TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Kristjan Kattus IADB185146

Andmemuudatuste kontrolljälje teenus mikroteenuste arhitektuuriga

Bakalaureusetöö

Juhendaja: Margus Jäger
Magister

Kaasjuhendaja: Mart Simisker
Magister

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: Kristjan Kattus

16.05.2022

Annotatsioon

Käesolev lõputöö käsitleb prototüüp auditlogimise teenuse Qure Audit analüüsimist ja arendamist mikroteenuse arhitektuuriga, kasutades sõnumite vahendajaid. Qure Audit teenuse loomise eesmärk on luua standardne lahendus, mis vähendab korduvat audilogimise funktsionaalsuse arendamist, tõstes arendajate efektiivsust ja hajutades kirjete hoidmist detailandmeid säilitavatest andmebaasidest, vähendades nende mahtu ja kasutust.

Töö eesmärgiks on anda ülevaade olemasolevatest auditlogi säilitamise teenustest ning välja selgitada nende murekohad. Kirjeldada ja võrrelda vabavaralisi sõnumite vahendajaid ja sooritada Qure Audit teenusele süsteemi analüüs, selgitades välja süsteemi nõuded. Kirjeldada arenduse käigus loodud lahendus ja selle arhitektuur ning tehnoloogiad. Lõpetuseks panna kirja loodud lahendus ja tuua välja lõputöö tulemused ja järeldused ning teenuse tulevik. Lõputöö tulemusena valmib edukalt analüüsitud ja arendatud auditlogimise mikroteenuse Qure Audit prototüüp, mida on võimalus tulevikus ettevõttes kasutusele võtta.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 28 leheküljel, 9 peatükki, 4 joonist, 3 tabelit.

Abstract

Data Change Audit Trail Service with Microservice Architecture

The given graduation deals with analysing and developing prototype audit logging service with message broker using microservice architecture. The goal of Qure Audit is to provide a standardized service, which will reduce repeated audit logging feature implementation whilst increasing the productivity of developers and distributing logging data from operative databases, which will lead to reduced size and usage.

The goal of graduation thesis is to give an overview of existing audit logging features and bring up their shortcomings. To describe and compare different open-source message brokers and to perform system analysis for Qure Audit service. Describe the service and its architecture, with explanation on used technologies. Finally write down the description for developed solution and results with conclusions and potential future for service. The result of the graduation thesis is successfully analysed and developed audit logging service prototype, which in future can be taken into production within the company.

The thesis is in Estonian and contains 28 pages of text, 9 chapters, 4 figures, 3 tables.

Lühendite ja mõistete sõnastik

AMQP	Advanced Message Queuing Protocol, standard ärisõnumite vahendamiseks aplikatsioonide ja organisatsioonide vahel
API	*Application Program Interface*, rakendusliides
Docker	Rakenduste ehitamise ja paigaldamise tööriist
Fanout	Vahetusviis, kus sissetulevad sõnumid saadetakse kõikidele järjekordadele, mis on seotud eelnimetatud vahetusega
GET	Meetod andmete pärimiseks
Java	Objekt-orienteeritud programmeerimiskeel
JMS	*Java Message Service*, Java sõnumi teenus
JSON	*JavaScript Object Notation*, Andmevahetusvorming
Lõpp-punkt	Üks kommunikatsiooni ahela otsast
Maven	Projekti ehitamise töörist
POJO	*Plain old Java Object*, tavaline Java objekt
POST	Meetod andmete lisamiseks või uuendamiseks
PostgreSQL	Objekt-relatsiooniline andmebaasimootor
REST	*Representational State Transfer*, arhitektuuriliste piirangute kogum
Spring Boot	Springil põhinev raamistik, mis pakub automaatset konfiguratsiooni parimate tavade järgi
Topic (ActiveMQ)	Sõnumite avaldamise viis, kus saadetud sõnum läheb kõikidele *topic*'u külge märgitud vastuvõtjatele

Topic (RabbitMQ)	Vahetusviis, kus sissetulevad sõnumid saadetakse kõikidele järjekordadele, mis on seotud eelnimetatud vahetusega või on seotud *wild-card*'ga
Wild-card	Tähemärk, mis vastab otsingus ükskõik millisele tähemärgile või tähemärkide järjestusele

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Mikroteenuste arhitektuur on tänapäeval järjest enam kasutatav infosüsteemide arendamisel ja paljud seni monoliitsed süsteemid viiakse üle mikroteenuste kasutamisele. Paljudel juhtudel on infosüsteemile seatud nõudeks säilitada andmete muudatuste kontrolljälg ehk auditlogi. See võimaldab täpselt ajas tagasi liikuda ja näha käsitletava kirje kohta, kes ja millal, milliseid andmeid lisas, muutis või kustutas. Ühes infosüsteemis võib olla kümneid mikroteenuseid, mis peavad andmete ajalugu säilitama ja suutma kasutajale esitada inimloetaval kujul.

Lõputöö eesmärgiks on arendada mikroteenusel põhinev korduvkasutatav andmemuudatuste kontrolljälje teenus Qure Audit prototüüp, mis omab tulevikus potentsiaali sirguda tarnevalmis teenuseks ja omada piisavalt väärtust ettevõttes kasutusele võtmiseks. Auditlogimise teenus aitaks vähendada funktsionaalsuse kordumist ja tõstes detailandmeid omava andmebaasi jõudlust ja mahtu.

Käesolevas lõputöös kirjeldatakse auditlogimise põhimõtteid ning nende lahendusi monoliitse arhitektuuriga rakenduses ja samuti mikroteenuse arhitektuuriga teenuses. Tuuakse esile mõlema lahenduse nõrkused ja kirjeldatakse autori poolt loodava teenuse võimalusi lahendada mõlema poolsed puudused. Lisaks antakse ülevaade sõnumitele orienteeritud vahevarast ja minnakse süvitsi sõnumite vahendajatega. Kirjeldatakse detailselt kolme vabavaralist sõnumite saatjat, keskendudes arendatava teenuse omapäradele. Põhjendati valitud analüüsi ja arenduse talitusviis. Analüüsis tuuakse välja arendatava teenuse nõudmiste kogumise viisid ja nende tulemusena tekkinud nõuded ning jaotatakse need vastavalt funktsionaalseteks ja mittefunktsionaalseteks. Valitakse probleemi lahendamiseks sobivad tehnoloogiad ja kirjeldatakse loodud teenuse arhitektuuri. Tulemuse peatükis kirjeldatakse loodud lahendust Qure Audit ning analüüsitakse vastavust süsteemi nõuetele ja tuuakse välja lahenduse tulevik.

