НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Навчально-науковий інститут атомної та теплової енергетики

Кафедра цифрових технологій в енергетиці

«До захисту допущено»
Завідувач кафедри ЦТЕ
_____ Наталія АУШЕВА
«___» _____ 2025р.

Дипломна робота
на здобуття ступеня бакалавр

За освітньою програмою «Цифрові технології в енергетиці»
Спеціальність 122 «Комп'ютерні науки»
на тему: «Торговий бот для криптовалютних бірж»

Виконав: студент 4 курсу, групи ТР-11
Білий Данило Юрійович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник: професор д.т.н., доц. Олексій ШУШУРА _____
(посада, науковий ступінь, вчене звання, ім'я, ПРІЗВИЩЕ) (підпис)

Рецензент: професор каф. ІПЗЕ, д.т.н., проф. Євген ГАВРИЛКО _____
(посада, науковий ступінь, вчене звання, ім'я, ПРІЗВИЩЕ) (підпис)

Н.контроль: ст. викладач Ольга БЕСПАЛА _____
(посада, ім'я, ПРІЗВИЩЕ) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2025

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ АТОМНОЇ ТА ТЕПЛОВОЇ ЕНЕРГЕТИКИ

Кафедра ЦИФРОВИХ ТЕХНОЛОГІЙ В ЕНЕРГЕТИЦІ

Рівень вищої освіти – перший (бакалаврський)

спеціальність 122 «Комп'ютерні науки»

Освітньо-професійна програма «Цифрові технології в енергетиці»

ЗАТВЕРДЖУЮ
Завідувач кафедри ЦТЕ
Наталія АУШЕВА
«___» _____ 2025р.

ЗАВДАННЯ
на дипломну роботу студенту

Білому Данилу Юрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Торговий бот для криптовалютних бірж

Науковий керівник Шушура Олексій Миколайович, д.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «02» червня 2025 року № 1875-с

2. Термін подання роботи студентом _____

3. Вихідні дані до роботи торгові стратегії; технічні індикатори; API криптовалютної біржі Binance (testnet); Офіційна бібліотека Binance Connector Java; Сервіс хмарного розгортання AWS

4. Зміст роботи аналіз проблем ручної торгівлі на ринку криптовалют; огляд методів автоматизації криптотрейдингу, існуючих платформ і бібліотек; проектування архітектури торгового бота; розробка модулів взаємодії з біржею, обробки даних, технічного аналізу, конфігурації стратегій; реалізація логіки прийняття рішень і торгівлі; тестування роботи системи у хмарному середовищі AWS.

5. Орієнтовний перелік графічного матеріалу технічні індикатори; архітектура системи; торговий цикл стратегії.

6. Дата видачі завдання 13.09.2024

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1.	Вибір теми роботи	15.09.24	Виконано
2.	Аналіз методів та засобів розв'язання задачі	17-20.04.25	Виконано
3.	Розробка архітектури та загальної структури системи	21-25.04.25	Виконано
4.	Розробка окремих підсистем	26.04-02.05.25	Виконано
5.	Програмна реалізація системи	03-07.05.25	Виконано
6.	Оформлення пояснювальної записки	08.-12.05.25	Виконано
7.	Захист програмного забезпечення	12-16.05.25	Виконано
8.	Передзахист	26-28.05.25	Виконано
9.	Захист	16-20.06.25	Виконано

Студент _____ Білий Д.Ю
(підпис) (прізвище та ініціали)

Керівник роботи _____ Шушура О.М.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Дипломна робота виконана на 54 сторінках, містить 5 ілюстрацій, 2 додатки, 19 джерел в переліку посилань.

Мета роботи: розробка програмної системи – торгового бота для автоматизації процесу торгівлі криптовалютними активами на біржах, що функціонує на основі комбінації технічних індикаторів з можливістю гнучкого конфігурування параметрів торгових стратегій.

Методи та засоби: технічний аналіз ринку криптовалют, використання комбінованої торгової стратегії на основі індикаторів MACD та RSI, модульне проєктування архітектури системи; мова програмування Java з фреймворком Spring Boot, офіційна бібліотека Binance Connector Java для взаємодії з API біржі Binance, конфігураційні JSON-файли для визначення торгових стратегій, хмарна платформа AWS для розгортання та тестування.

Результати: розроблено програмну систему «Торговий бот для криптовалютних бірж», що автоматизує торгові операції на біржі Binance. Система забезпечує аналіз ринку на основі індикаторів MACD та RSI, гнучке конфігурування торгових стратегій через зовнішні JSON-файли, автоматичне розміщення ринкових ордерів та управління ризиками за допомогою ордерів Stop-Loss та Take-Profit. Функціональність бота перевірена в тестовому середовищі Binance, а результати апробовані під час переддипломної практики. Розроблений бот є функціональним прототипом, рекомендованим для подальшого розвитку та адаптації під реальні торгові умови.

Ключові слова: ТОРГОВИЙ БОТ, КРИПТОВАЛЮТА, BINANCE, API, ТЕХНІЧНИЙ АНАЛІЗ, MACD, RSI, JAVA, SPRING BOOT, АВТОМАТИЗОВАНА ТОРГІВЛЯ

ANNOTATION

The thesis is 54 pages long, contains 5 figures, 2 appendices, and 19 references.

Objective: development of a software system – a trading bot for automating the process of trading cryptocurrency assets on exchanges, operating based on a combination of technical indicators with the possibility of flexible configuration of trading strategy parameters.

Methods and tools: technical analysis of the cryptocurrency market, use of a combined trading strategy based on MACD and RSI indicators, modular design of the system architecture; Java programming language with the Spring Boot framework, the official Binance Connector Java library for interacting with the Binance exchange API, JSON configuration files for defining trading strategies, AWS cloud platform for deployment and testing.

Results: a software system "Trading Bot for Cryptocurrency Exchanges" that automates trading operations on the Binance exchange has been developed. The system provides market analysis based on MACD and RSI indicators, flexible configuration of trading strategies via external JSON files, automatic placement of market orders, and risk management using Stop-Loss and Take-Profit orders. The bot's functionality was verified in the Binance test environment, and the results were approbated during the pre-diploma practice. The developed bot is a functional prototype recommended for further development and adaptation to real trading conditions.

Keywords: TRADING BOT, CRYPTOCURRENCY, BINANCE, API, TECHNICAL ANALYSIS, MACD, RSI, JAVA, SPRING BOOT, AUTOMATED TRADING.

ЗМІСТ

ВСТУП

Ринок криптовалют, що характеризується стрімким зростанням та залученням все більшої кількості учасників, від індивідуальних інвесторів до великих фінансових інституцій, представляє собою унікальне середовище для торгівлі. Водночас його ключові особливості – висока волатильність та цілодобова доступність – створюють як значні можливості для отримання прибутку, так і підвищені ризики. [1] У таких умовах традиційні підходи до ручної торгівлі стикаються з низкою суттєвих викликів. По-перше, надзвичайно висока динамічність ринку, де ціни на криптовалюти можуть змінюватися дуже швидко, ускладнює ручне відстеження всіх значущих рухів та прийняття своєчасних, обґрунтованих рішень. По-друге, цілодобовий режим роботи криптовалютних бірж робить практично неможливим постійний ручний контроль за ринковою ситуацією з боку окремого трейдера [1, 2]. По-третє, невід'ємною складовою ручної торгівлі є емоційний фактор: такі емоції, як страх втратити кошти або жадібність, часто підштовхують до прийняття необґрунтованих рішень, що може призвести до значних фінансових збитків. Крім того, реалізація деяких комплексних торгових стратегій, що вимагають одночасного аналізу багатьох ринкових параметрів та миттєвого виконання торгових операцій, стає надзвичайно складною, якщо не неможливою, при ручному управлінні. Нарешті, трейдери часто працюють з декількома криптовалютними біржами, кожна з яких має власний програмний інтерфейс (API) та специфічні особливості функціонування, що створює додаткові труднощі для інтеграції та уніфікованого управління [3].

Актуальність теми даної дипломної роботи зумовлена нагальною потребою у створенні та впровадженні ефективних інструментів для автоматизації торгових процесів на ринку криптовалют. Торгові боти, як спеціалізоване програмне забезпечення, покликані вирішити зазначені проблеми, мінімізуючи вплив людського фактору, забезпечуючи можливість цілодобового моніторингу ринкової кон'юнктури та дозволяючи реалізовувати складні торгові алгоритми, які важко або неможливо виконати вручну. Розробка гнучкого, надійного та ефективного

торгового бота, здатного інтегруватися з API провідних криптовалютних бірж, зокрема Binance, та виконувати торгові операції на основі чітко визначених та легко налаштовуваних користувачем стратегій, є важливим практичним завданням. Таке програмне забезпечення має на меті оптимізувати торговий процес, знизити ймовірність прийняття емоційно зумовлених рішень та, як наслідок, підвищити потенційну прибутковість операцій на висококонкурентному криптовалютному ринку [3]. Хоча на сучасному ринку існує низка комерційних платформ для автоматизованої торгівлі, вони часто характеризуються або обмеженою гнучкістю у налаштуванні унікальних, кастомізованих торгових стратегій, або вимагають значних регулярних фінансових витрат на підписку , що робить розробку власної, адаптованої під конкретні потреби програмної системи актуальним та економічно виправданим напрямком [1].

Мета роботи полягає у розробці програмної системи – торгового бота для автоматизації процесу торгівлі криптовалютними активами на біржах, що функціонує на основі комбінації технічних індикаторів з можливістю гнучкого конфігурування параметрів торгових стратегій.

Для досягнення поставленої мети в рамках дипломної роботи визначено наступні завдання:

- проаналізувати підходи, методи та алгоритми автоматизованої торгівлі;
- здійснити огляд існуючих платформ для автоматизації торгівлі, визначити вимоги до системи;
- здійснити огляд та вибір технічних індикаторів для застосування в стратегіях торгівлі;
- розробити архітектуру системи відповідно до визначених вимог та зробити вибір засобів розробки;
- розробити програмне забезпечення торгового бота, включаючи компоненти для конфігурації стратегій, взаємодії з API бірж, розрахунку технічних індикаторів та реалізації торгової логіки з управлінням ризиками;

– провести комплексне тестування розробленого бота, включаючи інтеграцію з тестовим середовищем, функціональну перевірку стратегій та стабільність роботи.

Дипломна робота складається зі вступу, п'яти основних розділів, що послідовно розкривають етапи дослідження та розробки, висновків, списку використаних джерел та додатків. Загальний обсяг роботи становить 54 сторінки машинописного тексту. Робота містить 5 рисунків, що ілюструють архітектуру системи, діаграми активності та результати роботи. Список використаних джерел налічує 19 найменувань. У додатках наведено результати апробації та фрагмент коду розрахунку технічних індикаторів.

1 ПОСТАНОВКА ЗАДАЧІ

Ринок криптовалют демонструє стрімке зростання та приваблює все більше учасників, від індивідуальних інвесторів до великих фінансових інституцій. Характерною особливістю цього ринку є його висока волатильність та цілодобова доступність, що створює як значні можливості для отримання прибутку, так і підвищені ризики. В таких умовах автоматизація торгових процесів за допомогою спеціалізованого програмного забезпечення – торгових ботів – стає не просто перевагою, а необхідністю для ефективного управління активами та реалізації торгових стратегій [3].

Щоб відповідати потребам сучасного ринку, такі торгові боти мають забезпечувати наступний функціонал:

- взаємодіяти з однією або декількома криптовалютними біржами через їх API;
- отримувати ринкові дані (ціни, обсяги, біржовий стакан тощо) в реальному часі;
- аналізувати отримані дані на основі заздалегідь визначених та налаштовуваних користувачем торгових стратегій;
- приймати рішення про купівлю/продаж криптовалютних активів;
- автоматично розміщувати та керувати торговими ордерами на біржі;
- вести логування своїх дій та результатів торгівлі;

Виходячи з вищезазначеного, основною задачею розробки є створення торгового бота для криптовалютних бірж, який реалізує перелічений функціонал. Система повинна бути спроектована з урахуванням модульності для забезпечення гнучкості, легкою масштабованістю для підтримки нових бірж та торгових стратегій, а також гарантувати надійне, безпечне та ефективне виконання торгових операцій в автоматичному режимі. Розроблюваний бот має на меті оптимізувати торговий процес, знизити вплив людського фактору та підвищити потенційну прибутковість операцій на криптовалютному ринку.

Для успішної реалізації поставленої задачі необхідно виконати низку взаємопов’язаних етапів розробки, кожен з яких охоплює ключові аспекти проєктування, програмування, налаштування та тестування системи.

