Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Факультет інформаційних технологій
(факультет)

Кафедра системного аналізу та управління
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеню бакалавра

Студентки *Іванчика Данiїла*

академічної групи *124-19-1*

спеціальності *124 Системний аналіз*

на тему: *«Автоматизація торгової діяльності на ринку децентралізованих невзаємозамінних активів»*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*к.ф.-м.н., доц. Коряшкіна Л.С.*			
розділів:				
Інформаційно-аналітичний розділ	*к.ф.-м.н., доц. Коряшкіна Л.С.*			
Спеціальний розділ	*к.ф.-м.н., доц. Коряшкіна Л.С.*			
Рецензент				
Нормоконтролер	*к.ф.-м.н., доц. Хом'як Т.В.*			

Дніпро
2023

ЗАТВЕРДЖЕНО:
завідувач кафедри

Системного аналізу та управління
(повна назва)

к.т.н., доц. Желдак Т.А.
(підпис) (прізвище, ініціали)

«____» _____ 20 ___ року

ЗАВДАННЯ

на кваліфікаційну роботу ступеню *бакалавра*

студента *Іванчика Д.* **академічної групи** *124-19-1*
спеціальності: *124 Системний аналіз*
на тему *«Автоматизація торгової діяльності на ринку децентралізованих невзаємозамінних активів»*
затверджену наказом ректора НТУ «Дніпровська політехніка»
від *16.05.2023 № 350-с*

Розділ	Зміст виконання	Термін виконання
1. Інформаційно-аналітичний	*Основи блокчейну, маркетплейс Magic Eden та блокчейн Solana*	20.03.23-01.04.23
2. Спеціальний	*Проектування та реалізація програмного зебеспечення.*	01.04.23-14.05.23
3. Експерементально-аналітичний	*Робота програмного зебеспечення та аналіз статистичних даних*	14.05.23-10.06.23

Завдання видано _____ *доц. Коряшкіна Л.С.*
(підпис керівника) (прізвище, ініціали)

Дата видачі: 15.03.2023

Дата подання до екзаменаційної комісії: 13.06.23

Прийнято до виконання _____ Іванчик Д.
(підпис студента) (прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 54 с., 7 рис., 2 табл., 4 додатки, 20 джерел.

Об’єктом дослідження в роботі є процес автоматизації торгової діяльності на ринку NFT на блокчейні Solana та платформі Magic Eden. Включаючи аналіз ринку, автоматичне виявлення та покупку вигідних активів, контроль за цінами та управління портфелем активів.

Предметом дослідження програмне забезпечення, яке забезпечує автоматизацію торгової діяльності на ринку NFT на блокчейні Solana та на платформі Magic Eden. Це включає розробку алгоритмів для автоматичного виявлення та аналізу вигідних активів, реалізацію механізмів автоматичної покупки та управління портфелем, а також інтерфейс користувача для зручного керування та моніторингу процесу торгівлі.

Завдання дослідження: проектування та розробка легко масштабованого програмного забезпечення здатного приносити прибуток.

Метою даної кваліфікаційної роботи є розробка програмного забезпечення для автоматизації торгової діяльності на ринку децентралізованих невзаємозамінних активів (NFT) на блокчейні Solana та на платформі Magic Eden. Основним завданням є створення ефективного інструменту, який допоможе автору здійснювати швидкі та прибуткові операції з NFT.

Методи дослідження: методи об’єктноорієнтованого аналізу та проектування, принципи об'єктно-орієнтованого програмування на мові Java.

В *інформаційно-аналітичному розділі* розглянуто основні поняття стосовно технологій блокченйу, NFT та галузі функціонування програмного забезпечення та проведено аналіз цього ринку.

У *спеціальному розділі* застосовано стратегії до розробки програмного забезпечення з використанням патернів моделювання.

У *експерементально-аналітичному* розділі наведено результати роботи програмного забезпечення та проведено їх аналіз.

Практична цінність полягає в тому, що розроблене програмне забезпечення для автоматизації торгової діяльності на ринку NFT на блокчейні Solana та на платформі Magic Eden надає конкретні переваги та можливості. Воно дозволяє значно підвищити ефективність його торгівельних операцій, зменшити час, необхідний для виявлення та покупки вигідних активів, та забезпечити точний контроль над цінами та аналізом ринку

Ключові слова: NFT, SOLANA, MAGIC EDEN, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, JAVA, STATE, ПАТЕРН, ПРИБУТОК.

ABSTRACT

Explanatory note:: 54 pages, 7 figures, 2 tables, 4 appendices, 20 sources.

The object of research in this paper is the process of automating trading activities in the NFT market on the Solana blockchain and the Magic Eden platform. This includes market analysis, automatic detection and purchase of profitable assets, price monitoring, and portfolio management.

The subject of research is the software that enables the automation of trading activities in the NFT market on the Solana blockchain and the Magic Eden platform. This involves the development of algorithms for automatic detection and analysis of profitable assets, implementation of mechanisms for automatic purchasing and portfolio management, as well as a user interface for convenient control and monitoring of the trading process.

The research objective is the design and development of scalable software capable of generating profits. The aim of this qualification work is to develop software for automating trading activities in the market of decentralized non-fungible assets (NFT) on the Solana blockchain and the Magic Eden platform.

The main task is to create an efficient tool that will help the author carry out fast and profitable NFT operations. Research methods: object-oriented analysis and design methods, principles of object-oriented programming in Java.

The information-analytical section discusses the main concepts related to blockchain technology, NFT, and the functioning of software in this industry, and conducts an analysis of this market.

In the special section, strategies for software development using design patterns are applied.

The experimental-analytical section presents the results of the software's work and conducts their analysis.

The practical value lies in the fact that the developed software for automating trading activities in the NFT market on the Solana blockchain and the Magic Eden platform provides specific advantages and capabilities. It significantly improves the

efficiency of trading operations, reduces the time required to identify and purchase profitable assets, and ensures accurate price control and market analysis.

Keywords: NFT, SOLANA, SOFTWARE, JAVA, STATE, PATTERN, PROFIT.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Public key (публічний ключ) — криптографічне цифрове значення, доступне всім користувачам. В термінах блокчейну являє собою номер гаманця, на який необхідно переводити цифрову валюту, або, у випадку смарт-контрактів, адреса сторони, яка зафіксувала домовленість.

Private key (приватний ключ) — криптографічне цифрове значення, доступне лише власнику. За допомогою приватного ключа здійснюється цифровий підпис транзакції

NFT (Non-Fungible Token) - це особливий тип криптовалютного токена, який використовується для представлення унікальних цифрових активів, таких як мистецькі твори, музика, відео, власність в грі та інше. Відмінністю NFT є те, що вони є неповторними та не можуть бути заміненими одне на одного, оскільки кожен NFT має свій унікальний ідентифікатор. Це дозволяє встановлювати власність та автентичність цифрових активів, а також створює нові можливості для торгівлі та зберігання цифрових майнових цінностей. NFT базуються на технології блокчейн, зазвичай на основі мережі Ethereum, що забезпечує їх надійність, безпеку та переносимість між різними платформами

API – application programming interface

SDK - software development kit

Blockchain (блокчейн) — архітектурна концепція збереження інформації у формі блоків, пов’язаних один із одним

Block — одиниця блокчейну, яка включає в собі набір транзакцій

Transaction — одиниця блоку, яка зберігає корисну інформацію

Smart contract (розумний контракт) — інформаційна одиниця, яка зберігає в собі дані про домовленості між сторонами

JSON (JavaScript Object Notation) - це легкий формат обміну даними, який використовується для передачі структурованої інформації між різними програмними системами. Він базується на синтаксисі мови JavaScript, але є

незалежним від конкретної мови програмування. Формат JSON зручний для людей та легко зрозумілий комп'ютерами. JSON представляє дані у вигляді пар «ключ-значення», де ключі є рядками, а значення можуть бути різних типів, таких як рядки, числа, булеві значення, масиви або навіть вкладені об'єкти. Він підтримує прості структури даних, що дозволяє легко обробляти та передавати інформацію

ПЗ – праграмне забезпечення

API – прикладний програмний інтерфейс

ВСТУП

Сучасний розвиток блокчейн технології сприяє створенню нових можливостей у фінансовому секторі. Одним із найбільш обговорюваних напрямків є ринок децентралізованих невзаємозамінних активів (NFT) [2], який набуває значної популярності серед інвесторів та колекціонерів активів.

У рамках цього ринку з'являється потреба у автоматизації торгової діяльності. Особливо, якщо мова йде про ринок NFT на блокчейні Solana та платформі Magic Eden [8]- двох важливих гравців, які забезпечують ефективну інфраструктуру для торгівлі та обміну цифровими активами.

Актуальність обраної теми полягає у потребі вдосконалення процесів торгівлі на ринку децентралізованих NFT активів на блокчейні Solana [9]. Цей ринок знаходиться в стадії активного розвитку, привертаючи увагу інвесторів та творчих спільнот. Забезпечення ефективної та безпечної торгівлі на цьому ринку стає важливим завданням.

Автоматизація торгової діяльності на ринку NFT на блокчейні Solana та платформі Magic Eden включає в себе розробку спеціалізованого програмного забезпечення, яке допомагає управляти та оптимізувати процеси торгівлі. Це може включати автоматичне виконання угод, управління портфелем активів, аналіз ринкових умов та забезпечення безпеки транзакцій.

Однак, у контексті розробки програмного забезпечення, що спрямоване на автоматизацію торгової діяльності на ринку NFT, виникають питання щодо обмежень доступу до цього рішення. У даному випадку, цей проект розробляється суто для власного використання автором і не передбачається його розповсюдження.

Торгівля на ринку NFT на блокчейні Solana та платформі Magic Eden вимагає швидкого реагування на зміни у цінах та попиті на активи. Ручна торгівля може бути важкою і витратною, особливо при великому обсязі

операцій. Тому автоматизація торгівельного процесу стає важливим фактором успіху для інвесторів та трейдерів.

Мета роботи – розробка програмного забезпечення для автоматизації торгової діяльності на ринку децентралізованих невзаємозамінних активів (NFT) на блокчейні Solana та на платформі Magic Eden. Основним завданням є створення ефективного інструменту, який допоможе автору здійснювати швидкі та прибуткові операції з NFT.

Об’єкт дослідження – процес автоматизації торгової діяльності на ринку NFT на блокчейні Solana та платформі Magic Eden. Включаючи аналіз ринку, автоматичне виявлення та покупку вигідних активів, контроль за цінами та управління портфелем активів.

Предмет дослідження – програмне забезпечення, яке забезпечує автоматизацію торгової діяльності на ринку NFT на блокчейні Solana та на платформі Magic Eden. Це включає розробку алгоритмів для автоматичного виявлення та аналізу вигідних активів, реалізацію механізмів автоматичної покупки та управління портфелем, а також інтерфейс користувача для зручного керування та моніторингу процесу торгівлі.

Практична значущість полягає в тому, що розроблене програмне забезпечення для автоматизації торгової діяльності на ринку NFT на блокчейні Solana та на платформі Magic Eden надає конкретні переваги та можливості. Воно дозволяє значно підвищити ефективність його торгівельних операцій, зменшити час, необхідний для виявлення та покупки вигідних активів, та забезпечити точний контроль над цінами та аналізом ринку.

Завдяки автоматизації, автор може отримати перевагу в конкурентному середовищі ринку NFT, реагуючи швидше на зміни у цінах та попиті на активи. Програмне забезпечення допомагає знизити ризики помилок, пов'язаних з ручним введенням та обробкою даних, та забезпечує автоматичне виконання стратегій торгівлі, що дозволяє уникнути емоційних рішень та покращити результативність.

Крім того, зосередження автора на стратегічних аспектах торгівлі, замість витрати часу на рутинні задачі, сприяє підвищенню ефективності його діяльності та може привести до більш значних прибутків. Також, розроблене програмне забезпечення може стати основою для подальшого дослідження та розвитку торгівельних стратегій на ринку NFT, сприяючи вдосконаленню процесу торгівлі та підвищенню професійного рівня автора.

РОЗДІЛ 1
ІНФОРМАЦІЙНО-АНАЛІТИЧНИЙ

1.1 Вступ до блокчейн технології

Вступ до блокчейн технології є важливим кроком у розумінні сутності та потенціалу цього інструменту. У сучасному цифровому світі блокчейн вважається одним з найбільш обговорюваних і перспективних технологічних розвитків.

