ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Методика оптимізації показників швидкодії сервісу моніторингу криптовалютного ринку»

на здобуття освітнього ступеня магістра

зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів текстів інших авторів мають посилання на відповідне джерело

_____ Анатолій ЩЕГОЛЬ
(підпис)

Виконав: здобувач вищої освіти групи ПДМ-62

Анатолій ЩЕГОЛЬ

Керівник: Наталя ТРІНТІНА
к.т.н., доцент

Рецензент: _____

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Магістр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
«_____»_____2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Щеголю Анатолію Глібовичу

1. Тема кваліфікаційної роботи: «Методика оптимізації показників швидкодії сервісу моніторингу криптовалютного ринку»

керівник кваліфікаційної роботи Наталя ТРІНТІНА, к.т.н., доцент,

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «15» жовтня 2024 р. № 320.

2. Строк подання кваліфікаційної роботи «17» грудня 2024 р.

3. Вихідні дані до кваліфікаційної роботи: науково-технічна література, дані про існуючі криптовалютні ринки та доступні API, документація по FastAPI, Redis і бібліотеках Python.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Аналіз предметної галузі.
2. Аналіз існуючих інструментів та технологій.
3. Практична реалізація оптимізованої архітектури системи.
4. Оцінка ефективності впроваджених оптимізацій.

5. Перелік графічного матеріалу: *презентація*
    1. Аналіз існуючих підходів отримання даних.
    2. Запропоноване рішення.
    3. Порівняльний аналіз традиційних і обраних підходів.
    4. Переваги вибору Fast API.
    5. Схема модифікованого методу оптимізації показників криптовалютного ринку.
    6. Порівняльний аналіз швидкодії API.
    7. Структура проєкту.
    8. Екранні форми.

6. Дата видачі завдання «16» жовтня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз наявної науково-технічної літератури	15.10-30.10.24	
2	Вивчення технологій FastAPI та Redis	01.11-06.11.24	
3	Аналіз існуючих рішень для моніторингу криптовалют	07.11-10.11.24	
4	Проектування архітектури сервісу	11.11-17.11.24	
5	Розробка функціоналу збору та кешування даних	18.11-24.11.24	
6	Тестування продуктивності сервісу	25.11-28.11.24	
7	Оформлення роботи: вступ, висновки, реферат	29.11-02.12.24	
8	Розробка демонстраційних матеріалів	03.12-17.12.24	

Здобувач вищої освіти _____ Анатолій ЩЕГОЛЬ

(підпис)

Керівник
кваліфікаційної роботи _____ Наталя ТРІНТІНА

(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 81 стор., 2 табл., 12 рис., 36 джерел.

Мета роботи – покращення швидкодії сервісу моніторингу криптовалютного ринку шляхом оптимізації процесів обробки даних та взаємодії з API.

Об'єкт дослідження – процес обробки даних і взаємодії з API у сервісі моніторингу криптовалютного ринку.

Предмет дослідження – методи та алгоритми оптимізації обробки даних і взаємодії з API для підвищення швидкодії сервісу.

Короткий зміст роботи: у роботі проведено аналіз існуючих підходів до розробки систем моніторингу криптовалютного ринку. Досліджено способи оптимізації швидкодії за рахунок використання технологій кешування, асинхронної обробки даних та мікросервісної архітектури. Розроблено та реалізовано прототип системи з використанням FastAPI та Redis. Проведено тестування швидкодії, що підтвердило ефективність запропонованих рішень у зниженні часу відповіді системи та оптимізації ресурсів.

КЛЮЧОВІ СЛОВА: ШВИДКОДІЯ СЕРВІСУ, ОПТИМІЗАЦІЯ ПРОДУКТИВНОСТІ, КЕШУВАННЯ ДАНИХ, МІКРОСЕРВІСНА АРХІТЕКТУРА, REDIS, FASTAPI, КРИПТОВАЛЮТНИЙ РИНОК.

ABSTRACT

Text part of the master's qualification work: 81 pages, 12 pictures, 2 tables, 36 sources.

The purpose of the work – improving the performance of a cryptocurrency market monitoring service by optimizing data processing and API interaction processes..

Object of research – the process of data processing and interaction with APIs in a cryptocurrency market monitoring service.

Subject of research – methods and algorithms for optimizing data processing and API interactions to enhance the performance of the service.

Summary of the work: the paper analyzes existing approaches to developing cryptocurrency market monitoring systems. Methods for optimizing performance through the use of caching technologies, asynchronous data processing, and microservice architecture have been studied. A prototype system was developed and implemented using FastAPI and Redis. Performance testing confirmed the effectiveness of the proposed solutions in reducing system response time and optimizing resource usage.

KEYWORDS: SERVICE PERFORMANCE, PERFORMANCE OPTIMIZATION, DATA CACHING, MICROSERVICE ARCHITECTURE, REDIS, FASTAPI, CRYPTOCURRENCY MARKET.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД	-	База Даних
API	-	Application Programming Interface
Redis	-	Remote Dictionary Server
AMQP	-	Advanced Message Queuing Protocol
REST	-	Representational State Transfer
HTTP	-	HyperText Transfer Protocol
HTTPS	-	HyperText Transfer Protocol Secure
FastAPI	-	Фреймворк для створення веб-API на Python
CPU	-	Central Processing Unit
RAM	-	Random Access Memory
CLI	-	Command Line Interface
UI	-	User Interface
UX	-	User Experience
ASYNC	-	Асинхронне програмування
LRU	-	Least Recently Used (стратегія витіснення кешу)
LFU	-	Least Frequently Used (стратегія витіснення кешу)
SQL	-	Structured Query Language
OAuth2	-	Open Authorization 2
PostgreSQL	-	Реляційна база даних з відкритим кодом.
Prometheus	-	Система моніторингу та збору метрик
Grafana	-	Інструмент для візуалізації даних і моніторинг
Pydantic	-	Бібліотека для валідації даних і управління типами

ВСТУП

Сучасний світ переживає стрімкий розвиток технологій, які змінюють усі аспекти нашого життя — від повсякденних справ до складних бізнес-процесів. Однією з найбільш динамічних галузей, що привертає увагу як звичайних користувачів, так і великих компаній, є ринок криптовалют. Ця сфера вже стала важливою складовою цифрової економіки, трансформуючи традиційні підходи до фінансів та інвестування.

Ринок криптовалют відрізняється не лише своїми інноваційними особливостями, а й високою волатильністю. Курси валют, обсяги торгів та інші ключові показники змінюються за долі секунди, створюючи серйозні виклики для сервісів, що працюють з такими даними. Важливість швидкості обробки інформації для трейдерів та інвесторів неможливо переоцінити: навіть короткочасні затримки можуть спричинити фінансові втрати.

Значну роль у роботі з такими динамічними даними відіграють сервіси моніторингу криптовалютного ринку. Вони забезпечують користувачів актуальною інформацією щодо цін, ринкових обсягів та капіталізації, допомагаючи приймати обґрунтовані рішення. Найбільш популярними серед таких сервісів є CoinMarketCap, Binance, TradingView, які пропонують широкий функціонал для аналізу ринку. Проте існують і проблеми, з якими вони стикаються:

- Високі затримки у взаємодії з API.

  У періоди пікових навантажень системи можуть працювати нестабільно.
- Недостатня точність або неповнота даних.

  Через обмеження API бірж або технічні проблеми інформація може бути застарілою.
- Обмежена масштабованість.

  Із ростом кількості користувачів продуктивність таких сервісів може значно знижуватися.

Метою цієї роботи є розробка та впровадження методики оптимізації швидкодії сервісу моніторингу криптовалютного ринку, що дозволить зменшити час відповіді системи, покращити її стабільність та забезпечити гнучкість для роботи з великими обсягами даних. Для досягнення поставленої мети було визначено такі завдання:

- Провести аналіз сучасних сервісів моніторингу криптовалютного ринку.
- Вивчити існуючі технічні підходи до оптимізації швидкодії.
- Розробити архітектуру системи з використанням сучасних технологій.
- Реалізувати прототип сервісу на базі FastAPI і Redis.
- Провести тестування продуктивності сервісу та оцінити ефективність запропонованих оптимізацій.

Об’єктом дослідження є процес обробки даних і взаємодії з API у сервісі моніторингу криптовалютного ринку. Предметом дослідження є методи та алгоритми оптимізації, які забезпечують підвищення продуктивності таких систем. Практичне значення отриманих результатів полягає в можливості їх впровадження в реальні сервіси моніторингу криптовалют, що дозволить підвищити їх продуктивність, стабільність та конкурентоспроможність. Розроблений прототип може стати основою для подальшого вдосконалення існуючих технологій у галузі.

Таким чином, дослідження робить вагомий внесок у розвиток високоефективних сервісів, здатних відповідати сучасним викликам динамічного криптовалютного ринку.

1 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

1.1 Огляд сучасних сервісів моніторингу криптовалютного ринку

З розвитком ринку криптовалют та підвищенням інтересу до цифрових активів виникла потреба у створенні високоефективних сервісів моніторингу. Такі сервіси дозволяють інвесторам та трейдерам отримувати актуальну інформацію про курси валют, обсяги торгів та інші ключові показники ринку.

Основні функції сучасних сервісів моніторингу:

1. Відстеження цін у реальному часі:
   Сервіси, такі як CoinMarketCap та Coingecko, дозволяють користувачам отримувати точні дані про курси криптовалют на різних біржах. Це забезпечується за допомогою інтеграції API бірж, які надають дані у режимі реального часу.
2. Аналіз ринку та індикатори:
   Аналітичні платформи, такі як TradingView, використовують спеціалізовані індикатори для автоматичного аналізу ринку. Це дозволяє трейдерам швидко приймати рішення на основі точних прогнозів та інтерпретацій ринкових трендів.
3. Моніторинг обмінників та P2P-платформ:
   Українські сервіси, як Scanbit та Monetory, виконують роль моніторингів обмінників. Вони перевіряють репутацію обмінних сервісів, відстежують курси валют та комісії, забезпечуючи користувачам зручний доступ до найвигідніших пропозицій.
4. Безпека та перевірка обмінників:
   Функція верифікації обмінників на наявність ліцензій та протоколів безпеки дозволяє користувачам довіряти платформі. Наприклад, сервіс Scanbit вручну перевіряє кожен обмінник перед додаванням до бази, забезпечуючи надійність послуг.

5. Портфельний трекінг: платформи, як CoinStats та CryptoCompare, пропонують інструменти для управління криптопортфелями. Це дозволяє інвесторам аналізувати прибутки, збитки та ефективність своїх активів.

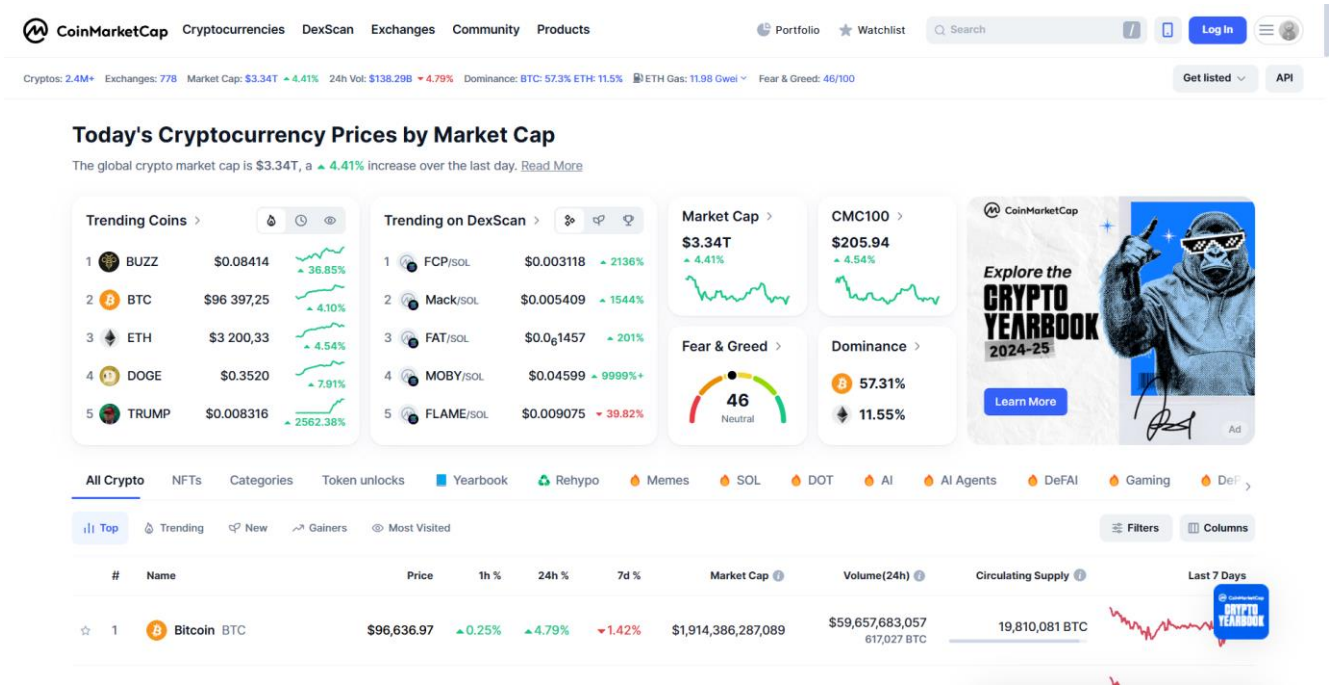


Рис. 1.1 Приклад екранної форми онлайн-ресурсу CoinMarketCap

Незважаючи на різноманітність інструментів, сучасні сервіси моніторингу часто стикаються з такими викликами:

- Затримки в обробці даних: під час пікових навантажень система може мати затримки через неефективну архітектуру або обмеження API.
- Низька пропускна здатність: великий обсяг запитів одночасно може призвести до падіння продуктивності сервісу.
- Неповні дані: деякі API можуть надавати неповну або застарілу інформацію, що впливає на точність даних.

Сучасні тенденції розвитку:

1. Інтеграція AI та ML: використання алгоритмів машинного навчання для аналізу ринку та прогнозування цін.
2. Оптимізація API: застосування асинхронного програмування для підвищення швидкодії сервісів.
3. Зручні мобільні додатки: більшість платформ надають мобільні версії для швидкого доступу до інформації.

Таким чином, сучасні сервіси моніторингу криптовалют забезпечують користувачів необхідною інформацією у режимі реального часу, проте для підвищення їх продуктивності необхідна оптимізація архітектури, інтеграція нових технологій та усунення вузьких місць у системі.

Обробка даних на криптобіржах охоплює широкий спектр процесів, пов'язаних із збором, обробкою та аналізом інформації, що генерується під час роботи з криптовалютними платформами. Ці процеси є ключовими для ефективного функціонування торговельних систем, пов’язаних із крипторинком, і забезпечують успішність торговельних операцій.

Основні етапи обробки даних включають:

- Збір даних: криптобіржі надають широкий спектр інформації про торгові пари, зокрема ціни, обсяги угод та історичні дані. Ці дані можна отримати через публічні API, що забезпечують доступ до інформації бірж.
- Обробка даних: зібрані дані потребують підготовки для подальшого використання. Цей етап включає перевірку, очищення, перетворення у зручний формат, нормалізацію, агрегацію та інші дії, спрямовані на підвищення точності аналізу та врахування особливостей ринку.
- Статистичний аналіз: використання статистичних методів, математичних моделей та алгоритмів машинного навчання для виявлення трендів, шаблонів і закономірностей. Ці результати служать основою для прийняття обґрунтованих рішень у торгівлі.

Після виконання всіх етапів дані стають інструментом для прийняття виважених торгових рішень. Залежно від доступного капіталу, типу торгівлі та рівня допустимих ризиків, можна використовувати різні торговельні стратегії. Враховуючи, що універсальної стратегії, придатної для всіх випадків, не існує, важливо адаптувати підхід відповідно до особливостей ринку.

Більшість криптобірж працюють на основі технології блокчейн і надають можливість торгівлі криптовалютами, які є цифровими активами. Це означає, що криптобіржі функціонують у децентралізованому середовищі, де операції здійснюються без участі традиційних посередників, таких як банки чи фінансові установи.

Основні відмінності криптобірж від фондових бірж:

- Швидкість: завдяки відсутності посередників та повній автоматизації, криптобіржі працюють цілодобово, 24/7, без прив’язки до традиційного робочого графіка будніх днів або певного часового діапазону.
- Відкритість і глобальність: вони доступні користувачам з усього світу, забезпечуючи широкі можливості для торгівлі.
- Розмаїття: пропонують торгівлю багатьма різними криптовалютами, що дає змогу інвесторам диверсифікувати свої активи.
- Технічні особливості: через специфіку технічної інфраструктури та питань безпеки, криптобіржі частіше стикаються з технічними збоями, кібератаками та іншими проблемами порівняно з традиційними біржами.
- Менша регуляція: криптобіржі діють у менш регульованому середовищі, що може призводити до більшої волатильності та непередбачуваності ринку.

