

VILNIAUS UNIVERSITETAS  
MATEMATIKOS IR INFORMATIKOS FAKULTETAS  
INFORMATIKOS STUDIJŲ PROGRAMA

Pagrindiniai komunikavimo integracijų mikroservisų  
architektūrose tipai ir jų analizė

Main communication integration types in microservices  
architectures comparison and analysis

Kursinis darbas

Atliko: Lukas Milašauskas (parašas)

Darbo vadovas: Dr. Saulius Minkevičius (parašas)

Vilnius – 2020

## TURINYS

ĮVADAS .....	3
Temos aktualumas .....	3
Problema.....	3
Darbo tikslas.....	3
Uždaviniai tikslui pasiekti .....	3
1. KAS YRA MIKROSERVISAI .....	4
1.1. Monolitinės sistemos ir mikroservisų atsiradimas.....	4
1.2. Mikroservisų pranašumai prieš monolitus.....	4
1.2.1. Technologijų nevienalytiškumas (angl. „ <i>Technology Heterogeneity</i> “) .....	4
1.2.2. Atsparumas (angl. „ <i>Resilience</i> “).....	5
1.2.3. Plečiamumas (angl. „ <i>Scaling</i> “).....	5
1.2.4. Lengvas diegimas (angl. „ <i>Ease of Deployment</i> “).....	5
1.2.5. Organizacinis pasiskirstymas (angl. „ <i>Organizational Allignment</i> “) .....	5
1.2.6. Kompozicija (angl. „ <i>Composability</i> “) .....	5
1.2.7. Optimizuotas pakeičiamumas (angl. „ <i>Optimizing for Replaceability</i> “) .....	5
1.3. Monolitinių sistemų skaidymas į mikroservisus .....	6
2. MIKROSERVISŲ SISTEMOS VIDINIŲ INTEGRACIJŲ TIPAI .....	7
3. SINCHRONINĖS (ANGL. “SYNCHRONOUS”) INTEGRACIJOS .....	8
3.1. Sinchroninių integracijų principas .....	8
3.2. Sinchroninių integracijų technologijų tipai .....	8
4. ASINCHRONINĖS (ANGL. “ASYNCHRONOUS”) INTEGRACIJOS .....	9
4.1. Asinchroninių integracijų principas .....	9
4.2. Asinchroninių integracijų technologijų tipai.....	9
5. SKIRTINGŲ INTEGRACIJŲ TIPŲ PRIVALUMAI IR TRŪKUMAI .....	10
6. REZULTATAI IR IŠVADOS .....	11
6.1. Rezultatai.....	11
6.2. išvados.....	11
SAVOKŲ APIBRĖŽIMAI .....	12

# Įvadas

## Temos aktualumas

Tobulėjant programinės įrangos (toliau PĮ) kūrimo įrankiams ir vis augant informacinių sistemų sudėtingumo poreikiams, kyla daug klausimų PĮ kūrėjams kokią architektūros modelį ir kokiais technologijas pasirinkti, pradedant kurti naują informacinę sistemą. Vis daugėja skirtingų technologijų ir programavimo kalbų, kurios yra pranašesnės už kitas tik siaurose srityse, dėl to dažnu atveju neužtenka pasirinkti vieną technologiją ar programavimo kalbą norint sukurti kokybišką ir tvirtą PĮ. Kai kurios technologijos yra pranašesnės resursų taupyme, kitos pranašesnės daug skirtingų bibliotekų palaikymu ir lengvai naudojama aplikacijų programavimo sąsają (angl. „*Application programming interface*“ arba „*API*“) ir t.t. Kuriant naują PĮ reikia gerai išsianalizuoti tuo metu esamas technologijas ir jų privalumus. Įmonės yra linkusios kurti PĮ tokiais technologijomis, kurių specialistų yra daug ir kurie norėtų palaikyti ir kurti jomis. Renkantis skirtingas technologijas, atsiranda problema kaip jas apjungti, kad veiktų vieningai. Tokiu atveju, galima naudotis saitynų tarnybų pagalba (angl. „*Web services*“), tačiau to neužtenka, nes kas, jeigu norime, skirtingus funkcionalumus įgyvendinti skirtingų technologijų pagalba. Tada dažnu atveju naudojamas mikroservisų (angl. „*Microservices*“) architektūrinis modelis. Remiantis šiuo modeliu būtų kuriami atskiri moduliai (taip pavadinsime atskirus sistemos vienetus), kurių kiekvienas būtų atsakingas už savo funkcionalumą. Problema su šia architektūra, kad reikia priversti šiuos skirtingus modulius bendrauti tarpusavyje, o tai nebūna taip paprasta.

