TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Jarmo Jevonen 193380

Rakendusliideste haldusplatvormide analüüs mikroteenuste näitel

Bakalaureusetöö

Juhendaja: Einar Kivisalu
MSc

Tallinn 2022

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Jarmo Jevonen

16.05.2022

Annotatsioon

Mikroteenuste arhitektuuri kasutamine, mille puhul iga äriprotsess või äriline väärtus on eraldatud eraldi teenuseks, toob kaasa endaga mitmeid murekohti. Üheks neist on mikroteenuste kasutajatele esitamise viis ja ühise loogika rakendamine.

Käesoleva bakalaureusetöö eesmärk oli analüüsida rakendusliideste haldusplatvormides leiduvaid funktsioone ja käsitleda levinud arendusmustreid, võttes arvesse mikroteenuste arhitektuuri eripärasid. Samuti oli eesmärgiks leida analüütiliste hierarhiate protsessi abil parim rakendusliideste haldusplatvorm.

Analüütiliste hierarhiate protsessi käigus anti nelja rakendusliideste haldusplatvormide omavahelisel võrdlemisel 90 hinnangut 15 kriteeriumi osas ning lõpptulemusena osutus parimaks rakendusliideste haldusplatvormiks WSO2.

Töö tulemusena esitati rakendusliideste halduses kasutatavaid arendusmustreid koos loodud joonistega. Esitati levinud lähenemisi koos nende nõrkustega ning pakuti neile alternatiive. Koostati analüütiliste hierarhiate otsustusmudel, mis on abiks platvormi valimisel.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 30 leheküljel, 5 peatükki, 8 joonist, 24 tabelit.

Abstract

Analysis of API Management Platforms Using Microservices

Microservices architecture where every business capability or functionality is separated into an independent service raises many technical difficulties. One of them is presenting the system's endpoints for the users and applying common logic for services.

The goal of this bachelor's thesis was to analyze application programming interface management platforms, cover different development patterns used in microservices architecture, and find the best management platform using the analytical hierarchy process.

Some application programming interface management platforms label their whole solution as a gateway even if it provides extra functionality which does not belong under gateway's responsibilities.

During the analytical hierarchy process, a comparison of four application programming interface management platforms was performed based on 15 criteria with 90 assessments. Authorization and authentication, resiliency and observability were viewed as the most important criteria during the evaluation. The results of this process deemed WSO2 as the best platform among the alternatives. Ambassador platform came in close second place even though it does not support as many functions as others.

As a result of the work, the development patterns used in the management of APIs were presented together with the corresponding figures. Weaknesses of different approaches were presented and alternatives for them were given. A decision model for analytical hierarchies was developed to assist in the selection of the platform.

The thesis is in Estonian and contains 30 pages of text, 5 chapters, 8 figures, 24 tables.

Lühendite ja mõistete sõnastik

Antimuster	*Anti-pattern*, arendusmuster, millel on selgelt negatiivsed tagajärjed, mis ei pruugi koheselt märgatavad olla
API	*Application Programming Interface*, rakendusliides
GraphQL	Päringukeel rakendusliideste loomiseks
gRPC	*Good Remote Procedure Call*, kaugprotseduurikutse
HMAC	*hash-based message authentication code*, räsil põhinev sõnume autoriseerimise kood
HTML	*HyperText Markup Language*, veebilehtede märgendamise keel
JSON	*JavaScript Object Notation*, andmevahetusvorming
JWT	*JSON Web Token*, JSON veebitunnus
Komposiitmikroteenus	Mikroteenus, millesse on koondatud teiste teenuste kutsumise loogika
LDAP	*Lightweight Directory Access Protocol*, kataloogipöörduse kergprotokoll
OAuth	*Open Authorization*, Standardne autoriseerimisprotokoll, mille abil saab anda kolmandale osapoolele piiratud ligipääsu
REST	*Representational State Transfer*, veebiteenuste tarkvara arhitektuuri liidese laad
SAML	*Security Assertion Markup Language*, Standard autentimise ja autoriseerimise andmete vahetamiseks osapoolte vahel
XML	*Extensible Markup Language*, Andmete struktureerimiseks mõeldud märgistuskeel
YAML	*YAML Ain't Markup Language*, inimsõbralik andmete jadastamise keel, tavaliselt konfiguratsioonifailide kirjutamiseks

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Viimastel aastatel on mikroteenused muutunud aina populaarsemaks. Mikroteenuste arhitektuuris koosneb tarkvara väikestest sõltumatutest teenustest, millel on üks kindel funktsionaalsus. Sellisel arhitektuuril on mitmeid eeliseid monoliitrakenduse ees, kuid toob endaga kaasa ka mitmeid murekohti [1]. Eraldiseisvate rakenduste liideseid on võrreldes monoliitrakenduse liidestega raskem kajastada ja samal ajal on neile keerulisem rakendada ühist loogikat. See tähendaks, et ka teistel arendusmeeskondadel oleks keerulisem kasutusele võtta olemasolevaid teenuseid. Sellise probleemi vältimiseks oleks mõistlik kasutusele võtta rakendusliideste haldusplatvorm, millel lisaks eelnimetatule on veel mitmeid erinevaid funktsioone, mis mikroteenuste arhitektuuris võivad vajalikuks osutuda. Parima platvormi valik võib olla aeganõudev ja keeruline arvestades, et turul on mitmeid teenuseid erinevas hinnaklassis.

Töö eesmärgiks on leida, millist funktsionaalsust pakuvad rakendusliideste haldusplatvormid, milliseid eripärasid esineb mikroteenuste arhitektuurile platvormi valides, milliseid arendusmustreid haldusplatvormi kasutusele võtmisel kasutatakse ning analüüsi ja võrdluste kaudu leida sobivaim rakendusliideste haldusplatvorm turul levinud lahenduste seast.

Käesolev lõputöö on kasuks mikroteenuste arhitektuuril põhinevate rakenduste arendajatele, kes soovivad teada, milliseid funktsionaalsusi hõlmavad endas rakendusliideste haldamisplatvormid ja on valimas mikroteenustele vastavat platvormi.

Lõputöö tulemusena antakse ülevaade mikroteenuste arhitektuuris kasutatavatest disainimustritest koos vastavate joonistega, rõhutatakse võimalike lähenemiste probleeme ning pakutakse sellistele lähenemistele alternatiive. Võrreldakse rakendusliideste haldamise lahendusi, mille seast leitakse parim platvorm.

Lõputöö esimeses osas antakse põhjalik ülevaade rakendusliideste halduse olemusest, selle funktsoonidest ning mikroteenuste arendamisel kasutatavatest arendusmustritest.

Töö teises osas uuritakse lähemalt vaatluse alla võetud rakendusliideste haldusplatvorme.

Töö kolmandas osas valitakse välja, võetakse vaatluse alla ning antakse ülevaade neljast erinevast rakendusliideste haldusplatvormist.

Töö neljandas osas rakendatakse analüütiliste hierarhiate protsessi, et välja selgitada parim rakendusliideste haldusplatvorm vaatluse alla võetud lahenduste seast.

2 Rakendusliideste haldus ja lüüs

Käesolevas peatükis analüüsitakse rakendusliideste halduse komponente ning nende funktsioone, millele põhinedes selgitatakse välja parima rakendusliideste haldusplatvorm neljanda peatükis.

Rakendusliideste halduse eesmärgiks on aidata organisatsioonil esitada rakendusliideseid nii sisestele kui välistele tarkvaraarendajatele, pakkudes põhifunktsionaalsusi, mille abil saab tagada edukat rakendusliidese programmi, kaasates arendajaid, analüütikat, turvalisuse meetmeid ja ärilist ülevaadet [2]. Rakendusliideste haldusplatvorm aitab äritegevusel kiirendada oma väljaulatuvust digitaalsetel kanalitel ja edendada partnerite omaksvõttu [2]. See hõlmab endas liideste elutsüklihaldust, juurdepääsu andmist, seiret, turvalisust, arendajaportaali jm. Arenduses kasutatakse selliseid tööriistasid eesmärgiga luua ja kasutusele võtta standardiseeritud rakendusliideseid.

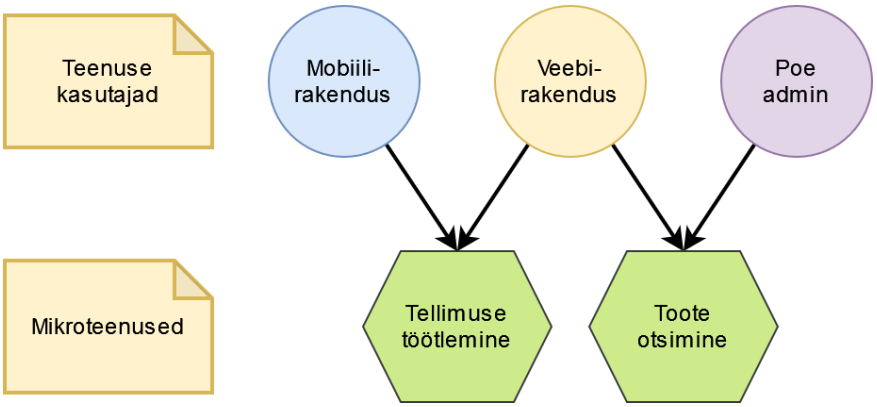
Rakendusliidese lüüs on kõige tähtsam osa haldusplatvormist [2]. Enamik haldusplatvormide pakkujad nimetavad kogu enda lahendust rakendusliidese lüüsiks, mis hõlmab endas ka osaliselt haldamise funktsioone [3]. Lüüsi põhiliseks ülesandeks on suunata päringud edasi teistele teenustele ja teha mikroteenused ühest kohast kättesaadavaks [4].