1.1 Probleemi kirjeldus

Riigile erinevaid infosüsteeme arendatavas ettevõttes on infosüsteemide nõuetes tihtipeale ettenähtud andmemuudatuste kontrolljälje ehk auditlogi säilitamine. Ettevõttes riigi jaoks arendatud infosüsteemid on siiamaani olnud monoliitse arhitektuuriga. See tähendab, et nõuete täitmiseks tuleb iga infosüsteemi põhiselt rakendada andmemuudatuste kontrolljälje funktsionaalsust iseseisvalt. Infosüsteemis arendatud auditlogimise funktsionaalsus arendatakse ühekordselt ning pole reeglina võimalik taaskasutada järgmises infosüsteemis.

Ettevõttes arendatavad infosüsteemid aga hakkavad liikuma järjest enam mikroteenuste arhitektuuri suunas. Andmemuudatuste kontrolljälje nõudmise täitmiseks peavad kõik infosüsteemis olevad mikrosüsteemid rakendama auditlogimise funktsionaalsust iseseisvalt, integreerides lahendus konkreetset ülesannet täitva mikroteenuse külge.

Mõlema infosüsteemi arhitektuuri puhul esineb sama probleem. Ühekordne süsteemi põhine lahendamine nõuab lisaressursse. Nii monoliitne kui ka mikroteenuse siseselt integreeritud auditlogimine koormab süsteemi, teostades mitte eesmärgipõhist ülesannet, koormates detailandmete andmebaasi seisvate andmetega.

Kuna ettevõttes riigi jaoks arendatavad infosüsteemid liiguvad mikroteenuste arhitektuuri suunas, siis on ettevõttel vajadus pakkuda mikroteenuste arhitektuuril põhinevate infosüsteemide jaoks üldistatud lahendust, mis võimaldaks säilitada andmete muudatuste kontrolljälge ehk auditlogi. Seega on vaja arendada välja uus lahendus, mida saaks koduvalt kasutada erinevates süsteemides, pakkudes töökindluse ning võimalikult suure jõudluse ja suudaks tulla toime liidestavate komponentide katkestustega. Uuelt lahenduselt oodatakse sõltumatut asünkroonset mitteblokeerivat töötamist ning sobivust erinevate teenuste ja Javal põhinevate infosüsteemide jaoks.

1.2 Eesmärk

Lõputöö eesmärgiks on ettevõtte Quretec OÜ jaoks analüüsida ja välja arendada mikroteenusel põhinev korduvkasutatav andmemuudatuste kontrolljälje teenus Qure Audit prototüüp. Teenus omab tulevikus potentsiaali sirguda tarnevalmis teenuseks ja omada piisavalt väärtust ettevõttes kasutusele võtmiseks.

Auditlogimise teenus peab olema suuteline vastu võtma klientide (teised mikroteenused) poolt salvestamist vajavad kirjed (auditlogid), säilitama kirjeid liidestatud teenuste katkestuste korral ning kirjed salvestama eraldiseisvasse andmebaasi. Teenus peab võimaldama kliendil pärida kirjega seotud auditlogisid. Teenuselt oodatakse sõltumatut asünkroonset mitteblokeerivat töötamist ning sobivust erinevate teenuste ja Javal põhinevate infosüsteemide jaoks.

Lõputöö eesmärgi saavutamiseks jagatakse kontrolljälje teenuse loomine kaheks etapiks. Esimeses etapis analüüsitakse lõputöö eesmärgina välja toodud potentsiaalsele lahendusele konkureerivaid lahendusi. Seejärel leitakse analüüsi tulemusena kõige sobilikum sõnumite vahendaja. Sõnumite vahendaja peab olema suuteline tagama võimalikult suure jõudluse ja töökindluse ning säilitama auditlogitavad kirjed, kui liidestatavad teenused ei ole kogu aeg kättesaadavad. Seejärel kogutakse auditlogimise teenuse nõuded ja pannakse nende põhjal paika kasutuslood.

Teises etapis kirjeldatakse analüüsi põhjal tehnoloogiate valikut. Antakse detailne ülevaade loodud teenuse arhitektuurist ja alamteenustest. Samuti fikseeritakse loodud lahendus ja esitatakse lõputöö käigus saadud tulemused ja järeldused ning pakutakse välja loodud lahenduse tulevik.

Tulemusena valmib prototüüp auditlogimise teenus Qure Audit, mis on standardne taaskasutatav lahend, vähendades seega funktsionaalsuse kordumist ja tõstes detailandmeid omava andmebaasi jõudlust ja mahtu.

2 Auditlogimine

Peatükis annab autor ülevaate auditlogimisest, selle eesmärgist ja kasulikkusest. Samuti antakse ülevaade analoogsetest auditlogimiste viisidest ning nende erinevustest ja probleemidest.

Auditlogi, teise nimega auditjälg, on põhimõttelisel tasandil kirje sündmustest ja muutustest. Auditlogid jäädvustavad sündmused, salvestades järgnevad väljad: tegevuse sooritaja, milline tegevus sooritati ja kuidas süsteem sündmusele vastas. Auditlogid teevad märkmeid kõikidest auditeeritavatest või jälgitavatest muutustest süsteemis, pakkudes täielikku kirjete kogumite süsteemi operatsioonidest [1].

2.1 Andmete rikastamine

Andmete rikastamine on protsess, mille käigus lisatakse esimese osapoole andmetele kas sisemise süsteemi teised andmed või kolmanda osapoole andmed, et saavutada veelgi kvaliteetsem andmete kogum [2]. Klassifikaatori puhul olgu selleks antud aja hetkel kehtiv ja kuvatav tekst. Keerukama kirje puhul olgu selleks kehtiva seisu kuupäev.

2.2 Auditlogimise eesmärk

Kliendil (teine mikroteenus) on vaja konkreetse objekti muutuste ajalugu. Kontekstist lähtuvalt võib see olla näiteks haiglas oleva patsiendi haiguslugu, kinnisvaras hoone kasutamine või hoopis auto hooldusraamat. Suures kasutajale kuvatavas tabelis koostatakse andmed mitme teenuse abil mitmest andmebaasi tabelist. Kui suures kuvatavas tabeli andmetes tekib uus seis, tuleb vastavat andmeveergu omaval teenusel eraldi salvestada uus objekti olek. Kliendil võib tekkida vajadus liikuda ajaloos tagasi ja näha objektiga seotud muudatusi, olgu selleks objekti uus, muudetud või kustutatud seis. Selleks tuleb peale uue kirje jõustumist jäädvustada ajalugu ka eelmisest seisust. Andmemuudatuste ajalugu ehk auditlogi peab tagama kellaajalise vastuse küsitava informatsiooni üle. Selleks vastuseks on milline oli info ajahetkel, kes tekitas uue seisu või kes muutis seisu ja milline on uus seis.