Блокчейн - це розподілена, незмінна та безпечна система зберігання даних, яка працює на основі криптографічних принципів [1]. Він використовується для створення децентралізованого реєстру, в якому інформація зберігається у вигляді блоків, які ланцюгово пов'язані між собою.

Однією з основних особливостей блокчейну є його децентралізована природа. У традиційних системах зберігання даних, таких як банки або централізовані сервери, контроль та управління знаходяться у руках центральної організації. В блокчейні ж кожен учасник мережі має копію повного реєстру та спільно співпрацює у процесі його оновлення та підтвердження транзакцій. Це створює безпеку, надійність та незмінність даних, оскільки будь-яка спроба зміни існуючих блоків стає помітною для всіх учасників мережі.

Крім того, блокчейн забезпечує прозорість операцій. Кожна транзакція відслідковується та фіксується в блокчейні, що дозволяє всім зацікавленим сторонам перевірити їх легітимність та достовірність. Це особливо важливо в сферах, де потрібно перевыряти надійність та автентичність даних, наприклад, у фінансових транзакціях чи управлінні постачанням ліків.

Останнім часом блокчейн знайшов своє застосування в різних галузях, включаючи фінанси, логістику, медицину, нерухомість та мистецтво.

Найвідомішим прикладом використання блокчейну є криптовалюта, така як Bitcoin, яка дозволяє безпечні та швидкі фінансові операції без посередників.

Крім того, в блокчейні можуть бути створені та обмінюватися децентралізовані невзаємозамінні активи (NFT). Це унікальні цифрові активи, які можуть представляти мистецтво, музику, власність, віртуальні предмети гри та інше. Блокчейн дозволяє створювати, власність та торгувати цими активами безпосередньо між учасниками мережі, забезпечуючи автентичність та непідробність кожного активу.

У цьому розділі буде детально розглянемо основні принципи блокчейн технології, роль розподіленого реєстру, застосування блокчейну в фінансових сферах та введемо в поняття децентралізованих невзаємозамінних активів (NFT). Ви отримаєте чітке уявлення про функціонування блокчейну та його потенціал у різних сферах життя.

Цей розділ спрямований на слухачів, які не мають попереднього досвіду з блокчейн технологією та NFT, тому надається вичерпна та доступна інформація, щоб допомогти вам зрозуміти ці концепції та зробити перші кроки в світі блокчейну та NFT. Після ознайомлення з цим розділом ви будете мати міцні основи для подальшого вивчення та розуміння автоматизації торгової діяльності на ринку децентралізованих невзаємозамінних активів на блокчейні Solana та платформі Magic Eden.

У наступних розділах буде розглянуто розглянемо практичний аспект розробки програмного забезпечення для автоматизації торгової діяльності на ринку NFT на блокчейні Solana. Буде розглянуто ключові етапи розробки, включаючи аналіз вимог, проектування архітектури, реалізацію функціональності та тестування. Крім того, буде розглянуто особливості мови програмування Java та її застосування у розробці програмного забезпечення.

1.2 Структура блокчейну, криптографія та NFT

Зв'язані поняття блокчейну, криптографії та NFT [4] мають велике значення в сфері криптовалют та цифрових активів. Давайте розглянемо кожен з них окремо:

Блокчейн: Блокчейн - це розподілена технологія зберігання та передачі інформації, яка базується на концепції ланцюжка блоків. Кожен блок у ланцюжку містить набір транзакцій, які були підтверджені та засвідчені мережею. Кожен новий блок містить посилання на попередній блок, що утворює неперервний ланцюжок.

Структура блокчейну включає такі складові:

1) блоки: кожен блок містить дані про транзакції, хеш (унікальний ідентифікатор блоку) та посилання на попередній блок;

2) хеш: це унікальний ідентифікатор блоку, який генерується на основі даних блоку. Він дозволяє впізнати будь-які зміни в блоку;

3) децентралізація: блокчейн зазвичай працює на основі децентралізованої мережі комп'ютерів, відомих як вузли, які спільно підтримують та підтверджують транзакції;

4) криптографія: блокчейн використовує криптографічні методи для захисту даних, забезпечення безпеки та перевірки транзакцій.

Криптографія: Криптографія - це наука про захист інформації шляхом шифрування та розшифрування даних. У контексті блокчейну криптографія використовується для забезпечення безпеки, конфіденційності та цілісності даних.

Деякі основні аспекти криптографії [1], що використовуються в блокчейні, включають:

1) хеш-функції: Хеш-функції використовуються для створення унікального хешу (хеш-коду) з даних будь-якого розміру. В блокчейні хеш-функції застосовуються для створення ідентифікаторів блоків та транзакцій. Зміна навіть найменшого фрагмента даних призведе до зміни хешу, що дозволяє виявити будь-які спроби змінити дані;

2) публічний ключ і приватний ключ: Криптографія з використанням публічного та приватного ключів використовується для забезпечення конфіденційності та автентифікації транзакцій. Публічний ключ використовується для шифрування даних, які тільки власник приватного ключа може розшифрувати. Приватний ключ використовується для розшифрування даних та підпису транзакцій, що підтверджує автентичність власника;

3) криптографічні підписи: Криптографічні підписи використовуються для підтвердження автентичності та цілісності даних. Вони генеруються за допомогою приватного ключа і перевіряються за допомогою публічного ключа. У блокчейні криптографічні підписи застосовуються для підтвердження авторства транзакцій та забезпечення неможливості їх подальшої зміни.

Non fungible token або NFT [2] є цифровими активами, які використовують технологію блокчейн для забезпечення їх унікальності, автентичності та незмінності. Основні риси NFT включають:

Унікальність: кожен NFT має унікальний ідентифікатор, що відрізняє його від інших токенів. це дозволяє точно встановити власника та характеристики конкретного цифрового активу.

Власність: NFT дозволяють відстежувати та підтверджувати власність цифрового активу. Завдяки технології блокчейну можна точно визначити, кому належить певний NFT, і забезпечити автентичність власництва.

Незмінність: Блокчейн забезпечує незмінність записів, що стосуються NFT. Це означає, що інформація про NFT і його власника не може бути змінена або втрачена.

Цифрові активи, які можуть бути представлені у вигляді NFT, можуть бути різними: мистецтво, музика, відео, графічні зображення, віртуальні предмети в іграх тощо. Власність NFT дає можливість власникам використовувати, продавати, обмінювати або показувати цифрові активи в онлайн просторі, відкриваючи нові можливості для творців і колекціонерів.

Останнім часом NFT [8] набули великої популярності, особливо в області мистецтва і розважальної індустрії. Вони створюють нові шляхи для артистів,

музикантів та інших творчих особистостей отримати визнання, продавати свої твори безпосередньо покупцям і взаємодыяти з ними.

NFT також відкривають можливості для творців отримувати винагороду за використання їхніх творів у різних контекстах, таких як відеоігри, віртуальна реальність, колекціонування тощо.

Однією з ключових переваг NFT є їх потенціал для забезпечення прозорості та простежуваності. Завдяки технології блокчейн, історія транзакцій з NFT може бути переглянута публічно, що забезпечує довіру та доказує автентичність активу.

Однак, варто відзначити, що світ NFT також супроводжується деякими викликами і питаннями. Наприклад, проблеми енергоефективності, пов'язані з майнінгом криптовалют, що базуються на блокчейні, а також питання щодо авторських прав і власності над цифровими активами, які можуть бути невіддільними від NFT.

Незважаючи на це, NFT залишається цікавою інноваційною галуззю, яка відкриває нові можливості для цифрової власності, творчості та торгівлі. Це допомагає перетворити цифрові активи на унікальні цінності, які можуть бути володіні, обмінювані та підтримані за допомогою технології блокчейну.

1.3 Блокчейн Solana та маркетплейс Magic Eden

Блокчейн Solana [9] є одним з передових блокчейнів у розвитку NFT. Його технологічні переваги дають користувачам та розробникам сприятливі умови для розвитку та будиівництва своїх проектів на цьому блокчені. Поговоримо про це докладніше.

Solana є швидким, масштабованим і високоефективним блокчейном, який надає ряд переваг для NFT ком'юніті та розробників, які планують запустити свій NFT проект. Ось основні переваги Solana:

1) швидкість та масштабованість: Solana може обробляти тисячі транзакцій в секунду з низькими комісіями. Це особливо важливо для NFT

проектів, оскільки велика кількість користувачів може одночасно взаємодіяти з маркетплейсами та торгувати NFT. Швидкість Solana дозволяє уникнути перевантажень мережі та забезпечує низьку латентність;

2) низькі комісії: Solana має низькі комісії за транзакції порівняно з іншими блокчейнами, такими як Ethereum. Це робить Solana більш доступною для користувачів, особливо при великому обсязі торгівлі NFT або взаємодії з маркетплейсами. У випадку автора, цей фактор є особливо важливим для комфортного тестування розробленого програмного забезпечення;

3) екосистема та інтеграції: Solana має розширену екосистему, включаючи різноманітні інструменти розробки, SDK та додатки. Це дозволяє розробникам легко створювати та розгортати свої NFT проекти на блокчейні Solana. Крім того, Solana також підтримує інтеграцію з різними зовнішніми сервісами та платформами;

4) складні функції та програмування: Solana пропонує розширені можливості програмування, що дозволяє розробникам створювати складні функції та взаємодіяти з іншими смарт-контрактами. Це важливо для NFT проектів, які можуть вимагати виконання складних логічних умов, автоматизації торгівлі, створення різноманітних умовних договорів та багато іншого. Розширені можливості програмування Solana надають розробникам гнучкість і творчий контроль над їхніми NFT проектами;

5) спільнота та підтримка: Solana має активну та зростаючу спільноту розробників та ентузіастів. Це означає, що завжди є доступ до підтримки, документації та ідеїв спільноти. Також, заряджений спільнотою екосистеми Solana стимулює інновації та співпрацю між розробниками, що може позитивно вплинути на розвиток NFT проекту;

6) екосистема DeFi: Solana є популярною платформою для розробки децентралізованих фінансових (DeFi) додатків. Це дає можливість NFT проектам використовувати додаткові фінансові інструменти, такі як обмін токенів, ліквідність та інші DeFi протоколи, щоб створити більш розширені функціональні можливості та інтеграції.

В цілому, блокчейн Solana надає швидкість, масштабованість, низькі комісії, розширені можливості програмування, сильну спільноту та підтримку, а також доступ до екосистеми DeFi. Це робить Solana привабливим вибором для NFT ком'юніті та розробників, які прагнуть запустити успішний та ефективний NFT проект. Мій вибір пав саме на Solana через зручний api для роботи з блокченом та ще більш зручний маркетплейс – Magic Eden. Який надає усі необхідні перевагии для автоматизації торгової діяльності. Далі детальніше поговоримо саме про нього.

Magic Eden - це децентралізований NFT-маркетплейс на блокчейні Solana. Маркетплейс був запущений 21 вересня 2021 р.. Розробкою проекту займається команда спеціалістів, з великим досвідом роботи в таких компаніях, як Uber і Meta (колишня Facebook). Генеральним директором платформи є розробник на ім'я Джек Лю, а технічним директором — Сідней Чжан. Проекту вдалося неодноразово залучати значні фінансові кошти. Наприклад, у березні 2022 р. платформа отримала 27 млн доларів. Через кілька місяців, у червні 2022 р., проект залучив 130 млн доларів при оцінці в 1,6 млрд.

Ідея Magic Eden полягає в тому, щоб зробити роботу з NFT якомога простішою. Для цього розробники представили інтуїтивно зрозумілий користувацький інтерфейс. Все, що потрібно зробити, щоб додати свій токен - завантажити файл на платформу. Система автоматично перетворить його в формат NFT.

На платформі Magic Eden є можливість:;

1) купувати та продавати невзаємозамінні токени;

2) відстежувати показники перспективних проектів;

3) відстежувати статистику ринку NFT;

4) вивчати матеріали, пов'язані з ринком невзаємозамінних токенів.

Зручний API та велика популярність Magic Eden роблять очевидним вибір маркетплейсу для розробки даного програмного забезпечення. Через великі обсяги торгів, з платформи легко можна збирати великі обсяги інформації та

зрусно пряцювати з нею. Саме через це, вибір був зроблений у бік Magic Eden, без розгляду його конкурентів.

API Magic Eden розроблено з метою допомогти розробникам створювати інноваційні додатки, використовуючи NFT та наші обширні дані про ринок. Набір потужних ендпоінтів надає доступ до метаданих NFT та основних компонентів маркетплейсу, включаючи події, колекції, лоти, заявки та багато іншого. Це дає можливість не лише отримувати дані, але й здійснювати різні дії, які зазвичай виконуються через користувацький інтерфейс, наприклад, розміщувати NFT, купувати їх, робити ставки тощо.

