

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра інформаційних управляючих систем та технологій

"Допущений до захисту"

Завідувач кафедри ІУСТ

к.т.н, доц. Михелєв І.Л.

" ____ " _____ 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти "магістр"

**на тему: «Системний аналіз та розробка сервісу smart-кешування для
конкурентного доступу до веб-ресурсів з обмеженнями»**

Виконала: студентка групи 6145м

(підпис) *Осадчук Д.Р.*

Керівник роботи: старший викладач

(підпис) *Беркунський Є. Ю.*

м. Миколаїв – 2023 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова
Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра інформаційних управляючих систем та технологій
Спеціальність 124 "Системний аналіз"
Освітня програма "Системний аналіз"

"ЗАТВЕРДЖУЮ"

Гарант освітньої програми

д.ф.-м.н., проф. Ушкац М.В.

" ____ " _____ 2023 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ на здобуття ступеня вищої освіти "магістр"

Студентці Осадчук Дар'ї Русланівні

(прізвище, ім'я, по батькові)

1. Тема роботи: «Системний аналіз та розробка сервісу smart-кешування для конкурентного доступу до веб-ресурсів з обмеженнями»

Керівник роботи Беркунський Євген Юрійович

(прізвище, ім'я, по батькові)

Затверджені наказом ректора № 1098-уч від "18" вересня 2023 року.

2. Термін подання роботи 15 грудня 2023 р.

3. Вихідні дані до проекту (роботи) ДСТУ щодо обробки інформації, літературні джерела, технічна документація

4. Перелік питань, що належать до розробки (найменування розділів): дослідження системи використання веб-додатку який взаємодіє з веб-ресурсом з обмеженнями, проектування архітектури та розробка концепції нового сервісу для веб-додатка який взаємодіє з веб-ресурсом з обмеженнями, розробка проектних рішень щодо нового сервісу для веб-додатка який взаємодіє з веб-ресурсом з обмеженнями, розрахунок та оцінка економічного ефекту від розробки нового сервісу для веб-додатка який взаємодіє з веб-ресурсом з обмеженнями, охорона праці, охорона навколишнього середовища.

5. Перелік презентаційних матеріалів: схема архітектури нового сервісу для веб-додатка, модель варіантів використання нового сервісу, концептуальна модель нового сервісу, логічна та фізична модель, діаграма класів та діаграма послідовності нового сервісу, сема автоматизації технічного забезпечення,

організаційна структура компанії, функціональну структуру веб-додатку з новим сервісом та функціональна структура нового сервісу.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Беркунський Є.Ю.		
2	Беркунський Є.Ю.		
3	Беркунський Є.Ю.		
4	Беркунський Є.Ю.		
5	Беркунський Є.Ю.		
6	Беркунський Є.Ю.		

7. Дата видачі завдання 05 вересня 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів роботи	Примітка
1	Вивчення і опис предметної області		
2	Складання вимоги до нового сервісу		
3	Розробка концепції системи		
4	Розділ з розробки проектних рішень по системі (та її частинах)		
5	Реалізація проекту та розробка робочої документації		
6	Вирішення питань охорони праці		
7	Виконання графічних документів		
8	Оформлення пояснювальної записки		
9	Подання КР на перевірку на плагіат		
10	Подання КР рецензенту		
11	Подання КР на державний захист		

Студент

(підпис)

Керівник роботи

(підпис)

Осадчук Д.Р.

(ПІБ)

Беркунський Є.Ю.

(ПІБ)

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Системний аналіз та розробка сервісу smart-кешування для конкурентного доступу до веб-ресурсів з обмеженнями».

Актуальність даної теми зумовлена сучасними тенденціями розвитку технологій, що вимагають неперервне підвищення продуктивності.

Мета кваліфікаційної роботи є розробка нового сервісу smart-кешування для конкурентного доступу до веб-ресурсів з обмеженнями за кількістю запитів.

Кваліфікаційна робота направлена на системний аналіз веб-додатку, який взаємодіє з веб-ресурсом з обмеженнями за кількістю запитів та розробка нового сервісу smart-кешування для конкурентного доступу до веб-ресурсів з обмеженнями. У ролі веб-ресурсу з обмеженнями за кількістю запитів буде виступати великий онлайн-сервіс обміну цифрових валют Binance. У ролі веб-додатку буде розглянутий торговий бот Binance від додатку Good Crypto.

Робота виконана на 90 сторінок, містить 15 рисунків, 2 таблиці, 1 додаток, який містить інструкцію з використання.

Робота виконана українською мовою.

ABSTRACT

Theme of the diploma project: "System analysis and development of a smart-caching service for competitive access to restricted web resources".

The urgency of this topic is due to current trends in technology, which require continuous productivity.

The goal of the qualification work is to develop a new smart-caching service for competitive access to web resources with restrictions on the number of requests.

The qualification work is aimed at the system analysis of a web application that interacts with a web resource with restrictions on the number of requests and the development of a new smart-caching service for competitive access to web resources with restrictions. The large online digital currency exchange service Binance will act as a web resource with restrictions on the number of requests. The Binance trading bot from the Good Crypto app will be considered as a web application.

The work is performed on 90 pages, contains 15 figures, 2 tables, 1 appendix containing instructions for use.

The work is done in Ukrainian.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ.....	10
ВСТУП.....	11
РОЗДІЛ 1. ДОСЛІДЖЕННЯ СИСТЕМИ ВИКОРИСТАННЯ ВЕБ-ДОДАТКУ ЯКИЙ ВЗАЄМОДІЄ З ВЕБ-РЕСУРСОМ З ОБМЕЖЕННЯМИ	14
1.1 Аналіз предметної області.....	14
1.2 Структура веб-додатку.....	21
1.2.1 Загальний опис веб-додатку	21
1.2.2 Функціональні можливості веб-додатку.....	23
1.2.3 Архітектура веб-додатку.....	24
1.3 Аналіз існуючих рішень для обмеження запитів до веб-ресурсів.....	25
1.4 Вітчизняний та зарубіжний досвід застосування веб-додатку який взаємодіє з веб-ресурсом з обмеженнями.....	33
1.5 Постановка задачі та опис нового сервісу для веб-додатка який взаємодіє з веб-ресурсом з обмеженнями	35
1.6 Вимоги та архітектура нового сервісу для веб-додатка який взаємодіє з веб-ресурсом з обмеженнями	36
1.6.1 Дослідження проблем побудови нового сервісу	36
1.6.2 Розробка архітектури для нового сервісу	37
1.6.3 Вимоги до якості розподілення ресурсів у новому сервісі.....	38
Висновок до розділу 1	39
РОЗДІЛ 2. ПРОЕКТУВАННЯ АРХІТЕКТУРИ ТА РОЗРОБКА КОНЦЕПЦІЇ НОВОГО СЕРВІСУ ДЛЯ ВЕБ-ДОДАТКА ЯКИЙ ВЗАЄМОДІЄ З ВЕБ- РЕСУРСОМ З ОБМЕЖЕННЯМИ.....	40
2.1 Побудова моделі варіантів використання	40
2.2 Побудова концептуальної моделі	41
2.3 Рішення з інформаційного забезпечення	42
2.3.1 Логічна модель	42
2.3.2 Фізична модель	43
2.3.3 Діаграма класів	43
2.3.4 Діаграма послідовності	44
2.4 Рішення з технічного забезпечення.....	46
2.5 Рішення з математичного забезпечення.....	48

2.6 Рішення з програмного забезпечення	48
Висновок до розділу 2	50
РОЗДІЛ 3. РОЗРОБКА ПРОЕКТНИХ РІШЕНЬ ЩОДО НОВОГО СЕРВІСУ ДЛЯ ВЕБ-ДОДАТКА ЯКИЙ ВЗАЄМОДІЄ З ВЕБ-РЕСУРСОМ З ОБМЕЖЕННЯМИ	51
3.1 Загальносистемні рішення.....	51
3.1.1 Представлення схеми організаційної структури	51
3.1.2 Представлення схеми функціональної структури веб-додатку з новим сервісом smart-кешування веб-запитів	53
3.1.3 Описання функцій, що автоматизуються	55
3.1.4 Загальний опис нового сервісу	56
3.2 Рішення з організаційного забезпечення	57
Висновок до розділу 3	58
РОЗДІЛ 4. РОЗРАХУНОК ТА ОЦІНКА ЕКОНОМІЧНОГО ЕФЕКТУ ВІД РОЗРОБКИ НОВОГО СЕРВІСУ ДЛЯ ВЕБ-ДОДАТКА ЯКИЙ ВЗАЄМОДІЄ З ВЕБ-РЕСУРСОМ З ОБМЕЖЕННЯМИ	59
4.1 Загальна характеристика роботи	59
4.2 Розрахунок витрат на проектування та впровадження системи	60
4.2.1 Техніко-економічний аналіз	60
4.2.2 Вартісний аналіз	62
4.3 Розрахунок економічної ефективності розробки і впровадження системи	64
Висновок до розділу 4	66
РОЗДІЛ 5. ОХОРОНА ПРАЦІ	68
5.1 Вступ.....	68
5.2 Аналіз небезпечних і шкідливих факторів	68
5.2.1 Освітленість робочого місця	69
5.2.2 Вплив випромінювання	71
5.2.3 Вплив електричного струму	71
5.2.4 Виробничий шум	72
5.2.5 Умови мікроклімату	72
5.2.6 Організація робочого місця	73
5.3 Розрахунок освітленості робочого місця	74
5.4 Розробка заходів щодо зменшення впливу небезпечних і шкідливих факторів	75

5.4.1. Заходи щодо зниження величини виникаючих зарядів статичної електрики	76
5.4.2 Заходи щодо зниження шуму на робочому місці	78
Висновок до розділу 5	79
РОЗДІЛ 6. ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА.....	81
6.1 Вступ.....	81
6.2 Загальні відомості про забруднення.....	81
6.3 Заходи щодо запобігання забруднення навколишнього середовища.....	82
Висновок до розділу 6	82
ВИСНОВКИ	84
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	86
ДОДАТОК А	87

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

БД – база даних.

API – application programming interface.

ООП – об'єктно-орієнтоване програмування.

ORM – Object Relation Mapping.

URL – Uniform Resource Locator.

ВСТУП

Обсяг інформації, яка передається зараз телекомунікаційними каналами зв'язку, величезний, і кожен розробник прагне знизити кількість звернень до сервера та інформації, що отримується за кожен запит.

Ступінь інформатизації сучасного суспільства та забезпечення бізнес-процесів зараз дуже висока. Інформатизація та обробка даних сильно пов'язані, а отже, для ефективної інформатизації необхідно ефективно обробляти дані. Для зберігання та обробки даних у сучасних інформаційних системах використовуються бази даних, побудовані на основі різних моделей. Найпоширеніші це реляційна та постріляційна моделі. Інформаційні системи для автоматизації бізнес-процесів зазвичай є веб-додатком. Причин популярності веб-додатків для реалізації інформаційних систем багато, тому веб-додатки як тип архітектури для системи автоматизації бізнес-процесів зараз домінує над іншими типами архітектури.

Зазвичай веб-додаток є клієнт-серверний додаток, в якому як клієнт виступає браузер користувача, як сервер програми виступає будь-який додаток, реалізований за принципом REST API, а також є третій рівень системи – сервер бази даних. Користувачі під час виконання бізнес-функцій взаємодіють у вигляді браузера із сервером додатків, отже, вся бізнес-логіка реалізується у ньому ж. Таким чином, що ефективніше працює сервер, то швидше клієнт отримає дані. Для проектування та розробки сервера використовують ООП та такі механізми, як ORM (Object-Relational Mapping), які дозволяють подати реляційну базу даних у вигляді об'єктів програми.

Продуктивність АІС залежить від швидкості обробки даних у додатку і переважно швидкість обробки залежить від сервера. Звісно можна припустити, що на швидкість обробки даних впливає сервер бази даних, але це не так. На сервері бази даних виконуються лише стандартні CRUD-операції, хай і з сортуваннями та угрупованнями, але якщо сервер реалізований неефективно і десь допускає неправильну обробку даних, то вся оптимізація сервера баз даних зводиться до мінімуму.

Основним показником ефективності роботи сервера є час обробки запиту, тобто час читання, оновлення чи видалення даних, необхідних для виконання бізнес-процесів. Час виконання запиту залежить від часу роботи сервера БД над запитом, часу обробки результатів на сервері та часу передачі на клієнт. Як вже було сказано, на час роботи сервера БД над запитом впливати не виходить, отже, потрібно звернути увагу на час виконання запиту.

Найпростіший варіант скорочення часу виконання запиту – це зниження кількості запитів. Більшість API веб-ресурсів мають обмеження частоти веб-запитів. Ці обмеження спрямовані на підвищення продуктивності та є важливим компонентом інтернет-безпеки. Але з іншого боку, веб-запит до веб-ресурсу є одним з процесів програмних проектів багатьох фірм та компанії. Тобто програмні проекти фірм, які залежать від кількості запитів та часу на отримання відповіді цих запитів, страждають і зазнають збитків, бо через обмеження час на отримання відповідей стає більшим. Отож потрібно кешувати веб-запити на клієнті. Якщо необхідна для роботи інформація перебуватиме на робочій станції, де відбувається виконання бізнес-процесу, то не доведеться щоразу звертатися до сервера за інформацією та витратити час на новий запит.

Кваліфікаційна робота направлена на системний аналіз веб-додатку, який взаємодіє з веб-ресурсом з обмеженнями за кількістю запитів та розробка нового сервісу smart-кешування для конкурентного доступу до веб-ресурсів. У ролі веб-ресурсу з обмеженнями за кількістю запитів буде виступати великий онлайн-сервіс обміну цифрових валют Binance. У ролі веб-додатку буде розглянутий торговий бот Binance від додатку Good Crypto.

Мета кваліфікаційної роботи є розробка нового сервісу smart-кешування для конкурентного доступу до веб-ресурсів з обмеженнями за кількістю запитів.

Об'єктом дослідження виступає веб-додаток – торговий бот Binance від додатку Good Crypto. Предметом дослідження виступає новий сервіс, як спосіб обходу обмежень частоти запитів на веб-ресурс Binance.

Основними задачами кваліфікаційної роботи є аналізувати предметну область веб-додатку, який взаємодіє з веб-ресурсом з обмеженнями за кількістю

запитів, виявити проблему з якою зіштовхуються проекти пов'язані з веб-запитами, дослідити методи обмежень частот веб-запитів, розробити новий сервіс smart-кешування для конкурентного доступу до веб-ресурсів.

Веб-запити – це повідомлення, що надсилаються клієнтом на сервер, щоб викликати виконання деяких дій. Найчастіше для отримання доступу до певного ресурсу. API (Application programming interface) – це контракт, який надає програма, опис способів взаємодії однієї комп'ютерної програми з іншими. Зазвичай входить до опису будь-якого інтернет-протоколу, програмного каркасу (фреймворку) або стандарту викликів функцій операційної системи.

У комп'ютерних мережах обмеження швидкості використовується для контролю швидкості запитів, що надсилаються або приймаються контролером мережного інтерфейсу. Його можна використовувати для запобігання DoS-атак і обмеження веб-скрапінгу.

Binance – великий онлайн-сервіс обміну цифрових валют, блокчейн-система та постачальник інфраструктури для криптовалютної галузі з комплексом продуктів, серед яких – торгівля цифровими активами, інвестиції, децентралізація та інфраструктурні рішення, дослідження, освіта та інші напрямки. Good Crypto – це менеджер портфеля та мультибіржовий торговельний термінал, який надає просунуті торговельні інструменти, що дозволяють торгувати швидше та безпечніше, а також більш ефективно контролювати торговельні ризики.

РОЗДІЛ 1. ДОСЛІДЖЕННЯ СИСТЕМИ ВИКОРИСТАННЯ ВЕБ-ДОДАТКУ ЯКИЙ ВЗАЄМОДІЄ З ВЕБ-РЕСУРСОМ З ОБМЕЖЕННЯМИ

1.1 Аналіз предметної області

Багато фірм та компаній використовують у своїх проектах веб-запити на необхідних їм веб-ресурсах. Але в більшій мірі кожен з таких веб-ресурс має обмеження на частоту веб-запитів, бо розробники веб-ресурсу працюють над тим, щоб їхній веб-ресурс працював максимально ефективно. В іншому випадку всі, хто користується веб-ресурсом, страждатимуть від низької продуктивності.

Продуктивність це не єдина причина обмеження частоти запитів до веб-ресурсу. Обмеження доступу до API (API limiting), яке також відоме як обмеження частоти запитів (Rate limiting), є важливим компонентом інтернет-безпеки, оскільки DoS-атаки успішніші проти сервера без обмеження за запитами до API.

Обмеження частоти запитів також допомагає зробити API масштабованим. Якщо API раптом стане дуже популярним, то можуть бути несподівані сплески трафіку, що викликають серйозні затримки та серверні лаги.

Якщо надати необмежений доступ до свого API, то користувачі отримають всю силу та можливості. Будь-який бажаючий може використовувати ваш API з якою йому завгодно інтенсивністю, у будь-який час, або постійно. З якогось боку чудово, що люди хочуть використовувати ваш API і вважають його корисним, але відкритий доступ може знизити цінність API та обмежити його успіх у бізнесі. Обмеження доступу є критичним компонентом масштабування продукту API.

Власники API зазвичай використовують одиницю вимірювання «Транзакції за секунду» (TPS) для вимірювання лімітів обробки даних. Деякі системи можуть мати фізичні обмеження передачі даних. Обидва є частиною обмеження запитів на сервера (Backend Rate Limiting).

Щоб запобігти перевантаженню API, власники API часто використовують обмеження на кількість запитів або кількість даних, які можуть запитувати

клієнти. Це називається обмеженням запитів на стороні програми (Application Rate Limiting).

Якщо користувач надсилає занадто багато запитів, обмеження запитів до API може "тротлити" клієнтські з'єднання (пропускати деякі вхідні запити, або відкладати їх у чергу на обробку), замість того щоб негайно їх відключати. Троттлінг запитів дозволяє клієнтам як і раніше використовувати сервіс, одночасно захищаючи API. Однак, завжди існує ризик закінчення часу очікування запитів (timeout), а висячи відкриті з'єднання також підвищують ризик DoS-атак.

Один із підходів до обмеження на запити до API полягає в тому, щоб пропонувати безкоштовний доступ та преміальний доступ із різними лімітами за запитами для кожного. Є багато речей, які слід враховувати під час ухвалення рішення про те, яку стягувати плату за преміальний доступ до API. Постачальники API, як і раніше, повинні враховувати наступне при налаштуванні своїх обмежень для запитів до API:

- чи пропускаються запити, коли вони перевищують ліміт;
- чи викликають нові виклики та запити додаткову плату;
- чи отримують нові виклики та запити певний код помилки і якщо так, то який.

Для повного огляду розглянемо кілька прикладів. Нижче представлено, як кілька провайдерів (GitHub, LinkedIn та Bitly) API передають рекомендації щодо обмеження доступу до API своїм користувачам-розробникам.

Документація GitHub пояснює різні обмеження доступу для автентифікованих та неавтентифікованих користувачів. Він також визначає заголовок, який повертається, та значення деяких команд обмеження доступу.

Документація також дозволяє користувачам дізнатися, як перевірити їх поточні ліміти використання, інформує про те, як збільшити обмеження для певних додатків, і описує наслідки, коли клієнти зловживають лімітами з використання.[7]

Документація за лімітами на використання API у LinkedIn пояснює, що різні кінцеві точки API мають різні обмеження. API LinkedIn пропонує три різні типи лімітів: ліміт програми, ліміт користувача та ліміт розробника. Усі з них встановлюють максимальну кількість дзвінків API на день. У документації також вказується часовий пояс, який використовується для визначення початку та кінця дня.[8]

Документація Bitly не лише визначає їхні правила, вони також й пропонують рекомендації щодо найкращої практики для уникнення проблем з обмеженням запитів. Деякі з цих практик:

- використання аутентифікації;
- кешування;
- пакетна обробка;
- кодування URL.[9]

Ці приклади ілюструють, як можна допомогти розробникам уникнути деяких проблем з лімітами на використання API та уникнути тротлінгу запитів.

У зв'язку з тим, що існує безліч способів підключення програми до API, існують різні способи регулювати трафік за допомогою тротлінгу.

Обмеження викликів для сторонніх API та програм є однією з найскладніших ситуацій для обмеження. Наприклад, якщо клієнти безпосередньо звертаються до API Google Maps, обмежити їх не можна. Необхідно буде заплатити за відповідний рівень використання даних.