2.3 Auditlogimine monoliitse arhitektuuriga rakenduses

Alampeatükis kirjeldab autor monoliitset arhitektuuri ning toob välja ühe konkureerivatest auditlogimise mudelitest. Monoliitse arhitektuuriga lahendus täidab küll auditlogimise ülesande, kuid jätab soovida teatud aspektides, millest autor järgnevalt kirjutab.

2.3.1 Monoliitne arhitektuur

Monoliitne arhitektuur on traditsioonilise ja ühtse tarkvara rakenduste disainimise mudel. Monoliit on ühte serverisse paigaldatud tarkvararakendus, kus kõik rakenduses olevad tarkvara osad on omavahel tihedalt seotud ning töötavad ühtse kooslusena. Monoliitne rakendus pakendatakse ühe paigaldatava tükina.

2.3.2 Korduv funktsionaalsuse arendamine monoliitse arhitektuuriga lahenduses

Monoliitse rakenduse puhul on kõik tarkvara osad kokku liidetud. Soovides lisada rakendusele auditlogimist, tuleb lisada vastav funktsionaalsus kõikidele teenustele, protsessidele ja andmetabelitele. Antud lähenemine toob kaasa auditlogimise funktsionaalsuse korduvat arendamist ja rakendamist, mis omakorda nõuab ettevõttelt rohkem aega, analüüsi ja arendajaid, mille tõttu tõuseb nii auditlogimise lahenduse väljatöötamise aeg kui ka lahenduse maksumus.

2.3.3 Jõudlus monoliitse arhitektuuriga lahenduses

Monoliitses rakenduses suurendab auditlogimine teenuste vahelist päringute teostamiseks kuluvat aega. Ülesanne peab ootama seni, kuni on suudetud ära logida toimunud sündmus. Samuti peab auditlogimine ka arvestama andmebaasi poolsete ressurssidega, hoides auditlogisid ühes ja samas detailandmete andmebaasis, tõuseb kirje lugemisel kuluv aeg. Hoides auditlogi kirjed eraldi tabelites võib tekkida olukord, kus auditlogi koostamisel lukustatakse andmebaasis teise teenuse poolt kasutatav tabel seni, kuni auditlogija pole enda poolset lukku eemaldanud, niikaua pole teisel teenusel võimalik sooritada päringuid. Alles pärast auditlogimise protsessi lõppemist on töös oleval ülesandel võimalik oma tegevustega jätkata ning saata vastus. Ajakulu on võimalik vähendada paralleelse alamülesannete täitmisega, kuid nõuab asünkroonset ülesande täitmist.

2.4 Auditlogimine mikroteenuses

Alampeatükis kirjeldab autor lühidalt mikroteenuse arhitektuuri, seejärel toob esile mikroteenuse arhitektuuriga konkureeriva auditlogimise viisi, milleks on mikroteenusesse integreeritud auditlogimine. Autor kirjeldab konkureeriva auditlogija mudeli töö põhimõtteid ning toob esile auditlogija puudused.

2.4.1 Mikroteenuse arhitektuur

Mikroteenuse arhitektuur hõlmab endas väiksemaid teenuseid, mis on iseseisvalt kasutusele võetud, kergelt seotud ja omavahel ühendatud läbi mikroteenuse integratsiooni. Üks mikroteenus omab ühte eesmärki ja on eraldiseisev teistest mikroteenustest [3].

2.4.2 Korduv funktsionaalsuse arendamine integreeritud auditlogimisega teenuses

Kuigi mikroteenuse arhitektuur omab väiksemaid eraldiseisvaid teenuseid, vajavad siiski kõik andmeid haldavad (tootvad/muutvad) teenused auditlogimise jaoks funktsionaalsuse arendamist ja rakendamist. Kuigi suur osa auditlogis olevatest andmetest on vastava teenuse põhine, tuleb väiksemas osas andmeid väljast poolt andmete rikastamise käigus. Andmete rikastamiseks tuleb mikroteenustel omavahel liidestada, mis nõuab lisatööd ja aega.

2.4.3 Jõudlus integreeritud auditlogimisega

Mikroteenus, millesse on integreeritud auditlogimine, lahendab monoliitses auditlogimises oleva probleemi, kus auditlogimine koormab oma tegevusega tervet süsteemi. Mikroteenusena saavad teised teenused jätkata enda ülesannetega ja auditlogi kirjutav teenus saab salvestada kirje uue seisu. Siiski jääb õhku probleem, kus teenuse peamise ülesande läbiviimine on blokeeritud seni, kuni auditlogimise ülesanne on sooritatud. Mikroteenusena auditlogimine tõstab jõudlust, kuid andmebaasi poolne pudelikael võib siiski tekkida, kui auditlogikirjed pole eraldatud detailandmete andmebaasist.

3 Sõnumite vahendajad

Peatükis annab autor ülevaate sõnumitele orienteeritud vahevarast ja sellele kuuluva alamliigi sõnumite vahendajatest. Täpsemalt kirjeldatakse ja tuuakse välja erinevaid sõnumite vahendajaid koos nende eripärade ja funktsionaalsustega.

3.1 Sõnumitele orienteeritud vahevara

Sõnumitele orienteeritud vahevara on arhitektuur hajutatud süsteemidele, mis lubab suhtlust ja andmete (sõnumite) vahetamist. Suhtlus põhineb andmete edastamisel rakenduste vahel, kasutades suhtluskanaleid, mis kannavad iseseisvaid informatsiooni ühikuid. Sõnumitele orienteeritud vahevara pakub asünkroonset, kergelt ühendatud, vastupidavat, skaleeritavat ja turvalist viisi suhtlemiseks hajutatud rakenduste ja süsteemide vahel [4]. Alljärgnevalt uuritakse lähemalt sõnumitele orienteeritud vahevara arhitektuuri sõnumite vahendajate näol.