2.1 Rakendusliidese lüüs

Üheks heaks tavaks suurte ja keeruliste mikroteenuste süsteemi juures on kasutada rakendusliidese lüüsi mustrit (ingl *API gateway pattern*), kus kõik päringud läbivad ühte algpunkti. Sellist mustrit kasutades saab loobuda klient-mikroteenus suhtlusest, mille suurimateks muredeks on [5]:

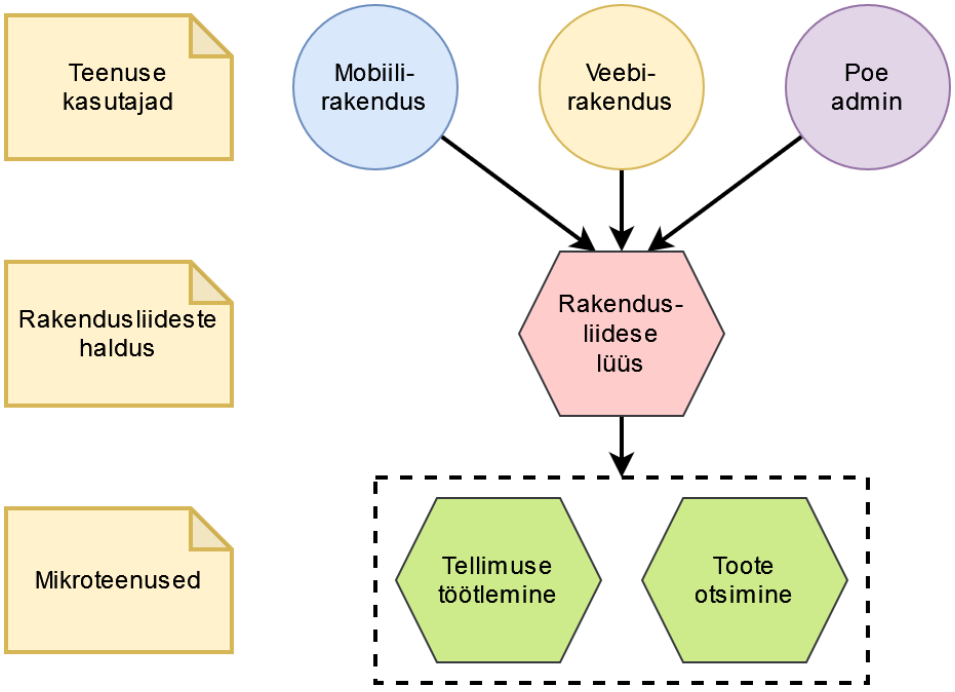
- ühtselt struktureerimata liidesed ja protokollid;
- mikroteenuste lõppsõlmede või loogika muutmine;
- klientrakenduse päringute arv ja keerukus;

Klient-mikroteenus suhtlusarhitektuur on mõistlik ainult juhul, kui tegu on väiksese mikroteenusega, mis ei teosta mitmeid eraldi päringuid. Selline lähenemine omakorda pikendaks latentsusaega ja lisaks keerukust kliendi poolele [4]. Klient-mikroteenus suhtlusarhitektuuri lähenemist esitab joonis 1.



Joonis 1. Klient-mikroteenus suhtlusarhitektuur.

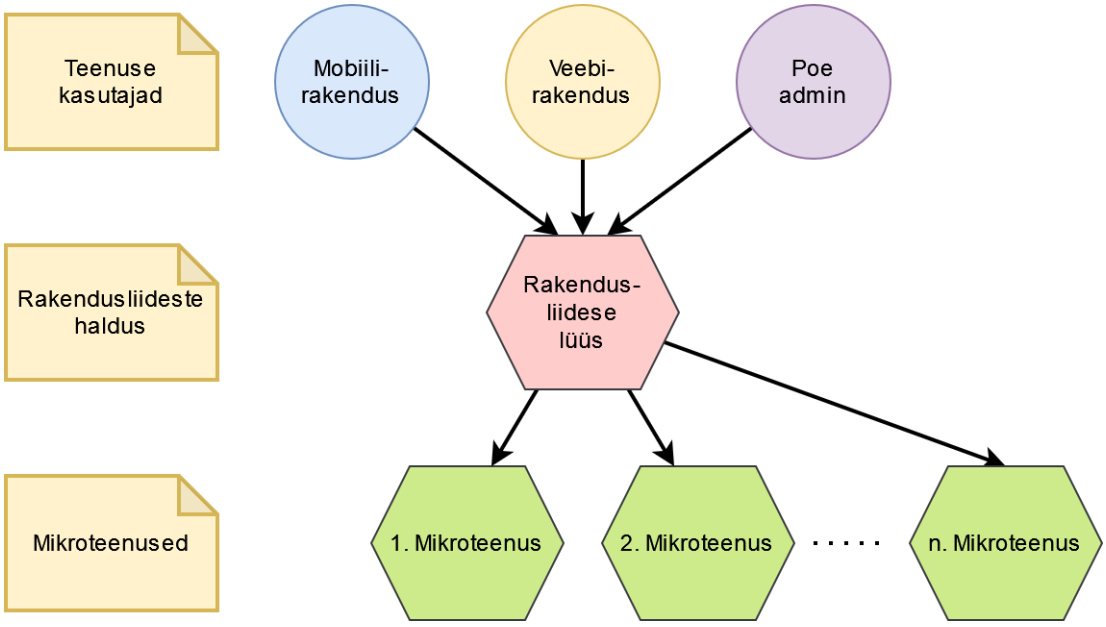
Rakendusliidese lüüsi mustri põhimõte on tekitada üks kindel alguspunkt teenustega suhtlemiseks, mis päringud nendeni edasi suunaks. Sellist mustrit kasutades ei pea klientrakendus teadma, kus erinevad mikroteenused paiknevad ja laseb olemasolevaid teenuseid veel omakorda muuta või tükeldada ilma klientrakendust muutmata, kasutades lüüsi abstraktsiooni kihina [6]. Rakendusliidese lüüsi mustrit esitab joonis 2.



Joonis 2. Rakendusliidese lüüsi muster.

Enamik rakendusliideste lüüse on mikroteenuste arhitektuuri tulekuga hakanud toetama mikro-lüüsi võimalust, kus lüüs töötab konteineri põhisel käitlusajal (ingl *runtime*). See võimaldab võtta kasutusele mitu üksteisest sõltumatut lüüsi, mida hallatakse ühe tsentraalse juhtpaneeli abil [3]. Üks monoliit lüüs toob endaga kaasa üksikrikke (ingl *single point of failure*) võimaluse [7]. Sellise rikke tekkimisel ei ole enam klientrakendusel võimalik saada ühendust mikroteenustega.

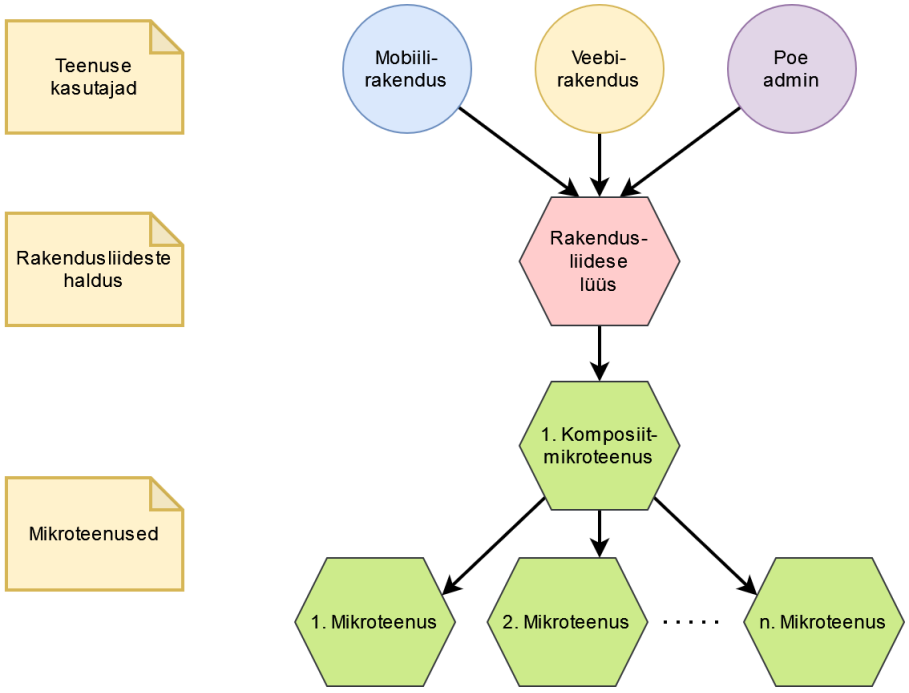
Mitmed rakendusliideste lüüsi tööriistad pakuvad teenuste koondamise (ingl *aggregate*) võimalust, mille käigus tehakse päringud mitmete mikroteenuste poole ja saadud vastused ühendatakse kokku lüüsi tasemel (Joonis 3). Selline lähenemine on tuntud ka kui rakendusliideste orkestreerimine (ingl *orchestration*). Enamik kommertslahendustest propageerivad andmete teisenduse ja koondamise funktsionaalsust, üritades sellega eristuda konkurentidest [8]. Tungivalt soovituslik on selliseid lähenemise vältida, sest äriloogikat vahevaras (ingl *middleware*) on raske kasutusele võtta, testida ja hallata [8], [9]. Selline lähenemine on levinud antimuster, millel esineb selgelt negatiivseid tagajärgi. Tuntud voogedastusplatvorm Netflix kasutas sellist mustrit enda algses lähenemises [3].



Joonis 3. Rakendusliideste orkestreerimine.

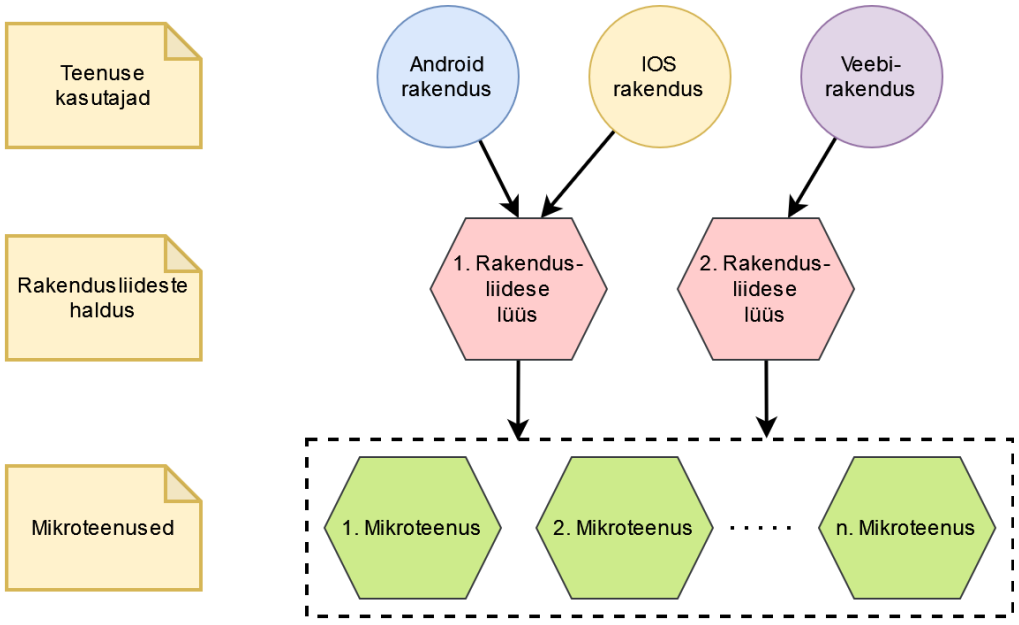
Päringute vastuste koondamise asemel on parem luua komposiitmikroteenus, mis suhtleb mitmete teiste teenustega [3], [8], [9] ja esitab enda rakendusliidest haldusplatvormis.

