиснута кнопка «Вихід із акаунту». Результат: вихід із гаманця виконано успішно; будь-які виклики команд «Меню» потребують процесу авторизації.

При виконанні таких функцій як «Редагування угод» та «Пошук за доменом», необхідно було переконатися у тому, що усі внесені зміни до ЖО успішно виконуються в SN.

Для перевірки внесених змін функцією «Пошук за доменом» спочатку було виконано перевірку поточних угод. Далі виконано запуск функції «Пошук за доменом», після того як АСКК знайшла вигідний актив, виконано перевірку ЖО (рис. 4.1).

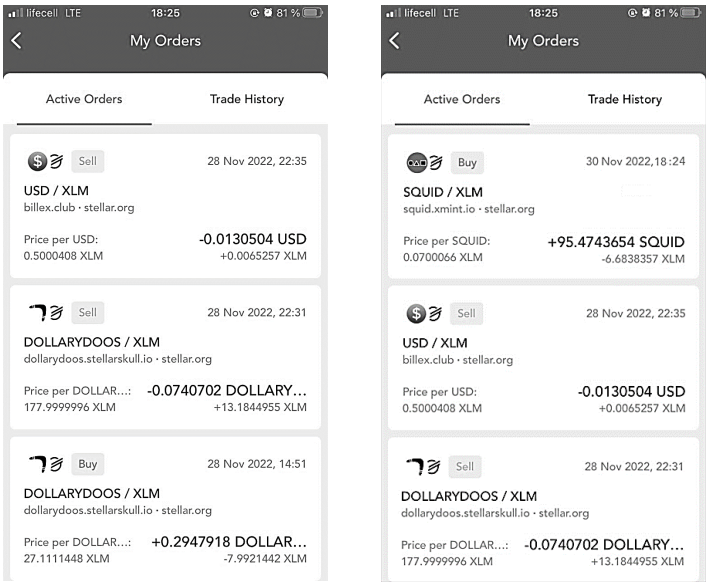

Рисунок 4.1 – Загальний вигляд ЖО до/після виконання функції «Пошук за доменом»

Для перевірки внесених змін функцією «Редагування угод» спочатку було виконано перевірку поточних угод, які підлягають редагуванню. Далі виконано запуск функції «Редагування угод», після того як АСКК знайшла вигідний актив, виконано перевірку ЖО (рис. 4.2).

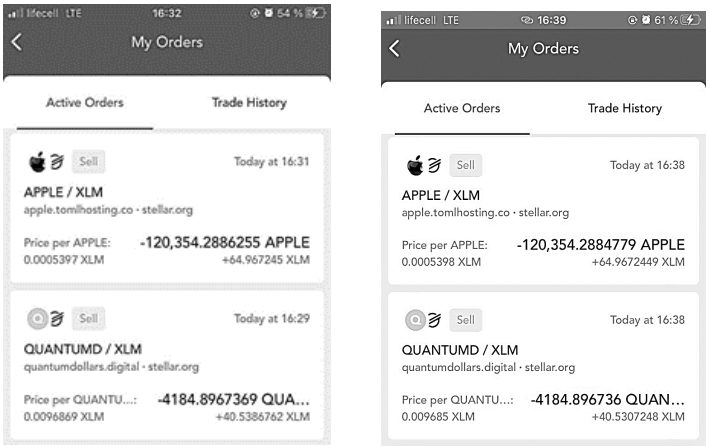

Рисунок 4.2 – Загальний вигляд ЖО до/після виконання функції «Редагування угод»

Як результат – функції «Пошук за доменом» та «Редагування угод» виконується вірно, параметри угод редагуються, а нові угоди додаються в SN.

Виконавши перевірку АСКК «Stellar Jobber» було отримано успішні результати щодо безпомилкового виконання її функціонування, обробки даних та відповідності раніше сформованих задач на розробку. Виходячи з цього, АСКК «Stellar Jobber» можна оприлюднювати до користування.

4.2 Послідовність дій при роботі з автоматизованою системою керування криптоактивами

Для початку роботи із АСКК «Stellar Jobber», користувачу необхідно додати чат-бот до своїх діалогів в СМ «Telegram». Для цього, у полі «Пошук», вводимо псевдонім «StellarJobber_bot» (рис. 4.3) або «Stellar Jobber».

Далі необхідно перейти до чату із «Stellar Jobber» та натиснути кнопку «Старт» (рис.4.4).

Після цього, до АСКК автоматично буде надіслано команду «/start», яка в свою чергу почне процес авторизації користувача.

Рисунок 4.3 – Пошук АСКК «Stellar Jobber»

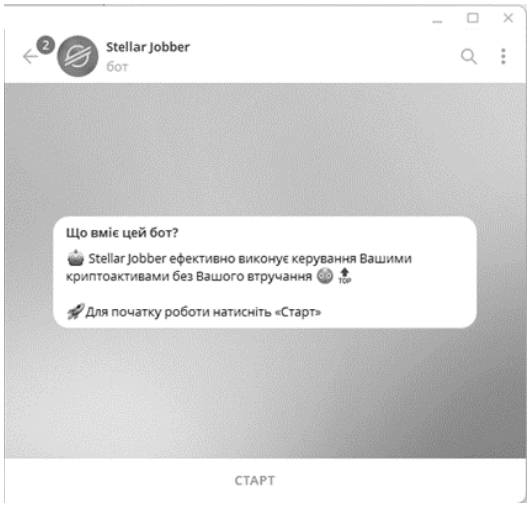

Рисунок 4.4 – Початкова сторінка часу із АСКК «Stellar Jobber»

Для проходження авторизації, користувачу необхідно натиснути кнопку «Авторизуватися» під отриманим повідомленням (дод. В, рис. В.1). Після цього, повідомлення зміниться на напис «Введіть секретний ключ» (дод. В, рис. В.2), а користувачу необхідно буде надіслати свій PrK у діалог із АСКК. При успішній авторизації, користувач отримає повідомлення із головним меню (ГМ) АСКК (дод. В, рис. В.3).

У випадку, якщо PrK було введено невірно, користувач отримає відповідне повідомлення та нову спробу авторизації (рис.4.5).

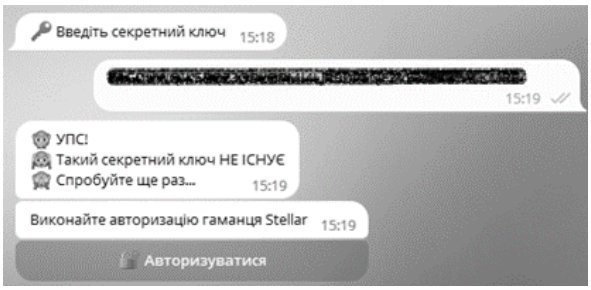

Рисунок 4.5 – Повідомлення про невірно введений PrK

У ГМ «Stellar Jobber» відображені: адреса гаманця, основна інформацію про налаштування АСКК (режим роботи, налаштування ТС) та шість управляючих кнопок (рис. 4.6).

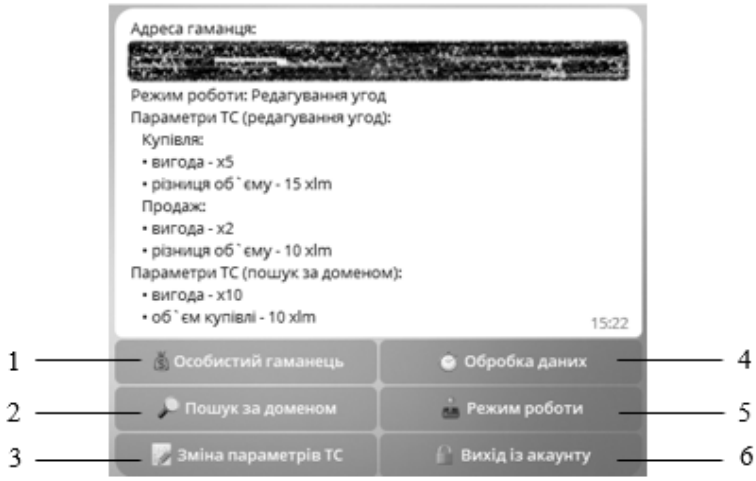

Рисунок 4.6 – Головне меню АСКК «Stellar Jobber»

Для роботи із гаманцем користувача, необхідно натиснути кнопку «1». ГМ перетвориться на меню із трьома кнопками: «Баланс рахунку», «Надіслати кошти» та «До головного меню» (дод. В, рис. В.4).

Кнопка «Баланс рахунку» слугує для отримання балансу на рахунку користувача. Для отримання цієї інформації необхідно вказати відповідний параметр. Для отримання балансу усіх активів, необхідно надіслати «1» (рис. 4.7), а для отримання балансу одного активу – його скорочену назву (рис. 4.8).

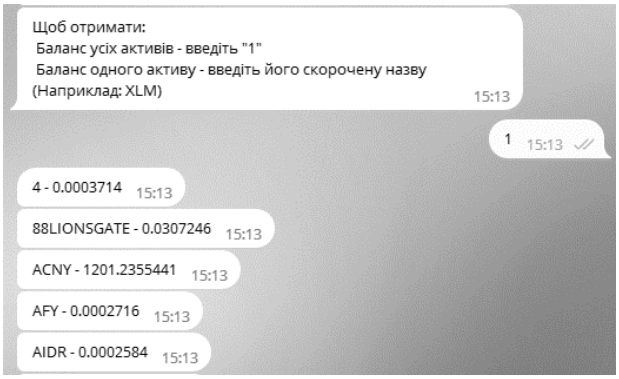

Рисунок 4.7 – Баланс кожного активу

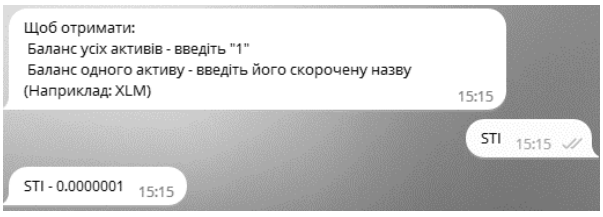

Рисунок 4.8 – Баланс одного активу («STI»)

Кнопка «Надіслати кошти» слугує для переказу активу між користувачами. Для виконання переказу необхідно послідовно відповісти на запитання (як показано на рис. 4.9).

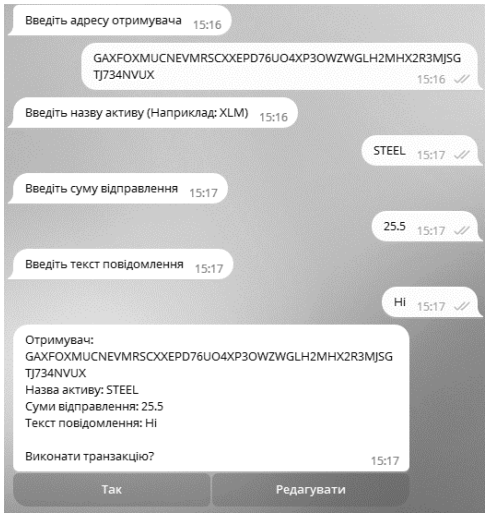

Рисунок 4.9 – Процес створення транзакції

Після формування транзакції, необхідно перевірити та підтвердити введені дані. Для підтвердження транзакції, необхідно натиснути кнопку «Так». Для редагування – «Редагування». Процес редагування даних відбувається аналогічно створенню транзакції (рис.4.9).

При підтвердженні транзакції буде отримано повідомлення (рис. 4.10).

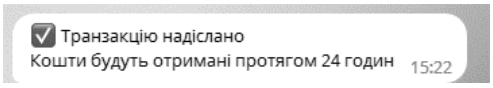

Рисунок 4.10 – Повідомлення про успішну транзакцію

Кнопка «До головного меню» виконує повернення до ГМ АСКК.

Щоб почати процес пошуку вигідних активів, необхідно натиснути на кнопку «2». Після цього, до АСКК треба надіслати адресу бажаного домена. Якщо домен вказано вірно, АСКК почне пошук. Результатом пошуку та створення нових угод будуть повідомлення (рис. 4.11).

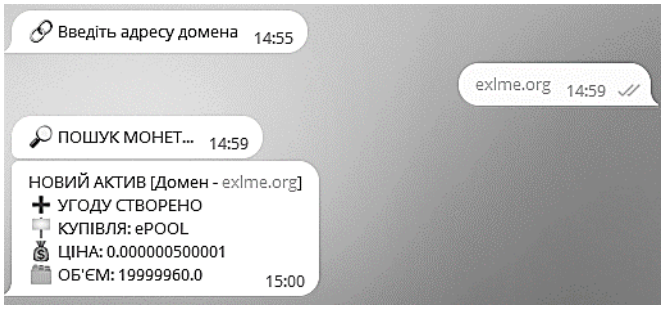

Рисунок 4.11 – Результат пошуку активів

Для зупинки пошуку необхідно натиснути кнопку «Зупинити пошук» (рис. 4.12) або дочекатися автоматичного завершення пошуку за усіма активами домену.