3.2 Sõnumite vahendajad

Sõnumite vahendaja on vaheprogramm, mida rakendused ja teenused kasutavad, et omavahel suhelda ja vahetada informatsiooni. Sõnumite vahendaja töötab põhimõttel, kus saadetav sõnum tõlgitakse saatja poolt kasutatud protokollist vastuvõtja poolt kasutatavasse protokolli. Sõnumite vahendajaid on võimalik kasutada sõnumite valideerimiseks, hoidmiseks, suunamiseks ja kohale toimetamiseks ettenähtud sihtkohta. Kuna sõnumite vahendaja on vaheprogramm, on võimalik vahetada informatsiooni isegi siis, kui saatja ja vastuvõtja on rakendatud erinevates programmeerimise keeltes ning saatja ei pea teadma, kui palju on vastuvõtjaid või kas vastuvõtjad on sõnumi saatmise hetkel üldse kättesaadavad. See võimaldab saata sõnumeid asünkroonselt, kuna saatja ei pea muretsema sõnumi kohale toimetamise eest ja saab tegeleda järgmiste ülesannetega. Sõnumite vahendajatel on neli peamist komponenti [5]:

- saatja – lähtepunkt, mis saadab andmed sõnumi vahendajale, kes omakorda sõnumeid hoiab, et sõnumid hiljem laiali jagada;
- vastuvõtja – lõpp-punkt, mis võtab vastu sõnumeid sõnumi vahendajalt ja tegeleb sõnumitega edasi vastavalt äriloogikale;

- järjekord – puhver, mida sõnumi vahendaja kasutab sõnumite hoidmiseks. Järjekord töötab loogikal: esimesena sisse, esimesena välja. Järjekord hoiustab sõnumeid kuni vastuvõttev teenus võtab sõnumeid töötlemiseks vastu. Järjekorrad avavad võimaluse asünkroonseks programmeerimiseks, kuna saatja ei pea tegelema sõnumi kohale toimetamisega;
- vahetaja – võib olla kas loogiline konfiguratsioon või hoopis olem, mis võtab vastu sõnumeid saatjatelt ja lükkab need edasi järjekordadesse vastavalt sätestatud vahetaja tüübile.

Sõnumile orienteeritud vahevara raames on olemas üle 25 sõnumi vahendaja [6]. Kõigi olemasolevate sõnumite vahendajate vahel on väga palju erinevaid tunnusjooni, valiku tegemisel peetakse kinni süsteemi nõuetest, kuid kattuvate tunnusjoonte korral langeb valik tihti detailidele.

Sõnumite vahendajaid hinnatakse tuginedes järgnevatele tunnusjoontele, kuid need pole ainsad, mis aitavad sõnumi vahendajaid valida [3]:

- toetatud programmeerimiskeeled – kasulik valida sõnumi vahendaja, mis toetab mitmeid programmeerimiskeeli;
- toetatud sõnumi saatmise standardid (protokollid) – kas sõnumi vahendaja toetab standardeid nagu JMS, AMQP või STOMP? Kas standard on patenteeritud?
- sõnumite järjekorra hoidmine – kas sõnumi vahendaja säilitab sõnumite järjekorra?
- sõnumite püsivus – kuidas sõnumeid säilitatakse ja kas sõnumid suudavad „üle elada“ sõnumi vahendaja katkestused?
- vastupidavus – kas sõnumi vastuvõtja saab vastu võtta sõnumid, mida saadeti ajal, mil sõnumi vastuvõtja oli lahti ühendatud sõnumi vahendajast?

Alljärgnevalt uuritakse lähemalt kolme populaarsemat vabavaral põhinevat sõnumite vahendajat, nende omapärasid ja sarnasusi teiste sõnumite vahendajatega ning vastavust eelnevalt välja toodud nõuetele. Uurimusse võetud sõnumite vahendajad valiti järgmiste kriteeriumite näol: avatud lähtekoodiga, mitmeid protokolle toetav, iseseisev ja aktiivselt arenduses olev.

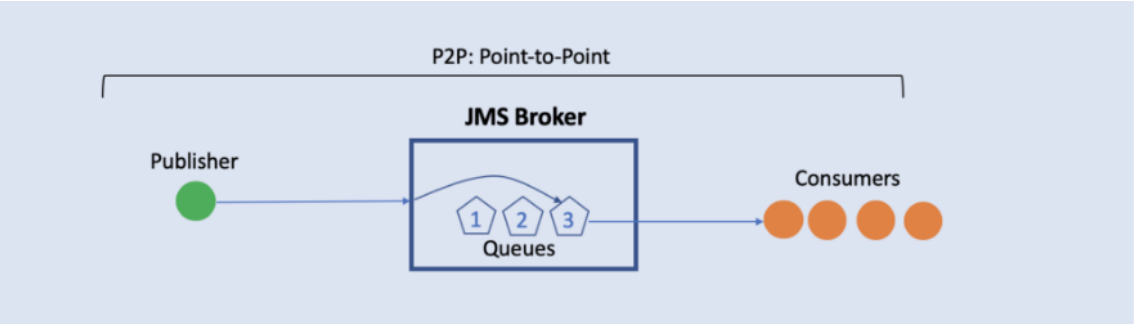
3.2.1 ActiveMQ

ActiveMQ on Apache tarkvara sihtasutuse (Apache Software Foundation) poolt arendatud, avatud lähtekoodiga, Java-le põhinev sõnumite vahendaja, mis pakub kõrget kättesaadavust, jõudlust, skaleeritavust, töökindlust ja turvalisust ettevõtte tasandil sõnumite saatmiseks [7]. Alljärgnevalt tuuakse välja ActiveMQ tunnusjooned ja võimalused.

JMS 1.1 vastavus – Java sõnumi teenus (*Java Messaging Service)* on Sun mikrosüsteemide (Sun Microsystems) ja teiste ettevõtete poolt väljatöötatud API, mis võimaldab rakendusel luua, saata, võtta vastu ja lugeda sõnumeid. JMS standard defineerib ühiste liidestuste komplekti ning seotud semantika, mis võimaldab Java programmeerimiskeeles kirjutatud rakendustel omavahel suhelda, kasutades teisi sõnumi saatmise rakendusi [8].

