

Барченко Павло Васильович, здобувач вищої освіти

Харківський національний університет радіоелектроніки, Україна

Науковий керівник: Мазурова Оксана Олексіївна, кандидат технічних наук,

доцент, доцент кафедри програмної інженерії,

Харківський національний університет радіоелектроніки, Україна

ДОСЛІДЖЕННЯ МЕТОДІВ ОПТИМІЗАЦІЇ ІНТЕГРАЦІЇ СЕРВІСІВ

Анотація. Дана робота присвячена оптимізації інтеграції сервісів фреймворку для e-commerce рішень. Розроблена система з 4х сервісів для експериментального тестування показників викликів API. Розглядалося виконання запитів для вибору даних користувачів у декількох варіаціях з використанням механізму кешування та оптимізації механізму багатопоточності у роботі одного з сервісів.

КЛЮЧОВІ СЛОВА: сервіс, виклик, інтеграція.

Abstract. The given work is devoted to optimizing the integration of framework services for e-commerce solutions. A system of 4 services has been developed for experimental testing of API call indicators. The implemented system has the functions of basic calling of a regular request for selecting user data in several variations using the caching mechanism and optimization of the multithreading mechanism in the work of one with the service.

KEY WORDS: service, call, integration.

Вступ. API (інтерфейси прикладного програмування) стали стрижнем безперебійного зв'язку між різними програмними додатками. Як канали, що полегшують обмін даними, API незамінні для створення надійних цифрових екосистем. В умовах постійно зростаючої залежності від API вкрай важливо оптимізувати їх роботу і уникнути повільного відгуку, що може суттєво утруднити роботу користувачів.

Хороша продуктивність API гарантує, що програми які їх використовують, зможуть забезпечити безперебійну роботу користувачів. Низька продуктивність може призвести до повільного часу відгуку, зниження функціональності або навіть збоїв у роботі програми, що призведе до невдоволення користувачів та втрати клієнтів.. Вирішення проблем затримки та підвищення пропускної спроможності вимагає реалізації ефективних алгоритмів та структур даних, а також оптимізації мережових протоколів та інфраструктури. Це

вимагає від розробників глибокого розуміння базових систем та властивих їм обмежень [1].

Надійність і доступність API є важливими вимогами, оскільки API повинні бути стійкими до збоїв і легко справлятися з різними сценаріями помилок. Це потребує впровадження механізмів відмовостійкості та самовідновлення, а також комплексних систем моніторингу та оповіщення.

Нарешті, задоволення потреб користувачів і забезпечення безперебійної масштабованості вимагає впровадження еластичних і модульних архітектур. Отже, розробники повинні вміти використовувати передові технології та методології, такі як контейнеризація та мікросервіси, для підтримки бездоганної продуктивності API в умовах технологій, що постійно розвивається.

Підвищення продуктивності API вимагає багатогранного підходу, який охоплює різні аспекти життєвого циклу розробки

програмного забезпечення, від проектування та реалізації до моніторингу та оптимізації [1].

Основна частина. Проблема продуктивності API полягає у досягненні тонкого балансу між швидким та надійним наданням послуг та дотриманням обмежень інфраструктури, бюджету та масштабованості. Оскільки API стають все більш невід'ємною частиною сучасних програмних систем, обслуговуючи безліч користувачів та сценаріїв використання, підтримання оптимальної продуктивності стає все більш необхідним [2].

Мета даного дослідження - висвітлити основні методи та кращі практики для підвищення продуктивності API, які тим самим забезпечують швидкий час відгуку для користувача.

Ретельно вивчивши базову архітектуру, розумно використовуючи механізми кешування, оптимізуючі навантаження запитів та відповідей, розробники зможуть по-справжньому використати потенціал API для створення гнучких та ефективних цифрових платформ.

Таким чином, була поставлені наступні задачі:

- провести аналіз існуючих методів оптимізації інтеграції сервісів;
- спланувати експериментальну частину дослідження, що включає розробку сервісів та інтеграцій між ними;
- провести експериментальне дослідження методів обробки запитів у мікросервісній архітектурі.

В роботі було розглянуто та проаналізовано наступні методи оптимізації інтеграцій між сервісами при розробці фреймворку для e-commerce рішень:

- використання ефективних алгоритмів та структур даних для зниження складності обчислень, а також багатопоточної обробки запитів;
- балансування навантаження: розподіл вхідних API-запитів між кількома серверами дозволяє усунути вузькі місця та забезпечити стабільну продуктивність у періоди підвищеного попиту;
- кешування: впровадження стратегій кешування, таких як прикордонне кешування або кешування на рівні програми, може значно скоротити час відгуку за рахунок мінімізації надлишкового пошуку та обробки даних [2].