Висновок

Отже, у цьому розділі було розібрано основні теоретичні відомості про блогчейн, його структуру та сфери застосування. Обговорили блокчейн Solana, який має кілька ключових переваг для NFT ком'юніті та розробників. Він пропонує швидкість та масштабованість, низькі комісії, розширені можливості програмування, сильну спільноту та підтримку, а також доступ до екосистеми DeFi. Ці переваги роблять Solana привабливим вибором для тих, хто планує створити та розвивати свій NFT проект. Все це допомагає забезпечити ефективність, гнучкість та інноваційні можливості для успішного функціонування та розвитку NFT проектів на блокчейні Solana. Також було проаналізовано маркетплейс Magic Eden саме під який й розраблялося програмне забезпечення.

РОЗДІЛ 2
СПЕЦІАЛЬНИЙ

2.1 Інструменти для розробки програмного забезпечення

При створенні системи важливе значення мають обрані технології реалізації. Для проектування системи була обрана мова програмування Java [20], яка має велику кількість стандартних бібліотек з інкапсульованим функціоналом. Однією з основних причин вибору саме цієї мови програмування є бажання автора вдосконалити свої навички розробки на Java.

Окрім стандартної JDK (Java Development Kit) в проекті використовується багато інших бібліотек [10] та SDK [7] для роботи з блокчейном Solana – SolanaJ. Solana blockchain client, написаний на чистій Java. Solanaj є API для інтеграції з блокчейном Solana за допомогою API RPC Solana [5].

RPC, або Remote Procedure Call — це протокол комунікації між різними процесами або комп'ютерами, який дозволяє викликати функції або процедури віддаленого комп'ютера і отримувати результати виконання цих функцій. В основі RPC лежить ідея зробити виклик функції на віддаленому комп'ютері аналогічним виклику локальної функції.

RPC дозволяє розподіленій системі використовувати функціональні можливості віддаленого комп'ютера так, наче вони знаходяться на локальному комп'ютері. Клієнтська програма викликає функцію на віддаленому сервері через мережу, і сервер виконує цю функцію і повертає результат клієнту. Вся ця комунікація відбувається через мережеві протоколи, такі як TCP/IP.

RPC може бути використаний у різних сферах, включаючи розподілені системи, веб-сервіси, мікросервісну архітектуру та інші. Він дозволяє зручно взаємодіяти між компонентами системи, що працюють на різних фізичних або віртуальних машинах.

У проекті наявні наступні бібліотеки [11]:

1) jackson;

2) lombok;

3) apache httpclient;

4) selenium.

Jackson - це потужна бібліотека для роботи з форматами обміну даними в Java, такими як JSON. Вона надає можливість зчитувати і записувати дані у форматі JSON з об'єктами Java і навпаки. Jackson є однією з найпопулярніших бібліотек для роботи з JSON в екосистемі Java.

Основні функції та можливості, які надає бібліотека Jackson:

1) простота використання: Jackson надає простий і зрозумілий API для роботи з JSON. Вона легко інтегрується з існуючими проектами на Java і не потребує багато коду для роботи з JSON-даними;

2) зчитування та запис JSON: Jackson надає можливість зчитувати JSON-дані з різних джерел, таких як рядки, файли або потоки. Він також може генерувати JSON-дані з об'єктів Java або структур даних;

3) підтримка анотацій: Jackson використовує анотації Java для налаштування процесу серіалізації та десеріалізації. Це дозволяє визначати специфічні правила серіалізації для полів або методів об'єктів Java;

4) гнучкі налаштування: Jackson надає різноманітні налаштування для керування процесом серіалізації та десеріалізації. Ви можете налаштувати формат дати, обробляти поліморфні типи даних, пропускати або ігнорувати певні поля тощо;

5) підтримка широкого спектру типів даних: Jackson може працювати з різними типами даних, включаючи прості типи (рядки, числа), колекції, масиви, об'єкти Java, вкладені об'єкти, дерева об'єктів тощо.;

6) висока продуктивність: Jackson працює досить швидко і має високу продуктивність. Вона має оптимізовані алгоритми парсингу і серіалізації JSON, що дозволяє швидко обробляти великі об'єми даних;

7) підтримка різних форматів: Jackson не обмежується тільки форматом JSON. Вона також підтримує інші формати, такі як XML, YAML і протоколи обміну даними, такі як Protocol Buffers і CBOR;

8) інтеграція з іншими бібліотеками: Jackson легко інтегрується з іншими бібліотеками і фреймворками Java, такими як Spring, Hibernate, JAX-RS, і допомагає спростити процес обробки даних.

Lombok - це бібліотека для мови програмування Java, яка пропонує аннотації для автоматичної генерації коду. Її основна мета - спростити розробку Java-програм, зменшити кількість повторюваного коду і покращити читабельність.

Apache HttpClient є популярною бібліотекою для виконання HTTP-запитів у Java-додатках. Вона надає потужні засоби для взаємодії з веб-серверами, виконання HTTP-запитів різних типів (GET, POST, PUT, DELETE і т. д.), обробки відповідей та управління сесіями.

Ось деякі ключові особливості та функціональні можливості Apache HttpClient:

1) підтримка протоколів: HttpClient підтримує HTTP/1.1 і HTTP/2, а також проксі-сервери, SSL/TLS-шифрування і автентифікацію;

2) конфігурація: HttpClient надає гнучкі налаштування через об'єкти конфігурації, такі як RequestConfig і HttpClientBuilder. Це дозволяє налаштувати параметри, такі як таймаути, обмеження підключення і обробка перенаправлень;

3) робота з запитами: Бібліотека дозволяє створювати різні типи запитів, встановлювати заголовки, передавати параметри та виконувати взаємодію з різними ресурсами. Вона підтримує розподілену роботу з додатками, таку як робота з файлами, обробка форм, передача файлів тощо;

4) обробка відповідей: HttpClient дозволяє отримувати відповіді у різних форматах, таких як рядки, байтові масиви, JSON-об'єкти тощо. Ви можете обробляти вихідні дані в зручний для вас спосіб;

5) керування сесіями: HttpClient підтримує управління сесіями, яке дозволяє зберігати стан між послідовними запитами. Ви можете встановлювати кукі, автентифікуватись і виконувати авторизовані запити;

6) пул підключень: HttpClient має пул підключень, що дозволяє ефективно використовувати ресурси і зменшувати затримки при виконанні багатьох послідовних запитів. Пул підключень дозволяє багаторазово використовувати вже встановлені з'єднання з сервером замість створення нових для кожного запиту;

7) асинхронні запити: HttpClient надає підтримку асинхронної взаємодії з веб-серверами. Ви можете виконувати запити асинхронно, що дозволяє ефективніше використовувати ресурси та покращує продуктивність додатка;

8) обробка помилок: HttpClient надає механізми для обробки помилок, таких як недоступність сервера, помилки взаємодії, перенаправлення і т. д. Ви можете виконати відповідну обробку помилок і прийняти необхідні дії.

Apache HttpClient є потужною бібліотекою, яка дозволяє зручно та ефективно виконувати HTTP-запити в Java-додатках. Вона широко використовується у багатьох проектах для взаємодії з веб-серверами та реалізації різноманітних HTTP-задач.

Selenium - це набір інструментів для автоматизації веб-браузерів. Він дозволяє розробникам і тестувальникам автоматично виконувати дії, які зазвичай виконуються веб-користувачами, такі як відкриття веб-сторінок, заповнення форм, натискання кнопок, навігація по сторінках тощо.

Selenium може бути корисним для автоматизації тестування веб-додатків, де потрібно перевірити, чи працює програма правильно в різних сценаріях. Він дозволяє встановити автоматичний контроль над браузером і перевірити, чи відображаються правильні дані, чи працюють функції, чи не виникають помилки та інші аспекти, які потрібно перевірити.

Крім тестування, Selenium також використовується для веб-скрапінгу, тобто автоматичного отримання даних з веб-сторінок. Це дозволяє отримувати

інформацію з веб-сайтів без необхідності вручну переглядати та копіювати дані.

Selenium у даному проекту використовується не у привичному вигляді, який був описаний вижче. Через особливості Magic Eden API, у звичайних користувачей немає можливості отримувати частину підпису транзакції напряму від маркетплейсу, тож було прийняте рішення про імітацію виконання запиту, для отримання її напряму від сайту сарвіса. Детальніше цей аспект буде описано у наступних розділах даної роботи.

Загалом, кожна з цих бібліотек являє собою дуже сильний інструмент, котрий у значній мірі спрощує розробку буд якого програмного забезпечення, даний проект не став виключенням.

2.2 Типи даних: огляд та аналіз

В даному розділі буде розглянуто основні аспекти роботи програмного забезпечення, отримання даних через формат JSON (JavaScript Object Notation) та їх аналіз, обробку та підписання транзакцій.

Дане програмне забезпечення розроблялося переважно для маркетплейса Magic Eden, який є веб застосунком. Одним з найпоширеніших форматів передачі даних, використованих веб-застосунками, є JSON (JavaScript Object Notation). JSON є легким у використанні та зрозумілому форматом, який дозволяє структурувати дані та передавати їх між клієнтською та серверною частинами веб-застосунку. Цей формат дозволяє програмістам легко серіалізувати та десеріалізувати об'єкти, що дозволяє ефективно передавати складні дані, такі як списки, об'єкти, числа, рядки та багато іншого. Використання JSON у веб-застосунках сприяє забезпеченню стандартизованого, зрозумілого та ефективного обміну даними між різними компонентами системи, роблячи їх більш масштабованими та інтегрованими.

Обраний маркетплей не став виключенням, Magic Eden API надає зручний набір інструментів для отримання даних у форматі JSON. Далі у цьому

розділі ми детальніше розглянемо дані які отримуємано у відповід на запит до сервера.

Слід зазначити, що Magec Eden Solana є одним з розповсюджених маркетплейсів, побудованих на основі клієнт-серверної архітектури. Цей маркетплейс використовує технологію блокчейн Solana для забезпечення безпеки, швидкості та масштабованості операцій. Давайте розглянемо основні компоненти архітектури Magec Eden Solana:

1) клієнтська частина: Клієнтська частина маркетплейса - це інтерфейс, доступний користувачам для взаємодії з маркетплейсом. Вона може бути представлена у вигляді веб-додатку або мобільного додатку. Клієнтська частина дозволяє користувачам створювати облікові записи, шукати товари чи послуги, розміщувати замовлення, відстежувати статус операцій та взаємодіяти зі смарт-контрактами на блокчейні

2) серверна частина: Серверна частина маркетплейса відповідає за обробку запитів користувачів та взаємодію з блокчейном Solana. Вона забезпечує збереження даних про товари, користувачів та замовлення, а також взаємодію зі смарт-контрактами. Серверна частина забезпечує валідацію та обробку транзакцій, керування безпекою та захистом даних, а також інтеграцію з різними сторонніми службами

3) блокчейн Solana: Magec Eden Solana використовує блокчейн Solana як основу для забезпечення безпеки та прозорості операцій. Solana - це високошвидкісна платформа блокчейн, яка використовує новаторську консенсус-модель Proof of History (PoH) та інші технології для досягнення високо транзакційної пропускної здатності та масштабованості. Блокчейн Solana забезпечує швидке підтвердження транзакцій і високу продуктивність, що дозволяє Magec Eden Solana обробляти великий обсяг операцій і забезпечувати ефективну роботу маркетплейсу

4) смарт-контракти: У рамках архітектури Magec Eden Solana використовуються смарт-контракти, які дозволяють автоматизувати та узгоджувати умови та дії на маркетплейсі. Смарт-контракти є програмними

кодами, що зберігаються на блокчейні і автоматично виконуються при заданих умовах. Вони контролюють процеси покупки, продажу, розподілу коштів, а також надають додаткові функціональні можливості, які сприяють безпекі та довірі всередині маркетплейсу.

Як було зазначено вище, сервер Magic Eden надсилає відповідь у форматі JSON, який складається з JsonNode. Спробуємо розглянути вигляд відповді на прикладі наступного запита:

```
GET https://api-
mainnet.magiceden.dev/v2/wallets/3PhZvHdJPiGhrLfjrciJiDJQSwhG2y5jk
a14KFQarh7b/activities?limit=2
Content-Type: application/json
```

Рис. 2.1 – Приклад запиту на орімання активностей гаманця до сервера Magic Eden.