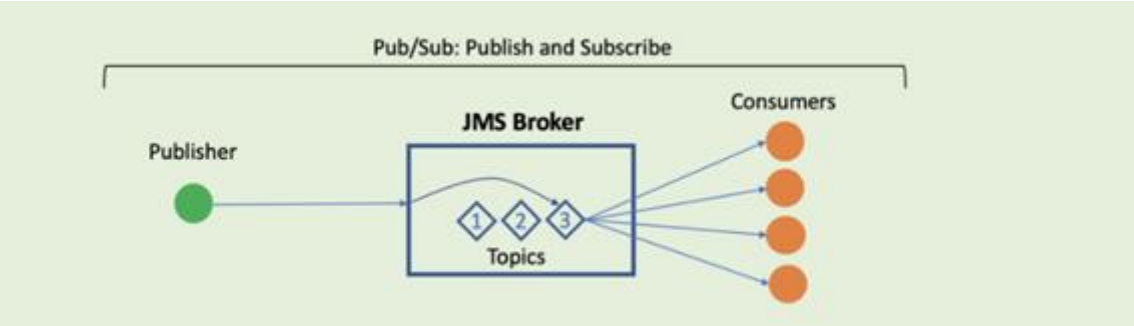
Sõnumite saatmine – Kasutades JMSi, ActiveMQ pakub kahte sõnumi edastuse viisi. Punkt punktile sõnumi edastuse viisi puhul ei ole garanteeritud sõnumite järjekord. Sõnumite järjekord säilib ainult juhul, kui järjekorral on ainult üks vastuvõtja ja sõnumid on saadetud ainult ühe saatja poolt. Mitme vastuvõtja puhul jaotab ActiveMQ sõnumid laiali erinevatesse järjekordadesse koormuse vähendamise eesmärgil. „Avalda ja telli“ puhul on garanteeritud sõnumite järjekord juhul, kui sõnumid on saadetud ühe saatja poolt.

Punktist punkti (PTP) – Sõnumeid saadetakse ja võetakse vastu sünkroonselt või asünkroonselt, kasutades järjekordasid. Iga vastuvõetud sõnum toimetatakse ühele vastuvõtjale vaid ühe korra. Järjekord hoiab sõnumi endas kuni see on kohale toimetatud või kuni sõnum on aegunud. Järjekorra külge on võimalik registreerida mitu kuulajat, kuid ainult üks vastuvõtja saab sõnumi ning peab sõnumi kohale toimetamise individuaalselt kinnitama (Joonis 1).



Joonis 1. Punktist punkti sõnumi saatmine [9]

Avalda ja telli (P/S) – Sõnumeid on võimalik saata *topic*'le sünkroonsel või asünkroonsel viisil, mille järel saadetakse sõnum automaatselt laiali kõigile vastuvõtjatele, kes on ennast *topic*'u külge märkinud (Joonis 2).



Joonis 2. Avalda ja telli sõnumi saatmine [9]

Ühilduvus – ActiveMQ poolt ühenduseks pakutavate protokollide alla kuuluvad: AMQP, AUTO, MQTT, OpenWire, REST, STOMP, XMPP ja veel [10]. Lai protokollide valik aitab kaasa paindlikkusele. Süsteemidel on võimalik valida endale sobiv protokoll, mis võimaldab võtta kasutusele ActiveMQ sõnumi vahendaja.

Kliendipõhine API – ActiveMQ pakub kliendi APIsid mitmetele programmeerimiskeeltele, nendeks võivad olla: Java, JavaScript, C, C++, .NET, Perl, PHP, Python, Ruby ja veel. Kuigi ActiveMQ töötab siiski Java VMl, on võimalik kirjutada klienti kõigis eelnimetatud keeltes.

Sõnumite säilitamine – JMS spetsifikatsioon toetab kahte tüüpi sõnumi saatmist: püsiv ja mittepüsiv. Püsiva saatmise korral kirjutatakse sõnum kõigepealt valitud sõnumite hoiustamise viisile ning seejärel üritatakse sõnumit kohale toimetada. Mittepüsiva

saatmise korral ei hoiustada sõnumeid ja üritatakse toimetada sõnum kohale aktiivsetele vastuvõtjatele. ActiveMQ toetab mõlemat neist variantidest ning toetamaks sõnumi taastamist, omab võimalust ka seadistada vahepealset olekut, kus sõnumit hoitakse vahemälus. ActiveMQ toetab ühendavat strateegiat sõnumite hoiustamiseks ja pakub valikute näol mälus, failis või relatsioonilises andmebaasis sõnumite hoiustamist [7].

3.2.2 RabbitMQ

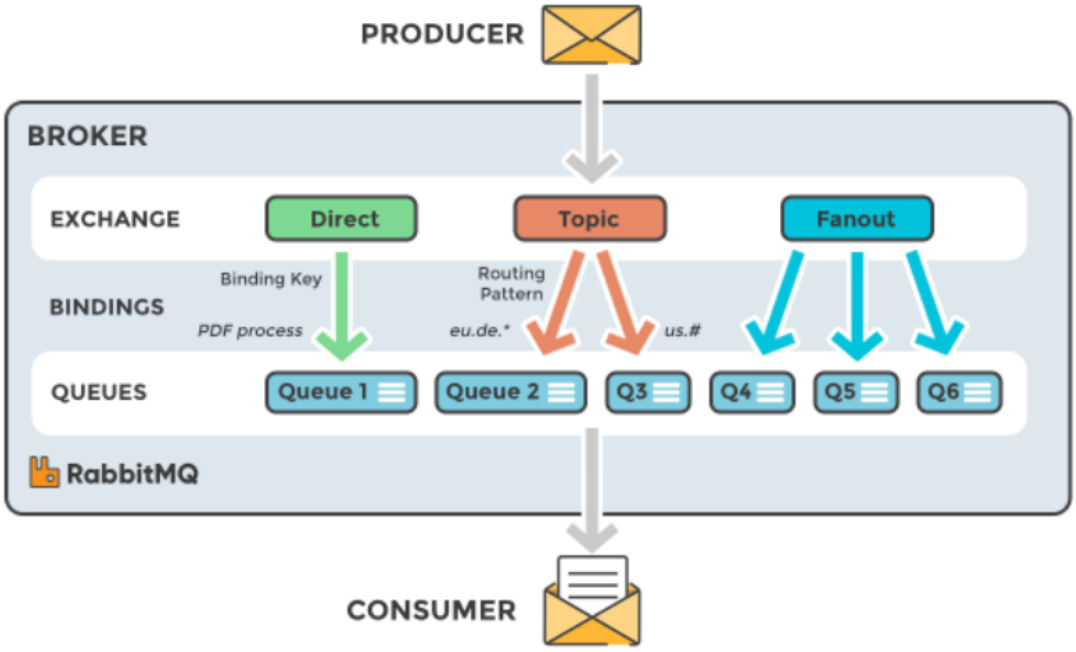
RabbitMQ on avatud lähtekoodiga sõnumi järjekorra mudelit laiendav lahendus, mis on äärmiselt kättesaadav, veakindel ja skaleeritav. Sõnumi vahendaja on RabbitMQ valinud kasutatavaks protokolliks AMQP, kuid omab võimalust kasutada ka teisi populaarseid sõnumite saatmise protokolle nagu MQTT ja STOMP. RabbitMQ on kirjutatud Erlang programmeerimiskeeles, mistõttu sõnumi vahendaja suudab käidelda väga suurtes numbrites samaaegseid operatsioone [11].