## Problema

Kokią integracijų tipą rinktis mikroservisų architektūrose, norint įgyvendinti komunikaciją tarp skirtingų servisų.

## Darbo tikslas

Palyginti skirtingus komunikacijų tipus, apžvelgti jų pranašumus ir trūkumus, pateikti technologijų šiem tipam realizuoti pavyzdžių.

## Uždaviniai tikslui pasiekti

1. Remiantis literatūra apibūdinti, kas yra mikroservisai, kuo jie ypatingi, kaip jie projektuojami.
2. Išskirti pagrindinius integracijų ir komunikavimo tipus ir jų savybes.
3. Palyginti skirtingus komunikavimo būdus mikroservisų architektūrose.
4. Pateikti konkrečius komunikavimo integracijų ir technologijų pavyzdžius.
5. Pateikti rekomendacijas, kokiais atvejais, kokius tipus būtų geriau naudoti.

# 1. Kas yra mikroservisai

## 1.1. Monolitinės sistemos ir mikroservisų atsiradimas

Pagal autorių Nicola Dragoni, Saverio Giallorenzo, Alberto Lluch Lafuente, Manuel Mazara publikuotą straipsnį „Microservices: yesterday, today, and tomorrow“ [Mis7] jau 1960-iais buvo susiduriama su problemomis susijusiomis su dižiulio masto PĮ kūrimu ir kaip tai projektuoti. Buvo kuriama daug būdų kaip tai daryti, ir daug toerijų kaip turėtų atrodyti PĮ kodas, ir kaip projektuoti informacines sistemas. Daug vėliau apie 2000 metus, susiformavo sąvokos „Service-Oriented Computing“ (toliau SOC) ir „Service-Oriented Architecture“ (toliau SOA), kurių idėja buvo ir paradigmos buvo apie tai, kad aplikacija, arba kitaip pavadinus servisas, turėtų būti atsakingas už konkretaus resursų ar verslo logikos informaciją. Šį servisą turi būti galima pasiekti su konkrečia technologija ir taip komunikuoti ir gauti informaciją. Taigi remiantis jau anksčiau minėtu straipsniu „Microservices: yesterday, today, and tomorrow“ [Mis7] iš SOC ir SOA kiek vėliau susiformavo mikroservisų idėja ir paradigmos. Pagal Martin Fowler ir James Lewis 2016-ais metais publikuotą straipsnį „Microservices“ [FL14] šis terminas mikroservisai buvo pirmą kartą diskutuotas 2011 metais, Venecijoje vykusiose PĮ kūrimo architektų dirbtuvėse (angl. „*workshop*“). Po metų ta pati grupė architektų nusprendė, labiausiai tinkantis pavadinimas šiam architektūriniam tipui yra mikroservisai (angl. „*microservices*“). Po šiuo terminu slypi daug idėjų ir paradigmų, tačiau pagrindinė mintis yra skaidymas didelės sistemos į mažas „granules“ ir mažus servisus. Taip pat norint apibūdinti senas sistemas, kurios buvo nedalomos ir sistemą yra paleidžiamas kaip vienas vienetas, atsirado terminas „monolitas“ (angl. „*monolith*“). Nors šis terminas ir buvo naudojamas Unix bendruomenės jau ilgą laiką iki mikroservisų susikūrimo. Remiantis Eric Steven Raymon knyga „The Art of UNIX Programming“ [Ray03], kuri buvo išleista 2003 metais, terminas „monolitas“ buvo naudojamas apibūdinti sistemoms, kurios yra per didelės. Taigi, šie du terminai naudojami iki šios apibūdinti informacinių sistemų architektūriniais tipams.