1. Проектування архітектури програмної системи. Розробити детальну архітектуру торгового бота, визначивши ключові компоненти: модуль взаємодії з біржами; сервіси обробки ринкових даних та виконання торгових операцій; модуль прийняття рішень; модуль технічного аналізу та управління стратегіями. Спираючись на спроектовану систему обрати відповідний технологічний стек для реалізації.

2. Розробка сервісів комунікації з криптовалютними біржами. Реалізувати програмні модулі для інтеграції з API криптовалютних бірж, забезпечивши функціонал для автентифікації, отримання ринкових даних в реальному часі та управління торговими ордерами.

3. Розробка функціоналу аналізу даних з бірж для прийняття рішень щодо торгівлі криптоактивами. Створити модуль, відповідальний за аналіз отриманих ринкових даних, розрахунок технічних індикаторів та реалізацію логіки торгових стратегій для генерації сигналів на купівлю або продаж криптоактивів.

4. Розробка функціоналу конфігурування стратегій торгівлі. Забезпечити можливість гнучко визначати та налаштовувати параметри торгових стратегій, включаючи вибір торгових інструментів, індикаторів, умов для входу та виходу з позиції, а також параметрів управління ризиками (наприклад, стоп-лос, тейк-профіт).

5. Комплексне тестування і налагодження програмної системи. Провести ретельне тестування всіх розроблених модулів та системи в цілому, включаючи перевірку коректності інтеграції з API бірж (з використанням тестових середовищ бірж, зокрема випробування системи в хмарному сервісі AWS), функціональне тестування торгових стратегій та перевірку надійності роботи бота в різних ринкових умовах.

2 АНАЛІЗ МЕТОДІВ ТОРГІВЛІ КРИПТОАКТИВАМИ

Розробка ефективного торгового бота для криптовалютних бірж неможлива без глибокого розуміння теоретичних засад трейдингу та аналізу існуючого інструментарію для його автоматизації. Цей розділ присвячений дослідженню фундаментальних аспектів торгівлі криптоактивами, включаючи ключові методи аналізу ринку, поширені торгові стилі та ефективні стратегії. Далі буде проведено огляд наявних на ринку програмних рішень для автоматизації торгівельних операцій, що дозволить виявити їхні переваги, недоліки та визначити нішу для розробки власної, більш гнучкої та адаптованої системи.

2.1 Загальні підходи та види торгівлі криптоактивами

Ефективна торгівля криптоактивами базується на розумінні ключових аналітичних методів, розповсюджених торгових стилів та специфічних стратегій [4]. Цей підрозділ розкриває фундаментальні аспекти, знання яких є необхідною передумовою для розробки автоматизованих торгових систем, здатних успішно функціонувати на динамічному криптовалютному ринку.

2.1.1 Методи аналізу крипторинку

Для успішного прогнозування динаміки цін на криптоактиви та визначення найсприятливіших моментів для входу в ринок або виходу з нього, трейдери послуговуються трьома ключовими підходами до аналізу: технічним, фундаментальним та аналізом настроїв ринку. Кожен із цих методів має свої особливості та інструментарій, а їх комплексне застосування дозволяє отримати більш повне уявлення про поточну ринкову ситуацію та потенційні напрямки руху цін [4].

Технічний аналіз зосереджується на вивченні минулої поведінки ринку, зокрема історичної динаміки цін та обсягів торгів. Прихильники цього підходу вважають, що всі значущі фактори вже враховані в ціні активу, а історичні цінові моделі мають тенденцію повторюватися. Для виявлення таких закономірностей і трендів трейдери активно використовують графічний аналіз та математичні індикатори [3]. Серед найпоширеніших інструментів технічного аналізу можна виділити ковзні середні (Moving Averages), які допомагають згладити цінові коливання та визначити загальний напрямок тренду; індекс відносної сили (RSI), що вказує на перекупленість або перепроданість активу; MACD (Moving Average Convergence Divergence), який відображає співвідношення між двома ковзними середніми; смуги Болліджера, що окреслюють діапазон цінових коливань; та рівні Фібоначчі, які допомагають визначити потенційні рівні підтримки та опору. Основною перевагою технічного аналізу є його універсальність, оскільки він може бути застосований до будь-якого фінансового інструменту, що має достатню історію торгів [4].

На відміну від технічного аналізу, фундаментальний аналіз спрямований на оцінку так званої "внутрішньої" або "справедливої" вартості криптоактиву. Цей підхід передбачає глибоке вивчення всіх аспектів, що можуть впливати на довгострокові перспективи проекту. До уваги беруться такі фактори, як технологічна інноваційність та унікальність блокчейн-проекту, досвід та репутація команди розробників, модель розподілу та використання токенів (токеноміка), наявність стратегічних партнерств та інтеграцій, а також конкурентне середовище [3]. Крім того, фундаментальний аналіз враховує вплив зовнішніх чинників, таких як загальноекономічна ситуація, важливі новини у сфері криптовалют та зміни в законодавчому регулюванні різних країн. Головна мета фундаментального аналізу – визначити, чи є актив недооціненим або переоціненим ринком, що дозволяє приймати обґрунтовані довгострокові інвестиційні рішення [4].

Третім важливим компонентом ринкового аналізу є аналіз настроїв ринку, також відомий як Sentiment Analysis. Цей метод фокусується на виявленні

загального емоційного фону та ставлення учасників ринку до конкретного криптоактиву або ринку в цілому. Інформація для такого аналізу збирається з різноманітних джерел, включаючи соціальні мережі (Twitter, Reddit, Telegram), тематичні форуми, новинні портали та блоги. Переважання позитивних настроїв, оптимістичних прогнозів та активного обговорення проекту може свідчити про потенційне зростання попиту та, відповідно, ціни активу. Навпаки, поширення негативної інформації, страху та невпевненості може призвести до зниження інтересу інвесторів та падіння ціни. Аналіз настроїв ринку рідко використовується як самостійний інструмент для прийняття торгових рішень; частіше він слугує важливим доповненням до технічного та фундаментального аналізу, дозволяючи врахувати психологічний аспект поведінки ринку [4].

2.1.2 Стилі торгівлі криптоактивами

Залежно від часових горизонтів утримання відкритих позицій та інтенсивності здійснення торгових операцій, у світі торгівлі криптоактивами прийнято розрізняти декілька основних стилів. Кожен стиль має на увазі власний підхід до ризиків, вимагає різного рівня залученості та набору навичок від трейдера. Вибір конкретного стилю часто залежить від індивідуальних цілей інвестора, його темпераменту та кількості часу, який він готовий присвятити торгівлі [4].

Одним із найпоширеніших підходів, особливо серед новачків та тих, хто вірить у довгостроковий потенціал криптовалют, є HODLing. Цей термін, що походить від навмисно спотвореного англійського слова "hold" (тримати), означає довгострокове інвестування. Інвестори, які дотримуються цього стилю, купують криптоактиви з розрахунком на їх суттєве зростання в ціні протягом тривалого періоду – місяців, а то й років. "Ходлери" зазвичай менше уваги приділяють короткостроковим коливанням ринку та більше орієнтуються на фундаментальні

показники проектів, такі як технологія, команда, перспективи розвитку та загальний потенціал ринкової ніші [5].

Наступний стиль – Swing Trading – передбачає середньострокову торгівлю. Свінг-трейдери прагнуть отримати прибуток від цінових "махів" або коливань, утримуючи позиції від кількох днів до кількох тижнів. Вони аналізують графіки цін, шукаючи потенційні точки розвороту трендів або продовження існуючих рухів, щоб увійти в угоду на початку коливання та вийти з неї, коли рух, на їхню думку, вичерпується. Цей стиль вимагає хорошого розуміння технічного аналізу та вміння ідентифікувати середньострокові тренди [5].

Для тих, хто віддає перевагу більш активній торгівлі, існує Day Trading, або внутрішньоденна торгівля. Дей-трейдери відкривають та закривають усі свої позиції протягом одного торгового дня, не переносячи їх на наступний. Їхня мета – отримати прибуток від невеликих цінових рухів, що відбуваються протягом дня. Цей стиль вимагає значної концентрації, швидкої реакції на ринкові зміни та вміння приймати рішення в умовах обмеженого часу. Дей-трейдери часто використовують технічний аналіз та стежать за актуальними новинами, які можуть вплинути на ціни активів у короткостроковій перспективі [5].

Найбільш інтенсивним та короткостроковим стилем торгівлі є скальпінг. Скальпери здійснюють велику кількість угод протягом торгового дня, намагаючись заробити на дуже малих коливаннях цін – буквально кількох пунктах або "піпсах". Тривалість утримання позиції може становити від кількох секунд до кількох хвилин. Успішний скальпінг вимагає надзвичайної точності у вході та виході з угод, високого рівня дисципліни, здатності швидко аналізувати ринкову ситуацію та наявності торгового рахунку з низькими комісіями, оскільки часті угоди можуть суттєво збільшити транзакційні витрати [4].

2.1.3 Загальні торгові стратегії на ринку

На основі різних підходів до аналізу ринкової ситуації та обраних стилів торгівлі трейдери формують конкретні торгові стратегії. Стратегія визначає чіткі правила для входу в угоду, виходу з неї та управління капіталом. Розуміння та правильне застосування різноманітних стратегій є ключовим елементом для досягнення успіху на висококонкурентному та волатильному ринку криптоактивів [5].

Однією з найпопулярніших є торгівля за трендом. Ця стратегія полягає у відкритті позицій у напрямку поточного домінуючого ринкового руху. Трейдери, що використовують цей підхід, намагаються ідентифікувати висхідний тренд (коли ціни переважно зростають) для здійснення покупок, або низхідний тренд (коли ціни переважно падають) для відкриття коротких позицій (продажу). Успіх такої торгівлі значною мірою залежить від вміння правильно розпізнавати початок нового тренду, його силу та потенційні точки розвороту або завершення. Для цього активно використовуються інструменти технічного аналізу, такі як трендові лінії, ковзні середні та різноманітні осцилятори [2].

Протилежним підходом є контртрендова торгівля, також відома як стратегія повернення до середнього (Mean Reversion). Трейдери, що дотримуються цієї стратегії, діють проти основного поточного руху ринку. Вони виходять з припущення, що після сильного цінового руху в одному напрямку ринок часто коригується, повертаючись до свого середнього значення. Таким чином, вони намагаються купувати, коли актив вважається перепроданим (після значного падіння), або продавати, коли він перекуплений (після значного зростання), розраховуючи на розворот ціни. Ця стратегія може бути ризикованою, оскільки "ловити ножі, що падають" або йти проти сильного тренду вимагає точного розрахунку та суворого управління ризиками [2].

Арбітраж представляє собою стратегію, що базується на використанні різниці в цінах одного й того ж криптоактиву на різних торгових майданчиках (біржах) або в різних торгових парах на одній біржі. Арбітражери одночасно

купують актив там, де він дешевший, і продають там, де він дорожчий, отримуючи прибуток від цієї цінової різниці за вирахуванням торгових комісій. Ця стратегія вимагає швидкого реагування та доступу до інформації про ціни на різних платформах у режимі реального часу, оскільки арбітражні можливості зазвичай існують недовго [3].

Стратегія маркет-мейкінгу полягає у виставленні одночасних ордерів на купівлю (бід) та продаж (аск) певного криптоактиву. Маркет-мейкери прагнуть отримати прибуток зі спреду – різниці між найкращою ціною купівлі та найкращою ціною продажу. Окрім отримання прибутку, маркет-мейкери відіграють важливу роль у забезпеченні ліквідності ринку, тобто можливості для інших учасників швидко купувати або продавати активи без значного впливу на їхню ціну. Ця стратегія часто реалізується за допомогою автоматизованих торгових систем (ботів) [3].

Ще однією поширеною стратегією є торгівля на новинах. Вона передбачає швидку реакцію на важливі події, оголошення або новини, які можуть суттєво вплинути на настрої ринку та ціну криптоактиву. Це можуть бути новини про технологічні оновлення проекту, укладання значних партнерств, зміни в регуляторній політиці, хакерські атаки на біржі або інші глобальні події, що впливають на фінансові ринки. Трейдери, що використовують цю стратегію, намагаються спрогнозувати реакцію ринку на новину та відкрити відповідну позицію до того, як ціна повністю відреагує [5].

2.2 Огляд існуючих рішень для автоматизації торгівлі

Ринок програмного забезпечення для автоматизації торгівлі криптоактивами пропонує широкий спектр рішень, що варіюються від готових комерційних продуктів до інструментарію для створення власних ботів. Розуміння їхніх характеристик є важливим для позиціонування та обґрунтування розробки власної системи, орієнтованої на специфічні потреби та гнучкість.