Sõnumite saatmine – RabbitMQ’s ei avaldata sõnumeid otse järjekorrale. Saatja saadab sõnumi vahetajasse. Vahetaja vastutab suunamise eest erinevatesse järjekordadesse, kasutades selleks sidumisi ja suunamisvõtit. Sidumine on lüli vahetuse ja järjekorra vahel. Kuna RabbitMQ kasutab samuti koormuse vähendamiseks sõnumite jaotamist erinevate vastuvõtjate vahel, kaob ka selle tõttu sõnumite järjestus. Kui selleks on olemas lahendused räsi vahetuse ja ühe aktiivse vastuvõtja näol. RabbitMQ pakub nelja erinevat sõnumite vahetuse viisi, alljärgnevalt antakse neist lühike funktsionaalsuse põhine ülevaade:

Otsene vahetus – Toimetab sisse tulevad sõnumid kõikidele järjekordadele, kelle sidumisvõti vastab üks ühele sõnumi marsruutimise võtmele. Kui sidumisel kasutada marsruutimisvõtmena järjekordade endi nimesid, on võimalik tõlgendada otsest vahetust kui ühelt ühele sõnumi saatmist (Joonis 3)..

***Topic* põhine vahetus –** Toimetab sisse tulevad sõnumid järjekordadele, kelle *wild-card* sidumisvõti vastab publitseeritud sõnumi marsruutimisvõtmele. Sidumisvõtmed võivad sisaldada *wild-card*’le vastavat kriteeriumit liitmarsruutimise võtit. Näiteks sidumisvõti „auditlog.*. entity” vastab marsruutimis võtmetele “auditlog.one.entity” ja “auditlog.two.entity” (Joonis 3)..

***Fanout* põhine vahetus –** Toimetab saadetud sõnumid kõikidele vastuvõtjatele, kes on oma poolt kuulatavad järjekorrad *fanout* vahetuse külge sidunud. Vastavalt vahetustüübile pole ka sõnumi marsruutimiseks vajadust sidumisvõtmele. Isegi kui on sidumisvõti antud, *fanout* vahetus ignoreerib sidumisvõtit ja suunab või avaldab sõnumi kõikidele järjekordadele, kes on vahetuse külge ennast sidunud (Joonis 3).



Joonis 3. Sõnumite marsruutimine RabbitMQ's [12]

Päise põhine vahetus – AMQPi kasutades on võimalik ära kasutada protokolli poolset marsruutimise loogikat ning suunata sõnumit päise välja kaudu. Sarnanedes *topic*'le kasutab suunamiseks päise põhine vahetus võtme-väärtus paare, ignoreerides suunamisvõtit. Toetab sõnumi päises atribuuti „x-match=all/any", kus „*all*" vastab terve sõnumi atribuutidele ja „*any*" otsib ainult ühe vastet.

Sõnumite säilitamine – Sõnumite säilimiseks RabbitMQ's taaskäivitamisel peab olema täidetud kaks alljärgnevalt kirjeldatud nõudmist: järjekordade ja vahetajate püsivus ning sõnumi püsivus. Kui üks nõudmistest on täitmata, ei jää ka sõnumid sõnumi vahendaja taaskäivitamisel „ellu". See on aga ainult üks võimalikest kihtidest kindlustamaks sõnumi kohale jõudmist ja sõnumitega tehtava töötluse täitmist. Lisakihtidena on võimalik veel lisada sõnumite kinnitamine ja saatja poolne kinnitamine.

RabbitMQ’s on võimalik järjekordasid ja/või vahetajaid deklareerides seadistada nende vastupidavus vastupidavana või mitte vastupidavana. Vastupidavana jäävad alles ja taastavad töövõime nii vahetused kui ka järjekorrad, kirjutades metaandmed kettale, peale RabbitMQ taaskäivitamist. Mitte vastupidavana kaovad deklareeritud järjekorrad ja vahetajad.

Sõnumite saatmisel on võimalik lisada sõnumi püsivust. RabbitMQ’s on sõnumi loomisel kaks võimalikku sõnumi püsivuse varianti: mittepüsiv ja püsiv. Püsiva puhul kirjutatakse sõnumid kettale ja taaskäivitades on kettalt võimalik sõnumeid uuesti toimetada. Mittepüsiva sõnumi puhul hoitakse sõnumit mälus ja sõnumi vahendaja katkemisel kustub sõnum mälust ja pole võimalik taastada.

3.2.3 Kafka

Kafka on arendatud 2011 LinkedIn poolt ning hiljem annetatud Apache Software Foundation’le. Sõnumi vahendaja on Apache Kafka avatud lähtekoodiga, hajutatud ja jaotatud voogesituse platvorm, mis võimaldab arendada reaalajas sündmuspõhised rakendusi. Kafka toetab mitmeid erinevaid programmeerimiskeeli nagu näiteks Python, Java, C, C++, JavaScript, Swift.

Sündmus – Sündmused Kafkas on iseenesestmõistetavad, nad jäädvustavad mingi protsessi toimumist kas maailmas või äris. Sündmus koosneb võtmest, väärtusest, ajatemplist ja valikulisest päise metaandmetest. Kafkas kutsutakse sündmust ka kirjeks või sõnumiks.

***Topic* –** *Topic* on järjestatud sõnumite logi. *Topicud* Kafkas on alati nulli, ühe või mitme saatja või vastuvõtjaga. Sõnumid kategoriseeritakse *topic*’tesse. *Topic*’ud jagatakse laiali erinevateks jaotusteks. Sõnumid kirjutatakse jaotustesse ainult lisamise eesmärgil ning loetakse algusest lõpuni. Sõnumeid hoitakse alles kuni nende säilitamise aeg on aegunud, isegi kui sõnumit on juba loetud. Kuna *topic* omab enamasti mitu jaotust, ei ole garanteeritud sõnumite ajapõhine järjestus üle terve *topic*’u, vaid ainult ühe jaotuse sees. Kafkas on võimalik kasutada nelja põhilist APIt:

Saatja API – Saatjad loovad uusi sõnumeid spetsiifilise *topic*’u jaoks. Vaikimisi saatja ei vali millisele jaotusele spetsiifiline sõnul läheb, vaid jagab sõnumid kõikide jaotuste vahel *topic*’s ühtlaselt. Saatja siiski suudab sõnumeid saata spetsiifilisele jaotusele: kui

on sõnumis olemas võti ja jaotus genereerib võtme pealt räsi. Räsi kindlustab, et kõik sõnumid antud võtmega kirjutatakse samale jaotusele. Saatjal on ka võimalik kasutada kohandatud jaotust, mis võib vastata teistele ärireeglitele [13].