## 1.2. Mikroservisų pranašumai prieš monolitus

Pagrindinius principus ir mikroservisų pranašumus detalai išdėstė Sam Newman savo knygoje „Building Microservices: Designing Fine-Grained Systems“ [New15]. Šiame leidinyje autorius išgrynina keletą labai svarbių mikroservisų privalumų.

### 1.2.1. Technologijų nevienalytiškumas (angl. „*Technology Heterogeneity*“)

Kiekvienas servisas turi atlikti skirtingas funkcijas ir turėti skirtingas atsakomybes. Norint pasiekti geriausius rezultatus galima rinktis skirtingas technologijas, kurios būtų geriausiai pritaikytos konkrečiam uždaviniui spręsti. Todėl mikroservisuose kiekvieną servisą galim projektuoti skirtingomis technologijomis, pavyzdžiui skirtingomis programavimo kalbomis.

#### 1.2.2. Atsparumas (angl. „**Resilience**“)

Atsitikus problemai ir sugriuvus vienai sistemos komponentei, monolitinėje sistemoje tektų perkraudinėti arba taisyti visą sistemą. Mikroservisų architektūroje stambių pasikeitimų būtų išvengta ir žlugtų tik vieno serviso veikimas. Tokiu atveju kiti servais apie tai nežinotų ir veiktų toliau, o norint ištaisyti problemą, užtektų sutaisyti ir perkrauti vieną servisą.

#### 1.2.3. Plečiamumas (angl. „**Scaling**“)

Stambioje monolitinėje sistemoje visos plečiamumo ir efektyvumo problemos sprendžiamos kartu. Mikroservisų sistemoje kiekvieną tokio tipo problemą sprendžiame atskiruose servisuose. Tokiu atveju servais, kurie reikalauja mažiau resursų galima suteikti mažiau resursų, o sunkesniems ir mažiau efektyviems servais išskirti daugiau. Tačiau verta paminėti, kad mikroservisų sistemos plečiamumas ne visada yra geras, tai aprašyta Omar Al-Debagy ir Peter Martinek straipsnyje „A Comparative Review of Microservices and Monolithic Architecture“ [AM18]

#### 1.2.4. Lengvas diegimas (angl. „**Ease of Deployment**“)

Vystant PĮ dažnai susiduriama su diegimo problema. Su naujais funkcionalumais reikia iš naujo diegti naują PĮ versiją. Monolitinėje sistemoje tenka iš naujo sudiegti visą sistemą, tačiau mikroservisų sistemoje galima sudiegti tik tuos servais, kurie yra susiję su pakeitimais.

#### 1.2.5. Organizacinis pasiskirstymas (angl. „**Organizational Alignment**“)

Dažnai įmonėse prie informacinės sistemos dirba daug žmonių. Jie būna pasiskirstę komandomis ir turi skirtingas atsakomybes. Mikroservisų sistemose galima išvengti komunikavimo incidentų ir kiekvienai komandai dirbti su skirtingais servais. Tokiu pavydžiu dirba stambi informacinių technologijų (toliau IT) įmonė „Netflix“.

#### 1.2.6. Kompozicija (angl. „**Composability**“)

Įmonės dažnai susiduria su problema, kai tas pats funkcionalumas reikalingas keliose informacinėse sistemose. Mikroservisų architektūroje, kadangi sistema susideda iš mažų autonomiškų servisų, juos galima atskirti ir perpanaudoti skirtingose sistemose, arba kitai sistemai suteikti prieigą tik prie konkrečių resursų, o ne visos sistemos.