2.2.1 Комерційні платформи та інтегровані рішення

Комерційні платформи є домінуючим сегментом ринку, пропонуючи користувачам готовий набір інструментів, попередньо налаштовані торгові стратегії та технічну підтримку. Такі платформи, як 3Commas, надають широкий спектр функцій, включаючи інструменти для ручного управління з розширеними опціями (SmartTrade), популярні GRID-боти та DCA-боти (Dollar-Cost Averaging) [5]. Важливою їхньою особливістю часто є можливість підключення до зовнішніх сигналів та забезпечення сумісності з великою кількістю криптовалютних бірж. До переваг таких рішень зазвичай відносять інтуїтивно зрозумілий інтерфейс, значний вибір доступних ботів, гнучкі налаштування, а також додаткові можливості, такі як копіювання стратегій інших трейдерів та функція паперової торгівлі для тестування без фінансових ризиків.

Інша категорія – це хмарні сервіси на кшталт CryptoHopper, які пропонують доступ до маркетплейсу готових стратегій, інструментарій для бектестінгу (перевірки ефективності стратегій на історичних даних), арбітражних ботів та іноді візуальні конструктори для створення власних торгових логік. Перевагою хмарних рішень є відсутність необхідності встановлення програмного забезпечення на власний сервер користувача та зазвичай широка підтримка різних бірж [5].

Окремо варто виділити криптовалютні біржі, такі як Pionex, що інтегрують власні торгові боти безпосередньо у свою платформу. Вони можуть пропонувати близько 16 типів ботів, включаючи GRID-боти, арбітражні боти та DCA-боти, часто надаючи їх безкоштовно, а монетизацію здійснюючи за рахунок торгових комісій. Простота налаштування та наявність мобільних додатків роблять їх привабливими для певного сегмента користувачів [5].

Незважаючи на різноманітність функцій, головним недоліком більшості комерційних рішень є необхідність регулярних фінансових витрат на підписку, причому доступ до найбільш просунутих функцій часто відкривається лише на дорожчих тарифних планах. Крім того, вони можуть мати суттєві обмеження щодо гнучкості налаштування дійсно унікальних торгових стратегій [5].

2.2.2 Рішення з відкритим вихідним кодом та власна розробка

Для розробників та досвідчених трейдерів, які прагнуть максимальної кастомізації та контролю, альтернативою комерційним продуктам є використання рішень з відкритим вихідним кодом (open-source) та розробка власних торгових ботів "з нуля" або на базі спеціалізованих фреймворків.

Розробка власної системи, відкриває можливості, які часто недоступні або обмежені в готових комерційних платформах. Це включає створення унікальних торгових алгоритмів, інтеграцію нестандартних джерел даних для аналізу, повний контроль над логікою прийняття рішень та можливість тонкого налаштування кожного аспекту торгової системи. Хоча цей шлях вимагає значних початкових інвестицій часу та зусиль на розробку та тестування, він може забезпечити значні переваги в довгостроковій перспективі, особливо для реалізації складних та нетипових торгових стратегій.

2.2.3 Ключові переваги розробки власної системи автоматизації торгівлі

Обґрунтування розробки власного торгового бота випливає з потреби подолати обмеження існуючих комерційних рішень та отримати низку суттєвих переваг, що є критичними для реалізації специфічних торгових підходів.

Неперевершена гнучкість та кастомізація: власна розробка надає повну свободу у проектуванні архітектури та реалізації будь-якої, навіть найскладнішої та унікальної торгової логіки. Немає обмежень щодо типів індикаторів, комбінацій сигналів чи умов для входу/виходу з позиції, які можуть нав'язуватися готовими платформами [5, 6].

Модульність та масштабованість: спроектована з нуля система може бути побудована на модульній основі, що значно спрощує подальше розширення функціоналу: додавання підтримки нових криптовалютних бірж, інтеграцію нових аналітичних інструментів чи торгових стратегій.

Пряма та оптимізована інтеграція з API бірж: розробка власного коннектора до API біржі дозволяє максимально ефективно отримувати ринкові дані в реальному часі (ціни, обсяги, біржовий стакан) та надійно керувати торговими ордерами, уникаючи можливих затримок або обмежень сторонніх сервісів.

Реалізація комплексного та багатофакторного аналізу: власна система дозволяє інтегрувати не лише стандартні технічні індикатори, але й підключати альтернативні джерела даних, такі як аналіз новинного фону, дані соціальних мереж (sentiment analysis) або навіть власні предиктивні моделі, створюючи по-справжньому комплексну основу для прийняття рішень.

Повний контроль над безпекою та конфіденційністю даних: усі критично важливі дані, зокрема API-ключі до бірж, історія угод та параметри стратегій, залишаються під повним контролем розробника/користувача, що мінімізує ризики, пов'язані з передачею цієї інформації третім сторонам [6].

Економічна ефективність у довгостроковій перспективі: хоча початкові витрати на розробку можуть бути значними, у довгостроковій перспективі відсутність регулярних щомісячних або щорічних платежів за підписку, характерних для комерційних платформ, може зробити власне рішення економічно вигіднішим. Витрати в основному зводяться до хостингу системи (наприклад, у хмарних сервісах типу AWS, як передбачено у даному проекті) та її технічної підтримки [7].

Оптимізація під конкретний стиль торгівлі та управління ризиками: можливість точно налаштувати параметри управління ризиками (наприклад, стоп-лос, тейк-профіт, розмір позиції) відповідно до індивідуальних вимог та толерантності до ризику.

Саме ці переваги роблять розробку власної системи автоматизації торгівлі привабливою для трейдерів та розробників, які прагнуть створити потужний, гнучкий та повністю контрольований інструмент для роботи на ринку криптоактивів, що відповідає специфіці їхніх торгових підходів та дослідницьких завдань.

3 ПРОЕКТУВАННЯ АРХІТЕКТУРИ ТА ВИБІР ЗАСОБІВ РОЗРОБКИ

Перехід від теоретичного аналізу предметної області та існуючих рішень до безпосередньої розробки програмного продукту вимагає ретельного проектування. Цей етап є фундаментальним, оскільки закладає основи майбутньої системи, визначає її структуру, взаємозв'язки між компонентами та технологічний стек, що буде використовуватися. Якість проектних рішень безпосередньо впливає на надійність, гнучкість, масштабованість та загальну успішність розроблюваного торгового бота. В даному розділі будуть детально розглянуті вимоги до системи, спроектована її архітектура з описом ключових модулів, обґрунтовано вибір мови програмування, основних фреймворків, бібліотек та платформи для розгортання.

3.1 Вимоги до системи

Перед тим, як розпочати безпосереднє проектування архітектури та вибір технологій, необхідно чітко визначити вимоги до майбутньої системи. Формулювання детальних та однозначних вимог дозволяє забезпечити відповідність кінцевого продукту очікуванням та поставленим завданням.

3.1.1 Функціональні вимоги

Функціональні вимоги визначають основні можливості та поведінку розроблюваного торгового бота. Основуючись на поставленій задачі до системи висуваються наступні функціональні вимоги.

1. Взаємодія з криптовалютними біржами. Система повинна забезпечувати підключення щонайменше до однієї криптовалютної біржі

(наприклад, Binance) через її програмний інтерфейс додатка (API), включаючи автентифікацію за допомогою API-ключів користувача. Архітектура має передбачати можливість подальшого розширення для інтеграції з іншими торговими майданчиками.

2. Отримання ринкових даних. Бот має отримувати історичні ринкові дані, такі як котирування у вигляді свічок (Klines) для заданих торгових пар та часових інтервалів (таймфреймів). Також необхідне отримання актуальної ціни для торгових пар у режимі, наближеному до реального часу, та інформації про торговий інструмент, включаючи специфікації торгівлі (мінімальний обсяг ордера, крок ціни тощо).

3. Аналіз ринкових даних та генерація торгових сигналів. Система повинна реалізовувати розрахунок ключових технічних індикаторів, зокрема MACD (Moving Average Convergence Divergence), з можливістю використання різних типів ковзних середніх (SMA, EMA, SMMA), та RSI (Relative Strength Index). На основі аналізу комбінації значень цих індикаторів та відповідно до логіки визначених торгових стратегій (наприклад, перетин індикатором MACD нульової лінії, досягнення індикатором RSI заданих рівнів перекупленості/перепроданості), бот генерує сигнали для відкриття позицій на купівлю або продаж.

4. Управління торговими ордерами. Ключовою функцією є автоматичне розміщення ринкових (Market) ордерів на купівлю та продаж. Система має розраховувати обсяг ордера, базуючись на попередньо визначеній сумі та поточній ринковій ціні, з обов'язковим коригуванням розрахованого обсягу відповідно до правил щодо мінімального розміру лоту (Lot Size), встановлених біржею.

5. Управління ризиками. Бот повинен реалізовувати механізми управління ризиками шляхом автоматичного встановлення та моніторингу ордерів типу Stop-Loss (для обмеження потенційних збитків) та Take-Profit (для фіксації прибутку).

6. Відстеження стану угод. Система має відстежувати статус всіх активних ордерів та відкритих позицій.

7. Конфігурація торгових стратегій. Необхідно забезпечити можливість гнучкого завантаження та управління параметрами торгових стратегій із зовнішнього конфігураційного файлу (наприклад, у форматі JSON). Кожна стратегія повинна визначати такі параметри, як торговий символ, часовий інтервал для аналізу, суму для відкриття ордера, відсоткові рівні для Stop-Loss та Take-Profit, а також специфічні налаштування для кожного використовуваного технічного індикатора (наприклад, періоди розрахунку).

8. Підтримка множинних стратегій. Система має бути спроможною одночасно виконувати декілька торгових стратегій, потенційно для різних торгових пар або з різними налаштуваннями для однієї пари.

9. Логування операцій: для забезпечення можливості аналізу роботи та діагностики потенційних проблем, система повинна вести детальне логування всіх ключових подій та дій. Це включає логування процесу отримання даних, розрахунку значень індикаторів, генерації торгових сигналів, розміщення та виконання ордерів, спрацювання ордерів Stop-Loss/Take-Profit, а також фіксацію будь-яких помилок або виняткових ситуацій.

3.1.2 Нефункціональні вимоги

Нефункціональні вимоги визначають атрибути якості та експлуатаційні характеристики розроблюваного торгового бота. До системи висуваються наступні нефункціональні вимоги.

1. Продуктивність. Система повинна оперативно обробляти ринкові дані та приймати торгові рішення з мінімальною можливою затримкою для забезпечення своєчасного реагування на швидкі зміни ринкової кон'юнктури. Час відповіді при взаємодії з API криптовалютної біржі та безпосереднє виконання торгових операцій мають відповідати високим стандартам швидкодії, встановленим самою біржею та очікуваннями користувача.

2. Надійність та відмовостійкість. Система має бути спроектована для стабільної та безперервної роботи в цілодобовому режимі (24/7), враховуючи природу функціонування криптовалютних ринків. Необхідно забезпечити коректну обробку потенційних збоїв та помилок, що можуть виникати при взаємодії з API біржі (наприклад, проблеми з мережевим з'єднанням, помилки з боку API біржі, тимчасова недоступність окремих сервісів біржі), без повної зупинки функціонування торгового бота.

3. Безпека. Першочерговою вимогою є забезпечення безпечного зберігання та використання API-ключів користувача, які надають доступ до його акаунту на біржі. Конфігураційні файли або будь-які інші сховища, що містять API-ключі, не повинні зберігатися у системах контролю версій у незашифрованому або легкодоступному вигляді [8].

4. Масштабованість. Архітектура системи має бути спроектована таким чином, щоб дозволяти відносно просте додавання підтримки нових криптовалютних бірж у майбутньому. Також важливою є можливість легкого розширення набору підтримуваних технічних індикаторів та типів торгових стратегій без необхідності значної перебудови основного коду системи. Система повинна бути здатною ефективно обробляти зростаючу кількість торгових пар та одночасно активних стратегій без суттєвого погіршення загальної продуктивності.

5. Розгортання. Система повинна бути придатною для розгортання та стабільного функціонування в хмарному середовищі. Зокрема, згідно з індивідуальним завданням, має бути забезпечена можливість розгортання на платформі AWS (Amazon Web Services) [9].

3.2 Проектування архітектури системи

Після визначення функціональних та нефункціональних вимог наступним логічним кроком є проектування архітектури системи. Архітектура програмного забезпечення визначає основні компоненти системи, їхні взаємозв'язки та

принципи взаємодії. Грамотно спроектована архітектура є запорукою створення надійної, гнучкої, масштабованої та легко підтримуваної системи, що відповідає всім висунутим вимогам. У цьому підрозділі буде детально розглянуто загальну архітектуру розроблюваного торгового бота та сценарії його роботи.

3.2.1 Загальна архітектура

Загальна архітектура розроблюваного торгового бота представлена на рисунку 3.1. Дана діаграма ілюструє ключові компоненти системи та потоки даних і керування між ними.

