ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені І.І.МЕЧНИКОВА

(повне найменування вищого навчального закладу)

Факультет математики, фізики та інформаційних технологій

(повне найменування інституту, назва факультету (відділення))

Кафедра математичного забезпечення комп'ютерних систем

(повна назва кафедри (предметної, циклової комісії))

Кваліфікаційна робота

на здобуття ступеня вищої освіти «магістр»

(освітньо-кваліфікаційний рівень)

на тему «Підвищення надійності мікросервісної архітектури освітньої платформи»,

«Improving the reliability of the microservice architecture of the educational platform»

Виконав: студент денної форми навчання

спеціальності 126 – Інформаційні системи і технології

(шифр і назва напряму підготовки, спеціальності)

Освітня програма «Інформаційні системи та технології»

(назва освітньої програми)

Козлов Микита Сергійович

(прізвище, ім'я, по-батькові)

Керівник канд. фіз.-матем. наук, доцент Петрушина Т.І.

(науковий ступінь, вчене звання, прізвище та ініціали, **підпис**)

Рецензент канд. фіз.-матем. наук, доцент Крапівний Ю.М.

(науковий ступінь, вчене звання, прізвище та ініціали)

Рецензент _____.

(науковий ступінь, вчене звання, прізвище та ініціали)

Рекомендовано до захисту:

Протокол засідання кафедри

№__від «__»_____2022 р.

Завідувач кафедри

_____ Євгеній МАЛАХОВ

(підпис) (прізвище, ініціали)

Захищено на засіданні ЕК № ___

протокол № __ від «___» ____ 2022 р.

Оцінка _____/_____/_____

(за національною шкалою, шкалою ECTS, бали)

Голова ЕК

_____ Володимир ВИЧУЖАНІН

(підпис) (прізвище, ініціали)

Одеса – 2022

АНОТАЦІЯ

Магістерська робота сприяє підвищенню надійності мікросервісної архітектури. Задля цього використані методи, алгоритми їх спільного застосування і практики підвищення надійності мікросервісної архітектури веб-додатку.

Мета роботи – у вразливих місцях програми застосувати різні методи для підвищення надійності мікросервісної архітектури, а отже і всього веб-додатку.

В рамках роботи спроектовано сервіси, які в сукупності забезпечують роботу освітньої платформи, побудованої на мікросервісах. Сервіс управління контентом, сервіс забезпечення безпеки, з'єднувальний модуль, що забезпечує зв'язок між усіма сервісами, існуючими і потенційними до появи в майбутньому. Для підвищення надійності застосовані такі методи і практики:

– балансувальник навантажень;

– паттерн CirquitBreaker;

– метод динамічних тайм-аутів.

Вищезазначені методи використовується у комбінації.

Для тестування використовувались тести навантаження, написані за допомогою фреймворка Gatling.

Інструментами розробки були:

– фреймворк – Spring Boot;

– мова програмування Java (v17);

– СУБД для зберігання призначених для користувача даних і даних контенту – PostgreSQL;

– технологія Spring Cloud.

Результатом роботи є розроблена модель (правила) поведінки системи при великому навантажені, яка не погіршують користувацький досвід. Також, реалізовані мікросервіси і з'єднувальний модуль, створений і протестований механізм спілкування між розробленими сервісами.

ABSTRACT

Master's work contributes to improving the reliability of microservice architecture. For this purpose, methods, algorithms for their joint application and practices for improving the reliability of microservice architecture of a web application are used.

The purpose of the work is to apply various methods in vulnerable places of the program to increase the reliability of the microservice architecture, and hence the entire web application.

As part of the work, services designed that together ensure the operation of an educational platform built on microservices. Content management service, security service, connecting module that provides communication between all services, existing and potential to appear in the future.

To improve reliability, the following methods and practices applied:

– load balancer;

– CirquitBreaker pattern;

– dynamic timeout method.