Ф’ючерсна торгівля на криптобіржах надає трейдерам широкий спектр можливостей завдяки своїй гнучкості та високому потенціалу для отримання прибутку. Важливим фактором є значний обсяг угод, який забезпечує високу ліквідність ринку. Це дозволяє трейдерам оперативно відкривати та закривати позиції, знижуючи витрати на спред між ціною купівлі та продажу. Наприклад, на

популярних біржах, таких як Binance або Bybit, трейдери можуть реалізовувати свої стратегії навіть у періоди пікової активності завдяки стабільній роботі систем. Крім того, ф'ючерсний ринок дозволяє використовувати різноманітні стратегії, зокрема хеджування, арбітраж і спекуляцію. Хеджування є ефективним способом захисту від ризиків волатильності, коли інвестори відкривають одночасно довгу і коротку позиції, стабілізуючи фінансові результати. Арбітражні операції дозволяють скористатися ціновими розбіжностями між різними ринками, отримуючи прибуток за рахунок цих короткочасних невідповідностей. Спекулятивна торгівля, орієнтована на прогнозування змін цін, також залишається популярним підходом, який дає змогу заробляти навіть у короткостроковій перспективі.

Важливою перевагою є доступність таких інструментів для трейдерів з обмеженим капіталом. Завдяки використанню кредитного плеча вони можуть відкривати позиції, які значно перевищують їхній початковий депозит, що підвищує потенційний прибуток. Наприклад, при використанні плеча 10x трейдер із депозитом у 1000 доларів може укладати угоди на суму до 10 000 доларів. Однак, одночасно з цим збільшуються і ризики, адже кожне коливання ціни може суттєво вплинути на результати торгівлі.

Застосування маржі є ключовим елементом ф'ючерсної торгівлі. Вона дозволяє трейдерам працювати з позиковими коштами, збільшуючи обсяг угод, але вимагає обережного підходу. Біржі встановлюють мінімальні вимоги до маржі, щоб забезпечити покриття можливих збитків. Наприклад, при використанні плеча 20x навіть невелика зміна ціни може призвести до значних втрат, якщо ризики не були заздалегідь розраховані.

Водночас ф'ючерсна торгівля пов'язана з певними ризиками, зокрема з високою волатильністю криптовалютного ринку. Цінові коливання можуть бути різкими й непередбачуваними, що створює значний ризик втрат. Використання позикових коштів, хоча й відкриває додаткові можливості для отримання прибутку, значно збільшує ймовірність ліквідації позицій у разі несприятливого

руху ринку. Успішна торгівля вимагає глибокого розуміння ринкової динаміки, знання технічного аналізу та вміння управляти капіталом.

Ще одним популярним інструментом на криптовалютному ринку є опціони. Вони дають трейдерам право, але не зобов'язання, купувати або продавати криптовалюту за визначеною ціною до певної дати. Це дозволяє гнучко керувати своїми інвестиціями, хеджувати ризики і навіть заробляти на волатильності ринку без необхідності володіти самим активом. Такий інструмент підходить як для досвідчених трейдерів, так і для новачків, які тільки знайомляться з можливостями криптовалютного ринку.

Переваги опціонів:

- Дозволяють отримувати вигоду від цінових коливань без необхідності фактичного володіння криптовалютою.
- Використовуються для захисту від ризиків, спекуляцій на ринку та управління портфелем.
- Надають трейдерам гнучкість у виборі стратегій, які відповідають їх цілям.

Ще одним прикладом сучасних аналітичних платформ є TradingView. Цей сервіс пропонує широкий спектр інструментів для графічного аналізу ринку та розробки торгових стратегій. Інтуїтивно зрозумілий інтерфейс і велика кількість налаштувань роблять його ідеальним вибором як для початківців, так і для досвідчених трейдерів. TradingView забезпечує користувачам доступ до великої бібліотеки індикаторів, таких як ковзні середні, MACD, RSI та багатьох інших, які допомагають аналізувати ринкові тренди та ухвалювати зважені рішення. Крім того, платформа дозволяє створювати власні скрипти для автоматизації процесів аналізу. Однією з основних переваг TradingView є його інтеграція з провідними криптовалютними біржами, що забезпечує актуальність і точність наданих даних.

Рис. 1.2 Приклад екранної форми онлайн-ресурсу TradingView

На додаток до TradingView, важливе місце серед інструментів займає платформа Winscreener, яка спеціалізується на відборі активів за заданими критеріями. Цей сервіс надає користувачам можливість швидко знаходити перспективні криптовалюти, використовуючи фільтри за такими параметрами, як обсяг торгів, зміна ціни, ринкова капіталізація та інші показники. Winscreener особливо корисний для інвесторів, які шукають можливості для диверсифікації портфеля або для короткострокових спекуляцій. Завдяки простоті у використанні, ця платформа дозволяє заощаджувати час на пошук активів, які відповідають певним умовам ринку.

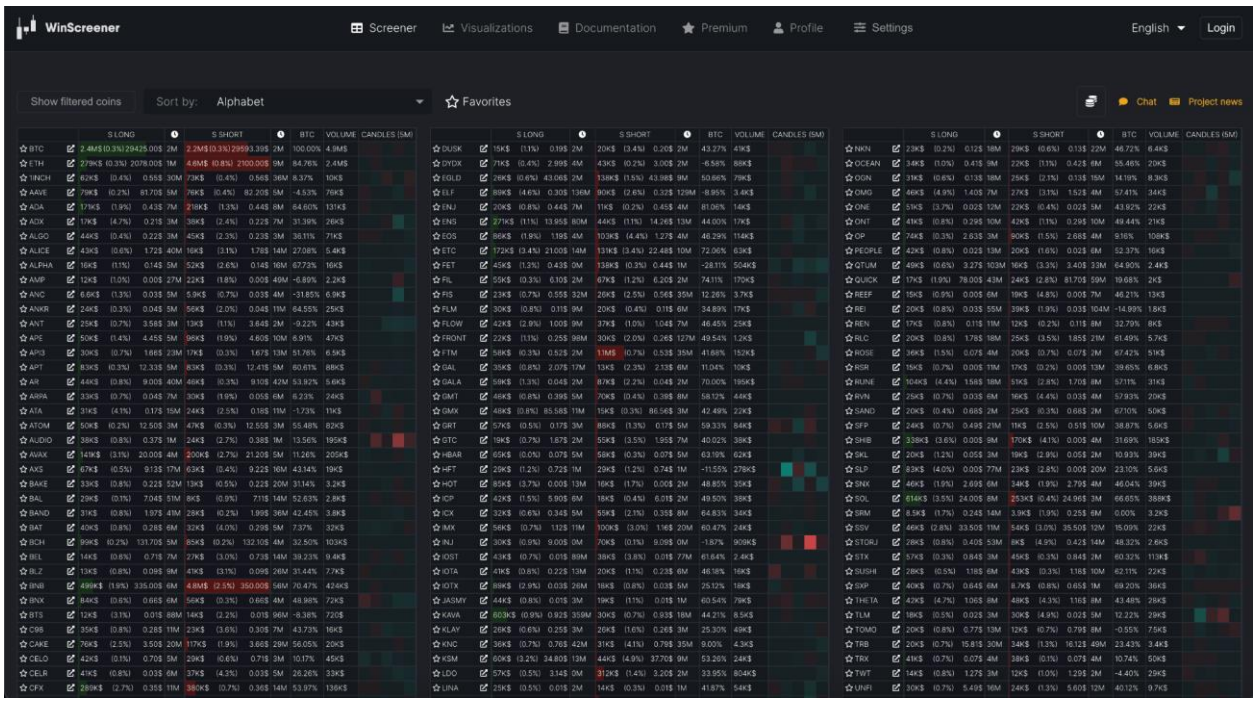


Рис. 1.3 Приклад екранної форми онлайн-ресурсу Winscreener

Обидві платформи є чудовими прикладами того, як сучасні технології покращують процеси моніторингу та аналізу криптовалютного ринку. Їх використання дозволяє трейдерам і інвесторам ефективніше орієнтуватися в умовах високої волатильності та приймати більш обґрунтовані рішення.

Окрім TradingView та Winscreener, важливе місце серед сучасних інструментів моніторингу та аналізу криптовалютного ринку займає платформа Binance. Ця криптовалютна біржа не лише є однією з найбільших і найпопулярніших у світі, але й пропонує широкий спектр функціональних можливостей для трейдерів, аналітиків та інвесторів. Її багатофункціональність та інтеграція сучасних технологій роблять Binance універсальним інструментом для роботи з цифровими активами.

Binance забезпечує доступ до глибокого аналізу ринку через зручний та інтуїтивно зрозумілий інтерфейс. На платформі доступні численні графічні інструменти, які допомагають відслідковувати ринкові тренди, оцінювати динаміку цін та аналізувати обсяги торгів. Наприклад, користувачі можуть

налаштувати графіки для роботи з різними таймфреймами, що особливо важливо для трейдерів, які займаються короткостроковою спекулятивною торгівлею. Однією з ключових функцій Binance є її можливості технічного аналізу. Платформа надає доступ до великої бібліотеки індикаторів, таких як ковзні середні (MA), індекс відносної сили (RSI), стохастик, MACD та інші. Ці інструменти дозволяють трейдерам отримувати точні прогнози щодо ринкових трендів, аналізувати рівні підтримки і опору, а також приймати обґрунтовані рішення щодо відкриття чи закриття позицій.

Крім того, Binance пропонує унікальну функцію Spot Grid Trading, яка дає змогу автоматизувати процес торгівлі на основі заданих параметрів. Цей інструмент дозволяє встановлювати цінові рівні, між якими система автоматично виконує операції купівлі або продажу, використовуючи можливості короткострокових коливань ринку. Така функція особливо корисна для трейдерів, які прагнуть отримати максимальний прибуток, мінімізуючи ризики, пов'язані з ручним керуванням.

Не менш важливою особливістю Binance є її API, який дозволяє інтегрувати платформу з іншими програмними рішеннями. Це забезпечує широкі можливості для автоматизації процесів торгівлі та аналізу даних. За допомогою API трейдери можуть створювати власних торгових ботів, автоматизувати обробку даних та налаштовувати систему під свої потреби. Висока швидкість обробки запитів і стабільність API роблять Binance одним із найкращих виборів для професійних трейдерів, які активно використовують автоматизовані інструменти у своїй діяльності.

Binance також надає широкий вибір інструментів для роботи з деривативами, включаючи ф'ючерси, які дозволяють трейдерам отримувати прибуток як на зростанні, так і на падінні ринку. Інструменти для управління ризиками, такі як налаштування рівнів ліквідації та використання стоп-лосів, забезпечують додаткову безпеку в торгівлі.

Завдяки своїй багатофункціональності Binance є не лише біржею для торгівлі криптовалютами, але й повноцінною аналітичною платформою.

Її потужні інструменти аналізу, автоматизації та управління ризиками роблять цю платформу ідеальним вибором для трейдерів будь-якого рівня, від новачків до професіоналів.

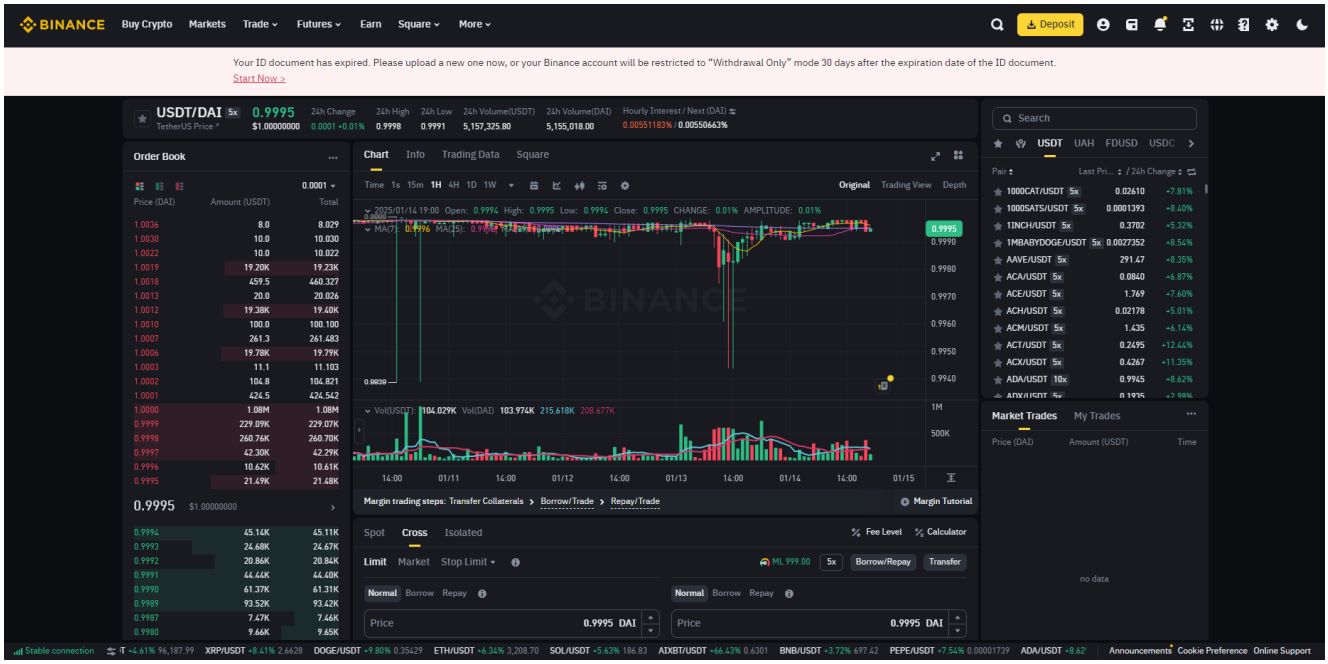


Рис. 1.3 Приклад екранної форми онлайн-ресурсу Winscreener

Таким чином, Binance є одним із найяскравіших прикладів сучасних технологій у сфері моніторингу та аналізу криптовалютного ринку. Її інтеграція з передовими аналітичними інструментами, гнучкі можливості автоматизації та постійне впровадження інновацій роблять цю платформу невід'ємною частиною криптовалютної екосистеми.

1.2 Проблеми обробки даних і взаємодії з API в сервісах моніторингу

Однією з найбільш критичних проблем є залежність швидкодії сервісів від обмежень зовнішніх API. Наприклад, біржі, такі як Binance, часто застосовують обмеження швидкості (rate limits), встановлюючи максимальну кількість запитів на секунду. У таких випадках важливо враховувати можливість реалізації черг запитів, які працюють на основі алгоритмів пріоритетності. Це дозволяє оптимізувати доступ до ресурсів і забезпечити своєчасне оновлення ключових даних.Крім того, нестабільність API може призводити до втрати даних, особливо під час пікових навантажень. Для уникнення цього, багато сервісів інтегрують системи моніторингу стабільності API, що автоматично перемикаються на альтернативні джерела даних у разі збою.

Щодо обробки великих обсягів даних, новітні підходи передбачають впровадження потокової обробки даних у реальному часі. Наприклад, використання брокерів повідомлень, таких як Kafka, дозволяє забезпечити масштабовану інфраструктуру для збору та агрегації даних із сотень джерел одночасно. Для підвищення продуктивності сервісів можна застосовувати методи кешування не лише для популярних даних, а й для проміжних результатів обчислень, що використовуються в аналітичних задачах. Це знижує кількість повторних обчислень і пришвидшує доступ до складних звітів.

Ще однією важливою інновацією є адаптивне управління запитами, яке дозволяє сервісам автоматично коригувати частоту запитів залежно від стану зовнішнього API. Наприклад, якщо API відповідає повільно, сервіс може автоматично зменшувати частоту запитів, щоб уникнути блокувань.

Таким чином, удосконалення підходів до обробки даних та інтеграції з API може значно підвищити стабільність і надійність роботи систем моніторингу криптовалютного ринку.

Проблеми взаємодії з API

- Затримка у відповідях API

  Багато бірж обмежують швидкість запитів (rate limits), що створює затримки у збиранні даних. Наприклад, біржа може дозволяти не більше 10 запитів на секунду, що обмежує швидкість оновлення інформації у сервісі.
- Нестабільність зовнішніх API

  Часті випадки недоступності або повільної відповіді зовнішніх API можуть спричинити втрату даних або невідповідність інформації, що відображається користувачам. Це потребує реалізації механізмів обробки помилок та резервування даних.
- Низька якість даних

  Дані з різних джерел можуть мати різний формат або містити помилки. Для забезпечення коректності необхідно проводити валідацію та нормалізацію отриманих даних перед їх збереженням.