Komposiitmikroteenuse abil välditakse äriloogika kirjutamist rakendusliidese lüüsi tasemele. Sellist lähenemist esitab joonis 4.



Joonis 4. Komposiitteenuse muster.

Rakendusliideste lüüsi mustri üks variatsioone on tagarakend esirakendile muster (ingl *backend for frontend*), kus iga kasutajaliidese vormiteguri kohta on eraldi rakendusliidese lüüs. Tagarakend esirakendile mustrit esitab joonis 5.



Joonis 5. Tagarakend esirakendile muster.

Lüüsist võib saada aja jooksul raskesti hallatav monoliit kui sellele lisandub palju uusi klientrakendusi või uusi funktsioone [4]. Selle vältimiseks oleks mõistlik kasutusele võtta mitu rakendusliidese lüüsi, kus iga eraldiseisev lüüs käsitleb teatud klientrakenduse vaadet. Mobiilrakenduse kasutajaliides esitab mõningatel juhtudel veebirakendusega võrreldes teistsugust vaadet, kus ekraani suuruse ja võimekuse tõttu on vaja kajastada vähem või keskkonnale vastavat informatsiooni.

2.2 Autoriseerimine ja autentimine

Mikroteenuste arhitektuuri üheks suurimaks väljakutseks on turvalisus [10]. Nimelt toimub monoliitrakenduse puhul autentimine ja autoriseerimine ühes kohas ja suhtlus erinevate süsteemi osadega käib rakendusesiseselt. Mikroteenuste kasutamisel olulisel kohal võimalus edastada kasutaja identiteeti ja õiguseid ühelt teenuselt teisele. See toob endaga kaasa ka ristlõikuva funktsionaalsuse mure (ingl *cross-cutting concern*), kus iga teenus peaks duplitseerima autoriseerimiseks ja autentimiseks vajaminevat koodi osa enda koodibaasi. Juhul kui iga teenus kordab autentimise või autoriseerimise loogikat, on loogika muudatuste korral vaja uuendada kõiki mikroteenused, mis võib vajada üheaegset uuendamist ja nõuab palju aega.

Rakendusliideste haldusplatvormid pakuvad enda lahendustes erinevaid standardiseeritud autoriseerimise protokolle. Nendeks on näiteks OAuth (Open Authorization), LDAP (*Lightweight Directory Access Protocol*), SAML (*Security Assertion Markup Language*). Käesolevas töös ei käsitleta autoriseerimise protokolle, sest kõne all olevad standardid on laialdaselt kaetud ja dokumenteeritud.

Igal lähenemisel ja tehnoloogial on omad positiivsed ja negatiivsed küljed ning valiku peab tegema vastavalt süsteemi vajadustele, võttes seejuures arvesse, kuidas lahendus sobib mikroteenuste arhitektuuri.

Olukorras, kus autoriseerimiseks ja autentimiseks kasutatakse näiteks rakendusliidese võtmeid (ingl *application programming interface key*), mida mikroteenus iseseisvalt ei saa valideerida, peab võtma arvesse, et võtme valideerimise ja väljastaja teenus peab teenindama väga palju päringuid. Sellises olukorras pöörduvad mikroteenused võtme valideerimise jaoks autoriseerimise teenuse poole, mis võib teenuse üle koormata suure päringute hulgaga, millega kaasneb ka ajalise kulu. Lisaks sellele peaks autoriseerimise

ja autentimise teenus olema skaleeritav, et süsteemis ei ilmneks üksikrikke võimalust, kus kõik süsteemi funktsionaalsused ei ole kättesaadavad ühe rikke tõttu, ning suudaks teenindada kasutajad ka juhul, kui päringute hulk kasvab.

Avaliku võtme krüptograafial põhineva JWT (*JSON Web Token*) põhiliseks eeliseks on võimalus *token*it valideerida mikroteenuse tasandil. Sellega vähendatakse päringute hulka autoriseerimise teenuse poole, sest *token* sisaldab endas juba kasutaja andmeid.

Erinevad rakendusliideste lüüsid toetavad erineval moel autentimist ja autoriseerimist. Osad rakendusliideste haldusplatvormid kasutavad kohandatud lahenduse lisamiseks pistikprogramme või võimaldavad kirjutada vajalikku loogikat mõnes programmeerimise keeles. Samas kui mikroteenuste süsteemis on kasutusel teenuste võrk, siis peaks kaaluma ka sealseid pakutavaid lähenemisi või nende kombinatsioone. Teenuste võrgu autentimise ja autoriseerimise meetodeid käesolevas töös ei käsitleta.

2.3 Arendajaportaal

Üheks rakendusliideste haldusplatvormide osaks on arendajaportaal, kuhu on koondatud erinevad mikroteenused, rakendusliidesed ja vastav dokumentatsioon. Selline portaal aitab arendajatel leida, suhelda ja kasutusele võtta teenuseid. See leht on ka kasulik ettevõtte arendajatele, kes soovivad suhelda teiste teenustega, mis ei ole nende arendusmeeskonna vastutusalas.

Arendaja peaks saama sellel lehel teiste arendajatega vestelda, jätta tagasisidet rakendusliidestele ja registreerida ennast kasutajaks ning saama vajalikud võtmed, et teenustega suhelda [2].

2.4 Rakendusliideste elutsüklihaldus

Rakendusliides on süsteemi fassaadiks, mistõttu korralikult välja töötatud, läbi mõeldud ja hallatud rakendusliidesed on oluliseks vahendiks ettevõtte kasvuks.

Rakendusliideste elutsükli haldamine on üheks osaks rakendusliideste haldusplatvormist, mille abil saab kavandada, arendada, testida, kasutusele võtta ja ammendada (ingl *deprecate*) rakendusliideseid [11]. Selle funktsionaalsuse alla peaks kuuluma ka kättesaadavuse ja muudatustest teavitamine emaili või muude suhtluskanalite kaudu.

Lisaks peaksid kasutajad saama teatada rakendusliideste vigadest ja vajadusel saama abi liideste kasutusele võtmise osas [2].

Väljastatud rakendusliidesed kajastatakse arendajaportaalis ning tehakse kättesaadavaks ka rakendusliidese lüüsile [3].

2.5 Seire

Rakendusliideste haldusplatvormid pakuvad ka seire funktsionaalsust. Kõik sissetulevad päringud ja tagastatavad vastused kirjutatakse logidesse koos ajamärgisega. See võimaldab äritegevusel jälgida, millist funktsionaalsust enim kasutatakse ning selle põhjal teha ka tulevasi otsuseid süsteemi täiendamiseks. Arendajatele on see kasulik tööriist, mis võimaldab saada informatsiooni süsteemis toimunud vigade ja päringute kohta.

Vahetevahel sõlmivad ettevõtted enda partneritega teenustasemeleppe, mille raames lepitakse kokku teenuse vastamise ajas, läbilaske võimekuses, kättesaadavuses jne [2]. Rakendusliideste haldusplatvorm võimaldab selliseid aruandeid koostada ja esitada ning varakult märgata teenuste ebatavalist jõudlust. Kui lepet rikutakse, võib teine osapool nõuda hüvitist ebakvaliteetse teenuse eest.

2.6 Vastupidavus

Mikroteenuste arhitektuuris käib erinevate teenuste vahel tihe suhtlus. Juhul kui ühel teenusel peaks tekkima rike või vastamise aeg oluliselt pikenema, võib selle mõjuala laieneda kogu süsteemile ning viia süsteemi töö katkemiseni [12]. Selleks, et sellised olukordasid vältida, kasutatakse mitmeid vastupidavuse meetmeid.

2.6.1 Koormuse tasakaalustamine

Kui kasutusel olevat mikroteenuste süsteemi tabab suur päringute hulk, siis luuakse juurde teenuste koopiaid, mis suudaks suurenenud koormust teenindada. Selleks, et kõik päringud ei jääks ühe teenuse töödelda, küsib lüüs teenuste registrist olemasolevate teenuste asukohtasid ning rakendab koormuse tasakaalustamise algoritmi, millega valitakse välja teenus, kuhu päring saadetakse [13].

2.6.2 Päringute piiramine

Rakendusliidese tasemel on võimalik seadistada, mitu päringut on kliendil lubatud süsteemi vastu teatud aja jooksul teha. Selline lähenemine aitab piirata süsteemi ülekoormamist ja on üheks abivahendiks, et ära hoida teenusest keeldumise rünnakuid (ingl *denial of service*). Sellist piirangut on võimalik teha nii kliendi-, teatud lõppsõlme- või kogu süsteemipõhiselt [14].

Kui lubatud päringute arv ületatakse, siis on võimalik kas edastada veateade kohe või selle asemel päring töödelda ära hilisemal hetkel, kui piirangute piiri enam ei ületata [14].

2.6.3 Kaitselüliti

Kuna mikroteenuste suhtlus käib üle võrgu, mitte mälusiseselt nagu monoliitrakendusel, võib esineda olukordi, kus teenus ei ole kättesaadav või vastab päringutele aeglaselt, mis võib mõjutada ka teisi teenused.