Vastuvõtja API – Vastuvõtja märgib ennast ühe või mitme *topic'u* külge ning loeb sõnumeid vastavalt sõnumite saatmise järjekorrale. Vastuvõtja hoiab sõnumite vastuvõtmisel järjekorda kasutades nihet. Nihe on järjest suurenev arv, mida Kafka lisab igale saadetavale sõnumile ja igal sõnumil jaotuses on nihe unikaalne. Vastuvõtjad töötavad koos ühe vastuvõtjate grupina, et võtta vastu *topic'*u poolt tulevaid sõnumeid. Grupp kinnitab, et iga jaotus tarbitakse ainult ühe vastuvõtja poolt [13].

Voogesituse API – Ehitab nii saatja kui ka vastuvõtja API pealt ning lisab komplektse protsessimise võimekuse, mis võimaldab rakendusel sooritada pidevat eest-taha voogesituse protsessimist. Täpsemalt kirje vastu võtmiseks ühest või mitmest *topic'*st, et analüüsida, koondada või moondada vastavalt nõudmistele ning seejärel avaldada tulemusena voogesitus tagasi samale või samadele *topic'*le. Voogesituse API annab saatja API ja vastuvõtja API kõrval võimaluse arendada keerulisemaid andme- ja sündmusvoogesituse rakendusi [14].

Ühendaja API – Võimaldab ehitada ja käivitada taaskasutatavaid andmete import/eksport ühendajaid, mis loevad ja kirjutavad sündmuseid voogesitusena välistest ja välistesse süsteemidesse ning rakendustesse, et nad saaksid integreeruda Kafka-ga [15].

Sõnumite säilitamine – Vastupidav sõnumite säilitamine garanteerib, et sõnumi vastuvõtjad ei pea alati reaalajas töötama. Sõnumid kinnitatakse kettale ja hoitakse alles vastavalt konfigureeritavatele hoiustamise reeglitele. Hoiustamise reegleid on võimalik valida *topic'*u põhiselt, lubades erinevaid võimalusi hoiustamiseks, sõltudes vastuvõtja vajadustest [13]. Vastupidav sõnumite säilitamine garanteerib andmete säilimise, kui vastuvõtja jääb töötlemisega hätta, olgu selleks aeglane andmete töötlemine, korraga suur hulk liiklust või on vastuvõtjas tekkinud katkestus.

4 Analüüsi ja arenduse talitusviis

Peatükis autor kirjeldab enda poolt valitud analüüsi ja arenduse talitusviisi, milleks on agiilne talitusviis, mille põhjal alustatakse auditlogimise teenuse analüüsi ja arenduse sooritamist. Autor annab enda poolse seisukoha, mis hõlmab talitusviisi tugevuste ja nõrkuste analüüsi.

Agiilne meetod keskendub paindlikkusele ja kohanemisvõimele. Agiilne meetod hõlmab mitut iteratiivset arenduse tähtaega, mis sihivad tarkvara väljalaske suurendamisele. Iga iteratsioon läbib kõik arenduse protsessi etapid, milleks on disain, koodimine ja testimine. Tarkvara disain pole lõplik ja hoitakse muudatuste jaoks avatuna viimase minutini. Vähem aega kulutatakse dokumentatsioonile ning rohkem rõhutatakse töötava rakenduse tarnele [16]. Agiilse meetodi tugevused on järgnevad: võimalus kiirelt reageerida muutuvatele nõudmistele, muudatused integreeritakse koheselt, kliendi sisend on kiiresti kättesaadav ning dokumendid on konkreetsed ja teemakohased. Agiilse meetodi nõrkused on järgnevad: suurte projektide puhul on raske hinnata ajakulu ja vajaminevat ressurssi, puudub pikaajaline planeerimine, keeruline suurt projekti “rajal” hoida ja on keerulisem tööjaotuse koordinatsioon. Autor eeldab nõuete muutumist teenuse arendamisel, mistõttu soosib analüüsi meetodina kasutada samuti agiilset, mis pakub võimalust nõudeid kergemini muuta. Kuna teenuse arhitektuuri nõudeks on mikroteenus, mis hõlmab endas väikest kergelt ühendatud teenust, on kergem arendada agiilse mustriga teenust, kuna agiilne meetod toetab iteratiivset ja evolutsioonilist arendamist.

5 Süsteemi analüüsi talitusviis

Süsteemi analüüs on tarkvara arenduses esimene samm. Süsteemi analüüs defineerib ja toob esile välja pakutud probleemi lahenduse. Pädeva süsteemi analüüsi puudumine arendamise alguses mõjutab süsteemi arendamist terve protsessi ulatuses, mis võib tuua lisakulusid ja kliendi rahulolematust. Kui süsteemi analüüs on korrektselt läbi viidud, juhib ja kindlustab see etapp tarkvara arenduse edukat läbiviimist.

5.1 Süsteemi nõuded

Süsteemi nõuded on kõik tegevused, mida uus süsteem peab olema võimeline sooritama või toetama ja kõik piirangud, millele uus süsteem peab vastama. Üldiselt jaotatakse süsteemi nõuded kahte kategooriasse: funktsionaalsed ja mittefunktsionaalsed nõuded [17]. Autor kasutas nõuete kogumiseks järgnevaid nõuete kogumise tehnikaid:

- ajurünnak – ajurünnak aitab luua loovaid lahendusi spetsiifilistele probleemidele. Ajurünnak koosneb kahest etapist. Ideede kogumise etapp - sellel etapil ei tohiks ideid kritiseerida ega hinnata. Seejärel tuleb hindamise etapp, kus ideid kollektiivselt arutletakse [18]. Autor valis ajurünnaku teema, kuna antud tehnika annab parema arusaama probleemist.
- nõuete töötuba – nõuete töötoas osalevad huvirühmad töötavad koos, et avastada, täiendada, prioritiseerida ja valideerida süsteemi nõudeid. Autor lisas antud nõuete kogumise viisi, et veelgi efektiivsemalt koguda ja dokumenteerida nõudeid.
- dokumendi analüüs – dokumendi analüüs on olemasoleva sarnaneva süsteemi dokumentidest nõuete kogumise viis, et anda töös olevale süsteemile lisainformatsiooni. Autor kasutab kirjeldatud nõuete kogumise viisi, et hoida ettevõttes arendatava lahenduse kvaliteet ühtsena.

Alljärgnevalt kirjutab autor täpsemalt lahti mõlemad funktsiooninõuded ning valib võrdluse alusel nõuete kirjelduse viisi.