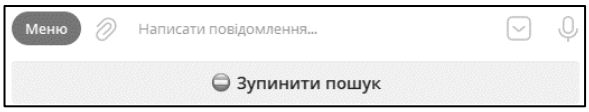

Рисунок 4.12 – Кнопка зупинки пошуку за доменом

Для зміни параметрів ТС АСКК необхідно натиснути кнопку «3», після чого користувач отримає меню із інформацією про поточні параметри ТС та три кнопки управління (дод. В, рис. В.5): «Параметри для пошуку за доменом», «Параметри для редагування угод», «До головного меню».

Кнопка «Параметри для пошуку за доменом» – виконує перехід до меню налаштування параметрів пошуку вигідних активів за доменом. Для цього, користувач отримує меню із наступними кнопками (дод. В, рис. В.6):

– «Вигода» відповідає за налаштування параметру вигоди, що бажає отримати користувач, від активу;

– «Об’єм купівлі» відповідає за максимальну суму, що бажає витратити користувач;

– «Назад» – повертає до початкового повідомлення із вибором параметрів ТС.

Для того, щоб встановити бажаний параметр, необхідно натиснути на відповідну кнопку та вказати число (рис. 4.13).

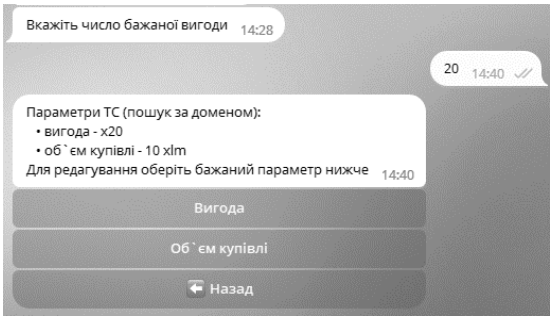

Рисунок 4.13 – Зміна параметру ТС

Кнопка «Параметри для редагування угод». Дане меню налаштування має три кнопки: «Купівля», «Продаж» та «Назад» (дод. В. рис. В.7).

Кнопка «Купівля» відповідає за редагування параметрів для угод на купівлю. Кнопка «Продаж» – на продаж. Подальше виконання дій аналогічне до меню редагування «Параметри ТС (пошук за доменом)».

Кнопка «До головного меню» – повертає до ГМ АСКК.

Для того, щоб система почала обробку даних (в залежності від встановленого режиму роботи), необхідно натиснути кнопку «4». Після цього ГМ зміниться на меню із двома кнопками (дод. В, рис. В.8):

– кнопка «Почати обробку» запускає процес оброблення даних гаманця користувача, в залежності від встановленого режиму роботи;

– кнопка «До головного меню» – виконує перехід до ГМ АСКК.

При натисканні кнопки «Почати обробку» з’являється кнопка «Зупинка редагування»/«Зупинка відстеження» (в залежності від параметру «Режим роботи») (рис. 4.14).

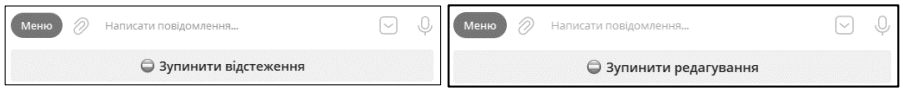

Рисунок 4.14 – Кнопки зупинки обробки даних

Якщо режим роботи встановлений на «Редагування угод», то АСКК буде виконувати обробку ЖО користувача. До обробки входять процеси: створення вигідних угод, редагування параметрів вже існуючих угод, видалення

невигідних угод. Результатом обробки ЖО будуть отримані повідомлення про виконану дію АСКК над угодою (рис. 4.15).

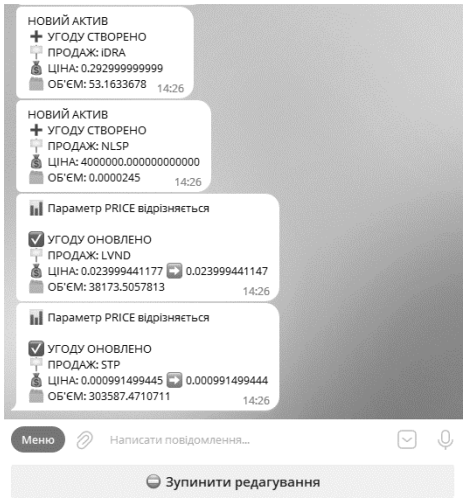

Рисунок 4.15 – Результати обробки ЖО

Якщо режим роботи встановлений на «Відстеження подій», то АСКК буде відслідковувати стан гаманця (продаж активу, купівля активу, результат транзакцій між гаманцями та інші) в реальному часі. Результатом відстеження будуть отримані повідомлення про поточну подію (рис. 4.16).

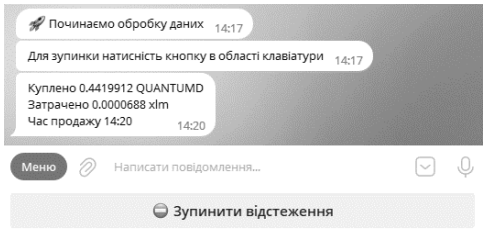

Рисунок 4.16 – Результати відстеження подій

Процес обробки ЖО виконується доти, доки користувач не натисне кнопку «Зупинка редагування/відстеження». Після зупинки обробки, буде отримано ГМ АСКК (дод. В, рис. В.9).

Для того щоб змінити параметри режиму роботи АСКК необхідно натиснути кнопку «5». ГМ змінить загальний вигляд на меню із трьома кнопками (дод. В, рис. В.10) :

– кнопка «Редагування угод» – застосує режим редагування угод, в якому АСКК буде виконувати тільки керування угодами (додавання, редагування, видалення) в ЖО користувача;

– кнопка «Відстеження подій» – застосує режим відстеження подій, в якому АСКК буде надсилати повідомлення до чату про події, що відбуваються всередині гаманця користувача;

– кнопка «До головного меню» – повертає ГМ АСКК.

Встановлений режим буде відображено до повідомлення та ГМ користувача (рис. 4.17).

Рисунок 4.17 – Зміна режиму роботи АСКК

Для зміни гаманця або виходу із АСКК, необхідно натиснути кнопку «6». Вихід із АСКК виконається автоматично, а користувач отримає повідомлення (рис. 4.18).

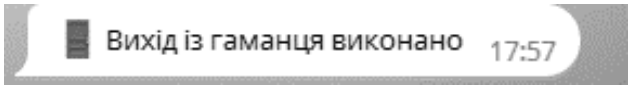

Рисунок 4.18 – Вихід із АСКК

Для повторного входу в АСКК необхідно виконати процес авторизації, відправивши команду «/start».

4.3 Дослідження ефективності автоматизованої системи керування криптоактивами

Після проведення перевірки АСКК, необхідно визначити, чи буде така система ефективною для користувача. Оскільки основними цілями АСКК є

збільшення прибутку та зменшення часу користувача на самостійну перевірку та редагування даних, було виконано дослідження за такими направленнями:

– ефективність часу роботи функції «редагування угод»;

– ефективність часу роботи функції «пошук за доменом»;

– ефективність росту прибутку користувача.

До проведення досліджень було долучено дві групи користувачів:

– I група – користувачі без системи;

– II група – користувачі з системою.

Нижче детально описано кожне направлення виконаного дослідження [32].

Ефективність часу роботи функції «редагування угод».

Ціль даного направлення – визначити середній час, який затрачає користувач на виконання ручної обробки своїх угод між користувачем, який для обробки буде використовувати АСКК.

Щоб визначити затрачений час, необхідно було розбити процес «редагування угод» на декілька етапів:

– час процес перегляду угоди (до цього входить: перегляд маркетплейсу активу та порівняння отриманих даних із поточними даними угоди;), T_{view};

– час процесу редагування (зміна необхідних параметрів угоди, за умови що буде відредаговано принаймні один параметр), T_{edit}.

I група користувачів, за допомогою таймеру, відслідковувала затрачений час на виконання вищезазначених етапів редагування. Отримавши результати кожного користувача, було розраховано середнє значення (СЗ) часу користувачів I групи.

Для II групи користувачів, було задіяно таймер у середині програмного коду. Отримавши результати кожного користувача, було розраховано СЗ часу користувачів II групи.

Після того, як СЗ двох груп сформовано, було визначено загальний час (T), що витрачають користувачі на обробку однієї угоди. Таким чином, маючи значення загального часу, часу перегляді та часу редагування для однієї угоди,

також було виконано розрахунок для редагування кількістю 10, 20, 40, 60 та 100 угод. Отримані результати зведено до двох таблиць – табл. 4.1 та табл. 4.2.

Таблиця 4.1 – Результати дослідження I групи «редагування угод»

№	$T_{I,view}$	$T_{I,edit}$	T_I, c	T_I, хв
1	6	15	21	0,35
10	60	150	210	3,50
20	120	300	420	7,00
40	240	600	840	14,00
60	360	900	1260	21,00
100	600	1500	2100	35,00

Таблиця 4.2 –Результати дослідження II групи «редагування угод»

№	$T_{II,view}$	$T_{II,edit}$	T_{II}, c	T_{II}, хв
1	0,28	7	7,28	0,12
10	2,8	70	72,8	1,21
20	5,6	140	145,6	2,43
40	11,2	280	291,2	4,85
60	16,8	420	436,8	7,28
100	28	700	728	12,13

За допомогою наведених результатів було побудовано результуючий графік (рис. 4.19), аби наявно продемонструвати різницю між отриманими даними.

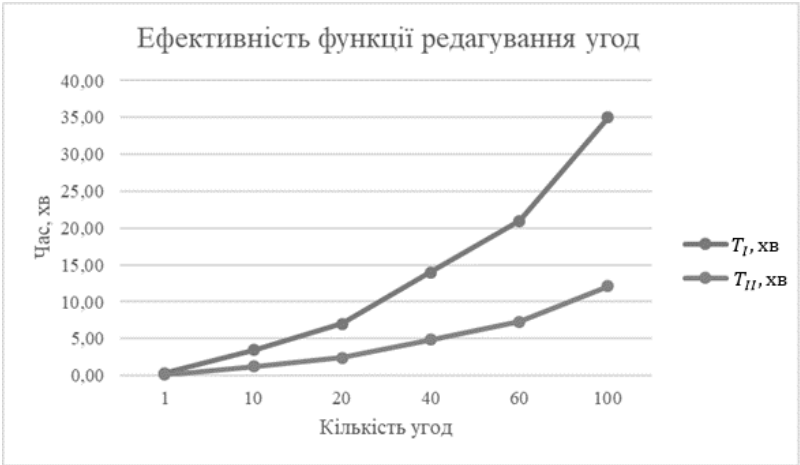

Рисунок 4.19 – Графік даних дослідження часу «редагування угод»

Судячи із отриманого графіку, та даних у таблицях маємо такий висновок – I група користувачів, що виконувала редагування особистих угод самостійно, впоралась менш ефективно ніж II група, що користувалася АСКК, в ≈ 2.9 рази.

Ефективність часу роботи функції «пошук за доменом».

Ціль даного направлення – визначити середній час, який затрачає користувач на виконання пошуку вигідних активів за доменом між користувачем, який для пошуку буде використовувати АСКК.

Щоб визначити затрачений час, процес «пошук за доменом» було розбито на декілька етапів:

– час процес перегляду активу, T_{view};

– час на процес додавання (налаштування необхідних параметрів угоди), T_{add}.

I група користувачів розрахувала затрачений час на виконання вищезазначених етапів редагування. На основі отриманих результатів було отримано СЗ часу користувачів I групи.

Для II групи користувачів, було розраховано час у середині програмного коду. Результати кожного користувача, було задіяно до визначення СЗ часу користувачів II групи.

Сформувавши СЗ часу вищенаведених етапів двох груп, було визначено загальний час (T), що витрачають користувачі на перегляд одного активу. Таким чином було виконано розрахунок перегляду в кількості 10, 20, 40, 60 та 100 активів. Отримані результати зведено до двох табл. 4.3 та табл.4.4.