Selleks, et kasutuskõlbmatule teenusele ei saadetaks uusi päringuid, kasutatakse kaitselüliti mustrit, mille põhimõtteks on jälgida ebaõnnestunud või kaua aega võtnud päringute hulka. Kui ebaõnnestunud päringute hulk ületab teatud lävendi, siis järgnevatele päringutele edastatakse koheselt veaerind, et vältida mõjutatud teenuse ülekoormamist. Teatud aja möödudes seatakse kaitselüliti poollahtisesse olekusse, eeldades, et teenus on valmis vastu võtma uusi päringuid. Juhul kui järgnev päring õnnestub, seatakse kaitselüliti kinnisesse olekusse. Vastasel juhul seatakse lüliti taas lahtisesse olekusse ja proovitakse uuesti teatud aja möödudes [15].

Muudatused kaitselülitis peaks olema alati jäädvustatud logides ja reaalajas jälgitavad seire tarkvaras. Sagedased kaitselüliti oleku muutused viitavad tõsisele veale [15].

2.6.4 Puhverdamine

Rakendusliideste lüüs võimaldab salvestada tagarakendilt saadud vastuse või päringu töötlemiseks vajaliku informatsiooni vahemällu, mida tagastatakse korduva päringu puhul koheselt, ilma pöördumiseta tagarakendi poole. Puhvermälu kasutades vähendatakse koormust mikroteenustele ning tänu sellele vastatakse päringule kiiremini [2].

Puhvris olevate andmetele määratakse unikaalne võti, millega järgneval samasugusel päringul leitakse varem salvestatud vaste juhul, kui vahemälus olevad andmed ei ole veel kehtivuse kaotanud. Puhvris on mõtet hoida andmeid, mis ei muutu tihti ja aegumise peaks määrama vastavalt andmete uuendamise sagedusele või äriloogikast tulenevale loogikale. Puhvris võiks hoida näiteks toodete infot või kaupluste nimekirja [2].

2.7 Monetiseerimine

Osade ettevõtete tuluallikaks on enda teenuste pakkumine läbi maksustatud rakendusliideste, kus tasutakse vastavalt eelnevalt kokku lepitud tingimustele. Seire funktsionaalsuse abil on võimalik jälgida, milliseid rakendusliidesed on enim kasutatud ja monetiseerida vastavad lõppsõlmed.

Erinevateks monetiseerimise mudeliteks on [2]:

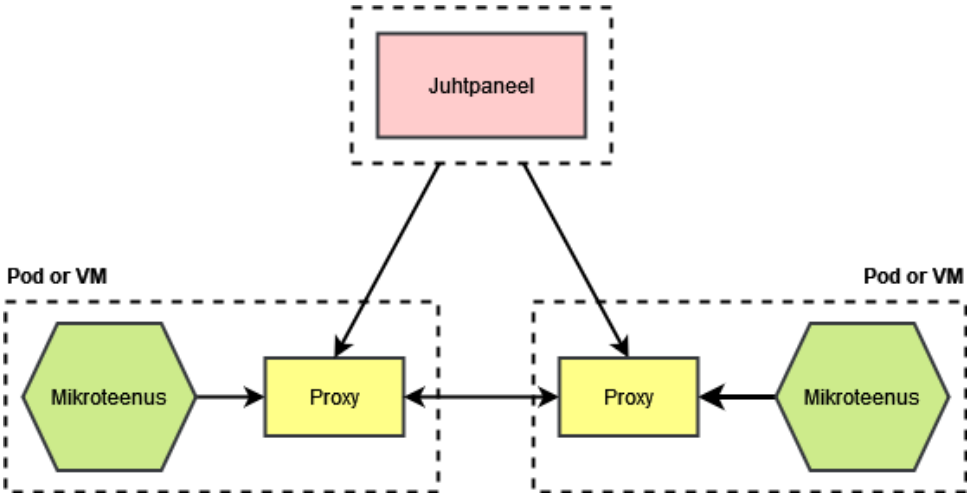
- Tasuta mudel – Ilma tasudeta rakendusliidese kasutamine. Enamasti kasutusel enda toote reklaamimiseks. Näiteks: tootekataloog, poodide asukohad, jne.

- Tasupõhine mudel – Maksmine toimub iga tehingu pealt või vastavalt andmetele, mida küsitakse. Põhiliselt kasutusel teenustes, millel on kõrge väärtus. Näiteks: makse töötlemine, krediidivõime kontroll või analüütiliste andmete küsimine.
- Tulude jagamise mudel – Ettevõte jagab enda teenitud tulu kasutatud rakendusliidese pakkujaga. Näiteks: reklaamide näitamise eest enda lehel jagatakse saadud tulu reklaamfirma ja reklaami pinna pakkuja vahel.
- Kaudne mudel – Mõlemapoolne kasu rakendusliidese pakkuja ja kasutaja vahel. Näiteks: Facebook pakub lihtsat sisselogimise võimalust enda teenustega, millega saab kergesti laiendada tarbijaskonda.

2.8 Sarnasus teenuste võrguga

Rakendusliideste haldamisplatvormides on sarnaseid omadusi teenuste võrguga (ingl *service mesh*), mille tõttu aetakse neid vaatamata nende arhitektuurilistele ja funktsionaalsetele erinevustele aeg-ajalt segamini [3]. Mikroteenuste arhitektuuris on teenuste võrgu abil kergem hallata ja rakendada teenustele ühist loogikat.

Teenuste võrku kasutatakse mikroteenuste omavaheliseks suhtluseks, rakendades igale teenusele kaasa üks vahendaja (ingl *proxy*) (Joonis 6). Vahendajad on ühendatud juhttasandiga, mille kaudu saab rakendada kaitselüliti mustrit, muid teenuse stabiilsuse võtteid ja teha päringute ning teenuste seiret [3].



Joonis 6. teenuste võrk.

Vaatamata sellele, et rakendusliideste haldus- või lüüsilahendustes on sarnane funktsionaalsus olemas, võib jätta mitteärilised tegevused teenuste võrgu kanda, sest see on disainitud käsitlema teenuste omavahelist suhtlust [3].

3 Rakendusliideste haldusplatvormid

Selles peatükis valitakse välja ning antakse ülevaade rakendusliideste haldusplatvormidest, mille seast valitakse parim lahendus. Võrdluse alla võetakse neli erinvat platvormi. Esimeseks platvormiks valis autor Kubernetese poolt pakutava Ambassador Edge Stack rakendusliideste haldusplatvormi, sest see on kasutusel ettevõttes, kus autor töötab. Teiseks platvormiks valiti Google'i väljatöötatud Apigee rakendusliideste haldusplatvorm, mida on mainitud mitmetes kirjalikes allikates [2], [3]. Kolmandaks platvormiks on avatud lähtekoodiga WSO2, mis on Forrester Reaserch Inc. 2020. aasta raporti kohaselt rakendusliideste haldamise lahenduste liidrikohal [16]. Neljandaks platvormiks valiti Tyk rakendusliideste platvorm, millel on autori arvates kergesti jälgitav dokumentatsioon ja kaasaegne kasutajaliides. Autor ei võtnud vaatluse alla osade pilvandmetöötlusteenuste pakkujate rakendusliideste haldamise lahendusi, sest nende pakutavat lahendust on mõistlikum kasutada juhul, kui kogu tarkvara lahendus töötab nende pakutavas pilveteenuse keskkonnas [17].

3.1 Esimese haldusplatvormi ülevaade

Ambassador on Kubernetese-põhine rakendusliidese platvorm, mis sisemuses kasutab avatud lähtekoodiga Envoy vastajat, mida juhitakse ja seadistatakse Ambassadori kaudu. Seadistust hoitakse Kuberneteses ning ei sõltu andmebaasist. Iga uus teenuse eksemplar töötab teistest eraldatuna ning toetub Kubernetesele, mis koordineerib erinevate eksemplaride seadistust.

Platvorm toetab ka gRPC (*google Remote Procedure Call*) ja WebSockets suhtlusprotokolle.

Platvorm toetab erinevaid standardiseeritud autoriseerimise protokolle ning pistikprogrammide ja eraldiseisva autoriseerimise teenuse kasutamist. Kasutaja võtme puhverdamiseks kasutatakse Redise mälusisest hoidlat, millele on ligipääs kõigil Ambassadori eksemplaridel.

Ambassador ei paku enda lahenduses monetiseerimise võimalust ning seire jaoks on vaja eraldi seadistada vastav süsteem, näiteks Prometheus, mille kohta on olemas vastav dokumentatsioon. Lisaks on võimalik seadistada hajutatud jälgimist (ingl *distributed tracing*), millega kogutakse infot kogu päringu teekonna kohta.

Ambassador on kasutusele võtmise ja testimise lihtsustamiseks loonud lehe, kus saab valida endale sobiva seadistusega YAML (*YAML Ain't Markup Language*) faili, mis loob vastava seadistusega teenused.

Uute lõppsõlmede reeglite lisamine käib YAML failide abil. Eeldatakse, et arendaja lisab vastava faili versioonihalduskeskkonda, kust hiljem seadistus rakendatakse kasutusele võtmise protsessis.

Ambassador ei paku enda lahenduses rakendusliideste elutsüklihaldust ega väliste arendajate portaali. Arendajaportaal on mõeldud eeskätt ettevõttesiseseks kasutamiseks ning võimaldab lisada teenusega seotud seire, intsidentide, vigade ja arendusmeeskonnaga seonduvat informatsiooni.

3.2 Teise haldusplatvormi ülevaade

Apigee rakendusliidese haldusplatvorm pakub enda lahenduses nii rakendusliidese analüütikat, arendajaportaali, rakendusliidese versioonihaldust kui ka monetiseerimise võimalust. See platvorm toetab gRPC ja WebSockets suhtlusprotokolle.

Apigee platvormil saab lisada uue rakendusliideste seadistuse, kasutades REST (*Representational State Transfer*) päringuid või juhtpaneeli kasutajaliidest. Lisaks sellele saab kasutada XML (*Extensible Markup Language*) faile, mille seadistuse saab lisada kasutusele võtmise protsessis.

Apigee arendajaportaal on väga kohandatav ning võimaldab arendajatel luua enda konto ning saada rakendusliideste kasutamiseks vajalikud võtmed. Apigee kasutab WorldPay maksete töötlemise teenust ning võimaldab liidestust mõne teise maksete töötlemise teenusega. Süsteem võimaldab koostada rakendusliideste kasutamise raporteid.