Цей веб-запит є типом GET і виконується до адреси «https://api-mainnet.magiceden.dev/v2/wallets/3PhZvHdJPiGhrLfjrciJiDJQSwhG2y5jka14KFQarh7b/activities«. Запит включає параметр «limit=2», що вказує на обмеження кількості отриманих активностей до 2.Також вказаний заголовок «Content-Type: application/json», що вказує на те, що дані, отримані у відповіді, будуть в форматі JSON.

Отримуючи цей запит, сервер буде виконувати дії, пов'язані з отриманням активностей для гаманця з вказаним ідентифікатором «3PhZvHdJPiGhrLfjrciJiDJQSwhG2y5jka14KFQarh7b». Обмеження «limit=2» вказує, що сервер має повернути лише дві активності.

Відповідь на цей запит буде містити дані про активності, пов'язані з вказаним гаманцем, та їх відповідні атрибути, такі як дата, тип операції, вартість тощо. Формат відповіді буде відповідати вказаному у заголовку Content-Type, тобто він буде у форматі JSON.

Слід зазначити, що у контексті веб-розробки і взаємодії з сервером, існує кілька типів запитів, які можуть бути використані [17]:

1) GET: Запит GET використовується для отримання даних з сервера. Клієнтська сторона ініціює запит до сервера, передаючи параметри запиту у URL. Сервер обробляє запит і повертає відповідь, яка зазвичай містить запитані дані;

2) POST: Запит POST використовується для відправки даних на сервер для обробки. Цей тип запиту часто використовується при створенні нових записів або збереженні змін на сервері. Дані передаються у тілі запиту і оброблюються сервером;

3) PUT: Запит PUT використовується для оновлення існуючих даних на сервері. Клієнт передає нові дані на сервер, які потім замінюють відповідні записи на сервері;

4) DELETE: Запит DELETE використовується для видалення даних на сервері. Цей тип запиту надсилає серверу інструкцію видалити вказані дані.

Ці типи запитів (GET, POST, PUT, DELETE) є основою для взаємодії клієнтської та серверної частин веб-додатків. Вони дозволяють передавати та обробляти дані, забезпечуючи можливість читання, створення, оновлення та видалення ресурсів на сервері.

Відповіді наведено нижче:

```
[
  {
    "signature": "4M4KqAvE3SYjKAH29MuercVGDcfuXcRTS725wpYDo1TQt56oPnn7HrZkWqtfudRiPpd5jzv5gTKtSrK4r2BFfBye",
    "type": "buyNow",
    "source": "magiceden_v2",
    "tokenMint": "F7FBr8xaf7WDDUGhzKdUFRioWoDLGvBdhGX5utk1KiJd",
    "collection": "solcasino",
    "collectionSymbol": "solcasino",
    "slot": 198349046,
    "blockTime": 1686178963,
    "buyer": "Boztjqq31QNmkoq7nzk5ASH9kGY3reKmdUQcWCcWFc12",
    "buyerReferral": "",
    "seller": "3PhZvHdJPiGhrLfjrciJiDJQSwhG2y5jka14KFQarh7b",
    "sellerReferral": "",
    "price": 44.498
  },
  {
    "signature": "yAknA3kAujGzDoFaaupuonGAVphVRyTPz5pHrA3nH4JSQxDW37ifHPwkUKdcXRcqypSH7vsoW4arndehNP8khUg",
    "type": "list",
    "source": "magiceden_v2",
    "tokenMint": "F7FBr8xaf7WDDUGhzKdUFRioWoDLGvBdhGX5utk1KiJd",
    "collection": "solcasino",
    "collectionSymbol": "solcasino",
    "slot": 198348995,
    "blockTime": 1686178940,
    "buyer": null,
    "buyerReferral": "",
    "seller": "3PhZvHdJPiGhrLfjrciJiDJQSwhG2y5jka14KFQarh7b",
    "sellerReferral": "",
    "price": 44.498
  }
]
```


Рис. 2.2 – Приклад відповіді сервера Magic Eden на запит на оримання активностей гаманця.

Надана відповідь є масивом JSON об'єктів, який містить дві активності пов'язані з маркетплейсом Magec Eden Solana. Отже, кожен елемент масиву представляє окрему активність. Основні атрибути активностей включають:

1) «signature»: Унікальний підпис активності, який ідентифікує її у системі;

2) «type»: Тип активності, який може бути «buyNow» (покупка) або «list» (виставлення на продаж);

3) «source»: Джерело активності, в даному випадку «magiceden_v2» вказує на походження активності з маркетплейсу Magec Eden Solana;

4) «tokenMint»: Унікальний ідентифікатор токена, пов'язаного з активністю;

5) «collection»: Назва колекції, до якої належить токен;

6) «collectionSymbol»: Символ колекції, яка представлена токеном;

7) «slot»: Номер слоту, пов'язаного з активністю;

8) «blockTime»: Час блоку, у якому була зареєстрована активність;

9) «buyer»: Адреса гаманця покупця, якщо активність відноситься до покупки. Якщо поле має значення null, це означає, що активність є списком;

10) «seller»: Адреса гаманця продавця, пов'язаного з активністю.;

11) «sellerReferral» та «buyerReferral»: Посилання на реферала продавця і покупця (якщо такі є), які можуть бути використані для винагороди або реферальної програми;

12) «price»: Ціна, пов'язана з активністю.

Ця відповідь містить інформацію про дві активності: одна з типом «buyNow» і інша з типом «list». Кожна активність має свої унікальні властивості та значення, що характеризують їх специфічні дії на маркетплейсі Magec Eden Solana. З допомогою бібліотеки Jackson програмне забезпечення розпаршує отримані JSON ноди та саме з них бере найбільшу кількість необхідної нам інформації.

Інший формат представлення данних – це байтові масиви. У блокчейні, робота з даними зазвичай відбувається у формі байтових масивів. Блокчейн - це розподілена база даних, де інформація зберігається у вигляді блоків, а кожен блок містить певну кількість даних. Дані, які зберігаються у блоках, перетворюються на байтовий формат перед збереженням.

У багатьох блокчейн-платформах, таких як Bitcoin і Ethereum, дані представляються у вигляді байтових масивів, де кожен байт має своє значення і може кодувати різноманітну інформацію. Зазвичай, це означає, що дані, які включаються до блокчейну, мають бути перетворені у байтовий формат, перш ніж їх можна буде зберегти на ланцюжку блоків.

Байтові масиви даних у блокчейні можуть містити різні типи інформації, такі як транзакції, адреси гаманців, підписи, хеші, публічні ключі та інші дані, які є необхідними для функціонування блокчейн-платформи.

Для роботи з байтовими масивами даних у блокчейні, розробники можуть використовувати спеціальні бібліотеки або функції, які надаються блокчейн-платформою. Ці інструменти дозволяють кодувати дані у байтовий формат,

декодувати дані з байтів назад у зрозумілий формат та виконувати різні операції з цими даними, такі як перевірка підпису, обчислення хешів тощо.

Робота з байтовими масивами даних у блокчейні також включає в себе розуміння структури даних, яка використовується в конкретній блокчейн-платформі. Наприклад, у деяких блокчейн-платформах, таких як Ethereum, дані можуть бути організовані у вигляді «розумних контрактів», які включають функції та змінні. Ці розумні контракти можуть бути написані мовами програмування, такими як Solidity, та компілюватися до байтового коду, який зберігається на блокчейні.

Крім того, робота з байтовими масивами даних включає в себе розуміння протоколів комунікації з мережею блокчейн. Наприклад, для надсилання транзакцій на блокчейн необхідно скласти правильний байтовий формат транзакції, включаючи адреси гаманців, значення, дані та інші параметри. Цей байтовий формат потім підписується приватним ключем, щоб забезпечити автентифікацію та безпеку транзакції.

Також, при розробці додатків, які взаємодіють з блокчейн-платформою, можуть використовуватися байтові масиви даних для зберігання результатів операцій, отримання стану контрактів, взаємодії з розумними контрактами тощо. Розробники повинні бути знайомі з протоколами і структурами даних, які використовуються у конкретній блокчейн-платформі, щоб ефективно працювати з цими байтовими масивами даних.

У нашому випадку, байтові масиви отримуються у вигляді віповіді від сервера Magic Eden, який являє собою половину транзакії зібраної маркетеплейсом, яку нам, як користувачу, залишається лише підписати. Детальніше про це у наступному розділі – розробка та структура ПЗ.

2.3 Онсовні принципи моделювання системи

Основна ідея роботи ПЗ полгяаэ у наступному: бот автоматично виставляє ордера на купівлю NFT обраної коллекції, регулюючи ціну згідно з

коєфіцієнтом, який обирає сам користувач. Принцип виставленої ціни у офері розраховується за (2.1):

$$\text{offerPrice} = \text{collectionFloor} * \text{userCoefficient} \quad (2.1)$$

Слід зазначити, що оффер на покупку виставляється заголом на усю колекції, бідь-хто з інших користувачив маркетплейсу може відгукнутися на нього та моментально продати свій лот.

У випадку вдалої купівлі NFT, програмне забезпечення автоматично виставляэ об’яву по продажу купленого лота за ціною яка отримується в результаті роботи певного алгоритму, та регулює ціну лістінга доки не відбудиться продаж. Після чого, круг починається знов. Так як програмне забезпечення розроблялося з ціллю особистого використання, деталі реалізації цього алгоритму не буде розповсюджено.

У випадку, якщо NFT не було куплено ПЗ забезпечення автоматично регулює ціну виставленого ордера згідно з флором колекції, доки не відбудеться купівля.

Легко побачити що дане програмне забезпечення постійно змінює свій стан, тож для розробки даного програмного забезпечення був обраний один за патернів проектування [15]– patern state.

Паттерн State [16] дозволяє програмі змінювати свою поведінку в залежності від стану, без необхідності використання складних умовних конструкцій або великої кількості розгалужень. Він полегшує додавання нових станів та відповідних поведінок, оскільки кожен стан може бути реалізований окремим класом, і програма може просто переключатися між ними.

У випадку програмного забезпечення, яке автоматично купує та продає NFT, паттерн State може бути застосований шляхом визначення окремих класів для кожного стану, таких як «Очікування купівлі NFT» та «Очікування продажу купленого лоту». Кожен клас буде мати власні методи, які визначають його поведінку у відповідних станах.

Наприклад, у класі «Очікування купівлі NFT» можуть бути методи для виставлення ордерів на купівлю NFT залежно від введеного користувачем коефіцієнта, регулювання ціни офери та слідкування за рухом цін на ринку. При успішній купівлі програма переключається до класу «Очікування продажу купленого лоту», де можуть бути визначені методи для виставлення оголошення про продаж купленого NFT та регулювання ціни лістингу.

Застосування паттерна State дозволяє програмі бути гнучкою і легко розширюватись. Якщо в майбутньому з'явиться необхідність додати нові стани або змінити поведінку вже існуючих станів, це можна зробити шляхом створення нових класів станів та зміни логіки відповідних методів.

Крім того, паттерн State допомагає уникнути великих умовних конструкцій і повторюваного коду. Замість цього, кожен клас стану відповідає за свою власну поведінку, і програма просто переключається між станами за необхідності.

У випадку розглянутого програмного забезпечення, паттерн State дозволяє розбити складну логіку автоматичного купівлі та продажу NFT на окремі стани. Кожен стан визначає свою власну стратегію купівлі та продажу, що спрощує розробку та розуміння програмного коду.

Наприклад, якщо у майбутньому з'явиться потреба додати новий стан, то це можна зробити шляхом створення нового класу стану з відповідними методами та логікою. Існуючі класи стану не потребуватимуть змін, і програма зможе без проблем працювати з новим станом.

Узагальнюючи, паттерн State дозволяє програмному забезпеченню змінювати свою поведінку в залежності від внутрішнього стану. Він полегшує додавання нових станів та розширення функціональності, спрощує розробку та підтримку коду, а також сприяє збереженню чистоти та читабельності програми. Застосування паттерна State у розробці програмного забезпечення для автоматичної торгівлі NFT дозволяє ефективно управляти різними станами та поведінкою програми у кожному з них.

Переваги використання шаблону State для реалізації поліморфної поведінки добре видно: ймовірність помилки менша, і додавати додаткові стани для додаткової поведінки дуже легко, що робить його більш стійким, легким у підтримці та гнучким. Крім того, шаблон State допомагає уникнути умовної логіки if-else або switch-case в даному сценарії. Приклад реалізації патерна state наведено у додатуку Г.