Якщо доступ до API з обмеженням на запит здійснюється через будь-який проміжний обробник на сервері, буде набагато простіше обмежити запити API за допомогою коду. Скажімо, якщо API дозволяє лише 20 запитів за секунду, тоді можна налаштувати обробник, який пропускає лише 20 запитів за секунду.

Якщо всі ці запити виконуються синхронно, це може не мати значення, але можна швидко побачити різницю, коли справа доходить до асинхронних завдань.

У ролі веб-ресурсу з обмеженнями за кількістю запитів в кваліфікаційній роботі виступає великий онлайн-сервіс обміну цифрових валют Binance. Binance – великий онлайн-сервіс обміну цифрових валют, блокчейн-система та

постачальник інфраструктури для криптовалютної галузі з комплексом продуктів, серед яких – торгівля цифровими активами, інвестиції, децентралізація та інфраструктурні рішення, дослідження, освіта та інші напрямки. Криптокошелек – це інструмент, який можна використовувати для взаємодії з блокчейн-мережею. Гаманці поділяються на три групи: програмні, апаратні та паперові.

Більшість компаній, що надають гаманці для зберігання криптовалюти, використовують програмне забезпечення, що робить їх зручнішими порівняно з апаратними аналогами. У свою чергу апаратні гаманці є безпечнішими – так само як і паперові, які складаються з даних, надрукованих на аркуші паперу. Проте нині їх використання вважається застарілим та ненадійним.

Попри поширену думку, криптовалютні гаманці не призначені для фактичного зберігання цифрових активів. Натомість вони надають інструменти для взаємодії з блокчейном. Іншими словами, ці гаманці можуть генерувати необхідні дані для відправлення та отримання криптовалюти за допомогою транзакцій. Ці дані складаються з однієї або кількох пар публічних та приватних ключів.

Гаманець також включає адресу з буквено-цифровим ідентифікатором, який генерується на основі публічного і приватного ключів. Такий набір символів є певним «місцем» у блокчейні, на яке можна відправляти монети.

Binance API дозволяє підключатися до серверів Binance за допомогою різних мов програмування, і використовувати дані, що отримуються з Binance, в сторонніх додатках. Наприклад, у них можна переглядати інформацію про гаманець та дані про транзакції, здійснювати угоди, вводити та виводити кошти через сторонні програми.

Розробники Good Crypto, використовують Binance API для взаємодії із платформою Binance. Сторонні програми взаємодіють із API біржі, використовуючи HTTP-запити, такі як HTTP GET, POST та інші методи. Вони можуть надсилати запити на створення ордерів, отримання балансу рахунку, отримання історії торгів та інші операції.[12]

Існує три різні типи обмежень Binance API, кожен з яких може бути змінений у будь-який час:

- суворі обмеження;
- обмеження ML (Machine Learning/машинне навчання);
- обмеження WAF (Web Application Firewall/фаїрвола веб-додатків).

Суворі обмеження: 6000 одиниць ваги запитів за хвилину (зверніть увагу: цей показник не обов'язково відповідає 6000 запитам), 50 ордерів за 10 секунд та 160 000 ордерів за добу.

Обмеження машинного навчання. Торговельну поведінку оцінюють щодо середнього користувача. У разі надмірних відхилень ви будете заблоковані на термін від 5 хвилин до 3 днів. Припустимо, що ваш показник свідчить про порушення в торгівлі. Показник зростає у разі:

- постійного «фронтраннінгу», або «випередження на одиницю» кращої ставки бід/аск у книзі ордерів;
- масового створення ордерів та їх дуже швидкого скасування без виконання;
- низького рівня конверсії, «рівень конверсії» - відношення кількості угод до загальної кількості створених та скасованих ордерів за добу (кількість угод / (ордери + скасування));
- низької «ваги», «вага» - відношення загальної кількості активів в угодах до загальної кількості створених та скасованих ордерів за добу (загальна кількість активів у угодах / (ордери + скасування));
- торгові пари з BNB як базовий чи котируваний актив збільшують показник порушень.

Binance не може дати нам більш детальну інформацію, ніж та, що представлена вище. Користувачам пропонується розробити свою ідеальну торгову систему експериментальним шляхом або обговорити оптимальну поведінку з іншими користувачами API.

Обмеження WAF (Web Application Firewall/фаїрвола веб-додатків). Binance не надає точних відомостей про власні правила WAF, але якщо ми зіткнулися з

помилкою HTTP 403, ми порушили це правило. Найімовірніше, це пов'язано із надмірною кількістю запитів протягом 5 хвилин. При цьому, якщо ваш запит віднесли до категорії шкідливих, це може спричинити більш тривале блокування.

У разі строгих обмежень перевищення хвилинного ліміту сукупної ваги запитів (зараз 6000) призведе до блокування IP. Ліміт ордерів (зараз – 50 за 10 секунд і 160 000 на добу) залежати від облікового запису. У разі перевищення ліміту ордерів користувачам також буде заборонено створювати нові ордери на сайті.

У разі лімітів машинного навчання до облікових записів будуть застосовуватись обмеження. У разі обмеження користувача системою машинного навчання про причину і тривалість можна буде дізнатися через кінцеву точку [/sapi/v1/account/status]. Обмеження фаєрволу веб-застосунків будуть застосовуватись лише по IP. Зазвичай блокування WAF триває 5 хвилин. Однак, якщо сервер визначить запит користувача як шкідливий, тривалість може збільшитись.

Проаналізувавши об'єкт дослідження можна виявити проблему я якою зштовхуються розробники проектів які пов'язанні з запитами на веб-ресурсах.

Програмних проекти, які пов'язані з запитом до певного ресурсу на веб-сервері, досить часто мають труднощі з обмеженнями частоти запитів на цих веб-серверах. Для деяких проектів дуже важливо отримувати дані з ресурсу вчасно та як можна частіше. Від кількості часу очікування відповіді та кількості частоти отримування відповідей залежить, як швидко буде працювати проект. Тому розробникам таких проектів потрібно шукати способи обходу та вирішення цієї проблеми пов'язаної з обмеженням частоти запитів.

REST API – це спосіб взаємодії сайтів та веб-додатків із сервером. Термін складається з двох аббревіатур, які розшифровуються в такий спосіб. API (Application Programming Interface) – це код, який дозволяє двом програмам обмінюватися даними із сервера. REST (Representational State Transfer) – це спосіб створення API за допомогою протоколу HTTP. [10]

REST API ґрунтується на протоколі передачі гіпертексту HTTP (Hypertext Transfer Protocol). Це стандартний протокол у інтернеті, створений для передачі гіпертексту. Зараз за допомогою HTTP надсилають будь-які інші типи даних. Кожен об'єкт на сервері в HTTP має свою унікальну URL-адресу у строгому послідовному форматі. У REST API є 4 методи HTTP, які використовують для дій з об'єктами на серверах:

- GET (отримання інформації про дані або список об'єктів);
- DELETE (видалення даних);
- POST (додавання або заміна даних);
- PUT (регулярне оновлення даних).

Такі запити ще називають ідентифікаторами CRUD: create (створити), read (прочитати), update (оновити), delete (видалити). Це стандартний набір дій для роботи із даними. У кожному HTTP-запиті є заголовок, за яким слідує опис об'єкта на сервері – це і є його стан.



Рисунок 1.1 – Архітектура праці REST API

Створення обмежень частоти запитів на веб-ресурсі в одно час є необхідним компонентом масштабування веб-ресурсу, але й в той же час надає складність та проблему для тих, хто хоче активно користуватися інформацією на цьому веб-ресурсі.

Основна задача кваліфікаційної роботи, дослідити існуючі методи обмежень частоти запитів на веб-ресурсі та розробити новий сервіс smart-кешування для конкурентного доступу до веб-ресурсів.

Мета кваліфікаційної роботи є розробка нового сервісу smart-кешування для конкурентного доступу до веб-ресурсів з обмеженнями за кількістю запитів.

1.2 Структура веб-додатку

1.2.1 Загальний опис веб-додатку

У ролі об'єкта дослідження виступає – веб-додаток, торговий бот Binance від додатку Good Crypto. Торгові боти для Binance – це комп'ютерні програми, розроблені для автоматичної торгівлі на популярній криптовалютній біржі Binance. Вони засновані на алгоритмах та стратегіях торгівлі, заданих розробниками.

Основна мета торгових ботів – спростити процес торгівлі криптовалютами та підвищити ефективність операцій. Вони можуть автоматично аналізувати ринок, приймати рішення про купівлю та продаж активів, а також стежити за змінами цін та обсягами торгів.

Переваги використання торгових роботів включають можливість роботи цілодобово, швидку реакцію на зміни ринку, відсутність емоційних факторів при прийнятті рішень та можливість автоматичної торгівлі на основі заздалегідь заданих умов.

Торгові боти отримують дані з Binance через API, який надає доступ до функцій і даних біржі. Binance обмежує доступ до запитів на гаманець через API-ключ з метою забезпечення безпеки та запобігання зловмисницьким діям. Захист доступу до API-ключа має кілька аспектів:

1. Обмеження швидкості запитів. Binance встановлює обмеження кількості запитів, які можна виконати за певний проміжок часу. Це допомагає запобігти навантаженню серверів і захищає від DDoS-атак.

2. Аутентифікація. API-ключ повинен бути правильно автентифікований для кожного запиту. Це гарантує, що лише власник ключа має доступ до свого гаманця.

3. Рівні доступу. Binance надає різні рівні доступу до API-ключа, які можна налаштувати відповідно до потреб користувача. Деякі методи можуть

бути заборонені або дозволені лише для читання залежно від дозволів, встановлених для ключа.

4. IP-обмеження. Binance надає можливість встановити обмеження на доступ до API-ключу тільки з певних IP-адрес. Це дозволяє підвищити безпеку, обмежуючи можливість доступу з ненадійних та неавторизованих джерел.

Торгові боти, розроблені компанією Good Crypto, надають можливість автоматизованої торгівлі на біржі Binance. Good Crypto – це платформа, яка поєднує функціональність терміналу торгівлі та можливості торгових ботів, надаючи трейдерам широкий спектр інструментів для ефективної торгівлі на криптовалютних ринках.[5]

Основні особливості торгового бота Good Crypto для Binance включають:

1. Простота використання. Good Crypto розроблено з урахуванням зручності користувачів, забезпечуючи простий та інтуїтивно зрозумілий інтерфейс. Це дозволяє трейдерам швидко розібратися у налаштуваннях та використанні бота.

2. Безліч стратегій. Good Crypto пропонує кілька готових стратегій торгівлі, які можна вибрати та налаштувати відповідно до ваших уподобань. Це дозволяє трейдерам здійснювати автоматичну торгівлю відповідно до обраної стратегії без необхідності безперервно відстежувати ринок.

3. Технічний аналіз. Good Crypto пропонує низку інструментів технічного аналізу, які допомагають трейдерам приймати поінформовані рішення. Це включає в себе індикатори, що настроюються, графіки, аналіз історичних даних та інші функції, які допомагають визначити оптимальні точки входу і виходу з позицій.

4. Управління ризиками. Good Crypto надає можливості для управління ризиками, такі як встановлення обмежень зі збитків та прибутку, використання стоп-лосс та тейк-профіт ордерів, а також динамічне управління обсягом позиції. Все це допомагає мінімізувати потенційні збитки та захищати прибуток.

5. Реальний час та сповіщення. Good Crypto надає можливість відстежувати трейдингові операції в режимі реального часу та надсилати

повідомлення про важливі події на ринку. Це допомагає трейдерам бути в курсі поточної ситуації та приймати рішення своєчасно.

1.2.2 Функціональні можливості веб-додатку

Торгові боти зв'язуються з Binance за API, який надає доступ до функцій і даних біржі. Для зв'язку з Binance через API, торгові боти зазвичай виконують такі кроки:

1. Отримання API-ключа. Користувач повинен зареєструватися на Binance і створити API-ключ для доступу до свого облікового запису. Це включає генерацію ключа доступу і секретного ключа.
2. Аутентифікація. Торговий бот повинен використовувати отримані ключі для аутентифікації при кожному запиті до API Binance. Це забезпечує безпеку та гарантує, що тільки власник ключа має доступ до свого облікового запису.
3. Формування запиті. Торгові роботи створюють HTTP-запити для відправки на сервер Binance, використовуючи методи API, такі як отримання балансів, розміщення ордерів, отримання історії торгівлі і т. д. Кожен запит повинен містити параметри, необхідні для виконання відповідних операцій.
4. Надсилання запитів. Боти надсилають сформовані запити на сервер Binance з використанням правильної URL-адреси API і параметрів аутентифікації (API-ключ і секретний ключ). Запити можуть бути надіслані через протоколи HTTP або HTTPS, залежно від вимог Binance.
5. Обробка відповідей. Після надсилання запиту Binance повертає відповідь, що містить запитані дані або результат операції. Торгові роботи повинні обробляти ці відповіді та аналізувати інформацію, щоб приймати відповідні рішення на основі стратегії торгівлі.[11]

Важливо мати на увазі, що зв'язок з Binance через API вимагає правильної реалізації протоколу аутентифікації та має відповідати вимогам безпеки, встановленим біржою.

1.2.3 Архітектура веб-додатку

Архітектура програми торгового бота для Binance включає такі компоненти:

1. З'єднання з Binance API.

Цей компонент відповідає за встановлення з'єднання з API Binance та отримання даних про ринок, розміщення ордерів та відстеження рахунку. Тут враховується безпека з'єднання та обробку помилок.

2. Модуль стратегій.

Це основний компонент програми, що відповідає за реалізацію різних стратегій торгівлі. Тут включені модулі для технічного аналізу, статистичного моделювання, машинного навчання та інших підходів. Кожна стратегія має певні правила входу та виходу з позицій, управління ордерами та управління ризиками.

3. Управління портфелем та ризиками.

Цей компонент відповідає за управління розміром позиції, встановлення стоп-лосс та тейк-профіт ордерів, а також управління ризиками відповідно до заданого рівня ризику. Тут розроблено ефективну стратегію управління капіталом та ризиками, щоб мінімізувати втрати та максимізувати прибуток.

4. Моніторинг та сповіщення.

Цей компонент відповідає за моніторинг поточних позицій, виконання стратегій торгівлі та надсилання повідомлень про важливі події на ринку. Тут використовуються різні інструменти для моніторингу, наприклад, повідомлення електронною поштою, SMS-повідомлення або месенджери.

5. База даних.

Для зберігання та керування даними, такими як історичні дані ринку або журнали торгівлі, може знадобитися використання бази даних. Тут вибрана відповідна база даних, враховуючи вимоги до продуктивності, масштабованості та безпеки.

6. Інтерфейс користувача.

Цей компонент надає інтерфейс користувача для управління і налаштування торгового бота. Тут реалізовано функції для зміни налаштувань

стратегій, перегляду результатів торгівлі, встановлення параметрів ризику та отримання звітів.

Загальна ідея полягає в тому, що модульна архітектура дозволяє гнучко налаштовувати та комбінувати різні компоненти відповідно до вимог трейдера.

1.3 Аналіз існуючих рішень для обмеження запитів до веб-ресурсів

Багато сервісів, які використовують REST API, мають обмеження запитів на API як захист від DoS-атак і перевантажених серверів. Деякі API мають м'які обмеження, які дозволяють користувачам перевищувати обмеження протягом короткого часу. Інші використовують більш жорсткий підхід, негайно повертаючи помилку HTTP 429 Too Many Requests та час очікування, змушуючи користувача надсилати новий запит. Помилка 429 "Занадто багато запитів", також може супроводжуватися заголовком Retry-After, що вказує, через який час можна повторити запит.

Встановлення тайм-ауту – це найпростіший спосіб обмежити запити до API. Встановити затримку, а потім повернути потрібне повідомлення своїм користувачам. Цей метод працює як годинник для тих, хто хоче швидко щось запустити.

Принципи REST API:

- **Відділення клієнта від сервера (Client-Server).** Клієнт – це інтерфейс сайту або програми, наприклад, пошуковий рядок відеохостингу. У REST API код запитів залишається на стороні клієнта, а код доступу до даних – на стороні сервера. Це спрощує організацію API, дозволяє легко переносити інтерфейс користувача на іншу платформу і дає можливість краще масштабувати серверне зберігання даних.
- **Відсутність запису стану клієнта (Stateless).** Сервер не повинен зберігати інформацію про стан (проведені операції) клієнта. Кожен запит від клієнта повинен містити тільки інформацію, яка потрібна для отримання даних від сервера.

- Кешування (Cachable). У даних запиту має бути зазначено, чи потрібно кешувати дані (зберігати у спеціальному буфері для частих запитів). Якщо така вказівка є, клієнт отримає право звертатися до цього буфера за потреби.
- Єдність інтерфейсу (Uniform Interface). Усі дані повинні запитуватися через одну URL-адресу стандартними протоколами, наприклад, HTTP. Це спрощує архітектуру сайту або програми та робить взаємодію з сервером зрозумілішим.
- Багаторівневість системи (Layered System). У сервері REST API можуть розташовуватися на різних рівнях, при цьому кожен сервер взаємодіє тільки з найближчими рівнями і не пов'язаний запитами з іншими.
- Надання коду на запит (Code on Demand). Сервери можуть надсилати клієнтові код (наприклад, скрипт для запуску відео). Так загальний код програми або сайту стає складнішим лише за необхідності.
- Початок від нуля (Starting with the Null Style). Клієнт знає лише одну точку входу на сервер. Подальші можливості взаємодії забезпечуються сервером.[4]

Три основних способи реалізації обмеження частоти запитів до API:

1. Черги запитів. Існує безліч бібліотек з реалізацією черг запитів, і кожна мова програмування чи середовища розробки має свої власні рішення. Це означає, що велика важка робота вже виконана. Є навіть каталоги бібліотек з реалізацією черг, що полегшує пошук заздалегідь написаного коду. Вже є кілька бібліотек з обмеженням потоку запитів.

Одна конкретна бібліотека встановлює обмеження швидкості два запити в секунду і поміщає інші чергу запитів. Існує безліч бібліотек черг запитів, які готові до використання.

Android Volley. Volley – це особливо популярна бібліотека черги запитів для розробників Android. Не кожна бібліотека Android може використовувати переваги Volley, оскільки деякі з них потребують ширших мережесх

можливостей. Перегляньте документацію своєї бібліотеки Android, щоб перевірити, чи сумісна вона з Volley.

Amazon Simple Queue Service (SQS). Проста служба черги Amazon (SQS) – це готова бібліотека черг, яка ідеально підходить для черг запитів та повідомлень. Проект регулярно підтримується, тому вам не доведеться постійно налагоджувати апаратне або програмне забезпечення, щоб змусити працювати SQS.

2. Троттлінг. Троттлінг є ще одним поширеним способом реалізації обмеження потоку запитів. Він дозволяє розробникам API контролювати їх використання API, встановлюючи проміжний стан, дозволяючи API оцінювати кожен запит. Коли троттлінг активується, користувач може або вимкнутись, або просто зменшити навантаження.

Ймовірно, на рівні програм, API або користувачів троттлінг є популярним методом обмеження навантаження API. Тому на ринку є кілька комерційних продуктів, які готові для розробників. Прогрес пропонує нам Hybrid Data Pipeline, який представляє троттлінг API для: IBM DB2, Oracle, SQL Server, MySQL, PostgreSQL, SAP Sybase, Hadoop Hive, Salesforce, Google Analytics.

Також пропонують OpenAccess SDK для пропрієтарних API. OpenAccess SDK пропонує стандартні інтерфейси SQL, такі як ODBC, JDBC, ADO.NET чи OLE-DB. OpenAccess SDK легко інтегрується з більшістю систем безпеки та авторизації, що робить його корисним міжмережевим екраном між API та внутрішніми системами.

3. Алгоритми обмеження запитів до API. Алгоритми є ще одним способом створення API, що масштабуються, з обмеженням за запитами. Як і у випадку бібліотек черг та троттлінг-служб, вже є багато алгоритмів для обмеження запитів.

Leaky Bucket. Алгоритм Leaky bucket («поточного відра») – це просте та легке у використанні рішення для обмеження запитів. Він переводить запити до структури «першим прийшов – першим вийшов» (FIFO), обробляючи елементи у черзі з постійною швидкістю.

Алгоритм Leaky Bucket згладжує сплески трафіку, його легко впровадити на одному сервері або балансувальнику навантаження. Він також малий та економить пам'ять через обмежений розмір черги.