Таблиця 4.3 – Результати дослідження I групи «пошук за доменом»

№	$T_{I,view}$	$T_{I,add}$	T_I, c	T_I, хв
1	6	15	21	0,35
10	60	150	210	3,50
20	120	300	420	7,00
40	240	600	840	14,00
60	360	900	1260	21,00
100	600	1500	2100	35,00

Таблиця 4.4 –Результати дослідження II групи «пошук за доменом»

№	$T_{II,view}$	$T_{II,add}$	T_{II}, c	T_{II}, хв
1	0,28	7	7,28	0,12
10	2,8	70	72,8	1,21
20	5,6	140	145,6	2,43
40	11,2	280	291,2	4,85
60	16,8	420	436,8	7,28
100	28	700	728	12,13

На основі наведених результатів було отримано графік (рис. 4.20), який наявно показує різницю між отриманими даними.

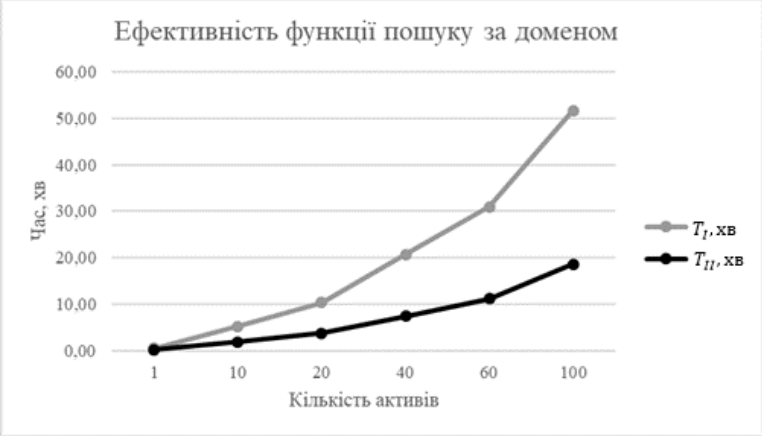

Рисунок 4.20 – Графік даних дослідження часу «пошук за доменом»

В результаті отриманих даних таблиць та графіку, маємо такий висновок – I група користувачів, що виконувала пошук вигідних активів власноруч, впоралась менш ефективно ніж II група, що користувалася АСКК, в ≈ 2.8 рази.

Ефективність росту прибутку користувача.

Ціль даного направлення – визначити ефективність зростання прибутку користувачів, що регулюють свій гаманець без/з системою АСКК.

Для виконання цього дослідження, для користувачів обох груп необхідно було:

– забезпечити наявність на гаманці по 300 xlm;

– регулювати гаманець кожен день на протязі тридцяти одного дня.

Від I групи користувачів, потребувалося виконання наступних вимог:

– власноруч виконати пошук вигідних активів на суму 200 xlm;

– кожен день, виконувати редагування угод на шаманці, принаймні три рази на день;

– кожного дня фіксувати прибуток із продажу активів.

II група користувачів, потребувалося виконання наступних вимог:

– за допомогою функції «пошук за доменом» АСКК виконати пошук вигідних активів на суму 200 xlm;

– кожен день виконувати запуск функції «редагування» АСКК, принаймні на період із 9:00 до 18:00;

– кожен день виконувати фіксацію отриманого прибутку.

Після місяця спостережень, було отримано об'єм даних, за допомогою яких було сформовано СЗ прибутку користувачів на кожний тридцять один день дослідження.

Дані I групи користувачів було зведено до табл. 4.5, II групи – до табл. 4.6.

СЗ затрачених xlm на пошук активів:

– I групи складає – 200,15 xlm;

– II групи – 200,25 xlm.

Як результат маємо, що СЗ заробітку I групи складає ≈ 235.9 xlm, а II групи – ≈ 606.3 xlm.

Таблиця 4.5 – Дані дослідження прибутку I групи

День	1	2	3	4	5	6	7	8	9	10	
Прибуток	13,8	11,7	14,5	10,2	7,3	1,1	3,7	59,7	12,3	14,1	
Баланс	113,6	125,3	139,8	150,0	157,3	158,4	162,1	221,8	234,1	248,2	
День	11	12	13	14	15	16	17	18	19	20	
Прибуток	2,5	4,6	13,3	4,0	9,8	87,8	1,7	6,4	14,0	1,4	
Баланс	250,7	255,3	268,6	272,6	282,4	370,2	371,9	378,3	392,3	393,7	
День	21	22	23	24	25	26	27	28	29	30	31
Прибуток	13,4	23,8	3,3	1,3	8,9	13,4	52,1	15,3	12,8	2,0	0,1
Баланс	407,1	430,8	434,1	435,3	444,3	457,6	509,7	525,0	537,8	539,8	539,9
Загальник прибуток											≈440

Таблиця 4.6 – Дані дослідження прибутку II групи

День	1	2	3	4	5	6	7	8	9	10	
Прибуток	8,8	8,1	3,6	11,0	19,9	3,5	78,4	110,7	3,9	15,0	
Баланс	108,6	116,7	120,3	131,3	151,2	154,7	233,1	343,8	347,7	362,7	
День	11	12	13	14	15	16	17	18	19	20	
Прибуток	12,3	13,4	2,0	2,7	91,6	8,7	13,4	6,9	5,7	91,3	
Баланс	375,0	388,4	390,4	393,1	484,7	493,4	506,7	513,6	519,3	610,6	
День	21	22	23	24	25	26	27	28	29	30	31
Прибуток	7,5	6,8	17,0	2,6	113,1	32,5	79,1	2,9	15,2	6,5	12,4
Баланс	618,1	625,0	641,9	644,5	757,6	790,2	869,2	872,1	887,3	893,9	906,2
Загальник прибуток											≈806,5

Отримані дані було відображено на загальний графік ефективності зростання прибутку 4.21.

Рисунок 4.21 – Графік даних дослідження ефективності зростання прибутку

Виходячи із даних, що видно на графіку та зазначено у таблицях, випливає висновок – II група, що використовувала АСКК отримала ріст прибутку в ≈ 3 рази, у той час як I група – в ≈1.8 рази.

Як результат, II група, що регулювала свої угоди за допомогою АСКК, отримала в 2.5 рази більше прибуток ніж I група, що управляла своїми угодами самостійно.

Виходячи із отриманих результатів досліджень за визначенням ефективності затраченого часу на пошук активів, на редагування угод, та ефективності прибутку. Було визначено, що використання АСКК є беззаперечно ефективним

4.4 Висновки за розділом

У даному розділі було виконано остаточну перевірку розробленої АСКК. Для цього було:

– виконано перевірку функціоналу АСКК згідно поставлених на початку роботи задач;

– описано послідовність дій при використанні АСКК користувачем;

– виконано дослідження на ефективність роботи.

Як результат, АСКК задовольняє сформованим задачам та показала свою ефективність при дослідженні між користувачами з системою та без системи.

ВИСНОВКИ

Виконання даної роботи було спрямовано на досягнення поставленої, на початку, мети у створенні засобу, що забезпечить керування особистими криптоактивами користувача SN із максимальною ефективністю, за рахунок його автоматизування.

Для досягнення мети було проведено великий об’єм роботи, що включав у себе:

– аналіз проблем, що виникають в результаті керування криптоактивами (КК), та методи їх вирішення;

– пошук та дослідження вже існуючих систем, які теоретично можливі виступити як рішення наявних проблеми із КК;

– збір засобів для реалізації системи, та вирішення зазначених проблеми із КК;

– постановку задач на розробку АСКК;

– проєктування таких складових АСКК як: модель функціонування, логічну модель, поведінка в процесі ЖЦ та математичну модель ТС;

– розробку класів та функцій АСКК, процес налаштування в Telegram;

– встановлення системи на віддалене середовище, що забезпечить безперервну роботу АСКК із користувачами;

– перевірку функціоналу та відповідність АСКК згідно поставлених задач;

– дослідження ефективності роботи АСКК за допомогою груп користувачів із різними умовами;

– опис послідовності дій при роботі користувача із АСКК.

За результатами досліджень, щодо ефективності розробленої АСКК, отримано такий висновок, що АСКК в повній мірі задовольняє поставленій меті та задачам, увесь набір функціоналу КК виконує автоматично, швидко, просто, безпомилково та ефективно.

Наразі АСКК виступатиме як найбільш досконала та завершена за своїм функціоналом ніж її аналоги. Швидкість та ефективність роботи, ріст прибутку були доведені за допомогою досліджень серед групи користувачів..

Таким чином, розроблена АСКК отримає широке використання серед користувачів SN, які бажають забезпечити автономну, ефективну та чітку роботу із криптоактивами, що у свою чергу, буде сприяти її подальшому розвитку, за рахунок отримання рекомендацій та потреб від користувачів, щодо додаткового функціоналу або модифікації вже наявного. При цьому, дослідження ефективності нововведень АСКК буде виконуватися на протязі її подальшого функціонування.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Blockchains: The great chain of being sure about things. [Електронний ресурс] // Economist Staff – 2015. – 10 p. – Режим доступу: https://www.economist.com/briefing/2015/10/31/the-great-chain-of-being-sure-about-things

2. Journal of Monetary Economics / Ricardo R. Reis – Elsevier, 2019 – 106 p.

3. Bitcoin Futures—What use are they? / S. Corbet, B. Lucey, M. Peat, S. Vigne // Economics Letters, 2018 – P. 23-27.

4. A Majority Consensus Approach to Concurrency Control for Multiple Copy Databases / Robert H. Thomas. // ACM Transactions on Database Systems 4, 2016 – P. 180–209

5. Bitcoin: A Peer-to-Peer Electronic Cash System [Електронний ресурс] – Режим доступу: https://bitcoin.org/bitcoin.pdf

6. Blockchain Revolution: How the Technology Behind Bitcoin Is Changing Money, Business, and the World. / D. Tapscott, A. Tapscott – New York, 2016. – 365 p.

7. Blockchain Technology Overview / D. Yaga, P. Mell, N/ Roby, K. Scarfone // National Institute of Standards and Technology Internal Report, 2018 – 68 p.

8. Blockchain technology innovations / T. Ahram; A. Sargolzaei; S. Sargolzaei // IEEE Technology & Engineering Management Conference (TEMSCON), 2017 – P. 137-141.

9. Research Handbook on Digital Transformations / F. Xavier Olleros, Majlinda Zhegu. – Published by Edward Elgar Publishing Limited, 2016 – 480 p.

10. Принципи роботи технології блокчейн та її властивості / Невмержицький Є. І. – НУКМА: Київ, 2021 – 53 с.

11. Blockchain: Blueprint for a New Economy. / M. Swan. – Published by O’Reilly Media, 2015. – 152 p.

12. Is Bitcoin Really Digital Gold? [Електронний ресурс] – Режим доступу: https://www.forbes.com/sites/forbesfinancecouncil/2020/05/11/is-bitcoin-reallydigital-gold/?sh=d73c0c62a842

13. Altcoin [Електронний ресурс] – Режим доступу: https://academy.binance.com/en/glossary/altcoin

14. Price volatilities of bitcoin futures [Електронний ресурс] – Режим доступу: https://www.sciencedirect.com/science/article/abs/pii/S1544612321001033

15. Andreas M. The Internet of Money: A collection of talks by Andreas M. Antonopoulos / Andreas M. Antonopoulos – Merkle Bloom LLC, 2016, – 152 p.

16. Roles of stable versus nonstable cryptocurrencies in Bitcoin market dynamics [Електронний ресурс] – Режим доступу: https://www.sciencedirect.com/science/article/abs/pii/S0275531922001088

17. Рубанов П. М. Аналіз розвитку світового ринку криптовалют. // П. М. Рубанов // Науковий вісник Ужгородського національного університету. Серія «Міжнародні економічні відносини та світове господарство». – 2019. – № 28, (ч.2). – С. 82–87

18. Stellar (payment network). [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Stellar_(payment_network)

19. How to use StellarBot, a free making market robot? [Електронний ресурс]. – Режим доступу: https://steemit.com/utopian-io/@ety001/how-to-use-stellarbot-a-free-making-market-robot

20. How to Tip with the Stellar Subreddit Tipping Bot. [Електронний ресурс]. – Режим доступу: https://www.lumenauts.com/tutorials/how-to-tip-with-the-stellar-subreddit-tipping-bot

21. How to Tip with the Stellar Twitter Tipping Bot. [Електронний ресурс]. – Режим доступу: https://www.lumenauts.com/tutorials/how-to-tip-with-the-lumen-twitter-tipping-bot

22. Stellarium bot. [Електронний ресурс]. – Режим доступу: https://github.com/jaracil/stellariumbot

23. Crypto-watcher. [Електронний ресурс]. – Режим доступу: https:// github.com/adrijano/crypto-watcher

24. Stellar Portal. [Електронний ресурс]. – Режим доступу: https://github.com/pakokrew/stellar-portal

25. Is Stellar As Secure As You Think? / M. Kim, Y. Kwon, Y. Kim // Proceedings of the IEEE Security & Privacy on the Blockchain, 2019.

