УДК 681.518.54

Кваша О.О. студент спеціальності 126 Інформаційні системи та технології
Науковий керівник: Каштан В.Ю., к.т.н., доц., доцент кафедри інформаційних технологій та комп'ютерної інженерії
(*Національний технічний університет «Дніпровська політехніка», м. Дніпро, Україна*)

ДОСЛІДЖЕННЯ ТА ВПРОВАДЖЕННЯ ПАТЕРНІВ CACHE-ASIDE І CLAIM-CHECK ВЕБ-ЗАСТОСУНКУ "ТЕЛЕГРАМ-БОТ ІТКІ"

Анотація. У роботі досліджено застосування патернів проектування Cache-Aside та Claim-Check веб-застосунку "Телеграм-бот ІТКІ" з метою оптимізації його продуктивності та безпеки. Розглянуто впровадження технологій Redis для кешування даних, що знижує навантаження на базу даних та прискорює аутентифікацію через JWT-токени, а також використання Minio для ефективної обробки та зберігання файлів.

Ключові слова: "Телеграм-бот ІТКІ", Cache-Aside, Claim-Check, Redis, Minio, кешування даних, зберігання файлів, JWT-токени.

Вступ. У сучасних умовах стрімкого зростання обсягу даних та вимог до продуктивності веб-застосунків виникає необхідність впровадження ефективних архітектурних патернів. Актуальність даної теми обумовлена потребою в оптимізації роботи веб-застосунку "Телеграм-бот ІТКІ" для підвищення його швидкодії та безпеки при одночасному покращенню гнучкості та масштабованості додатку.

Постановка задачі. Дослідити та впровадити патерни Cache-Aside і Claim-Check у веб-застосунку "Телеграм-бот ІТКІ" для підвищення його продуктивності та безпеки.

Основний зміст роботи. У ході дослідження здійснено аналіз актуальних методів підвищення продуктивності та безпеки веб-додатків в умовах високих навантажень. Досліджено та систематизовано підходи, спрямовані на оптимізацію продуктивності, посилення захисту, а також підвищення гнучкості системи. Особливу увагу приділено практичним рішенням, які широко застосовуються в індустрії для забезпечення надійності, стійкості та можливості масштабування веб-сервісів [1].

Одним із важливих аспектів дослідження є захист аутентифікаційних токенів від можливих загроз їхнього несанкціонованого доступу. У веб-додатках токени часто використовуються для авторизації доступу до API, що створює потенційні вразливості для атак. Впровадження вдосконалених методів захищеного зберігання і передачі токенів сприяє мінімізації ризику несанкціонованого доступу [2]. Завдяки впровадженню цих методів у розробці веб-додатків вони вже стали стандартом для багатьох високонавантажених систем [3].

З метою підвищення продуктивності API було застосовано патерн Cache-Aside для кешування даних із використанням Redis, що замінило прямі звернення до основної бази даних PostgreSQL. Redis, як високошвидкісне сховище даних у пам'яті, значно зменшує час обробки запитів і розвантажує базу даних [5]. Це рішення стало поширеною практикою у багатьох високонавантажених веб-додатках, адже воно підвищує ефективність системи та знижує затримки [6].

Крім того, для ефективної обробки файлів було інтегровано Minio як об'єктне сховище відповідно до патерну Claim-Check. Хоча це не є впровадженням мікросервісної архітектури, такий крок сприяє можливості відокремити масові розсилки повідомлень в окремий сервіс і надалі впровадити мікросервісну архітектуру [7]. Це дозволяє легше керувати певними компонентами системи та залишає можливості для майбутньої декомпозиції інших компонентів.

У результаті впровадження зазначених патернів та технологій вдалося підвищити продуктивність при отриманні токену аутентифікації на 5.33% і безпеку веб-застосунку

"Телеграм-бот ІТКІ", що підтверджено порівняльним аналізом показників системи до і після змін. Це можна побачити на графіку часу відповіді без використання Cache-Aside (Рис. 1) відносно часу відповіді з використанням Cache-Aside (Рис. 2)