Fixed Window. Алгоритми виду Fixed Window використовують фіксовану швидкість для відстеження швидкості запитів за допомогою простого інкрементного лічильника. Вікно визначено для заданої кількості секунд, наприклад, 3600 години. Якщо лічильник перевищить межу для встановленої тривалості, наступні запити будуть відхилені.

Алгоритм фіксованого вікна – це простий спосіб переконатися, що ваш API не ув'язує у старих запитах. Однак ваш API все ще може бути перевантажений під час використання цього методу. Якщо при оновленні вікна надходить безліч запитів, то ваш API може виявитися перевантаженим.

Sliding Log. Алгоритм Sliding Log («ковзного журналу») включає відстеження кожного запиту використовуючи журналування з позначкою часу. Записи з тимчасовими мітками, що перевищують ліміт частоти запитів, відкидаються. Коли надходить новий запит, сума логів розраховується визначення частоти запитів. Якщо запит перевищує граничне значення, вони просто ставляться у чергу.

Алгоритми виду Sliding Log не страждають від проблем із перевантаженням, як у алгоритмах із фіксованими вікнами. Однак, зберігати необмежену кількість журналів для кожного запиту може бути досить дорого. Розрахунок кількості запитів на кількох серверах також може бути дорогим. Алгоритми ковзного журналу безумовно не найкраще рішення для запобігання перевантаженню або запобіганню DoS-атак, а також масштабованих API.

Sliding Window. Алгоритми Sliding Window («ковзного вікна») поєднують усе найкраще з алгоритмів фіксованого вікна та журналу, що ковзає. Використовується накопичувальний лічильник за заданий період, аналогічний алгоритму з фіксованим вікном. Попереднє вікно також оцінюється, щоб допомогти згладити сплески трафіку.

Невелика кількість встановлених даних, необхідних для оцінки кожного запиту, робить алгоритм ковзного вікна ідеальним вибором обробки великої кількості запитів, залишаючись при цьому легким і швидким у виконанні.

Щоб отримати дані з сайтів, використовується готовий API, проте він не завжди доступний. Тоді доводиться використовувати «парсер» веб-сторінки – в автоматичному або напівавтоматичному режимі розпізнавати код і отримувати дані з необхідних полів. Обходом безлічі сторінок і сайтів займається програмне забезпечення, яке зазвичай називається «краулер».

Процес збирання даних із сайтів краулером називається веб-скрейпінгом.

Більшість популярних сайтів активно захищають свої ресурси від скрейпінгу, але скрейпери не відстають від них та вигадують нові стратегії обходу.

Ще одним поширеним рішенням є використання системи розпізнавання шкідливої активності (WAF), яка дозволяє відфільтровувати небажані запити на основі визначених правил або алгоритмів. WAF може аналізувати дані запиту, такі як IP-адреса відправника, заголовки, зміст запиту та інші атрибути, щоб визначити, чи є запит підозрілим чи потенційно шкідливим.

І, нарешті, можна використовувати аналітику даних запитів виявлення аномальних чи шкідливих дій. Моніторинг та аналіз журналів серверів можуть допомогти виявити незвичайні або підозрілі патерни запитів, що дозволить вживати відповідних заходів для їхнього обмеження.

Кожне з цих рішень має свої переваги та недоліки, і вибір того, яке використання залежить від конкретних вимог та обмежень вашого веб-додатку чи сервісу. Деякі компанії також можуть комбінувати кілька рішень для максимальної ефективності та безпеки.[3]

Проаналізувавши принципи обмежень та способи їх реалізації наведемо методи обходу обмежень частоти веб-запитів на веб-ресурсах:

1. Задати випадкові інтервали між запитами

Жодна реальна людина не відправлятиме запити на сайт щосекунди протягом 24 годин – такий збір інформації дуже легко виявити. Щоб уникнути

блокування, використовуйте випадкові затримки (наприклад, близько 2–10 секунд). І не надсилайте запити занадто часто, інакше сканування сайту буде схоже на мережеву атаку.

Особливо ретельним варто перевірити файл robots.txt (як правило, знаходиться на <http://адреса сайту/robots.txt>). Іноді там можна знайти параметр Crawl-delay, який каже скільки секунд потрібно почекати між запитами, щоб не шкодити роботі сервера.

```
#
# robots.txt
#
# This file is to prevent the crawling and indexing of certain parts
# of your site by web crawlers and spiders run by sites like Yahoo!
# and Google. By telling these "robots" where not to go on your site,
# you save bandwidth and server resources.
#
# This file will be ignored unless it is at the root of your host:
# Used:    http://example.com/robots.txt
# Ignored: http://example.com/site/robots.txt
#
# For more information about the robots.txt standard, see:
# http://www.robotstxt.org/robotstxt.html

User-agent: *
Crawl-delay: 10
# CSS, JS, Images
Allow: /misc/*.css$
Allow: /misc/*.css?
Allow: /misc/*.js$
Allow: /misc/*.js?
Allow: /misc/*.gif
```

Рисунок 1.2 – Приклад файлу robots.txt

2. Встановити адекватний User Agent

User Agent – HTTP-заголовок, який повідомляє відвідуваному веб-сайту інформацію про ваш браузер. Якщо не налаштувати User Agent, ваш краулер буде дуже легко виявити. Крім того, сайти іноді блокують запити агентів користувача від невідомих браузерів, тому потрібно встановити один з популярних користувачів агентів.

Досвідчені скрейпери можуть спробувати встановити свій агент на Googlebot User Agent – пошуковий робот Google. Більшість веб-сайтів, очевидно,

хочуть потрапити у видачу Google та пропускають Googlebot. Хорошою практикою буде чергування різних User Agent.

3. Використовувати проксі

Навряд чи справжній користувач зможе запитувати 20 сторінок за секунду на тому самому сайті. Щоб обдурити веб-сервер, змусіть його думати, що всі ці запити приходять з різних місць. Іншими словами, використовуйте проксі.

Тут є безліч варіантів, наприклад послуги SmartProxy, Luminati Network, Blazing SEO. Безкоштовні проксі не завжди підійдуть для таких цілей: часто повільні і ненадійні. Також можна створити свою проксі-мережу на сервері, наприклад, за допомогою Scrapoxy – API з відкритим вихідним кодом.

4. Додати referer

Referer – це заголовок HTTP-запиту, який дає зрозуміти, з якого сайту ви прийшли. Непоганий варіант зробити так, щоб він показував, ніби ви перейшли з Google. Варто міняти referer для веб-сайтів у різних країнах (.ua). Замість Google можна підставити адреси соцмереж: YouTube, Facebook тощо. Referer допоможе зробити так, щоб запити виглядали як трафік із того сайту, звідки зазвичай приходить найбільше відвідувачів.

5. Використовуйте headless-браузер

Особливо хитромудрі сайти можуть відстежувати веб-шрифти, розширення, cookie-файли, цифрові відбитки (фінгерпринти). Іноді вони навіть вбудовують JavaScript-код, який відкриває сторінку тільки після його запуску – так часто можна визначити, чи надходить запит із браузера. Для обходу таких ресурсів вам знадобиться headless-браузер. Він емулює поведінку справжнього браузера та підтримує програмне управління. Найчастіше для цього вибирають Chrome Headless.

Якщо ресурс відстежує цифровий відбиток браузера, то навіть багаторазова зміна IP і очищення cookie не завжди допомагають, оскільки вас все одно можуть дізнатися про фінгерпринт. За часту зміну IP при тому самому відбитку цілком можуть заблокувати, і одне із завдань Chrome Headless – не допустити цього.

Найпростіший спосіб працювати з Chrome Headless – використовувати фреймворк, який поєднує всі його функції у зручний API. Найвідоміші рішення можна знайти тут. Але деякі веб-ресурси намагаються відслідковувати і їх: триває постійна гонка між сайтами, які намагаються виявити headless-браузери, та headless-браузерами, які видають себе за справжні.

6. Підключення програми для вирішення CAPTCHA

Існують веб-сайти, які систематично просять вас підтвердити, що ви не робот за допомогою капч. Зазвичай капчі відображаються лише для підозрілих IP-адрес, і з цим допоможуть проксі. В інших випадках використовуйте автоматичний виконувач CAPTCHA – скажімо, 2Captcha або AntiCaptcha. Розпізнаваний персонал найчастіше платний, бо капчі вручну вирішують реальні люди. Тому варто зрозуміти, чи витрати виправдають поставлену мету.

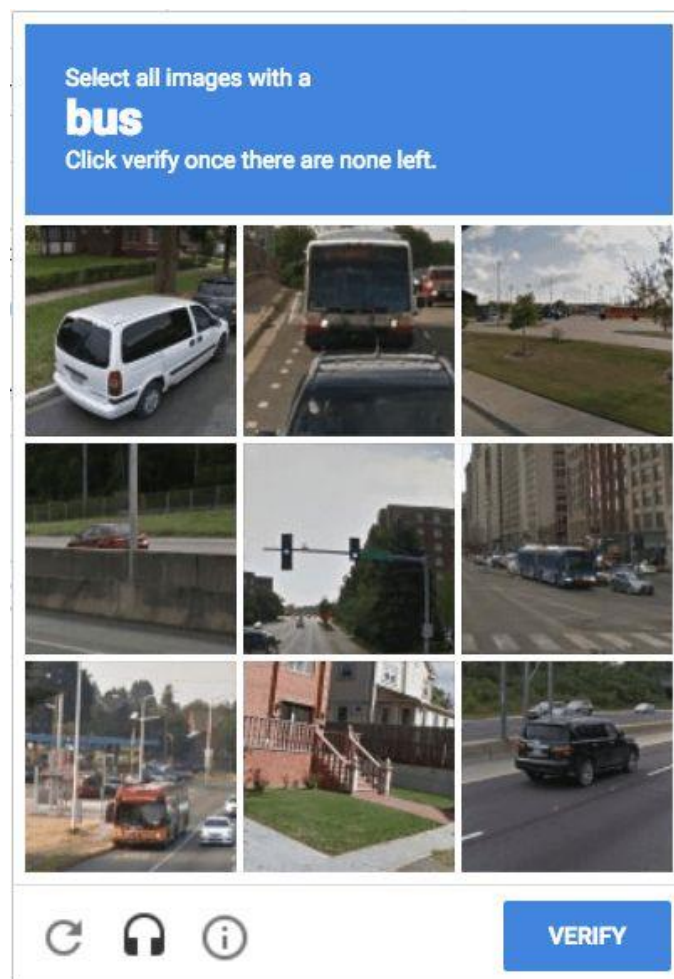


Рисунок 1.3 – Приклад капчі

7. Уникайте honeypot-пасток

"Honeypot" – це фальшиве посилання, яке невидиме для звичайного користувача, але є в HTML-кодi. Як тільки ви почнете аналізувати сайт, honeypot може перенаправити вас на порожні та марні сторінки-приманки. Тому завжди перевіряйте, чи для посилання CSS-властивості встановлені «display: none», «visibility: hidden» або «color: #fff;».

Якщо наслідувати хоча б один метод, шанс бути заблокованим зменшиться в багато разів. Але для вірності краще комбiнувати кілька методів і завжди стежити, щоб краулер не надто навантажував чужі веб-сервери.

1.4 Вітчизняний та зарубіжний досвід застосування веб-додатку який взаємодіє з веб-ресурсом з обмеженнями

Вітчизняний та зарубіжний досвід застосування веб-додатків для організації взаємодії та контролю роботи з веб-ресурсом з обмеженнями демонструє різноманітні рішення та підходи. Нижче наведено деякі приклади таких додатків та їх особливостей:

1. Веб-аналітика та моніторинг

Існують різні веб-додатки для аналізу та моніторингу роботи веб-ресурсів з обмеженнями. Ці програми надають інформацію про завантаження сторінок, час відповіді сервера, помилки доступності та інші метрики, щоб забезпечити контроль та оптимізацію роботи веб-ресурсів. Прикладом цієї програми може бути Google Analytics або Yandex.Metrica.

2. Керування контентом

Веб-програми для керування контентом дозволяють обмежувати доступ до певних розділів або функцій веб-ресурсу залежно від прав та ролей користувачів. Це особливо актуально для сайтів з багатьма авторами або великою кількістю контенту. Прикладом цієї програми може бути система управління контентом (CMS) WordPress.

3. Маркетинг та продажі

Веб-програми для організації маркетингу та продажів на веб-ресурсі можуть пропонувати різні обмеження та контрольні механізми. До них відносяться управління доступом до промо-кодів, налаштування знижок для

певних груп клієнтів, обмеження кількості товарів для продажу та інші функції, які допомагають в управлінні процесом продажу. Прикладом цієї програми може бути платформа електронної комерції Shopify.

4. Системи управління проектами

Веб-застосунки для управління проектами дозволяють обмежувати доступ до завдань, документів та інших ресурсів залежно від ролей та відповідальності учасників проекту. Вони також можуть надавати функціонал для контролю прогресу, розподілу завдань та спільної роботи над проектом. Прикладами таких програм можуть бути Jira або Asana.

Всі ці приклади відображають різні аспекти взаємодії та контролю роботи з веб-ресурсом з обмеженнями. Конкретний вибір веб-застосування залежить від потреб і особливостей конкретної організації.

Конкретний вибір веб-програми для організації взаємодії та контролю роботи з веб-ресурсом з обмеженнями залежить від конкретних цілей та вимог вашого проекту. Важливо враховувати такі чинники як бюджет, масштаб проекту, рівень складності, потреба в інтеграції з іншими системами тощо.

Деякі інші приклади веб-застосунків для організації взаємодії та контролю роботи з веб-ресурсом з обмеженнями включають:

5. Системи керування базами даних

Веб-програми для керування даними можуть запропонувати можливість контролювати доступ до інформації, налаштовувати різні рівні доступу для користувачів, редагувати та оновлювати дані. Прикладом може бути SQL-сервер Microsoft, який надає засоби управління базами даних і контролю доступу до них.

6. CRM-платформи

CRM (Customer Relationship Management) є веб-додатками, які допомагають керувати взаємодією з клієнтами та контролювати продажі та маркетингові зусилля. Вони дозволяють стежити за контактами з клієнтами, управляти угодами, створювати звіти та аналізувати дані. Прикладом може бути Salesforce, одна із провідних CRM-платформ.

7. Веб-портали

Веб-програми для створення веб-порталів дозволяють організовувати інформацію, послуги та функціональність на веб-ресурсі з різними рівнями доступу для користувачів. Це може бути корисним для інтеграції різних функцій, включаючи управління контентом, комунікацію, навчання та багато іншого. Прикладом цієї програми може бути Liferay або Drupal.

8. Інструменти керування проектами

Деякі веб-програми пропонують спеціалізовані інструменти для керування проектами з обмеженнями. Вони дозволяють контролювати прогрес проекту, керувати завданнями та термінами, спільно працювати над документами та іншими ресурсами. Прикладом може бути Trello чи Basecamp.

Кількість та різноманітність веб-додатків для організації взаємодії та контролю роботи з веб-ресурсом з обмеженнями постійно зростає. Важливо визначити свої вимоги та провести аналіз ринку, щоб вибрати відповідний додаток, який відповідає потребам.

1.5 Постановка задачі та опис нового сервісу для веб-додатка який взаємодіє з веб-ресурсом з обмеженнями

Аналізуючи предметну область та структуру веб-програми торгового робота Good Crypto можна приступити до постановки завдання та опису нового сервісу для веб-програми.

Веб-додаток торговий бот Good Crypto тісно взаємодіє з веб-ресурсом API Binance. З'ясувалося, що досить велику проблему веб-додатку завдає саме робота з веб-ресурсом, оскільки він має обмеження. Щоб вирішити цю проблему, необхідно розробити новий сервіс для цієї веб-програми, через яку будуть надсилатися веб-запити на API Binance.

Новий сервіс повинен виступати як сервер із smart-кешуванням веб-запитів. Він повинен приймати веб-запити з інших мікросервісів веб-програми. У тілі веб-запитів будуть передаватися дані про те, куди саме мікросервіс хоче надіслати запит, так само до цих даних належать ключі доступу до API та проміжок часу, за який мікросервіс хоче отримати відповідь. Далі сервіс

вибудовуватиме впорядковану послідовність цих запитів, так звану чергу веб-запитів. В порядку черги запити надсилаються на API Binance, проте в момент надходження нових запитів відбуватиметься кешування. Дані, які будуть отримані на один із веб-запитів з черги, можуть задовольняти результат відповіді і для інших запитів, які прийшли в чергу в момент часу, коли цей запит знаходився в черзі. Тим самим ми скоротимо кількість звернень до веб-ресурсу, збільшимо продуктивність та точність стратегії торгового робота.

Необхідно розробити сервіс, який виступатиме у ролі сервера зі smart-кешуванням веб-запитів до API Binance, з вибудовуванням правильної черги, кешуванням даних та контролем часу очікування на отримання відповіді від сервера.

1.6 Вимоги та архітектура нового сервісу для веб-додатка який взаємодіє з веб-ресурсом з обмеженнями

1.6.1 Дослідження проблем побудови нового сервісу

Побудова нового сервісу для торгового робота із smart-кешуванням веб-запитів для API Binance стикається з декількома проблемами, які слід досліджувати:

1. Обмеження API Binance

Необхідно вивчити документацію API Binance, щоб зрозуміти обмеження кількості запитів в одиницю часу, доступні методи та інші обмеження. Це необхідно для розробки ефективної стратегії кешування та управління запитами.

2. Контроль за даними

Враховуючи, що торгові роботи працюють з фінансовими даними, необхідно забезпечити безпеку та конфіденційність даних. Необхідні методи шифрування та захисту даних, щоб гарантувати їх цілісність та безпеку.

3. Управління кешем

Розробити алгоритми та стратегії управління кешем запитів до API Binance. Створення стратегії з упорядкування запитів та розробка алгоритму з розумного кешування веб-запитів.

4. Масштабованість

Необхідно врахувати можливість масштабування сервісу обробки більшої кількості запитів. Зрозуміти методи горизонтального та вертикального масштабування, щоб забезпечити високу продуктивність та чуйність системи.

5. Тестування та налагодження

Провести тестування та налагодження сервісу, щоб виявити та виправити можливі проблеми та помилки. Використовувати інструменти для автоматичного тестування та моніторингу продуктивності.

Дослідження цих проблем допоможе побудувати надійний та ефективний сервіс для торгового робота із smart-кешуванням для API Binance.

1.6.2 Розробка архітектури для нового сервісу

Архітектура сервісу для торгового бота зі smart-кешуванням складається з мультисховища та потоків. Мультисховище складається з сховищ, які містять впорядковану множини, тобто черги веб-запитів, їх кількість дорівнює кількості ботів, які запросили будь-яку інформацію у API Binance. Кількість створених потоків, що обробляють черги в сервісі, дорівнюватиме кількості створених черг. Кожен потік опрацьовує свою чергу.

Схема архітектури нового сервісу для веб-додатка зі smart-кешуванням веб-запитів до API Binance представлена на рисунку 1.4.