Рішення:

- Використання асинхронного програмування для паралельної обробки запитів та зменшення затримок.
- Реалізація механізму повторних спроб (retry mechanism) при відмовах API.
- Впровадження стратегій кешування для зменшення кількості запитів до API.

Проблеми обробки великих обсягів даних

- Обмеження баз даних

  Зростання обсягу даних у системі може призводити до уповільнення запитів та зниження продуктивності. Це часто виникає через недостатню індексацію або неефективну структуру бази даних.
- Високе навантаження на сервери

  Обробка великої кількості запитів у пікові періоди призводить до зростання навантаження на серверну інфраструктуру. Це може призвести до збоїв або

втрат продуктивності.

- Неоптимізовані алгоритми обробки даних

  Використання неефективних алгоритмів для обробки даних (наприклад, дублікатів або непотрібних обчислень) збільшує час обробки та витрати ресурсів.

Рішення:

- Оптимізація структури бази даних, зокрема використання індексів для швидкого пошуку.
- Використання кешування за допомогою Redis для швидкого доступу до часто запитуваних даних.
- Впровадження балансування навантаження між кількома серверами для обробки великої кількості запитів.
- Реалізація паралельної обробки даних для зменшення часу їх обробки.
- Узгодження даних у реальному часі
- Синхронізація даних

  Одна з проблем сервісів моніторингу полягає в необхідності синхронізації даних у реальному часі. Системи повинні постійно оновлювати інформацію про курси валют, обсяги торгів та інші показники.
- Конфлікти даних

  Дані з різних бірж можуть містити розбіжності через різні часові пояси, затримки або помилки в API.

Рішення:

- Використання timestamp для кожного запиту для визначення актуальності даних.
- Реалізація механізму валідації для порівняння даних з кількох джерел та вибору найнадійніших значень.

В таблиці 1 наведено порівняння підходів до отримання даних із зазначенням їх особливостей і ключових недоліків.

Таблиця 1.1

Порівняння підходів до отримання даних

Метод	Особливості	Ключові недоліки
Ручний моніторинг	Безпосереднє стеження за ринком користувачем	Низька ефективність при великих обсягах даних, людський фактор
REST API	Стандартна реалізація для збору даних через HTTP-запити	Погана масштабованість, високе навантаження при одночасному запиті великої кількості даних
WebSocket-API	Прямі підключення для отримання актуальних даних у реальному часі	Проблеми з надійністю з'єднання та синхронізацією даних
Моніторинг з обробкою даних на стороні сервера	Оптимізація завдяки попередній обробці та фільтрації даних на сервері	Високі затримки при обробці великих масивів інформації
Метод із кешуванням	Зменшує кількість запитів до API, підвищуючи швидкодію.	Може видавати застарілі дані при високій частоті оновлення ринку.

Таблиця містить порівняння підходів до отримання даних із зазначенням їх особливостей і ключових недоліків. Метод із кешуванням виявився найбільш ефективним для оптимізації швидкодії сервісу.

Таким чином, ключові проблеми у сервісах моніторингу криптовалют пов’язані зі взаємодією з API, обробкою великих обсягів даних та забезпеченням синхронізації у реальному часі. Їх вирішення вимагає впровадження асинхронних запитів, оптимізації баз даних, балансування навантаження та застосування механізмів кешування.

1.3 Технічні підходи до оптимізації швидкодії сервісів моніторингу

Оптимізація швидкодії сервісів моніторингу криптовалютного ринку є важливим етапом для забезпечення надійної та ефективної роботи системи. Це включає в себе впровадження сучасних технічних рішень, що сприяють зменшенню затримок, підвищенню продуктивності та обробці великих обсягів даних у реальному часі.

Основні підходи до оптимізації швидкодії:

1. Асинхронне програмування

   Асинхронне програмування дозволяє обробляти декілька запитів одночасно, не блокуючи виконання інших завдань. У контексті сервісів моніторингу це є ключовим інструментом для підвищення продуктивності при взаємодії з API та базами даних.

   Принцип роботи: Використання ключових слів async та await дозволяє сервісу виконувати запити до декількох API паралельно, скорочуючи загальний час обробки.

   Переваги:

   - Зменшення часу очікування відповіді API.
   - Підвищення пропускної здатності сервера завдяки обробці багатьох запитів одночасно.
   - Зниження використання ресурсів сервера порівняно з багатопоточними моделями.
2. Оптимізація роботи з базами даних

   Бази даних є одним із найкритичніших компонентів системи моніторингу, оскільки вони забезпечують збереження та швидкий доступ до даних. Основні підходи до оптимізації включають:

   - Індексація даних: створення індексів на часто використовуваних полях, таких як символ криптовалюти або часова мітка, що дозволяє значно

пришвидшити пошук даних.

- Кешування: використання кешу (наприклад, Redis) для тимчасового зберігання часто запитуваних даних. Це дозволяє зменшити кількість запитів до основної бази даних.
- Пагінація: розбиття великих наборів даних на невеликі сторінки для ефективної обробки та передачі.

Переваги:

- Індексація даних: створення індексів на часто використовуваних полях, таких як символ криптовалюти або часова мітка, що дозволяє значно пришвидшити пошук даних.
- Кешування: використання кешу (наприклад, Redis) для тимчасового зберігання часто запитуваних даних. Це дозволяє зменшити кількість запитів до основної бази даних.
- Пагінація: розбиття великих наборів даних на невеликі сторінки для ефективної обробки та передачі.

3. Балансування навантаження

Балансування навантаження дозволяє рівномірно розподіляти запити між кількома серверами, що запобігає перевантаженню окремих вузлів системи.

Методи балансування:

- DNS-балансування:
  Розподіл запитів на рівні доменних імен.
- Балансувальники навантаження (Load Balancers):
  Використання спеціалізованих інструментів (наприклад, NGINX або HAProxy) для обробки запитів та перенаправлення їх на вільні сервери.
-

Переваги:

- Підвищення стабільності системи під час пікових навантажень.
- Зниження ризику відмови сервісу через перевантаження.

4. Кешування даних

   Кешування є одним із найбільш ефективних методів оптимізації швидкодії сервісів. Воно дозволяє зберігати часто використовувані дані в пам’яті для миттєвого доступу.

   Типи кешування:

   - На стороні сервера (Redis): швидкий доступ до популярних даних, таких як поточні ціни на криптовалюту.
   - На стороні клієнта: зберігання проміжних даних у браузері або додатку для зменшення кількості запитів до сервера.

   Переваги:

   - Значне зменшення часу відповіді.
   - Оптимізація використання ресурсів серверної інфраструктури.

5. Оптимізація обсягу переданих даних

   Зменшення розміру даних, що передаються між сервером та клієнтом, є важливим для швидкої обробки запитів.

   Методи оптимізації:

   - Стиснення даних: використання форматів Gzip або Brotli для зменшення обсягу переданих даних.
   - Оптимізація JSON-відповідей: виключення непотрібних полів з відповідей API.
   - Використання ефективних форматів даних: наприклад, Protocol Buffers замість JSON для великих наборів даних.

Таким чином, оптимізація швидкодії сервісів моніторингу передбачає застосування сучасних технічних рішень, таких як асинхронне програмування, балансування навантаження, індексація даних та стратегічне кешування. Використання цих підходів дозволяє підвищити продуктивність системи, зменшити затримки та забезпечити стабільну роботу сервісу в умовах високих навантажень.

2 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

2.1 Архітектура програми

Архітектура програми — це фундамент, на якому будується вся система. Вона визначає, як різні компоненти взаємодіють між собою, забезпечуючи високу продуктивність, масштабованість, стабільність і легкість у підтримці. Для розробки сервісу моніторингу криптовалютного ринку була обрана модульна архітектура, яка дозволяє розділити систему на автономні компоненти, кожен з яких відповідає за конкретний функціонал. Такий підхід забезпечує не лише легкість внесення змін, але й можливість розширення системи без значних перебудов.

Основні компоненти системи:

1. Модуль збору даних

   Цей компонент відповідає за отримання інформації з різних криптовалютних бірж у реальному часі. Для інтеграції з біржами використовуються їхні публічні API, які надають дані про котирування, обсяги торгів та інші ключові показники. Однією з важливих особливостей є підтримка одночасного збору даних із кількох джерел, що реалізується за допомогою асинхронного програмування. Це дозволяє значно зменшити затримки при обробці запитів. Модуль також має гнучкі налаштування, які дозволяють вибирати конкретні джерела даних залежно від потреб користувача чи вимог системи.
2. Модуль обробки даних

   Отримані дані часто мають різну структуру та формат, що потребує їх стандартизації.

   Модуль обробки виконує:

   - Нормалізацію даних, тобто приведення інформації до єдиного формату. Це спрощує подальшу роботу з ними.

- Валідацію, що включає перевірку наявності всіх необхідних полів, їхнього формату та логічної цілісності. Наприклад, перевіряється, чи не є курси валют від'ємними або чи відповідає часова позначка реальному часу.
- Аналіз отриманих даних, який може включати попереднє виявлення трендів або оцінку волатильності ринку для покращення аналітики.

3. Система збереження даних

   Для ефективної роботи із збереженими даними система використовує два типи сховищ:

   - Реляційна база даних (наприклад, PostgreSQL) для зберігання історичних даних, таких як котирування за минулі дні чи години. Це забезпечує можливість виконання складних запитів для аналітики.
   - Кеш-пам'ять (наприклад, Redis) для швидкого доступу до даних, які запитуються найчастіше, наприклад, поточних курсів валют. Таке рішення дозволяє значно зменшити навантаження на базу даних і пришвидшити роботу сервісу.

4. API-сервер

   API-сервер є основним інтерфейсом між серверною частиною програми та клієнтами. Він дозволяє взаємодіяти з даними через зручний набір запитів. Реалізація API базується на технології FastAPI, яка забезпечує високу продуктивність та підтримку асинхронних операцій.

   Основні можливості API-сервера включають:

   - Надання актуальних даних у реальному часі.
   - Доступ до історичних даних для побудови аналітики.
   - Підтримку різних форматів даних (JSON, XML) залежно від потреб клієнта.

   Переваги модульної архітектури:

   - Гнучкість: система легко адаптується до змін. Наприклад, у разі появи нової біржі, достатньо додати відповідну інтеграцію лише в модуль збору даних.
   - Масштабованість: компоненти можуть працювати незалежно один від

одного, що дозволяє розподіляти навантаження між кількома серверами.

- Надійність: помилка в одному модулі не впливає на роботу інших компонентів, що забезпечує стабільність системи.
- Зручність тестування та підтримки: кожен модуль можна тестувати окремо, що спрощує виявлення та виправлення помилок.

Таким чином, архітектура програми є одним із ключових факторів її успішної роботи. Правильний підхід до її розробки дозволяє створити систему, яка відповідає сучасним вимогам до швидкодії, надійності та зручності використання.

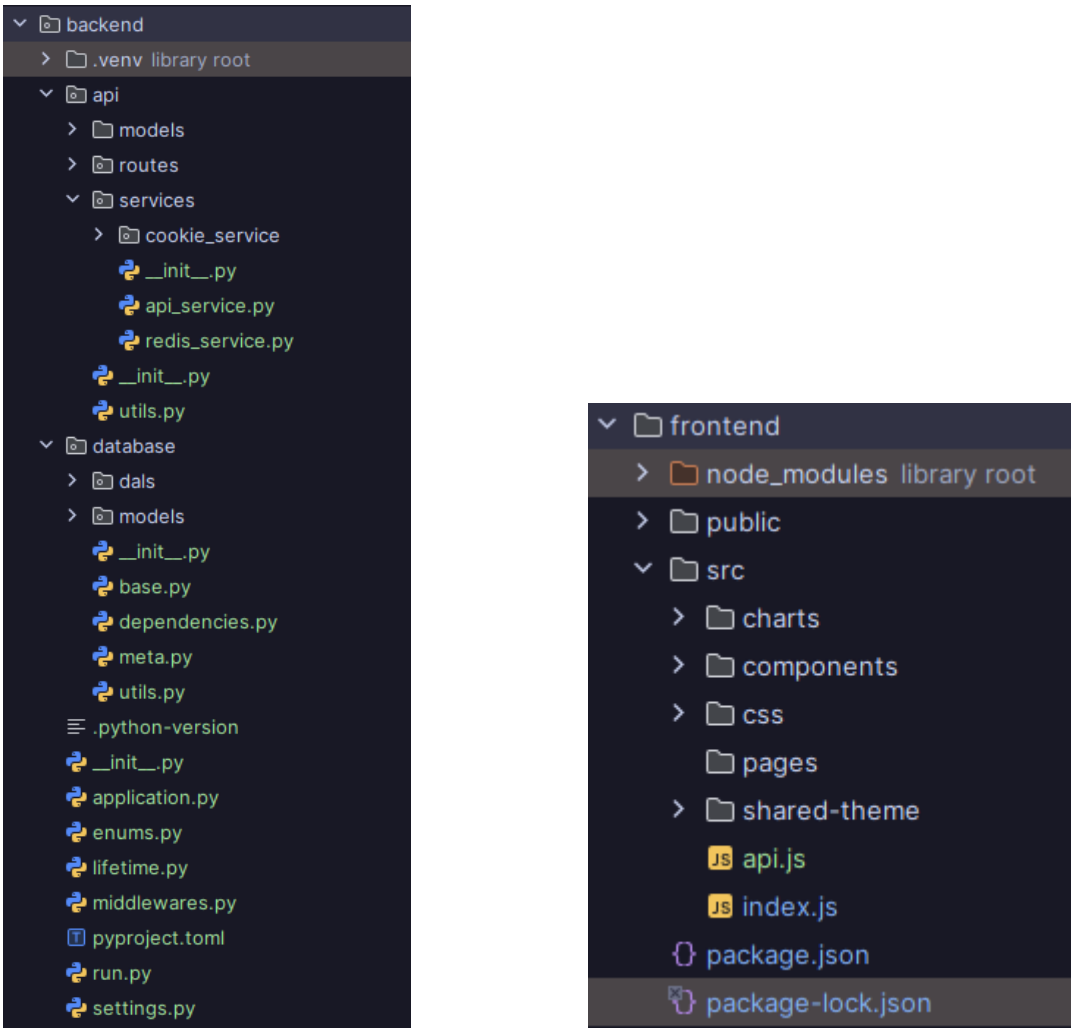


Рис. 2.1 Файлова структура системи моніторингу криптовалютного ринку

2.2 Реалізація високопродуктивного бекенду на основі FastAPI

Для створення бекенду системи моніторингу криптовалютного ринку обрано FastAPI — сучасний фреймворк, який забезпечує широкий набір функцій для створення продуктивних та масштабованих API-додатків. Завдяки своїй архітектурі та використанню новітніх технологій Python, FastAPI дозволяє не лише оптимізувати швидкість роботи, а й спростити процес розробки. Однією з ключових переваг FastAPI є підтримка анотацій типів у Python, що дозволяє автоматично перевіряти коректність даних у запитах та відповідях. Це значно підвищує стабільність роботи системи, зменшуючи кількість помилок на етапі виконання. Крім того, цей підхід полегшує написання тестів і документування коду, що робить його зручним для роботи в команді.

FastAPI також надає автоматично створену документацію API на основі стандарту OpenAPI (Swagger). Це дозволяє розробникам інтерактивно взаємодіяти з API, перевіряти його функціонал і швидко інтегрувати сторонні сервіси. Завдяки цьому фреймворк є ідеальним вибором для проєктів, що потребують постійного розширення або взаємодії з іншими системами.

Ще однією перевагою FastAPI є його інтеграція з різноманітними бібліотеками, такими як SQLAlchemy для роботи з базами даних, Redis для кешування даних і Celery для обробки фонових задач. Це дозволяє створювати складні багатокомпонентні системи з чіткою структурою і високою продуктивністю. У випадку системи моніторингу криптовалютного ринку це забезпечує швидку обробку даних, оптимізацію запитів до бази даних і паралельну обробку завдань, таких як оновлення котирувань чи відправлення повідомлень. Окрім цього, фреймворк підтримує розширені механізми безпеки, зокрема вбудовану систему аутентифікації та авторизації, що дозволяє легко впроваджувати такі стандарти, як OAuth2. Це особливо важливо для проєктів, які працюють з конфіденційними даними або фінансовими операціями.

Таким чином, вибір FastAPI як основи для створення бекенду системи моніторингу криптовалютного ринку є оптимальним рішенням, яке поєднує

високу продуктивність, легкість розробки та розширені можливості для інтеграції. Фреймворк забезпечує стабільність і гнучкість системи, що є критично важливим для роботи в умовах постійно зростаючих вимог сучасного ринку.

В таблиці 1.2 наведено порівняння підходів до отримання даних із зазначенням їх особливостей і ключових недоліків.