2.4 Моделювання та розробка

2.4.1 Загальна структура програмного забезпечення

У цьому розділі буде детальніше розібрано структуру ПЗ, яке побудоване на патерні State, з використанням таких станів [14]: Аналіз (Analyze), Подання пропозиції (Make Offer) та Обробка продажу (Process Sell).

Метою цього програмного забезпечення є автоматизація процесу купівлі та продажу NFT-активів на маркетплейсі Magic Eden. Залежно від поточного стану системи, вона буде виконувати різні дії для досягнення оптимальних результатів.

Стан “Analaize” буде відповідати за аналіз ринкових умов, колекцій NFT та збирати інформацію для подальшої роботи з купленими лотами. У цьому стані система буде проводити дослідження, аналізувати дані та обчислювати оптимальні параметри для покупки та продажу.

Стан «подання пропозиції» буде відповідати за автоматичне виставлення замовлень на купівлю обраних NFT-активів з урахуванням коефіцієнта ціни, встановленого користувачем. Тут система автоматично регулюватиме ціну пропозицій відповідно до заданих параметрів, з урахуванням мінімальної ціни колекції.

Стан “обробка продажу” буде відповідати за автоматичне виставлення активів на продаж та управління ціною лотів до їх продажу. У цьому стані система використовуватиме певний алгоритм для визначення оптимальної ціни

та автоматично регулюватиме ціну лотів до того моменту, коли відбудеться продаж.

У цьому розділі буде детальніше розглянуто кожен зі станів, а також покажемо, як вони взаємодіють між собою для досягнення бажаних результатів у процесі купівлі та продажу NFT-активів.

Основний принцип роботи ПЗ базується на зміні стану классу контекста [12], який створюється у класі Main, через який й виконується старт роботи програми. Ця програма є точкою входу (entry point) для виконання коду. Вона містить метод main, який викликається при запуску програми. Приклад коду:

```
package com.example.soltest;

import com.example.soltest.Model.SolanaBidder;
import lombok.SneakyThrows;


public class Main {
    @SneakyThrows
    public static void main(String[] args) {

        SolanaBidder solanaBidder = new
SolanaBidder("lily","FE8g2P8RF6eb2Kf8izp4UokgLc2Qb5YsNKp2aAatvE5
Q");

        while (true) {
            solanaBidder.changeState();
            Thread.sleep(3 * 60 * 1000);

        }
    }
}
```

У головному методі main створюється об'єкт SolanaBidder, який ініціалізується з двома аргументами: рядком «lily» і рядком «FE8g2P8RF6eb2Kf8izp4UokgLc2Qb5YsNKp2aAatvE5Q». Це, очевидно, передає дані для створення екземпляра класу SolanaBidder. Після цього виконується безкінечний цикл while (true). У кожній ітерації циклу викликається метод changeState об'єкта solanaBidder. При цьому, ймовірно, змінюється стан solanaBidder або виконується певна логіка, пов'язана з об'єктом solanaBidder.

Після виклику методу changeState програма викликає Thread.sleep(3 * 60 * 1000), що призупиняє виконання програми на 3 хвилини (3 * 60 * 1000 мілісекунд). Блок-схема роботи головного нескінченого циклу програми наведена у додатку Ґ.

Роздивимость детальніше клас контексту – solanaBidder. Клас SolanaBidder є модельним класом, який представляє біддера на платформі Solana. Він містить поля і методи, які використовуються для зміни стану біддера та виконання певних дій.

Основний принцип роботи класу SolanaBidder полягає у зміні стану об'єкта state залежно від його поточного значення. Клас має кілька можливих станів: Analyze, MakeOffer і ProcessSale. Зміна стану відбувається у методі changeState(), який перевіряє поточний стан та виконує відповідні дії.

Основні поля класу SolanaBidder включають:

1) state: представляє поточний стан об'єкта SolanaBidder;

2) collectionName: назва колекції, пов'язаної з біддером;

3) wallet: гаманець, пов'язаний з біддером;

4) floor: нижня межа ціни поточної колекції;

5) currentTokenMintAddress: адреса поточного токена;

6) currentData: об'єкт типу DataClass, який містить дані для аналізу;

7) publicKey: публічний ключ, пов'язаний з гаманцем .

У конструкторі класу SolanaBidder встановлюються значення полів collectionName і wallet, а також ініціалізується об'єкт Analyze, щоб отримати значення floor.

Метод changeState() виконує перевірку поточного стану і змінює його наступним чином:

Якщо поточний стан є екземпляром Analyze, виконується аналіз даних за допомогою методу process(), об'єкт currentData оновлюється результатами аналізу, створюється об'єкт MakeOffer, йому передаються відповідні дані і стан змінюється на MakeOffer.

Якщо поточний стан є екземпляром MakeOffer, стан змінюється на ProcessSale, а поточні дані обробляються методом process() для ProcessSale.

Якщо поточний стан є екземпляром ProcessSale, стан змінюється на Analyze, а поточні дані обробляються методом process() для Analyze.

Приватний метод processData() в класі SolanaBidder оновлює значення поля floor на основі поточних даних currentData.

Отже, клас SolanaBidder має наступний код:

```
package com.example.soltest.Model;
import lombok.Data;


@Data

public class SolanaBidder {

    private ProcessBotState state;

    private final String collectionName;
    private final String wallet;

    private Double floor;
    private String currentTokenMintAddress;
    private DataClass currentData = new DataClass(floor);
    private String publicKey =
"FE8g2P8RF6eb2Kf8izp4UokgLc2Qb5YsNKp2aAatvE5Q";




    public void changeState(){
        if (state instanceof Analyze){
            this.currentData = ((Analyze) state).process();
            processData();
            MakeOffer makeOffer = new MakeOffer(collectionName, floor,
publicKey);
            makeOffer.setDataClass(currentData);
            setState(makeOffer);
        } else if (state instanceof MakeOffer) {
            setState(new ProcessSale(publicKey, currentData));
            this.currentData = ((ProcessSale) state).process();
        } else if (state instanceof ProcessSale) {
```

```
            setState(new Analyze(collectionName, floor, currentData));
            this.currentData = ((Analyze) state).process();
        }
    }

    public SolanaBidder(String collectionName, String wallet){
        this.collectionName = collectionName;
        this.wallet = wallet;
        Analyze analyze = new Analyze(collectionName, 0.0, currentData);
        floor = analyze.getFloor();
    }

    private void processData(){
        this.floor = currentData.getFloor();
    }

}
```

2.4.2 Система комунікаціїї даних між класами

Клас DataClass є модельним класом, який представляє дані, що використовуються для передачі інформації між класами в контексті платформи Solana.

Клас DataClass слугує для зберігання і передачі різноманітних даних між різними класами в програмі, зокрема в класі SolanaBidder (який представлений у попередньому коді). Він містить поля для зберігання інформації про ціни, вузли даних, активні пропозиції, старі значення тощо.

Наявність цього класу дозволяє зручно передавати і обмінюватись даними між різними частинами програми. Крім того, він допомагає утримувати стан і зберігати необхідну інформацію, яка може змінюватись та використовуватись в різних частинах програми.

Класс DataClass має наступний вигляд:

```
package com.example.soltest.Model;

import com.fasterxml.jackson.databind.JsonNode;
import lombok.Data;
import lombok.NonNull;

import java.util.ArrayList;
```

```
import java.util.HashMap;
import java.util.List;
import java.util.Map;

@Data
public class DataClass {

    private String name;
    @NonNull
    private Double floor;

    private List<Double> prices;

    private  List<JsonNode> listingsNodes;

    public Map<String, String> activeOffers = new HashMap<>();

    private Double oldFloor;

    private Map<String, String> inventory = new HashMap<>();
    private Map<String, String> oldInventory = new HashMap<>();
    private List<String> listings = new ArrayList<>();

}
```

Перед розбором структури станів в класі SolanaBidder, важливо зазначити, що стани відіграють ключову роль [18] у визначенні основної бізнес-логіки програми. Кожен стан представляє конкретний етап або фазу, у якій може перебувати об'єкт SolanaBidder. Зміна стану дозволяє виконувати відповідні дії, використовувати певні алгоритми та переходити до наступних етапів.

У класі SolanaBidder зустрічаються три основних стани: Analyze, MakeOffer та ProcessSale. Кожен з цих станів відповідає певній функціональності та має свою власну бізнес-логіку. Кожен стан має свої власні методи та виконує певні дії, які відповідають бізнес-логіці програми. Це дозволяє SolanaBidder ефективно працювати з платформою Solana, виконуючи необхідні кроки та операції для біддінгу та обробки продажу. Першим з станів буде Analize.

2.4.3 Стан «Analyze»

Клас Analyze є реалізацією інтерфейсу ProcessBotState і відповідає за аналіз даних колекції в контексті біддера на платформі Solana.

Основні поля класу Analyze включають:

1) name: назва колекції, яка підлягає аналізу;

2) floor: нижня межа ціни для колекції;

3) dataClass: об'єкт типу DataClass, який містить дані, що зібрані під час аналізу;

4) prices: список цін, зібраних під час аналізу;

5) istingsNodes: список вузлів (JSON-об'єктів), що представляють списки елементів колекції.

Клас Analyze має методи, які виконують наступні дії:

1) checkFloor(): отримує нижню межу ціни (floor) для заданої колекції шляхом звернення до зовнішнього API і отримання даних про колекцію. Метод рекурсивно перевіряє статус відповіді, повторюючи виклик до API, якщо відповідь не успішна;

2) checkFloor(String collectionName): аналогічний метод checkFloor(), але приймає назву колекції як параметр і повертає значення floor для цієї колекції;

3) collectStatsBuyNow(Integer offSet): збирає статистику цін купівлі на основі здійснених транзакцій у вказаній колекції. Метод отримує список цін із купівельних транзакцій, перебираючи JSON-об'єкти з відповіді API. Також виконується рекурсивний виклик до API для отримання більшої кількості даних, якщо недостатньо даних для аналізу;

4) getFirstListings(Integer offset): отримує перелік перших елементів колекції на основі відповіді API;

5) httpResponseGetter(String apiUrl): це приватний метод, який виконує HTTP-запит GET до вказаного URL-адреси apiUrl і повертає відповідь у формі

об'єкту HttpResponse. Використовується для звернення до зовнішнього API та отримання даних;

6) jsonToJsonNode(HttpResponse httpResponse): це приватний метод, який отримує об'єкт HttpResponse, отримує з нього тіло відповіді у вигляді рядка, а потім розпаковує його в об'єкт JsonNode. Використовується для перетворення JSON-відповіді на об'єкт JsonNode для подальшого використання;

7) serializeDataToJson(): це приватний метод, який серіалізує об'єкт Analyze в JSON-рядок. Використовується для перетворення об'єкту в JSON-формат для збереження або передачі даних;

8) process(): це реалізація методу з інтерфейсу ProcessBotState.

Клас Analyze використовує бібліотеку Jackson для роботи з JSON та Apache HttpClient для виконання HTTP-запитів. Він забезпечує функціональність для отримання даних про колекцію, аналізу цін та списків елементів колекції на основі зовнішнього API.

Основна задача класу Analyze полягає у зборі та аналізі даних про колекцію на платформі Solana. Метод process() виконує кілька кроків для отримання необхідної інформації.

По-перше, виконується метод checkFloor(), який отримує поточну найнижчу ціну (floor price) для колекції та зберігає її у змінній floor.

Далі, викликаються методи collectStatsBuyNow() та getFirstListings(), які збирають статистику цін на купівлю та отримують перші списки елементів колекції відповідно. Отримані дані про ціни та списки зберігаються у відповідних полях prices та listingsNodes об'єкта Analyze.

Нарешті, дані зберігаються в об'єкті DataClass, який повертається в результаті методу process(). Об'єкт DataClass містить інформацію про колекцію, найнижчу ціну, ціни на купівлю та списки елементів колекції, які можуть бути використані для подальшої обробки або відображення.

2.4.4 Стан «MakeOffer»

Клас MakeOffer є моделлю і містить реалізацію інтерфейсу ProcessBotState. Він використовується для створення та редагування пропозицій на платформі Magic Eden.

Деякі основні поля класу включають:

1) collectionSymbol: символ колекції, для якої створюється або редагується пропозиція;

2) floor: нижня межа ціни пропозиції;

3) publicKey: публічний ключ користувача, який створює або редагує пропозицію;

4) dataClass: об'єкт класу DataClass, який містить інформацію про активні пропозиції та інші дані;

5) pool: адрес пула оферу через який відбувається зміна ціни ордера.