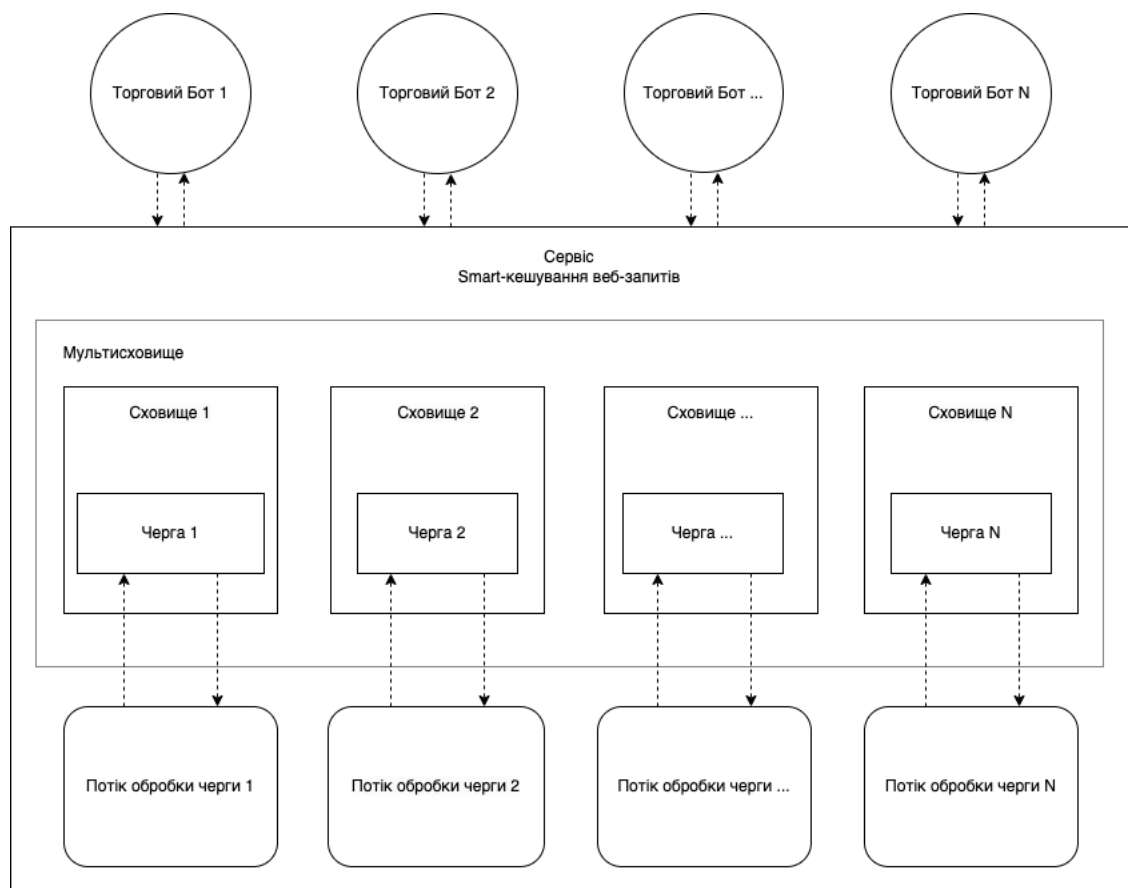


Рисунок 1.4. Архітектура нового сервісу для веб-додатка

Даний сервіс не матиме бази даних або ще якихось сторонніх сховищ. Така архітектура обумовлюється тим, що окремі звернення до бази даних, збір інформації займає певну кількість часу, оскільки ми працюємо з кешуванням запитів і намагаємося прискорити процес отримання даних з API Binance, додавання бази даних буде тільки збільшувати час на отримання даних. Зберігання черги запитів у процесі роботи сервісу немає великих обсягів довгострокової інформації. Черга постійно змінюється і дані, які у відповідь них теж. Відповідно, всі дані про веб-запити зберігаються в черзі в рамках роботи веб-додатка сервісу.

1.6.3 Вимоги до якості розподілення ресурсів у новому сервісі

Важливою складовою нового сервісу для smart-кешування є забезпечення коректної взаємодії між новими сервісами та вже існуючими сервісами. Кожен вже існуючий сервіс спілкується з API Binance зараз безпосередньо, але з впровадженням нового сервісу, всі веб-запити до API відбуватимуться через

новий сервіс. Відповідно всі сервіси будуть налагоджені на те, щоб надсилати запит на новий сервіс за описаним форматом та чекати від нього відповіді отриманих даних з API Binance.

Для організації взаємодії з новим сервісом для кожного вже існуючого сервісу буде створено свій API-ключ. API-ключі сервісів будуть зберігатися окремо в конфігурації API-ключів у новому сервісі.

При кожному зверненні до нового сервісу відбуватиметься перевірка API-ключа і лише потім запит потрапить у чергу веб-запитів, а далі в потік обробки запитів.

Висновок до розділу 1

Исследуя возможные ограничения веб-ресурсов с ограничениями, ознакомившись с ограничениями веб-ресурса API Binance, изучив работу торговых ботов были выявлены сложности по взаимодействию веб-приложений торговых ботов с веб-ресурсом API Binance. Веб-ресурсы могут иметь различные ограничения на веб-запросы к ним. API Binance имеет ограничения на определенное количество запросов в определённый промежуток времени, а также необходимо предоставлять API-ключ криптокошелька для получения информации, связанной с ним.

Для увеличения качества работы торговых ботов необходимо увеличить точность их расчетов, для этого нужно увеличить скорость получения данных. Разработать новый сервис со смарт-кешированием веб-запросов, выстоянием правильной очереди, созданием для каждой очереди своего потока и контролем времени на получения ответа.

РОЗДІЛ 2. ПРОЕКТУВАННЯ АРХІТЕКТУРИ ТА РОЗРОБКА КОНЦЕПЦІЇ НОВОГО СЕРВІСУ ДЛЯ ВЕБ-ДОДАТКА ЯКИЙ ВЗАЄМОДІЄ З ВЕБ-РЕСУРСОМ З ОБМЕЖЕННЯМИ

2.1 Побудова моделі варіантів використання

Сервіс smart-кешування для конкурентного доступу до веб-ресурсів з обмеженнями виступає в ролі проміжного сервера, через який інші сервіси веб-програми будуть спілкуватися з веб-ресурсом з обмеженнями.

Схема моделі варіантів використання нового сервісу для веб-додатка представлена на рисунку 2.1.

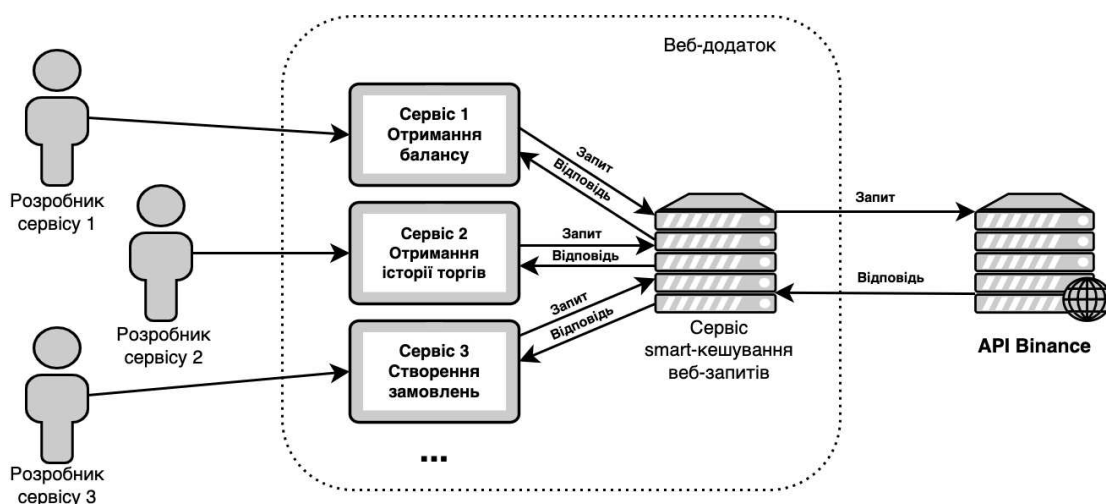


Рисунок 2.1. Модель варіантів використання нового сервісу для веб-додатка

Розробники, які створюють логіку праці ботів звертаються до різних вже написаних сервісів. Сервіси надсилають веб-запити до сервісу smart-кешування веб-запитів відповідно документації з взаємодії з тим сервісом. Далі сервер вибудовує чергу та послідовно надсилає певні веб-запити до API Binance.

Варіант використання сервісу є подібний клієнт-серверній архітектурі, але в даному випадку клієнтами виступають інші сервіси веб-додатку. [2]

2.2 Побудова концептуальної моделі

Визначивши усі вимоги та архітектуру нового сервісу smart-кешування веб-запитів для конкурентного доступу к веб-ресурсу з обмеженнями можна побудувати концептуальну модель виділивши основні об'єкти для роботи сервісу.

Схема концептуальної моделі нового сервісу зі smart-кешуванням веб-запитів представлено на рисунку 2.2.

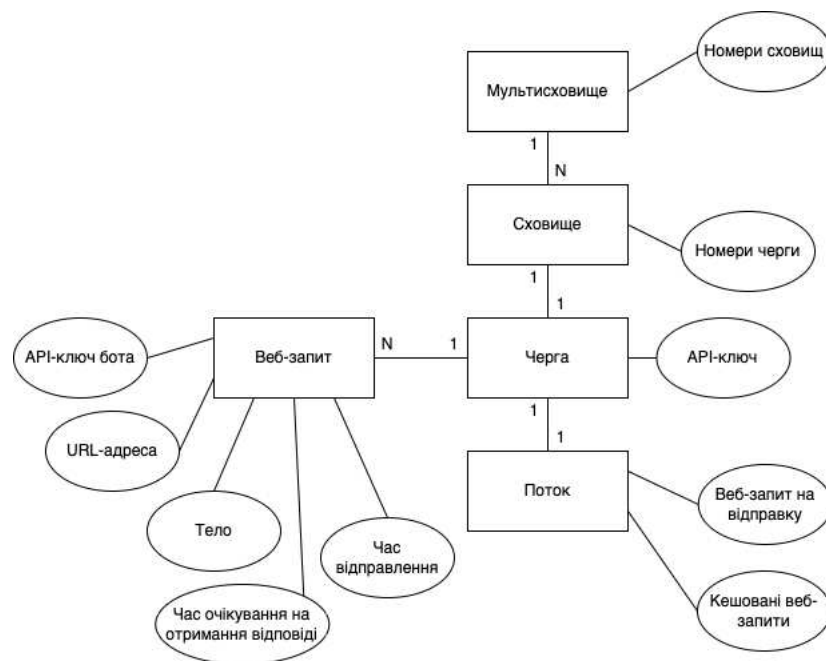


Рисунок 2.2. Концептуальна модель нового сервісу зі smart-кешуванням веб-запитів

Об'єкти та їх атрибути нового сервісу smart-кешування веб-запитів:

1. Мультисховище. Це загальне сховище для усіх сховищ сервісу. Воно об'єднує сховища які пов'язані с чергами різних ботів.

Атрибути: номери сховищ.

2. Сховище. Це об'єкт, який знаходиться у мультисховище та зберігає в собі унікальну чергу для відповідного бота.

Атрибути: Номери черги.

3. Черга. Це об'єкт, який відповідає за впорядкована множину веб-запитів.

Атрибути: API-ключ.

4. Веб-запит. Головний об'єкт, над котрим проводяться усі важливі дії роботи сервісу. Він зберігається у черзі за відповідним сортуванням та має особливий API-ключ бота, який запросив данні з API Binance.

Атрибути: API-ключ, URL-адрес, тіло (об'єкт тіла веб-запиту), час очікування отримання відповіді та час відправлення.

5. Потік. Це об'єкт, котрий постійно працює с чергою та відправляє веб-запити на API Binance.

Атрибути: веб-запит на відправку, кешовані веб-запити.

2.3 Рішення з інформаційного забезпечення

2.3.1 Логічна модель

Концептуальна модель є базою для побудови логічної моделі даних нового сервісу. Логічна модель є більш детальним представленням даних. В ній сутності трансформуються в таблиці даних, атрибути сутностей стають полями таблиць та виділяються відношення між таблицями.

Логічна модель даних нового сервісу зі smart-кешуванням веб-запитів представлена на рисунку 2.3.

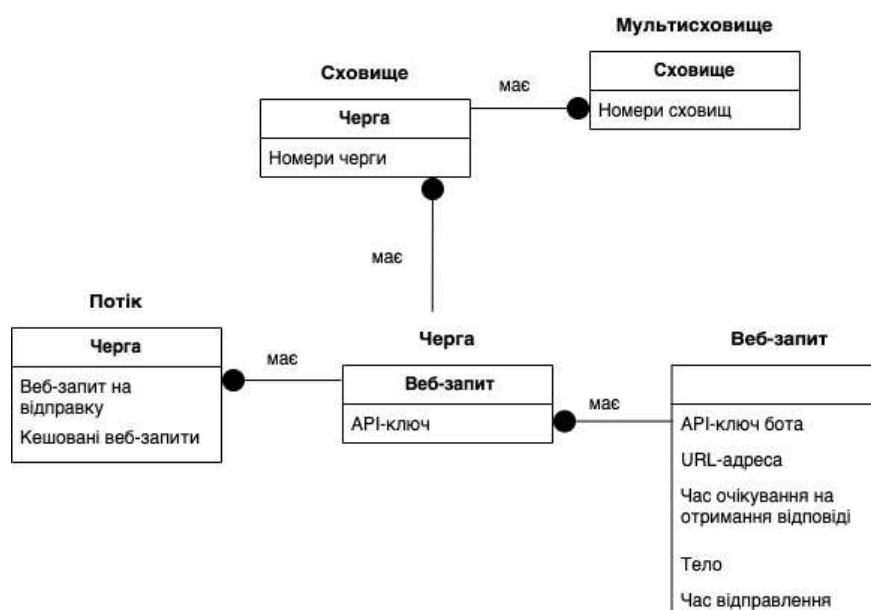


Рисунок 2.3. Логічна модель даних нового сервісу зі smart-кешуванням веб-запитів

2.3.2 Фізична модель

Розробка фізичної моделі даних – це створення схеми даних для нового сервісу. Фізична модель будується з урахуванням логічної моделі та з урахуванням обмежень. Фізична модель даних містить усі деталі, необхідні для створення об’єктів нового сервісу.

Фізична модель даних нового сервісу smart-кешування веб-запитів представлена на рисунку 2.4.

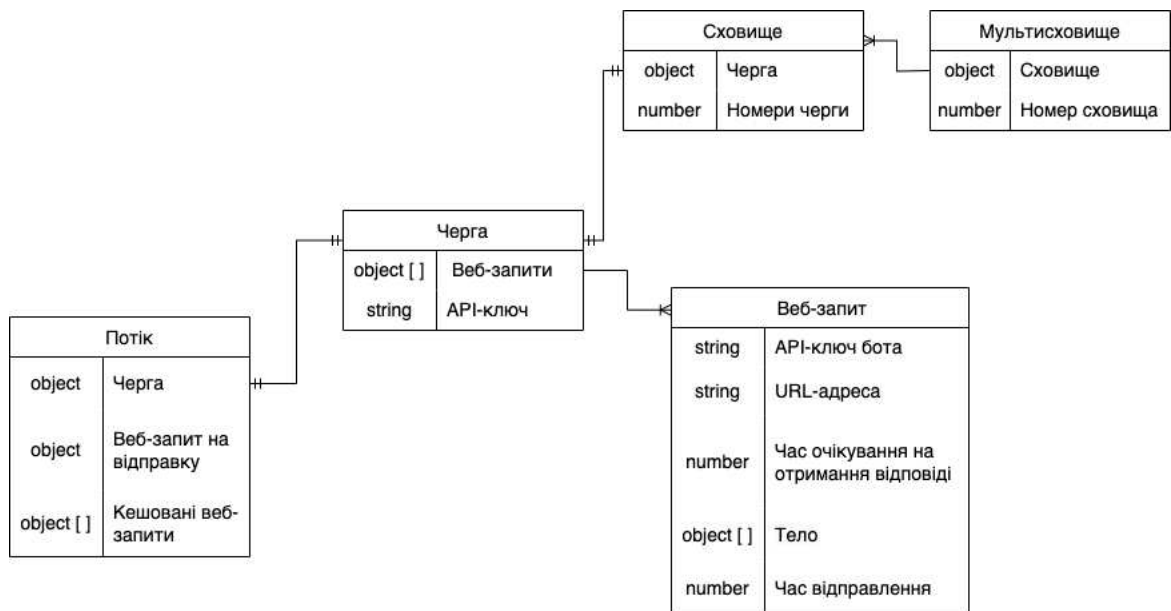


Рисунок 2.4. Фізична модель даних нового сервісу smart-кешування веб-запитів

2.3.3 Діаграма класів

Діаграма класів нового сервісу зі smart-кешуванням веб-запитів представлено на рисунку 2.5.

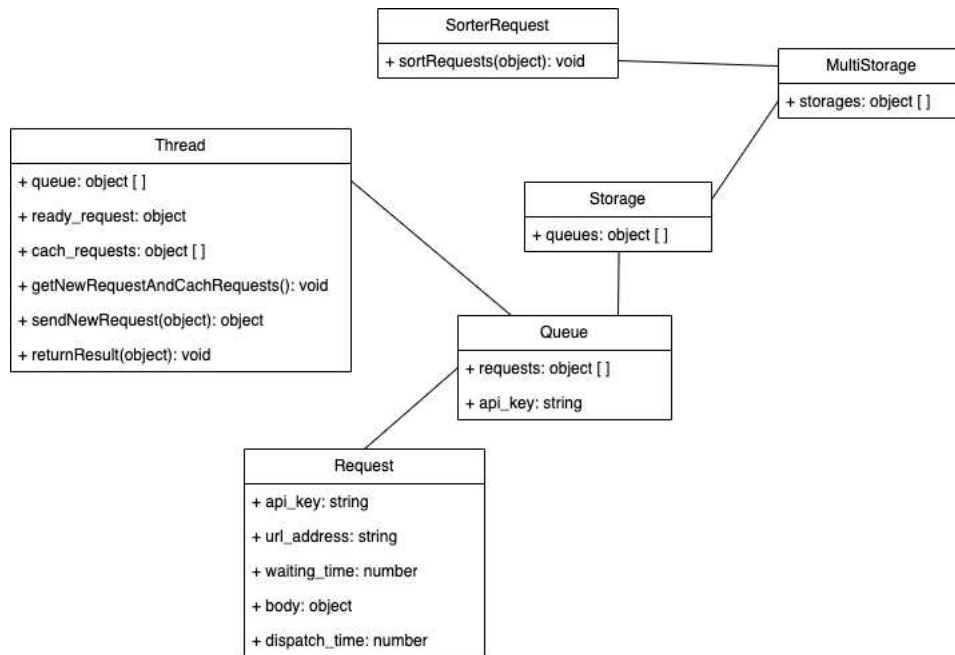


Рисунок 2.5. Діаграма класів нового сервісу зі smart-кешуванням веб-запитів

Діаграма класів визначає типи класів системи та різного роду статичні зв'язки, які існують між ними. На діаграмах класів зображуються також атрибути класів, операції класів та обмеження, що накладаються на зв'язки між класами. У діаграмі класів нового сервісу smart-кешування веб-запитів є шість основних класів: потік, мультисховище, сховище, черга, веб-запит та сортувальник веб-запитів.

Потік має такі атрибути, як черга, веб-запит на відправку та кешовані веб-запити, а також методи: получити новий веб-запит та кешовані веб-запити, надіслати веб-запит, вернути відповідь. Сховище має атрибут черги. Мультисховище має атрибут сховищ. Клас сортувальника веб-запитів має метод, який сортирує запити відповідно API-ключу та часу. Черга має атрибути: веб-запити та API-ключ. Веб-запит має атрибути: API-ключ, URL-адрес, час очікування отримання відповіді, час відправи та тіло для веб-запиту.

2.3.4 Діаграма послідовності

Діаграма послідовності нового сервісу зі smart-кешуванням веб-запитів представлено на рисунку 2.6.