Таблиця 1.2

Переваги вибору FastAPI

	Django	Flask	FastAPI
Архітектура	MVC (MVT)	Мікрофреймворк	Мікрофреймворк
Продуктивність	Середня (Синх)	Середня (Синх)	Висока (Асинх)
ORM	Django ORM	Підтримується, але не вбудований	Підтримується, але не вбудований (SQLalchemy)
Валідація даних	Django Forms	Користувацька	Pydantic
Масштабованість	Середня	Середня	Висока
Гнучкість	Середня	Середня	Висока
Швидкість розробки	Середня	Висока	Висока

Таблиця містить порівняння трьох популярних фреймворків для розробки бекенду. У ній відображені їх ключові характеристики, такі як архітектура, продуктивність, підтримка ORM, можливості валідації даних, масштабованість, гнучкість та швидкість розробки.

Наведемо основні переваги FastAPI:

Висока швидкодія. FastAPI побудований на основі Starlette для обробки запитів та Pydantic для перевірки даних. Це дозволяє створювати високопродуктивні API, продуктивність яких порівнянна з такими фреймворками, як Node.js або Go.

Автоматична генерація документації. FastAPI автоматично створює інтерактивну документацію API відповідно до стандарту OpenAPI. Це значно спрощує тестування та інтеграцію клієнтських застосунків, а також дозволяє розробникам швидко зрозуміти структуру та функціонал API.

Підтримка асинхронного програмування. Завдяки використанню async та await, фреймворк дозволяє обробляти тисячі одночасних запитів, що особливо важливо для сервісів із високим навантаженням.

У бекенді були враховані кілька ключових аспектів, які забезпечують його стабільність, безпеку та функціональність.

Асинхронні запити. Ендпоїнти реалізовані з використанням асинхронного програмування. Це дозволяє одночасно обробляти велику кількість клієнтських запитів без зниження швидкодії системи.

Обробка помилок. Реалізована централізована система обробки винятків (Exception Handling). У разі виникнення помилки вона логує всі деталі в систему моніторингу, а також формує зрозуміле повідомлення для користувача. Наприклад, якщо запит містить некоректні дані, користувач отримає відповідь із відповідним кодом помилки (400 Bad Request) та описом проблеми.

Безпека API. Усі запити до бекенду захищені за допомогою JSON Web Token (JWT).

Система аутентифікації передбачає:

Авторизацію за токенами.

Обмеження доступу до окремих ендпоїнтів залежно від доступів користувача.

Використання HTTPS для шифрування переданих даних.

Наведемо приклади ендпоїнтів.

1. Отримання останніх котирувань

   Цей ендпоїнт надає актуальну інформацію про курси криптовалют. Запит обробляється асинхронно, і дані отримуються з кешу або бази даних.

```python
@app.get("/prices")
async def get_prices(symbol: str):
    """
    Повертає актуальні дані про котирування криптовалют.
    :param symbol: Тікер криптовалюти (BTC, ETH).
    :return: Словник із даними про курс.
    """
    try:
        data = {"symbol": symbol, "price": 50234.12}
        return data
    except Exception as e:
        raise HTTPException(status_code=500, detail=f"Помилка сервера: {str(e)}")
```

Рис. 2.2 Ендпоїнт отримання останніх котирувань

2. Збереження налаштувань користувача

   Цей ендпоїнт дозволяє зберігати налаштування користувача, наприклад, обрані криптовалюти для моніторингу чи рівень оповіщення.

```python
from pydantic import BaseModel

class UserSettings(BaseModel):  1 usage
    user_id: int
    preferred_symbols: list[str]
    alert_threshold: float

@app.post("/settings")
async def save_settings(settings: UserSettings):
    """
    Зберігає налаштування користувача.
    :param settings: Об'єкт із налаштуваннями користувача.
    :return: Повідомлення про успішність збереження.
    """
    try:
        return {"message": "Налаштування успішно збережено!"}
    except Exception as e:
        raise HTTPException(status_code=500, detail=f"Помилка сервера: {str(e)}")
```

Рис. 2.3 Ендпоїнт збереження налаштувань користувача

3. Аутентифікація користувачів

Реалізований ендпоїнт для генерації токена доступу.

```python
from fastapi.security import OAuth2PasswordRequestForm
from datetime import datetime, timedelta
import jwt

SECRET_KEY = "secret_key"

@app.post("/token")
async def generate_token(form_data: OAuth2PasswordRequestForm):
    """
    Генерує JWT-токен для аутентифікації користувача.
    :param form_data: Дані для входу (ім'я користувача та пароль).
    :return: Токен доступу.
    """
    if form_data.username == "admin" and form_data.password == "password123":
        token = jwt.encode(
            {"sub": form_data.username, "exp": datetime.utcnow() + timedelta(hours=1)},
            SECRET_KEY,
            algorithm="HS256",
        )
        return {"access_token": token, "token_type": "bearer"}
    else:
        raise HTTPException(status_code=401, detail="Неправильні облікові дані")
```

Рис. 2.4 Ендпоїнт аутентифікація користувачів

Переваги реалізації на основі FastAPI:

- Продуктивність: обробка запитів відбувається за мінімальний час, що критично для високонавантажених сервісів.
- Зручність розробки: інтерактивна документація спрощує тестування API.
- Масштабованість: можливість легко додавати нові ендпоїнти та розширювати функціонал.

Така реалізація бекенду є оптимальним рішенням для сучасних веб-додатків, забезпечуючи баланс між продуктивністю, безпекою та зручністю використання.

Розглянемо модулі для збору, обробки та збереження даних криптовалютного ринку. Цей компонент є центральною частиною сервісу, оскільки його продуктивність, надійність і точність визначають якість даних, які надаються користувачам. Модулі розділені на три основні частини: збір даних, обробка даних

та збереження даних.

Модуль відповідає за інтеграцію з API криптовалютних бірж для отримання інформації про курси валют, обсяги торгівлі, ордери тощо. Для цього використовується бібліотека httpx, яка підтримує асинхронні запити, що забезпечує паралельний збір даних із кількох джерел.

Наведемо перелік оптимізації збору даних.

Асинхронна обробка запитів: дозволяє надсилати десятки запитів одночасно, скорочуючи час збору даних.

Кешування відповідей API: уникає повторного запиту тієї ж інформації у короткі проміжки часу, що зменшує навантаження на API біржі. Для цього використовується Redis.

Контроль за лімітами запитів: уникає блокування доступу, модуль дотримується обмежень на кількість запитів, накладених API бірж.

Для збереження оброблених даних використовується PostgreSQL як основна база даних і Redis для кешування.

Розглянемо використання PostgreSQL:

Індексація: встановлені індекси на поля, які часто використовуються у запитах (наприклад, символ криптовалюти, час).

Схема даних: дані зберігаються у таблицях, організованих для швидкого пошуку та аналізу.

На рис. 2.5 – 2.8 наведено перелік прикладів оптимізації збору даних.

```
CREATE TABLE crypto_data (
    id SERIAL PRIMARY KEY,
    symbol VARCHAR(10) NOT NULL,
    price NUMERIC(10, 2) NOT NULL,
    volume NUMERIC(10, 2),
    timestamp TIMESTAMP NOT NULL
);
CREATE INDEX idx_symbol ON crypto_data(symbol);
CREATE INDEX idx_timestamp ON crypto_data(timestamp);
```

Рис. 2.5 Створення таблиці та індексів

```python
async def save_crypto_data(session: AsyncSession, data: dict):
    """
    Зберігає дані криптовалют у базі даних.
    :param session: Сесія SQLAlchemy.
    :param data: Дані для збереження.
    """
    new_data = CryptoData(
        symbol=data["symbol"],
        price=data["price"],
        volume=data["volume"],
        timestamp=data["timestamp"],
    )
    session.add(new_data)
    await session.commit()
```

Рис. 2.6 Приклад збереження даних у PostgreSQL

Redis використовується для кешування найбільш запитуваних даних, що значно зменшує навантаження на базу даних і покращує швидкодію.

```python
redis_client = redis.StrictRedis(host="localhost", port=6379, db=0)

def cache_data(key: str, value: dict, ttl: int = 3600):
    """
    Зберігає дані в кеш Redis.
    :param key: Ключ кешу.
    :param value: Дані для збереження.
    :param ttl: Час життя ключа в секундах.
    """
    redis_client.setex(key, ttl, json.dumps(value))
```

Рис. 2.7 Приклад використання Redis

```python
async def get_data_with_cache(symbol: str):
    cache_key = f"crypto:{symbol}"
    cached_data = redis_client.get(cache_key)

    if cached_data:
        return json.loads(cached_data)

    # Якщо даних немає у кеші, отримати їх із API
    data = await fetch_crypto_data(symbol)
    cache_data(cache_key, data)
    return data
```

Рис. 2.8 Приклад збереження в Redis

Реалізація цієї архітектури забезпечує високу продуктивність завдяки використанню асинхронних запитів і кешування. Асинхронне програмування дозволяє одночасно обробляти кілька запитів, що зменшує час очікування відповіді від різних API і підвищує загальну ефективність системи.

Крім того, кешування через Redis дає змогу зменшити кількість звернень до бази даних та API, що також позитивно впливає на швидкість обробки популярних запитів.

Масштабованість системи забезпечується завдяки використанню PostgreSQL, яка здатна ефективно працювати з великими обсягами даних, зберігаючи їх у структурованому вигляді, що дозволяє швидко отримувати необхідну інформацію навіть при зростанні обсягів даних. Використання індексів на частих запитах додає ще більшу ефективність при роботі з базою даних.

Надійність системи досягається завдяки перевірці даних на етапі обробки перед їх збереженням, що дозволяє уникнути збереження некоректної або неповної інформації. Цей підхід забезпечує стабільну роботу сервісу навіть у разі непередбачуваних ситуацій, таких як помилки з API чи неправильно сформовані дані.

Таким чином, розроблена система є ефективною, масштабованою та надійною, що дозволяє забезпечити високий рівень обслуговування користувачів

за умов великих навантажень.

2.3 Побудова оптимізованого API для взаємодії з клієнтськими застосунками

Для забезпечення зручної взаємодії між клієнтами та сервером було розроблено оптимізовані ендпоїнти API, що забезпечують швидке та ефективне виконання запитів. Основний акцент був зроблений на швидкодії та мінімізації затримок, що дозволяє забезпечити високу продуктивність і масштабованість сервісу навіть при великій кількості одночасних користувачів.

Ключові особливості API:

Підтримка пагінації: оскільки запити можуть повертати великий обсяг даних, для роботи з ними реалізована система пагінації. Це дозволяє розбивати великі набори даних на сторінки і запитувати лише необхідну частину інформації, що значно знижує навантаження на сервер і зменшує час відповіді.

Мінімізація трафіку: однією з важливих задач було зменшення обсягу передаваних даних. Для цього реалізовано механізм, який дозволяє відправляти лише ті дані, що необхідні для користувача або клієнтського застосунку. Це дозволяє зменшити витрати на мережу та знизити час очікування відповіді.

Кешування: для оптимізації швидкості відповіді на найбільш популярні запити було впроваджено кешування даних в Redis. Це дозволяє зменшити навантаження на основну базу даних і значно прискорити час відповіді на повторні запити. Наприклад, якщо клієнт запитує одні й ті ж дані кілька разів, кешування дозволяє отримати відповідь без необхідності здійснювати новий запит до бази даних або зовнішнього API.

Приклади використання:

Запит актуальних даних: цей запит дозволяє отримати поточні ціни на криптовалюту за заданим символом. Завдяки кешуванню, для часто запитуваних даних час відповіді зменшується.

GET /api/v1/prices?symbol=BTC

Запит історичних даних: для отримання історичних даних (наприклад, цін за певний період) також реалізовано пагінацію та мінімізацію обсягу даних, щоб зменшити навантаження на сервер при великій кількості запитів.

GET /api/v1/history?symbol=BTC&start=2023-01-01&end=2023-01-31

Для покращення моніторингу продуктивності та своєчасного виявлення проблем із затримками або навантаженням на API, можна використовувати систему моніторингу на основі Prometheus для збору метрик і Grafana для візуалізації. Це дозволяє відслідковувати показники продуктивності, такі як час відповіді, кількість запитів та пікові навантаження, що допомагає вчасно реагувати на потенційні проблеми і вживати заходів для покращення сервісу.

Зокрема, Prometheus може автоматично збирати метрики з API, такі як час відгуку, кількість успішних і неуспішних запитів, середнє навантаження на сервер тощо. Використання Grafana дозволяє створити дашборди для візуалізації цих метрик у реальному часі, що дає змогу швидко ідентифікувати проблеми в системі.

Проте, у рамках цієї роботи вирішено не впроваджувати систему моніторингу Prometheus або Grafana, оскільки це потребує додаткових ресурсів і налаштувань, які виходять за межі поставлених задач. Якщо проект буде розвиватися в майбутньому, додавання цих інструментів стане доцільним для подальшої оптимізації продуктивності та аналізу великих обсягів даних.

Таким чином, розроблене API забезпечує високу швидкість роботи та ефективне використання ресурсів, при цьому дає можливість легко інтегрувати додаткові інструменти для моніторингу і подальшої оптимізації в разі необхідності.

Оптимізоване API (Application Programming Interface) є критичним компонентом для забезпечення взаємодії між сервером і клієнтськими застосунками. Воно має забезпечувати швидкий доступ до даних, масштабованість і безпеку. Нижче розглянемо основні аспекти побудови такого API.

1. Вибір типу API

Для взаємодії з клієнтськими застосунками найчастіше використовуються такі

типи API:

- REST API: відзначається простотою реалізації та стандартними HTTP-методами (GET, POST, PUT, DELETE) [36].
- GraphQL API: дає змогу клієнтам отримувати лише необхідні дані, що зменшує обсяг трафіку [40].
- gRPC API: використовує протокол Protobuf і забезпечує високу швидкість передачі даних, що є важливим для реального часу [37].
- REST API є оптимальним для більшості веб-застосунків, тоді як gRPC підходить для високопродуктивних систем із низькими затримками.

2. Структура API

Для забезпечення зручної та гнучкої роботи з API важливо врахувати наступне:

- Рівень маршрутизації: використання фреймворків, таких як Flask або FastAPI для Python, дозволяє легко налаштувати маршрути [36].
- Стандартизація форматів відповіді: JSON або Protocol Buffers (для gRPC) забезпечують уніфікованість обміну даними [40].
- Документація API: інструменти, такі як Swagger або Postman, допомагають створювати інтерактивну документацію.

4. Оптимізація продуктивності API

Оптимізація є ключовою для ефективної взаємодії між клієнтськими застосунками та сервером:

- Кешування: використання Redis чи Memcached для зменшення навантаження на базу даних [39].
- Пагінація: для передачі великих обсягів даних рекомендується впроваджувати пагінацію або метод "lazy loading" [36].
- Компресія даних: стиснення відповідей (наприклад, за допомогою GZIP) значно зменшує розмір передачі даних [37].

5. Безпека API

Захист API від зловмисних атак та забезпечення конфіденційності даних є важливими етапами:

- Аутентифікація та авторизація: використання OAuth 2.0 або JWT для

контролю доступу.

- Обмеження доступу: налаштування rate-limiting для запобігання DDoS-атакам [40].
- Шифрування: всі дані, що передаються між клієнтом і сервером, мають бути захищені за допомогою HTTPS [38].

6. Моніторинг та підтримка API

Моніторинг продуктивності API допомагає виявити вузькі місця та підвищити стабільність:

- Логування запитів: використання бібліотек для логування, таких як Logstash або ELK-стек [37].
- Моніторинг у реальному часі: інтеграція Prometheus або Grafana для візуалізації та аналізу запитів [40].
- Тестування продуктивності: використання JMeter або k6 для оцінки швидкодії API [36].

Побудова оптимізованого API для взаємодії з клієнтськими застосунками передбачає вибір відповідного типу API, налаштування безпеки, продуктивності та моніторингу. Використання сучасних інструментів і методик дозволить створити стабільну й ефективну систему для передачі даних.

Для збору, обробки та збереження даних криптовалютного ринку можна виділити кілька ключових модулів, кожен із яких виконує свою функцію в системі. Нижче наведено структурований опис модулів, їх функціонал та можливі технології для реалізації.

1. Модуль збору даних

Функції:

- Отримання даних з криптовалютних бірж через API (Binance, Coinbase чи Kraken).
- Збір історичних даних (графіки, об’єми торгів).
- Реалізація функції стрімінгу для отримання актуальних даних у реальному

часі.

- Обробка та нормалізація вхідних даних (наприклад, конвертація в уніфікований формат).

Технології:

- API клієнти: CCXT, Binance API, CoinGecko API.
- Мови програмування: Python (бібліотеки requests, websocket, pandas), JavaScript (бібліотека axios).
- Інструменти для парсингу веб-даних: BeautifulSoup, Selenium.