26. How Stellar works: a quick, non-technical guide. [Електронний ресурс]. // Stellar – 2022. – 8 p. – Режим доступу: https://resources.stellar.org/ hubfs/Proof%20of%20Agreement%20explainer.pdf

27. Creating equitable access to the global financial system [Електронний ресурс]. – Режим доступу: https://stellar.org/foundation

28. Stellar Consensus Protocol (SCP) [Електронний ресурс]. – Режим доступу: https://developers.stellar.org/docs/fundamentals-and-concepts/stellar-consensus-protocol

29. Fast and secure global payments with Stellar / M. Lokhava, G. Losa, D. Mazières // SOSP '19: Proceedings of the 27th ACM Symposium on Operating Systems Principles, 2019 – P.80-96.

30. How does Stellar work. [Електронний ресурс]. – Режим доступу: https://www.stellar.org/learn/intro-to-stellar

31. Proof of Concept Agreement [Електронний ресурс]. – Режим доступу: https://www.contractstandards.com/public/contracts/proof-of-concept-agreement

32. Fast and secure global payments with Stellar. / M. Lokhava, G. Losa, et al. // SOSP ’19. Electronic funds transfer. – Canada, 2019. – 17 p.

33. Гусєва А. А. Засоби для створення автоматизованого управління криптовалютним портфелем на основі технології Blockchain з застосуванням Stellar Network / А. А. Гусєва, Н. О. Маслова // Матеріали III Міжнародної студентської наукової конференції «Теоретичне та практичне застосування результатів сучасної науки» (м. Біла Церква, 07.10.2022) – С. 177-178

34. Horizon API Reference. [Електронний ресурс]. – Режим доступу: https://developers.stellar.org/api/introduction/

35. Python Cookbook: Recipes for Mastering Python 3. / D. Beazley, B. Jones – O'Reilly Media, 2013. – 704 p.

36. Why Python? / J. Thomas – University of Arizona Department of Mathematics, 2016 – 5 p.

37. Telegram Applications [Електронний ресурс]. – Режим доступу: https://telegram.org/apps

38. Building telegram bots: develop bots in 12 programming languages using the telegram bot API / N. Modrzyk – Apress, 2018 –277 p.

39. Гусєва А. А. Застосування мережі Stellar при управлінні криптоактивами // А. А. Гусєва, Н. О. Маслова // Наукові праці Донецького національного технічного університету. Серія: «Інформатика, кібернетика та обчислювальна техніка» – 2022, №35. – С.

40. Ларман К. Застосування UML 2.0 і шаблонів проектування. / К. Ларман – Київ: Діалектика, 2019. – 736 с.

41. Automatic Transformation of User Stories into UML Use Case Diagrams using NLP Techniques / M. Elallaoui // Procedia Computer Science, 2018 – P. 42- 49

42. Automatic generation of TestNG tests cases from UML sequence diagrams in Scrum process. / M. Elallaoui, K. Nafil, R. Touahni // IEEE International Colloquium on Information Science and Technology (CiSt), 2016 – P. 65-70

43. Booch G. The Unified Modeling Language Usere Guide. / G. Booch, J. Rumbaugh, I. Jacobson – Addison-Wesley, 2010. – 496 p.

44. How are UML class diagrams built in practice? A usability study of two UML tools: Magicdraw and Papyrus / E. Planas // Computer Standards & Interfaces, 2020

45. UML Class Diagrams: Similarity Aspects and Matching [Електронний ресурс]. // Lecture Notes on Software Engineering – 2016. – 7 p. – Режим доступу: http://www.lnse.org/vol4/221-TD038.pdf

46. A Systematic Comparison of Roundtrip Software Engineering Approaches applied to UML Class Diagram / D. Rosca // A Systematic Comparison

of Roundtrip Software Engineering Approaches applied to UML Class Diagram, 2021 – P.861-868.

47. Seidl M. UML @ Classroom. / M. Seibl, M. Scholz, C. Huemer, G. Kappel – Germany: Springer, 2015. – 206 p.

48. From state diagrams to sequence diagrams: a requirements acquisition approach / B. Wei // International Journal of Computers and Applications, 2017 – P. 89-111

49. UML Diagrams in Software Engineering Research: A Systematic Literature Review / Yousef Barjakly // Department of Management Information Systems, 2021

50. About PythonAnywhere, an Anaconda company [Електронний ресурс]. – Режим доступу: https://www.pythonanywhere.com/about/company_details/

Додаток А

Зауваження нормоконтролера

Таблиця А.1 – Зауваження нормоконтролера

Сторінка	Позначення	Зміст зауваження
ПЗ		Кожний розділ починається із службового слова РОЗДІЛ приписними літерами

Додаток Б

Програмний код

Б.1 Лістинг класу «User»

```
class User:
    def __init__(self):
        self.id = None
        self.user_name = None
        self.private_key = None
```

Б.2 Лістинг змінних класу «StellarAccount»

```
class StellarAccount:
    def __init__(self):
        self.private_key = None
        self.public_key = None
        self.keypair = None
        self.source = None
```

Б.3 Лістинг функції «start_save_keys»

```
@bot.message_handler(commands=['start', 'menu'])
def start_save_keys(message):
    global authorization_state
    if not authorization_state:
        bot.send_message(message.chat.id, 'Виконайте авторизацію гаманця Stellar',
                         reply_markup=MarkupInterface.markup_author)
    if authorization_state:
        text_markup = main_markup_menu(message.chat.id)
        bot.send_message(message.chat.id, text=text_markup,
reply_markup=MarkupInterface.markup_main)
```

Б.4 Лістинг функції «take_private_key»

```
def take_private_key(message):
    global authorization_state
    try:
        user_id = message.chat.id
        mess_private_key = message.text
        User.id = user_id
        User.private_key = mess_private_key
    except Exception as e:
        bot.reply_to(message, 'Помилка при отриманні даних')
        start_save_keys()

    if mess_private_key[0] == "S" and len(mess_private_key) == 56:
```

```
        print("private key is OK")

        cursor.execute("SELECT * FROM Users_keys WHERE private_key=?",
(User.private_key,))
        data = cursor.fetchone()
        if data is None:
            cursor.execute("INSERT INTO Users_keys (user_id, private_key) VALUES (?, ?)",
(User.id, User.private_key,))
            connect.commit()

        authorization_state = True
        text_markup = main_markup_menu(message.chat.id)
        bot.send_message(chat_id=message.chat.id, text=text_markup,
reply_markup=MarkupInterface.markup_main)
    else:
        authorization_state = False
        bot.send_message(user_id, "🐵 УПС!"
                                  "\n🙈 Такий секретний ключ НЕ ІСНУЄ"
                                  "\n🙈 Спробуйте ще раз...")
        msg = bot.send_message(message.chat.id, 'Виконайте авторизацію гаманця Stellar',
                               reply_markup=MarkupInterface.markup_author)
```

Б.5 Лістинг класу «OfferAccount»

```
class OfferAccount:
    def __init__(self):
        self.id = None
        self.seller = None
        self.asset = None
        self.code = None
        self.issuer = None
        self.state = None
        self.price = None
        self.amount = None
        self.balance = None
```

Б.6 Лістинг функції «go»

```
@bot.message_handler(commands=['go'])
def go(message):
    user_id = message.chat.id
    User.user_name = message.from_user.username
    get_user_id = cursor.fetchall()

    if len(get_user_id) != 0:
        del get_user_id[0:len(get_user_id) - 1]

        button_stop_process = types.KeyboardButton('⛔ Зупинити редагування')
        markup = types.ReplyKeyboardMarkup(resize_keyboard=True,
one_time_keyboard=True).add(button_stop_process)
        bot.edit_message_text(chat_id=message.chat.id, message_id=message.id, text='🚀
```

```
Починаємо обробку даних')
        bot.send_message(message.chat.id, 'Для зупинки натисність кнопку в області
клавіатури', reply_markup=markup)

        StatusProgram.break_program = False
        connect_Stellar(get_user_id[0][0], user_id)

    elif len(get_user_id) == 0:
        bot.send_message(user_id, '😁 Спочатку необхідно авторизуватися! \n  Натисни /start')
```

Б.7 Лістинг функції «stop»

```
@bot.message_handler(commands=['stop'])
@bot.message_handler(content_types=['text'])
def stop(message):
    StatusProgram.break_program = True

    if message.text == '⛔ Зупинити відстеження':
        bot.send_message(message.chat.id, "✅ Відстеження зупинено",
reply_markup=types.ReplyKeyboardRemove())
        text_markup = main_markup_menu(message.chat.id)
        bot.send_message(message.chat.id, text=text_markup,
reply_markup=MarkupInterface.markup_main)
```

Б.8 Лістинг функції «search_on_domain»

```
@bot.message_handler(commands=['search_on_domain'])
def search_on_domain(message):
    msg = bot.send_message(message.chat.id, '🔗 Введіть адресу домена')
    bot.register_next_step_handler(msg, take_domain_address)
```

Б.9 Лістинг функції «take_domain_address»

```
try:
    user_id = message.chat.id
    mess_domain_address = message.text
    User.id = user_id

    extractor = urlextract.URLExtract()
    domain_address = extractor.find_urls(mess_domain_address)
    if len(domain_address) == 0:
        bot.send_message(user_id, '✍🏻 Невірна адреса домена \n Спробуйте ще раз...')
    elif len(domain_address) > 1:
        bot.send_message(user_id, '☝🏻 Не більше однієї адреси домена\n Спробуй ще раз...')
    else:
        cursor.execute("SELECT private_key FROM Users_keys WHERE user_id=?", (user_id,))
        connect.commit()
        get_user_id = cursor.fetchall()

        if len(get_user_id) != 0:
```

```
            del get_user_id[0:len(get_user_id) - 1]
            bot.send_message(user_id, '🔎 ПОШУК МОНЕТ...')

            add_new_order_from_domen(get_user_id[0][0], user_id, mess_domain_address)

        elif len(get_user_id) == 0:
            bot.send_message(user_id, '😁 Спочатку необхідно авторизуватися!')
            bot.send_message(message.chat.id, 'Виконайте авторизацію гаманця Stellar',
                             reply_markup=MarkupInterface.markup_author)
except Exception as e:
    bot.reply_to(message, '⛔Помилка при отриманні даних з домена')
```

Б.10 Лістинг функції «add_new_order_from_domen»

```
def add_new_order_from_domen(key_list, user_id, domain_address):
    StellarAccount.private_key = key_list

    StellarAccount.keypair = Keypair.from_secret(StellarAccount.private_key)
    StellarAccount.public_key = StellarAccount.keypair.public_key

    StellarAccount.source = server.load_account(StellarAccount.public_key)

    account_assets = server.accounts().account_id(StellarAccount.public_key).call()

    array_account_assets = []
    for my_assets in account_assets['balances']:
        if my_assets['asset_type'] != 'native' and float(my_assets['balance']) > 0:
            array_account_assets.append([my_assets['asset_code'], my_assets['balance'],
my_assets['asset_issuer']])

    domainAsset = stellar_sdk.sep.stellar_toml.fetch_stellar_toml(domain_address, client=None,
use_http=False)

    for assets in domainAsset['CURRENCIES']:
        offers = server.offers().for_seller(StellarAccount.public_key).limit(200).call()

        asset_domain = Asset(assets['code'], assets['issuer'])
        MessageTemplate.messReturnCheck = f"НОВИЙ АКТИВ [Домен - {domain_address}]"
        flag_message = 0
        try:
            flag_offers = 0
            for offer in offers['_embedded']['records']:
                if offer['buying']['asset_type'] != 'native':
                    if assets['issuer'] == offer['buying']['asset_issuer']:  # acc_ass[2]
                        flag_offers = 1

            flag_assets = flag_assets_balance = 0
            if flag_offers != 1:

                for acc_ass in array_account_assets:
                    if assets['issuer'] == acc_ass[2]:
```

```
                flag_assets = 1
                if acc_ass[1] < 1:  # если баланс монеты меньше 1
                    flag_assets_balance = 1
        if flag_assets == 0:
            price, amount = domain_asset_check(asset_domain, assets['code'], assets['issuer'])
            if price != 0:
                try:
                    add_trust_transaction =
(TransactionBuilder(source_account=StellarAccount.source,
                                                network_passphrase=network_passphrase,
                                                base_fee=base_fee, )
                                    .append_change_trust_op(asset=asset_domain)
                                    .set_timeout(30)
                                    .build())
                    add_trust_transaction.sign(StellarAccount.private_key)
                    server.submit_transaction(add_trust_transaction)
                    print("Добавлено линию доверия")

                    append_buy_offer = update_buy_offer(0, asset_domain, amount, price)
                    server.submit_transaction(append_buy_offer)
                    flag_message = 1
                    message =
f"{MessageTemplate.messReturnCheck}\n{MessageTemplate.messAddOffer}\n" \
                            f"□ КУПІВЛЯ: {assets['code']}\n" + \
                            "💰 ЦІНА: {:.12f}".format(price) + \
                            f"🏘 ОБ'ЄМ: {amount}"
                except stellar_sdk.exceptions.BadRequestError:
                    flag_message = -1
            # else: flag_message = 0
        else:
            if flag_assets_balance == 1:
                if price != 0:
                    try:
                        price, amount = domain_asset_check(asset_domain, assets['code'],
assets['issuer'])
                        append_buy_offer = update_buy_offer(0, asset_domain, amount, price)
                        # server.submit_transaction(append_buy_offer)
                        flag_message = 1
                    except stellar_sdk.exceptions.BadRequestError:
                        flag_message = -1

        if flag_message == 1:
            print("ВЫСТАВЛЕНО НОВУЮ МОНЕТУ НА ПОКУПКУ\n")
            message =
f"{MessageTemplate.messReturnCheck}\n{MessageTemplate.messAddOffer}\n" \
                    f"□ КУПІВЛЯ: {assets['code']}\n" + \
                    "💰 ЦІНА: {:.12f}".format(price) + \
                    f"\n🏘 ОБ'ЄМ: {amount}"
            bot.send_message(user_id, message)
        elif flag_message == -1:
            print("new buy order -", assets['code'], "- Transaction Failed")
            message =
```

```
f"{MessageTemplate.messReturnCheck}\n{MessageTemplate.messFailedOffer}\n" \
                    f"КУПІВЛЯ: {assets['code']}\n- Угоду не додано"

            bot.send_message(user_id, message)

        except TypeError:
            iterable = iter(assets.items())
            next(iterable)

    bot.send_message(user_id, '🔎 ПОШУК АКТИВІВ ЗАВЕРШЕНО')
```

