Зборник радова Факултета техничких наука, Нови Сад

UDK: 004:007
DOI: https://doi.org/10.24867/25OI05Djordjevic

РАЗВОЈ МИКРОСЕРВИСНЕ АПЛИКАЦИЈЕ ЗА ОНЛАЈН ТРГОВИНУ

DEVELOPMENT OF A MICROSERVICES APPLICATION FOR ONLINE COMMERCE

Софија Ђорђевић, *Факултет техничких наука*, Нови Сад

Област – ИНЖЕЊЕРСТВО ИНФОРМАЦИОНИХ СИСТЕМА

Кратак садржај – *Како се тржиште и потребе корисника мењају, зарад успешнијег пословања, компаније стално развијају и унапређују своје апликације, а са њима се уједно мењају комплексност и величина података. У таквим случајевима, микросервиси се често испостављају као ефикасна опција за руковање комплексношћу и обимом. Микросервисну архитектуру имплементира све већи број великих компанија у циљу повећања скалабилности, смањења сложености апликација, олакшавања проширења развојних тимова и успешног постизања агилности. Циљ рада јесте развој апликације за онлајн трговину која подржава микросервисну архитектуру, ради остварења бољих перформанси, одрживости и ефикацности система. Имплементација је спроведена у Spring Boot и Spring Cloud технологијама, док је за базу података коришћен алата PostgreSQL.*

Кључне речи: *информациони системи, микросервиси, веб апликација, Spring Cloud*

Abstract *– As the market and customer needs evolve, companies continually develop and enhance their applications for more successful business operations. Alongside these changes, the complexity and volume of data also evolve. In such cases, microservices often emerge as an efficient option for handling complexity and data volume. An increasing number of large companies are implementing microservices architecture to increase scalability, reduce application complexity, facilitate the expansion of development teams, and achieve successful agility. The goal of this work is the development of an online commerce application that supports a microservices architecture to achieve better system performance, sustainability, and efficiency. Implementation has been carried out using Spring Boot and Spring Cloud technologies, with PostgreSQL as the database platform.*

Keywords *– Information systems, microservices, web application, Spring Cloud*

1. УВОД

Како величина апликација расте током година развоја, постаје све теже одржавати и мењати овакве апликације. Могуће је одржавати и развијати традиционални монолитни софтверски систем, али на крају постаје очигледно да се морају извршити промене у архитектури целе апликације [1].

Данас је развој софтверских архитектура усмерен на постизање веће јасноће и ефикасности путем боље организације и поделе одговорности унутар система. Овај концепт раздвајања одговорности значи способност декомпозиције система на мање, међусобно повезане модуле који чувају тајност својих имплементација и комуницирају преко прецизно дефинисаних интерфејса како би пружили функционалности. Такви модули се зову микросервиси.

Неколико компанија је недавно мигрирало или размишља о миграцији са постојећих апликација на микросервисну архитектуру, а такође врши се и креирање нових апликација које су засноване на принципу микросервиса [2]. Компаније као што су *Amazon, Netflix, Linkedin, SoundCloud* и многе друге су прешле на микросервисну архитектуру, јер је њихову постојећу монолитну апликацију било превише тешко одржавати, развијати и ширити [1].

2. СРОДНА ИСТРАЖИВАЊА

Студија [4] закључује да у случају мањег оптерећења, односно мање од неколико стотина корисника, монолитна апликација може радити мало боље од микросервисне апликације. Генерално говорећи, зарад испуњавања потреба великих компанија и обављања услуга за већи број корисника, потребно је осигурати микросервисну архитектуру како би се повећала скалабилност и перформансе. Истраживање [4] такође открива да употреба технологије *Consul service discovery,* у оквиру микросервисне архитектуре, постиже боље резултате у погледу протока или броја обрађених захтева у секунди, што доприноси побољшању од 4% у односу на друге механизме.