The above methods are used in combination.

For testing, load tests written using the Gatling framework used.

The development tools were:

– framework - Spring Boot;

– Java programming language (v17);

– DBMS for storing user data and content data - PostgreSQL;

– Spring Cloud technology.

The result of the work is a model (rules) of system behavior under heavy load that does not worsen the user experience. Also, microservices and a connecting module were implemented, a communication mechanism between the developed services tested.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ

1) Ендпоінт (з англ. – endpoint) – точка зв’язку між сервісами. Використовується в організації API.

2) Лог (з англ. – Log) – повідомлення, які повертає метод, що виконується під час роботи програми. Можуть бути інформативними, попереджувальними про можливу проблему та проблемними, тобто такими, що сповіщають про наявну проблему.

3) Оркестрація контейнерів – це автоматизація та управління життєвим циклом контейнерів і послуг.

4) Фреймворк (з англ. – framework) – програмна платформа, що визначає структуру програмної системи.

5) Персентиль (з англ. – percentile) – методика вимірювання в статистиці, яка показує відсоток значень вимірюваної метрики, що перебуває нижче значення персентиля.

6) Gradle – система автоматичного складання, побудована на принципах Apache Ant і Apache Maven, але надає DSL на мовах Groovy і Kotlin замість традиційної XML-подібної форми представлення конфігурації проекту.

7) Вебхук (з англ. – Webhook) – метод збільшення або розширення функціональності веб-сторінки або веб-застосунку за допомогою користувацьких зворотних викликів.

8) EDA (з англ. – Event-driven architecture) – шаблон архітектури програмного забезпечення, який призначений для створення подій, їх виявлення, споживання і реагування на них.

9) ACID (з англ. – Atomicity, Consistency, Isolation, Durability) – це набір властивостей, що гарантують надійну роботу транзакцій бази даних: атомарність, узгодженість, ізольованість, довговічність.

ВСТУП

В наші дні веб-додатки є важливим джерелом отримання інформації, яким користуються мільйони людей.

Цей факт показує, що важливо створювати програмні рішення так, щоб вони справно працювали під великим навантаженням і були здатні повною мірою покрити запити користувачів.

Для досягнення цієї мети існує безліч рішень, проте питання їх правильного використання у кожній конкретній ситуації є невирішеним.

Справність роботи веб-додатку показує його надійність. У цій роботі розглянуті існуючі методи підвищення цього показника, також за яких умов і що потребує застосування того чи іншого методу в конкретному місці програми. Місця застосування таких методів – є вузлами зменшення надійності програми. Зниженню показника надійності сприяє збільшення кількості даних програми, ускладнення структури зв'язків між її компонентами та багато іншого.

Об'єктом дослідження є процес підвищення надійності веб-додатку. Цей показник, надійність, складається з відмовостійкості та живучості додатку.

Саме використання мікросервісної архітектури покращує стійкість до відмови. Відомо, що одиницею мікросервісної архітектури є власне сервіс. Таким чином, правильна організація зв'язків між сервісами дозволяє отримати від програми таку поведінку, що зупинка одних сервісів не спричиняє зупинку інших. В результаті робота користувача з додатком не буде перервана і він не помітить проблем у роботі системи, а команда розробників під час збою зможе відновити окремі сервіси, не торкаючись робочих сервісів та роботи програми в цілому.

Тим не менш, цього не достатньо для повністю якісної роботи програми. Сильні сторони мікросервісної архітектури слід доповнити методами, які точково підвищують одну з основних характеристик програми – надійність.

Часто найоптимальніше у кожному конкретному випадку використання цих методів породжує багато компромісів. Так, застосування, наприклад, одного з найпростіших методів «so_timeout» спричинить скасування запиту на сервіс, що вийшов з ладу. Безперечно, в результаті застосування цього методу програма буде працювати стабільніше, однак буде певний час, коли користувачі не отримають інформацію, яку вони запитують.