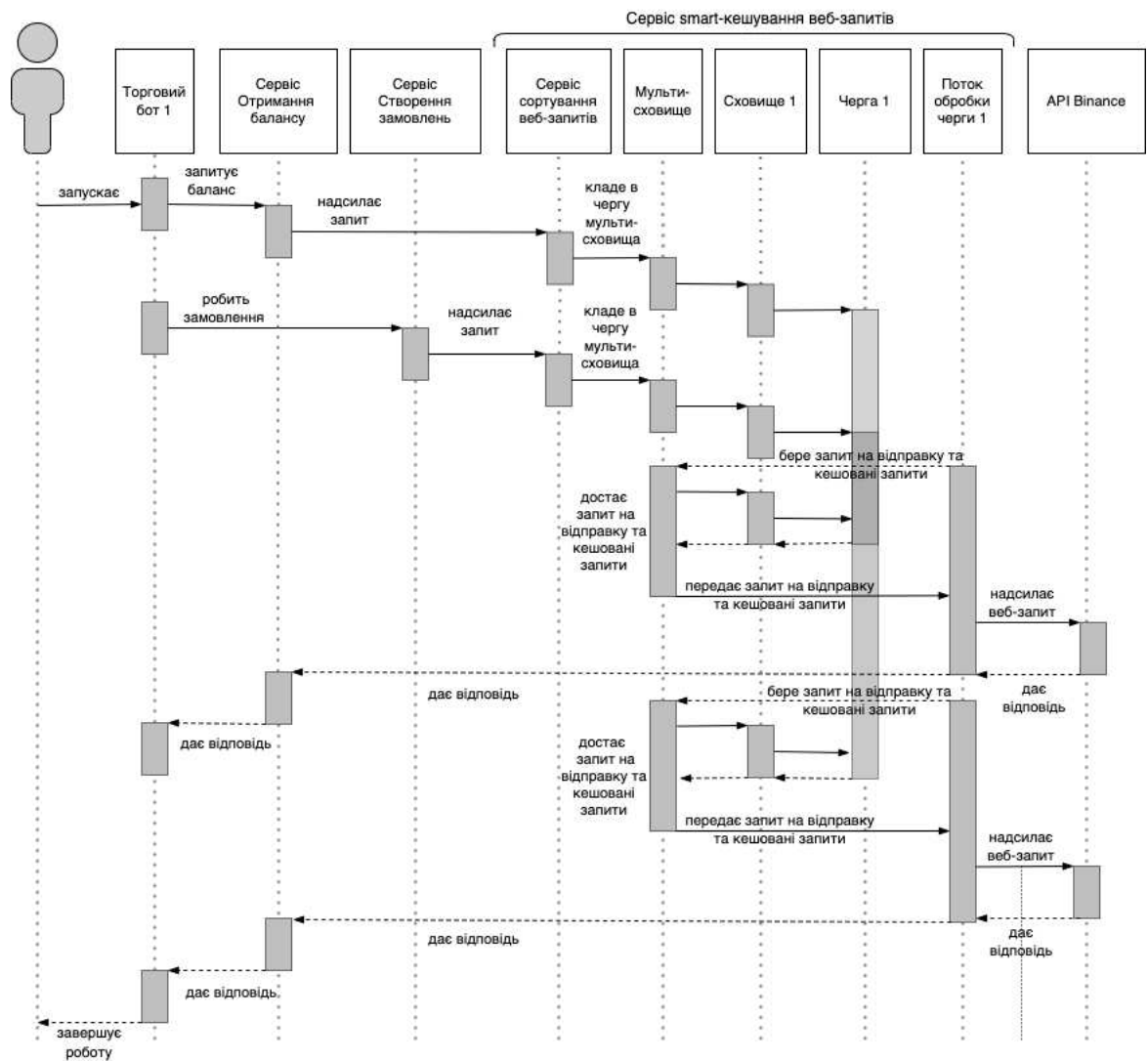


Рисунок 2.6. Діаграма послідовності нового сервісу зі smart-кешуванням веб-запитів

Послідовність процесів нового сервісу smart-кешування веб-запитів відбувається так:

1. Розробник запускає торгового бота, який буде працювати за обраною стратегією.
2. Торговий бот виконує процеси пов'язані з стратегією та звертається до двох сервісів. Перший сервіс отримання балансу, другий створення замовлення.
3. Сервіс отримання балансу та сервіс створення замовлення надсилають веб-запити до сервісу smart-кешування веб-запитів, а якщо точніше до сервісу сортування запитів.

4. Сервіс сортування запитів створює дві черги у відповідних сховищах у мультисховище.

5. Далі потік, який постійно оброблює нові запити, звертається до черги мультисховища, бере веб-запит на отримання балансу на відправку. Надсилає веб-запит до API Binance та очікує відповіді. Дочекавшись відповіді він повертає її до сервісу який відправляв запит на отримання цих даних.

6. Сервіс отримання балансу передає ці данні до торгового бота.

7. Після передачі даних потік одразу звертається до серги мультисховища та бере новий запит на відправку, це веб-запит на створення замовлення.

8. Далі потів надсилає запит до API Binance та очікує відповіді. Дочекавшись відповіді він повертає її до сервісу створення замовлень.

9. Сервіс замовлень передає данні до торгового бота. Торговий бот виконує далі свої процеси та завершує роботу.

2.4 Рішення з технічного забезпечення

Для забезпечення роботи нового сервісу smart-кешування веб-запитів потрібен 5 комп'ютерів з доступом до інтернету. Три комп'ютери для розробників сервісу та два для тестувальників.

Персональний комп'ютер для роботи з системою повинен мати наступні характеристики:

- комп'ютер класу Intel i7/Xeon і вище;
- оперативна пам'ять від 32Гб і вище;
- SSD-диск або HDD-диск, з 30Гб вільного місця (для розрахунків та збереження інформації);
- монітор, миша, клавіатура;
- Wi-Fi роутеру, для підключення до інтернету;
- кабель UTP, для підключення до роутеру;

Схема автоматизації технічного забезпечення представлено на рисунку

2.7.

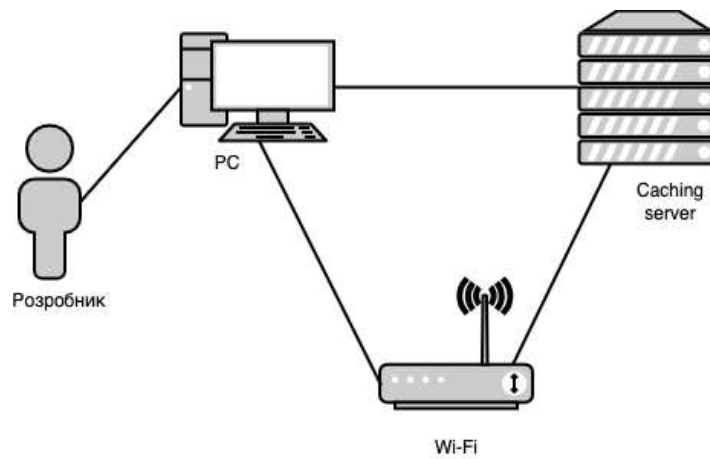


Рисунок 2.7. Схема автоматизації технічного забезпечення

Сервер повинен мати наступні характеристики:

1. Процесор. Рекомендується вибирати сервер із потужним багатоядерним процесором, оскільки сервіс кешування досить інтенсивний з погляду обчислень. Наприклад, сервер із процесором Intel Xeon або AMD EPYC серії може бути хорошим вибором.

2. Оперативна пам'ять (ОЗП). Велика ОЗП дозволить серверу зберігати велику кількість даних у кеші, збільшуючи ефективність роботи сервісу. Рекомендується вибирати сервер з щонайменше 32 ГБ ОЗУ. Якщо бюджет дозволить, можна розглянути сервер із ще більшою ОЗУ.

3. Сховище даних. Для зберігання кешу необхідно вибрати сервер із високошвидкісним та надійним сховищем даних. SSD-накопичувачі краще, оскільки вони забезпечують більш високу швидкість читання та запису, що підвищує продуктивність сервісу.

4. Мережевий інтерфейс. Сервер повинен бути оснащений кількома мережевими інтерфейсами, щоб забезпечити високу пропускну здатність та стійкість зв'язку. Рекомендується використовувати сервер із підтримкою Gigabit Ethernet або навіть 10-Gigabit Ethernet.

5. Система охолодження. Сервер працюватиме тривалий час і, ймовірно, з високим навантаженням. Тому необхідно вибирати сервер з ефективною системою охолодження, щоб запобігти перегріву та забезпечити стабільну роботу.

Для успішного функціонування нового сервісу smart-кешування веб-запитів досить цього наявного технічного забезпечення.

2.5 Рішення з математичного забезпечення

Загалом в роботі нового сервісу smart-кешування не має процесів пов'язаних з математичними розрахунками. Єдине що потрібно врахувати це роботу з часом. В об'єкті веб-запиту буде враховуватися два часи. Перший, це час відправлення веб-запиту, час коли веб-запит відправився на новий сервіс. Це потрібно для упорядкування множини веб-запитів. Другий час, це час протягом якого сервіс який відправив веб-запит буде чекати відповіді. З двома часом легше буде працювати, якщо вони буду фіксовані у секундах. Тобто перший час, час відправлення веб-запиту буде фіксуватися в форматі Unix time. Unix time, також POSIX-час) – система опису моментів у часі, прийнята в Unix та інших POSIX-сумісних операційних системах. Визначається як кількість секунд, що минули з півночі (00:00:00 UTC) 1 січня 1970 (четвер); цей момент називають «епохою Unix» (англ. Unix Epoch). А другий час, час протягом якого сервіс який відправив веб-запит буде чекати відповіді буду переводитися до секунд завдяки звичайного множення годин на хвилини, хвилини на секунди.

2.6 Рішення з програмного забезпечення

Для розробки сервісу smart-кешування веб-запитів будуть використатися дві основні програми:

- Docker
- Visual Studio Code

Docker – це платформа для розробки, доставки та запуску контейнерних програм. Docker дозволяє створювати контейнери, автоматизувати їх запуск та розгортання, керує життєвим циклом. Він дозволяє запускати безліч контейнерів на одній машині.

Контейнеризація схожа на віртуалізацію, але це не одне й те саме. Віртуалізація запускає повноцінний хост на гіпервізорі зі своїм віртуальним

обладнанням та операційною системою. При цьому всередині однієї ОС можна запустити іншу ОС. У разі контейнеризації процес запускається прямо з ядра основної операційної системи та не віртуалізує обладнання. Це означає, що контейнеризоване додаток може працювати тільки в тій самій ОС, що й основна. Контейнери не віртуалізують обладнання, тому споживають менше ресурсів.

У випадку з новим сервісом Docker необхідний для запуску сервісу локально на комп'ютерах розробників та тестувальників.

Visual Studio Code або VS Code – потужний редактор коду, призначений для різних мов розробки. Гнучкий і зручний, має велику кількість функцій. Visual Studio C – це продукт від корпорації Майкрософт. Інструмент, призначений для верстальників та розробників. Один із найпопулярніших у відповідній ніші.

Є редактором коду від Microsoft, який виступає «полегшеною» інтерпретацією Visual Studio. За допомогою нього можна не лише займатися написанням додатків. Visual Studio Code підтримує велику кількість плагінів, які дозволяють "розігнати" редактор до повноцінного середовища програмування.

Доглядаючи мови програмування початковий вибір упав на JavaScript, однак для зручності розробки обрана мова TypeScript. TypeScript – це строго типізована мова програмування, побудована на JavaScript. Розробники додали до нього додаткові можливості, такі як статична типізація, класи та модулі, щоб створювати більш надійні та підтримувані програми.

TypeScript дозволяє виявляти помилки на етапі розробки, полегшує спільну роботу в команді та покращує продуктивність розробки у великих проектах. Він перетворюється на звичайний JavaScript, тому код, написаний на TypeScript, може виконуватися в будь-якому сучасному браузері або оточенні, що підтримує JavaScript.

JavaScript – це однопотокова мова програмування. Це означає, що він є тільки один потік виконання і один стек, який містить функції в чергу на виконання. Отже в один момент часу JavaScript може виконати тільки одну операцію, інші операції при цьому чекатимуть на свою чергу в стеку, поки їх не викличуть.

Реактивне програмування – парадигма програмування, орієнтована на потоки даних і поширення змін. Потік – це масив даних, відсортованих за часом, який може повідомляти про те, що дані змінилися. RxJS – це бібліотека для роботи з асинхронними та заснованими на подіях програмами з використанням послідовностей, що спостерігаються. Бібліотека надає основний тип Observable, кілька допоміжних типів (Observer, Schedulers, Subjects) та оператори роботи з подіями як з колекціями (map, filter, reduce, every та подібні до JavaScript Array).

Також потрібна бібліотека для роботи з часом в JavaScript – це Luxon. Бібліотека Luxon є інструментом для роботи з датами та часом на мові JavaScript. Вона надає зручні методи для парсингу, форматування, маніпулювання та відображення дат та часу. Luxon є легковагою та ефективною бібліотекою, що має зрозумілий API. Вона дозволяє спростити роботу з датами і часом у JavaScript і вважається одним з найбільш відданих інструментів для цих завдань.

Висновок до розділу 2

Підбиваючи підсумки другого розділу потрібно відмітити що були побудовані схеми моделі варіантів використання, схема концептуальної моделі, логічна та фізична модель даних, діаграма класів та діаграма послідовності процесів. Завдяки побудові схем стало чітко виявлена архітектура нового сервісу. Розподілені головні об'єкти їх атрибути та методи. В діаграмі послідовності процесів чітко видно місце нового сервісу smart-кешування веб-запитів у веб-додатку торгового бота.

В процесі були виявленні рішення щодо технічного, математичного та програмного забезпечення. В враховуючи варіанти використання нового сервісу виявили технічне обладнання для розробників веб-додатку та обрати в якості програмного забезпечення такі програми, як Docker та Visual Studio Code. Також програмною мовою було обрано мову TypeScript з використанням бібліотеки RxJS та Luxon.

РОЗДІЛ 3. РОЗРОБКА ПРОЕКТНИХ РІШЕНЬ ЩОДО НОВОГО СЕРВІСУ ДЛЯ ВЕБ-ДОДАТКА ЯКИЙ ВЗАЄМОДІЄ З ВЕБ-РЕСУРСОМ З ОБМЕЖЕННЯМИ

3.1 Загальносистемні рішення

3.1.1 Представлення схеми організаційної структури

Організаційна структура компанії Good Crypto представлена на рисунку 3.1.

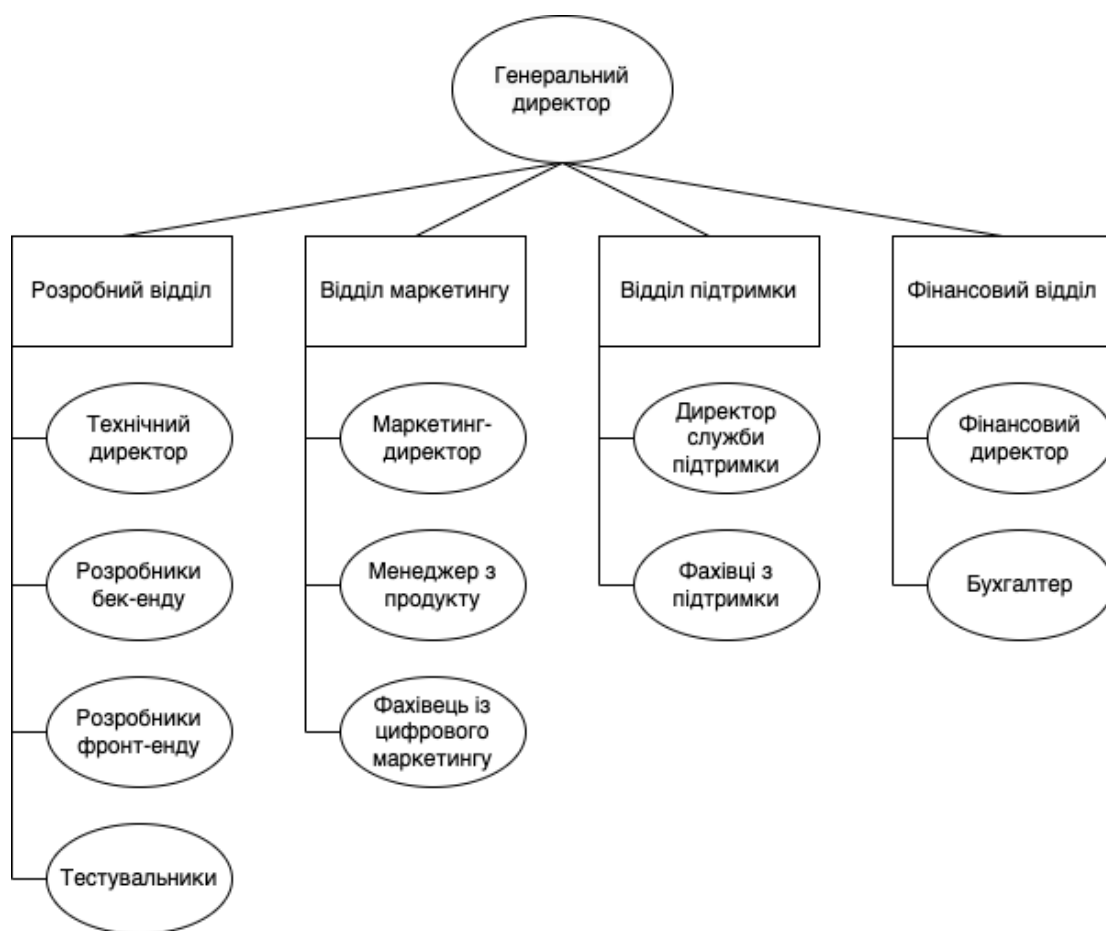


Рисунок 3.1. Організаційна структура компанії Good Crypto

1. Генеральний директор

Генеральний директор відповідає за загальне управління та стратегічний розвиток компанії.

2. Розробний відділ

- Технічний директор

Технічний директор відповідає за управління розробкою та технічними аспектами проекту.

- Розробники backend

Ці розробники займаються написанням та підтримкою серверного програмного забезпечення, а також забезпечують його безпеку та масштабованість.

- Розробники frontend

Ці розробники створюють користувацький інтерфейс веб-додатку та клієнтську частину, приділяючи увагу зручності використання та привабливому дизайну.

- Тестувальники

Тестувальники забезпечують якість розробки через тестування та налагодження програмного забезпечення, а також перевіряють відповідність функціональних та нефункціональних вимог.

3. Відділ маркетингу

- Маркетинг-директор

Маркетинг-директор планує маркетингову стратегію, здійснює просування продукту та вибудовує комунікацію з клієнтами.

- Менеджер з продукту

Менеджер з продукту аналізує ринок та зворотний зв'язок користувачів, щоб оптимізувати продукт та врахувати потреби користувачів.

- Фахівець із цифрового маркетингу

Фахівець із цифрового маркетингу управляє соціальними мережами, рекламними кампаніями та залученням користувачів через різні онлайн-канали.

4. Відділ підтримки

- Директор служби підтримки

Директор служби підтримки забезпечує якісне надання підтримки клієнтам, а також координує роботу фахівців з підтримки.

- Фахівці з підтримки

Фахівці з підтримки відповідають за оперативне реагування на питання та проблеми користувачів, а також допомагають користувачам вирішити непорозуміння.

5. Фінансовий відділ

- Фінансовий директор

Фінансовий директор займається фінансовим плануванням, веде облік та аналіз фінансових показників компанії.

- Бухгалтер

Бухгалтер відповідає за ведення фінансової звітності та бухгалтерський облік, забезпечуючи фінансову прозорість та облік активів компанії.

У цій структурі кожен відділ та співробітник відіграє певну роль, необхідну для забезпечення ефективного функціонування компанії та успіху проекту Good Crypto.

3.1.2 Представлення схеми функціональної структури веб-додатку з новим сервісом smart-кешування веб-запитів

Функціональну структуру веб-додатку з новим сервісом зі smart-кешуванням веб-запитів представлено на рисунку 3.2.

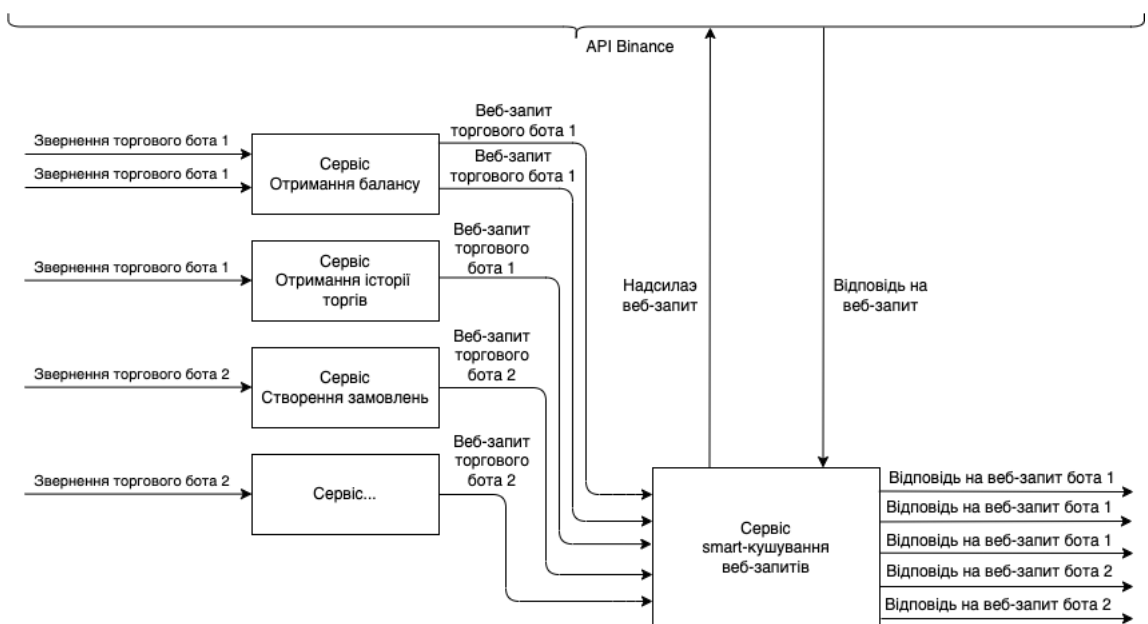


Рисунок 3.2. Функціональна структура веб-додатку з новим сервісом зі smart-кешуванням веб-запитів

Послідовність дії функціонування веб-додатку з новим сервісом smart-кешування веб-запитів:

1. Є боти, які звертаються до окремих сервісів.
2. Окремі сервіси, які створенні в веб-додатку кожен під своє завдання відправляють запити до нового сервісу.
3. Новий сервіс, який робить кешування веб-запитів, надсилає веб-запити до API Binance, чекає, отримує відповіді та відповідає сервісам на їх веб-запити.