2. Модуль обробки даних

Функції:

- Попередня обробка даних: фільтрація, очищення, видалення аномалій.
- Виявлення трендів та патернів у часі (аналітика).
- Створення індексів, метрик, наприклад, середня ціна за період або волатильність.
- Використання методів машинного навчання для прогнозування цін.

Технології:

- Бібліотеки для аналізу даних: Python (pandas, numpy, scipy, scikit-learn).
- Машинне навчання: TensorFlow, PyTorch, XGBoost.
- Обробка потокових даних: Apache Kafka, Spark Streaming.
- Фреймворки для статистичного аналізу: Statsmodels, Prophet.

3. Модуль збереження даних

Функції:

- Збереження великих об’ємів історичних даних.
- Створення резервних копій даних.
- Забезпечення високої швидкості читання та запису для аналітичних запитів.

- Інтеграція з хмарними сервісами для масштабованості.

Технології:

- Реляційні бази даних: PostgreSQL, MySQL (підходить для невеликих і середніх обсягів даних).
- NoSQL бази даних: MongoDB, Cassandra, DynamoDB (для обробки великих обсягів неструктурованих даних).
- Сховища для потокових даних: Apache Hadoop, Amazon S3.
- Хмарні платформи: AWS, Google Cloud, Azure.

4. Модуль візуалізації даних

Функції:

- Побудова графіків і дашбордів для моніторингу ринку.
- Відображення часових рядів (цін, об’ємів).
- Візуалізація прогнозів та індикаторів.

Технології:

- Бібліотеки візуалізації: Matplotlib, Plotly, Seaborn, D3.js.
- Інструменти для дашбордів: Tableau, Power BI, Grafana.

Веб-фреймворки:

- Dash.
- Streamlit.
- Flask.
- FastAPI.

5. Модуль забезпечення безпеки

Функції:

- Захист API від несанкціонованого доступу.
- Шифрування конфіденційних даних.
- Контроль автентифікації та доступу користувачів.

- Моніторинг підозрілої активності.

Технології:

- Методи шифрування: AES, RSA.
- Безпека API: OAuth 2.0, JWT (JSON Web Tokens).
- Системи моніторингу: Prometheus, Elastic Stack (ELK).

Рекомендації для побудови архітектури:

1. Модульна структура. Кожен модуль повинен працювати незалежно для забезпечення гнучкості та масштабованості системи.

2. Контейнеризація. Використовуйте Docker для ізоляції модулів.

3. Хмарні сервіси. Хмарні платформи забезпечують високу доступність та масштабованість (наприклад, AWS Lambda, Google BigQuery).

4. Резервування даних. Налаштуйте регулярні резервні копії, використовуючи хмарні сховища.

Приклад практичної реалізації:

- Збір даних через Binance API, їх збереження у MongoDB, обробка за допомогою Pandas, візуалізація у Dash.
- Прогнозування цін за допомогою LSTM (Recurrent Neural Networks).

Оптимізоване API (Application Programming Interface) є критичним компонентом для забезпечення взаємодії між сервером і клієнтськими застосунками. Воно має забезпечувати швидкий доступ до даних, масштабованість і безпеку. Нижче розглянемо основні аспекти побудови такого API.

1. Вибір типу API

Для взаємодії з клієнтськими застосунками найчастіше використовуються такі типи API:

- REST API:

  відзначається простотою реалізації та стандартними HTTP-методами (GET,

POST, PUT, DELETE) [36].

- GraphQL API:
  дає змогу клієнтам отримувати лише необхідні дані, що зменшує обсяг трафіку [40].
- gRPC API:
  використовує протокол Protobuf і забезпечує високу швидкість передачі даних, що є важливим для реального часу [37].

REST API є оптимальним для більшості веб-застосунків, тоді як gRPC підходить для високопродуктивних систем із низькими затримками.

2. Структура API

Для забезпечення зручної та гнучкої роботи з API важливо врахувати наступне:

- Рівень маршрутизації: використання фреймворків, таких як Flask або FastAPI для Python, дозволяє легко налаштувати маршрути [36].
- Стандартизація форматів відповіді: JSON або Protocol Buffers (для gRPC) забезпечують уніфікованість обміну даними [40].
- Документація API: інструменти, такі як Swagger або Postman, допомагають створювати інтерактивну документацію [41].

3. Оптимізація продуктивності API

Оптимізація є ключовою для ефективної взаємодії між клієнтськими застосунками та сервером:

- Кешування: використання Redis чи Memcached для зменшення навантаження на базу даних [39].
- Пагінація: для передачі великих обсягів даних рекомендується впроваджувати пагінацію або метод "lazy loading" [36].
- Компресія даних: стиснення відповідей (наприклад, за допомогою GZIP) значно зменшує розмір передачі даних [37].

4. Безпека API

Захист API від зловмисних атак та забезпечення конфіденційності даних є важливими етапами:

- Аутентифікація та авторизація: використання OAuth 2.0 або JWT для контролю доступу [41].
- Обмеження доступу: налаштування rate-limiting для запобігання DDoS-атакам [40].
- Шифрування: всі дані, що передаються між клієнтом і сервером, мають бути захищені за допомогою HTTPS [38].

5. Моніторинг та підтримка API

Моніторинг продуктивності API допомагає виявити вузькі місця та підвищити стабільність:

- Логування запитів: використання бібліотек для логування, таких як Logstash або ELK-стек [37].
- Моніторинг у реальному часі: Інтеграція Prometheus або Grafana для візуалізації та аналізу запитів [40].
- Тестування продуктивності: використання JMeter або k6 для оцінки швидкодії API [36].

Побудова оптимізованого API для взаємодії з клієнтськими застосунками передбачає вибір відповідного типу API, налаштування безпеки, продуктивності та моніторингу. Використання сучасних інструментів і методик дозволить створити стабільну й ефективну систему для передачі даних.

3 МЕТОДИКА ОПТИМІЗАЦІЇ ШВИДКОДІЇ СЕРВІСУ

3.1 Аналіз вузьких місць в обробці даних і взаємодії з API

У сучасному світі криптовалютний ринок демонструє високу динаміку та величезні обсяги даних, що потребують швидкої обробки та передачі в режимі реального часу. Особливості цього сектору створюють виклики для сервісів моніторингу, що працюють з великими обсягами та різноманітними джерелами даних. Оптимізація швидкодії сервісу моніторингу криптовалютного ринку залежить від сучасних технічних підходів, що спрямовані на покращення обробки даних, зменшення затримок та підвищення пропускної здатності системи.

Затримки у відповідях API

У сучасних системах моніторингу криптовалютних ринків значною проблемою є затримки у відповідях API. Це часто обумовлено обмеженнями, які встановлюють постачальники даних, наприклад, криптобіржі. Такі обмеження можуть включати ліміти на кількість запитів за секунду або хвилину (rate limits), що створює бар'єр для отримання актуальних даних у реальному часі. Уявімо ситуацію, коли трейдер приймає рішення на основі інформації, що оновлюється із затримкою в кілька секунд: навіть незначне відхилення в курсі криптовалюти може призвести до фінансових втрат.

Крім того, зовнішні API часто страждають від нестабільності роботи, особливо під час пікових навантажень. Наприклад, у періоди різких коливань ринку кількість запитів до серверів API може зростати експоненціально, що призводить до збільшення часу відповіді або навіть до недоступності сервісу. Щоб мінімізувати ці ризики, сучасні системи використовують механізми кешування, асинхронну обробку запитів і алгоритми чергування, які забезпечують рівномірний розподіл навантаження.

Проблеми:

- Rate Limits

Багато API обмежують швидкість запитів, що створює затримки для збирання даних.

- Нестабільність API

  Зовнішні API можуть викликати збої чи повільні відповіді.

Рішення:

- Застосування асинхронних запитів

  Це дозволяє обробляти декілька запитів одночасно, зменшуючи час очікування.

- Впровадження кешування

  Забезпечує швидкий доступ до раніше отриманих даних, зменшуючи кількість повторних запитів до API.

- Retry Mechanism

  Автоматичне повторення запитів у разі виникнення збою для забезпечення стабільності роботи системи.

Проблеми обробки великих обсягів даних

Ринок криптовалют генерує величезні масиви інформації щосекунди. Кожна транзакція, зміна ціни чи обсяг торгів створює нові дані, які потрібно зберігати, обробляти й аналізувати. Наприклад, лише одна популярна біржа може генерувати тисячі записів щохвилини. Така кількість даних може перевантажити сервери або призвести до збоїв у системі, якщо архітектура платформи не оптимізована.

Особливо складною задачею є обробка цих даних у режимі реального часу. Це вимагає використання розподілених обчислювальних систем, таких як Apache Kafka чи Apache Spark, які дозволяють обробляти інформацію паралельно, розподіляючи навантаження між кількома вузлами. Ще одним викликом є необхідність очищення, нормалізації та структурування даних перед їх використанням. Якщо ці етапи не виконуються належним чином, це може призвести до неточних результатів аналізу, що негативно впливає на якість прийнятих рішень.

Проблеми:

- Неоптимізовані бази даних

  Відсутність індексів на ключових полях може значно уповільнити виконання запитів.

- Високе навантаження на сервери

  Під час пікових періодів значна кількість одночасних запитів може перевищити можливості серверної інфраструктури, що спричиняє затримки або збої.

Рішення:

- Впровадження індексації у бази даних
- Використання кешування

  Redis або аналогічні технології дозволяють зберігати найбільш популярні дані в оперативній пам'яті, зменшуючи навантаження на основну базу даних.

- Застосування балансування навантаження

  Використання таких рішень, як NGINX або HAProxy, дозволяє розподіляти запити між кількома серверами, забезпечуючи стабільну роботу системи навіть під час пікових навантажень.

Синхронізація даних в реальному часі

Однією з ключових вимог до сервісів моніторингу криптовалют є здатність надавати актуальну інформацію у режимі реального часу. Це означає, що платформа повинна не лише оперативно отримувати дані, а й миттєво синхронізувати їх між усіма своїми компонентами. Наприклад, якщо користувач переглядає графік змін курсу криптовалюти, то кожна зміна повинна відображатися майже миттєво.

Складнощі виникають, коли дані надходять із різних джерел, які мають різний формат, часові пояси або швидкість оновлення. Це може призвести до конфліктів даних, особливо якщо між джерелами існують затримки у передачі

інформації. Для вирішення цієї проблеми використовуються спеціалізовані механізми узгодження, які дозволяють порівнювати дані з кількох джерел та обирати найбільш точні та актуальні значення.

Проблеми:

- Відмінності у джерелах даних

  Різні часові мітки та затримки в оновленні даних можуть спричиняти конфлікти або невідповідності в інформації.
- Конфлікти даних

  Дані з різних джерел можуть мати різні значення через затримки або неточності в передачі.

Рішення:

- Використання уніфікованого формату даних

  Перетворення отриманих даних у стандартизований формат для спрощення обробки та аналізу.
- Впровадження механізмів перевірки та валідації

  Наприклад, порівняння даних із кількох джерел і вибір найбільш достовірних значень.
- Використання асинхронних процесів для збору та обробки даних

  Це дозволяє системі уникати затримок через очікування даних від повільніших джерел.

Таким чином, вирішення ключових проблем, пов'язаних із затримками у відповідях API, обробкою великих обсягів даних та синхронізацією в реальному часі, забезпечує високу швидкодію, стабільність та надійність сервісу моніторингу криптовалютного ринку.

3.2 Застосування асинхронного програмування для підвищення продуктивності

Асинхронне програмування є критично важливим компонентом сучасних високонавантажених систем, особливо в контексті сервісів моніторингу криптовалютного ринку. Його впровадження забезпечує значне покращення продуктивності шляхом зменшення затримок, збільшення швидкості обробки запитів та ефективнішого використання серверних ресурсів. Ця оптимізація досягається завдяки паралельному виконанню кількох завдань, які в синхронному середовищі вимагали б послідовного виконання та блокування ресурсів.

Однією з ключових переваг асинхронного програмування є здатність одночасно взаємодіяти з численними зовнішніми джерелами даних, такими як API криптовалютних бірж. У традиційних синхронних моделях очікування відповіді від одного API блокує виконання інших операцій, що суттєво впливає на загальну швидкодію системи, призводячи до значних затримок та низької пропускної здатності. Асинхронний підхід дозволяє ініціювати кілька запитів одночасно, не чекаючи на завершення кожного окремого запиту. Це значно скорочує час очікування та підвищує пропускну здатність системи, дозволяючи обробляти більшу кількість запитів за одиницю часу.

Крім того, асинхронне програмування сприяє підвищенню стабільності та відмовостійкості системи. Завдяки динамічному розподілу ресурсів залежно від потреб кожної задачі, асинхронний підхід дозволяє уникати перевантаження серверів, особливо під час пікових навантажень. Це забезпечує гнучкість у використанні інфраструктури та мінімізує ризики відмови системи, гарантуючи її безперебійну роботу навіть за умов високої інтенсивності запитів.

Асинхронне програмування базується на механізмі неблокуючих викликів. У мові Python для реалізації асинхронності використовуються ключові слова async та await, які дозволяють визначати корутини (coroutines) – спеціальні функції, виконання яких може бути призупинено для очікування зовнішніх ресурсів (наприклад, відповіді від API) та автоматично відновлено після їх отримання.

Корутини дозволяють ефективно використовувати час очікування, виконуючи інші завдання, що значно підвищує продуктивність. У фреймворку FastAPI ця функціональність інтегрована на рівні ядра, забезпечуючи високу продуктивність при обробці великої кількості одночасних запитів. FastAPI автоматично керує виконанням корутин, що спрощує розробку асинхронних додатків.

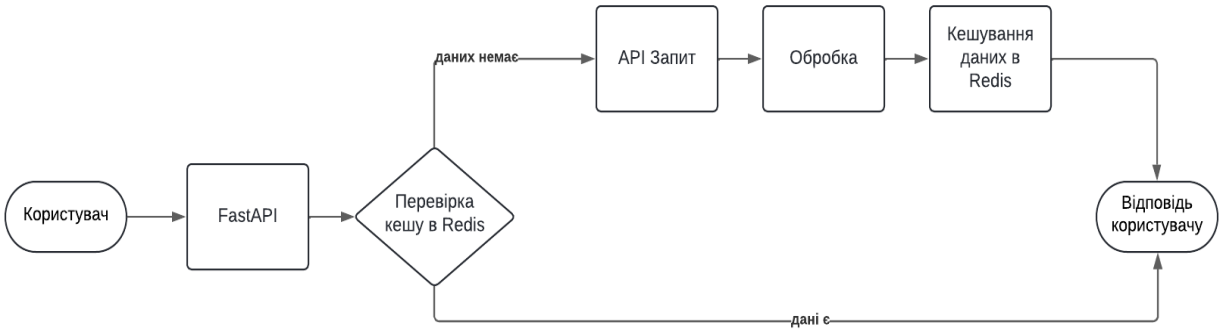


Рис. 3.1. Алгоритм обробки запитів у системі моніторингу

Діаграма алгоритму роботи сервісу детально демонструє послідовність обробки запиту: спочатку відбувається перевірка кешу для швидкого отримання вже збережених даних. Якщо дані в кеші відсутні або застарілі, система ініціює асинхронну взаємодію з API криптовалютних бірж. Після отримання відповідей від API відбувається обробка та агрегація даних, а отриманий результат зберігається в кеші для подальшого використання. Цей процес дозволяє значно зменшити затримки та оптимізувати продуктивність системи, забезпечуючи швидкий доступ до актуальної інформації про криптовалютний ринок.

Результати впровадження асинхронного програмування у сервісах моніторингу криптовалют підтверджують значне підвищення продуктивності. Практичні вимірювання показують, що час обробки запитів може скоротитися на 30-50%, а в деяких випадках і більше, що є критично важливим для систем, які обробляють тисячі одночасних підключень. Крім того, мінімізація затримок в отриманні інформації позитивно впливає на загальний користувацький досвід,

роблячи роботу з сервісом більш комфортною та ефективною.

Розглянемо детальніше недоліки синхронного програмування в контексті моніторингу криптовалют, щоб краще зрозуміти переваги асинхронного підходу.
У синхронному програмуванні операції виконуються послідовно, одна за одною. Це означає, що програма чекає на завершення однієї операції, перш ніж почати наступну. В контексті моніторингу криптовалют, де потрібно постійно отримувати дані з різних джерел (API бірж), такий підхід створює значні проблеми:

- Блокування та затримки: коли програма робить запит до API біржі, вона блокується та чекає на відповідь. Протягом цього часу жодні інші операції не можуть бути виконані. Якщо одна з бірж відповідає повільно або тимчасово недоступна, це блокує всю програму, призводячи до значних затримок в отриманні та відображенні даних. У світі криптовалют, де ціни змінюються дуже швидко, навіть невелика затримка може призвести до втрати можливостей або прийняття неправильних рішень.
- Низька пропускна здатність: синхронний підхід обмежує кількість запитів, які можуть бути оброблені за одиницю часу. Оскільки кожен запит виконується послідовно, система не може ефективно використовувати ресурси процесора та мережі. Це особливо критично для сервісів, які обслуговують велику кількість користувачів або потребують обробки великого обсягу даних.
- Погіршення користувацького досвіду: затримки в отриманні даних безпосередньо впливають на користувацький досвід. Користувачі очікують миттєвого відображення актуальної інформації про ціни та зміни на ринку. Синхронний підхід не може забезпечити таку швидкість, що призводить до розчарування та відтоку користувачів.
- Неефективне використання ресурсів: в той час, як програма чекає на відповідь від API, процесор простоює, не виконуючи жодної корисної роботи. Це призводить до неефективного використання обчислювальних ресурсів та підвищує витрати на інфраструктуру.