Apigee kasutab kasutajate võtmete ja seadistuse hoiustamiseks Apache Cassandra hajutatud mitterelatsioonilist andmebaasi. Apache Zookeeperit kasutatakse seadistuse ja

erinevate komponentide asukoha talletamiseks ja haldamiseks. Analüütikat hoiustatakse relatsioonilises andmebaasis PostgreSQL, mille vahekihiks on Qpid sõnumside süsteem.

3.3 Kolmanda haldusplatvormi ülevaade

WSO2 rakendusliideste haldusplatvorm kasutab sisemuses avatud lähtekoodiga Envoy vastajat. Rakendusliideste seadistamine toimub läbi kasutajaliidese või YAML failidega. Eeldatakse, et YAML failid lisatakse koodibaasi ning hiljem käivitatakse kasutusele võtmise protsessis.

WSO2 platvormi on sisse ehitatud mitmeid autoriseerimise standardeid ning see võimaldab kasutada ka kolmandate osapoolete identiteedi pakkujaid. Platvormile sisseehitatud identiteediserver kasutab H2 relatsioonilist andmebaasi, mis dokumentatsiooni kohaselt soovitatakse asendada mõne parema jõudluse ja töökindlusega relatsioonilise andmebaasiga, näiteks PostgreSQL või Microsoft SQL.

Platvormile on võimalus lisada monetiseerimise võimekus, kasutades Stripe maksete töötlemise pistikprogrammi. Soovi korral on võimalik kasutusele võtta ka mõni teine maksete töötlemise teenus kasutades nende monetiseerimise liidest.

Platvormile on sisse ehitatud Reactil põhinev ning väga kohandatav arendajaportaal koos rakendusliideste elutsükli haldamisega.

3.4 Neljanda haldusplatvormi ülevaade

Tyk rakendusliidese haldusplatvorm pakub nii rakendusliidese analüütikat, arendajaportaali kui ka haldamise juhtpaneeli. Platvorm toetab ka gRPC, GraphQL ning WebSockets suhtlusprotokolle.

Tyk rakendusliidese lüüs kasutab kasutajate võtmete puhverdamiseks Redise mälusisest hoidlat ning talletab seal ka ajutiselt analüütika andmeid, mis hiljem saadetakse edasi MongoDB andmebaasi. Seadistuses on võimalik välja vahetada MongoDB andmebaas mõne kasutusel oleva vaatlusplavormiga, näiteks Logz.io või Datadog.

Tyk platvormi on võimalik seadistada, kasutades JSON (*JavaScript Object Notation*) faile, mis sisaldavad rakendusliidese definitsioone ning seadistust. Seadistuse saamiseks

ja lisamiseks on võimalik saata vastava sisuga päring juhtpaneeli. Seadistamiseks on JSON failide asemel võimalik kasutada ka YAML faile.

Tyk süsteemi on sisse ehitatud mitmeid autoriseerimise standardeid ning on võimalik lisada ka enda lahendus. Lisaks on võimalik platvorm liidestada makse töötlemise teenusega, et maksustatud lõppsõlmede eest tasusuid koguda.

4 Parima rakendusliideste haldusplatvormi valimine kasutades analüütiliste hierarhiate protsessi

Selles peatükis kasutab autor väljavalitud platvormide võrdlemiseks analüütiliste hierarhiate meetodit. Seda meetodit kasutatakse põhiliselt subjektiivsete hinnangute alusel tegutsevate süsteemide korrastamiseks [18] ning vabastab otsustaja absoluutsete hinnangute andmisest, kasutades paari kaupa võrdlemist, mis on inimeste hindamisvõimeid arvestades vastuvõetavam [19]. Autor valis analüütiliste hierarhiate protsessi rakendusliideste omavaheliseks võrdluseks, sest meetodi abil vaadeldakse võimalikke alternatiive mitmete kriteeriumite põhjal. Kriteeriumite alusel on võimalik alternatiivid paremuse järjekorda seada ning nende seast parim valik teha.

Analüütiliste hierarhiate protsessi otsustusmudel koosneb eesmärgist, põhikriteeriumitest, alamkriteeriumitest ja otsustusalternatiividest järjestikustel eritasemetel [18]. Käesoleva töö eesmärgiks on leida parim rakendusliideste haldusplatvorm. Autor leidis eesmärgi saavutamiseks seitse põhikriteeriumit ja neliteist alamkriteeriumit (Tabel 1). Kriteeriumid põhinevad eelnevalt uuritud rakendusliideste halduses leiduvatel funktsionaalsustel. Alternatiivideks on autori väljavalitud rakendusliideste haldusplatvormid, mille kohta anti ülevaade eelmises peatükis.

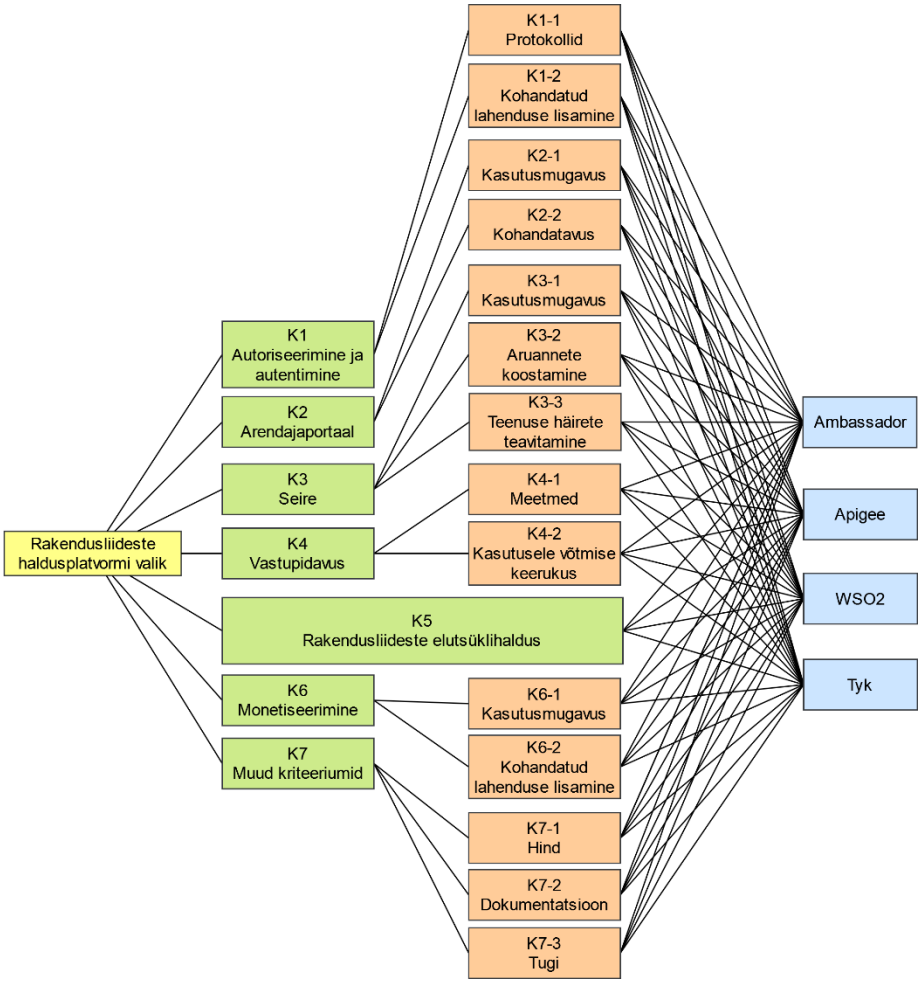
Tabel 1. Otsustusmudeli põhi- ja alamkriteeriumid.

Põhikriteeriumid	**Alamkriteeriumid**
Autoriseerimine ja autentimine	Protokollid
	Kohandatud lahenduse lisamine
Arendajaportaal	Kasutusmugavus
	Kohandatavus
Seire	Kasutusmugavus
	Aruannete koostamine
	Teenuse häirete teavitamine
Vastupidavus	Meetmed
	Kasutusele võtmise keerukus

Põhikriteeriumid	**Alamkriteeriumid**
Rakendusliideste elutsüklihaldus	
Monetiseerimine	Kasutusmugavus
	Kohandatud lahenduse lisamine
Muud kriteeriumid	Dokumentatsioon
	Tugi

Algselt tahtis autor võtta võrdlusesse ka hinna, kuid enamik valitud ettevõtetest ei avalikusta enda lahenduse hinda ning autor ei esitanud ühtegi hinnapäringud ettevõtetele. Tyk platvormi kolm esimest taset neljast maksavad 1 800 kuni 6 800 dollarit kuus. Piiranguvaba neljanda taseme jaoks on vaja saada hinnapakkumine.

Nelja tasemega analüütiliste hierarhiate otsustustmudelit esitab joonis 6.



Joonis 7. Rakendusliideste haldusplatvormi valiku otsustusmudel.

4.1 Kriteeriumite paari kaupa võrdlemine

Analüütiliste hierarhiate protsessi läbiviimiseks kasutab autor Super Decisions tarkvara, mis on arendatud Thomas Saaty ja Rozann W. Saaty asutatud sihtasutuses Creative Decisions Foundation.

Põhikriteeriumite, alamkriteeriumite ja alternatiivide omavaheliseks võrdlemiseks kasutab autor Saaty fundamentaalskaalat, millel on arvulised väärtused ühest üheksani ning tähistavad sõnalisi hinnanguid. Paaritute arvude tähistust esitab tabel 2. Nendevahelisi väärtusi tähistavad paarisarvud.

Tabel 2. Saaty fundamentaalskaala sõnaline tähistus

Skaala väärtus	**Sõnaline hinnang**
1	Võrdselt tähtis/eelistatud
3	Mõõdukalt tähtsam/eelistatud
5	Tugevalt tähtsam/eelistatud
7	Väga tugevalt tähtsam/eelistatud
9	Ekstreemselt tähtsam/eelisatud

Super Decisions tarkvara arvutab välja hinnangute vasturääkivuse või hinnangute ebakõla (ingl *inconsistency*) määra, mille väärtus võiks jääda alla 0,1 [20].

4.1.1 Põhikriteeriumite kaalude leidmine

Kriteeriumite paari kaupa võrdlemiste tulemusena selgus, et autori hinnangul on kõige olulisemaks kriteeriumiks autoriseerimine ja autentimine. Sellele järgnes vastupidavuse kriteerium ja kolmandaks jäi seire. Tabel 3 esitab põhikriteeriumite omavaheliste võrdluste tulemusi. Hinnangute ebakõla määr on 0,04928.