Функціональну структуру нового сервісу зі smart-кешуванням веб-запитів представлено на рисунку 3.3.

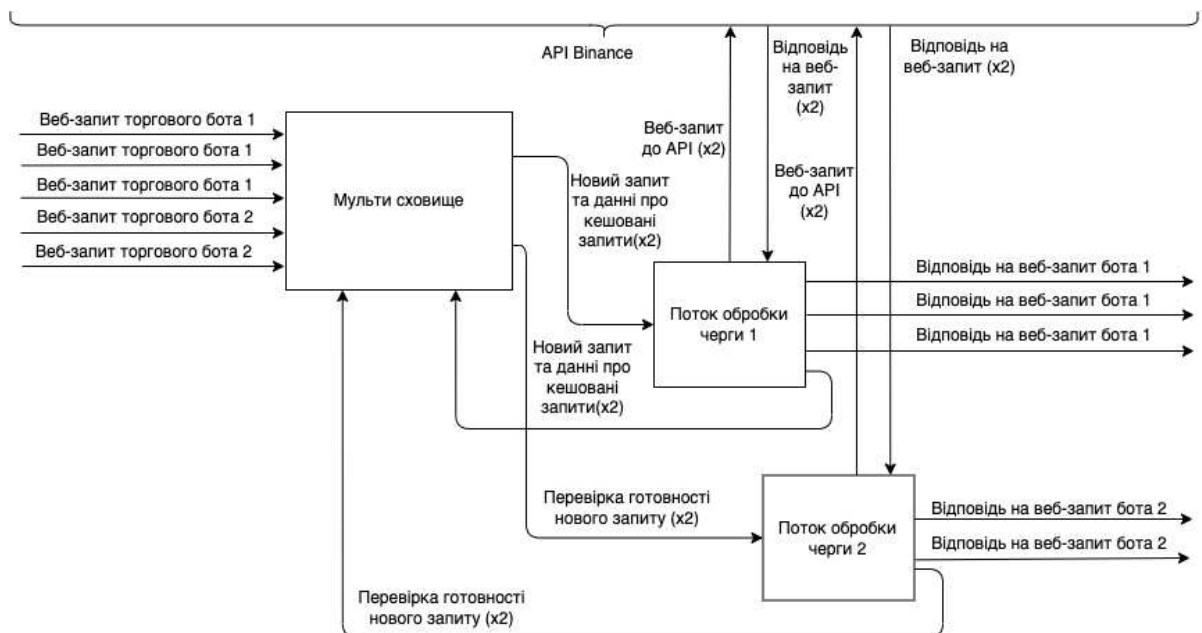


Рисунок 3.3. Функціональна структура нового сервісу зі smart-кешуванням веб-запитів

Функціональна структура нового сервісу smart-кешування веб-запитів налагоджена наступним чином. Спочатку усі веб-запити потрапляють у мультисховище, далі для кожного сховища створюється окремий потік обробки черги веб-запитів. Кожен потік один раз в інтервал заданого часу звертається до

сховища і бере наступний веб-запит на відправку, а також дані про веб-запити, яким буде задовільна відповідь відправленого запиту. Потім робить веб-запит на API Binance, чекає та дає відповідь відповідним сервісам.

Функціональну структуру мультисховища для нового сервісу smart-кешування веб-запитів представлено на рисунку 3.4.

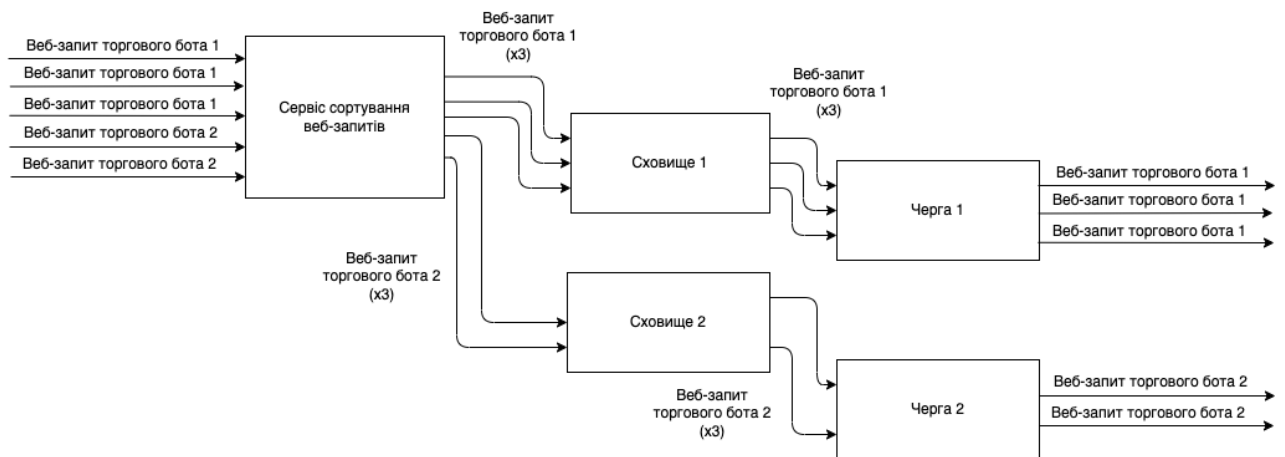


Рисунок 3.3. Функціональна структура мультисховища нового сервісу

Функціональна структура мультисховища нового сервісу має сервіс сортування веб-запитів та сховища, які мають черги. Для кожного бота створюється своє сховище з чергою, це закладено логікою сервісу. Кожен бот має свій API-ключ за яким кожен потік надсилає веб-запити до API Binance. Тому було вирішено, що кожен бот матиме свою чергу веб-запитів та свій потік обробки цих веб-запитів.

3.1.3 Описання функцій, що автоматизуються

Головним завданням, яке виконуватиме новий сервіс, є розумне кешування веб-запитів веб-програми торгового бота Good Crypto. Раніше кожен сервіс веб-програми окремо відправляли веб-запити через заданий інтервал. З додаванням нового сервісу весь контроль веб-запитів до веб-ресурсу буде у ньому. Інші сервіси звільняться від логіки створення послідовності веб-запитів та логіки спілкування з іншими сервісами для безпечного надсилання веб-запиту до веб-ресурсу.

Кожен уже існуючий сервіс, який працює з API Binance, надсилатиме запити на новий створений сервіс. У тілі запиту будуть передаватися дані, про те куди необхідно надіслати запит, API-ключ із яким необхідно надіслати запит, дані для запиту, який необхідно надіслати, і час, за який він очікує отримати відповідь.

3.1.4 Загальний опис нового сервісу

Сервіс smart-кешування для конкурентного доступу до веб-ресурсів з обмеженнями виступає в ролі проміжного сервера, через який інші сервіси веб-програми будуть спілкуватися з веб-ресурсом з обмеженнями.

Під час аналізу правильної роботи з веб-ресурсом API Binance з обмеженнями було виявлено, що веб-запити можна надсилати певну кількість разів і з певним інтервалом часу, а також індивідуальним API-ключом торгового робота. Для цього створюється окрема черга веб-запитів для кожного торгового робота, а також окремий потік для її обробки.

Впорядкована безліч веб-запитів це і є та сама черга, з якою доведеться працювати потоку. Коли один із сервісів надсилає веб-запит до нового сервісу, насамперед дані веб-запиту потрапляють один із механізмів сортування веб-запитів нового сервісу, а сам новий сервіс тримає з'єднання відкритим доти, доки не буде готова відповідь.

Механізм сортування веб-запитів спочатку дивиться на отримані дані про веб-запит і ідентифікує його по API-ключу, що передається. Визначившись, що це за веб-запит, механізм сортування кладе дані про веб-запит у властиву йому чергу і додає необхідних два поля для більш детальної ідентифікації. Перше поле час, коли запит було отримано найновішим сервісом, друге поле час, коли веб-запит повинен відправитися, враховуючи інтервал часу, в який можна буде надіслати веб-запит до веб-ресурсу з обмеженнями. Далі таким чином набирається черга запитів, коли сервіси кілька разів звертаються до нового сервісу від одного торгового бота. З цією чергою безперервно працює потік, що її обробляє.

Потік бере з черги веб-запит, перевіряє чи немає таких веб-запитів, якщо є теж забирає їх з черги. Отримавши дані про запит, який необхідно надіслати на веб-ресурс з обмеженнями та дані про інші такі ж запити, потік надсилає запит до веб-ресурсу API Binance. Очікуючи відповіді, потік маючи дані про веб-запити, які прийшли віддає відповідь, що прийшов з API Binance і після цього знову приступає до обробки черги та інших веб-запитів.

Заснована складність розробки даного сервісу полягає у побудові правильної логіки та обліку всіх деталей завантаження, очікування та отримання даних, пов'язаних із веб-запитами. Однак із вибудовуванням логічних закономірностей очікувань роботи нового сервісу, розробка сервісу стає контрольованою та послідовною.

3.2 Рішення з організаційного забезпечення

Ознайомившись з організаційною структурою компанії Good Crypto, можна дійти невтішного висновку що у компанії немає команди розробки та підтримки даного сервісу. Необхідно набрати команду з трьох розробників та двох тестувальників. Розробники спочатку пророблять роботу зі створення даного сервісу, а далі підтримуватимуть даний сервіс. Сервіс smart-кешування завжди потребуватиме підтримки, оскільки він повинен працювати без перебоїв, і розробники повинні контролювати всі зміни в роботі з API Binance та миттєво реагувати на внесення нових обмежень на веб-запити до веб-ресурсу, а також доопрацьовувати та вдосконалювати можливості роботи із сервісом.

Тестувальники необхідні як для початкового тестування, так і для подальшої роботи нового сервісу. Як згадано вище, сервіс може допрацьовуватися і змінюватися, у зв'язку з цим йому буде потрібно щоразу нове тестування, чим і займатимуться тестувальники цієї команди. Крім цього, тестувальники відіграватимуть важливу роль у написанні документації для використання даного сервісу.

На кожен сервіс, що вже існує, є команда розробників, а десь і не одна. Їм необхідно буде змінити логіку роботи з веб-запитами у своїх сервісах та почати

працювати з ними через новий сервіс. Тому для розробників, які будуть працювати з API потрібна чітка документація по роботі з новим сервісом smart-кешування веб-запитів. Це входить до обов'язків тестувальників команди, що розробляє даний сервіс.

Висновок до розділу 3

Підсумовуючи цього розділу, важливо відзначити, що проектне рішення, що стосується нового сервісу для організації взаємодії та контролю роботи веб-додатка з веб-ресурсом з обмеженнями, засноване на аналізі предметної області, виявленні складнощів, з якими зіткнулося веб-додаток торговий бот Good Crypto, на дослідженні компанії Good Crypto та роботи системи веб-додатку торговельного бота.

Для впровадження даного проекту в компанію, організаційна структура компанії розшириться ще однією командою розробників, яка відповідатиме за розробку та підтримку цього проекту.

Проект для впровадження є сервіс smart-кешування для конкурентного доступу до веб-ресурсів з обмеженнями. Він виступає в ролі проміжного сервера, через який інші сервіси веб-програми будуть спілкуватися з веб-ресурсом з обмеженнями.

РОЗДІЛ 4. РОЗРАХУНОК ТА ОЦІНКА ЕКОНОМІЧНОГО ЕФЕКТУ ВІД РОЗРОБКИ НОВОГО СЕРВІСУ ДЛЯ ВЕБ-ДОДАТКА ЯКИЙ ВЗАЄМОДІЄ З ВЕБ-РЕСУРСОМ З ОБМЕЖЕННЯМИ

4.1 Загальна характеристика роботи

Торговий бот з обмеженням за кількістю веб-запитів на Binance та без обмежень за кількістю запитів мають свої переваги та недоліки з економічної точки зору.

З обмеженням кількості веб-запитів на Binance:

Переваги:

1. Економічна ефективність. Обмеження за кількістю веб-запитів дозволяє знизити витрати на використання API Binance. Якщо робота надсилає безліч запитів, обмеження допомагає скоротити використання ресурсів і, отже, економити гроші.

2. Управління ризиками. Обмеження за кількістю запитів може допомогти запобігти надмірній активності торгового бота та встановлювати обмеження за розміром позицій або частотою торговельних операцій для зменшення потенційних втрат.

3. Дотримання правил біржі. Обмеження за кількістю запитів допомагає дотримуватись правил та обмежень, встановлених Binance, що може запобігти негативним наслідкам, таким як блокування облікового запису.

Недоліки:

1. Обмеження можливостей. Обмеження за кількістю запитів обмежує активність торгового робота, що може обмежити його здатність оптимально реагувати зміни у ринкових умовах.

2. Пропущені можливості. Обмеження може призвести до пропуску можливостей для виконання вигідних торгових операцій, особливо в умовах ринку, що швидко змінюється.

Без обмежень щодо кількості запитів:

Переваги:

1. Велика свобода дій. Без обмежень кількості запитів торговий бот має велику свободу дій, оскільки може взаємодіяти з API Binance без будь-яких обмежень.

2. Швидка реакція на ринкові зміни. Без обмежень кількості запитів бот може миттєво реагувати зміни ринкових умов і виконувати операції у режимі реального часу.

Недоліки:

1. Додаткові витрати. Без обмежень кількості запитів потрібна більша кількість ресурсів і пропускну здатність підтримки високої активності бота. Це може спричинити додаткові витрати на сервери, хостинг або преміум-підписки на API Binance.

2. Ризик перевантаження або блокування. Висока активність без обмежень може призвести до перевантаження API Binance або блокування облікового запису бота через порушення правил платформи.

Отже, вибір між торговим ботом з обмеженням кількості веб-запитів на Binance і без обмежень кількості запитів залежить від пріоритетів проекту, доступного бюджету і торгових стратегій.

4.2 Розрахунок витрат на проектування та впровадження системи

4.2.1 Техніко-економічний аналіз

У техніко-економічний аналіз впровадження нового сервісу для кешування веб-запитів торгового робота Good Crypto до API Binance входять такі пункти:

1. Вимоги. Сервіс кешування повинен кешувати дані про курси валют, історичні дані про торгівлю, а також запити на отримання інформації про портфель та баланс користувача. Це допоможе знизити навантаження на сервери біржі та прискорити виконання запитів торгового робота.

2. Архітектура. Для забезпечення горизонтального масштабування у веб-додатку буде розроблено окремий сервіс.

3. Інтеграція. Для інтеграції сервісу кешування з торговим ботом Good Crypto впроваджується відповідна логіка кешування в код бота. При цьому

враховується Binance API біржі та визначається, які запити можуть бути ефективно кешовані. Наприклад, інформація про курси валют може рідко змінюватися, тому її можна закешувати на тривалий термін. Однак дані про баланс користувача можуть змінюватися частіше, тому необхідно розробити механізм оновлення кешу за необхідності.

4. Інвестиції. Оцінюючи витрати, пов'язані з розробкою та використанням сервісу кешування, необхідно врахувати витрати на розробку програмного забезпечення, придбання нового сервера, сховища даних, а також навчання розробників і підтримку сервісу.

5. Зручності. Переваги, які принесе новий сервіс – це поліпшення продуктивності робота, скорочення часу відповіді на запити, збільшення загальної ефективності торгового алгоритму, а також зниження навантаження на сервери та мінімізація можливих помилок, пов'язаних із зверненням до API біржі.

6. Ризики. Ідентифікуючи можливі ризики та перешкоди, пов'язані з впровадженням нового сервісу у веб-додаток, можна винести такі пункти:

- Уразливості безпеки

Новий сервіс, що працює як окремий сервер, може стати мішенню для потенційних атак. Незахищений або неправильно налаштований сервіс може стати точкою входу для злоумисників. Тому вкрай важливо гарантувати безпеку цього сервісу, включаючи захист від вразливостей, використання методів автентифікації та авторизації, що добре зарекомендували себе, а також моніторинг і реагування на потенційні атаки.

- Стійкість та доступність

Будучи окремим сервером, цей новий мікросервіс повинен мати надійні механізми, що забезпечують його стійкість та доступність. Непередбачені збої в роботі можуть призвести до недоступності сервісу і, як наслідок, призвести до втрати функціональності. Для мінімізації ризику необхідно здійснювати регулярне резервне копіювання даних, розробляти механізми резервування та відновлення, а також встановлювати моніторинг сервісу для своєчасного виявлення та усунення проблем.

- **Масштабованість**

Якщо новий мікросервіс почне отримувати велику кількість запитів або веб-додаток вимагає масштабування, ризик полягає в тому, що сервер не буде здатний обробляти всі запити вчасно або необхідним чином. Тому необхідно проектувати сервіс з урахуванням масштабованості, використовувати горизонтальне масштабування за потреби та налаштовувати систему балансування навантаження.

- **Залежність та взаємодія з іншими сервісами**

Якщо новий мікросервіс буде взаємодіяти з іншими компонентами веб-програми, ризик полягає в несумісності або помилках при взаємодії. Необхідно провести ретельне тестування взаємодії з іншими сервісами, а також розробити механізми обробки помилок та контролю цілісності даних під час передачі між сервісами.

- **Обслуговування та технічна підтримка**

Використання окремого сервера у вигляді мікросервісу означає, що потрібне додаткове керування, обслуговування та технічна підтримка. Необхідно мати надійний механізм оновлення та обслуговування сервісу, систему моніторингу для виявлення та усунення помилок, а також кваліфікований персонал для вирішення проблем, що виникають.

4.2.2 Вартісний аналіз

До оцінки витрат на проектування та впровадження нового сервісу, можна врахувати такі основні витрати:

1. Витрати на розробку – кваліфіковані розробники, фахівці з тісно пов'язаних технологій, тестувальники.
2. Витрати навчання та пристосування наявних систем.
3. Витрати на додаткове обладнання або програмне забезпечення.
4. Витрати на тестування.

Приблизний розрахунок витрат на проектування та впровадження нового сервісу smart-кешування веб-запитів до API Binance для веб-програми торгового бота Good Crypto в українській валюті можна провести наступним чином:

1. Витрати розробку:

- Оклади розробників: 50 000 грн/місяць (3 розробники протягом 4 місяців)

Разом витрати на розробку = 550 000 грн

2. Витрати на навчання та пристосування наявних систем:

- Вартість навчання персоналу роботи з новим сервісом
- Пристосування системи до проекту

Разом витрати на навчання та пристосування систем = 100 000 грн

3. Витрати на додаткове обладнання та програмне забезпечення:

- Придбання необхідного нового серверу
- Обладнання для персоналу та програмне забезпечення

Разом витрати на обладнання та програмне забезпечення = 150 000 грн

4. Витрати на тестування:

- Тестувальники: 30 000 грн/місяць (2 тестувальника протягом 2 місяців) = 60 000 грн

Разом витрати на тестування = 60 000 грн

Приблизні витрати на проектування та впровадження нового сервісу smart-кешування становлять: 550 000 грн + 100 000 грн + 150 000 грн + 60 000 грн = 860 000 грн.

Виходячи з наданого, можна зробити такі висновки про витрати на проектування та впровадження нового сервісу. Необхідно врахувати витрати на оплату кваліфікованих розробників, фахівців з технологій та тестувальників. Це важливий аспект, оскільки якість розробки та функціональність сервісу залежить від кваліфікації цих фахівців. Витрати навчання та пристосування наявних систем. Впровадження нового сервісу потребує навчання персоналу до роботи з ним. Також знадобиться пристосування існуючих систем до нового сервісу. Витрати на додаткове обладнання чи програмне забезпечення. Для впровадження

нового сервісу потрібно придбати додаткового обладнання або програмного забезпечення. Це включає покупку сервера, оновлення програмного забезпечення або інші необхідні ресурси.