#### 1.2.7. Optimizuotas pakeičiamumas (angl. „**Optimizing for Replaceability**“)

Dirbant vidutinio dydžio arba didelėse imonėse dažnai susiduriama su problema, kai naudojama sena kodo bazė ir pasenusios technologijos. Dažnai tokią sistemą reikia atnaujinti siekiant efektyvumo arba palaikymo paprastumo. Tokiu atveju norint atnaujinti bibliotekas arba technologijas, tenka iš naujo perprogramuoti dalį sistemos. Mikroservisų architektūros pagalba, tai tampa žymiai paprasčiau, kai užtenka perrašyti konkretų servisą, neličiant likusios informacinės sistemos.

### 1.3. Monolitinių sistemų skaidymas į mikroservisus

Paskutinį dešimtmetį tapo gan populiariu stambaus masto monolitus skaidyti į mikroservisus, tačiau tai nėra taip paprasta. Visų pirma skaidant monolitą į atskirus servisus labai svarbu identifikuoti, kokie mažesni servisai bus. Serviso riboms apibrėžti panaudosime Micheal C. Feathers knygoje „Working Effectively with Legacy Code“ [Fea04] apibrėžtą terminą „siūle“ (angl. „seam“). Siūle šiuo atveju yra kodo dalis, kuri yra izoliuota ir autonomiška. Siūles ir bus mūsų atskiri mikroservisai. Autorė Susan J. Fowler savo knygoje „Production-Ready Microservices“ [Fow17] aprašė patarimus ir žingsnius, kaip reikėtų skaidyti monolitą į mikroservisus. Ji pamini, kad tai reikėtų daryti etapais:

1. Monolitinę aplikaciją paleisti su tiek kopijų kiek turėsime siūlių.
2. Išskirstyti kvietimus į kopijas, pagal tai kokias siūles kopijos reprezentuoja.
3. Išvalyti aplikacijos kopijas, paliekant tik siūlių funkcionalumą.

Verta paminėti, kad po kiekvieno išvardinto žingsnio bus atliekami testavimai, kurie patikrintų ar sistema veikia korektiškai.

## 2. Mikroservisų sistemos vidinių integracijų tipai

Mikroservisų sistemos vidinių integracijų tipai

### 3. Sinchroninės (angl. “synchronous”) integracijos

#### 3.1. Sinchroninių integracijų principas

Sinchroninių integracijų principas

#### 3.2. Sinchroninių integracijų technologijų tipai

Sinchroninių integracijų technologijų tipai



## 4. Asinchroninės (angl. “asynchronous”) integracijos

### 4.1. Asinchroninių integracijų principas

Asinchroninių integracijų principas

### 4.2. Asinchroninių integracijų technologijų tipai

Asinchroninių integracijų technologijų tipai

## 5. Skirtingų integracijų tipų privalumai ir trūkumai

Skirtingų integracijų tipų privalumai ir trūkumai

## 6. Rezultatai ir išvados

### 6.1. Rezultatai

Rezultatai

### 6.2. išvados

išvados

- [AM18] Omar Al-Debagy and Peter Martinek. A comparative review of microservices and monolithic architectures. <https://arxiv.org/pdf/1905.07997.pdf>, 2018.
- [Fea04] Micheal C. Feathers. *Working Effectively with Legacy Code*. Prentice Hall PTR, Upper Saddle River, United States of America, 2004. 29 psl.
- [FL14] Martin Fowler and James Lewis. Microservices. <https://martinfowler.com/articles/microservices.html>, 2014.
- [Fow17] Susan J. Fowler. *Production-Ready Microservices*. O'Reilly Media, Sebastopol, United States of America, 2017. 1 psl.
- [New15] Sam Newman. *Building Microservices: Designing Fine-Grained Systems*. O'Reilly Media, Sebastopol, United States of America, 2015. 39 psl.
- [Ray03] Eric Steven Raymond. *The Art of UNIX Programming*. Addison-Wesley, Upper Saddle River, United States of America, 2003. 1 psl.

## Sąvokų apibrėžimai

Sąvokų apibrėžimai ir santrumpų sąrašas sudaromas tada, kai darbo tekste vartojami specialūs paaiškinimo reikalaujantys terminai ir rečiau sutinkamos santrumpos.