Клас також містить ряд методів, зокрема:

1) makeOrder(): метод для створення замовлення на платформі Magic Eden;

2) parseInventory(): метод для парсингу інвентаря;

3) getSign(): метод для отримання підпису;

4) editOffer(): метод для редагування пропозиції;

5) checkBalance(): метод для перевірки балансу користувача;

6) getStringSignData(): метод для отримання підпису у вигляді рядка;

7) getStringSignDataEdit(): метод для отримання редагованого підпису у вигляді рядка;

8) signOffer(): метод для підпису пропозиції;

9) signEdit(): метод для підпису редагованої пропозиції.

Метод process() перевіряє, чи існує активна пропозиція для заданого символу колекції. Якщо немає активної пропозиції і нижня межа ціни (floor) менша за баланс користувача, викликається метод signOffer() для створення пропозиції. Якщо нижня межа ціни була змінена (dataClass.getOldFloor() != floor), викликається метод signEdit() для редагування пропозиції. На кінці

методу dataClass.getOldFloor() оновлюється зі значенням floor, і повертається об'єкт dataClass.

Цей клас використовує Selenium та WebDriver для взаємодії з веб-сторінками. Він використовує браузер Chrome (потрібно вказати шлях до драйвера chromedriver у системній властивості webdriver.chrome.driver).

У методі getSign() відбувається виконання JavaScript-коду за допомогою JavascriptExecutor, щоб отримати підпис з платформи Magic Eden. Після виконання коду, веб-драйвер закривається.

Методи getStringSignData() та getStringSignDataEdit() використовують отриманий підпис для отримання даних та оновлення значення поля pool.
Методи signOffer() та signEdit() використовують Solana RPC-клієнта для створення та відправки транзакцій з підписом для створення або редагування пропозицій. В цілому, клас MakeOffer представляє модель, яка взаємодіє з платформою Magic Eden, створюючи та редагуючи пропозиції на основі заданих даних.

2.4.5 Стан « **ProcessSale** »

Клас ProcessSale представляє модель обробки продажу і має ряд методів для виконання різних операцій пов'язаних з продажами на платформі Magiceden.io. Давайте розберемо його на основі методу process() і опишемо основні кроки, які він виконує.

Метод process() є реалізацією інтерфейсу ProcessBotState і повертає об'єкт класу DataClass. Цей метод викликається для обробки продажу та оновлення даних. Спочатку метод порівнює розмір старого і нового інвентаря (dataClass.getOldInventory().size() != dataClass.getInventory().size()). Якщо розмір змінився, виконується метод showSells().

Метод showSells() перевіряє, які елементи з інвентаря були продані, порівнюючи старий і новий інвентарі (dataClass.getOldInventory() і dataClass.getInventory()). Для кожного проданого елемента, він викликає метод

getAccountActivities() для отримання інформації про купівлю та обчислює прибуток. Після цього метод викликає listAndEdit().

Метод listAndEdit() перевіряє, чи всі елементи з інвентаря були виставлені на продаж на Magiceden.io. Якщо якісь елементи не були виставлені, вони перераховуються в missingElements, а потім викликається метод listing() для кожного елемента.

Далі, метод перевіряє, чи ціни на всі виставлені елементи відрізняються від рекомендованої ціни, викликаючи метод recomendedPrice(). Якщо ціна не відповідає рекомендованій, викликається метод editListing() для редагування ціни.

В кінці методу старий інвентар змінюється на новий (dataClass.setOldInventory(dataClass.getInventory())) і повертається об'єкт dataClass.

На основі методу process() можна сказати, що клас ProcessSale служить для відстеження продажу предметів, визначення прибутку, розміщення нових предметів на продаж та зміни цін існуючих предметів.

Отже ПЗ коду включає наступні аспекти:

1) чітка структура: ПЗ код має визначену структуру, включаючи розділення на класи та методи. Це сприяє організації та покращенню читабельності коду;

2) функціональність: Код включає методи, які виконують конкретні завдання, такі як пошук ціни покупки, отримання кількості токенів та отримання активностей облікового запису. Це дозволяє зручно та ефективно управляти процесом продажу;

3) гнучкість: Код може бути доповнений додатковими методами або алгоритмами для покращення функціональності. Наприклад, метод findBuyPrice може бути розширений для врахування додаткових факторів, що впливають на ціну;

4) використання даних: Код може використовувати різні дані, такі як ціни, кількість токенів та активності облікового запису, для виконання своїх

завдань. Це дозволяє програмі взаємодіяти зі зовнішніми джерелами даних та забезпечує гнучкість у роботі з різними типами інформації.

Висновок

Отже, у нашому дослідженні було розроблено програмне забезпечення з метою впровадження функціоналу, що пов'язаний з управлінням процесом торгівлі на ринку NFT, на платформі Magic Eden. Результати показують, що розроблена програма є ефективним інструментом для автоматизації торгової діяльності, які будуть проаналізовані у наступному розділі.

В ході аналізу і розробки програмного забезпечення було звернуто особливу увагу на структуру програми. Було використано чітке розділення на класи та методи, що сприяє кращій організації коду та полегшує його розуміння та розширення у майбутньому.

Крім того, програма володіє широким функціоналом. Різні методи були розроблені для виконання різноманітних завдань, таких як пошук ціни покупки, отримання кількості токенів та активностей облікового запису. Це дозволяє забезпечити гнучкість та зручність управління процесом торгівлі.

Особлива увага була приділена використанню даних. Програма взаємодіє з різними джерелами даних, такими як ціни, кількість токенів та активності облікового запису. Це дозволяє підтримувати актуальність та точність інформації, що використовується в процесі продажу.

Тож, розроблене програмне забезпечення є важливим кроком у поліпшенні процесу продажу товарів. Воно має чітку структуру, широкий функціонал та використовує різноманітні дані для ефективного управління.

РОЗДІЛ 3
ЕКСПЕРЕМЕНТАЛЬНО-АНАЛІТИЧНИЙ

Розділ, присвячений аналізу роботи програмного забезпечення та отриманих статистичних даних, є важливою частиною нашої роботи. В цьому розділі буде розглянуто результати, отримані від програмного забезпечення, що використовується для фінансового аналізу. Буде розглянуто форму виводу даних та проведемо аналіз отриманих статистичних даних з метою отримання цінних уявлень про фінансовий стан та прибутковість.

3.1 Аналіз формату виводу даних

У зв'язку з динамічністю ринку, про ще буде йтися у наступному підрозділі, для проведення аналізу даних було використано інше аналогічне програмне забезпечення, розроблене консультантом Ментомором, який надавав мені поради з питань розробки. Його програмне забезпечення було написане на мові програмування Python, тоді як дана розробка базувалась на мові програмування Java.

Варто відмітити, що програмне забезпечення, розроблене на мові програмування Python, для роботи з блокчейном - Solana ру, є більш розвиненим. Python має широкий вибір бібліотек та інструментів для обробки та аналізу даних, що дозволяє здійснювати більш зручний аналіз.

Використання аналогічного програмного забезпечення, розробленого на Python, дозволяє отримати достовірні результати аналізу даних, хоча розробка самої програми не була завершена у запланований термін. Це дозволяє зберегти продуктивність та точність аналізу шляхом використання готових рішень, розроблених експертом у відповідній області.

Незважаючи на різні мови програмування, використання аналогічного програмного забезпечення забезпечує послідовність та порівнянність

результатів аналізу даних, що дозволяє отримати інформацію про прибутковість активів та тренди на основі доступних даних.

Програмне забезпечення надає нам звітні дані за добу при у наступному форматі:

```
================================================================
                Report for day 02.27-02.28
Balance: 65.9
Balance in $: 1461.66
Profit: 0.48627666100000155
Profit in $: 10.785616340980035
                SOL price today: 22.18$
================================================================
```

Рис. 3.1 – Виведення статистичних данних отриманих під час роботи ПЗ у період з 02.27 по 02.28.

У даному форматі ПЗ отримує зведену інформацію за вказаний період, яка включає баланс рахунку, його еквівалент у доларах, прибуток та його еквівалент у доларах, а також ціну криптовалюти SOL на поточний день.

Аналізуючи формат виведення даних після продажу , який має такий вигляд:

```
Sold: oogy
                                    Purchase Price: 11.305150717
                                    Selling Price: 13.368
                                    Profit: 2.0628
                                    Profit in $: 43.93764000000001
                                    Time: 23.383333333333333
```

Рис. 3.2 – Вигляд виведення статистичних данних отриманих після продажу NFT.

Цей формат надає зручну структуру для представлення інформації про продаж. Кожен рядок містить конкретні дані, що дозволяє легко розрізняти різні аспекти операції. Наприклад, назва товару, закупівельна ціна, ціна

продажу, прибуток та прибуток у доларах відображені окремо. Час також вказується для відстеження, коли саме була здійснена операція.

Формат є зрозумілим та лаконічним, що сприяє легкому сприйняттю та аналізу даних. Він надає вичерпну інформацію про купівлю-продаж активу, його ціну та прибуток, що дозволяє зробити висновки щодо успішності операції. Такий формат виведення даних може бути корисним для моніторингу та аналізу фінансових операцій, оцінки результативності та виявлення трендів у прибутковості активів.

3.2 Аналіз отриманих результатів

У даному розділі буде здійснено аналіз фінансових результатів, зокрема стану балансу, прибутку та курсу криптовалюти Solana. Крім того, буде проведений аналіз денної статистики продажів та покупок. Ці дані допоможуть зрозуміти динаміку фінансових результатів та прийняти відповідні рішення для оптимізації діяльності. Далі будуть наведені конкретні висновки та аналіз кожного звіту за окремі дні, а також загальний аналіз за місяць.

Першим розглянемо внутриденну вибірку продажів NFT, у період з 23 по 24 лютого включно. Дані наведені у наступній таблиці:

Таблиця 3.1 – Внутриденна статистика продажів за період з 23 по 24 лютого 2023 р.

Number	Purch. price	Selling price	Profit	Profit in $	Time
1	3,137	3,331	0,193	4,459	23,91
2	3,196	3,299	0,102	2,355	81,7
3	6,553	6,749	0,195	4,491	39,66
4	3,836	3,928	0,091	2,106	55,633
5	4,534	4,699	0,164	3,789	212,66
6	4,53	4,651	0,121	2,799	26,81
7	4,654	4,69	0,036	0,838	153,4
8	3,263	3,251	-0,012	-0,277	254,083
9	4,54	4,728	0,188	4,342	3,233
10	6,122	6,249	0,126	2,916	55,183
11	3,176	3,44	0,263	6,053	92,716

Продовження табл. 3.1

12	10,821	11,089	0,267	6,152	1569,783
13	4,442	4,644	0,201	4,642	21,316
14	3,818	3,839	0,02	0,48	486,33
15	3,721	3,765	0,043	0,996	31,466
16	3,34	3,252	-0,088	-2,029	359,933
17	6,864	6,249	-0,615	-14,161	337,683
18	6,864	6,485	-0,37	-8,72	198,55
19	3,172	3,331	0,193	4,469	23,916
20	3,192	3,299	0,102	2,359	81,7
21	6,553	6,749	0,195	4,498	39,666
22	3,836	3,928	0,091	2,109	55,633
23	4,534	4,699	0,164	3,793	212,666
24	4,53	4,651	0,121	2,802	26,816
25	4,654	4,69	0,036	0,839	153,4
26	3,363	3,251	-0,012	-0,277	254,083
27	4,54	4,728	0,188	4,348	3,233
28	6,122	6,249	0,126	2,92	55,183
29	3,176	3,44	0,263	6,059	92,716
30	10,821	11,089	0,267	6,157	1569,783
31	4,442	4,644	0,201	4,648	21,316
32	3,818	3,839	0,02	0,481	486,33
33	3,721	3,765	0,0433	0,996	31,466
34	17,014	17,599	0,584	12,81	2,2
35	2,71	2,749	0,038	0,843	14,16
36	3,757	3,848	0,09	1,974	86,283
37	3,662	3,877	0,214	4,693	34,066
38	10,581	10,298	-0,282	-6,182	442,65
39	3,689	3,749	0,059	1,296	65,533
40	10,581	10,878	0,297	6,507	60,95
41	3,698	3,699	0,0005	0,0124	187,516
42	6,142	6,378	0,235	5,152	181,716
43	11,339	11,094	-0,245	-5,373	715,133
44	8,524	8,759	0,234	5,132	70,15

Для аналізу статистичних даних покупок протягом двох днів, а також для розрахунку відсотка збиткових угод та кореляції між часом продажу та профітом, використаємо надані дані.