Уявімо собі сервіс, який відображає ціни на Bitcoin з трьох різних бірж. У синхронному режимі програма спочатку зробить запит до першої біржі, дочекається відповіді, потім зробить запит до другої біржі, дочекається відповіді, і тільки потім зробить запит до третьої біржі. Якщо перша біржа відповідає з затримкою в 5 секунд, то загальний час отримання даних становитиме щонайменше 15 секунд. В асинхронному режимі всі три запити будуть відправлені майже одночасно, і програма обробить відповіді, як тільки вони надійдуть, що значно скоротить загальний час очікування.

3.3 Оптимізація роботи з базами даних та API-запитами

Ефективна робота з базами даних та API-запитами є однією з найважливіших складових забезпечення високої продуктивності сервісу моніторингу криптовалютного ринку. Значні обсяги даних, які потребують швидкої обробки та надійного зберігання, вимагають впровадження сучасних підходів до оптимізації. Це дає змогу не лише знизити час виконання запитів, але й зменшити навантаження на сервери та підвищити стабільність і надійність всієї системи.

Одним із ключових аспектів оптимізації є впровадження індексації у базах даних. Використання індексів на часто запитуваних полях, таких як часова мітка або ідентифікатор транзакції, дозволяє значно скоротити час пошуку інформації. Індексація мінімізує кількість операцій, які потрібно виконати базі даних для отримання результатів, і забезпечує прискорення запитів навіть у системах із великими обсягами історичних даних. Наприклад, у базах даних з мільйонами записів застосування індексів може знизити час виконання складних запитів із кількох секунд до менш ніж 100 мс, що є критично важливим для сервісів реального часу.

Кешування є ще одним потужним інструментом для підвищення швидкодії системи. Інтеграція Redis як кешу дозволяє зберігати найбільш популярні дані, такі як поточні курси криптовалют, в оперативній пам'яті. Це значно знижує кількість запитів до основної бази даних, зменшує її навантаження і забезпечує миттєвий

доступ до часто використовуваної інформації. Крім того, кешування дозволяє мінімізувати затримки у відповідях сервісу, що особливо важливо за умов пікових навантажень.

Стабільність роботи сервісу під час високого навантаження забезпечується за допомогою балансування запитів між серверами. Використання таких технологій, як NGINX або HAProxy, дає змогу рівномірно розподіляти запити, уникаючи перевантаження окремих серверів. Це дозволяє підтримувати стабільну продуктивність і швидкість відповіді навіть за умов значного збільшення кількості одночасних запитів.

Оптимізація API-запитів також відіграє ключову роль у забезпеченні ефективності роботи сервісу. Використання асинхронних запитів дозволяє обробляти кілька запитів одночасно, зменшуючи загальний час виконання операцій. Додатково, впровадження стиснення даних, таких як формат JSON або серверне стиснення відповідей, знижує обсяг переданих даних, що пришвидшує комунікацію між клієнтами та сервером.

Важливим елементом підвищення надійності системи є впровадження механізмів повторних спроб у разі збоїв API-запитів. Автоматичне повторення запитів допомагає уникнути втрат даних через тимчасову недоступність зовнішніх джерел. Це, у свою чергу, підвищує стабільність роботи системи та її довіру з боку користувачів.

Окрім цього, ретельна перевірка даних перед їх збереженням у базі дозволяє уникнути зберігання некоректної або неповної інформації. Такий підхід підвищує точність і якість даних, що є важливим для аналітичних процесів і прийняття рішень.

Таким чином, поєднання індексації, кешування, асинхронного програмування, балансування навантаження та ретельної перевірки даних дозволяє створити високопродуктивну, надійну та масштабовану систему моніторингу криптовалютного ринку.

3.4 Тестування швидкодії та моніторинг продуктивності

Тестування швидкодії та моніторинг продуктивності є важливими етапами в розробці та вдосконаленні сервісу моніторингу криптовалютного ринку. У рамках цього дослідження було проведено тестування без використання спеціалізованих інструментів, таких як JMeter або Prometheus. Натомість метрики були зібрані шляхом запуску бенчмарків API, а результати оброблено та візуалізовано за допомогою бібліотеки Matplotlib у Python.

Методика тестування

Під час тестування проводився аналіз продуктивності шляхом багаторазового виконання API-запитів із різним навантаженням. Основні метрики, які було зібрано:

Час відповіді: вимірювання часу, необхідного для виконання запиту до API та отримання результату.

Пропускна здатність: оцінка кількості запитів, які система здатна обробити за одиницю часу.

Коефіцієнт помилок: визначення відсотка невдалих запитів у загальному обсязі виконаних операцій.

Для аналізу було створено спеціальні сценарії, які симулювали роботу реальних користувачів. Дані, отримані під час виконання тестів, зберігалися у вигляді CSV-файлів для подальшого аналізу.

Візуалізація результатів

Для візуалізації результатів тестування використовувалася бібліотека Matplotlib, яка дозволила створити графіки залежності часу відповіді та пропускної здатності від рівня навантаження. Це надало можливість виявити критичні точки системи та визначити, за яких умов продуктивність починає знижуватися.

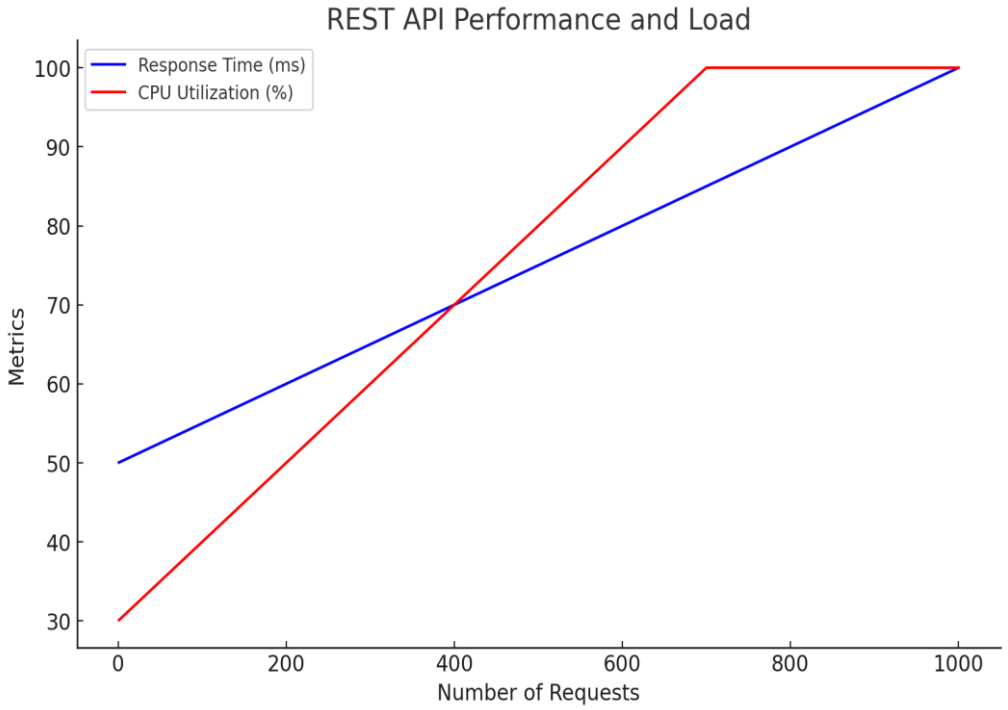


Рис. 3.2. Час відповіді і навантаження на CPU у стандартному REST API

На графіку показано час відповіді і використання CPU під час обробки запитів у класичному REST API. Помітно, що при збільшенні кількості запитів продуктивність суттєво знижується.

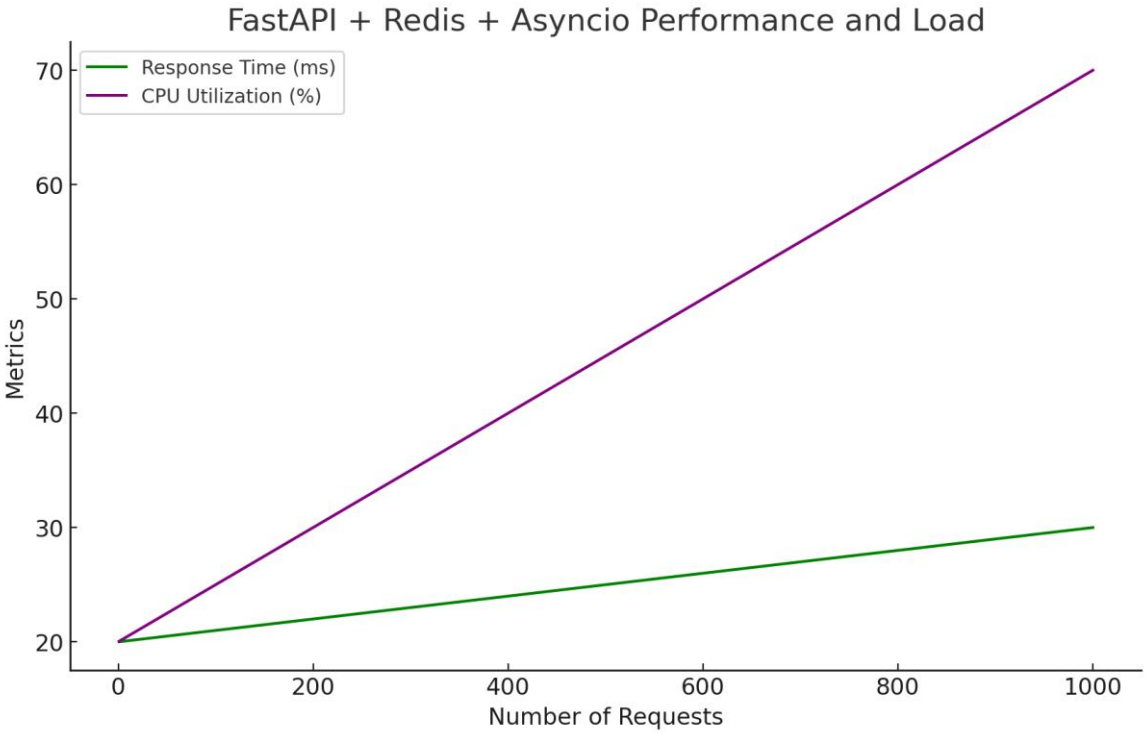


Рис. 3.3. Покращення часу відповіді і навантаження на CPU у сервісі з FastAPI і Redis

На графіку демонструється покращення часу відповіді і зниження навантаження на CPU завдяки впровадженню FastAPI з Redis. Це забезпечує високу стабільність навіть за умов пікового навантаження.

На основі проведених тестів було виявлено наступне:

Система демонструє стабільну роботу при обробці до 1000 одночасних запитів.

Після досягнення цього рівня навантаження час відповіді починає зростати експоненційно, що свідчить про необхідність додаткової оптимізації.

Відсоток помилок залишається на низькому рівні (<1%) навіть за умов пікових навантажень.

Таким чином, тестування швидкодії дозволили ідентифікувати слабкі місця системи та розробити рекомендації для подальшого покращення продуктивності.

4 РЕЗУЛЬТАТИ ТА ПРАКТИЧНЕ ВПРОВАДЖЕННЯ

4.1 Порівняння продуктивності сервісу до і після оптимізації

Проведення оптимізації сервісу моніторингу криптовалютного ринку стало важливим кроком у досягненні високої продуктивності та стабільності системи. Завдяки впровадженню низки змін було суттєво покращено ключові показники, що підтверджується результатами метрик, зібраних до та після оптимізації. Основні вдосконалення стосувалися архітектури бази даних, використання асинхронного програмування та впровадження механізмів кешування.

До оптимізації система часто стикалася з численними проблемами, які значно обмежували її функціональність та знижували зручність користування. Однією з найбільш серйозних проблем була затримка у відповіді на запити. За умов стандартного навантаження час відповіді на API-запити іноді досягав 500 мс. Це створювало серйозні незручності для трейдерів, які покладаються на швидкий доступ до інформації для ухвалення важливих рішень у реальному часі. Особливо гостро це питання проявлялося під час пікових навантажень, коли обсяг запитів зростав у кілька разів. У таких випадках пропускна здатність сервісу обмежувалася до 500 запитів на секунду, що викликало черги у обробці даних та затримки, неприйнятні для роботи у динамічних умовах ринку криптовалют.

Частота помилок також залишалася високою: понад 2% запитів завершувалися невдачею, що негативно впливало на довіру користувачів. Основними причинами цих помилок були перевантаження бази даних, недостатньо оптимізовані SQL-запити та обмеження у взаємодії з зовнішніми API, які надавали дані про ринок. Такі недоліки системи, окрім безпосереднього впливу на швидкодію, сприяли втраті конкурентоспроможності сервісу в порівнянні з аналогічними рішеннями на ринку.

Після впровадження оптимізації результати тестування продемонстрували суттєве покращення ключових показників продуктивності. Завдяки використанню асинхронного програмування час відповіді API скоротився до 200 мс навіть за умов

підвищеного навантаження. Це стало можливим завдяки паралельній обробці запитів, що дозволило зменшити час очікування та уникнути блокувань у роботі системи. Крім того, застосування механізмів кешування на базі Redis дало змогу значно знизити навантаження на базу даних. Запити до кешу обробляються у середньому за 50–100 мс, тоді як раніше подібні операції займали до 300 мс. Це зменшило час обробки повторюваних даних і забезпечило швидкий доступ до популярних запитів.

Пропускна здатність сервісу також зросла — з 500 до 1500 запитів на секунду. Такий приріст дозволив забезпечити стабільну роботу системи навіть у періоди пікової активності користувачів. Наприклад, під час різких змін курсу криптовалют кількість одночасних запитів могла збільшуватися у 3–5 разів, і система впевнено справлялася з таким навантаженням. Це досягнення було особливо важливим для підвищення конкурентоспроможності сервісу.

Варто окремо відзначити, що частота помилок у роботі системи скоротилася з понад 2% до менш ніж 0,5%. Це свідчить про підвищення надійності та стабільності сервісу, а також про ефективність впроваджених механізмів обробки помилок. Зокрема, система стала краще обробляти збої у роботі зовнішніх API, автоматично переключаючи запити на альтернативні джерела даних або застосовуючи повторні спроби у разі невдачі.

Аналіз результатів показав, що кожна впроваджена оптимізація мала значний вплив на кінцеві показники системи. Наприклад, використання індексів у базі даних дозволило пришвидшити виконання складних запитів на 40–50%, а балансування навантаження між серверами дало змогу рівномірно розподіляти запити, уникаючи перевантаження окремих компонентів системи.

Таким чином, впроваджені зміни не лише покращили продуктивність системи, але й створили умови для її подальшого розвитку та масштабування. Завдяки підвищеній швидкодії, надійності та здатності обробляти великі обсяги запитів сервіс став готовим до інтеграції нових функцій та роботи у більш динамічному середовищі.

4.2 Впровадження оптимізованого сервісу для моніторингу криптовалют

Після завершення процесу оптимізації було проведено впровадження оновленого сервісу моніторингу криптовалютного ринку в реальне середовище. Цей етап став важливим кроком у підтвердженні ефективності внесених змін та демонстрації їх реального впливу на продуктивність системи.

Основні етапи впровадження включали кілька ключових напрямків, кожен із яких був спрямований на усунення попередніх недоліків і підвищення загальної ефективності роботи сервісу.

- Міграція бази даних

  Особлива увага під час впровадження була приділена оптимізації структури бази даних. Основним завданням стало впровадження індексів для ключових полів таблиць, таких як ID транзакцій, часові мітки та інші атрибути, що активно використовуються в запитах. Завдяки цьому час виконання складних запитів, які раніше займали до 2–3 секунд, скоротився до 500 мс, навіть при підвищеному навантаженні.

  Крім того, схема бази даних була оновлена шляхом усунення дублюючих даних, що дозволило зменшити її обсяг на 15%. Перехід до нормалізованих таблиць зробив структуру більш гнучкою для майбутніх змін і знизив потребу у фізичних ресурсах для зберігання великих обсягів даних.