Tabel 3. Põhikriteeriumite võrdlusmaatriks.

	K1	**K2**	**K3**	**K4**	**K5**	**K6**	**K7**	**Kaal**
K1 Autoriseerimine ja autentimine	1	4	2	2	4	6	5	0,30977
K2 Arendajaportaal	1/4	1	1/3	1/3	2	4	2	0,09336
K3 Seire	1/2	3	1	1/4	4	4	4	0,17006
K4 Vastupidavus	1/2	3	4	1	5	5	4	0,28105
K5 Rakendusliideste elutsüklihaldus	1/4	1/2	1/4	1/5	1	2	2	0,06124
K6 Monetiseerimine	1/7	1/4	1/4	1/5	1/2	1	1	0,03904
K7 Muud kriteeriumid	1/5	1/2	1/4	1/4	1/2	1	1	0,04548

4.1.2 Alamkriteeriumite kaalude leidmine

Autor hindab autoriseerimise ja autentimise puhul olulisemaks kohandatud lahenduse lisamise võimalust (Tabel 4), sest sedasi on võimalik kasutusele võtta erilahendust, mida platvorm ei paku ning võimaldab juba ettevõttes kasutusel olevat lahendust süsteemiga liidestada. Juhul kui võrdluse all on vaid kaks kriteeriumit, ei ole hinnagutevahelise ebakõla tekkimine võimalik, sest sellisel juhul ei saa tekkida olukorda, kus hinnangud räägiksid iseendale vastu – hinnangute ebakõla määraks on 0.

Tabel 4. K1 autoriseerimise ja autentimise alamkriteeriumite võrdlusmaatriks.

	K1-1	**K1-2**	**Kaal**
K1-1 Protokollid	1	1/2	0,33333
K1-2 Kohandatud lahenduse lisamine	2	1	0,66667

Arendajaportaali kasutajamugavust hindas autor võrdselt kuni mõõdukalt tähtsamaks kui kohandatavust (Tabel 5), sest tegu on tähtsa tööriistaga, mille kasutamine võiks olla kasutajale kerge ja meeldiv. Hinnangute ebakõla määraks on 0.

Tabel 5. K2 arendajaportaal alamkriteeriumite võrdlusmaatriks.

	K2-1	**K2-2**	**Kaal**
K2-1 Kasutusmugavus	1	1/2	0,33333
K2-2 Kohandatavus	2	1	0,66667

Seire puhul hindas autor kasutusmugavust ja aruannete koostamist võrdselt tähtsateks, kuid leidis, et teenuse häirete teavitamine on teistest mõõdukalt kuni tugevalt tähtsam, sest intsidendi tekkimisele või teenuse ebatavalisele käitumisele peaks reageerima õigeaegselt. Teenuse kasutajad peaksid olema ka teadlikud süsteemis esinenud riketest. Tabel 6 esitab seire alamkriteeriumite hindamismaatriksit. Hinnangute ebakõla määr on 0,00885.

Tabel 6. K3 seire alamkriteeriumite võrdlusmaatriks.

	K3-1	**K3-2**	**K3-3**	**Kaal**
K3-1 Kasutusmugavus	1	1	1/3	0,19192
K3-2 Aruannete koostamine	1	1	1/4	0,17437
K3-3 Teenuse häirete teavitamine	3	4	1	0,63371

Vastupidavuse alamkriteeriumite puhul hindas autor, et vastupidavuse meetmed on tugevalt tähtsamad kui kasutusele võtmise keerukus (Tabel 7), sest erinevate lahenduste olemasolu ja kasutamine võib olla suurema väärtusega kui kasutusele võtmisele kuluv aeg ja vaev. Hinnangute ebakõla määr on 0.

Tabel 7. K4 vastupidavuse alamkriteeriumite võrdlusmaatriks.

	K4-1	**K4-2**	**Kaal**
K4-1 Meetmed	1	5	0,83333
K4-2 Kasutusele võtmise keerukus	1/5	1	0,16667

Rakendusliideste elutsüklihaldusel (K5) puuduvad alamkriteeriumid, mida omavahel võrrelda. Sellele vaatamata osaleb kriteerium parima platvormi valimise protsessis.

Monetiseerimise alamkriteeriumite võrdlemisel leiab autor, et kohandatud lahenduse lisamine on võrdselt kuni mõõdukalt tähtsam kui kasutusmugavus, sest nii makseteenuse pakkuja valimise võimalus kui ka kasutusmugavus peaks platvormil olema hästi teostatud, kuna vastasel juhul ei ole funktsioon kasulik. Monetiseerimise hinnanguid esitab tabel 8. Hinnangute ebakõla määraks on 0.

Tabel 8. K6 monetiseerimise alamkriteeriumite võrdlusmaatriks.

	K6-1	**K6-2**	**Kaal**
K6-1 Kasutusmugavus	1	2	0,66667
K6-2 Kohandatud lahenduse lisamine	1/2	1	0,33333

Muude kriteeriumite võrdluses, leidis autor, et dokumentatsioon on toega võrreldes tähtsam (Tabel 9), sest hea dokumentatsioon võimaldab vigadeta kasutusele võtta funktsioone ja kiirendada arendusprotsessi. Tugi on oluline konkreetsete probleemide lahendamiseks, kuid dokumentatsioon võiks olla piisavalt detailne, et toe poole pöördumiseks, kust vastuse saamine võib venida ja pikendada arendusprotsessi, poleks vajadust. Hinnangute ebakõla määraks on 0,01759

Tabel 9. K7 muude kriteeriumite alamkriteeriumite võrdlusmaatriks.

	K7-1	**K7-2**	**Kaal**
K7-2 Dokumentatsioon	1	2	0,66667
K7-3 Tugi	1/2	1	0,33333

4.2 Alternatiivide paari kaupa võrdlemine

Alternatiivide hindamisel sai Apigee ja Tyk võrreldes teistega parema tulemuse autoriseerimise protokollide võrdluses (Tabel 10), sest toetavad HMAC (*Hash-based Message Authentication*) krüptograafilist autoriseerimist. Hinnangute ebakõla määr on 0,02271.

Tabel 10. K1-1 Protokollid (autoriseerimine ja autentimine) alternatiivide võrdlusmaatriks.

	Ambassador	**Apigee**	**Tyk**	**WSO2**	**Kaal**
Ambassador	1	1/2	1/2	1	0,17501
Apigee	2	1	1	1	0,28936
Tyk	2	1	1	1	0,28936
WSO2	1	1	1	1	0,24627

Autoriseerimise ja autentimise kohandatud lahenduse lisamises sai viimase koha Apigee, mis võimaldab ainult välise OAuth teenuse kasutamist. Samas teised platvormid

võimaldavad lisada mitmeid filtreid või kirjutada vastavat loogikat, kasutades mõnda programmeerimise keelt. Tabel 11 esitab kohandatud lahenduse lisamise võrdlusmaatriksit. Hinnangute ebakõla määr on 0,04417

Tabel 11. K1-2 kohandatud lahenduse lisamine (autoriseerimine ja autentimine) alternatiivide võrdlusmaatriks.

	Ambassador	**Apigee**	**Tyk**	**WSO2**	**Kaal**
Ambassador	1	3	3	1	0,40147
Apigee	1/3	1	1/2	1/2	0,11971
Tyk	1/3	2	1	1	0,20691
WSO2	1	2	1	1	0,27191

Ambassadori platvormi lahenduses on olemas arendajaportaal ettevõttesiseseks kasutamiseks, kuid ei paku avalikku dokumentatsiooni, mille tõttu sai alternatiividest kõige madalama hinnangu. Sellegi poolest on Ambassadori arendajaportaal kasulik, sest võimaldab määrata teenustele vastutava meeskonna, kontaktisiku, intsidentide hüperlingi ja seire veebilehe. WSO2 paistis silma enda hea struktuuri ning rakendusliideste hindamise funktsiooniga, mispärast sai see kõige kõrgema hinnangu alternatiivide seast. Arendajaportaali kasutusmugavuse hinnanguid esitab tabel 12. Hinnangute ebakõla määraks on 0,05787.

Tabel 12. K2-1 kasutusmugavus (arendajaportaal) alternatiivide võrdlusmaatriks.

	Ambassador	**Apigee**	**Tyk**	**WSO2**	**Kaal**
Ambassador	1	1/4	1/4	1/4	0,07343
Apigee	4	1	1	1/3	0,21359
Tyk	4	1	1	1/3	0,21359
WSO2	4	3	3	1	0,49938

Arendajaportaali kohandatavuse esimest kohta jagavad Apigee ja WSO2 (Tabel 13), milles mõlemas on võimalik ise kirjutada HTML-i (*HyperText Markup Language*) ja JavaScripti koodi. Ambassadori lahenduses ei ole võimalik lehte kohandada, kuid selleks puudub ka vajadus, sest leht on mõeldud ettevõttesiseseks kasutamiseks. Hinnangute ebakõla määraks on 0,02271.

Tabel 13. K2-2 kohandatavus (arendajaportaal) alternatiivide võrdlusmaatriks.

	Ambassador	**Apigee**	**Tyk**	**WSO2**	**Kaal**
Ambassador	1	1/9	1/9	1/9	0,03503
Apigee	9	1	2	1	0,37046
Tyk	9	1/2	1	1/2	0,22405
WSO2	9	1	2	1	0,37046

Seire kasutusmugavuse esimese koha sai Ambassador, mis enda lähenemises soovitab kasutada Prometheuse süsteemimonitori ja Grafana visualiseerimise kasutajaliidest. Tyk platvormi seire lahendus ei paku enda lahenduses kuigi detailset ülevaadet, mille tõttu sai võrdluses viimase koha. Tabel 14 esitab kasutusmugavuse võrdlusmaatriksit. Hinnangute ebakõla määr on 0,00388.

Tabel 14. K3-1 kasutusmugavus (seire) alternatiivide võrdlusmaatriks.

