Ministerul Educaţiei și Cercetării
Universitatea Tehnică a Moldovei
Facultatea Calculatoare Informatică şi Microelectronică
Departamentul Ingineria Software şi Automatică

Admis la susţinere
Şef departament: dr.conf.univ. Fiodorov I.
_____
„__” _____2022

ANALIZA TEHNOLOGIILOR ȘI METODELOR PENTRU OPTIMIZAREA CERERILOR ÎN REST API

Teza de master

Student: **Vlad Ganușceac**
Conducător: **Andrei Poștaru lect.univ.**
Consultanţi: **Svetlana Cojocaru lect.univ.**

Chişinău 2022

Rezumat

Teza intitulată ”**Analiza tehnologiilor și metodilor pentru optimizarea cererilor în REST API**”, prezentată de Ganușceac Vlad, a fost scrisă în limba română. Ea conține 17 figuri, 6 fragmente de cod și 15 referințe. Documentul are mai multe capitole și se începe cu o descriere succintă a problemei existente legată de performanța soluțiilor REST API cînd este necesitatea de a gestiona mai multe cereri concurente bazate pe evenimente. La sfărșitul acestui document sunt formulate concluzii cu privirea la utilizarea metodelor și tehnologiilor descrise în cadrul tezei date de master.

În această lucrare sunt atinse cele mai bune practici pentru dezvoltarea soluțiilor REST API, modalități existente pentru a îmbunătăți rapiditatea răspunsului cît din partea clientului, atît și din partea aplicației și serverului pe care ea este rulată. Sunt descrise mai multe metode care sunt des utilizate pentru îmbunătățirea performanței soluțiilor REST API. Pe de o altă parte, sunt descrise și unile concepte specifice pentru îmbunătățirea performanței cererilor: acțiunile întîrziate și orchestrarea multiierarhică a cererilor, audit de date și funcționalul de fallback. Este un capitol concret dedicat conceptelor descrise, care se referă la modele utilizate din teoria așteptărilor. În capitolul respectiv există argumentarea pentru invocarea sarcinilor cu priorități din mai multe cozi în paralel și cum sistemul de operare va interpreta paralelizmul în joburile rulate periodic.

A treilea capitol demonstrează aspectele specifice în timpul dezvoltării soluției finale prin fragmente de cod atașate și descrierea succintă despre modelarea arhitecturei aplicației, schemelor persistate în baza de date și utilizarea cheilor generice de referință pentru a reduce numărul de apelări către sistemul de gestionarea a bazei de date și de a primi răspunsuri cît mai rapide.

Rezultatul final este reprezentat de o aplicația REST API și cîteva proiecte mici adiționale pentru generarea datelor de testare, efecuare testelor de reper (benchmarking) și analiza rezultatelor testării nefuncționale de performanță. În timpul dezvoltării aplicației a fost utilizată o baza de date relațională (PostgreSQL) împreună cu contexte generice create direct din aplicație, iar pentru cazurile speciale au fost utilizate baza de date NoSql ElasticSearch și serverul de stocare a obiectelor MinIo.

Abstract

Thesis work entitled ”Analysis of technologies and methods for optimizing applications in REST API”, presented by Ganușceac Vlad, was written in Romanian language. It contains 17 figures, 6 code snippets and 15 references. The document has several chapters and begins with a brief description of the REST API solutions' related problem when there is strict necessity to manage multiple concurrent requests based on events. At the end of this document, conclusions are drawn regarding the use of methods and technologies described in this master's thesis.

This thesis document describes the best practices for developing REST API solutions, enumerates existing techniques which are used to improve the speed of response from both the client and the application and the server it is running on. There are described several methods that are often used for improvement are described performance of REST API solutions. On the other hand, there are are also described some specific concepts for improving the performance of requests: delayed actions and multi-hierarchical orchestration of requests, data audit used with the fallback functionality. There is a concrete chapter dedicated to the described concepts and it refers to the models used in the expectation theory. In the corresponding chapter there is argumentation for invoking tasks with priorities from multiple queues in parallel and there is explanation on how the operating system will interpret parallelism in periodically rolled jobs.

The third chapter demonstrates the specific aspects during the development of the final solution through attached fragments of code and a brief description of the application architecture modeling, persistent schemas based on application data and the use of generic reference keys in order to reduce the number of calls to the database management system and to receive responses as soon as possible.

The final result is a REST API application and few small additional projects for generation of test data, benchmarking and analysis of non-functional performance test results. A relational database was used during the development of the application (PostgreSQL) together with generic contexts which were created directly from the application. For the special cases the NoSql ElasticSearch database and the MinIo object storage server were used.