Б.11 Лістинг функції «domain_asset_check»

```
def domain_asset_check(asset_domain, assets_code, assets_issuer):
    top_buy_price, top_buy_amount = check_top_order(asset_domain, 'bids')
    top_sell_price, top_sell_amount = check_top_order(asset_domain, 'asks')

    if top_sell_price / top_buy_price >= domain_profit:
        print(f"{assets_code} - {assets_issuer}")
        print("Order Book TOP BUY Price: {:.12f}".format(top_buy_price))
        print("Order Book TOP SELL Price: {:.12f}".format(top_sell_price))

        price = top_buy_price + price_diff
        amount = sum_buy_xlm // price
    else:
        price = amount = 0
    return price, amount
```

Б.12 Лістинг функції «connect_Stellar»

```
def connect_Stellar(key_list, user_id):
    StellarAccount.private_key = key_list
    StellarAccount.keypair = Keypair.from_secret(StellarAccount.private_key)
    StellarAccount.public_key = StellarAccount.keypair.public_key

    StellarAccount.source = server.load_account(StellarAccount.public_key)

    while not StatusProgram.break_program:
        account_assets = server.accounts().account_id(StellarAccount.public_key).call()
        offers = server.offers().for_seller(StellarAccount.public_key).limit(200).call()

        array_account_assets = []
        for my_assets in account_assets['balances']:
            if my_assets['asset_type'] != 'native' and float(my_assets['balance']) > 0 \
                    and my_assets['asset_code'] != 'AQUA' and my_assets['asset_code'] != 'USDC':
                array_account_assets.append([my_assets['asset_code'], my_assets['balance'],
my_assets['asset_issuer']])

        for offer in offers["_embedded"]["records"]:
            if offer['selling']['asset_type'] != 'native':
                for i in range(len(array_account_assets)):
```

```
                if array_account_assets[i] != 0 and array_account_assets[i][0] ==
offer['selling']['asset_code']:
                    array_account_assets[i] = 0

        copy_array_account_assets = array_account_assets.copy()
        try:
            while True:
                copy_array_account_assets.remove(0)
        except ValueError:
            pass

        if copy_array_account_assets is not False:
            for i in range(len(copy_array_account_assets)):
                if not StatusProgram.break_program:
                    asset = Asset(copy_array_account_assets[i][0], copy_array_account_assets[i][2])
                    MessageTemplate.messReturnCheck = "НОВИЙ АКТИВ"
                    top_price, top_amount = check_top_order(asset, 'asks')
                    if float(copy_array_account_assets[i][1]) * top_price > 1:
                        try:
                            top_buy_price, top_buy_amount = check_top_order(asset, 'bids')
                        except TypeError:
                            top_buy_price = price_diff
                        try:
                            top_sell_price, top_sell_amount = check_top_order(asset, 'asks')
                        except TypeError:
                            top_sell_price = price_diff * 100

                        if top_sell_price / top_buy_price > 2:
                            price = top_sell_price - price_diff
                            append_sell_offer = update_sell_offer(0, asset,
copy_array_account_assets[i][1], price)
                            try:
                                server.submit_transaction(append_sell_offer)
                                print("ОРДЕР ДОБАВЛЕНО")
                                message =
f"{MessageTemplate.messReturnCheck}\n{MessageTemplate.messAddOffer}\n" \
                                      f"□ ПРОДАЖ: {copy_array_account_assets[i][0]}\n" + \
                                      "💰 ЦІНА: {:.12f}".format(price) + \
                                      f"\n📦 ОБ'ЄМ: {copy_array_account_assets[i][1]}"
                            except stellar_sdk.exceptions.BadRequestError:
                                print("new sell order -", copy_array_account_assets[i][0], "- Transaction
Failed")
                                message =
f"{MessageTemplate.messReturnCheck}\n{MessageTemplate.messFailedOffer}\n" \
                                      f"ПРОДАЖ: {copy_array_account_assets[i][0]}\n- Угоду не додано"
                            bot.send_message(user_id, message)
                else:
                    bot.send_message(user_id, "✅ Редагування зупинено",
reply_markup=types.ReplyKeyboardRemove())
                    text_markup = main_markup_menu(user_id)
                    bot.send_message(user_id, text=text_markup,
reply_markup=MarkupInterface.markup_main)
```

```
                return

        for offer in offers["_embedded"]["records"]:
            if not StatusProgram.break_program:
                print(f"\nUsername: {User.user_name}"
                      f"\nID: {offer['id']}"
                      f"\nSeller: {offer['seller']}")

                OfferAccount.id, OfferAccount.seller = offer['id'], offer['seller']
                OfferAccount.amount = offer['amount']

                if offer['selling']['asset_type'] != 'native':
                    print(f"Selling: {offer['selling']['asset_code']}")
                    OfferAccount.code = offer['selling']['asset_code']
                    OfferAccount.issuer = offer['selling']['asset_issuer']
                    OfferAccount.state = 'asks'
                    OfferAccount.price = float(offer['price_r']['n'] / offer['price_r']['d'])
                    OfferAccount.asset = Asset(offer['selling']['asset_code'],
offer['selling']['asset_issuer'])

                    for balance_asset in account_assets['balances']:
                        if balance_asset['asset_type'] != 'native':
                            if offer['selling']['asset_code'] == balance_asset['asset_code']:
                                OfferAccount.balance = balance_asset['balance']

                    sell_orders_check(user_id)

                elif offer['buying']['asset_type'] != 'native':
                    print(f"Buying: {offer['buying']['asset_code']}")
                    OfferAccount.code = offer['buying']['asset_code']
                    OfferAccount.issuer = offer['buying']['asset_issuer']
                    OfferAccount.state = 'bids'
                    OfferAccount.price = float(offer['price_r']['d'] / offer['price_r']['n'])
                    OfferAccount.asset = Asset(offer['buying']['asset_code'],
offer['buying']['asset_issuer'])

                    buy_orders_check(user_id)
            else:
                bot.send_message(user_id, "✓ Редагування зупинено",
reply_markup=types.ReplyKeyboardRemove())
                text_markup = main_markup_menu(user_id)
                bot.send_message(user_id, text=text_markup,
reply_markup=MarkupInterface.markup_main)
                return
```

Б.13 Лістинг функції «check_top_order»

```
def check_top_order(asset, state):
    top_order_asset = server.orderbook(asset, lumen).limit(1).call()
    for order in top_order_asset[state]:
        top_price = order['price_r']['n'] / order['price_r']['d']
```

```
    top_amount = order['amount']
    return top_price, top_amount
```

Б.14 Лістинг функції «sell_orders_check»

```
def sell_orders_check(user_id):
    top_price, top_amount = print_orders_info()

    result = chek_my_top_orders(float(top_price), top_amount, user_id)
    if result:
        change_sell_order(float(top_price), float(top_amount), user_id)
    elif not result:
        if float(OfferAccount.balance) - float(OfferAccount.amount) >= 1:
            print("БАЛАНС отличается")
            update_sell_offer_amount = update_sell_offer(int(OfferAccount.id), OfferAccount.asset,
OfferAccount.balance,
                                                         OfferAccount.price)
            try:
                server.submit_transaction(update_sell_offer_amount)
                print("БАЛАНС ОБНОВЛЕН")
                MessageTemplate.messReturnCheck = "Параметр BALANCE відрізняється"
                message =
f"{MessageTemplate.messReturnCheck}\n{MessageTemplate.messUpdateOffer}\n" \
                      f"□ ПРОДАЖ: {OfferAccount.code}\n" + \
                      f"■ ОБ'ЄМ: {OfferAccount.amount} ➞ {OfferAccount.balance}"
            except stellar_sdk.exceptions.BadRequestError:
                print("Transaction Failed")
                message =
f"{MessageTemplate.messReturnCheck}\n{MessageTemplate.messFailedOffer}\n" \
                      f"ПРОДАЖ: {OfferAccount.code}\n- Угоду не оновлено"
            bot.send_message(user_id, message)
        else:
            print(f"~ {OfferAccount.code} IS OK")
```

Б.15 Лістинг функції «update_sell_offer»

```
def update_sell_offer(offer_id, asset, amount, price):
    update_sell_offer_transaction = (TransactionBuilder(source_account=StellarAccount.source,
                                                        network_passphrase=network_passphrase,
                                                        base_fee=base_fee, )
                                     .append_manage_sell_offer_op(offer_id=offer_id,
                                                                  selling=asset,
                                                                  buying=lumen,
                                                                  amount=amount,
                                                                  price=str(price), )
                                     .set_timeout(30)
                                     .build())
    update_sell_offer_transaction.sign(StellarAccount.private_key)
    return update_sell_offer_transaction
```

Б.16 Лістинг функції «buy_orders_check»

```
def buy_orders_check(user_id):
    top_price, top_amount = print_orders_info()

    result = chek_my_top_orders(float(top_price), top_amount, user_id)
    if result:
        change_buy_order(float(top_price), float(top_amount), user_id)
    elif not result:
        print(f"~ {OfferAccount.code} IS OK")
```

Б.17 Лістинг функції «chek_my_top_orders»

```
def chek_my_top_orders(top_price, top_amount, user_id):
    if OfferAccount.price != top_price:
        MessageTemplate.messReturnCheck = "📊 Параметр PRICE відрізняється\n"
        print(" PRICE отличается")
        return True
    elif OfferAccount.amount != top_amount:
        MessageTemplate.messReturnCheck = "📊 Параметр AMOUNT відрізняється\n"
        print(" AMOUNT отличается")
        return True
    else:
        return False
```