	Ambassador	**Apigee**	**Tyk**	**WSO2**	**Kaal**
Ambassador	1	1	4	1	0,31800
Apigee	1	1	3	1	0,29512
Tyk	1/4	1/3	1	1/3	0,09176
WSO2	1	1	3	1	0,29512

Ambassadori platvormil on võimalik Grafana abil erinevaid aruannete vaateid vastavalt vajadustele koostada, mispärast sai see võrdluses kõrgeima hinnangu. WSO2 ja Apigee võimaldavad koostada samuti erinevaid vaateid, kuid ei ole nii kohandatavad. Viimased kaks koostavad ka monetiseerimisega seotud raporteid. Tabel 15 esitab aruannete koostamise võrdlusmaatriksit. Hinnangute ebakõla määr on 0,02271.

Tabel 15. K3-2 Aruannete koostamine (seire) alternatiivide võrdlusmaatriks.

	Ambassador	**Apigee**	**Tyk**	**WSO2**	**Kaal**
Ambassador	1	2	4	1	0,38582
Apigee	1/2	1	2	1	0,22667
Tyk	1/4	1/2	1	1/2	0,11333
WSO2	1	1	2	1	0,27418

Apigee platvorm jäi viimaseks teenuse häirete teavitamises, sest võimaldab seadistada maksimaalselt vaid 20 erinevat häire teavitust kogu organisatsiooni kohta. WSO2 platvormis on võimalik teavitust lisada otse arendajaportaali, mispärast sai ka kõrgeima hinnangu. Tabel 16 esitab teenuse häirete teavitamise võrdlusmaatriksit. Hinnangute ebakõla määr on 0,0266.

Tabel 16. K3-3 teenuse häirete teavitamine (seire) alternatiivide võrdlusmaatriks.

	Ambassador	**Apigee**	**Tyk**	**WSO2**	**Kaal**
Ambassador	1	3	2	1/2	0,29259
Apigee	1/3	1	1/2	1/3	0,10697
Tyk	1/2	2	1	1/2	0,18495
WSO2	2	3	2	1	0,41549

Vastupidavuse meetmete võrdluses selgus, et kõik platvormid võimaldavad samasuguseid vastupidavuse meetmeid ehk on võrdselt eelistatud (Tabel 17). Hinnangute ebakõla määr on 0.

Tabel 17. K4-1 meetmed (vastupidavus) alternatiivide võrdlusmaatriks.

	Ambassador	**Apigee**	**Tyk**	**WSO2**	**Kaal**
Ambassador	1	1	1	1	0,25
Apigee	1	1	1	1	0,25
Tyk	1	1	1	1	0,25
WSO2	1	1	1	1	0,25

Erinevate vastupidavuse meetmete kasutusele võtmise võrdluses selgus, et parimaks lahenduseks on WSO2 platvorm. Apigee platvormi puhul ei olnud meetmed eraldi välja

toodud vaid koosnesid mitme funktsiooni samaaegsest kasutamisest. Tabel 18 esitab kasutusele võtmise võrdlusmaatriksit. Hinnangute ebakõla määr on 0,02271.

Tabel 18. K4-2 kasutusele võtmise keerukuse (vastupidavus) alternatiivide võrdlusmaatriks.

	Ambassador	**Apigee**	**Tyk**	**WSO2**	**Kaal**
Ambassador	1	2	1	1	0,28085
Apigee	1/2	1	1/2	1/2	0,14042
Tyk	1	2	1	1/2	0,23902
WSO2	1	2	2	1	0,33971

Kõik lahendused, välja arvatud Ambassador, pakuvad rakendusliideste elutsüklihaldust ning neis ei esinenud märgatavaid erinevusi. Tabel 19 esitab rakendusliideste elutsüklihalduse võrdlusmaatriksit. Hinnangute ebakõla määr on 0,02271.

Tabel 19. K5 rakendusliideste elutsüklihalduse alternatiivide võrdlusmaatriks.

	Ambassador	**Apigee**	**Tyk**	**WSO2**	**Kaal**
Ambassador	1	1/9	1/9	1/9	0,03503
Apigee	9	1	1	1	0,31528
Tyk	9	1	1	1/2	0,26832
WSO2	9	1	2	1	0,38136

Monetiseerimise kasutusmugavuse poolest tõusis esile Apigee, mis pakub kõiki monetiseerimise mudeleid peale kaudse. WSO2 ei paku kaudset ega tulu jagamise mudelit. Tyk platvormi dokumentatsioon monetiseerimise kohta on puudulik ning antud lahendus tundub poolik. Autor võttis ühendust ka platvormi esindajaga, kelle sõnul ei ole see funktsioon platvormi poolt veel täielikult toetatud, kuid sellele vaatamata ei takista erilahendusega liidestamist. Ambassador ei paku monetiseerimise funktsiooni, mille tõttu sai kõige madalama hinnangu. Tabel 20 esitab kasutusmugavuse võrdlusmaatriksit. Hinnangute ebakõla määr on 0,06948.

Tabel 20. K6-1 kasutusmugavuse (monetiseerimine) alternatiivide võrdlusmaatriks.

	Ambassador	**Apigee**	**Tyk**	**WSO2**	**Kaal**
Ambassador	1	1/9	1/9	1/9	0,03370
Apigee	9	1	4	2	0,50040
Tyk	9	1/4	1	1/2	0,17946
WSO2	9	1/2	2	1	0,28644

Apigee ja WSO2 platvormid toetavad kohandatud lahenduse lisamist, mispärast on teistega võrreldes eelistatumad. Nagu eelnevalt mainitud, ei toeta Ambassador monetiseerimist ja Tyk platvormi dokumentatsioon on puudulik. Tabel 21 esitab kohandatud lahenduse lisamise võrdlusmaatriksit. Hinnangute ebakõla määr on 0,09334.

Tabel 21. K6-2 Kohandatud lahenduse lisamise (monetiseerimine) alternatiivide võrdlusmaatrik.

	Ambassador	**Apigee**	**Tyk**	**WSO2**	**Kaal**
Ambassador	1	1/9	1/9	1/9	0,03370
Apigee	9	1	4	1	0,40640
Tyk	9	1/4	1	1/4	0,15413
WSO2	9	1	4	1	0,40640

Dokumentatsiooni võrdluses jagasid esimest kohta Ambassador ja WSO2 (Tabel 22). Apigee dokumentatsioon on põhjalik, kuid raskesti jälgitav halva struktuuri tõttu. Tyk dokumentatsioon on kergesti jälgitav ja hea struktuuriga, kuid jääb kohati liiga pealiskaudseks. Hinnangute ebakõla määr on 0,0312.

Tabel 22. K7-1 dokumentatsiooni (muud kriteeriumid) alternatiivide võrdlusmaatriks.

	Ambassador	**Apigee**	**Tyk**	**WSO2**	**Kaal**
Ambassador	1	3	4	1	0,38091
Apigee	1/3	1	3	1/3	0,15921
Tyk	1/4	1/3	1	1/4	0,07898
WSO2	1	3	4	1	0,38091

Ambassador sai toe hindamisel kõrgeima koha, sest on olemas aktiivne kogukonna Slack suhtluskanal, kus süsteemi kasutajaid igapäevaselt aidatakse. WSO2 platvormil on samuti

olemas Slack suhtluskanal, kuid see pole kuigi aktiivne ning vastuse saamine võib võtta kaua aega. Apigee on loonud neljaminutilisi videoklippe arendajatele, mille abil seletatakse ja näidatakse erinevaid süsteemi funktsioone, kuid autor leidis, et videod ei anna piisavalt põhjalikku ülevaadet. Lisaks sellele on olemas foorum, mis täidab enda eesmärki. Tyk platvormil on samuti kogukonna foorum, kust on võimalik leida abi. Tyk pakub ka erineva taseme kliendituge, kuid puudub informatsioon, mida erineva taseme toed endas hõlmavad. Tabel 23 esitab toe võrdlusmaatriksit Hinnangute ebakõla määr on 0,02271.

Tabel 23. K7-2 tugi (muud kriteeriumid) alternatiivide võrdlusmaatriks.

	Ambassador	**Apigee**	**Tyk**	**WSO2**	**Kaal**
Ambassador	1	2	2	2	0,39521
Apigee	1/2	1	1	2	0,23218
Tyk	1/2	1	1	2	0,23218
WSO2	1/2	1/2	1/2	1	0,14042

4.3 Tulemuste analüüs

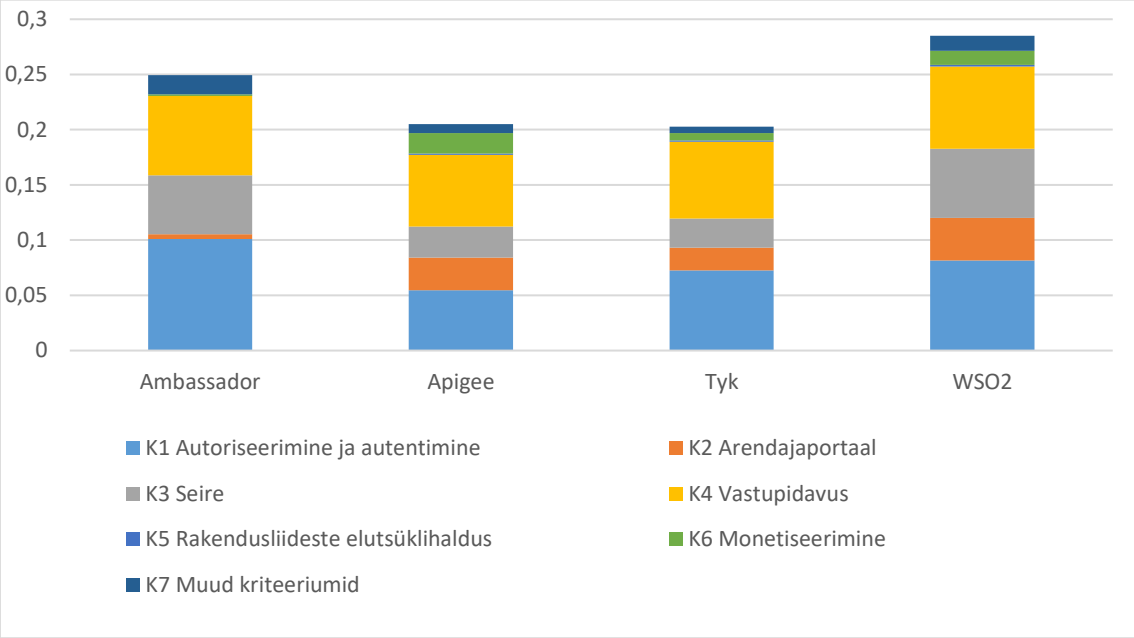
Eelmises jaotises võrreldi omavahel alternatiive paari kaupa alamkriteeriumite suhtes. Selle jooksul anti kokku alternatiividele 90 hinnangut, mille põhjal osutus parimaks WSO2 platvorm. Teisele kohale jäi Ambassador, kolmandale Apigee ja viimaseks Tyk. Tabel 24 esitab analüütiliste hierarhiate protsessi lõpptulemusi.