Витрати тестування. Тестування нового сервісу також потребує витрат. Необхідно врахувати оплату тестувальників та час, витрачений на проведення тестів. Зазначені аспекти є основними витратами, які можуть виникнути при проектуванні та впровадженні нового сервісу смарт-кешування веб-запитів.

4.3 Розрахунок економічної ефективності розробки і впровадження системи

При розрахунку прибутковості торгового робота важливо враховувати такі фактори:

1. Точність стратегії. Торгова стратегія робота грає ключову роль його прибутковості. Якщо стратегія заснована на аналізі ринку і має високу точність у визначенні точок входу та виходу з угод, можна очікувати більш високої прибутковості.

2. Розмір інвестицій. Розмір інвестицій, які ви готові вкласти в торгового робота, також впливає на його прибутковість. Чим більше коштів ви готові інвестувати, тим більший потенційний прибуток.

3. Волатильність ринку. Волатильність ринку може надати більше можливостей для прибуткових угод, але й несе ризик збитків. Висока волатильність може бути сприятливим фактором для торгівлі, особливо якщо стратегія робота здатна ефективно вловлювати зміни цін.

4. Комісії та спреди. Вартість комісій за торгівлю та різниця між ціною купівлі та продажу (спред) можуть сильно впливати на прибутковість. Якщо комісії та спреди високі, то боту потрібно більше прибуткових угод, щоб компенсувати ці витрати.

5. Термін використання. Тривалість використання торгового робота також важлива при розрахунку прибутковості. Більш тривалий період

використання може збільшити загальний прибуток, особливо якщо стратегія робота дозволяє знімати прибуток з довгострокових тенденцій на ринку.

Прогнозування прибутковості торговельного робота завжди пов'язане з певним ступенем невизначеності та ризику.

Розглянемо прибутковість торгових робіт з деякими факторами, такими як точність стратегії та обсяг інвестицій. Припустимо, що є два варіанти:

Варіант 1. Торговий бот з обмеженими запитами:

- Точність стратегії: 60%
- Розмір інвестицій: \$10,000
- Тривалість використання: 6 місяців
- Щомісячна доходність: 3%
- Комісії за торгівлю: 0.4% від обсягу правочину
- Обмеження на кількість запитів: 100 запитів на день

Варіант 2. Торговий бот без обмежень:

- Точність стратегії: 80%
- Розмір інвестицій: \$10,000
- Тривалість використання: 6 місяців
- Щомісячна доходність: 4%
- Комісії за торгівлю: 0.4% від обсягу правочину
- Без обмежень на кількість запитів

Сумарний прибуток для кожного варіанта:

Варіант 1:

Щомісячна доходність = Щомісячна доходність(%) * Розмір інвестицій

Прибуток = (Щомісячна доходність - Комісії) * Кількість місяців

= (0.03 * Розмір інвестицій - (0.004 * Розмір інвестицій)) * 6 * \$ 10,000

= (0.03 * \$10,000 - (0.004 * \$10,000)) * 6 * \$10,000

= (\$300 - \$40) * 6

= \$ 260 * 6

= \$ 1,560

Варіант 2:

$$\begin{aligned}
&\text{Щомісячна доходність} = \text{Щомісячна доходність}(\%) * \text{Розмір інвестицій} \\
&\text{Прибуток} = (\text{Щомісячна доходність} - \text{Комісії}) * \text{Кількість місяців} \\
&= (0.04 * \text{Розмір інвестицій} - (0.004 * \text{Розмір інвестицій})) * 6 * \$ 10,000 \\
&= (0.04 * \$10,000 - (0.004 * \$10,000)) * 6 * \$10,000 \\
&= (\$400 - \$40) * 6 \\
&= \$ 360 * 6 \\
&= \$ 2,160
\end{aligned}$$

В реальних умовах такі фактори як волатильність ринку, умови комісій та інші можуть значно впливати на прибутковість торгових ботів. Подані розрахунки є лише прикладами і може бути не відповідають реальній прибутковості торгових стратегій.

При порівнянні двох варіантів стає ясно, що торговий бот з обмеженнями на кількість запитів на Binance (Варіант 1) показує менший сумарний прибуток (\$1,560), ніж бот без обмежень на кількість запитів (Варіант 2) із прибутком у \$2,160. Цей приклад демонструє, наскільки важливо враховувати обмеження кількості запитів під час виборів торгових ботів та його стратегій. Обмеження можуть вплинути на загальну прибутковість торгової стратегії, якщо її щомісячна доходність змінюється завдяки точності стратегій.

Висновок до розділу 4

На підставі інформації, представленої вище, можна зробити наступні висновки про торгових ботів з обмеженням за кількістю веб-запитів на Binance мають такі переваги:

1. Економічна ефективність. Обмеження за кількістю веб-запитів дозволяє знизити витрати на використання API Binance. Відправляючи велику кількість запитів, обмеження допомагає знизити використання ресурсів і, відповідно, економить гроші.
2. Управління ризиками. Обмеження за кількістю запитів може допомогти запобігти надмірній активності торгового бота та встановлювати

обмеження за розміром позицій або частотою торгових операцій для зменшення потенційних втрат.

3. Дотримання правил біржі. Обмеження за кількістю запитів допомагає дотримуватись правил та обмежень, встановлених Binance. Це може запобігти негативним наслідкам, таким як блокування облікового запису.

Однак торгові боти з обмеженням кількості запитів мають деякі недоліки:

1. Обмеження можливостей. Обмеження кількості запитів обмежує активність торгового робота і може обмежити його здатність оптимально реагувати зміни у ринкових умовах.

2. Пропущені можливості. Обмеження може призвести до пропуску можливостей до виконання прибуткових торгових операцій, особливо у умовах стрімкого мінливого ринку.

З іншого боку, торгові боти без обмежень за кількістю запитів мають переваги:

1. Велика свобода дій. Без обмеження кількості запитів торговий бот має велику свободу дій, оскільки може взаємодіяти з API Binance без будь-яких обмежень.

2. Швидка реакція на ринкові зміни. Без обмеження кількості запитів бот може миттєво реагувати зміни ринкових умов і виконувати операції у режимі реального часу.

Однак слід пам'ятати, що торгові боти без обмежень також можуть бути схильні до високого ризику через більш інтенсивну активність і можливість скоєння більшої кількості помилок.

Зрештою, вибір між торговими ботами з обмеженням та без обмежень за кількістю запитів на Binance залежить від цілей та переваг трейдера, а також його рівня ризикової толерантності та фінансових можливостей. Необхідно ретельно оцінити всі переваги та недоліки кожного виду торгового бота та прийняти виважене рішення.

РОЗДІЛ 5. ОХОРОНА ПРАЦІ

5.1 Вступ

Сучасні технологічні розробки та новітні технології дозволяють безперервно автоматизувати та оптимізувати виробничі процеси. Сьогодні неможливо уявити підприємство без використання комп'ютерних технологій. Українське законодавство чітко регламентує правила та вимоги до використання комп'ютерної техніки на підприємствах, включаючи охорону праці при роботі з комп'ютерами, що обумовлено широкою сферою діяльності працівників, які використовують комп'ютерну техніку.

На робочих місцях бухгалтерів і менеджерів повинні бути вжиті заходи щодо запобігання можливого впливу небезпечних і шкідливих факторів. Рівень цих факторів не повинен перевищувати граничних значень, визначених нормативно-правовими, технічними та санітарно-гігієнічними нормами. Ці нормативні документи зобов'язують працівників створювати на робочому місці умови праці, що виключають вплив небезпечних і шкідливих факторів або приводять їх у допустимі межі.

5.2 Аналіз небезпечних і шкідливих факторів

Небезпеки при використанні персональних комп'ютерів Менеджери та бухгалтери можуть наражатися на такі небезпеки при використанні персональних комп'ютерів:

- Ураження електричним струмом;
- Енергетичні небезпеки (через коротке замикання: опіки, електричні дуги, витік розплавленого металу);
- Пожежна небезпека;
- Температурні небезпеки (вплив високих температур через нагрівання елементів конструкції);
- Механічні небезпеки (наприклад, травми внаслідок падінь, ударів рухомих частин, порізів гострими краями елементів конструкції);

- Радіаційні небезпеки (вплив видимого випромінювання від звукового (акустичного), інфрачервоного, радіочастотного, ультрафіолетового, іонізуючого випромінювання та когерентного інтенсивного лазерного світла).
- Хімічні небезпеки (контакт з певними хімічними речовинами, що використовуються при обслуговуванні обладнання).

Робота за комп'ютером характеризується високим розумовим напруженням, високим неврологічним та емоційним навантаженням, високим напруженням зору та значним навантаженням на м'язи рук при роботі з клавіатурою. Правильна конструкція і розміщення елементів робочого місця має вирішальне значення для підтримки оптимальної робочої пози.

При роботі з комп'ютером необхідно дотримуватися режиму праці та відпочинку. В іншому випадку працівники можуть перенапружувати зорову систему і скаржитися на незадоволеність роботою, головні болі, дратівливість, порушення сну, втому, біль в очах, спині, шії і руках.

У спортивних клубах має бути чітко визначена перерва на 10 або 15 хвилин кожні одну-дві години (без урахування обідньої перерви). У будь-якому випадку, підприємець повинен організувати такий графік роботи, щоб кількість часу, проведеного в контакті з комп'ютером у спортивному клубі, не перевищувала чотирьох годин. Крім того, з метою підтримання належного рівня здоров'я та професійної придатності працівників, рекомендується створити на підприємстві окремі побутові кімнати для відпочинку працівників та зняття нервово-емоційного напруження.

5.2.1 Освітленість робочого місця

Правильно спроектоване і впроваджене промислове освітлення покращує візуальні умови праці, знижує втому, підвищує продуктивність, позитивно впливає на робоче середовище, має позитивний психологічний вплив на персонал, підвищує безпеку і знижує травматизм.

Недостатнє освітлення викликає перенапруження очей, знижує концентрацію уваги і призводить до передчасної втоми. Занадто багато світла

може викликати сліпоту, подразнення і слезотечу. Неправильний напрямок світла на робочому місці може спричинити різкі тіні, відблиски. Це змушує працівників втрачати відчуття напруги.

Всі ці причини можуть призвести до нещасних випадків і професійних захворювань, тому дуже важливо правильно враховувати світло.

Існує три типи освітлення:

- Природне
- Штучне
- Комбіноване (поєднання природного та штучного освітлення).

Природне освітлення – це освітлення будівель денним світлом, що проникає через світлові прорізи в огорожувальних конструкціях. Природне освітлення характеризується значними коливаннями в залежності від часу доби, пори року, характеру місцевості та різних інших факторів.

Використання штучного освітлення в темних приміщеннях або під час денних робіт не забезпечує нормованих значень коефіцієнта природної освітленості (похмура погода, короткий світловий день). Освітлення, при якому недостатнє природне освітлення доповнюється штучним, називається комбінованим. Згідно з нормами, в приміщеннях повинні застосовуватися системи комбінованого освітлення.

Штучне освітлення можна розділити на наступні категорії:

- Робоче
- Аварійне
- Евакуаційне
- Охоронне освітлення

Робоче освітлення включає загальне та комбіноване освітлення. Загальне освітлення – це освітлення, при якому світильники розташовуються рівномірно по всій площі приміщення або по відношенню до розташування обладнання. Комбіноване освітлення – це освітлення, при якому до загального освітлення додається місцеве освітлення.

Вимоги до освітлення в приміщеннях з комп'ютерами такі: для високоточних зорових робіт загальне освітлення має становити 300 люкс, а комбіноване – 750 люкс; аналогічні вимоги для робіт середньої точності – 200 люкс і 300 люкс відповідно.

5.2.2 Вплив випромінювання

Максимальні рівні рентгенівського випромінювання на робочих місцях за комп'ютером зазвичай не перевищують 10 мЗв/год, а інтенсивність УФ та ІЧ випромінювання від екранів моніторів знаходиться в межах 10-100 мВт/м².

Для зменшення впливу цих випромінювань рекомендується використовувати монітори зі зниженим рівнем випромінювання (MPR-II, TCO-92, TCO-99), встановлювати захисні екрани та дотримуватися встановлених режимів праці та відпочинку.

5.2.3 Вплив електричного струму

Потенційна можливість виникнення небезпечних і шкідливих факторів, що впливають на людину та навколишнє середовище. Оскільки проект безпосередньо пов'язаний з використанням технічного обладнання, основною небезпекою, на яку наражаються працівники, є електричний струм.

Зовнішніх ознак ризику ураження електричним струмом немає, оскільки неможливо заздалегідь виявити можливість ураження електричним струмом. Тяжкість ураження електричним струмом: Інвалідність, спричинена ураженням електричним струмом, часто є довготривалою і навіть смертельною.

Струм частотою 50 Гц викликає сильні м'язові спазми, і людина стає "прикутою" до струмоведучих частин, не в змозі самотійно покинути їх.

Внутрішні струми впливають на людину зсередини, призводячи до зовнішніх пошкоджень, ураження електричним струмом і смерті. Внутрішні ефекти включають термічні, електролітичні та біологічні ефекти. Термічна дія – це опіки, нагрівання, пошкодження кровоносних судин і перегрів серця, мозку та інших внутрішніх органів, що призводить до порушення їхньої роботи.

Характер ураження залежить від величини, виду та шляху проходження струму, тривалості впливу, індивідуальних особливостей людини та фізіологічного стану в момент порушення.

5.2.4 Виробничий шум

При роботі з комп'ютером менеджери та бухгалтери також піддаються впливу ряду небезпечних і шкідливих факторів, серед яких електромагнітні поля, інфрачервоне випромінювання, іонізуюче випромінювання, шум, вібрація і статична електрика.

Шум негативно впливає на організм людини і погіршує умови праці. Люди, які працюють в умовах тривалого впливу шуму, відчують дратівливість, головний біль, запаморочення, погіршення пам'яті, підвищену втомлюваність, втрату апетиту і біль у вухах. Шум впливає на концентрацію уваги, порушує фізіологічні функції, спричиняє підвищені витрати енергії та втому через нервову та розумову перенапругу, а також погіршує вербальну комунікацію. Все це знижує працездатність людини, погіршує продуктивність, якість і безпеку праці. Тривалий вплив гучного шуму (>80 дБ) може призвести до часткової або повної втрати слуху.

Рівень шуму на робочому місці не повинен перевищувати 50 дБ. Для зменшення шуму стіни і стеля приміщення, де знаходиться комп'ютер, можуть бути покриті звукопоглинальним матеріалом.

5.2.5 Умови мікроклімату

Санітарні норми визначають значення параметрів мікроклімату, які створюють комфортне середовище. Ці норми встановлюються залежно від пори року, характеру трудового процесу та особливостей виробничого приміщення.

Норми параметрів мікроклімату для приміщень, де встановлені комп'ютери представлено у таблиці 4.1.

Період року	Параметр мікроклімату	Величина
-------------	-----------------------	----------

Холодний	Температура повітря в приміщенні	22-24C
	Відносна вологість	40-60%
	Швидкість руху повітря	до 0,1м/с
Теплий	Температура повітря в приміщенні	23-25C
	Відносна вологість	40-60%
	Швидкість руху повітря	0,1-0,2м/с

Таблиця 4.1. Норми параметрів мікроклімату для приміщень

Норми подачі свіжого повітря в приміщення, де розташовані комп'ютери представлено у таблиці 4.2.

Характеристика приміщення	Об'ємна витрата подаваного в приміщення свіжого повітря, м ³ /на одну людину в годину
Обсяг до 20м ³ на людину	Не менш 30
20-40м ³ на людину	Не менш 20
Більше 40м ³ на людину	Природна вентиляція

Таблиця 4.2. Норми подачі свіжого повітря в приміщення

Для забезпечення комфортних умов використовуються організаційні методи (раціональне розміщення робіт за сезоном і датою, чергування роботи і відпочинку) та технічні засоби (системи вентиляції, кондиціонування, опалення).

5.2.6 Організація робочого місця

Ергономічні вимоги до робочих місць менеджерів і бухгалтерів. Взаємне розташування робочого місця та всіх його елементів повинно відповідати антропометричним, фізичним і психологічним вимогам. Важливе значення має також характер роботи. Зокрема, при конфігурації робочого місця повинні бути дотримані такі основні вимоги: оптимальне розташування обладнання, що

міститься на робочому місці, і достатній робочий простір для всіх необхідних рухів.

Основними елементами робочого місця є столи та стільці. Основна робоча поза – сидячи.

Сидяча робота мінімізує втому. Раціональне планування робочого місця для менеджерів і бухгалтерів передбачає чіткий порядок і послідовність у розташуванні предметів, інструментів і документації. Все необхідне для частого виконання роботи, наприклад, пристрої введення інформації (клавіатури), пристрої виведення інформації (монітори) і друкарські пристрої, знаходяться в межах легкої досяжності від робочого простору.

5.3 Розрахунок освітленості робочого місця

Освітленість робочого місця програміста є важливим фактором для комфортної та ефективної роботи. Розрахунок освітленості може бути корисним для визначення, чи достатньо світла на робочій поверхні. Основний параметр, який використовується при розрахунку освітленості, – це одиниця виміру під назвою люкс (lx), яка вказує на кількість світла, що падає на одиницю площі.

Для розрахунку освітленості робочого місця програміста можна використовувати такі кроки:

1. Виміряйте площу робочої поверхні у квадратних метрах. Наприклад, якщо ширина столу дорівнює 1 м, а глибина – 0,6 м, площа робочої поверхні становитиме $1 \text{ м} \times 0,6 \text{ м} = 0,6 \text{ кв. м}$.

2. Визначте рівень освітленості, рекомендований для роботи на комп'ютері програміста. Зазвичай це значення становить близько 500-750 люксів. Однак може знадобитися яскравіше світло, якщо на робочій поверхні присутні завдання, що вимагають високої візуальної точності.

3. Уточніть коефіцієнт використання приміщення. Цей коефіцієнт відбиває здатність поверхні у приміщенні відбивати світло. Використовуйте типові значення, такі як 0,7 для приміщень із білими стінами та стелею.

4. Розрахуйте потрібну загальну світлову потужність. Для цього помножте площу робочої поверхні на значення рівня освітленості та розділіть на коефіцієнт використання приміщення. Наприклад, на площі 0,6 кв. м, рівня освітленості 750 люкс і коефіцієнта використовуваності 0,7, загальна потужність дорівнюватиме $(0,6 \text{ кв. м} \times 750 \text{ люкс}) / 0,7 = 642 \text{ Вт}$.

5. Розмістіть освітлювальні прилади таким чином, щоб забезпечити рівномірний розподіл світла по всій робочій поверхні. Використовуйте відповідні джерела світла, такі як настільні лампи або світлодіодні панелі для досягнення необхідного рівня освітлення.

Враховуючи ці вказівки, ви зможете розрахувати необхідну освітленість на робочому місці та забезпечити комфортні умови для роботи.

5.4 Розробка заходів щодо зменшення впливу небезпечних і шкідливих факторів

Розробка заходів щодо зменшення впливу небезпечних та шкідливих факторів на робочому місці програміста є важливим аспектом забезпечення безпеки та здоров'я працівників. Ось деякі заходи, які можна вжити:

1. Ергономіка робочого місця. Забезпечте зручне та правильне розташування робочого столу, стільця та комп'ютерного обладнання. Регульовані меблі, підставки для монітора та клавіатури, а також ергономічне крісло можуть допомогти знизити навантаження на спину, шию та руки.

2. Освітлення. Забезпечте робоче місце достатнім та приємним освітленням. Використовуйте світильники з регулюванням яскравості, щоб запобігти надмірному навантаженню на очі. Розмістіть джерела світла те щоб уникнути відблисків на моніторі.

3. Вентиляція. Забезпечте хорошу вентиляцію в приміщенні, щоб запобігти скупченню запахів та випарів. Регулярно провітрюйте приміщення та використовуйте системи кондиціонування повітря, якщо це необхідно.

4. Шум. Помістіть робоче місце в окрему зону приміщення або використовуйте звукоізолюючі матеріали для зниження рівня шуму, якщо навколо є джерела сторонніх звуків.

5. Перерви та фізична активність. рекомендуйте програмістам робити періодичні перерви від роботи за комп'ютером. Проведення вправ для очей та розтяжок може допомогти знизити напругу та втому.