Б.18 Лістинг функції «change_sell_order»

```
def change_sell_order(top_sell_price, top_sell_amount, user_id):
    message =
f"{MessageTemplate.messReturnCheck}\n{MessageTemplate.messNoUpdateOffer}\n□
ПРОДАЖ: {OfferAccount.code}"
    cause = ""
    try:
        top_buy_price, top_buy_amount = check_top_order(OfferAccount.asset, 'bids')
    except TypeError:
        top_buy_price = price_diff

    if top_sell_price / top_buy_price > sell_profit_coeff:
        if OfferAccount.price / top_sell_price < 4:
            if top_sell_amount >= sell_amount_difference_coeff:
                change_price = top_sell_price - price_diff
                if float(OfferAccount.balance) - float(OfferAccount.amount) >= 1:
                    amount = OfferAccount.balance
                    message =
f"{MessageTemplate.messReturnCheck}\n{MessageTemplate.messUpdateOffer}\n" \
                            f"□ ПРОДАЖ: {OfferAccount.code}\n" + \
                            "💰 ЦІНА: {:.12f}".format(OfferAccount.price) + " ➞
{:.12f}\n".format(change_price) + \
                            f"📦 ОБ'ЄМ: {OfferAccount.amount} ➞ {OfferAccount.balance}"
                else:
                    amount = OfferAccount.amount
                    message =
f"{MessageTemplate.messReturnCheck}\n{MessageTemplate.messUpdateOffer}\n" \
```

```
                    f"□ ПРОДАЖ: {OfferAccount.code}\n" + \
                    "💰 ЦІНА: {:.12f}".format(OfferAccount.price) + " ➞
{:.12f}\n".format(change_price) + \
                    f"📦 ОБ'ЄМ: {OfferAccount.amount}"

        update_sell_offer_transaction = update_sell_offer(int(OfferAccount.id),
OfferAccount.asset, amount, change_price)
        try:
            server.submit_transaction(update_sell_offer_transaction)
            print("ПРОДАЖА ОБНОВЛЕНА")
        except stellar_sdk.exceptions.BadRequestError:
            print("Transaction Failed")
            message =
f"{MessageTemplate.messReturnCheck}\n{MessageTemplate.messFailedOffer}\n" \
                    f"ПРОДАЖ: {OfferAccount.code}\n- Угоду не оновлено"
      else:
        cause = f"- Різниця об'єму між угодами меньше {sell_amount_difference_coeff}
xlm"
    else:
      cause = "- Різниця в ціні більше 4х"
  else:
    cause = f"- Вигода менше {sell_profit_coeff}х"
  bot.send_message(user_id, f"{message}\n{cause}")
```

Б.19 Лістинг функції «update_buy_offer»

```
def update_buy_offer(offer_id, asset, amount, price):
  update_buy_offer_transaction = (TransactionBuilder(source_account=StellarAccount.source,
                                  network_passphrase=network_passphrase,
                                  base_fee=base_fee, )
                  .append_manage_buy_offer_op(offer_id=int(offer_id),
                                  selling=lumen,
                                  buying=asset,
                                  amount=str(amount),
                                  price=str(price), )
                  .set_timeout(30)
                  .build())
  update_buy_offer_transaction.sign(StellarAccount.private_key)
  return update_buy_offer_transaction
```

Б.20 Лістинг функції «change_buy_order»

```
def change_buy_order(top_buy_price, top_buy_amount, user_id):
  message =
f"{MessageTemplate.messReturnCheck}\n{MessageTemplate.messNoUpdateOffer}\n□
КУПІВЛЯ: {OfferAccount.code}"
  cause = ""

  top_sell_price, top_sell_amount = check_top_order(OfferAccount.asset, 'asks')
  if top_sell_price / top_buy_price >= buy_profit_coeff:
    if float(OfferAccount.amount) <= buy_amount_difference_coeff:
```

```
            change_price = top_buy_price + price_diff
            amount = round(float(OfferAccount.amount) / change_price, 7)
            update_buy_offer_transaction = update_buy_offer(OfferAccount.id, OfferAccount.asset,
amount, change_price)
            try:
                server.submit_transaction(update_buy_offer_transaction)
                print("ПОКУПКА ОБНОВЛЕНА")
                message =
f"{MessageTemplate.messReturnCheck}\n{MessageTemplate.messUpdateOffer}\n" + \
                        f"□ КУПІВЛЯ: {OfferAccount.code}\n" + \
                        "💰 ЦІНА: {:.12f}".format(OfferAccount.price) + " ➞
{:.12f}\n".format(change_price) + \
                        f"🏬 ОБ'ЄМ: {OfferAccount.amount}"

            except stellar_sdk.exceptions.BadRequestError:
                print("Transaction Failed")
                message =
f"{MessageTemplate.messReturnCheck}\n{MessageTemplate.messFailedOffer}\n" \
                        f"КУПІВЛЯ: {OfferAccount.code}\n- Угоду не оновлено"
        else:
            cause = f"- Різниця об'єму між угодами більше {buy_amount_difference_coeff} xlm"
    else:
        cause = f"- Вигода менше {buy_profit_coeff}x"
        delete_buy_offer_transaction = update_buy_offer(int(OfferAccount.id),
OfferAccount.asset, 0, 1)
        try:
            server.submit_transaction(delete_buy_offer_transaction)
            print("ОРДЕР ОТМЕНЕН")
            message = f"🗑 УГОДУ ВИДАЛЕНО\n" \
                      f"КУПІВЛЯ: {OfferAccount.code}"
        except stellar_sdk.exceptions.BadRequestError:
            print("Transaction Failed")
            message =
f"{MessageTemplate.messReturnCheck}\n{MessageTemplate.messFailedOffer}\n" \
                    f"КУПІВЛЯ: {OfferAccount.code}\n- Угоду не видалено"
    bot.send_message(user_id, f'{message}\n{cause}')
```

Б.21 Лістинг функції «get_balance»

```
def get_balance(message):
    user_id = message.chat.id
    cursor.execute("SELECT private_key FROM Users_keys WHERE user_id=?", (user_id,))
    connect.commit()
    get_user_id = cursor.fetchall()

    if len(get_user_id) != 0:
        del get_user_id[0:len(get_user_id) - 1]

    StellarAccount.private_key = get_user_id[0][0]

    StellarAccount.keypair = Keypair.from_secret(StellarAccount.private_key)
    StellarAccount.public_key = StellarAccount.keypair.public_key
```

```
    StellarAccount.source = server.load_account(StellarAccount.public_key)

    account_assets = server.accounts().account_id(StellarAccount.public_key).call()
    array_assets = []
    if message.text == '1':
        for my_assets in account_assets['balances']:
            if float(my_assets['balance']) > 0 and my_assets['asset_type'] != 'native':
                array_assets.append(f"{my_assets['asset_code']} - {my_assets['balance']}")
            if my_assets['asset_type'] == 'native':
                array_assets.append(f"XLM - {my_assets['balance']}")
        bot.send_message(user_id, '\n'.join(array_assets))
    elif message.text == 'XLM':
        for my_assets in account_assets['balances']:
            if my_assets['asset_type'] == 'native':
                bot.send_message(user_id, f"XLM - {my_assets['balance']}")
    else:
        flag = 1
        for my_assets in account_assets['balances']:
            if my_assets['asset_type'] != 'native' and my_assets['asset_code'] == message.text:
                flag = 0
                bot.send_message(user_id, f"{my_assets['asset_code']} - {my_assets['balance']}")
        if flag == 1:
            bot.send_message(user_id, '☹ Даний актив відсутній у гаманці')
```

Б.22 Лістинг функції «get_payment_public_key»

```
def get_payment_public_key(message):
    receiver_public_key = message.text
    if receiver_public_key[0] == "G" and len(receiver_public_key) == 56:
        msg_asset = bot.send_message(message.chat.id, "Введіть назву активу (Наприклад:
XLM)")
        bot.register_next_step_handler(msg_asset, get_payment_asset, receiver_public_key)
    else:
        bot.send_message(message.chat.id, "Вказана адреса отримувача невірна \n"
                                          "Спробуйте ще раз")
        msg_Pk = bot.send_message(message.chat.id, "Введіть адресу отримувача")
        bot.register_next_step_handler(msg_Pk, get_payment_public_key)
```

Б.23 Лістинг функції «get_payment_asset»

```
def get_payment_asset(message, receiver_public_key):
    asset = message.text
    msg_total = bot.send_message(message.chat.id, "Введіть суму відправлення")
    bot.register_next_step_handler(msg_total, get_payment_total, asset, receiver_public_key)
```

Б.24 Лістинг функції «get_payment_total»

```
def get_payment_total(message, asset, receiver_public_key):
    total = message.text.replace(",", ".")
    if total.isdigit():
        msg_text_memo = bot.send_message(message.chat.id, "Введіть текст повідомлення")
```

```
        bot.register_next_step_handler(msg_text_memo, payment_asset, receiver_public_key,
asset, total)
    else:
        try:
            float(total)
            msg_text_memo = bot.send_message(message.chat.id, "Введіть текст повідомлення")
            bot.register_next_step_handler(msg_text_memo, payment_asset, receiver_public_key,
asset, total)
        except ValueError:
            bot.send_message(message.chat.id, "Вказане значення суми відправлення невірне\n"
                                              "Спробуйте ще раз")
            msg_total = bot.send_message(message.chat.id, "Введіть суму відправлення")
            bot.register_next_step_handler(msg_total, get_payment_total, asset,
receiver_public_key)
```

Б.25 Лістинг функції «payment_asset»

```
def payment_asset(message, receiver_public_key, asset, total):
    text_memo = message.text

    button_done_payment = types.InlineKeyboardButton('Так', callback_data='done_payment')
    button_edit_payment = types.InlineKeyboardButton(' Редагувати',
callback_data='edit_payment')

    markup_payment_confirm =
types.InlineKeyboardMarkup(row_width=2).add(button_done_payment, button_edit_payment)

    bot.send_message(message.chat.id, f'Отримувач: {receiver_public_key}'
                                      f'\nНазва активу: {asset}'
                                      f'\nСуми відправлення: {total}'
                                      f'\nТекст повідомлення: {text_memo}'
                                      f'\n\nВиконати транзакцію?',
reply_markup=markup_payment_confirm)

if getText() = 'ТАК':
    send_asset = (TransactionBuilder(source_account=StellarAccount.source,
                                     network_passphrase=network_passphrase,
                                     base_fee=base_fee, )
                  .add_text_memo(text_memo)
                  .append_payment_op(receiver_public_key, asset, str(total))
                  .set_timeout(30)
                  .build())
    send_asset.sign(StellarAccount.private_key)
```

Б.26 Лістинг функції «get_ts_edit_domain_profit»

```
def get_ts_domain_profit(message):
    global domain_profit
    domain_profit = message.text
    bot.send_message(chat_id=message.chat.id, text=f'Параметри ТС (пошук за доменом):\n' \
                                        f'  • вигода - x{domain_profit}\n' \
                                        f'  • об`єм купівлі - {sum_buy_xlm} xlm\n'
```

```
                                f'Для редагування оберіть бажаний параметр нижче',
            reply_markup=MarkupInterface.markup_change_ts_domain)
```

Б.27 Лістинг функції «get_ts_edit_domain_total»

```
def get_ts_domain_total(message):
    global sum_buy_xlm
    sum_buy_xlm = message.text
    bot.send_message(chat_id=message.chat.id, text=f'Параметри ТС (пошук за доменом):\n' \
                                f'   • вигода - x{domain_profit}\n' \
                                f'   • об`єм купівлі - {sum_buy_xlm} xlm\n'
                                f'Для редагування оберіть бажаний параметр нижче',
            reply_markup=MarkupInterface.markup_change_ts_domain)
```

Б.28 Лістинг функції «get_ts_edit_buy_profit»

```
def get_ts_edit_buy_profit(message):
    global buy_profit_coeff
    buy_profit_coeff = message.text
    bot.send_message(message.chat.id, text=f'Параметри ТС (редагування угод):\n' \
                                f'   Купівля:\n' \
                                f'   • вигода - x{buy_profit_coeff}\n' \
                                f'   • різниця об`єму - {buy_amount_difference_coeff} xlm\n'
                                f'Для редагування оберіть бажаний параметр нижче',
            reply_markup=MarkupInterface.markup_change_ts_edit_buy)
```

Б.29 Лістинг функції «get_ts_edit_buy_amount»

```
def get_ts_edit_buy_amount(message):
    global buy_amount_difference_coeff
    buy_amount_difference_coeff = message.text
    bot.send_message(message.chat.id, text=f'Параметри ТС (редагування угод):\n' \
                                f'   Купівля:\n' \
                                f'   • вигода - x{buy_profit_coeff}\n' \
                                f'   • різниця об`єму - {buy_amount_difference_coeff} xlm\n'
                                f'Для редагування оберіть бажаний параметр нижче',
            reply_markup=MarkupInterface.markup_change_ts_edit_buy)
```

Б.30 Лістинг функції «get_ts_edit_sell_profit»

```
def get_ts_edit_sell_profit(message):
    global sell_profit_coeff
    sell_profit_coeff = message.text
    bot.send_message(chat_id=message.chat.id, text=f'Параметри ТС (редагування угод):\n' \
                                f'   Продаж:\n' \
                                f'   • вигода - x{sell_profit_coeff}\n' \
                                f'   • різниця об`єму - {sell_amount_difference_coeff} xlm\n'
                                f'Для редагування оберіть бажаний параметр нижче',
            reply_markup=MarkupInterface.markup_change_ts_edit_sell)
```