С друге стране, студија [5] је спровела истраживање које указује да предност микросервисне архитектуре апликације лежи у једноставности одржавања, затим у чињеници да су функционалности раздвојене према модулима, као и у надежности и скалабилности, јер грешка у микросервису утиче само на тај микросервис.

Како је економски фактор од изузетног значаја у пословању, аутори [3] су се фокусирали на поређење инфраструктурних трошкова покретања и скалирања апликације у облаку. Изведен је закључак да употреба сервиса посебно дизајнираних за процес имплементације апликације у облаку и скалирање микросервиса, као што је *AWS Lambda*, омогућава компанијама да смање своје трошкове инфраструктуре до 77,08%.

Као главне покретаче миграције, аутори [6] идентификују побољшање скалабилности, доступности, могућности одржавања и толеранције на

НАПОМЕНА:
Овај рад проистекао је из мастер рада чији ментор је био др Срђан Сладојевић, ванр. проф.

грешке апликације. Ипак, апликације засноване на микросервисима долазе са појединим изазовима, укључујући:

- идентификовање оптималних граница микросервиса,
- оркестрацију сложених сервиса,
- одржавање конзистентности података и управљање трансакцијама између микросервиса,
- потешкоће у холистичком разумевању система и
- повећану потрошњу рачунарских ресурса.

Аутори [1] дефинишу следеће две групе изазова који се појављују у току преласка на микросервисну организацију:

- технички изазови и
- организациони изазови.

У даљем тексту биће описани неки од значајнијих примера изазова ових група.

2.1. Технички изазови

Калске, Мäкитало и Микконен у својој студији [1] истичу да је највећи технички изазов дефинисање микросервиса система, тачније граница њихових одговорности. Потребно је прецизно одредити задужења микросервиса, односно пронаћи одговарајућу величину која се подудара са једном функционалношћу и одговорношћу система, како би се избегли непотребни *HTTP (енг. Hypertext Transfer Protocol)* захтеви и одржале добре перформансе система.

Поред тога, студија [1] не препоручује везивање интеграције између микросервиса за специфичну технологију, јер се тиме нарушавају основни концепти микросервисне архитектуре, како тимови могу користити различите програмске језике када имплементирају микросервисе. Као решење, могу се користити технологије које не захтевају посебан програмски језик.

Интерфејс микросервиса треба бити једноставан за коришћење и поседовати подршку за старије верзије *(енг. backwards compatibilty).* На тај начин, увођењем нових функционалности у микросервис, клијенти који користе конкретан микросервис не морају нужно бити ажурирани. Додатно, попут сваког доброг интерфејса, потребно је сакрити детаље имплементације унутра.

Како је управљање подацима важан део сваке апликације, аутори рада [1] указују да за разлику од монолитних апликација, које користе једну релациону базу података за извршавање трансакција, код микросервиса се уочавају базе података по сервисима који успостављају договор о евентуалној конзистентности података како би се извршиле потребне трансакције. Сваки микросервис садржи податке везане искључиво за њега, те се аутоматски успоставља слабо повезивање између микросервиса *(енг. loose coupling).*

2.2. Организациони изазови

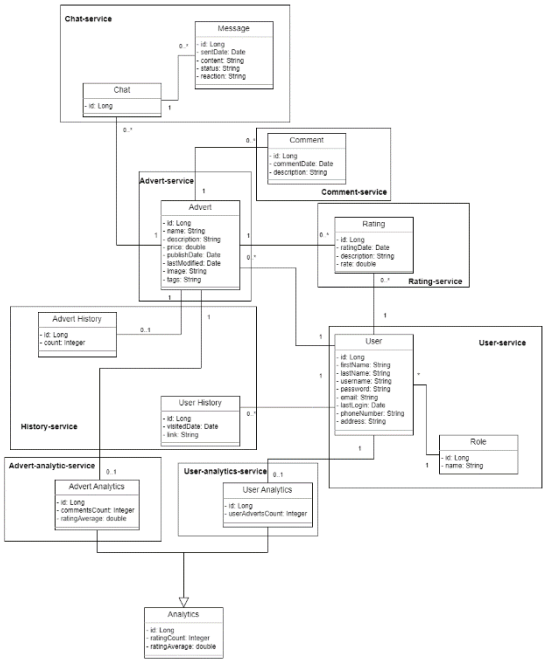
У циљу развоја добре апликације, организација мора ускладити своју структуру и вештине тако да нова архитектура апликације буде подржана [7]. У том смислу, организација треба бити подељена на мање тимове, који су аутономни и одговарају тачно једном микросервису, на коме је постављен фокус тима. Може се уочити значајна разлика од раније ситуације са монолитном апликацијом, када су постојали велики тимови који су имали јасне улоге, као што су обезбеђивање квалитета, развој производа и администрација базе података.