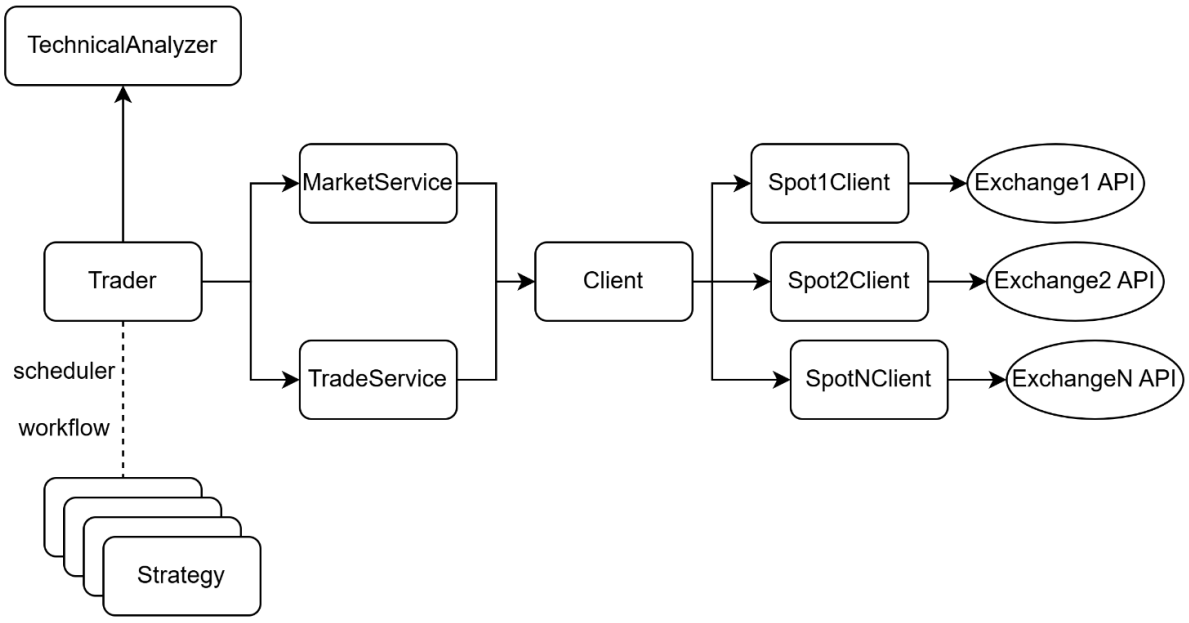

Рисунок 3.1 – Загальна архітектура торгового бота

Центральним елементом системи є модуль Trader. Робота модуля Trader ініціюється зовнішнім планувальником, який запускає основний робочий цикл з визначеною періодичністю. Для своєї роботи Trader завантажує та використовує одну або декілька торгових стратегій (Strategy), визначених користувачем. Ці стратегії містять налаштування торгових інструментів, параметри індикаторів та

правила для прийняття рішень. Конфігурація цих стратегій завантажується із зовнішнього файлу.

Для виконання своїх завдань модуль Trader взаємодіє з декількома спеціалізованими сервісами.

TechnicalAnalyzer. Модуль Trader передає ринкові дані цьому компоненту для проведення технічного аналізу. TechnicalAnalyzer обчислює необхідні індикатори (наприклад, MACD, RSI) на основі отриманих даних та параметрів активної стратегії. Результати аналізу (значення індикаторів, можливі торгові сигнали) повертаються до Trader для подальшого прийняття рішень.

MarketService. Цей сервіс відповідає за всю взаємодію з ринковими даними криптовалютної біржі. Trader звертається до MarketService для отримання актуальних цін, історичних даних (Klines), інформації про торгові пари та іншої ринкової інформації, необхідної для аналізу та виконання стратегій.

TradeService. Після того, як Trader на основі аналізу від TechnicalAnalyzer та логіки стратегії приймає рішення про відкриття або закриття позиції, він делегує виконання цієї операції сервісу TradeService. Цей сервіс відповідає за формування та відправку торгових ордерів на біржу, а також за управління активними ордерами (наприклад, моніторинг Stop-Loss/Take-Profit).

Обидва сервіси, MarketService та TradeService, для безпосередньої взаємодії з API криптовалютних бірж використовують узагальнений компонент Client. Цей компонент виступає як абстракція, що дозволяє працювати з різними біржами. На діаграмі показано, що Client може інстанціювати специфічні клієнти для кожної біржі (Spot1Client, Spot2Client, SpotNClient), кожен з яких взаємодіє з відповідним Exchange API (API Біржі 1, API Біржі 2, API Біржі N). Така структура забезпечує гнучкість та можливість розширення системи для підтримки декількох торгових майданчиків без необхідності суттєвих змін в основній логіці модулів MarketService та TradeService.

3.2.2 Сценарій роботи системи

Для наочного представлення логіки функціонування розроблюваного торгового бота ключовим є опис основного сценарію його роботи в рамках одного торговельного циклу для кожної активної стратегії. Цей процес ілюструється діаграмою активності, наведеною на рисунку 3.2.

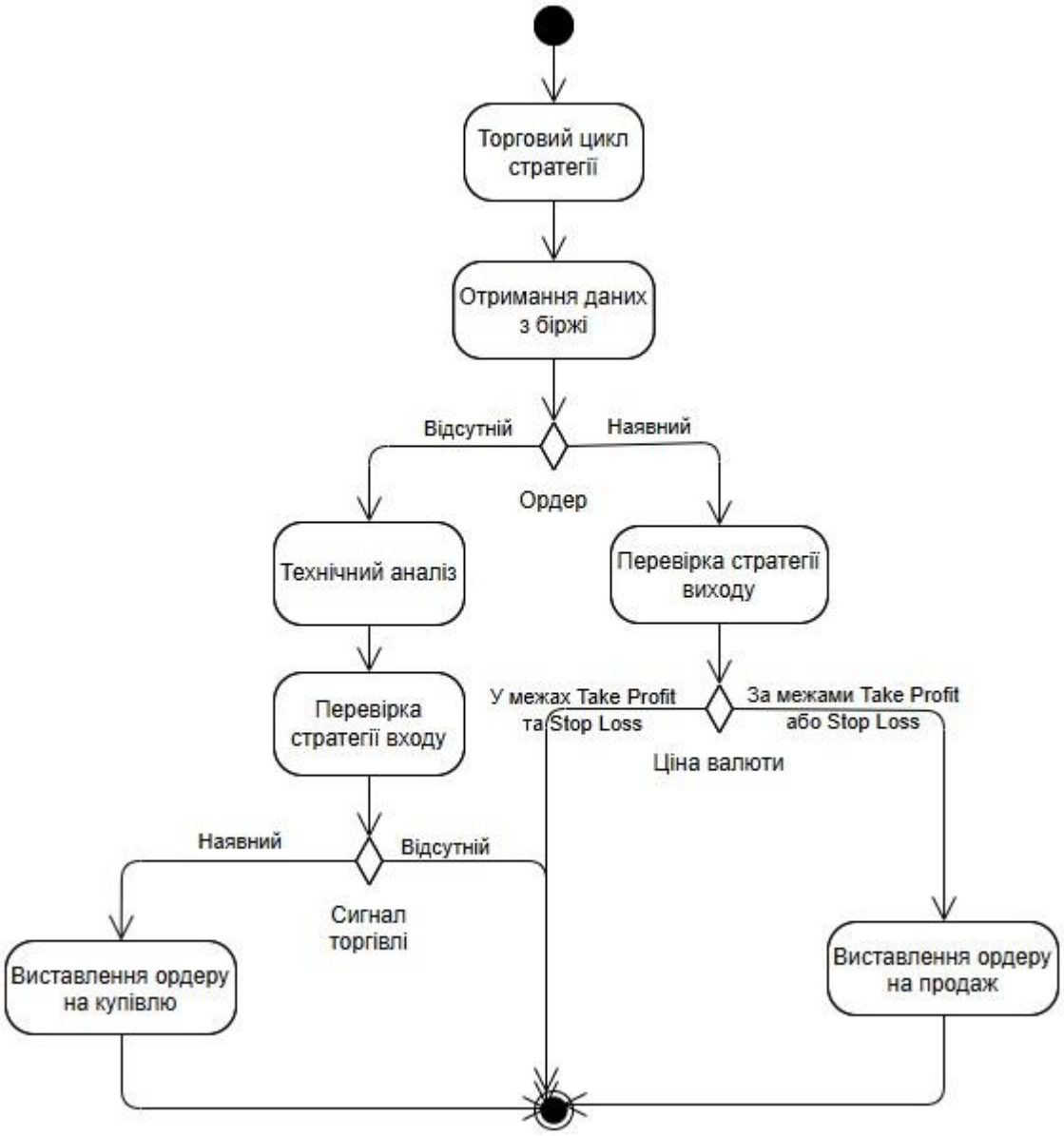

Рисунок 3.2 – Діаграма активності торгового циклу стратегії

Сценарій роботи системи для кожної торгової стратегії, яка запускається за розкладом, виглядає так:

1. Ініціація стратегії: обирається одна зі стратегій із конфігураційного файлу.

2. Отримання ринкових даних: система запитує з біржі актуальні котирування (Klines), потрібні для технічного аналізу.

3. Перевірка активного ордера: визначається, чи є вже відкрита позиція для цієї стратегії й інструменту. Далі обирається відповідна гілка обробки.

4. Відсутність активного ордера:

- технічний аналіз: розрахунок індикаторів (MACD, RSI) на основі стратегії;
- оцінка сигналу на вхід: порівняння значень індикаторів з умовами стратегії;
- сигнал є: формується ордер на купівлю, розраховується обсяг, встановлюються Stop-Loss і Take-Profit;
- сигналу немає: система завершує цикл без дій.

5. Наявність активного ордера:

- аналіз поточної ціни: перевірка на досягнення Take-Profit або Stop-Loss;
- умова виходу виконується: система формує ордер на продаж;
- умова не виконується: позиція супроводжується, дій не виконується.

6. Завершення циклу: після обробки стратегії система переходить до наступної або чекає нового запуску.

Цей процес забезпечує автоматизований аналіз ринку та прийняття торгових рішень.

3.3 Вибір засобів розробки

Вибір правильного набору технологій та інструментів є критично важливим етапом у розробці будь-якого програмного забезпечення, і торговий бот не є винятком. Від цього вибору залежить швидкість розробки, продуктивність, надійність, масштабованість та легкість підтримки готової системи. У цьому підрозділі буде обґрунтовано вибір мови програмування, основних фреймворків та бібліотек, а також платформи для розгортання розроблюваного торгового бота, спираючись на функціональні та нефункціональні вимоги, а також на специфіку завдання.

3.3.1 Мова програмування

Для реалізації програмної системи торгового бота було обрано мову програмування Java. Цей вибір зумовлений низкою переваг, які роблять Java вдалим інструментом для розробки складних, високонавантажених та надійних серверних додатків.

1. Надійність та стабільність. Java є однією з найбільш зрілих та перевірених часом мов програмування. Її сувора типізація, механізми обробки винятків та автоматичне управління пам'яттю (Garbage Collector) сприяють створенню стабільних додатків, що є критично важливим для торгового бота, який має працювати безперервно та без збоїв [9, 10].

2. Багатопотоковість. Торговий бот повинен одночасно виконувати декілька завдань: отримувати дані з біржі, аналізувати ринок за кількома стратегіями, моніторити активні ордери. Java має потужні вбудовані засоби для роботи з багатопотоковістю (класи пакету java.util.concurrent), що дозволяє ефективно реалізовувати паралельну обробку та забезпечувати високу продуктивність системи [13, 14].

3. Об'єктно-орієнтований підхід (ООП). Java є повністю об'єктно-орієнтованою мовою, що дозволяє проектувати систему у вигляді набору взаємодіючих об'єктів. Це сприяє створенню модульної, гнучкої та легко масштабованої архітектури, що відповідає вимогам до розроблюваного бота.

4. Платформонезалежність (Write Once, Run Anywhere). Хоча для даного проекту планується розгортання на конкретній платформі (AWS), принципова можливість запуску Java-додатків на будь-якій операційній системі, де встановлена Java Virtual Machine (JVM), є додатковою перевагою [11, 12].

5. Досвід розробки. Наявний досвід розробки на мові Java також став вагомим фактором при виборі, дозволяючи зосередитися на вирішенні бізнес-задач, а не на вивченні нового синтаксису чи парадигм.

Враховуючи ці фактори, Java є оптимальним вибором для створення надійного та ефективного торгового бота, здатного виконувати складні аналітичні завдання та взаємодіяти з фінансовими API.