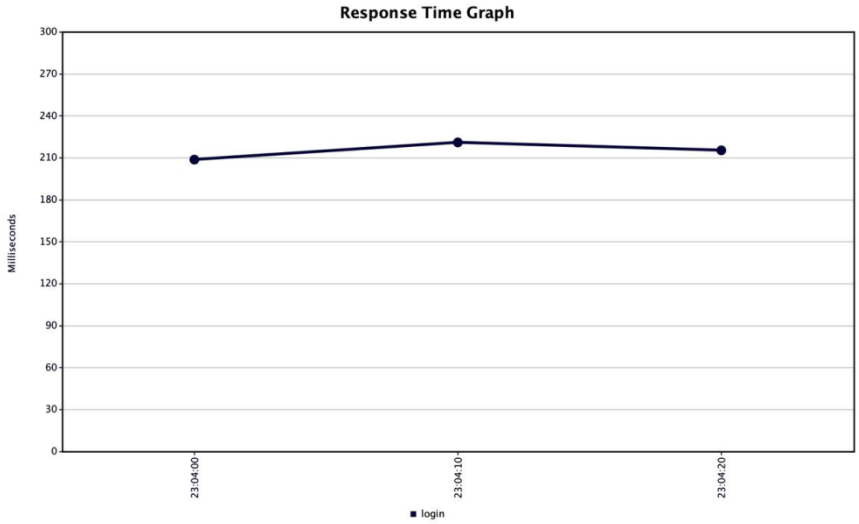

Рисунок 1 – Графік часу відповіді АПІ без кешування

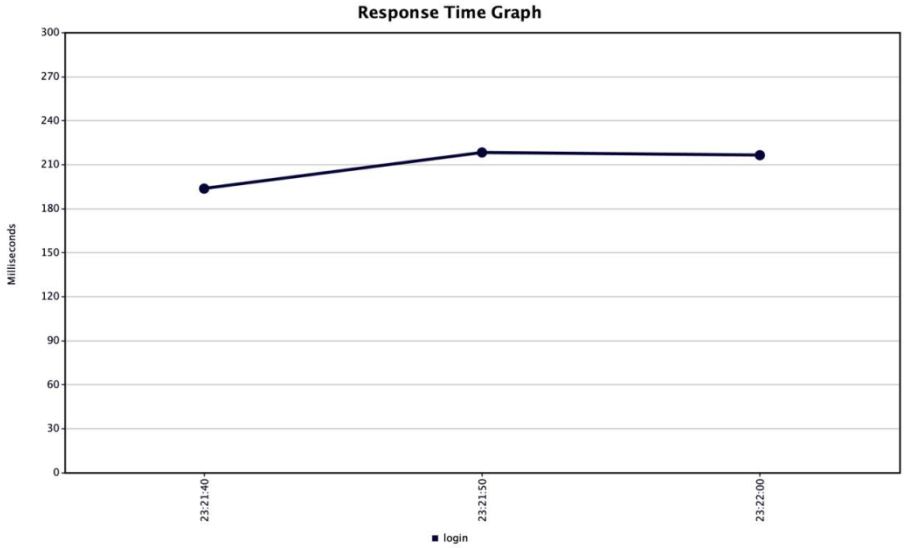

Рисунок 2 – Графік часу відповіді АПІ з Cache-Aside

Висновок. У даному дослідженні здійснено аналіз та впровадження патернів проектування Cache-Aside і Claim-Check у веб-застосунку "Телеграм-бот ІТКІ". Виокремлено основні проблеми поточної архітектури, пов'язані з продуктивністю та безпекою системи. Розроблено та реалізовано алгоритм кешування даних за допомогою Redis відповідно до патерну Cache-Aside, що дозволило знизити навантаження на базу даних та прискорити аутентифікацію через JWT-токени. Інтегровано Minio як сховище файлів та реалізовано патерн Claim-Check для ефективної передачі та зберігання даних між адміністративною платформою та клієнтами. Проведений порівняльний аналіз показав підвищення продуктивності при отриманні токену аутентифікації на 5.33% та покращення безпеки веб-застосунку після впровадження запропонованих патернів.

Список використаних джерел

1. WS Well-Architected [Електронний ресурс]. URL: https://aws.amazon.com/architecture/well-architected (9.11.2024).
2. Protecting data at rest [Електронний ресурс]. URL: https://docs.aws.amazon.com/wellarchitected/latest/security-pillar/protecting-data-at-rest.html (9.11.2024).
3. Introduction to JSON Web Tokens [Електронний ресурс]. URL: https://jwt.io/introduction (9.11.2024).
4. Caching Strategies and How to Choose the Right One [Електронний ресурс]. URL: https://codeahoy.com/2017/08/11/caching-strategies-and-how-to-choose-the-right-one/ (9.11.2024).
5. Your high-performance caching solution [Електронний ресурс]. URL: https://redis.io/solutions/caching/ (9.11.2024).
6. Simple microservices architecture [Електронний ресурс]. URL: https://docs.aws.amazon.com/whitepapers/latest/microservices-on-aws/simple-microservices-architecture-on-aws.html (9.11.2024).
7. RabbitMQ Tutorials [Електронний ресурс]. URL: https://www.rabbitmq.com/getstarted.html (9.11.2024).