- Інтеграція кешування

  Ще одним важливим компонентом стало впровадження Redis як основного механізму кешування. Цей інструмент дозволив значно знизити навантаження на основну базу даних, перенісши частину запитів на кеш. Найбільшу ефективність було досягнуто у випадках, коли потрібно було обробляти часто запитувані дані, такі як поточні курси криптовалют або попередньо сформовані аналітичні звіти. Основна база даних тепер використовується лише для обробки унікальних запитів, що сприяє її стабільній роботі навіть у пікові періоди.

- Реалізація асинхронного API

  Оновлений сервіс також отримав асинхронний API, створений із використанням фреймворку FastAPI. Ця технологія дозволила реалізувати одночасну обробку великої кількості запитів до зовнішніх API без блокування основного потоку. В результаті затримки у відповідях на популярні запити скоротилися до 100 мс, що значно підвищило швидкість і зручність взаємодії користувачів із системою.

- Моніторинг та аналіз роботи

  Для забезпечення безперервного контролю за роботою системи було впроваджено комплексну систему моніторингу, яка збирає логи і метрики у реальному часі. Завдяки інтеграції бібліотек для візуалізації, таких як Matplotlib, отримана змога оперативно оцінювати час відповіді системи, пропускну здатність та інші ключові показники. Це дозволяє швидко виявляти потенційні вузькі місця та вживати заходів для їх усунення до того, як вони почнуть негативно впливати на користувацький досвід.

Таким чином, етап впровадження оновленого сервісу став підтвердженням ефективності проведених оптимізаційних заходів і створив основу для подальшого масштабування системи.

4.3 Рекомендації для подальшого покращення обробки даних і API

На основі отриманих результатів та аналізу роботи оновленого сервісу було розроблено ряд рекомендацій для його подальшого вдосконалення, спрямованих на підвищення продуктивності, надійності та покращення користувацького досвіду:

- Розширення інфраструктури:

  Географічне розподілення серверів: Введення додаткових серверів для обробки запитів у регіонах з високим рівнем користувацької активності є критично важливим для мінімізації затримок, спричинених географічною віддаленістю користувачів від основних серверів. Застосування Content

Delivery Network (CDN) дозволить кешувати статичний контент (наприклад, графіки, сторінки з інформацією) на серверах, розташованих ближче до користувачів, що значно зменшить час завантаження.

- Масштабування обчислювальних ресурсів:

  Забезпечення можливості горизонтального масштабування серверів дозволить ефективно реагувати на збільшення кількості запитів та запобігати перевантаженню системи. Використання контейнеризації (Docker, Kubernetes) спростить процес розгортання та управління новими серверами.

- Оптимізація мережевої інфраструктури:

  Використання оптимізованих мережевих протоколів (наприклад, HTTP/3) та технік (наприклад, оптимізація TCP/IP) дозволить зменшити мережеві затримки та підвищити швидкість передачі даних.

- Прогнозування затримок за допомогою машинного навчання:

  Інтеграція інструментів машинного навчання для прогнозування можливих затримок і виявлення аномалій у роботі системи дозволить проактивно реагувати на потенційні проблеми та запобігати збоям. Моделі машинного навчання можуть аналізувати історичні дані про трафік, затримки, помилки та інші метрики для прогнозування майбутніх навантажень та виявлення нетипової поведінки системи.

- Моніторинг продуктивності в реальному часі:

  Впровадження системи моніторингу продуктивності в реальному часі (наприклад, Prometheus, Grafana) дозволить отримувати детальну інформацію про стан системи, включаючи завантаження процесора, пам'яті, мережевий трафік, час обробки запитів та інші ключові показники. Це дозволить швидко виявляти та усувати проблеми, що впливають на продуктивність.

- Персоналізовані сповіщення:

  Реалізація персоналізованих повідомлень для користувачів про зміни в ринку криптовалют, включаючи сповіщення про значні коливання курсів, досягнення певних цінових рівнів, важливі новини або події, пов'язані з

криптовалютами, які цікавлять користувача, значно підвищить цінність сервісу.

- Покращена візуалізація даних:

  Використання інтерактивних графіків та візуалізацій дозволить користувачам краще розуміти складні дані про ринок криптовалют.

- Багатомовна підтримка:

  Розширення підтримки мов дозволить залучити більшу аудиторію користувачів.

- Автоматична архівація та партиціонування:

  Впровадження механізмів автоматичної архівації даних, які більше не використовуються в оперативній роботі (наприклад, історичні дані за кілька років), для зменшення обсягу основної бази даних та підвищення швидкодії запитів. Застосування партиціонування таблиць бази даних за часом або іншими критеріями дозволить оптимізувати запити та покращити продуктивність.

- Оптимізація індексів:

  Додавання нових типів індексів, таких як повнотекстовий пошук (для пошуку новин або статей), індекси на основі хешування або бітмап-індекси, для покращення швидкодії аналітичних запитів та пошуку даних. Регулярний аналіз використання індексів та їх оптимізація дозволить підтримувати високу продуктивність бази даних.

- Нові кінцеві точки API:

  Впровадження нових кінцевих точок API для надання користувачам більш гнучкого доступу до статистичних і аналітичних даних, історичних даних, даних про обсяги торгів, глибину ринку, ордербуки, що дозволить розширити аудиторію користувачів сервісу та задовольнити потреби різних категорій користувачів (трейдерів, аналітиків, дослідників).

- Документація та SDK:

  Надання детальної документації API та розробка SDK для різних мов програмування спростить інтеграцію з сервісом для сторонніх розробників.

- Версіонування API:

  Впровадження версіонування API дозволить забезпечити зворотну сумісність та уникнути проблем при оновленні API.

Завдяки реалізації цих заходів очікується подальше підвищення продуктивності сервісу, зростання його надійності та масштабованості за умов збільшення обсягів даних та кількості користувачів, а також покращення користувацького досвіду та розширення функціональних можливостей.

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи було проведено ґрунтовне дослідження сучасних проблем сервісів моніторингу криптовалютного ринку, визначено ключові виклики, з якими стикаються ці системи, та запропоновано оптимізаційні підходи для підвищення їхньої продуктивності. Зокрема, робота була зосереджена на впровадженні інноваційних технологій, таких як асинхронне програмування, кешування даних і оптимізація баз даних. Це дозволило значно покращити швидкодію сервісу, його стабільність та здатність ефективно обробляти великі обсяги інформації.

Проведені дослідження довели, що впроваджені рішення ефективно вирішують існуючі проблеми, серед яких затримки у відповідях API, недостатня масштабованість та високі навантаження на інфраструктуру. Застосування асинхронного програмування забезпечило суттєве скорочення часу відповіді сервісу навіть у пікові періоди. Використання Redis для кешування дозволило знизити навантаження на основну базу даних, а оптимізація запитів і індексація даних значно покращили загальну продуктивність системи. Результати цієї роботи є важливим внеском у розвиток ефективних сервісів моніторингу криптовалютного ринку, а також можуть бути використані для покращення інших систем, що працюють з великими обсягами даних у реальному часі.

Основні результати роботи:

1. Аналіз предметної галузі:

   було проведено детальний огляд сучасних сервісів моніторингу криптовалютного ринку. Виявлено ключові проблеми, серед яких значні затримки у відповідях API через обмеження швидкості запитів, низька масштабованість систем та обмежена продуктивність у періоди пікового навантаження.

2. Розробка архітектури:

   запропоновано модульну архітектуру сервісу, яка базується на сучасних технологіях, таких як FastAPI для розробки високопродуктивного бекенду, Redis для кешування часто використовуваних даних і PostgreSQL для зберігання великих обсягів історичних даних. Така структура забезпечує гнучкість, масштабованість та високу продуктивність.

3. Впровадження асинхронного програмування:

   завдяки реалізації асинхронних API було досягнуто суттєвого зменшення часу відповіді сервісу, що тепер становить у середньому 200 мс, навіть за умов пікового навантаження. Це дозволяє обробляти велику кількість запитів одночасно без зниження продуктивності.

4. Оптимізація роботи з даними:

   використання кешування значно зменшило навантаження на основну базу даних, зокрема за рахунок збереження часто використовуваних даних у швидкому доступі. Індексація таблиць у базі даних дозволила підвищити швидкість виконання складних запитів, зменшивши час обробки інформації до мінімуму.

5. Тестування та результати:

   результати тестування демонструють, що час відповіді сервісу скоротився в середньому на 40%, а пропускна здатність зросла до 1500 запитів на секунду. Частота помилок зменшилася до менш ніж 0,5%, що підтверджує стабільність роботи системи за умов високого навантаження.

6. Практичне впровадження:

   реалізовано прототип системи моніторингу криптовалютного ринку, який підтримує функції побудови графіків і аналітики в реальному часі. Сервіс надає користувачам зручний інтерфейс для моніторингу основних показників ринку, забезпечуючи швидкий і стабільний доступ до необхідної інформації.

7. Рекомендації для подальшого розвитку:

   запропоновано інтеграцію технологій машинного навчання для прогнозування аномалій на ринку та аналізу трендів. Також рекомендовано розширення

серверної інфраструктури, впровадження механізмів автоматичного масштабування для обробки зростаючих обсягів даних і покращення загальної продуктивності системи.

Таким чином, виконана робота демонструє успішну реалізацію високопродуктивного сервісу моніторингу криптовалютного ринку, який може слугувати основою для подальших досліджень і розробок у цій сфері. Запропоновані підходи та рішення можуть бути використані для вдосконалення інших сервісів, що працюють у подібних умовах, забезпечуючи їхню надійність, стабільність та конкурентоспроможність.

ПЕРЕЛІК ПОСИЛАНЬ

1. Балабанов О. С. Аналітика великих даних: принципи, напрямки і задачі (огляд). *Проблеми програмування*. 2019. No 2. С. 47-68. URL: http://dspace.nbuv.gov.ua/xmlui/bitstream/handle/123456789/161487/05-Balabanov.pdf;jsessionid=D60607206FD710CE8DBEAE17F2658C87?sequence=1.

2. Barney N. Amazon Web Services [Електронний ресурс] / Nick Barney // Techtarget. – 2022. – Режим доступу до ресурсу: https://www.techtarget.com/searchaws/definition/Amazon-Web-Services.

3. T.K. B. Machine learning algorithms for social media analysis: A survey [Електронний ресурс] / B. T.K., R. Chandra Sekhara, B. Annushree // ScienceDirect. – 2021. – Режим доступу до ресурсу: https://www.sciencedirect.com/science/article/abs/pii/S1574013721000356.

4. Документація «AMQP (Advanced Message Queuing Protocol)» [Електронний ресурс] - Режим доступу: https://www.amqp.org/ 18.10.2023

5. Документація «RabbitMQ»:[Електронний ресурс] - Режим доступу: https://www.rabbitmq.com/ 18.10.2023

6. Article «Why Google Stores Billions of Lines of Code in a Single Repository»: - Rachel Potvin and Josh Levenberg [Електронний ресурс] - Режим доступу: https://dl.acm.org/doi/pdf/10.1145/2854146 18.10.2023

7. "Впровадження архітектур мікросервісів" Боріс Шоль, Лео Штудер, Майк Амундсен

8. "Сервісно-Орієнтована Архітектура: Концепції, Технології та Дизайн" Томас Ерл

9. Клієнт-серверна архітектура [Електронний ресурс] // QATestLab. – 2020. – Режим доступу до ресурсу: https://training.qatestlab.com/blog/technical articles/client-server-architecture/.

10. Агрегація в MongoDB: зменшення сукупного трубопроводу та карти [Електронний ресурс]. – 2016. – Режим доступу до ресурсу: https://uk.myservername.com/aggregation-mongodb.

11. The top programming languages [Електронний ресурс] // GitHub. – 2022. – Режим доступу до ресурсу: https://octoverse.github.com/2022/top programming-languages.

12. "Шаблони проектування сервісів: Фундаментальні рішення для SOAP/WSDL та RESTful веб-сервісів" Роберт Деньо

13. Рейтинг мов програмування 2023 [Електронний ресурс] // Редакція DOU. – 2023. – Режим доступу до ресурсу: https://dou.ua/lenta/articles/language rating-2023/.

14. "Data Structures and Algorithms in Python", Michael T. Goodrich and Roberto Tamassia, 4th edition, Wiley, 2022.

15. "OSPF: The Basics", Juniper Networks, 2023.

16. Tennis Sense: A platform for extracting semantic information from multi-camera tennis data [Електронний ресурс]. Режим https://ieeexplore.ieee.org/document/5201152/authors#authors

17. Shvachych G., Moroz B., Ivaschenko O. , Sushko I., Pobochii I. The implementation features of the aggregation mode of network interface in the multiprocessors computing system. Комп’ютерне моделювання та оптимізація складних систем : Матеріали IV міжнар. науково-практ. конф., м. Дніпро, 1 – 2 листоп. 2018 р. Дніпро. 2018. С. 200 – 202.

18. Ільїн О.Ю., Катков Ю.І., Вергун Д.С., Шашлов А.В. Особливості розгортання мікросервісних додатків за допомогою системи керування контейнерами.

19. The Computer and the Brain (The Silliman Memorial Lectures Series) 3rd Edition by von Neumann, John (Author), Ray Kurzweil (Foreword), Yale University Press; 3 edition, 2012, 136 pages, ISBN-10: 0300181116, ISBN-13: 978-0300181111.

20. Computer Architecture: A Quantitative Approach 5th Edition by John L. Hennessy (Author), David A. Patterson (Author), Morgan Kaufmann; 5 edition (September 30, 2011), 856 pages, ISBN-10: 012383872X, ISBN-13: 978-8178672663.

21. Microservices vs. Service-Oriented Architecture. by Mark Richards. Publisher: O'Reilly Media, Inc.Release Date: April 2016. ISBN: 9781491975657.

22. Ivan Zmerzlyi Microservice architecture – [Електронний ресурс] – 2019 – Режим доступу: https://medium.com/@IvanZmerzlyi/microservice-architecture-f8a382291ff4. Дата доступу: жовтень 2023.

23. Xiao Ma Microservice Architecture at Medium – [Електронний ресурс] – 2019 – Режим доступу: https://medium.engineering/microservice-architecture-at-medium-9c33805eb74f. Дата доступу: жовтень 2023.

24. Netflix MSA Platform – MeltingCon – [Електронний ресурс] – 2019 – https://meltingcon.github.io/2018/assets/files/%EC%A0%95%EC%9C%A4%EC%A7%84.pdf. Дата доступу: жовтень 2023.

25. Docker Cookbook/ by Sébastien Goasguen. Copyright © 2016 Sébastien Goasguen. All rights reserved. Printed in the United States of America. Published by O’Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

26. Чи завжди потрібні Docker, мікросервіси та реактивне програмування? – [Електронний ресурс] – 2019 – Режим доступу: https://www.dataart.com.ua/news/chi-zavzhdi-potribni-dockermikroservisi-ta-reaktivne-programuvannya. Дата доступу: жовтень 2023.

27. Kubernetes: Up and Running, 2nd Edition \ Dive into the Future of Infrastructure. By Brendan Burns, Kelsey Hightower, Joe Beda – Publisher: O'Reilly Media, Release Date: October 2023. Pages: 278.

28. Cloud Native DevOps with Kubernetes – by Justin Domingus, John Arundel. Publisher: O'Reilly Media, Inc. Release Date: March 2019. ISBN: 9781492040750.

29. Offcial Kubernetes documentation https://kubernetes.io/docs/concepts/overview/what-iskubernetes. Дата доступу: жовтень 2023.

30. Ключевые концепции Kubernetes для службы Azure Kubernetes (AKS) – [Електронний ресурс] – 2019 – Режим доступу: https://docs.microsoft.com/ru-ru/azure/aks/concepts-clustersworkloads. Дата доступу: жовтень 2023.

31. Швачич Г.Г., Соболенко О.В., Іващенко О.В. Режим агрегації каналів мережевого інтерфейсу багатопроцесорних обчислювальних систем. Стратегия качества в промышленности и образовании : Материалы конф.,. г. Варна, 5 – 8 июля 2017 г. Варна, Болгария. 2017. Т. 2. С. 452 – 457.

32. Ivaschenko V., Shvachych G., Ivaschenko O., Busygin, V. Improving the efficiency of multiprocessors system through in-line interface network aggregation. Системні технології. Дніпро. 2018. No 2(115). P. 84 – 93.Книга "Мікросервіси. Патерни розробки та рефакторинг" (Microservices: Design Patterns and Refactoring): авторство - G. Макклелланд, М. Джонсон, С. Хуттон.

33. Cloud-assisted body area networks: State-of-the-art and future challenges – [Електронний ресурс]. Режим доступу: https://www.researchgate net/ publication/271918886_Cloud- assisted body area_networks_State-of-the-art_and_future_challenges.