Још један изазов који се појављује јесте усвајање *DevOps (скраћено Development and Operations)* менталитета. Сваки тим мора бити способан за извршавање имплементације апликације у облаку, међутим може се десити одсутству у познавању одговарајућих вештина. Стога чланови тима морају стећи нове вештине кроз едукацију, искуство и адаптацију [1].

У неким случајевима, прелазак на микросервисну архитектуру укључује транзицију са традиционалних модела процеса на агилне методологије, укључујући промену начина размишљања тимова. У овом погледу се као најчешћи проблем појављује упознавање са разноврсношћу технологија и алата коришћених у поменутом процесу. Компаније уочавају потешкоће, пре свега у проналаску и регрутовању квалификованог особља, те касније у току аутоматизације и употребе *CI/CD (енгл. Continuous Integration / Continuous Delivery)* процеса, у циљу успостављања праксе [8].

3. АНАЛИЗА КОРИСНИЧКИ ЗАХТЕВА

У циљу бољег разумевања система и корисничких потреба, систем је пре свега представљен кроз различите *UML (енг. Unified Modeling Language)* дијаграме. Уочава се подела на микросервисе између класа, тако да се јасно издвајају класе које ће бити коришћене у микросервисима ради моделовања ентитета реалог система, као и које су одговорности сваког *(Слика 1.)*:



Слика 1. Дијаграм класа система

Након анализе реалног система, уочени су следећи ентитети: *User, Advert, Comment, Rating, Chat, Message, AdvertHistory, Role, UserHistory* и *Analytics.* Поменути ентитети се користе како би се реализовао пројектовани систем.

3. ИМПЛЕМЕНТАЦИЈА РЕШЕЊА

Након идентификованих корисничких захтева, анализе система пословања и одређивања домена, потребно је имплементирати микросервисну апликацију. Први корак јесте креирање одговарајућег система база података.

3.1. База података

Као што је раније наглашено, суштина микросервисне архитектуре јесте да сваки сервис поседује сопствену базу података, те ће у наставку бити наведене одговарајуће базе података које су послужиле као основа за изградњу сваког микросервиса:

- *advert-service*
- *advert-analytics-service*
- *user-analytics-service*
- *comment-service*
- *history-service*
- *user-history*
- *chat-service*
- *rating-service*
- *user-service*
- *auth-service*

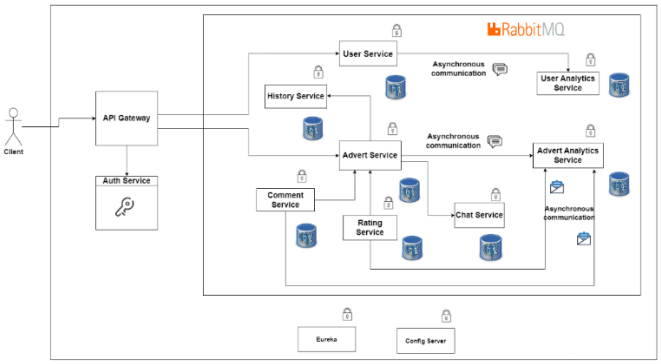
3.2. Микросервисна структура апликације

Предложена микросервисна структура се састоји од једанаест микросервиса са одговарајућим базама података. Користе се систем асинхроне комуникације, као и систем детекције микросервиса и ауторизације.