Б.31 Лістинг функції «get_ts_edit_sell_amount»

```
def get_ts_edit_sell_amount(message):
    global sell_amount_difference_coeff
    sell_amount_difference_coeff = message.text
    bot.send_message(chat_id=message.chat.id, text=f'Параметри ТС (редагування угод):\n' \
                                     f'  Продаж:\n' \
                                     f'  • вигода - x{sell_profit_coeff}\n' \
                                     f'  • різниця об`єму - {sell_amount_difference_coeff} xlm\n'
                                     f'Для редагування оберіть бажаний параметр нижче',
                     reply_markup=MarkupInterface.markup_change_ts_edit_sell)
```

Б.32 Лістинг класу «MarkupInterface»

```
class MarkupInterface:
    button_authorization = types.InlineKeyboardButton('🔐 Авторизуватися',
callback_data='authorization')
    button_logout = types.InlineKeyboardButton('🔒 Вихід із акаунту', callback_data='logout')

    button_wallet = types.InlineKeyboardButton('💰 Особистий гаманець',
callback_data='wallet')
    button_balance = types.InlineKeyboardButton('💳 Баланс рахунку', callback_data='balance')
    button_payment = types.InlineKeyboardButton('💸 Надіслати кошти',
callback_data='payment')

    button_state = types.InlineKeyboardButton('🕹 Режим роботи', callback_data='state')
    button_state_change_orders = types.InlineKeyboardButton('🖊 Редагування угод',
callback_data='state_change_orders')
    button_state_tracking = types.InlineKeyboardButton('🕮 Відстеження подій',
callback_data='state_tracking')

    button_process = types.InlineKeyboardButton('⏱ Обробка даних', callback_data='process')
    button_process_go = types.InlineKeyboardButton('⏳ Почати обробку',
callback_data='process_go')

    button_search_on_domain = types.InlineKeyboardButton('🔎 Пошук за доменом',
callback_data='search_on_domain')

    button_change_ts = types.InlineKeyboardButton('📝 Зміна параметрів ТС',
callback_data='change_ts')
    button_change_ts_domain = types.InlineKeyboardButton('Параметри для пошуку за
доменом',
                                                         callback_data='change_ts_domain')
    button_change_ts_order = types.InlineKeyboardButton('Параметри для редагування угод',
                                                        callback_data='change_ts_order')
    btn_ts_domain_profit = types.InlineKeyboardButton('Вигода',
callback_data='ts_domain_profit')
    btn_ts_domain_total = types.InlineKeyboardButton('Об`єм купівлі',
callback_data='ts_domain_total')
    btn_ts_domain_back = types.InlineKeyboardButton('⬅ Назад',
callback_data='ts_domain_back')
```

```
    btn_ts_edit_buy = types.InlineKeyboardButton('Купівля', callback_data='ts_edit_buy')
    btn_ts_edit_sell = types.InlineKeyboardButton('Продаж', callback_data='ts_edit_sell')
    btn_ts_edit_back = types.InlineKeyboardButton('⬅ Назад', callback_data='ts_edit_back')

    btn_ts_edit_buy_profit = types.InlineKeyboardButton('Вигода',
callback_data='ts_edit_buy_profit')
    btn_ts_edit_buy_total = types.InlineKeyboardButton('Різниця об`єму купівлі',
callback_data='ts_edit_buy_amount')
    btn_ts_edit_buy_back = types.InlineKeyboardButton('⬅ Назад',
callback_data='ts_edit_buy_back')

    btn_ts_edit_sell_profit = types.InlineKeyboardButton('Вигода',
callback_data='ts_edit_sell_profit')
    btn_ts_edit_sell_total = types.InlineKeyboardButton('Різниця об`єму купівлі',
callback_data='ts_edit_sell_amount')
    btn_ts_edit_sell_back = types.InlineKeyboardButton('⬅ Назад',
callback_data='ts_edit_sell_back')

    button_back_main = types.InlineKeyboardButton('⬅До головного меню',
callback_data='back_main')

    markup_author = types.InlineKeyboardMarkup(row_width=1).add(button_authorization)
    markup_main = types.InlineKeyboardMarkup(row_width=2).add(button_wallet,
button_process, button_search_on_domain,
                                                button_state, button_change_ts, button_logout)
    markup_wallet = types.InlineKeyboardMarkup(row_width=2).add(button_balance,
button_payment, button_back_main)
    markup_state = types.InlineKeyboardMarkup(row_width=2).add(button_state_change_orders,
button_state_tracking, button_back_main)
    markup_process = types.InlineKeyboardMarkup(row_width=2).add(button_process_go,
button_back_main)
    markup_change_ts =
types.InlineKeyboardMarkup(row_width=1).add(button_change_ts_domain,
button_change_ts_order, button_back_main)
    markup_change_ts_domain =
types.InlineKeyboardMarkup(row_width=1).add(btn_ts_domain_profit, btn_ts_domain_total,
btn_ts_domain_back)
    markup_change_ts_edit = types.InlineKeyboardMarkup(row_width=2).add(btn_ts_edit_buy,
btn_ts_edit_sell,btn_ts_edit_back)
    markup_change_ts_edit_buy =
types.InlineKeyboardMarkup(row_width=1).add(btn_ts_edit_buy_profit, btn_ts_edit_buy_total,
btn_ts_edit_buy_back)
    markup_change_ts_edit_sell =
types.InlineKeyboardMarkup(row_width=1).add(btn_ts_edit_sell_profit,
btn_ts_edit_sell_total,btn_ts_edit_sell_back)
```

Б.33 Лістинг функції «main_markup_men»

```
def main_markup_menu(user_id):
    cursor.execute("SELECT private_key FROM Users_keys WHERE user_id=?", (user_id,))
    connect.commit()
```

```
    get_user_id = cursor.fetchall()

    if len(get_user_id) != 0:
        del get_user_id[0:len(get_user_id) - 1]
        StellarAccount.private_key = get_user_id[0][0]
        StellarAccount.keypair = Keypair.from_secret(StellarAccount.private_key)
        text = f'Адреса гаманця: {StellarAccount.keypair.public_key}\n' \
               f'Режим робти: {bot_state}\n' \
               f'Параметри ТС (редагування угод):\n' \
               f'   Купівля:\n' \
               f'   • вигода - x{buy_profit_coeff}\n' \
               f'   • різниця об`єму - {buy_amount_difference_coeff} xlm\n' \
               f'   Продаж:\n' \
               f'   • вигода - x{sell_profit_coeff}\n' \
               f'   • різниця об`єму - {sell_amount_difference_coeff} xlm\n' \
               f'Параметри ТС (пошук за доменом):\n' \
               f'   • вигода - x{domain_profit}\n' \
               f'   • об`єм купівлі - {sum_buy_xlm} xlm\n'
        return text
```

Б.34 Лістинг функції «callback»

```
@bot.callback_query_handler(func=lambda call: True)
def callback(call):
    global authorization_state, bot_state
    if call.message:
        if call.data == 'authorization':
            if not authorization_state:
                msg = bot.edit_message_text(chat_id=call.message.chat.id,
message_id=call.message.id,
                                            text='🔑 Введіть секретний ключ')
                bot.register_next_step_handler(msg, take_private_key)

        if call.data == 'wallet':
            if authorization_state:
                bot.edit_message_text(chat_id=call.message.chat.id, message_id=call.message.id,
                                      text='💰Особистий гаманець',
                                      reply_markup=MarkupInterface.markup_wallet)
        if call.data == 'balance':
            msg = bot.send_message(call.message.chat.id, 'Щоб отримати:'
                                                         '\n Баланс усіх активів - введіть "1"'
                                                         '\n Баланс одного активу - введіть його скорочену назву
(Наприклад: XLM)')
            bot.register_next_step_handler(msg, get_balance)
        if call.data == 'payment' or call.data == 'edit_payment':
            msg_Pk = bot.send_message(call.message.chat.id, "Введіть адресу отримувача")
            bot.register_next_step_handler(msg_Pk, get_payment_public_key)
        if call.data == 'done_payment':
            time.sleep(1)
            bot.send_message(call.message.chat.id, "✅ Транзакцію надіслано \nКошти будуть
отримані протягом 24 годин")
            text_markup = main_markup_menu(call.message.chat.id)
```

```
        bot.send_message(call.message.chat.id, text=text_markup,
reply_markup=MarkupInterface.markup_main)

    if call.data == 'state':
        if authorization_state:
            bot.edit_message_text(chat_id=call.message.chat.id, message_id=call.message.id,
                                  text='🧹 Режим роботи',
                                  reply_markup=MarkupInterface.markup_state)
    if call.data == 'state_change_orders':
        if authorization_state:
            bot.edit_message_text(chat_id=call.message.chat.id, message_id=call.message.id,
                                  text='☑Увімкнено режим редагування угод')
            bot_state = 'Редагування угод'
            text_markup = main_markup_menu(call.message.chat.id)
            bot.send_message(chat_id=call.message.chat.id, text=text_markup,
                             reply_markup=MarkupInterface.markup_main)
    if call.data == 'state_tracking':
        if authorization_state:
            bot.edit_message_text(chat_id=call.message.chat.id, message_id=call.message.id,
                                  text='☑Увімкнено режим відстеження подій')
            bot_state = 'Відстеження подій'
            text_markup = main_markup_menu(call.message.chat.id)
            bot.send_message(call.message.chat.id, text=text_markup,
reply_markup=MarkupInterface.markup_main)

    if call.data == 'process':
        if authorization_state:
            bot.edit_message_text(chat_id=call.message.chat.id, message_id=call.message.id,
                                  text=f'⏱ Обробка даних ({bot_state})',
                                  reply_markup=MarkupInterface.markup_process)
    if call.data == 'process_go':
        if authorization_state:
            if bot_state == 'Редагування угод':
                go(call.message)
            elif bot_state == 'Відстеження подій':
                button_stop_process = types.KeyboardButton('⛔ Зупинити відстеження')
                markup = types.ReplyKeyboardMarkup(resize_keyboard=True,
one_time_keyboard=True).add(
                    button_stop_process)
                bot.edit_message_text(chat_id=call.message.chat.id, message_id=call.message.id,
                                      text='🚀 Починаємо обробку даних')
                bot.send_message(call.message.chat.id, 'Для зупинки натисність кнопку в області
клавіатури',
                                 reply_markup=markup)

    if call.data == 'search_on_domain':
        if authorization_state:
            msg = bot.send_message(call.message.chat.id, '🔗 Введіть адресу домена')
            bot.register_next_step_handler(msg, take_domain_address)

    if call.data == 'change_ts':
        if authorization_state:
```

```
            bot.edit_message_text(chat_id=call.message.chat.id, message_id=call.message.id,
                                  text='📝 Зміна параметрів ТС',
                                  reply_markup=MarkupInterface.markup_change_ts)
        if call.data == 'change_ts_domain':
            if authorization_state:
                bot.edit_message_text(chat_id=call.message.chat.id, message_id=call.message.id,
                                      text=f'Параметри ТС (пошук за доменом):\n' \
                                           f'  • вигода - x{domain_profit}\n' \
                                           f'  • об`єм купівлі - {sum_buy_xlm} xlm\n'
                                           f'Для редагування оберіть бажаний параметр нижче',
                                      reply_markup=MarkupInterface.markup_change_ts_domain)
        if call.data == 'ts_domain_profit':
            prm_profit = bot.edit_message_text(chat_id=call.message.chat.id,
message_id=call.message.id,
                                               text='Вкажіть число бажаної вигоди')
            bot.register_next_step_handler(prm_profit, get_ts_domain_profit)
        if call.data == 'ts_domain_total':
            prm_total = bot.edit_message_text(chat_id=call.message.chat.id,
message_id=call.message.id,
                                              text='Вкажіть число бажаного об`єму купівлі (xlm)')
            bot.register_next_step_handler(prm_total, get_ts_domain_total)
        if call.data == 'ts_domain_back' or call.data == 'ts_edit_back':
            bot.edit_message_text(chat_id=call.message.chat.id, message_id=call.message.id,
                                  text='📝 Зміна параметрів ТС',
                                  reply_markup=MarkupInterface.markup_change_ts)

        if call.data == 'change_ts_order':
            if authorization_state:
                bot.edit_message_text(chat_id=call.message.chat.id, message_id=call.message.id,
                                      text=f'Параметри ТС (редагування угод):\n' \
                                           f'  Купівля:\n' \
                                           f'  • вигода - x{buy_profit_coeff}\n' \
                                           f'  • різниця об`єму - {buy_amount_difference_coeff} xlm\n' \
                                           f'  Продаж:\n' \
                                           f'  • вигода - x{sell_profit_coeff}\n' \
                                           f'  • різниця об`єму - {sell_amount_difference_coeff} xlm\n',
                                      reply_markup=MarkupInterface.markup_change_ts_edit)
        if call.data == 'ts_edit_buy':
            bot.edit_message_text(chat_id=call.message.chat.id, message_id=call.message.id,
                                  text=f'Параметри ТС (редагування угод):\n' \
                                       f'  Купівля:\n' \
                                       f'  • вигода - x{buy_profit_coeff}\n' \
                                       f'  • різниця об`єму - {buy_amount_difference_coeff} xlm\n'
                                       f'Для редагування оберіть бажаний параметр нижче',
                                  reply_markup=MarkupInterface.markup_change_ts_edit_buy)
        if call.data == "ts_edit_sell":
            bot.edit_message_text(chat_id=call.message.chat.id, message_id=call.message.id,
                                  text=f'Параметри ТС (редагування угод):\n' \
                                       f'  Продаж:\n' \
                                       f'  • вигода - x{sell_profit_coeff}\n' \
                                       f'  • різниця об`єму - {sell_amount_difference_coeff} xlm\n'
                                       f'Для редагування оберіть бажаний параметр нижче',
```

```
                    reply_markup=MarkupInterface.markup_change_ts_edit_sell)
    if call.data == 'ts_edit_buy_profit':
        prm_profit = bot.edit_message_text(chat_id=call.message.chat.id,
message_id=call.message.id,
                                           text='Вкажіть число бажаної вигоди')
        bot.register_next_step_handler(prm_profit, get_ts_edit_buy_profit)
    if call.data == 'ts_edit_sell_profit':
        prm_profit = bot.edit_message_text(chat_id=call.message.chat.id,
message_id=call.message.id,
                                           text='Вкажіть число бажаної вигоди')
        bot.register_next_step_handler(prm_profit, get_ts_edit_sell_profit)
    if call.data == 'ts_edit_buy_amount':
        prm_amount = bot.edit_message_text(chat_id=call.message.chat.id,
message_id=call.message.id,
                                           text='Вкажіть число бажаного об`єму купівлі (xlm)')
        bot.register_next_step_handler(prm_amount, get_ts_edit_buy_amount)
    if call.data == 'ts_edit_sell_amount':
        prm_amount = bot.edit_message_text(chat_id=call.message.chat.id,
message_id=call.message.id,
                                           text='Вкажіть число бажаного об`єму купівлі (xlm)')
        bot.register_next_step_handler(prm_amount, get_ts_edit_sell_amount)
    if call.data == 'ts_edit_buy_back' or call.data == 'ts_edit_sell_back':
        bot.edit_message_text(chat_id=call.message.chat.id, message_id=call.message.id,
                              text=f'Параметри ТС (редагування угод):\n' \
                                   f'  Купівля:\n' \
                                   f'  • вигода - x{buy_profit_coeff}\n' \
                                   f'  • різниця об`єму - {buy_amount_difference_coeff} xlm\n' \
                                   f'  Продаж:\n' \
                                   f'  • вигода - x{sell_profit_coeff}\n' \
                                   f'  • різниця об`єму - {sell_amount_difference_coeff} xlm\n',
                              reply_markup=MarkupInterface.markup_change_ts_edit)

    if call.data == 'back_main':
        if authorization_state:
            text_markup = main_markup_menu(call.message.chat.id)
            bot.edit_message_text(chat_id=call.message.chat.id, message_id=call.message.id,
                                  text=text_markup,
                                  reply_markup=MarkupInterface.markup_main)

    if call.data == 'logout':
        if authorization_state:
            authorization_state = False
            bot.edit_message_text(chat_id=call.message.chat.id, message_id=call.message.id,
                                  text='🔒 Вихід із гаманця виконано')
```