За два дні проведено 44 угоди з криптовалютою. Середня ціна покупки становить 6.3344, а середня ціна продажу - 6.5119. Це вказує на невеликий

рівень виграшу від кожної угоди. Середній прибуток від кожної угоди становить 0.1775, або 4.0575 доларів. Це означає, що в середньому кожна угода приносить невеликий прибуток.

Рис. 3.3 – Динаміка прибутку за 2 дні.

За 2 дні проведено 8 збиткових угод, що становить 18.18% від загальної кількості угод. Це свідчить про наявність ризику в проведенні угод і може вказувати на потребу удосконалення стратегії торгівлі, аналізу ринку або ризикового управління.

Кореляція між часом продажу та профітом становить -0.0983. Це означає, що існує дуже слабка негативна залежність між цими двома змінними. У даному випадку, зменшення часу продажу не впливає суттєво на зміну прибутку.

З аналізу статистичних даних видно, що хоча середня ціна продажу та середній прибуток від кожної угоди є позитивними, є відносно невеликий ризик збиткових угод. Кореляція між часом продажу та профітом є дуже слабкою, що свідчить про те, що час продажу не має значущого впливу на дохідність угод.

Для проведення детального аналізу денної статистики за вказаний період часу, спочатку розглянемо основні показники, такі як баланс, прибуток та ціна SOL. Після цього будемо враховувати кореляцію між ціною SOL та прибутком, а також кореляцію між прибутком та балансом. Вибірка складається з даних

зубраних у період тестування ПЗ, тож стрімка зміна поточного балансу не є показником якості роботи загальної системи. Отримані дані у формі таблиці наведені нижче:

Таблиця 3.2 – Денні статистичні звіти з роботи ПЗ зібрані у період з 27 лютого по 19 березня 2023 р.

Number	Date	Balance	Profit	SOL price	Profit in $	Profit/Balance
1	02.27.2023	65,90	0,49	22,18	10,77948	0,00737481
2	02.28.2023	67,14	1,23	21,89	26,85903	0,018275246
3	03.01.2023	68,09	1,25	22,51	28,1375	0,018358056
4	03.02.2023	70,43	2,83	20,67	58,4961	0,040181741
5	03.03.2023	70,73	0,19	21,42	4,0698	0,002686272
6	03.04.2023	72,54	0,56	21,31	11,9336	0,007719879
7	03.05.2023	72,40	-0,62	20,89	-12,9518	-0,008563536
8	03.06.2023	73,01	0,45	20,76	9,342	0,006163539
9	03.07.2023	72,92	0,45	20,79	9,3555	0,006171146
10	03.08.2023	88,54	1,00	19,90	19,9	0,01129433
11	03.09.2023	83,07	6,37	18,63	118,6731	0,076682316
12	03.10.2023	143,12	3,69	17,21	63,5049	0,02578256
13	03.11.2023	145,45	1,23	18,58	22,8534	0,008456514
14	03.12.2023	154,59	1,01	17,97	18,1497	0,006533411
15	03.13.2023	145,90	0,74	20,28	15,0072	0,005071967
16	03.14.2023	144,66	0,00	21,63	0	0
17	03.15.2023	146,06	0,90	21,13	19,017	0,006161851
18	03.16.2023	160,48	0,57	22,49	12,8193	0,003551844
19	03.17.2023	155,83	0,00	21,70	0	0
20	03.18.2023	157,42	1,07	22,35	23,9145	0,006797103
21	03.19.2023	227,27	0,22	21,47	4,7234	0,000968012

Прибуток: Загальний прибуток складає 23,62.

1) максимальне значення прибутку: 6,37 (03.09.2023);

2) мінімальне значення прибутку: -0,62 (03.05.2023).

Ціна SOL: Варіація ціни SOL протягом періоду відбувається від 17,21 до 22,51.

3) максимальна ціна SOL: 22,51 (03.01.2023);

4) мінімальна ціна SOL: 17,21 (03.10.2023).

Кореляція між ціною SOL та прибутком: Коефіцієнт кореляції становить -0,55637091, що вказує на пряму зворотну залежність між ціною SOL та прибутком. Це означає, що коли ціна стрімко SOL зростає, прибуток зменшується, і навпаки.

Кореляція між прибутком та балансом: Коефіцієнт кореляції становить -0,085693163, що вказує на слабу зворотну залежність між прибутком та балансом. Це означає, що коли прибуток зростає, баланс зазвичай зменшується, і навпаки.

Побудуємо графіки ціни SOL та профіту по днях за вказаний період часу:

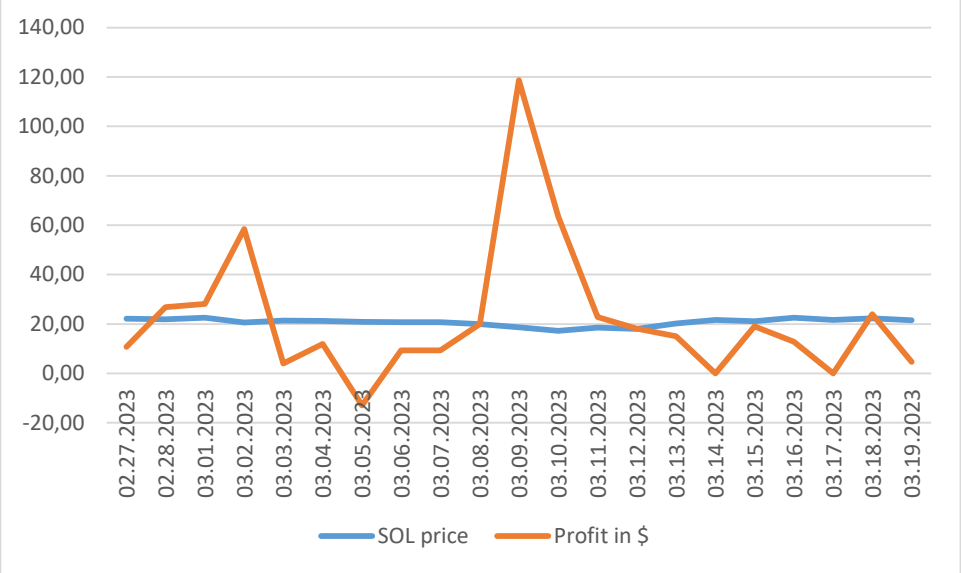

Рис. 3.4 – Динаміка зміна ціни $SOL та прибутку у період тестування.

Враховуючи ці висновки, рекомендується подальше моніторинг та аналіз ціни SOL, прибутку та балансу для прийняття виважених інвестувальних рішень. Слід звернути увагу на зміни в ціні SOL та їх можливий вплив на прибуток. Негативна кореляція між ціною SOL та прибутком вказує на необхідність уважного аналізу цінових тенденцій та можливих змін на ринку. Одночасно, слаба зворотна залежність між прибутком та балансом свідчить про

необхідність обережного планування і управління ризиками. При зростанні прибутку, можуть виникати ситуації, коли баланс зменшується. Тому важливо враховувати цей аспект при формуванні стратегії інвестування та визначенні меж ризику. Загалом, рекомендується проводити систематичний аналіз та моніторинг ринкових умов, особливо зміни ціни SOL, для прийняття обґрунтованих інвестиційних рішень.

Також слід звернути увагу на динаміку прибутку та його вплив на баланс, забезпечуючи належне управління ризиками. Ризик втрати коштів на курсі Solana (SOL) може бути великим через його волатильність і можливість швидких коливань ціни. Коли ви маєте відкриту позицію на SOL, зміни в ціні можуть впливати на вашу загальну вартість позиції. Якщо курс SOL знижується, це може призвести до збитків.

Хеджування шортовою позицією може бути одним з методів зменшення ризику втрати. При відкритті хеджувальної шортової позиції ви продаєте SOL на 20% від вашої загальної позиції за поточною ціною. Це дає вам можливість захистити себе від можливого зниження ціни SOL.

3.3 Робота програмного забезпечення у поточних реаліях ринку

Останні роки свідчать про швидкий розвиток ринку криптовалют та НФТ, а також про зміну тенденцій у цих напрямках. Однак, відомо, що жоден ринок не залишається незмінним і зазнає періодів зростання, стагнації або навіть спаду. У даному розділі буде розглянуто зміну тенденцій, що призвела до функціонування програмного забезпечення в цих сферах некоректно та з призупиненням доходів.

Ринок криптовалют відомий своєю динамічністю та нестабільністю. У перші роки свого існування, біткоїн і інші криптовалюти демонстрували вражаюче зростання цін, привертаючи увагу інвесторів та широкої громадськості. Однак, на протязі останнього часу ринок криптовалют

спостерігає значну зміну тенденцій, спричинену різними факторами, включаючи події «чорного лебедя».

Чорний лебідь – це термін, який використовується для позначення рідкісних подій, які мають непропорційно великий вплив та непередбачувані наслідки. На ринку криптовалют, події чорного лебедя можуть включати геополітичні кризи, натуральні катастрофи, фінансові кризи та інші події, які суттєво впливають на цінову динаміку та стабільність ринку.

Наприклад, криптовалютний ринок був під впливом пандемії COVID-19, яка призвела до глобальних економічних турбулентностей. Ця непередбачувана подія призвела до різкого падіння цін на більшість криптовалют, що викликало нервовість серед інвесторів та втрати капіталу. Подібні ситуації можуть також виникати через регулятивні зміни, які негативно впливають на легітимність та використання криптовалют.

Однією с таких подій став розвиток всесвітньої кризи, а саме ризик дефолту США та збільшення стелі государственного боргу. Учасники ринку почали обирату безпечнішу стратегію торгівлі, та частіше продавати свої активи виставляючи лот а не через ордер, спосіб через який ПЗ виконує покупку. Багато інвесторів почали массово продавати свої активи, що призвело до наступного:

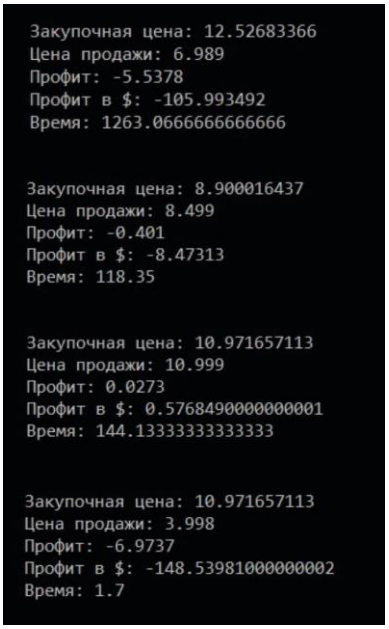

Рис. 3.5 – Збиткові угод у травні 2023 р.

Програмне забезпечення не було оптимізовано для корректного реагування на подібні різкі зміни на ринку, щи призвело до укладення збиткових та низькопрофітних угод. Для подальшої роботи і поліпшення програмного забезпечення необхідно провести його доопрацювання та адаптацію під нинішні реалії ринку. Враховуючи виявлені проблеми та недоліки, слід здійснити повторний аналіз ринку та цінових тенденцій, а також розробити та впровадити додаткові алгоритми, які дозволять програмному забезпеченню ефективніше реагувати на швидкі зміни та непередбачувані події на ринку криптовалют. Це може включати удосконалення стратегій хеджування, аналіз факторів, що впливають на курси криптовалют, та розробку імунітету до чорних лебедів.

Висновок

Отже, у цьому розділі було розглянуто динаміку цінової зміни та прибутку на ринку криптовалют, зокрема на прикладі SOL. Виявлено слабу зворотну залежність між прибутком та балансом, що вказує на необхідність уважного аналізу цінових тенденцій та управління ризиками. Звернуто увагу на волатильність ринку та можливість збитків через зміни ціни криптовалют, а також на вплив чорних лебедів, геополітичних криз, фінансових криз та інших подій на ринок. Рекомендується проводити систематичний аналіз та моніторинг ринкових умов для прийняття обґрунтованих інвестиційних рішень і застосування стратегій хеджування для зменшення ризиків збитків.

ВИСНОВКИ

Основна мета кваліфікаційної роботи була досягнута – розробка програмного зебеспечення для автоматизації торгової діяльності на ринку діцентралізованих невзаємозамінних активів. Проте, на поточний момент програма не функціонує належним чином і потребує подальшої доопрацювання. Використання структури проекту, заснованої на патерні State, може вирішити цю проблему швидко та ефективно.