Структуру апликације сачињавају следећи микросервиси:

1. ***rating-service*** – микросервис намењен за додавање, модификацију или уклањање оцена за неки оглас, односно за неког корисника,
2. ***discovery-service*** - механизам за детекцију микросервиса у апликацији,
3. ***comment-service*** – микросервис задужен за вођење евиденције о коментарима, односно омогућава креирање, модификацију и брисање коментара за огласе,
4. ***advert-analytics-service*** - ангажован за праћење статистичких података о неком огласу, попут броја коментара и оцена на неком огласу, броја прегледа огласа, те просечне оцене коју је оглас добио,
5. ***user-analytics-service*** - микросервис одговоран за статистику корисничких података, попут броја огласа које је корисник објавио, просечне оцене коју је добио од осталих корисника, као и укупног броја оцена,
6. ***history-service*** - микросервис који чува податке о броју прегледа неког огласа и које огласе је корисник најчешће гледао,
7. ***api-gateway*** - проверава све пристигле захтеве, одобрава их и дефинише све доступне руте и путање,
8. ***advert-service*** - задужен за евидентирање огласа, односно праћење огласа које корисници додају, ажурирају или бришу,
9. ***chat-service*** - надлежан за чување свих четова између корисника и порука које се размене,
10. ***user-service*** - намењен чувању евиденције о свим корисницима система и њиховим улогама. Поред тога, одржава основне податке о корисницима попут мејл адресе, броја телефона и адресе и
11. ***auth-service*** - извршава ауторизацију и аутентификацију.

Слика 2. приказује изглед система који је развијан за потребе рада.



Слика 2. Приказ развијане микросервисне апликације

Сваки микросервис чува податке неопходне за извршавање функционалности путем алата *PostgreSQL*. Уколико је потребна комуникација између микросервиса у циљу размене информација или података, користи се *RabbitMQ* механизам асинхроне комуникације како се не би блокирало даље извршавање апликације. Примењује се *API Gateway* сервер као улазна тачка апликације, како би се усмеравали долазни захтеви на исправне сервисе. Додатно, употребљава се *Netflix Eureka Service Discovery* механизам који обезбеђује сервисима међусобну детекцију, тачније аутоматску идентификацију доступних ресурса, попут микросервиса у тренутном окружењу. Поменути механизам представља једну од основних компоненти било које микросервисно-оријентисане апликације јер тачна локација микросервиса није позната и додељена у фази пројектовања.

Auth-service има значајну улогу која се огледа у извршавању ауторизације и аутентификације, те омогућује регистрацију, логовање и идентификацију корисника система. Све пристигле захтеве ће овај микросервис даље проследити управо *auth-service* микросервису. Након тога, микросервис проверава да ли захтевани корисник има потребне привилегије, те ће у складу са тиме одобрити или одбити захтев. Одобрен захтев ће даље *api-gateway* микросервис проследити *discovery-service* микросервису који исти прослеђује одговарајућем микросервису на обраду.

Унутар једног микросервиса се могу уочити компоненте груписане у следећим фолдерима:

- ***config*** - задужен за дефинисање потребних конфигурација у апликацији,
- ***controllers*** - задужен за чување контролера апликације,
- ***dto*** - задужен за складиштење дата трансфер објеката апликације,
- ***models*** - одговоран за дефинисање доменских класа,

- ***services*** - поверен за чување бизнис логике апликације, која користи услуге репозиторијума и
- ***repository*** - дефинише интеракцију са базом података, односно неопходне методе.

4. ЗАКЉУЧАК

У овом раду представљен је процес пројектовања и имплементације микросервисне апликације за онлајн трговину.