CUPRINS

Introducere

E deja o normă că pentru dezvoltarea aplicațiilor moderne avansate sunt utilizate serviciile care divizează logica de client și server. Sunt create API-urile care dau posibilitate clienților să facă explorarea resurselor. În practică orice API tinde de a respecta filosofia REST. În fiecare an numărul API-urilor crește, dezvoltatorii încerc să creeze aplicații care corespund tuturor standandelor contemporane. Iar pentru împlinirea acestui lucru, soluțiile backend trebuie să fie destul de avansate în ceea ce privește rapiditatea răspunsurilor, utilizarea adecvată a memoriei pe server și securitatea resurselor explorabile.

Sunt mai multe modalități care permit împlinirea condițiilor menționate mai sus. Dar nu orice optimizare conduce la rezolvarea problemei performanței. Pentru a optimiza timpul de răspuns de la REST API pot fi utilizate aspecte exterioare: diferite tehnici utilizate de dezvoltatori frontend, adăugarea memoriei adiționale pe server, unde aplicația este rulată etc. Optimizarea aspectelor externe conduce la îmbunătățirea performanței aplicației, dar ea nu rezolvă problema în "rădăcina" ei. Problema principală stă nemijlocit în aplicația backend. Ea poate să fie destul de complexă, deoarece trebuie de început cu analiza arhitecturii pe care o utilizează API. Următoarele investigații vor fi effectuate asupra interpelărilor către baza de date. E important ce, cum și în ce cantități este extras din baza de date. Următorul pas poate fi legat de păstrarea rezultatelor temporare în memorie și invalidarea obiectelor respective după un anumit timp. Sunt mai multe modalități și tehnologiicare permit efectuarea optimizărilor în backend.

Problemele cu performanță pot fi legate indirect cu aplicația care face gestionarea resurselor. E vorba de volumul de date persistat pe server, unde se află API. Problema dată este legată cu gestionarea contentului. Pentru păstrarea diferitor documente în aplicație, fie asta imagini sau orice alt tip de documente, este recomandat de a utiliza serviciile furnizorilor soluțiilor cloud, deoarece ele sunt scalabile și accesibile în orice moment.

Problemele niciodată nu apar din niciunde și nu se rezolv la clipirea ochiului – optimizarea performanței a unui REST API se începe de la colectarea metricilor care indică erori și eșecuri întîlnite pe parcursul rulării lui pe server. Pentru detectarea problemelor existente se face o analiză destul de complexă, începînd de la cerințele funcționale către sistem și terminînd cu testarea nefuncțională a soluției.

Optimizarea cererilor în aplicațiile REST API este destul de actuală și va rămîne așa, deoarece întotdeauna vor apărea probleme noi și metode care le vor rezolva.

References

1 Resursa web, *Cookies vs localstorage* https://atendesigngroup.com/articles/cookies-are-good-speed-better-local-best

2 Resursa web, *Lazy loading* https://developer.mozilla.org/en-US/docs/Web/Performance/Lazy_loading

3 Resursa web, *Autentificarea Bearer* https://swagger.io/docs/specification/authentication/bearer-authentication/

4 Resursa web, *Arhitectura Onion* https://code-maze.com/onion-architecture-in-aspnetcore/

5 Resursa web, *Circuit breaker pattern* https://docs.microsoft.com/en-us/azure/architecture/patterns/circuit-breaker

6 Resursa web, *Open source circuit breaker libraries on github* https://awesomeopensource.com/projects/circuit-breaker

7 Martin Fawler, Resursa web, *CQRS* https://martinfowler.com/bliki/CQRS.html

8 Resursa web, *Fallback pattern* https://badia-kharroubi.gitbooks.io/microservices-architecture/content/patterns/communication-patterns/ fallback-pattern.html

9 Resursa web, *serverul MinIo* https://min.io/

10 Paolo Gomes, Resursa web, *AAA pattern* https://medium.com/@pjbgf/title-testing-code-ocd-and-the-aaa-pattern-df453975ab80

11 Resursa web, *Teoria de așteptare: definiție, istorie și aplicații din viața reală* https://queue-it.com/blog/queuing-theory/

12 Resursa web, *Notație Kendall Lee pentru sistemele de coadă* https://www.chegg.com/homework-help/definitions/kendall-lee-notation-for-queue-systems-31

13 Resursa web, *Coada cu priorități* https://encyclopediaofmath.org/wiki/Queue_with_priorities

14 Resursa web, *Diferența dintre program, proces și fir de execuție* https://www.backblaze.com/blog/whats-the-diff-programs-processes-and-threads/

15 Resursa web, *Diferența dintre job, task și proces* https://www.geeksforgeeks.org/difference-between-job-task-and-process/