Додаток В

Екранні форми

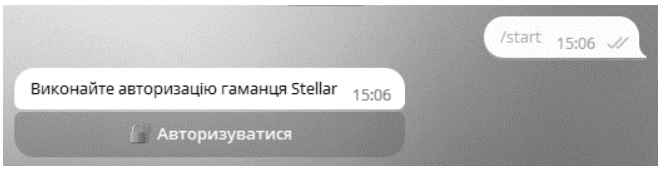

Рисунок В.1 – Повідомлення про авторизацію користувача

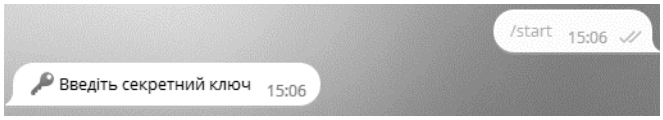

Рисунок В.2 – Загальний вигляд зміненого повідомлення «Авторизація»

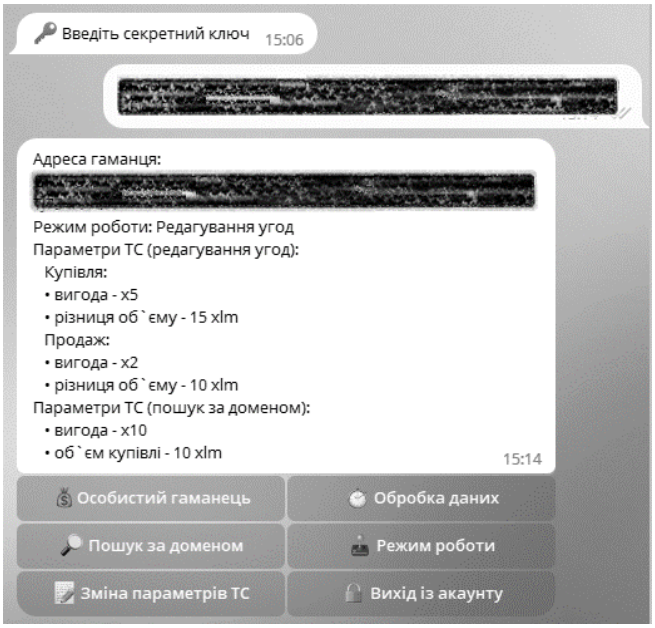

Рисунок В.3 – Успішна аторизація користувача

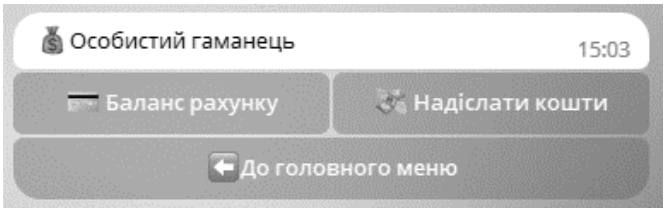

Рисунок В.4 – Меню «Особистий гаманець»

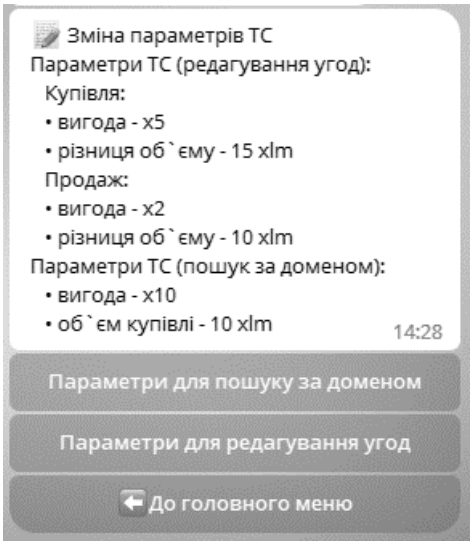

Рисунок В.5 – Меню «Зміна параметрів ТС»

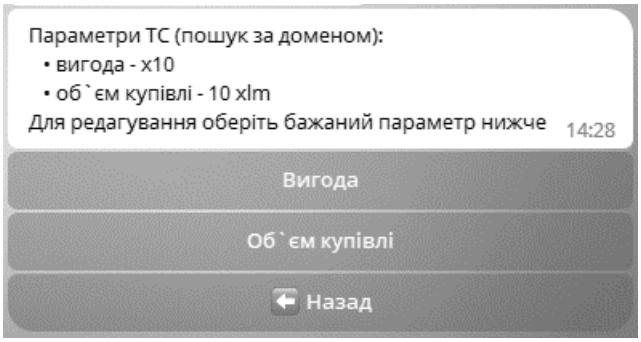

Рисунок В.6 – Меню налаштування параметрів ТС пошуку за доменом

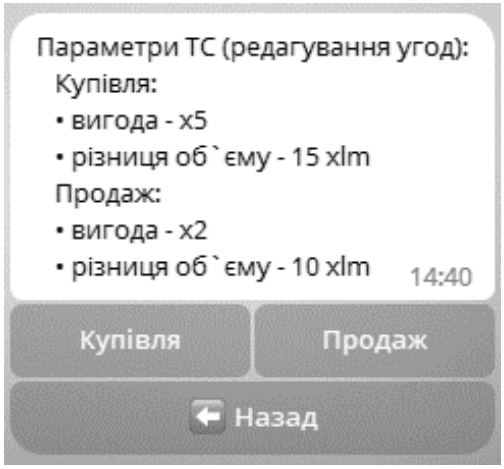

Рисунок В.7 – Меню налаштування параметрів ТС редагування угод

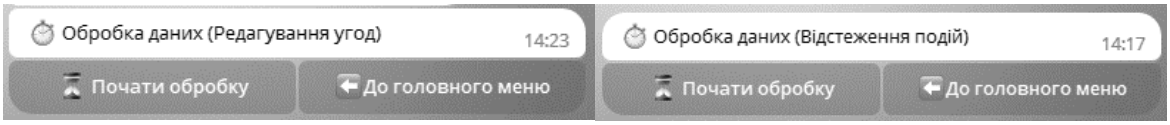

Рисунок В.8 – Меню «Обробка даних»

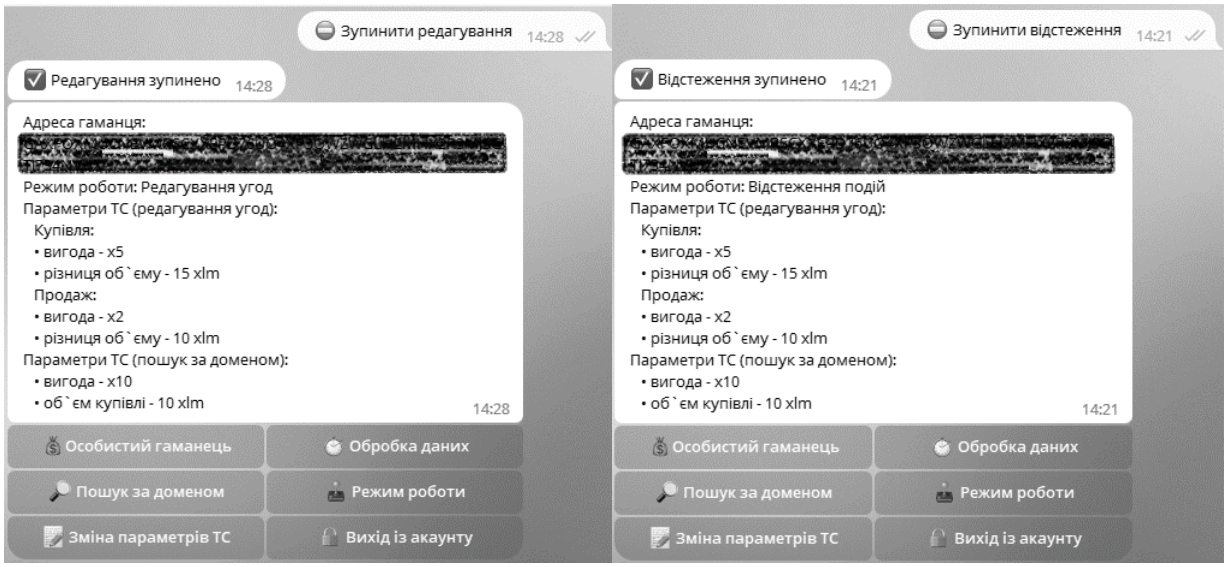

Рисунок В.9 – Зупинка редагування/відстеження подій АСКК

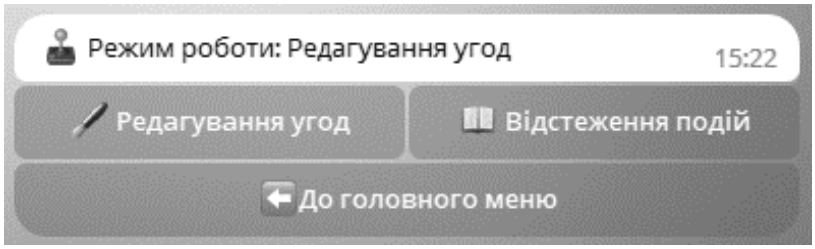

Рисунок В.10 – Меню «Режим роботи»

Додаток Г

Матеріали презентації