Развијана апликација намењена је корисницима који желе да поставе огласе о продаји неког производа, као и корисницима којима је потребан одговарајући производ, те се могу информисати о истом кроз коментаре, оцене других корисника или директну комуникацију са особом која је објавила оглас. Циљ апликације јесте омогућавање једноставнијег и поузданијег претраживања жељених производа по категоријама или кључним речима, као и објављивање огласа, где корисници могу брзо да поставе своје производе на тржиште.

Омогућавањем корисницима да оцењују производе и остављају коментаре, доприноси се изградњи поверења међу корисницима. Такође, овакав механизам помаже будућим купцима да стекну увид у оно што могу очекивати од производа.
Апликација олакшава комуникацију између особа које постављају огласе и потенцијалних купаца. Поменута карактеристика система посебно долази до изражаја када је потребно поставити питања, размотрити детаље или преговарати око цене. Куповина путем апликације омогућава корисницима да купују било где и било када, што је посебно корисно у савременом начину живота. Имплементација онлине платформе, попут описане апликације, осигурава безбедност трансакција и заштиту корисничких података.

У циљу остварења напредне аналитике и персонализације, корисницима се приказују препоруке на основу претходних претрага и интересовања, те се обезбеђује боље разумевање корисника и потреба.
Један од недостатака апликације и могућих унапређења представља регулисање евентуалног повратка производа. Решавање проблема и враћање производа тренутно није обезбеђено, како након договорене продаје корисник брише свој оглас а тиме и све контакт податке и комуникацију са другим корисницима.

На крају, може се извести закључак да развијана микросервисна апликација представља добро решење за повећање ефикасности продаје на тржишту, олакшавајући купцима да на брз начин пронађу жељене производе уз одговарајућу цену. Додатно, на основу потражње се може персонализовати приказ за сваког корисника.

5. ЛИТЕРАТУРА

[1] M. Kalske, N. Mäkitalo, T. Mikkonen, *Challenges When Moving from Monolith to Microservice Architecture*, Lecture Notes in Computer Science, vol. 10544, pp. 32–47, 2018, doi:10.1007/978-3-319-74433-9_3.

[2] L. Baresi, M. Garriga, A. De Renzis, *Microservices Identification Through Interface Analysis, Lecture Notes in Computer Science*, vol. 10465, pp. 19–33, 2017, doi:10.1007/978-3-319-67262-5_2.

[3] M. Villamizar, O. Garcés, et al., *Cost comparison of running web applications in the cloud using monolithic, microservice, and AWS Lambda architectures*, Service Oriented Computing and Applications, vol. 11, no. 2, pp. 233–247, doi:10.1007/s11761-017-0208-y.

[4] O. Al-Debagy, P. Martinek, *A Comparative Review of Microservices and Monolithic Architectures*, in 2018 IEEE 18th International Symposium on Computational Intelligence and Informatics (CINTI), Budapest, Hungary, 2018.

[5] K. Gos,W. Zabierowski, *The Comparison of Microservice and Monolithic Architecture*, in 2020 IEEE XVIth International Conference on the Perspective Technologies and Methods in MEMS Design (MEMSTECH), Lviv, Ukraine, 2020.

[6] G. Blinowski, A. Ojdowska, A. Przybyłek, *Monolithic vs. microservice architecture: A performance and scalability evaluation*, IEEE Access, vol 10, 2022, pp. 20357 – 20374, doi: 10.1109/ACCESS.2022.3152803.

[7] J. Fritzsch, J. Bogner, S. Wagner, A. Zimmermann, Microservices Migration in Industry: Intentions, Strategies, and Challenges, in 2019 IEEE International Conference on Software Maintenance and Evolution (ICSME), Cleveland, OH, USA, 2019.

[8] I. Sommerville, Software Engineering, edition 10th, Pearson, London, 2016 (Chap. 18).

Кратка биографија:

Софија Ђорђевић рођена је 23.6.1998. године у Смедереву. Дипломирала је 2021. године на Факултету Техничких наука на департману за Индустријско инжењерство и менаџмент. Исте године уписује мастер студије на Факултету Техничких наука, на смеру инжењерство информационих система.