Основною перевагою використання патерну State в структурі проекту є можливість визначення різних станів програми і пов'язаних з ними дій. Це дозволяє забезпечити гнучкість та контроль над поведінкою програми, а також швидко реагувати на змінні умови та події. Крім того, додавання нових станів та розширення функціональності стають простими завдяки структурі проекту на основі патерну State.

Отриманий результат також підтверджує важливість аналізу та оцінки динаміки ринку криптовалют та NFT для розуміння прибутковості та ризиків інвестицій. За результатами аналізу було виявлено, що ринок криптовалют та NFT є динамічним і зазнає значних змін. Цінова динаміка активів, зокрема ціна SOL, може значно впливати на прибуток та ризики інвесторів. Кореляційний аналіз показав, що існує зв'язок між ціноутворенням активів та прибутком.

Враховуючи головну мету програмного забезпечення - автоматизацію торгової діяльності, доопрацювання програми є необхідним кроком для досягнення успішного результату. Програмне забезпечення вимагає написання додаткових алгоритмів реагування на різкі стрибки ціни, додавання режиму очікування та більш точної системи ціноутворення, яка б дала можливість аналізувати доцільність продажу активу та фіксації збитків у поточний момент часу. Додавання цих функцій допоможе позитивно вплинути на роботу програмного забезпечення і знову вивести його роботу в прибуток.

Необхідно враховувати, що на ринку криптовалют існує значний вплив людського фактору та подій «чорного лебедя». Навіть найбільш ретельно

розроблені програмні рішення можуть бути піддані непередбачуваним впливам, що виникають внаслідок геополітичних криз, натуральних катастроф, фінансових криз та інших подій. Такі події можуть спричиняти різкі коливання цін криптовалют та створювати турбулентність на ринку. Програмне забезпечення повинне бути гнучким і адаптивним, здатним реагувати на такі непередбачувані події та надавати користувачам актуальну інформацію для прийняття розумних рішень. Додаткові алгоритми та аналітичні інструменти повинні бути включені для аналізу та оцінки ризиків, пов'язаних з подіями «чорного лебедя». Людський фактор, включаючи емоції, психологічні реакції та стратегії інвесторів, також має великий вплив на реалії ринку. Тому необхідно постійно вдосконалювати програмне забезпечення, збільшувати його адаптивність та забезпечувати надійну аналітику, щоб ефективно працювати в умовах непередбачуваності та забезпечувати успішне управління ризиками.

На даному етапі ми маємо готове програмне забезпечення, яке є зручним для розширення та модернізації. Протягом певного періоду воно відображало прибуткову статистику, але через непередбачувані реалії ринку, його прибутковість знизилася. Незважаючи на це, ми розуміємо, що вдосконалення програмного забезпечення є необхідним для подальшої роботи. В даний час проводиться ретельний аналіз та пошук основних причин, які призвели до некоректної роботи програми. Плануються вдосконалення алгоритму та адаптування програмного забезпечення під нинішні реалії ринку, забезпечуючи більш точний аналіз, керування ризиками та ефективність в умовах непередбачуваності.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Кравченко П. Блокчейн і децентралізовані системи / П. Кравченко. – Харків: DistributedLab, 2019. – 452 с.

2. Шлемин А. NFT. Технология, технологія яка змінить світ / А. Шлемин. – Київ, 2021. – 186 с.

3. Сушков В. А. Метавсесвіти, DeFi та NFT. Путівник у світі інновацій / Володимир Александрович Сушков. – Харків. – 156 с.

4. Non-fungible token [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Non-fungible_token.

5. Solana API Overview [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: https://docs.magiceden.io/reference/solana-overview.

6. JSON RPC API [Електронний ресурс] // Solana. – 2023. – Режим доступу до ресурсу: https://docs.solana.com/ru/api.

7. Morrell M. SolanaJ [Електронний ресурс] / Michael Morrell. – 2022. – Режим доступу до ресурсу: https://github.com/skynetcap/solanaj.

8. What Is Magic Eden? — and Why It’s the Top Solana NFT Marketplace [Електронний ресурс] // ByBit. – 2022. – Режим доступу до ресурсу: https://learn.bybit.com/nft/what-is-magic-eden/.

9. Solana | Web3 Infrastructure for Everyone [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://solana.com/ru.

10. Хорстманн К., Корнелл Р. Java 2. Бібліотека професіонала. Т. 1, 2. - М.: Віль-ямс, 2010. - 816 с., 992 с.

11. Еккель Брюс. Філософія Java. Бібліотека програміста. - 4-те вид. - СПб.: Пітер, 2009. - 640 с.

12. Бадд Т. Об'єктно-орієнтоване програмування у дії: Пер. з англ. - СПб.: Пітер, 1997. - 464 с. Буч Г. Об'єктно-орієнтований аналіз та проектування з прикладами додатків: Пер. з англ. - 3-тє вид. - М. Вільямс, 2010. - 720 с.

13. Коуд П., Норт Д., Мейфілд М. Об'єктні моделі. Стратегії, шаблони та програми: Пер. з англ. - М.: Лорі, 1999. - 446 с.

14. Гамма Е., Хелм Р., Джонсон Р., Вліссідес Дж. Прийоми об'єктноорієнтованого проектування. Патерни проектування. - СПб.: Пітер, 2005. - 368 с.

15. Стелтінг С., Маассен О. Застосування шаблонів Java. Бібліотека професіонала:Пер. з англ. - М.: Вільямс, 2002. - 576 с.

16. Bruce Eckel. Помітні в Patterns with Java. - 2004. [Електронний ресурс]. /– Режим доступу до ресурсу: http://mindview.net/Books/TIPatterns/

17. Ларман К. Застосування UML 2.0 та шаблонів проектування: Пер. з англ. - М.: Вільямс, 2009. - 736 с.

18. Блох Дж. Java. Ефективне програмування: Пров. з англ. - М.: Лорі, 2008. -223 с.

19. Хабібуллін І. Ш. Створення розподілених додатків на Java 2. - СПб.: БХВ-Петербург, 2002. - 704 с.

20. Java Documentation [Електронний ресурс] // Oracle. – 2023. – Режим доступу до ресурсу: https://docs.oracle.com/en/java/.

ДОДАТОК А. Відомість матеріалів кваліфікаційної роботи

№ з/п	Позначення	Назва	Кількість	Примітки
1				
2		Документація		
3				
4	САіУ.РД.23.07.ПЗ	Пояснювальна записка	54	Формат А4
5				
6	САіУ.РД.23.07.ДМ	Демонстарційні матеріали	15	Презентація на CD-R
7				
8	САіУ.РД.23.07.КР	Копія роботи	1	Диск CD-R
9				
10				
11				
12				
13				

					САіУ.РД.23.07.ДА.ПЗ
Змін.	Аркуш	№ докум.	Підпис	Дата	

Розроб.				**Матеріали кваліфікаційної роботи**	Літ.			Аркуш	Аркушів
К. розд.									
Керівн.					НТУ «ДП»; 124-19-1				
Н.контр.									
Зав.каф.									

ДОДАТОК Б. Відгук на кваліфікаційну роботу ступеню бакалавра

Відгук

на кваліфікаційну роботу ступеню *бакалавра*

студентки *Іванчика Д.* академічної групи *124-19-1*

спеціальності: *124 Системний аналіз*

Метою роботи є розробка автоматизованого програмного забезпечення здатного вести торгову діяльність на ринку децентралізованих невзаємозамінних активів та отримувати прибуток. Обрана тема є актуальною, так як цей ринок є швидкозростаючим, що забезпечує широкий спектр можливостей для заробітку.

Тема кваліфікаційної роботи безпосередньо пов'язана з об'єктом діяльності спеціаліста фаху «Системний аналіз».

Вміння бачити можливості для автоматизації з метою отримання прибутку, а також здатність працювати з науковою літературою разом з здібністю писати системи які обробляють великі обсяги даних дало Іванчику Д. змогу виконати завдання: розробити програмне забезпечення яке автоматизує торгівельну діяльність на ринку децентралізованих невзаємозамінних активів. Але, через динамічність даного ринку та наявність великого впливу людського фактора на нього, програмне забезпечення у даний момент не приносить дохід, втім значний період часу воно показувало позитивну динаміку та приносило прибуток, що продемонстровано у кваліфікаційній роботі.

Вважаю, що зацікавленість автоматизацією процесів та методами розробки програмного забезпечення дозволять Іванчику Д. і надалі отримувати вагомі результати.

Дипломна робота Іванчика Д. може бути допущена до захисту, а її автор заслуговує оцінки «відмінно» та присвоєння кваліфікації «бакалавра з системного аналізу».

Доцент кафедри системного аналізу
і управління, , к. ф.-м. н. Л.С. Коряшкіна

ДОДАТОК В. Приклад реалізації патерна state

Якщо нам потрібно змінити поведінку об'єкта залежно від його стану, ми можемо мати змінну стану в об'єкті і використовувати умовний блок if-else для виконання різних дій залежно від стану. Шаблон стану використовується для забезпечення систематичного та структурованого підходу до досягнення цього шляхом реалізації контексту та стану.

Контекст - це клас, який має посилання на стан для однієї з конкретних реалізацій стану і перенаправляє запит до об'єкта стану для обробки. Давайте розберемося з цим на простому прикладі.

Припустимо, що ми хочемо реалізувати пульт дистанційного керування телевізором за допомогою простої кнопки для виконання дій. Якщо стан ввімкнено, телевізор увімкнеться, а якщо стан вимкнено, телевізор вимкнеться. Ми можемо реалізувати це, використовуючи умову if-else, як показано нижче:

```
package com.journaldev.design.state;

public class TVRemoteBasic {

   private String state="";

   public void setState(String state){
      this.state=state;
   }

   public void doAction(){
      if(state.equalsIgnoreCase("ON")){
         System.out.println("TV is turned ON");
      }else if(state.equalsIgnoreCase("OFF")){
         System.out.println("TV is turned OFF");
      }
   }

   public static void main(String args[]){
      TVRemoteBasic remote = new TVRemoteBasic();

      remote.setState("ON");
      remote.doAction();

      remote.setState("OFF");
      remote.doAction();
   }

}
```

Зверніть увагу, що клієнтський код повинен знати конкретні значення, які слід використовувати для встановлення стану віддаленого керування. Крім

того, якщо кількість станів збільшиться, тісний зв'язок між реалізацією та клієнтським кодом буде дуже важко підтримувати та розширювати.

Тепер використаємо шаблон стану для реалізації вищезазначеного прикладу з пультом телевізора.

Спочатку було створено інтерфейс State, який визначить метод, який повинен бути реалізований різними конкретними станами і класом контексту.

```
package com.journaldev.design.state;

public interface State {

    public void doAction();
}
```

У нашому прикладі можуть бути два стани - один для увімкнення телевізора і інший для його вимкнення. Таким чином, було створено дві конкретні реалізації станів для цієї поведінки.

```
package com.journaldev.design.state;

public class TVStartState implements State {

    @Override
    public void doAction() {
        System.out.println("TV is turned ON");
    }

}
package com.journaldev.design.state;

public class TVStopState implements State {

    @Override
    public void doAction() {
        System.out.println("TV is turned OFF");
    }


}
```

Тепер реалізуємо Контекст, який змінить свою поведінку залежно від його внутрішнього стану.

```
package com.journaldev.design.state;

public class TVContext implements State {

    private State tvState;

    public void setState(State state) {
        this.tvState=state;
    }

    public State getState() {
        return this.tvState;
    }

    @Override
    public void doAction() {
```

```
        this.tvState.doAction();
    }

}
```

Зверніть увагу, що Контекст також реалізує інтерфейс State, зберігає посилання на поточний стан і перенаправляє запит до реалізації стану.

Теперь напишем простую программу для тестирования нашей реализации TV Remote с использованием шаблона State.

```
package com.journaldev.design.state;

public class TVRemote {

    public static void main(String[] args) {
        TVContext context = new TVContext();
        State tvStartState = new TVStartState();
        State tvStopState = new TVStopState();

        context.setState(tvStartState);
        context.doAction();

        context.setState(tvStopState);
        context.doAction();

    }

}
```

Вивід цієї програми буде таким самим, як і в базовій реалізації пульта телевізора без використання будь-якого шаблону.

ДОДАТОК Г. Блок-схема роботи класа Main