3.3.2 Фреймворки та бібліотеки

Для прискорення процесу розробки, забезпечення належної структури програмного додатку та використання надійних, перевірених рішень для взаємодії з зовнішніми сервісами, було обрано наступні ключові фреймворки та бібліотеки.

- Spring Boot. Цей фреймворк обраний як основа для побудови додатку. Spring Boot є розширенням платформи Spring, що суттєво полегшує створення автономних Java-додатків, готових до розгортання. Його ключові переваги для проекту включають автоконфігурацію, що зменшує кількість шаблонних налаштувань; можливість легкого створення виконуваних JAR-файлів для спрощення розгортання; ефективне управління залежностями через механізм "starters"; вбудовану підтримку планувальника завдань (@Scheduled), яка використовується для періодичного запуску торгового циклу; а також реалізацію принципу впровадження залежностей (Dependency Injection), що сприяє

модульності та тестуванню коду. Широка екосистема Spring також надає можливості для подальшого розширення функціоналу [15].

- Binance Connector Java. Це офіційна бібліотека від криптовалютної біржі Binance, призначена для взаємодії з її торговим та ринковим API з Java-додатків. Використання офіційного конектора гарантує сумісність з актуальними версіями API біржі та забезпечує надійний спосіб комунікації. Бібліотека надає готові методи для доступу до всіх необхідних ендпоінтів API Binance, включаючи отримання ринкових даних (історичні Klines, поточні ціни) та виконання торгових операцій (розміщення ордерів, перевірка їх статусу). Вона також інкапсулює складність формування запитів, обробки відповідей та механізмів автентифікації, дозволяючи розробнику зосередитись на бізнес-логіці торгового бота. У проекті ця бібліотека є основою для модулів, що відповідають за отримання ринкових даних та здійснення торгових операцій.

3.3.3 Платформа для розгортання

Для забезпечення стабільної та безперебійної роботи торгового бота, який повинен функціонувати цілодобово, обрано хмарну платформу Amazon Web Services (AWS). Цей вибір відповідає індивідуальному завданню та сучасним практикам розгортання подібних систем.

Ключовими факторами, що зумовили вибір AWS, є [16, 17, 18, 19]:

- висока надійність та доступність: AWS є лідером ринку хмарних послуг, що гарантує високий рівень безвідмовної роботи інфраструктури, що критично для цілодобової торгівлі;
- масштабованість ресурсів: платформа дозволяє гнучко налаштовувати та динамічно змінювати обчислювальні потужності (віртуальні сервери, пам'ять, сховище) відповідно до потреб торгового бота, забезпечуючи можливість росту системи;

- широкий набір сервісів: AWS надає різноманітні інструменти, які можуть бути використані для розгортання та підтримки. Зокрема, Amazon EC2 (Elastic Compute Cloud) буде використано для запуску Java-додатку бота на віртуальному сервері. Для моніторингу та логування може бути задіяний сервіс AWS CloudWatch, а для зберігання даних – Amazon S3.
- безпека та керованість: AWS пропонує розвинені засоби для забезпечення безпеки розгорнутих додатків та даних, а також інструменти для автоматизації управління інфраструктурою;
- розгортання Java-додатку, створеного за допомогою Spring Boot, на віртуальному сервері Amazon EC2 з встановленим середовищем виконання Java (JRE) є відносно стандартним процесом і дозволить забезпечити необхідні умови для ефективної роботи торгового бота;

4 ОПИС РЕАЛІЗАЦІЇ ПРОГРАМНИХ МОДУЛІВ

Після етапів аналізу предметної області, проектування архітектури та вибору засобів розробки, цей розділ присвячений детальному опису програмної реалізації ключових модулів торгового бота. Буде розглянуто внутрішню структуру та логіку роботи кожного з основних компонентів системи, що були визначені на етапі проектування. Особливу увагу буде приділено реалізації модуля технічного аналізу, включаючи опис алгоритмів розрахунку використовуваних індикаторів та їх інтерпретацію в контексті торгових рішень.

4.1 Модуль конфігурації стратегій

Модуль конфігурації стратегій відіграє важливу роль у забезпеченні гнучкості та адаптивності торгового бота. Його основне завдання полягає в завантаженні та наданні доступу до параметрів торгових стратегій, які визначають поведінку системи на ринку. Реалізація цього модуля дозволяє користувачеві або оператору системи налаштовувати торгові підходи без необхідності прямого втручання в програмний код, що значно спрощує управління та експериментування з різними торговими ідеями.

В розробленій системі цей модуль реалізований таким чином, що параметри торгових стратегій зберігаються у зовнішньому файлі у форматі JSON. Такий підхід відокремлює дані конфігурації від логіки програми, що є хорошою практикою розробки. Файл конфігурації містить масив об'єктів, кожен з яких представляє окрему торгову стратегію з усіма необхідними атрибутами: назва, торговий інструмент (символ), часовий інтервал для аналізу, обсяг інвестиції на одну угоду, відсоткові рівні для ордерів Stop-Loss та Take-Profit, а також специфічні налаштування для кожного технічного індикатора, що використовується в стратегії (наприклад, періоди для MACD та RSI, типи ковзних середніх).

Процес завантаження цих конфігурацій відбувається під час ініціалізації системи. Спеціалізований сервіс відповідає за читання даних з JSON-файлу, їх перетворення у відповідні об'єкти Java та збереження у пам'яті для подальшого використання. Це забезпечує швидкий доступ до параметрів стратегій під час виконання основного торгового циклу. Головний модуль торгової логіки звертається до цього сервісу для отримання списку активних стратегій, які потім послідовно обробляються.

4.2 Модуль взаємодії з криптовалютною біржею

Модуль взаємодії з криптовалютною біржею є критично важливим компонентом торгового бота, оскільки він забезпечує безпосередній зв'язок системи з торговим майданчиком. У розробленій системі цей функціонал інкапсульовано у двох основних сервісних класах, які використовують офіційну бібліотеку binance-connector-java для взаємодії з API біржі Binance.

4.2.1 Отримання ринкових даних

Сервіс, що відповідає за ринкові дані, призначений для збору усієї необхідної ринкової інформації.

1. Отримання історичних даних. Для аналізу та розрахунку технічних індикаторів система потребує історичних даних про ціни. Цей сервіс має функціонал для запиту історичних даних, приймаючи на вхід символ торгової пари, часовий інтервал та ліміт на кількість свічок. Він формує відповідний запит до API Binance, використовуючи для цього спеціалізовані компоненти офіційної бібліотеки для взаємодії з біржею. Отримана відповідь у форматі JSON далі обробляється внутрішнім компонентом для перетворення масиву даних про свічки

у структурований список. Кожен елемент цього списку містить ключову інформацію, таку як час відкриття/закриття та ціна закриття свічки.

2. Отримання поточної ціни. Для оперативного прийняття рішень та моніторингу активних позицій необхідно знати поточну ринкову ціну активу. Сервіс надає можливість для заданого символу запитувати актуальну ціну через відповідну точку доступу API біржі, повертаючи її у числовому форматі.

3. Отримання інформації про біржу та торгові інструменти. Для коректного формування ордерів, зокрема для врахування правил лотності, система запитує загальну інформацію про біржу та специфікації торгових інструментів. Ці дані використовуються іншими частинами системи, відповідальними за розміщення ордерів, для визначення, наприклад, мінімального кроку обсягу та інших торгових обмежень.

4.2.2 Виконання торгових операцій

Сервіс, що відповідає за управління ордерами, інкапсулює всю логіку, пов'язану з їх розміщенням та управлінням.

1. Розміщення ордера на купівлю. Відповідний функціонал приймає на вхід дані активної торгової стратегії. На основі суми, виділеної на угоду згідно зі стратегією, та поточної ринкової ціни (отриманої від сервісу ринкових даних), розраховується бажана кількість активу. Перед відправкою ордера на біржу, ця кількість коригується відповідно до правил лотності для даного торгового інструменту за допомогою спеціальної процедури. Далі, використовуючи відповідні компоненти офіційної бібліотеки для взаємодії з біржею, формується та відправляється ринковий ордер на купівлю. У разі успішного розміщення, інформація про ордер, включаючи його ідентифікатор, ціну входу та розраховані рівні для обмеження збитків і фіксації прибутку, зберігається у системному списку активних ордерів.

2. Коригування обсягу ордера. Ця допоміжна функція є важливою для відповідності вимогам біржі. Вона отримує інформацію про мінімальну/максимальну кількість та крок зміни обсягу для конкретного символу від сервісу ринкових даних, який надає загальну інформацію про біржу. Потім вона коригує розраховану кількість активу так, щоб вона відповідала цим правилам, зазвичай округлюючи її вниз до найближчого допустимого значення. Це запобігає помилкам при розміщенні ордерів через невідповідність обсягу правилам біржі.

3. Розміщення ордера на продаж (закриття позиції). Функціонал для продажу використовується для закриття активної позиції, як правило, при спрацюванні рівнів обмеження збитків або фіксації прибутку. Він приймає дані ордера, який потрібно закрити, та причину закриття. Аналогічно до ордера на купівлю, формується ринковий ордер на продаж відповідної кількості активу. Після успішного розміщення ордера на продаж, початковий ордер на купівлю оновлює свій статус, вказуючи на причину закриття (наприклад, спрацювання фіксації прибутку або обмеження збитків), та видаляється зі списку активних ордерів.

4. Моніторинг активних ордерів. Ця процедура періодично викликається основним модулем торгової логіки. Вона проходить по списку всіх активних ордерів. Для кожного ордера запитується поточна ринкова ціна через сервіс ринкових даних. Потім ця ціна порівнюється з попередньо встановленими рівнями фіксації прибутку та обмеження збитків. Якщо ціна досягає одного з цих рівнів, ініціюється закриття позиції шляхом використання функціоналу для розміщення ордера на продаж.

4.3 Модуль технічного аналізу

Модуль технічного аналізу є центральним обчислювальним компонентом торгового бота. Його головне завдання – перетворювати необроблені ринкові дані, отримані від біржі, на значущі показники, відомі як технічні індикатори. Ці індикатори слугують основою для прийняття торгових рішень основним модулем

логіки, оскільки вони допомагають ідентифікувати потенційні тренди, точки входу та виходу з ринку, а також оцінювати поточну ринкову ситуацію (наприклад, стан перекупленості або перепроданості активу).

В архітектурі системи цей модуль інкапсулює всю логіку розрахунку індикаторів. Він отримує на вхід часові ряди цін (Klines) та параметри з активної торгової стратегії, необхідні для налаштування кожного індикатора (наприклад, періоди розрахунку, типи згладжування). Результатом роботи модуля є набір обчислених значень індикаторів, які передаються далі для аналізу в рамках торгової стратегії. Такий підхід забезпечує модульність та можливість легкого додавання нових індикаторів або модифікації існуючих у майбутньому.

4.3.1 Середні ковзні (Moving Averages)

Середні ковзні є одним з найпростіших та найпоширеніших інструментів технічного аналізу. Їх основне призначення – згладжувати цінові коливання та допомагати ідентифікувати напрямок основного тренду на ринку. Шляхом усереднення ціни за певний період, ковзні середні фільтрують короткостроковий "шум" і дозволяють побачити більш чітку картину руху ціни.

У розробленій системі реалізовано декілька типів ковзних середніх, що дозволяє гнучко використовувати їх у складі інших, більш складних індикаторів, таких як MACD. Обчислення ковзних середніх базується на цінах закриття (close price) відповідних свічок (Klines) за обраний період.

1. Проста ковзна середня (Simple Moving Average – SMA): розраховується як середнє арифметичне цін закриття за певну кількість останніх періодів (свічок) [2]:

$$SMA = \frac{\sum_{i=1}^{N} Close_i}{N},$$

де $Close_i$ – ціна закриття i-го періоду,

N – кількість періодів для розрахунку.

2. Експоненційна ковзна середня (Exponential Moving Average – EMA): надає більшу вагу останнім ціновим даним, що робить її більш чутливою до поточних змін ціни порівняно з SMA. Розраховується за формулою [2]:

$$EMA_i = \left(Close_i * \frac{2}{(N+1)}\right) + EMA_{i-1}\left(1 - \frac{2}{(N+1)}\right),$$

де $Close_i$ – ціна закриття i-го періоду,

N – кількість періодів для розрахунку.

3. Згладжена ковзна середня (Smoothed Moving Average – SMMA): це тип ковзної середньої, який також надає вагу попереднім значенням, але робить це таким чином, щоб лінія була ще більш згладженою, ніж EMA. Формула розрахунку SMMA може відрізнятися, але одна з поширених виглядає так [2]:

$$SMMA_i = \frac{\left(SMMA_{i-1}(N-1)\right) + Close_i}{N},$$

де $Close_i$ – ціна закриття i-го періоду,

N – кількість періодів для розрахунку.