Таким чином, проблемою є – таке застосування методів підвищення надійності, при якому буде досягатися «розумний» компроміс між дійсно високою надійністю та повним задоволенням потреб усіх користувачів.

Мета роботи – застосувати різні методи у вразливих місцях програми для підвищення надійності мікросервісної архітектури, а отже і всього веб-додатку.

Предметом дослідження є методи, що підвищують надійність веб-додатку. А також таке їх застосування, при якому можна досягти високого ступеню надійності, задовольнити потребам користувачів та при цьому не використати всі існуючі ресурси.

Робочим процесом даного дослідження є пошук методів підвищення надійності додатка, визначення критичних місць, у яких необхідно застосувати вивчені методи та, власне, саме їх застосування. Середовищем останнього є раніше розроблена в рамках бакалаврської роботи – освітня платформа, побудована з застосуванням мікросервісної архітектури.

ВИСНОВКИ

В ході роботи розроблено систему з використанням мікросервісної архітектури. Саме використання мікросервісної архітектури покращує стійкість до відмови. Таким чином, правильна організація зв'язків між сервісами дозволила отримати від програми таку поведінку, що зупинка одних сервісів не спричиняє зупинку інших. В результаті робота користувача з додатком не переривається і він не помічає проблем у роботі системи, а команда розробників під час збою має можливість відновити окремі сервіси, не торкаючись робочих сервісів та роботи програми в цілому.

Для підвищення надійності розробленої навчальної платформи використані наступні технології:

– Load balancer. Сервіси програми масштабуються при необхідності та між ними балансується навантаження. Друга функція виконується за попередньо створеною конфігурацією балансера, перша – за результатами роботи створеної моделі, побудованої на метриках.

– Prometheus & Grafana. Використовуються для збирання метрик програми. До списку стандартних метрик, крім таких як: «завантаженість ЦП», «кількість пам'яті, що використовується», включені такі як «RPS» і «average Response time».

– Sentry. Використовується для збору та аналізу помилок, що виникають під час роботи програми. Як уже зазначалося раніше, цей фреймворк дозволяє групувати помилки як потрібно бізнесу. Аналіз помилок показує, на що необхідно звернути увагу і який сервіс вимагає виправлень у майбутньому, а на поточний момент – більших апаратних ресурсів;

– Circuit Breaker. Цей патерн замість виділення додаткових ресурсів сервісу дозволяє повідомити іншим сервісам про заборону надсилання запитів на зупинений вузол програми, щоб уникнути збоїв в інших місцях. Доповнюючи, у даній роботі це один з механізмів реалізації бізнес логіки, а саме – коли

масштабування чи виділення додаткових ресурсів сервісу недоцільно, треба деякий час до нього не звертатися.

– Gatling. Одна з головних складових кожного процесу перевірки надійності. Дозволяє навантажити кожен сервіс і перевірити усі існуючи сценарії взаємодії сервісів у системі. Надає швидких механізм впровадження у проекти, може бути використаний як у будь-якому сервісі, так і окремим модулем. Може бути автоматизований шляхом додання тестування навантаження в процес доставки системи кінцевим користувачам. По закінченню тестування надає зручний механізм аналізу результатів у вигляді графіків, які можна налаштувати під потреби бізнес логіки.

Вищезазначені методи працюють за допомогою додаткових метрик і деяких більш простих і розповсюджених практик. Таких як:

– Використання тайм-аутів, замість безкінечного очікування відповіді.

– Динамічне конфігурування сервісів. Механізмом такої конфігурації в роботі є Spring Cloud. Він дозволив як автоматично налаштовувати параметри сервісів, наприклад, тайм-аути. Так і вносити зміни вручну до більш відповідальних параметрів, наприклад, час перебування CircuitBreaker у half-open стані.