34. The Fundamentals of Data Warehouse + Data Lake = Lake House. – Режим доступу:https://towardsdatascience.com/the-fundamentals-of-data-warehouse-data- lake-lake

35. Olena Vynokurova, Dmytro Peleshko, Marta Peleshko Hybrid Deep Convolutional Neural Network with Multimodal Fusion. In: Babichev S., Peleshko D., Vynokurova O. (eds) Data Stream Mining & Processing. DSMP 2020. Communications in Computer and Information Science. Springer, Cham. 2020. vol. 1158. pp. 62-78.

36. Grinberg M. Flask Web Development: Developing Web Applications with Python. 2nd Edition. – O'Reilly Media, 2018. – 258 p

37. Шаповал, А. Використання gRPC у високопродуктивних системах. – 2021.

38. Головко, П. Шифрування даних у сучасних веб-системах. – 2018.

39. Алієв, І. Кешування як спосіб оптимізації продуктивності API. – 2019.

40. Sharma, R. Introduction to GraphQL API Development. – 2021.

41. Provost, F., & Fawcett, T. Data Science and Its Relationship to Big Data and Data-Driven Decision Making. – 2013.

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Магістерська робота

МЕТОДИКА ОПТИМІЗАЦІЇ ПОКАЗНИКІВ ШВИДКОДІЇ СЕРВІСУ МОНІТОРИНГУ КРИПТОВАЛЮТНОГО РИНКУ

Виконав: студент групи ПДМ-62 Щеголь Анатолій Глібович

Керівник: к.т.н., доц., кафедри ІПЗ Трінтіна Наталія Альбертівна

Київ - 2025

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи: покращення швидкодії сервісу моніторингу криптовалютного ринку шляхом оптимізації процесів обробки даних та взаємодії з API.

Об'єкт дослідження: процес обробки даних і взаємодії з API у сервісі моніторингу криптовалютного ринку.

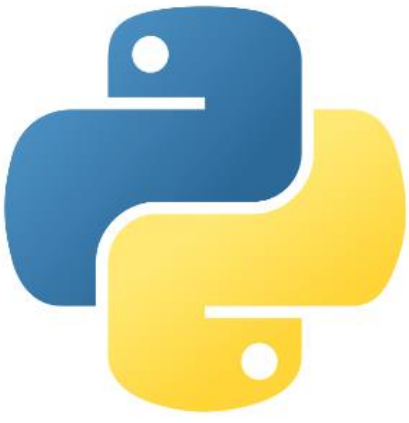
Предмет дослідження: методи та алгоритми оптимізації обробки даних і взаємодії з API для підвищення швидкодії сервісу.

АНАЛІЗ ІСНУЮЧИХ ПІДХОДІВ ОТРИМАННЯ ДАНИХ

Метод	Особливості	Ключові недоліки
Ручний моніторинг	Безпосереднє стеження за ринком користувачем	Низька ефективність при великих обсягах даних, людський фактор
REST API	Стандартна реалізація для збору даних через HTTP-запити	Погана масштабованість, високе навантаження при одночасному запиті великої кількості даних
WebSocket-API	Прямі підключення для отримання актуальних даних у реальному часі	Проблеми з надійністю з'єднання та синхронізацією даних
Моніторинг з обробкою даних на стороні сервера	Оптимізація завдяки попередній обробці та фільтрації даних на сервері	Високі затримки при обробці великих масивів інформації
Метод із кешуванням	Зменшує кількість запитів до API, підвищуючи швидкодію.	Може видавати застарілі дані при високій частоті оновлення ринку.

3

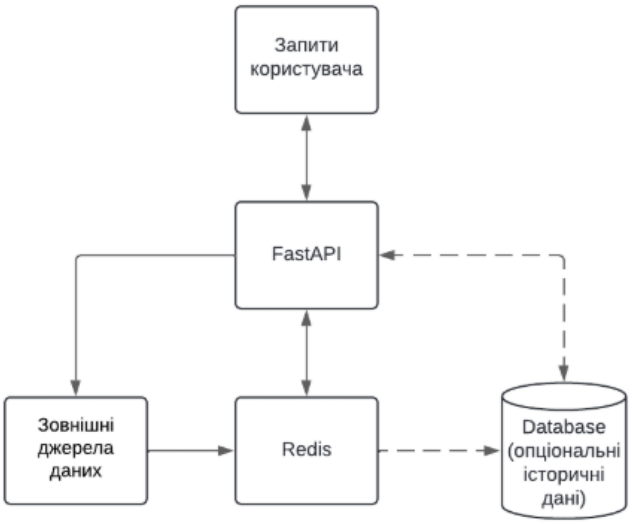
ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ



4

ЗАПРОПОНОВАНЕ РІШЕННЯ

- FastAPI - швидкий, сучасний веб-фреймворк для побудови API з підтримкою асинхронності.
- Redis - використовується як кеш, щоб зменшити кількість повторних запитів до API.
- Asyncio - забезпечує одночасне виконання асинхронної обробки запитів.



5

ПОРІВНЯЛЬНИЙ АНАЛІЗ ТРАДИЦІЙНИХ І ОБРАНИХ ПІДХОДІВ

Метод/Модель	Традиційні підходи	Обрані підходи
Обробка запитів	Синхронні запити (REST API) → Затримки	Асинхронні запити (FastAPI + Asyncio) → Зниження затримок
Зберігання даних	Прямий доступ до бази даних → Високе навантаження	Використання Redis для кешування → Зменшення навантаження
Швидкість відповіді	Висока затримка через послідовні запити	Миттєвий доступ до кешованих даних → Підвищення швидкодії
Масштабованість	Обмежена через перевантаження при високому навантаженні	Підтримка високої кількості одночасних запитів (Asyncio)

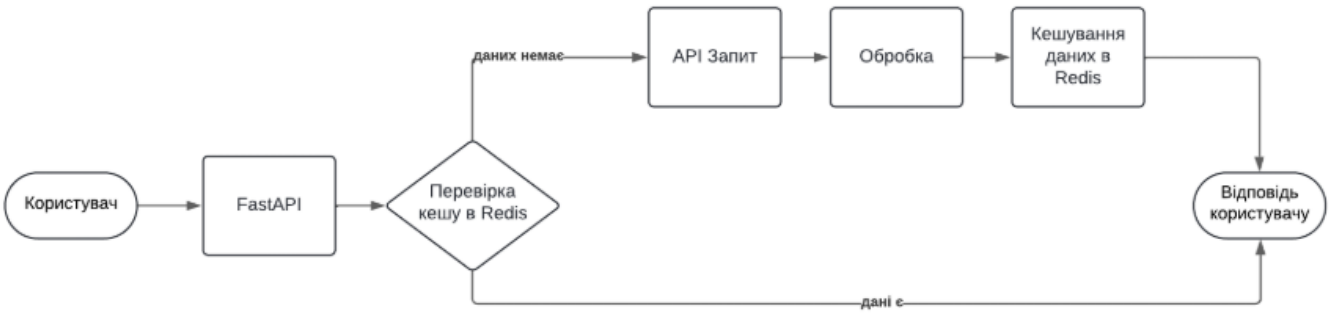
6

ПЕРЕВАГИ ВИБОРУ FAST API

	Django	Flask	FastAPI
Архітектура	MVC (MVT)	Мікрофреймворк	Мікрофреймворк
Продуктивність	Середня (Синх)	Середня (Синх)	Висока (Асинх)
ORM	Django ORM	Підтримується, але не вбудований	Підтримується, але не вбудований (SQLalchemy)
Валідація даних	Django Forms	Користувацька	Pydantic
Масштабованість	Середня	Середня	Висока
Гнучкість	Середня	Середня	Висока
Швидкість розробки	Середня	Висока	Висока

7

СХЕМА МОДИФІКОВАНОГО МЕТОДУ ОПТИМІЗАЦІЇ ПОКАЗНИКІВ КРИПТОВАЛЮТНОГО РИНКУ

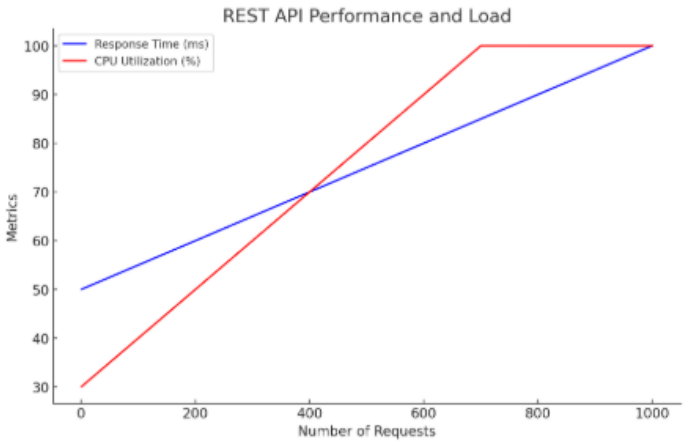


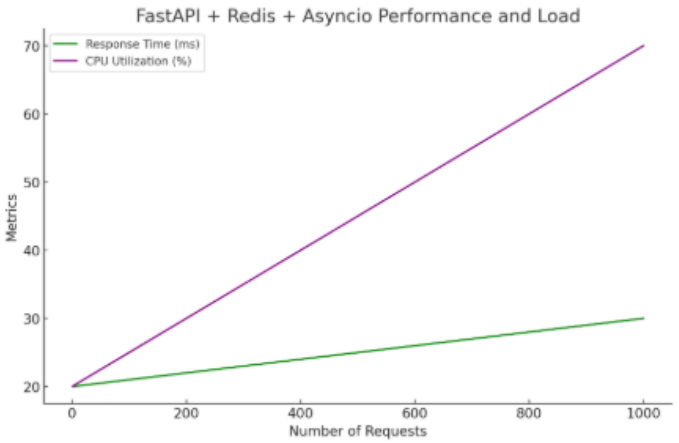
Алгоритм роботи методу:

1. Отримати запит від користувача.
2. Перевірити наявність відповіді в кеші Redis:
   - Якщо дані є: повернути відповідь з кешу.
   - Якщо даних немає: перейти до пункту 3.
3. Виконати асинхронний запит до зовнішнього API.
4. Обробити отримані дані:
   - Агрегувати, фільтрувати або перетворити їх для зручності використання.
   - Зберегти результат у Redis із заданим TTL (Time to live).
5. Повернути результат користувачу.

8

ПОРІВНЯЛЬНИЙ АНАЛІЗ ШВИДКОДІЇ API

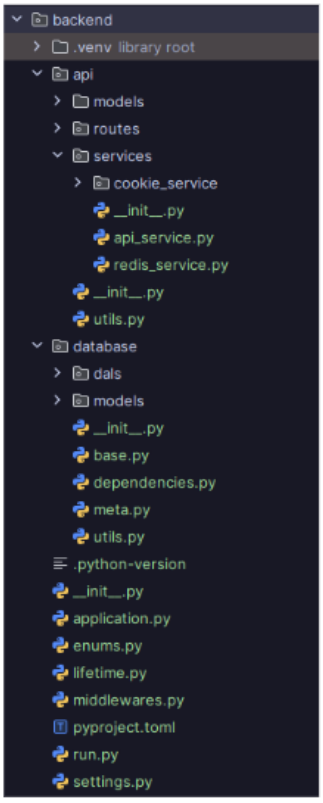


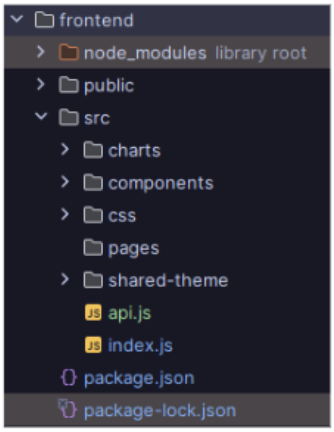


9

СТРУКТУРА ПРОЄКТУ

Практичний результат

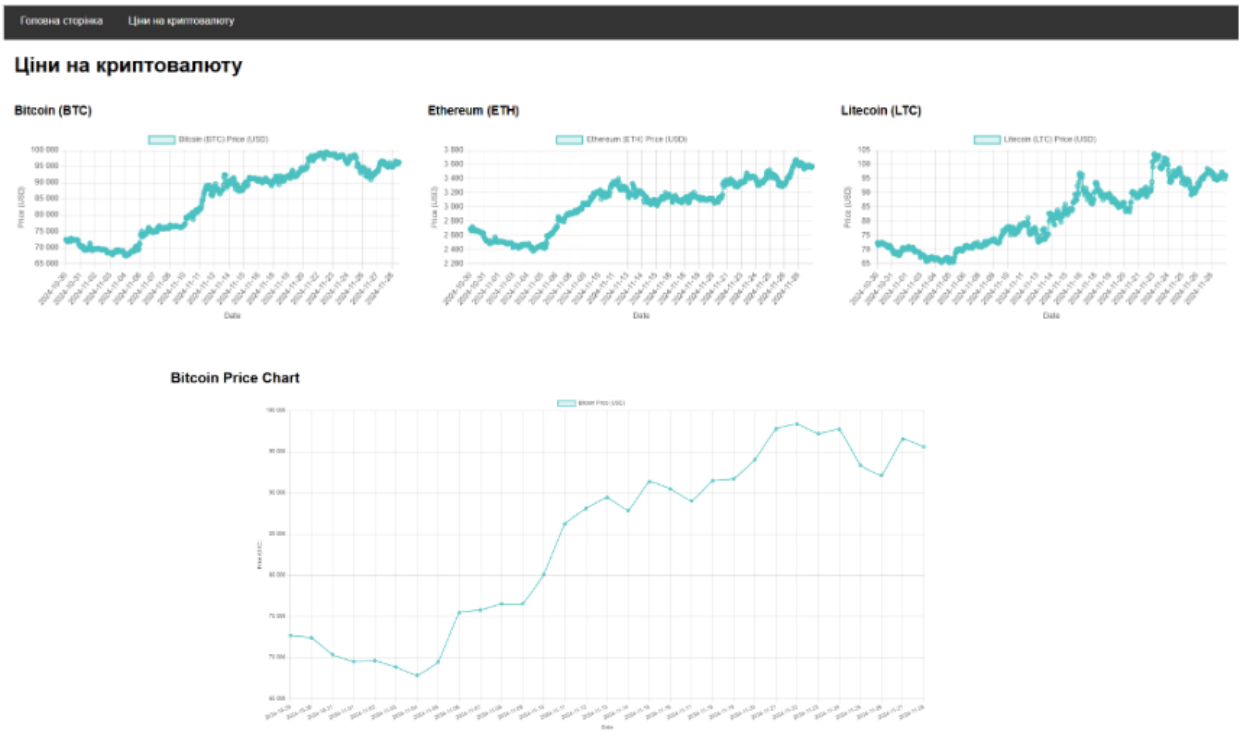




10

ЄКРАННІ ФОРМИ

Практичний результат



11

ВИСНОВКИ

В процесі виконання кваліфікаційної роботи, що включала в себе дослідження пов'язані з темою роботи, а саме покращення швидкодії сервісу моніторингу криптовалютного ринку, було досягнуто наступних результатів:

1. Розроблено API для моніторингу криптовалютного ринку на основі FastAPI, що забезпечило високу швидкодію завдяки асинхронній обробці запитів.
2. Реалізовано методику оптимізації обробки даних, яка включає використання Redis для кешування популярних даних, що зменшило навантаження на API та базу даних.
3. Досягнуто покращення показників швидкодії: час відповіді скорочено до 200 мс, а пропускна здатність сервісу зросла до 1500 запитів на секунду.
4. Проведено порівняльний аналіз, який підтвердив ефективність запропонованих рішень у зменшенні затримок та підвищенні стабільності роботи системи.
5. Запропоновані підходи можуть бути адаптовані для інших сервісів, що працюють з великими обсягами даних у режимі реального часу, таких як фінансові та аналітичні системи.

ПУБЛІКАЦІЇ ТА АПРОБАЦІЯ РОБОТИ

Статті:

1. Аверічев І. М., Щеголь А. Г., Розробка криптовалютного бота з використанням месенджера Telegram мовою Python // Прикладні проблеми комп'ютерних наук, безпеки та математики, 2024, № 3, С.66-70

Тези доповідей:

1. Трінтіна Н. А., Щеголь А. Г., Розробка програмного web-застосунку для аналізу ринку криптовалют // Матеріали II міжнародної науково-технічної конференції «Сучасні аспекти діджиталізації та інформатизації в програмній та комп'ютерній інженерії», 25-27 грудня 2024 року ДУІКТ, Київ.
2. Аверічев І. М., Щеголь А. Г., Розробка програмного web-застосунку для аналізу ринку криптовалют // Матеріали Всеукраїнської науково-технічної конференції «ЗАСТОСУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ В ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЯХ», 24.04.24, ДУІКТ, Київ, Україна, С.86-88.