Усі зазначені середні ковзні реалізовані за єдиним стандартом. Це дозволяє компоненту системи, відповідальному за розрахунок індикатора MACD, уніфіковано та динамічно використовувати будь-який з цих типів ковзних середніх. Вибір конкретного типу ковзної середньої (для короткострокового та довгострокового періодів) визначається налаштуваннями, вказаними у конфігурації торгової стратегії. Таким чином, реалізовані ковзні середні є важливим підготовчим компонентом для розрахунку основного трендового індикатора MACD, який використовується в системі для формування торгових сигналів [2].

4.3.2 Індикатор MACD (Moving Average Convergence Divergence)

Індикатор MACD (Moving Average Convergence Divergence – сходження/розходження ковзних середніх) є одним із ключових інструментів

технічного аналізу, який використовується в розробленій системі для ідентифікації потенційних сигналів на купівлю. Він належить до класу трендових осциляторів і допомагає оцінити силу та напрямок поточного ринкового тренду, а також визначити можливі моменти його зміни [2].

Розрахунок індикатора MACD інкапсульований у спеціалізованому компоненті системи. Цей компонент отримує на вхід історичні дані цін (у вигляді списку відповідних структур даних), а також параметри, визначені у конфігурації поточної торгової стратегії: тип та період для короткострокової ковзної середньої, а також тип та період для довгострокової ковзної середньої.

1. Обчислення ковзних середніх: на першому етапі відповідальний компонент обчислює значення двох ковзних середніх (наприклад, для 12-періодної та 26-періодної, якщо такі періоди задані у стратегії). Для цього він використовує відповідні реалізації стандартного механізму розрахунку ковзних середніх (наприклад, експоненційну ковзну середню). Це дозволяє гнучко обирати тип ковзної середньої (просту, експоненційну, згладжену) для кожної з компонент MACD через конфігураційний файл.

2. Розрахунок лінії MACD: після отримання значень обох ковзних середніх для останньої доступної свічки, сервіс обчислює значення самої лінії MACD за формулою [2]:

$$MACD = MA_{\text{коротка}} - MA_{\text{довга}},$$

де MA – ковзна середня обраного типу.

Отримане числове значення лінії MACD передається далі до основного модуля торгової логіки для аналізу та прийняття рішень. У поточній реалізації стратегії, як було зазначено, ключовим сигналом від MACD є його перетин нульової лінії знизу вгору, що інтерпретується як одна з умов для відкриття позиції на купівлю.

4.3.3 Індикатор RSI (Relative Strength Index)

Індекс відносної сили (Relative Strength Index - RSI) є ще одним важливим інструментом технічного аналізу, що належить до класу імпульсних осциляторів. Його основне завдання – вимірювати швидкість та амплітуду цінових змін активу за певний період. RSI допомагає трейдерам оцінити, чи не є актив "перекупленим" або "перепроданим", що може вказувати на ймовірність корекції або розвороту поточної цінової тенденції [2].

Розрахунок індикатора RSI реалізований у відповідному сервісному класі (RSIService). Цей сервіс приймає на вхід історичні дані цін (список об'єктів Kline) та параметр періоду, за який буде проводитися розрахунок (наприклад, класичний період 14).

1. Визначення цінових змін: Для кожної свічки в межах обраного періоду (починаючи з найстарішої до найновішої в цьому періоді) система обчислює різницю між ціною закриття поточної свічки та ціною закриття попередньої свічки.

2. Класифікація змін: Якщо цінова зміна позитивна (ціна зросла), вона класифікується як "приріст" (gain). Якщо цінова зміна негативна (ціна впала), її абсолютне значення класифікується як "втрата" (loss). Зміни, рівні нулю, ігноруються або не впливають на розрахунок середніх приростів/втрат [2].

3. Розрахунок середніх приростів та втрат: Далі обчислюється середня величина всіх приростів та середня величина всіх втрат за вказаний період.

4. Обчислення відносної сили (RS): Наступним кроком є розрахунок відносної сили, яка є відношенням середнього приросту до середньої втрати [2]:

$$RS = \frac{\text{СереднійПриріст}}{\text{СередняВтрата}}.$$

Якщо середня втрата дорівнює нулю (що трапляється, якщо ціна лише зростала протягом усього періоду), RS може вважатися нескінченно великим, що призведе до RSI = 100.

5. Обчислення RSI: Фінальне значення RSI розраховується за стандартною формулою нормалізації, яка приводить його до діапазону 0-100:

$$RSI = 100 - \frac{100}{(1 + RS)},$$

де RS – значення відносної сили.

Отримане числове значення RSI передається до основного модуля торгової логіки. В рамках реалізованої стратегії, умовою для купівлі є зниження RSI нижче певного встановленого рівня (acceptableLevel), що інтерпретується як сигнал про потенційну недооціненість активу або сприятливий момент для входу в довгу позицію в очікуванні зростання ціни.

4.4 Основний модуль торгової логіки

Основний модуль торгової логіки є центральним координаційним компонентом усієї системи, відповідальним за оркестрацію процесу автоматизованої торгівлі. В архітектурі системи ця роль покладена на клас Trader. Його головне завдання – періодично аналізувати ринкову ситуацію відповідно до завантажених торгових стратегій, приймати рішення про вхід або вихід з позицій та взаємодіяти з іншими модулями для отримання даних, розрахунку індикаторів та виконання торгових операцій.

Робота основного модуля торгової логіки ініціюється та підтримується планувальником завдань, реалізованим за допомогою механізмів відповідного програмного каркасу. Основна торгова функція в цьому керуючому компоненті запускається автоматично з фіксованою періодичністю (наприклад, кожні 6 секунд) за допомогою спеціального механізму планування завдань. Цей інтервал визначає частоту, з якою бот буде перевіряти ринкові умови та приймати торгові рішення.

Під час кожного запуску, ця основна торгова функція виконує наступні ключові кроки для кожної активної торгової стратегії, отриманої від сервісу конфігурації стратегій.

1. Отримання ринкових даних. Для поточної оброблюваної стратегії, головний модуль запитує необхідні історичні дані цін (у вигляді свічок) у сервісу

ринкових даних. Кількість запитуваних свічок (в поточній реалізації – 100) та часовий інтервал визначаються параметрами стратегії.

2. Технічний аналіз. Отримані дані свічок разом з параметрами індикаторів з поточної стратегії передаються до модуля технічного аналізу. Цей модуль розраховує поточні значення індикаторів MACD та RSI, які потім повертаються до центрального керуючого компонента.

3. Порівняння з попередніми значеннями. Для реалізації логіки, що базується на динаміці зміни індикаторів (наприклад, перетин ліній), центральний керуючий компонент зберігає результати аналізу попереднього циклу у спеціальній структурі даних. Перед тим, як генерувати сигнал, поточні значення індикаторів порівнюються зі збереженими попередніми значеннями. Якщо для поточної стратегії попередніх даних ще немає (перший запуск для цієї стратегії), поточні дані просто зберігаються, і рішення про вхід не приймається, очікуючи наступного циклу для формування динаміки.

4. Генерація торгових сигналів. На основі поточних та попередніх значень індикаторів, а також порогових рівнів, заданих у конфігурації стратегії, формується логічний сигнал на купівлю. У поточній реалізації сигнал на купівлю генерується, якщо виконуються наступні умови одночасно:

- для кожного MACD-індикатора, визначеного у стратегії: попереднє значення MACD було менше нуля, а поточне стало більше нуля (тобто відбувся перетин нульової лінії знизу вгору);
- для RSI-індикатора: поточне значення RSI менше заданого у стратегії порогового рівня (що може інтерпретуватися як стан перепроданості або просто низький рівень імпульсу).

5. Прийняття рішення про відкриття позиції. Якщо сигнал на купівлю є істинним, система додатково перевіряє, чи немає вже активних ордерів для даного торгового символу за цією ж стратегією (використовуючи відповідну функцію сервісу управління ордерами). Якщо активних ордерів немає, центральний керуючий компонент ініціює відкриття нової позиції на купівлю, викликаючи

відповідну функцію у сервісі управління ордерами та передаючи йому поточну торгову стратегію.

6. Поточні розраховані значення індикаторів зберігаються у відповідній структурі даних для використання у наступному торговому циклі.

Незалежно від генерації нових сигналів на вхід, після обробки всіх стратегій у кожному торговому циклі, центральний керуючий компонент ініціює перевірку всіх активних ордерів на предмет досягнення рівнів обмеження збитків або фіксації прибутку. Це здійснюється шляхом виклику відповідної функції у сервісі управління ордерами. Якщо умови для закриття позиції виконані, сервіс управління ордерами самостійно розміщує відповідний ордер на продаж.

Протягом усього циклу роботи центрального керуючого компонента та взаємодіючих з ним сервісів відбувається детальне логування ключових подій: початок та завершення торгового циклу, обробка кожної стратегії, отримані значення індикаторів, згенеровані сигнали, інформація про розміщення ордерів та їх закриття. Це забезпечує можливість моніторингу роботи бота та аналізу його ефективності.

Таким чином, основний модуль торгової логіки виступає як центральний процесор системи, який на основі періодичного аналізу ринкових даних та заздалегідь визначених правил приймає автоматизовані торгові рішення, прагнучи реалізувати закладені в стратегії торгові можливості.

5 РОБОТА ТОРГОВОГО БОТА

Цей розділ присвячений практичним аспектам роботи розробленого торгового бота. Після детального розгляду теоретичних основ, проектування архітектури, вибору засобів розробки та опису реалізації програмних модулів, важливо продемонструвати, як система функціонує в реальних або тестових умовах. Буде розглянуто процес налаштування та розгортання торгового бота, а також наведено приклад його роботи з демонстрацією ключових етапів: від аналізу ринку та генерації торгових сигналів до розміщення ордерів та моніторингу активних позицій. Це дозволить оцінити практичну реалізацію поставлених завдань та функціональність розробленої системи.

5.1 Налаштування та розгортання

Ефективна та стабільна робота торгового бота вимагає ретельного налаштування його параметрів та належної підготовки середовища для розгортання. Ці кроки є критично важливими для забезпечення коректної взаємодії з криптовалютною біржею та виконання торгових стратегій відповідно до заданих умов.

Першочерговим етапом налаштування є конфігурація доступу до API криптовалютної біржі. Для цього користувач повинен згенерувати API-ключ та відповідний йому секретний ключ в особистому кабінеті на сайті біржі Binance. Важливо при генерації ключів надати їм необхідні дозволи, зокрема, право на здійснення спотової торгівлі, якщо планується реальна торгівля, або відповідні дозволи для тестового середовища. Отримані ключі вказуються у конфігураційному файлі додатку (application.yml) у полях binance.api.key та binance.api.secret. Наголошується на необхідності забезпечення максимального рівня безпеки при зберіганні цих ключів: вони не повинні зберігатися у публічних репозиторіях у відкритому вигляді, а при розгортанні на сервері мають бути

захищені відповідними механізмами (наприклад, через змінні середовища, системи управління секретами або зашифровані конфігураційні файли).

Наступним важливим кроком є налаштування самих торгових стратегій. Як було описано в розділі 4.1, параметри стратегій визначаються у зовнішньому JSON-файлі. Користувач може гнучко модифікувати цей файл, додаючи нові стратегії або змінюючи існуючі. Для кожної стратегії необхідно вказати унікальну назву, торговий символ (наприклад, "DOGEUSDT"), часовий інтервал для аналізу ринкових даних (наприклад, "1h" для годинних свічок), суму, що виділяється на одну торгову операцію (в одиницях котируваної валюти, наприклад, USDT), а також відсоткові рівні для автоматичного встановлення ордерів Stop-Loss та Take-Profit. Крім того, для кожної стратегії задаються специфічні параметри для використовуваних технічних індикаторів, такі як періоди розрахунку та типи ковзних середніх для MACD, період та пороговий рівень для RSI.

Розгортання торгового бота для стабільної та цілодобової роботи передбачає використання серверної інфраструктури. Згідно з індивідуальним завданням, випробування системи проводилося в хмарному сервісі AWS (Amazon Web Services) . Це популярний вибір для подібних завдань завдяки високій надійності, масштабованості та широкому набору інструментів. Для розгортання на AWS типовим сценарієм є використання сервісу Amazon EC2 (Elastic Compute Cloud), який дозволяє створювати віртуальні сервери (інстанси) з необхідною операційною системою (наприклад, один з дистрибутивів Linux) та конфігурацією ресурсів. На такому інстансі встановлюється JRE, після чого на нього завантажується зібраний JAR-файл торгового бота. Запуск додатка може бути організований як фоновий процес або як системний сервіс (наприклад, через systemd в Linux) для забезпечення його автоматичного перезапуску у випадку збоїв або після перезавантаження сервера. Важливо також налаштувати мережеві параметри (Security Groups в AWS) для забезпечення доступу бота до API біржі та, за необхідності, для віддаленого управління сервером. Для зберігання лог-файлів та інших артефактів може використовуватися сервіс Amazon S3, а для моніторингу стану системи – AWS CloudWatch.