Для подальшого підвищення надійності може бути зроблено наступне:

1) Перехід деяких частин системи на безсервісний варіант архітектури, коли запит являє собою «подію», а дані представлені у вигляді «стану» системи.

2) Впровадження більшої кількості зв’язків саме між сервісами без участі Gateway.

Впровадження всіх вищеописаних практик збільшить швидкість роботи системи, зробить сервіси більш незалежними, дасть більше гарантій щодо цілісності даних при переході від одного сервісу до іншого.

За результатами даної роботи опубліковано тези на всеукраїнській конференції [17] та в спільній україно-китайській науковій студентській конференції [18].

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. J. Schneider. SRE with Java Microservices: Patterns for Reliable Microservices in the Enterprise [текст] / Т.: Орили, 2020. – 289.

2. J. Cox. Kubernetes: The Complete Guide to Master the Future of Infrastructure [текст] / Independently published, 2020. – 160.

3. Event-Driven Architecture Patterns. [Електронний ресурс] – Режим доступу: https://medium.com/wix-engineering/6-event-driven-architecture-patterns-part-1-93758b253f47.

4. AWS Documentation. [Електронний ресурс] – Режим доступу: https://docs.aws.amazon.com/.

5. Why AWS ?? ….Advantages and Disadvantages. [Електронний ресурс] – Режим доступу: https://medium.com/analytics-vidhya/why-aws-advantages-and-disadvantages-f0e666b869b3.

6. Circuit Breaker pattern. [Електронний ресурс] – Режим доступу: https://learn.microsoft.com/en-us/azure/architecture/patterns/circuit-breaker.

7. Common Application Properties. [Електронний ресурс] – Режим доступу: https://docs.spring.io/spring-boot/docs/current/reference/html/application-properties.html.

8. Designing Event-Driven Systems. [Електронне видання] – Режим доступу: https://sd.blackball.lv/library/Designing_Event-Driven_Systems_(2018).pdf.

9. Regular expression Denial of Service – ReDoS. [Електронний ресурс] – Режим доступу: https://owasp.org/www-community/attacks/Regular_expression_Denial_of_Service_-_ReDoS.

10. Facing Failure: Tips For Handling A Failed Web Project. [Електронний ресурс] – Режим доступу: https://www.smashingmagazine.com/2014/11/facing-failure-tips-handling-failed-web-project/.

11. What is Load Testing. [Електронний ресурс] – Режим доступу: https://www.dotcom-tools.com/load-testing-performance.

12. Load Testing Best Practices. [Електронний ресурс] – Режим доступу: https://www.blazemeter.com/blog/load-testing-best-practices.

13. A Practical Guide to Chaos Engineering. [Електронний ресурс] – Режим доступу: https://www.cigniti.com/blog/guide-chaos-engineering/.

14. Designing reliable services. [Електронний ресурс] – Режим доступу: https://livebook.manning.com/book/microservices-in-action/chapter-6/.

15. What Is Load Balancing? [Електронний ресурс] – Режим доступу: https://www.nginx.com/resources/glossary/load-balancing/.

16. Spring Boot Reference Documentation. [Електронний ресурс] – Режим доступу: https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/.

17. Козлов М.С., Петрушина Т.І. Аналіз деяких методів підвищення надійності мікросервісної архітектури веб-додатку / Інформатика, інформаційні системи та технології: тези доповідей дев'ятнадцятої всеукраїнської конференції студентів і молодих науковців. Одеса, 29 квітня 2022 р. – Одеса, 2022. – с. 103-104.

18. Kozlov M. Analysis of some methods of improving the reliability of microservation architecture of a web-application / Conference proceedings of the 1st Student Scientific Conference of Joint Research Cooperation between Odesa I.I. Mechnikov National University and Huaiyin Institute of Technology (March 28-29, May 16, 2022) / MOES of Ukraine; Odesa I.I. Mechnikov National University; – Dnipro: Serednyak TK, 2022. – p.134-136.