Tabel 24. Analüütiliste hierarhiate protsessi lõpptulemus.

Rakendusliidese haldusplatvorm	**Kokku**
Ambassador	0,250631
Apigee	0,226598
Tyk	0,222470
WSO2	0,300301

Selleks, et teada saada, kui suured osakaalud said alternatiivid alamkriteeriumite suhtes, on vaja korrutada alamkriteeriumi kaal alternatiivi hinnangu kaaluga ning vastava põhikriteeriumi kaaluga. Vastava põhikriteeriumi alamkriteeriumite osakaalude kokku

liitmisel saadakse alternatiivi põhikriteeriumi osakaal. Joonis 8 esitab analüütiliste hierarhiate protsessi lõpptulemusi graafiliselt.



Joonis 8. Analüütiliste hierarhiate protsessi lõpptulemus.

Ambassadori rakendusliideste haldusplatvorm sai teise koha vaatamata sellele, et puudus arendajaportaal välistele kasutajatele või monetiseerimise funktsioon. WSO2 platvorm sai nii autoriseerimise ja autentimise kui ka muude kriteeriumite (K7) osas halvema tulemuse kui Ambassador, kuid sellegi poolest osutus võitjaks tänu paremale arendajaportaalile ja monetiseerimise funktsioonile.

Lõpptulemustes suurt arvulist erinevust Tyk ja Apigee platvormide vahel ei esinenud. Apigee puhul oli hästi toestatud nii monetiseerimise funktsioon kui ka arendajaportaal. Tyk ei olnud parim ühegi kriteeriumi suhtes.

Analüüsi käigus andis autor hinnangud põhi- ja alamkriteeriumitele ning alternatiividele. Lisas 2 on link otsustusmudeli failile koos viitega Super Decisions dokumentatsioonile, mille abil on võimalik korrigeerida kriteeriumite kaale vastavalt vajadusele, et selgitada parim platvorm lähtuvalt kasutaja hinnangutest ja nõuetest.

5 Kokkuvõte

Mikroteenuste arhitektuuris koosneb tarkvara väikestest sõltumatutest teenustest ning omab eeliseid monoliitrakenduse ees. Samas kaasnevad sellega mitmed murekohad.

Käesolevas bakalaureusetöös käsitleti rakendusliideste haldust ja analüüsiti haldusplatvorme lahendamaks mikroteenuste esitamise ja ühise loogika rakendamisega seotud probleeme.

Lõputöö eesmärgiks oli rakendusliideste halduses olevate funktsioonide ja levinud arendusmustrite esitamine, võttes arvesse mikroteenuste arhitektuuri eripärasid, ning parima rakendusliideste haldusplatvormi leidmine analüütiliste hierarhiate protsessi abil.

Töö tulemusena anti ülevaade erinevatest rakendusliideste halduses leiduvatest funktsioonidest ja mikroteenuste arhitektuuris kasutatavatest arendusmustritest koos joonistega. Lisaks toodi esile erinevate lähenemiste puudujääke ning pakuti nendele alternatiive.

Analüütiliste hierarhiate protsessi läbiviimise tulemusena selgus, et valitud platvormide seast parim on WSO2.

Lõputöö tulemused on abiks tarkvaraarhitektidele rakendusliideste haldusplatvormi valimisel ja tehniliste lahenduste uurimisel.

Käesolevas töös mainiti, kuid ei uuritud teenuste võrgu ja rakendusliideste haldusplatvormi kasutamist. Üheks võimalikuks edasiarenduseks oleks nende süsteemide samaaegse kasutamise uurimine, sest osaliselt nende funktsioonid kattuvad.

Kasutatud kirjandus

[1] A. Monus, „What are microservices? The pros, cons, and how they work,“ Raygun, 09. oktoober 2018. [Võrgumaterjal]. Available: https://raygun.com/blog/what-are-microservices/. [Kasutatud 24. veebruaril 2022].

[2] B. De, API Management An Architect’s Guide to Developing and Managing APIs for Your Organization First Edition, New York: Apress, 2017.

[3] K. Indrasiri ja S. Prabath, Microservices for the Enterprise: Designing, Developing, and Deploying, New York: Apress Media LLC, 2018.

[4] N. Anil, G. Warren ja Y. Victor, „The API gateway pattern versus the Direct client-to-microservice communication,“ Microsoft, 15. september 2021. [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/dotnet/architecture/microservices/architect-microservice-container-applications/direct-client-to-microservice-communication-versus-the-api-gateway-pattern. [Kasutatud 27. veebruaril 2022].

[5] D. Ratnayake, „The Truth About API Management in a Microservice Architecture,“ Medium, 31. jaanuar 2019. [Võrgumaterjal]. Available: https://duckster.medium.com/the-truth-about-api-management-in-a-microservice-architecture-a3b001a713c5. [Kasutatud 4. märtsil 2022].

[6] A. Gaitonde ja M. Malik, „Using API Gateway as a Single Entry Point for Web Applications and API Microservices,“ Amazon, 22. oktoober 2019. [Võrgumaterjal]. Available: https://aws.amazon.com/blogs/architecture/using-api-gateway-as-a-single-entry-point-for-web-applications-and-api-microservices/. [Kasutatud 15. mail 2022].

[7] A. Broshar, „API Gateways: Improving performance, security and management of microservices,“ Koyeb, 27. mai 2021. [Võrgumaterjal]. Available: https://www.koyeb.com/blog/api-gateways-improving-performance-security-and-management-of-microservices. [Kasutatud 15. mail 2022].

[8] K. Sookocheff, „Overambitious API gateways Best practices for deploying API gateways,“ 22. veebruar 2019. [Võrgumaterjal]. Available: https://sookocheff.com/post/api/overambitious-api-gateways/. [Kasutatud 14. märtsil 2022].

[9] B. Shah, „Microservices Design - API Gateway Pattern,“ Medium, 04. juuli 2020. [Võrgumaterjal]. Available: https://blog.devgenius.io/microservices-design-api-gateway-pattern-980e8d02bdd5. [Kasutatud 14. märtsil 2022].

[10] M. G. de Almeida ja E. D. Canedo, „Authentication and Authorization in Microservices,“ *Applied Sciences, vol. 12, no. 6,* p. 3023, 16. märts 2022.

[11] K. Vasudevan, „What is API Lifecycle Management?,“ Swagger, 29. märts 2017. [Võrgumaterjal]. Available: https://swagger.io/blog/api-strategy/what-is-api-lifecycle-management/. [Kasutatud 28. märtsil 2022].

[12] R. Chandramouli, „Security Strategies for Microservices-based Application Systems,“ *Special Publication (NIST SP),* pp. 1-42, 7. august 2019.

[13] J. T. Zhao, „Management of API Gateway Based on Micro-service Architecture,“ *Journal of Physics: Conference Series,* p. 34, august 2018.

[14] M. Tanner, „API Management 101: Rate Limiting,“ Tyk, 03. veebruar 2021. [Võrgumaterjal]. Available: https://tyk.io/blog/api-management-101-rate-limiting/. [Kasutatud 20. märtsil 2022].

[15] M. T. Nygard, Release It! Desing and Deploy Production-Ready Software, Dallas: The Pragmatic Programmers, LLC, 2007.

[16] WSO2, „WSO2 NAMED A LEADER BY FORRESTER, Q3 2020,“ WSO2, [Võrgumaterjal]. Available: https://webcache.googleusercontent.com/search?q=cache:BklCQl8IqNoJ:https://wso2.com/resources/analyst-reports/the-forrester-wave-api-management-solutions-q3-2020/+&cd=1&hl=et&ct=clnk&gl=ee&client=firefox-b-d. [Kasutatud 6. aprillil 2022].

[17] G. Lawton, „Compare cloud API management tools from AWS, Azure and Google,“ TechTarget, 20. august 2018. [Võrgumaterjal]. Available: https://www.techtarget.com/searchcloudcomputing/tip/Compare-cloud-API-management-tools-from-AWS-Azure-and-Google. [Kasutatud 10. aprillil 2022].

[18] L. Võhandu, Subjektiivsetest hinnangutest objektiivsete tulemusteni : Loengukonspekt, Tallinn: Tallinna Tehnikaülikooli trükikoda, 1998.

[19] E. Forman ja M. A. Selly, Decision By Objectives How to Convince Others That You are Right, New Jersey: World Scientific Publishing Co Pte Ltd, 2001.

[20] Super Decisions, „Tutorial on Hierarchical Decision Models (AHP),“ [Võrgumaterjal]. Available: https://www.superdecisions.com/sd_resources/v28_man03.pdf. [Kasutatud 6. mail 2020].

Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[1]

Mina, Jarmo Jevonen

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Rakendusliideste haldusplatvormide analüüs mikroteenuste näitel“, mille juhendaja on Einar Kivisalu
   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

16.05.2022

1 Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või enam isikut oma ühise loomingulise tegevusega ning lõputöö kaas- või ühisautor(id) ei ole andnud lõputööd kaitsvale üliõpilasele kindlaksmääratud tähtajaks nõusolekut lõputöö reprodutseerimiseks ja avalikustamiseks vastavalt lihtlitsentsi punktidele 1.1. ja 1.2, siis lihtlitsents nimetatud tähtaja jooksul ei kehti.

Lisa 2 – Analüütiliste hierarhiate protsessi otsustusmudel

Parima rakendusliideste haldusplatvormi leidmise otsustusmudeli fail asub järgneval lingil: https://drive.google.com/drive/folders/1_ueDU3H45Sn3Hqiu40RZahPx3iI24ll1.

Super Decisions tarkvara dokumentatsioon asub järgneval lingil: https://www.superdecisions.com/tutorials/index.php?section=3_0.