Для автоматизації процесів збірки, тестування та доставки оновлень торгового бота можуть бути використані системи безперервної інтеграції та безперервного розгортання (CI/CD), такі як GitHub Actions. Це дозволяє налаштувати робочий процес, який автоматично компілює код, виконує тести та, у разі успіху, розгортає нову версію бота на цільовий сервер. Хоча конкретна реалізація CI/CD виходить за межі основного функціоналу бота, використання таких інструментів є рекомендованою практикою для забезпечення якості та оперативності оновлень програмного забезпечення. При налаштуванні CI/CD особливу увагу слід приділити безпечній передачі конфігураційних файлів та секретів (API-ключів) на сервер розгортання, використовуючи для цього вбудовані механізми секретів CI/CD системи та змінні середовища на сервері.

Таким чином, процес налаштування та розгортання торгового бота включає конфігурацію доступу до біржі, визначення торгових стратегій, підготовку середовища виконання та вибір надійної платформи для довготривалої роботи, якою в даному випадку є хмарна інфраструктура AWS.

5.2 Приклад роботи торгового бота

Для демонстрації практичного функціонування розробленого торгового бота розглянемо приклад його роботи на основі логів, згенерованих системою під час одного з торгових циклів. Ці логи ілюструють ключові етапи: від аналізу ринкових даних за визначеними стратегіями до прийняття рішень та взаємодії з біржею. У даному прикладі бот налаштований на роботу з тестовим середовищем біржі Binance (testnet) та обробляє дві торгові стратегії: одну для пари DOGE/USDT та іншу для BTC/USDT.

Початковий етап роботи бота в рамках одного торгового циклу відображено на рисунку 5.1. Система інформує про початок нового циклу ("Starting trading cycle") та кількість стратегій, що будуть оброблені ("Processing 2 trading strategies"). Далі відбувається послідовний аналіз кожної стратегії.

```
Starting trading cycle
Processing 2 trading strategies
Processing strategy: DOGE/USDT Strategy for symbol: DOGEUSDT
GET https://data-api.binance.vision/api/v3/klines?symbol=DOGEUSDT&limit=100&interval=1h
[DOGEUSDT] Indicator: macd(ema, 12, 26), previous: -0.001869757947511497, current: -0.0018705556683114972, signal: false
[DOGEUSDT] Indicator: macd(ema, 26, 36), previous: 4.540614318607678E-4, current: 4.5386123166076784E-4, signal: false
[DOGEUSDT] Indicator: rsi(14), current: 33.86773547094185, signal: true
[DOGEUSDT] Buy signal: false
[DOGEUSDT] No buy signal detected or there are active orders
Processing strategy: BTC/USDT Strategy for symbol: BTCUSDT
GET https://data-api.binance.vision/api/v3/klines?symbol=BTCUSDT&limit=100&interval=4h
[BTCUSDT] Indicator: macd(ema, 12, 26), previous: 699.8599517499346, current: 699.8599517499346, signal: false
[BTCUSDT] Indicator: rsi(14), current: 46.16733394700941, signal: false
[BTCUSDT] Buy signal: false
[BTCUSDT] No buy signal detected or there are active orders
Trading cycle completed
```

Рисунок 5.1 – Логи аналізу ринку та генерації сигналів

При обробці стратегії для DOGE/USDT бот спочатку виконує GET-запит до API Binance для отримання історичних даних – останніх 100 годинних свічок (Klines) для пари DOGEUSDT. На основі цих даних розраховуються значення технічних індикаторів, визначених у стратегії. У даному прикладі це два індикатори MACD з різними параметрами та один індикатор RSI. Логи показують попередні та поточні значення для кожного MACD, а також згенерований сигнал, що вказує на відсутність умови для купівлі (перетину нульової лінії знизу вгору). Для RSI(14) поточне значення склало 33.8677, і оскільки це значення, ймовірно, менше заздалегідь встановленого у стратегії acceptableLevel, система згенерувала сигнал "true" для цього індикатора. Однак, оскільки для відкриття позиції необхідне виконання умов по всіх індикаторах, загальний сигнал на купівлю ("Buy signal") для DOGEUSDT залишається "false". Відповідно, система фіксує, що сигнал на купівлю не виявлено або для даного символу вже існують активні ордери. Аналогічний процес відбувається і для стратегії BTC/USDT: запитуються дані (4-годинні свічки для BTCUSDT), розраховуються індикатори MACD та RSI, і, оскільки умови для купівлі не виконуються, загальний сигнал також "false". Після обробки обох стратегій на цьому етапі, перша частина торгового циклу, пов'язана з аналізом ринку, завершується, про що свідчить повідомлення "Trading cycle

completed". У цьому конкретному циклі жодного сигналу на відкриття нових позицій згенеровано не було.

Розглянемо інший сценарій, коли в одному з наступних торгових циклів умови на ринку змінилися таким чином, що для стратегії DOGE/USDT було сформовано сигнал на купівлю. Лог дій системи в такій ситуації представлено на рисунку 5.2.

```
Buy signal detected, placing order
GET https://data-api.binance.vision/api/v3/ticker/price?symbol=DOGEUSDT
GET https://data-api.binance.vision/api/v3/exchangeInfo
Adjusted quantity for DOGEUSDT to 4367.0 (stepSize: 1.0, precision: 0)
POST https://testnet.binance.vision/api/v3/order?symbol=DOGEUSDT&side=BUY&quantity=4367.00000000&signature=7d7db99e4a503fa3
Order placed: {"symbol":"DOGEUSDT","orderId":791818,"orderListId":-1,"clientOrderId":"oHPFJlnB1uBZ4WZQ4HNHCt","transactTime
Buy order created: Order(id=791818, symbol=DOGEUSDT, side=BUY, type=MARKET, quantity=4367.0, price=0.22897, status=FILLED,
Stop Loss: 0.2175215, Take Profit: 0.251867
```

Рисунок 5.2 – Логи розміщення ордера на купівлю

Після виявлення сигналу на купівлю система виконує підготовчі кроки перед розміщенням ордера. Спочатку вона надсилає GET-запити до API Binance для отримання актуальної ринкової ціни для торгової пари DOGEUSDT та для отримання загальної інформації про торговий інструмент. Остання інформація необхідна для врахування правил лотності біржі. На основі суми ордера, заданої у стратегії, та поточної ринкової ціни DOGE, розраховується первинна кількість токенів для купівлі. Далі ця кількість коригується відповідно до правил лотності для DOGEUSDT, наприклад, з урахуванням кроку зміни обсягу (stepSize: 1.0) та точності (precision: 0). У наведеному прикладі скоригована кількість склала 4367.0 DOGE. Після цього система формує та відправляє POST-запит до тестового API Binance для розміщення ринкового (MARKET) ордера на купівлю вказаної кількості DOGEUSDT. Успішне розміщення ордера підтверджується відповіддю від біржі, яка також логується і містить важливу інформацію, таку як ідентифікатор ордера. На основі цих даних та параметрів стратегії, система створює внутрішній об'єкт Order, де фіксується вся інформація про угоду: ID ордера, символ, сторона

(BUY), тип (MARKET), кількість, ціна входу, статус (FILLED), а також автоматично розраховані цінові рівні для Stop Loss та Take Profit.

Після успішного відкриття позиції, торговий бот у кожному наступному торговому циклі здійснює моніторинг стану всіх активних ордерів. Приклад такого моніторингу показаний на рисунку 5.3.

```
Checking 1 active order(s) for take profit/stop loss
GET https://data-api.binance.vision/api/v3/ticker/price?symbol=DOGEUSDT
Order 791818 still active. Current P&L: 0.017469537493988073%, Price: 0.22901
```

Рисунок 5.3 – Логи моніторингу активного ордера

Система повідомляє про перевірку активних ордерів ("Checking 1 active order(s) for take profit/stop loss"). Для кожного активного ордера, наприклад, для раніше відкритого ордера 791818 по DOGEUSDT, система запитує поточну ринкову ціну через GET-запит до API Binance. Після отримання ціни, система логує, що ордер все ще активний, і може відображати додаткову інформацію, таку як поточний нереалізований прибуток/збиток (P&L) та актуальну ціну активу. Якщо поточна ціна не досягла попередньо встановлених рівнів Stop Loss або Take Profit, ордер залишається відкритим, і система переходить до наступних завдань. У випадку, якби ціна досягла одного з цих лімітних рівнів, система автоматично ініціювала б розміщення ордера на продаж для закриття позиції, що також було б детально зафіксовано у логах.

Наведений приклад наочно демонструє повний цикл роботи торгового бота: від початкового аналізу ринку за допомогою визначених технічних індикаторів та правил торгової стратегії, через генерацію відповідних торгових сигналів, взаємодію з програмним інтерфейсом біржі для отримання необхідних даних та безпосереднього розміщення торгових ордерів, до подальшого моніторингу відкритих позицій та автоматичного управління ризиками за допомогою ордерів Stop-Loss та Take-Profit.

ВИСНОВКИ

Під час проходження практики було вирішено завдання розробки програмної системи – торгового бота для автоматизації операцій на криптовалютних біржах. Були отримані наступні основні результати:

1. На основі проведеного аналізу методів торгівлі криптоактивами, включаючи технічний аналіз (ковзні середні, MACD, RSI), стилі торгівлі (HODLing, Swing Trading, Day Trading, скальпінг) та загальні торгові стратегії (торгівля за трендом, контртрендова торгівля, арбітраж, маркет-мейкінг, торгівля на новинах), було обґрунтовано використання комбінованого підходу на основі технічних індикаторів MACD та RSI для формування торгових сигналів у розроблюваному боті. Зокрема, для генерації сигналу на купівлю було обрано стратегію, що враховує перетин індикатором MACD нульової лінії знизу вгору одночасно зі знаходженням індикатора RSI у зоні перепроданості (або нижче визначеного порогового рівня).

2. В результаті аналізу існуючих програмних рішень для автоматизації торгівлі (таких як 3Commas, CryptoHopper, Pionex, Bitsgap) та рішень з відкритим вихідним кодом, було обґрунтовано доцільність розробки власної системи. Для реалізації програмного забезпечення торгового бота було обрано мову програмування Java завдяки її надійності, багатопотоковості та великій екосистемі. В якості основного фреймворку було використано Spring Boot, що забезпечило швидку розробку, управління залежностями та реалізацію планувальника завдань. Для взаємодії з API криптовалютної біржі Binance було використано офіційну бібліотеку binance-connector-java. В якості платформи для розгортання та тестування системи було обрано хмарний сервіс AWS, що відповідає сучасним вимогам до надійності та масштабованості.

3. Розроблено програмну систему "Торговий бот для криптовалютних бірж". Система реалізує ключовий функціонал, необхідний для автоматизованої торгівлі: підключення до API біржі Binance (включаючи тестове середовище), отримання ринкових даних (історичні свічки, поточні ціни), розрахунок технічних

індикаторів (різні типи ковзних середніх, MACD, RSI), гнучке налаштування торгових стратегій через зовнішній JSON-файл, автоматичне розміщення ринкових ордерів на купівлю, управління ризиками за допомогою ордерів Stop-Loss та Take-Profit, а також детальне логування всіх операцій. Архітектура системи є модульною, що забезпечує можливість її подальшого розширення та модифікації.

4. На основі проведеного тестування програмного забезпечення, включаючи перевірку інтеграції з тестовим середовищем біржі Binance (testnet) та функціональне тестування реалізованих торгових стратегій на прикладі обробки даних для пар DOGE/USDT та BTC/USDT, було доведено коректність роботи розробленого торгового бота. Система продемонструвала здатність отримувати ринкові дані, коректно розраховувати технічні індикатори, генерувати торгові сигнали відповідно до заданих умов стратегії, розміщувати ордери на купівлю та супроводжувати активні позиції з використанням механізмів Stop-Loss та Take-Profit, що підтверджується логами роботи системи.

5. Розроблений торговий бот є функціональним прототипом, що реалізує основні завдання автоматизованої торгівлі. Перспективи подальшого вдосконалення системи включають: розширення набору підтримуваних технічних індикаторів та торгових стратегій; реалізацію можливості підключення до декількох криптовалютних бірж; додавання функціоналу для бектестінгу стратегій на історичних даних; розробку користувацького веб-інтерфейсу для зручного управління та моніторингу; інтеграцію з системами сповіщень (наприклад, Telegram); а також оптимізацію продуктивності та підвищення рівня відмовостійкості системи для роботи в умовах реальних ринкових навантажень.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Cryptoassets: The innovative investor's guide to bitcoin and beyond / ed. by T. J. author. McGraw-Hill Education, 2018. 325 p.