В ході експериментального дослідження було розроблено декілька сервісів, які були застосовані у розробці фреймворку для e-commerce рішень (див. рис. 1). Розглянемо прийняті рішення:

- інтеграція проходить через ESB сервіс, він і є декоратором для зовнішнього світу; тому запити робляться через нього до TDS сервісу;
 - деякі запити для замірів робляться через сервіс кешування або на пряму через Business logic layer сервіс;
- методи оптимізації сервісів досліджувалися шляхом порівняння продуктивності запитів до БД через API; база даних розгорнута у PostgreSQL:

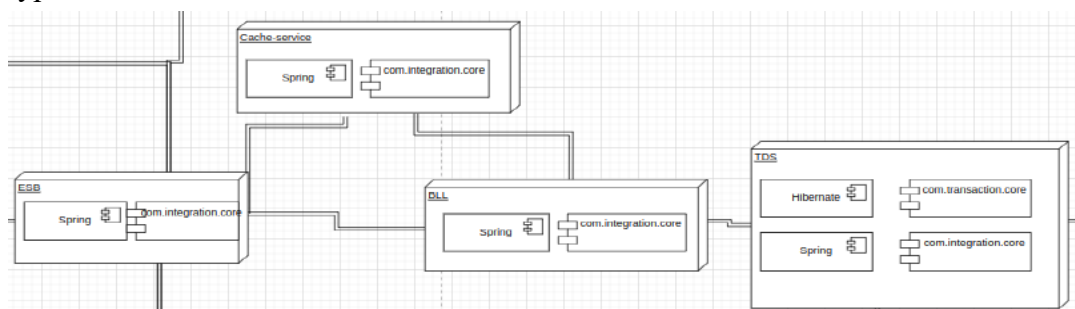


Рисунок 1 – Фрагмент архітектури розробленого фреймворку

На основі розробленого фреймворку було проведено серію експериментів (див. рис. 2), таких як:

- перший виклик базового API сервісу тривав 219 ms;
- багаторазовий виклик базового API тривав сервісу 21 ms;
- перший виклик сервісу з багатопоточною оптимізацією тривав 21ms;

- багаторазовий виклик сервісу з багатопоточною оптимізацією тривав 7ms;
- перший виклик сервісу з механізмом кешування тривав 427ms;
- багаторазовий виклик сервісу з механізмом кешування тривав 4ms.

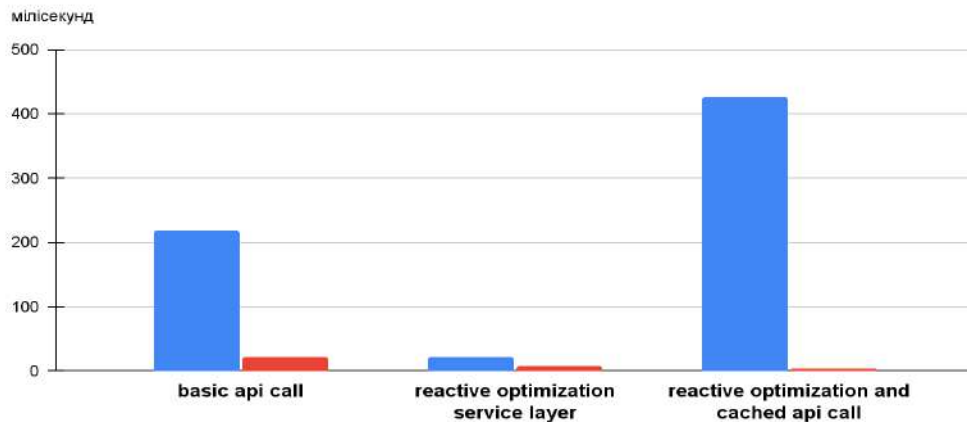


Рисунок 2 – Діаграма результатів експериментів першого та багаторазових викликів системи у декількох варіантах оптимізації

В залежності від вибору методу оптимізації можемо спостерігати зміну у швидкості обробки запитів.

Проведені експерименти дозволили зробити ряд висновків, зокрема, - незважаючи на те, що кешування запитів є затратна операція, але дозволяє зменшити час у 16 разів більше ніж звичайний GET HTTP запит.

Під час програмної реалізації фреймворку було використано реактивне програмування як засіб оптимізації сервісів. Тобто процес запиту замість звичайного поступового процесу розподіляється на декілька паралельних процесів на сервісі та виконується асинхронно. Цей спосіб дає приріст у швидкості обробки запиту у 5 разів навідміну від звичайного запиту.

Висновки. Оптимізація продуктивності роботи сервісів має першочергове значення для забезпечення ефективного та безперебійного обслуговування користувачів. Проведені експерименти дозволили сформулювати наступні рекомендації щодо напрямків підвищення продуктивності API:

- реалізація розумних стратегій кешування дозволяє тимчасово зберігати дані, що часто використовуються, тим самим знижуючи час очікування за рахунок мінімізації надлишкового пошуку та обробки даних;
- кешування є сенс використовувати лише у багато повторних запитах, у простих запитах краще використовувати звичайний запит до сервісу;
- використання методу PATCH для часткових оновлень може значно підвищити продуктивність;

Список інформаційних джерел

1. Bell, Michael. "Introduction to Service-Oriented Modeling". Service-Oriented Modeling: Service Analysis, Design, and Architecture. Wiley & Sons. 2006, pp. 3. ISBN 978-0-470-14111-3.
2. K. Mani Chandy Event-Driven Applications: Costs, Benefits and Design Approaches, California Institute of Technology, 2006.