6. Використання ергономічних клавіатур та мишей. Надайте співробітникам комфортні та ергономічні клавіатури та миші, які скорочують навантаження на руки та запобігають розвитку синдрому карпального каналу та інших рукоплечових захворювань.

7. Навчання та інформування. Проводьте навчання співробітників щодо правильних робочих методів, ергономіки та способів зниження негативного впливу небезпечних та шкідливих факторів. Інформуйте їх про можливі проблеми та способи їх запобігання.

8. Допомога фахівців. При необхідності звертайтеся до професійних фахівців, таких як ергономи, лікарі-травматологи та інші, які можуть надати керівництво та рекомендації щодо покращення умов робочого місця.

Облік та впровадження цих заходів може допомогти знизити вплив небезпечних та шкідливих факторів на робочому місці програміста та створити більш комфортне та безпечне середовище для роботи.

5.4.1. Заходи щодо зниження величини виникаючих зарядів статичної електрики

Статична електрика може створити неприємні ситуації робочому місці програміста, тому важливо вжити заходів для зниження величини виникаючих зарядів. Ось кілька рекомендацій:

1. Заземлення. Переконайтеся, що всі електричні пристрої, включаючи комп'ютери, монітори та периферійні пристрої, мають належну систему заземлення. Заземлення допомагає відводити статичний заряд у землю та знижує ризик його накопичення.

2. Антистатичні мати. Розмістити антистатичні мати на робочій поверхні, особливо навколо комп'ютера та клавіатури. Ці мати запобігають накопиченню статичного заряду та забезпечують безпечне заземлення.

3. Антистатичні наручні ремені. Використовуйте антистатичні наручні ремені та з'єднуйте їх із антистатичними матами або системами заземлення. Наручні ремені допомагають відводити статичний заряд з вашого тіла та запобігати його передачі на компоненти обладнання.

4. Антистатичні пакувальні матеріали. Під час роботи з електронними компонентами або іншими вразливими пристроями використовуйте антистатичні пакувальні матеріали, такі як мішечки, плівки або блістери. Це допоможе запобігти ненавмисним розрядам статичної електрики при поводженні з чутливими компонентами.

5. Вологість у приміщенні. Підтримуйте оптимальний рівень вологості у приміщенні. Сухе повітря сприяє нагромадженню статичного заряду. Використовуйте зволожувачі або інші способи підтримки комфортної вологості навколишнього середовища.

6. Уникайте синтетичних матеріалів. Одяг із синтетичних матеріалів може сприяти накопиченню статичного заряду. Перевагу слід віддавати натуральним матеріалам, таким як бавовна, шерсть або шовк, які менше накопичують статичну енергію.

7. Ізолювання робочої області. Крім перерахованих вище заходів, можна також розділити робочу область програміста від інших приміщень, особливо технічних чи виробничих областей, де може виникати більша кількість статичного заряду.

За дотримання цих заходів можна знизити ризик неприємних наслідків від статичної електрики робочому місці програміста.

5.4.2 Заходи щодо зниження шуму на робочому місці

Шум на робочому місці програміста може бути відволікаючим і негативно впливає на концентрацію та продуктивність. Ось кілька заходів, які можуть допомогти знизити рівень шуму:

1. Звукоізоляція приміщення. Зробіть звукоізоляцію стін, стелі та підлоги навколо свого робочого місця. Використовуйте спеціальні матеріали або панелі, що поглинають звук та зменшують його відображення.
2. Використання навушників із шумопоглинанням. Надягніть навушники з функцією активного шумоподавлення. Ця технологія дозволяє заблокувати зовнішні звуки, створюючи тишу навколо вас.
3. Ізоляційні штори. Підвісите важкі фіранки або спеціальні звукоізолюючі штори на вікна. Це допоможе зменшити проникнення шуму ззовні до приміщення.
4. Уникайте місця з високим рівнем шуму. Якщо у вас є можливість вибрати місце для розміщення робочого місця, намагайтеся уникати місця з високим рівнем шуму, наприклад, поряд з проходами, кухнями або приміщеннями з обладнанням, що створює шум.
5. Розташування обладнання. Розмістіть шумні пристрої, такі як сервери або принтери, віддалені від робочого місця програміста. Якщо це можливо, використовуйте безшумне обладнання для мінімізації шумових джерел.
6. Використання шумоподавлюючих матеріалів. Додати на робочу поверхню або навколо неї матеріали, здатні поглинати звук. Це можуть бути пінополіуретанові плити або спеціальні губки, які допоможуть знизити відображення звуку та знизити загальний рівень шуму.
7. Режим роботи. Якщо у офісі є пікові періоди шуму, спробуйте організувати свій робочий час так, щоб мінімізувати вплив шуму на роботу. Розгляньте можливість зосередитись на завданнях, що потребують максимальної концентрації, у більш спокійний час.[13]

Висновок до розділу 5

Випромінювання на робочому місці програмістів може впливати на їх здоров'я та благополуччя. Є кілька джерел випромінювання, які можуть бути пов'язані з роботою програмістів, такі як:

1. Екранні пристрої. Часте впровадження комп'ютерних моніторів може становити певні небезпеки, такі як випромінювання електромагнітних полів та світлове випромінювання. Тривала робота перед монітором може призвести до втоми очей, напруження зору та сухості очей. Для зниження цих ризиків можна використовувати екрани з фільтрами відображення та встановити правильну яскравість та контрастність.

2. Бездротові мережі. Wi-Fi-обладнання та інші бездротові технології можуть створювати певні рівні радіочастотного випромінювання. Хоча існує дебата про потенційні ризики такого випромінювання, дослідження поки що не дають кінцевих висновків. Однак для зменшення можливого впливу радіочастотного випромінювання можна спробувати використовувати дротове підключення, а не бездротове.

3. Електростатичне поле. При роботі з комп'ютерами та електронікою може виникати електростатичне поле. У сухих умовах воно може призвести до неприємних статичних розрядів, особливо якщо програміст працює із чутливими електронними компонентами. Використання антистатичних пристроїв, таких як антистатичні килимки або наручні стрічки, може допомогти знизити ризик.

4. Висвітлення. Неправильне освітлення на робочому місці може спричинити втому очей, напруження та нездужання. Переважно мати достатнє природне освітлення або регулювати штучне освітлення таким чином, щоб мінімізувати відображення та відблиски на моніторі.

5. Шум. Робоче середовище програміста може бути піддане шуму від комп'ютерних вентиляторів, серверів або інших електронних пристроїв. Шум може спричинити подразнення, стрес та погіршення концентрації. Використання навушників із шумопоглинанням або створення тихої зони для роботи може допомогти знизити вплив шуму.

Загалом дотримання деяких рекомендацій та створення комфортного та безпечного робочого середовища може допомогти мінімізувати негативний вплив випромінювання на робочому місці програмістів. Це може включати правильне налаштування обладнання, періодичні перерви та вправи для очей, а також вжиття заходів щодо зниження впливу інших факторів, таких як шум та освітлення.

РОЗДІЛ 6. ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА

6.1 Вступ

Охорона довкілля є актуальною проблемою у світі. Розвиток технологій та інформаційних технологій неминуче впливає на довкілля, включаючи діяльність програмістів. У даному рефераті розглянемо загальні відомості про забруднення навколишнього середовища професією програміста та заходи щодо його запобігання.

Охорона довкілля одна із найбільш актуальних проблем сучасного суспільства. Забруднення довкілля негативно позначається здоров'я людей, тварин і рослин, і навіть на екосистемах загалом. Програмісти, які займаються розробкою програмного забезпечення та створенням інформаційних систем, також роблять свій внесок у забруднення навколишнього середовища. У даному рефераті ми розглянемо основні аспекти цієї проблеми та заходи щодо її вирішення.

6.2 Загальні відомості про забруднення

Професія програміста здавалося б екологічно безпечною, оскільки основна діяльність пов'язана з роботою на комп'ютері. Однак, існують певні аспекти, які можуть негативно впливати на довкілля.

Один із таких аспектів – використання енерговитратних обчислювальних систем. Чим складніша і потужніша система, тим більше енергії вона споживає. При масовому використанні таких систем цей фактор може впливати на загальне споживання енергії і, відповідно, на виділення парникових газів.

Ще одним аспектом є використання електронних пристроїв та їх подальша утилізація. Більшість програмістів працюють на ноутбуках, смартфонах та інших електронних пристроях, які мають обмежений термін служби. При викиді таких пристроїв на сміттєзвалище утилізації може створюватися велика кількість електронного сміття, що містить небезпечні речовини.[6]

6.3 Заходи щодо запобігання забруднення навколишнього середовища

Для зниження забруднення навколишнього середовища професією програміста можна застосовувати такі заходи:

- **Енергоефективність.** використовувати більш енергоефективне обладнання або оптимізувати роботу програм для зниження споживання ресурсів.
- **Утилізація електронних пристроїв:** Організувати процес збирання та утилізації відпрацьованих електронних пристроїв з метою уникнення викиду небезпечних речовин на звалище.
- **Зменшення використання паперу.** Програмісти можуть максимально використовувати електронні документи та скоротити використання паперових носіїв інформації.
- **Розвиток віддаленої роботи.** Популяризувати практику віддаленої роботи, яка допомагає скоротити використання автотранспорту та, як наслідок, викиди шкідливих речовин.
- **Екологічне усвідомлення.** Програмісти можуть усвідомлено підходити до своєї діяльності, враховувати навколишнє середовище при розробці програмного забезпечення та оцінювати їх еко відбиток.

Охорона довкілля є колективним питанням, яке потребує спільних зусиль усіх учасників суспільства. Програмісти, своєю чергою, можуть істотно проводити забруднення довкілля і використовувати свої здібності та знання для реалізації екологічно стійких практик.

Висновок до розділу 6

Професія програміста, незважаючи на свою на перший погляд безпеку в екологічному відношенні, також робить свій внесок у забруднення навколишнього середовища. Це відбувається, зокрема, через використання енерговитратних обчислювальних систем та проблеми утилізації електронних пристроїв, які є невід'ємною частиною повсякденної роботи програмістів.

Тим не менш, існують заходи, які можна застосувати для зниження шкоди навколишньому середовищу. Це включає використання більш енергоефективних обчислювальних систем, а також продуману утилізацію електронного обладнання. Слід заохочувати застосування енергоефективних технологій у розробці програмного забезпечення та стимулювати переробку та вторинне використання електронних пристроїв.

Охорона довкілля – це завдання, що лежить на плечах усіх членів суспільства. Програмісти як частина цього суспільства також несуть свою відповідальність і можуть зробити свій внесок у збереження природи та екологічну стійкість.

ВИСНОВКИ

Новий сервіс smart-кешування веб-запитів для конкурентного доступу у веб-додатку торгового бота Good Crypto до веб-ресурсу з обмеженнями API Binance прискорює швидкість обробки веб-запитів та збільшує точність стратегій торгового боту. Розроблений сервіс не багато затратнім, але вирішує дуже важливе питання щодо швидкості та точності роботи торгового боту.

Проблема роботи з веб-ресурсами з обмеженнями є досить поширеною серед веб-додатків, які залежать від ресурсів сторонніх API. Smart-кешування веб-запитів є одним з найкращих способів вирішення цієї проблеми. Даний сервіс не підійде веб-додаткам, які надсилають унікальні веб-запити та очікують індивідуальну відповідь для кожного запиту. Але сервіс є надійним рішенням для веб-додатків веб-запити яких можна кешувати та відповіді можуть бути ідентичними.

Завдяки сервісу smart-кешування веб-запитів торгові боти почнуть працювати з більше коректними стратегіями та враховувати більш точні данні. Це підвищить попит на ринку та зробить веб-додаток торгового бота Good Crypto ще популярніше. В пошуках надійного помічника в роботі з крипто валютою люди будуть звертати увагу в першу чергу на точність стратегій торгового боту. Отже, додавши новий сервіс smart-кешування веб-запитів до веб-додатку торгового бота можна виділити наступні:

- Збільшення швидкості одержання даних. Кешування запитів дозволить значно зменшити час відповіді на запити до API Binance. Замість кожного разу надсилати запити на отримання даних з біржі, новий сервіс зберігатиме дані з попередніх запитів та надаватиме їх безпосередньо з кешу. Це скорочує час очікування та підвищує швидкодію веб-програми.
- Зниження навантаження на API. Веб-запити до API Binance можуть бути досить ресурсномісткими операціями, особливо якщо програма робить безліч однотипних запитів або оновлює дані з високою частотою. Кеш дозволить зменшити кількість запитів до API.

- Поліпшення чуйності програми. Швидке отримання даних дозволить торговому боту швидше приймати рішення і реагувати на зміни над ринком. Це особливо важливо для таких стратегій, які потребують швидкої реакції на зміни цін та обсягів торгів.

- Поліпшення відмовостійкості. Якщо API Binance тимчасово недоступна або має високу затримку, новий сервіс зможе продовжити надання даних із кешу, мінімізуючи простої роботи програми.

- Мінімізація витрат на використання API. Деякі біржі, включаючи Binance, можуть мати обмеження кількості запитів або стягувати плату за часте використання API. Використовуючи кешування запитів, ми скорочуємо кількість запитів та, відповідно, оптимізуємо використання API.

Результатом нового сервісу smart-кешування є проміжний сервер, через який інші сервіси веб-додатку звертаються до API Binance. Сервер знаходиться у локальній сеті компанії Good Crypto и та підтримується командою розробників.

Готова кваліфікована робота нового сервісу smart-кешування для конкурентного доступу до веб-ресурсів з обмеженнями може впровадяться у веб-додаток торгового боту компанії Good Crypto та стати вірним рішенням, щодо вирішення проблеми роботи веб-ресурсом з обмеженнями API Binance.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. В.Ф. Ажищев, К.В. Кошкін, А.С. Морозова «Методичні вказівки до виконання дипломного проектування» – Міністерство освіти та науки України, НУК ім. адмірала Макарова, 2014
2. В.І. Грекул, Г.Н. Денищенко, Н.Л. Коровкіна «Проектування інформаційних систем» – Навчальний посібник, 2005
3. Сергій Костянтинів «API»
4. Мехді Меджуї, Ерік Вайлд «Безперервний розвиток API. Правильні рішення у мінливому технологічному ландшафті, 2-ге вид.»
5. "How to Create a Crypto Trading Bot for Binance (Using Python)" – Матеріали сторінки: <https://python.plainenglish.io/automate-your-crypto-trading-with-python-a-step-by-step-guide-to-building-a-binance-trading-bot-a658f565f0b2>
6. В.І. Голінько, М.Ю. Іконніков, Я.Я. Лебедев «Охорона праці в галузі інформаційних систем» – Дніпропетровськ НГУ, 2015
7. Матеріали сторінки: <https://docs.github.com/rest?apiVersion=2022-11-28#rate-limiting>
8. Матеріали сторінки: <https://developer.linkedin.com/docs/rest-api?u=0#>
9. Матеріали сторінки: <https://dev.bitly.com>
10. Матеріали сторінки: <https://uk.wikipedia.org/wiki/REST>
11. Матеріали сторінки: <https://www.binance.com/en/binance-api>
12. Матеріали сторінки: <https://goodcrypto.app>
13. Матеріали сторінки: <https://pro-op.com.ua/article/1065-nstruktsya-z-ohoroni-prats-dlya-fahvtsya-z-nformatsynih-tehnology>

ДОДАТОК А

Інструкція користувача сервісу smart-кешування для конкурентного доступу до веб-ресурсів з обмеженнями

1. Вступ

1.1. Галузь застосування

Сервіс smart-кешування для конкурентного доступу до веб-ресурсів з обмеженнями у компанії Good Crypto призначений для використання в торгових роботах, які здійснюють торгівлю на криптовалютному ринку Binance. Цей сервіс допомагає оптимізувати процес запиту інформації з веб-ресурсу Binance, покращуючи швидкість виконання запитів та скорочуючи навантаження на мережу та сервери.

1.2. Короткий опис можливостей

Сервіс smart-кешування дозволяє кешувати та автоматично оновлювати дані, що отримуються з зовнішнього веб-ресурсу.

1.3. Рівень підготовки користувача

Для ефективного використання цього сервісу необхідно мати певні навички програмування та розуміння принципів роботи торгових роботів. Користувачам рекомендується знати основи веб-розробки, включаючи роботу з API та протоколами передачі даних.

Перелік необхідних знань:

- Основи програмування та мови програмування JavaScript.
- Знання протоколів передачі (HTTP, WebSocket).
- Розуміння різних видів запитів до веб-ресурсів (GET, POST, PUT тощо).
- Навички роботи з API веб-ресурсів та розуміння форматів даних (JSON, XML).

1.4. Перелік експлуатаційної документації, з яким необхідно ознайомитись користувачеві

Розробний відділ в обов'язковому порядку має вивчити експлуатаційну документацію, щодо використання сервісу smart-кешування для конкурентного доступу до веб-ресурсів з обмеженнями.

2. Призначення та умови застосування

2.1. Види діяльності, функції, для автоматизації яких призначений сервіс smart-кешування

Сервіс smart-кешування призначений для автоматизації наступних функцій:

- Виконання запитів до веб-ресурсів для отримання актуальної інформації щодо ринкових даних.
- Кешування даних для подальшого використання в торгових ботах.
- Управління частотою оновлення даних відповідно до заданих обмежень.
- Оптимізація роботи мережевого трафіку та збільшення швидкості виконання запитів.

2.2. Умов до дотримання (виконання, настання) яких забезпечується застосування системи відповідно до призначення

Успішне застосування та ефективна експлуатація системи можливі при неухильному дотриманні та виконанні наступних умов:

1. Наявність складу конфігурації персонального комп'ютера та програмного забезпечення.

Персональний комп'ютер повинен мати наступні характеристики:

- комп'ютер класу Intel i7/Xeon і вище;
- оперативна пам'ять від 32Гб і вище;
- SSD-диск або HDD-диск;

- монітор, миша, клавіатура;
- Wi-Fi роутеру, для підключення до інтернету або кабель UTP, для підключення до роутеру;

Загальносистемне програмне забезпечення повинно мати:

- операційну систему Windows 7 та вище або macOS;
- будь-який веб-браузер;
- інтегроване середовище розробки для розробки;
- програмне забезпечення Docker.

2. Наявність у користувачів системи достатнього рівня для грамотних дій при експлуатації системи. Вимоги до рівня підготовки користувача зазначені у пункті 1.3.

3. Підготовка до роботи

3.1 Склад та зміст дистрибутивного носія даних

Сервіс smart-кешування надається як програмне забезпечення, доступного для завантаження на сервер компанії торгового бота. Дистрибутивний носій містить необхідні файли та документацію для встановлення та налаштування сервісу.

3.2. Порядок завантаження даних та програм

Порядок до підключення к системі має наступні кроки:

1. Завантажте дистрибутивний носій даних сервісу smart-кешування.
2. Встановіть програмне забезпечення на сервер компанії торгового бота, дотримуючись інструкцій, наданих у документації.
3. Налаштуйте параметри кешування веб-запитів, враховуючи вимоги та обмеження веб-ресурсу.

3.3. Порядок перевірки працездатності

Порядок перевірки працездатності:

- Уважно ознайомитися з документацією щодо сервісу smart-кешування та переконайтеся, що всі необхідні налаштування виконані правильно.
- Використовуйте симуляцію запитів або тестові дані для перевірки роботи сервісу та коректності кешування веб-запитів.
- За потреби, зв'язатися з командою яка підтримує сервіс для отримання додаткової допомоги.

4. Опис операцій

Опис операцій сервісу smart-кешування для конкурентного доступу до веб-ресурсів з обмеженнями:

- Налаштування параметрів кешування веб-запитів.
- Створення запитів до веб-ресурсів із використанням сервісу smart-кешування.
- Обробка отриманих даних та використання їх у торгових ботах.
- Моніторинг роботи сервісу та його ефективності.

5. Аварійні ситуації

Аварійні ситуації сервісу smart-кешування для конкурентного доступу до веб-ресурсів з обмеженнями:

- У разі непередбачених помилок або збоїв у роботі сервісу, зв'язатися з командою яка підтримує цей сервіс для отримання допомоги.
- Можливі аварійні ситуації можуть містити проблеми мережного підключення, помилки в налаштуваннях сервісу або помилки в коді торгового бота, що використовує даний сервіс.
- Регулярно робіть резервні копії даних та перевіряйте їх цілісність, щоб відновити працездатність системи у разі втрати даних чи збоїв.