2. Murphy J. J. Technical Analysis of the Financial Markets. Pengiun Books.

3. Aries A. Blockchain Fundamentals and Bitcoin Basics. Independently Published, 2021.

4. Danial K., Staff D. Cryptocurrency Investing for Dummies. Wiley & Sons, Incorporated, John, 2019. 352 p.

5. Davey K. J. Building Winning Algorithmic Trading Systems: A Trader's Journey from Data Mining to Monte Carlo Simulation to Live Trading. Wiley & Sons, Incorporated, John, 2014. 288 p.

6. Martin R. Clean Architecture: A Craftsman's Guide to Software Structure and Design. Pearson Education, Limited, 2017.

7. Rozanski N., Woods E. Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives. Addison-Wesley Professional, 2005. 576 p.

8. Software Architecture: the Hard Parts: Modern Tradeoff Analysis for Distributed Architectures / M. Richards et al. O'Reilly Media, Incorporated, 2021. 450 p.

9. Clean Code: A Handbook of Agile Software Craftsmanship. Prentice Hall, 2008. 431 p.

10. Effective Java. Pearson Education, Limited, 2017. 414 p.

11. Horstmann C. Core Java Volume I--Fundamentals. Pearson Education, Limited, 2021.

12. Eckel B. Thinking in Java. Upper Saddle River, N.J : Prentice Hall, 1998. 1098 p.

13. Java Concurrency in Practice. Addison-Wesley Professional, 2006. 384 p.

14. Naughton P., Schildt H. Java: The Complete Reference (Complete Reference Series). Mcgraw-Hill Osborne Media, 1996. 886 p.

15. Spring Boot: up and Running: Building Cloud Native Java and Kotlin Applications. O'Reilly Media, Incorporated, 2021. 325 p.

16. Buddha J. P., Beesetty R. The Definitive Guide to AWS Application Integration: With Amazon SQS, SNS, SWF and Step Functions. Apress, 2019. 367 p.

17. Raje G. Security and Microservice Architecture on AWS: Architecting and Implementing a Secured, Scalable Solution. O'Reilly Media, Incorporated, 2021. 350 p.

18. Sayed I. AWS for Solutions Architects: The Definitive Guide to AWS Solutions Architecture for Migrating to, Building, Scaling, and Succeeding in the Cloud. de Gruyter GmbH, Walter, 2023.

19. Complete Guide to DevOps with AWS: Deploy, Build, and Scale Services with AWS Tools and Techniques. Apress L. P., 2023.

ДОДАТОК А

АПРОБАЦІЯ

УКР.НТУУ”КПІ ім. Ігоря Сікорського”

Аркушів 2

Київ 2025

Білий Д.Ю., Шушура О.М. Торговий бот для криптовалютних бірж. Матеріали XII Всеукраїнської науково-практичної конференції здобувачів вищої освіти та молодих вчених з автоматичного управління присвяченої Дню ракетно-космічної галузі України: збірник матеріалів конференції (10-12 квітня 2025р., м. Херсон, м. Хмельницький). Херсон: Книжкове видавництво ФОП Вишемирський В.С., 2025. - 145 с.

УДК 004.42

Д.Ю. Білий, О.М. Шушура
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»
danil22344bel@gmail.com

ТОРГОВИЙ БОТ ДЛЯ КРИПТОВАЛЮТНИХ БІРЖ

У теперішній час спостерігається стрімке зростання ринку криптовалют та значне збільшення обсягів торгівлі на криптовалютних біржах. Автоматизація торгівлі за допомогою спеціалізованих ботів стає все більш затребуваною, оскільки дозволяє підвищити ефективність торгових операцій, зменшити вплив емоційного фактору при прийнятті рішень та забезпечити цілодобову роботу без перерв. Однак існуючі рішення часто обмежені підтримкою лише однієї біржі, мають недостатню гнучкість налаштування торгових стратегій або потребують значних технічних знань для їх впровадження.

Ринок криптовалют характеризується високою волатильністю та працює цілодобово, що ускладнює ручне управління торговими операціями. Саме тому розробка універсального, надійного та масштабованого торгового бота з можливістю підключення до різних бірж та налаштування різноманітних торгових стратегій є актуальною задачею.

Аналіз існуючих рішень показує, що більшість торгових ботів для криптовалютних бірж мають обмежену функціональність і не дозволяють повною мірою використовувати потенціал алгоритмічної торгівлі. Згідно з дослідженнями, автоматизовані системи торгівлі з правильно налаштованими стратегіями можуть демонструвати кращі результати порівняно з ручним трейдингом, особливо в періоди високої волатильності ринку [1, 2].

Сучасні підходи до розробки торгових ботів включають використання методів технічного аналізу, машинного навчання для прогнозування цін, а також інтеграцію з хмарними сервісами для забезпечення високої доступності та масштабованості [3, 4]. Однак недостатньо уваги приділяється створенню гнучкої архітектури, яка б дозволяла швидко адаптуватися до змін на ринку та підключати нові біржі й стратегії без необхідності значної переробки коду.

Метою роботи є розробка програмної системи у вигляді торгового бота для криптовалютних бірж, який дозволить автоматизувати процес торгівлі, мінімізувати вплив людського фактора та підвищити ефективність прийняття рішень. Система має забезпечувати підтримку кількох біржових платформ, надавати можливість налаштування та тестування різних торгових стратегій, а також гарантувати гнучкість та масштабованість для подальшого розвитку. Реалізація цього проекту передбачає вирішення таких завдань:

- визначити функціональні вимоги до системи торгового бота на основі аналізу потреб та існуючих рішень;
- обрати оптимальну архітектуру системи, яка забезпечить гнучкість, масштабованість та надійність;
- розробити модульну структуру системи з можливістю легкого підключення нових бірж та впровадження різних торгових стратегій;
- створити механізми отримання та аналізу ринкових даних для використання в торгових стратегіях;
- реалізувати базові торгові стратегії та протестувати їх ефективність в тестовому середовищі;
- розробити механізми розгортання системи в хмарній інфраструктурі з забезпеченням високої доступності;
- провести тестування системи в реальних умовах на різних біржах та оцінити її ефективність.

На основі аналізу потреб та існуючих рішень визначено ключові функціональні вимоги до системи: підтримка кількох бірж, можливість налаштування різних торгових стратегій, цілодобова робота, високий рівень безпеки, можливість моніторингу та аналізу результатів торгівлі.

Для реалізації системи обрано мікросервісну архітектуру. Вибір саме цієї архітектури обумовлений наступними перевагами: гнучкість та масштабованість, незалежність розробки та оновлення, технологічна різноманітність, висока надійність, легке підключення нових бірж та стратегій.

Архітектура системи (рис. 1) забезпечує гнучкість, масштабованість та надійність при високих навантаженнях. Основними компонентами є: Trader – керує процесом торгівлі; Strategy – набір взаємозамінних модулів, що реалізують різні торгові стратегії (наприклад, Moving Average, RSI, MACD тощо); TechnicalAnalyzer – модуль для проведення технічного аналізу ринкових даних; MarketService – сервіс для отримання ринкових даних з різних бірж; TradeService – сервіс для виконання торгових операцій; Client – абстракція для роботи з різними біржами (Spot1Client, Spot2Client, SpotNClient); Telegram Bot та REST API – інтерфейси взаємодії користувача з системою.

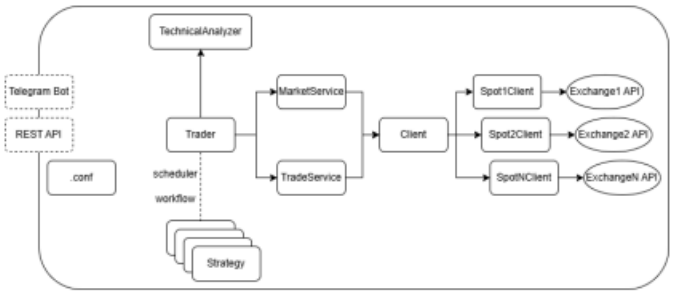

Рисунок 1 – Архітектура системи торгового бота

Система розгортається в хмарній інфраструктурі AWS з використанням сервісу Elastic Beanstalk, що забезпечує автоматичне масштабування та високу доступність. Компоненти системи запускаються в контейнерах Docker, що спрощує розгортання та оновлення. Для зберігання історичних даних та налаштувань використовується хмарна база даних.

Запропонована система торгового бота для криптовалютних бірж дозволяє автоматизувати процес торгівлі, зменшити вплив емоційного фактору при прийнятті рішень та забезпечити цілодобову роботу. Мікросервісна архітектура забезпечує високу надійність, масштабованість та гнучкість системи, дозволяючи швидко адаптуватися до змін на ринку та підключати нові біржі й стратегії.

Перспективи подальших досліджень включають інтеграцію методів машинного навчання для прогнозування цін, розробку більш складних торгових стратегій та впровадження механізмів адаптивного налаштування параметрів стратегій на основі аналізу їх ефективності.

ЛІТЕРАТУРА:

1. Hudson R., McGroarty F. Automated trading with performance weighted random forests and seasonality. Journal of Asset Management. 2023. Т. 24, № 3. С. 207–222.
2. Nakano M., Takahashi A., Takahashi S. Bitcoin technical trading with artificial neural network. Physica A: Statistical Mechanics and its Applications. 2018. Т. 510. С. 587–609.
3. Garcia-Martinez J. A., Vicedo P., Vicedo J. L. Application of technical, fundamental and sentiment analysis in automated algorithmic trading systems for cryptocurrencies. Mathematics. 2023. Т. 11, № 3. С. 572.
4. Hegazy S., Sayed H. H. Development of a cryptocurrency trading bot integrated with technical analysis. International Journal of Advanced Computer Science and Applications. 2022. Т. 13, № 6. С. 356-363.

ДОДАТОК Б

РЕАЛІЗАЦІЯ ЛОГІКИ ОБРАХУНКУ ТЕХНІЧНИХ ІНДИКАТОРІВ СЕРЕДНІХ КОВЗАЮЧИХ

УКР.НТУУ”КПІ ім. Ігоря Сікорського”

Аркушів 2

Київ 2025

Програмні засоби:

- мова програмування – Java;
- фреймворк – SpringBoot.

```
@Component
public class SimpleMovingAverage implements MovingAverage {

  @Override
  public MovingAverageType getType() {
    return MovingAverageType.SMA;
  }

  @Override
  public BigDecimal calculate(List<Kline> klines, int period) {
    BigDecimal sum = BigDecimal.ZERO;
    for (int i = klines.size() - period; i < klines.size(); i++) {
      sum = sum.add(klines.get(i).getClosePrice());
    }
    return sum.divide(BigDecimal.valueOf(period), 8,
RoundingMode.HALF_UP);
  }
}

@Component
public class SmoothedMovingAverage implements MovingAverage {

  @Override
  public MovingAverageType getType() {
    return MovingAverageType.SMMA;
  }

  @Override
  public BigDecimal calculate(List<Kline> klines, int period) {
    BigDecimal smma = BigDecimal.ZERO;
    for (int i = 0; i < period; i++) {
      smma = smma.add(klines.get(i).getClosePrice());
    }
    smma = smma.divide(BigDecimal.valueOf(period), 8,
RoundingMode.HALF_UP);
    for (int i = period; i < klines.size(); i++) {
      smma = smma.multiply(BigDecimal.valueOf(period - 1))
          .add(klines.get(i).getClosePrice())
          .divide(BigDecimal.valueOf(period), 8, RoundingMode.HALF_UP);
```

```
        }
        return smma;
      }

    }

    @Component
    public class ExponentialMovingAverage implements MovingAverage {

      public static final BigDecimal SMOOTHING_FACTOR =
BigDecimal.valueOf(2);

      @Override
      public MovingAverageType getType() {
        return MovingAverageType.EMA;
      }

      @Override
      public BigDecimal calculate(List<Kline> klines, int period) {
        BigDecimal ema = BigDecimal.ZERO;
        for (int i = 0; i < period; i++) {
          ema = ema.add(klines.get(i).getClosePrice());
        }
        ema = ema.divide(BigDecimal.valueOf(period), 8,
RoundingMode.HALF_UP);

        BigDecimal multiplier = SMOOTHING_FACTOR
            .divide(BigDecimal.valueOf(period).add(BigDecimal.ONE), 8,
RoundingMode.HALF_UP);

        for (int i = period; i < klines.size(); i++) {
          ema = klines.get(i).getClosePrice()
              .subtract(ema)
              .multiply(multiplier)
              .add(ema);
        }
        return ema;
      }
    }
```