5.2 Funktsionaalsed nõuded

Funktsionaalsed nõuded on kasutaja poolt ette nähtud tegevused, mida süsteem peab olema suuteline sooritama. Funktsionaalsed nõuded on ka nähtavad kasutajale.

Alljärgnevalt toob autor välja kaks konkureerivat funktsionaalsete nõuete esitamise viisi ning seejärel kirjeldab neid. Kirjeldamise järel teeb autor nõuete esitamise viiside vahel valiku, tuginedes loodava teenuse eesmärkidele ja võimalustele.

5.2.1 Kasutuslugu

Kasutuslugu on kirjeldus viisidest, kuidas kasutaja suhtleb süsteemi või tootega. Kasutuslugu võib luua edukad stsenaariumid või hoopis mitte edukad ning iga kriitilise variatsiooni või erandi. Kasutusmalli eesmärk on hallata skoopi, luua nõuded, välja tuua viisid, kuidas kasutaja suhtleb süsteemiga ja visualiseerida süsteemi arhitektuur [19].

5.2.2 Kasutajalugu

Kasutajalugu on lühike, informatiivne, üleüldine seletus kirjeldamaks tarkvara tunnusjoont, kirjutatud lõppkasutaja vaateväljast. Eesmärgiks on sõnastada, kuidas tarkvara tunnusjoon pakub kliendile väärtust. Kasutajaloo peamised tunnusjooned on iseseisvus, arutletav, väärtuslik, hinnatav, väike ja testitav [20].

5.2.3 Autori poolt valitud funktsionaalsete nõuete kogumise viis

Autor valib funktsionaalsete nõuete kogumiseks kasutusloo, kuna kasutajale pakutav funktsionaalsus on minimaalne, olles vaid auditlogimiseks kirje saatmine ja kirje auditlogi pärimine ning teenuses keskendutakse peamiselt tehnilistele omadustele nagu auditkirje kohale toimetamine või kirje säilitamine. Tehnilise omadustele toetudes tuleb aga väga selgelt kirja panna kõik tehnilised nõuded süsteemile. Kasutuslugu on selle jaoks optimaalsem lahendus, võimaldades detailselt ja süvitsi kirjeldada teenuse eesmärgid ning pakkuda viisi näha eelseisvat tööd ja visualiseerida teenuse arhitektuuri.

5.3 Mittefunktsionaalsed nõuded

Mittefunktsionaalsed nõuded on süsteemile disainile kehtestatud piirangud või nõuded, mis kindlustavad terve süsteemi kasutamise kõlblikkust ja efektiivsust. Mittefunktsionaalsed nõuded on piiratud, iseseisvad, läbiräägitavad ja testitavad [21].

Mittefunktsionaalsete nõuete kirjeldamise jaoks kasutab autor FURPS'i. Toetudes teenuses suures osas tehnilistele nõuetele, aitab FURPS efektiivselt kategoriseerida teenuses mittefunktsionaalsed nõuded. Alljärgnevalt annab auto ülevaate FURPS'st ning selgitab selle tugitalad mittefunktsionaalsetes nõuetes.

FURPS on akronüüm tarkvara kvaliteedi tunnuste esindamiseks, kus iga täht esindab ühte nõuete kategooriat. Nendeks on *Functionality*, *Usability*, *Reliablity*, *Performance* ja *Supportability* [17]. Järgnevalt kirjeldab autor mittefunktsionaalsusele vastavad nõuded, jättes välja *Functionality*, mis on funktsionaalne nõue:

- kasutatavus – kirjeldab operatsioonilisi omadusi seoses kasutajaga, näiteks kasutajaliides ja dokumentatsioon [17];
- töökindlus – kirjeldab süsteemi töökindlust. Kui tihti esineb probleeme nagu süsteemi katkestus või vigane protsess ning kuidas süsteem tuvastab ja taastub probleemist [17];
- jõudlus – kirjeldab operatsioonilisi omadusi seoses süsteemi koormusega, näiteks läbilaske ja vastuse kiirusega [17];
- toetatavus – kirjeldab kuidas süsteem installitakse, seadistatakse ja uuendatakse [17].

6 Süsteemi analüüsi tulemused

Peatükis sooritab autor arendatava teenuse Qure Audit analüüsiks valitud talitusviisidele põhinedes süsteemi analüüsi. Autor kirjeldab lühidalt arendatavat teenust ning läheb süvitsi funktsionaalsete ja mittefunktsionaalsete nõuetega.

6.1 Lahenduse kirjeldus

Luua auditlogi (andmemuudatuse kontrolljälje) teenus Qure Audit, mis rakendab sõnumi vahendaja teenust ja on suuteline kliendi (teine mikroteenus) poolt saadetavate kirjete toimetamist vastuvõtjasse asünkroonselt, et salvestada kirje eraldiseisvasse andmebaasi ning omada võimekust kuvada kirje ajalugu kliendile.

Kirjeldusele põhinev teenus lahendab mikrosüsteemi integreeritud auditlogimisel probleemi, kus auditlogimine jääb blokeerima teiste ülesannete sooritamist. Peamine mikroteenus saab loobuda auditlogimisest ja delegeerida ülesande iseseisvale auditlogijale, jätkates teiste sooritamist vajavate ülesannetega. Samuti kasutades eraldi andmebaasi, jääb detailandmete andmebaasi maht ja koormus väiksemaks.

6.2 Funktsionaalsed nõuded

Süsteemi analüüsi algfaasis kogutud nõuete põhjal on autoril võimalik välja tuua kaks viisi, kuidas klient suhtleb teenusega. Nende põhjal kirjutas autor välja kaks detailset kasutuslugu.

Funktsionaalsete nõuetena kogutud nõuded esitab autor kasutusmallidena. Funktsionaalseteks nõueteks said „Auditkirje salvestamine“ (Tabel 1) ja „Kirje auditlogi pärimine“ (Tabel 2). Autor on arvamusel, et just need kaks peamist kasutuslugu katavad ära kliendi poolse suhtluse teenusega ja pakuvad piisavat kirjeldust arendamiseks.

Tabel 1. Kasutusmall UC01 "Auditkirje salvestamine".

Nimetus	**Auditkirje salvestamine**
ID	**UC01**
Kirjeldus	Auditlogimise teenus Qure Audit salvestab kirje
Tegutseja(d)	Klient, Qure Audit, klassifikaatorite teenus
Eeltingimus(ed)	Kliendi poolt saadetav objekt on konfigureeritud Qure Audit teenuses
Põhivoog	1. Klient saadab REST POST päringu kaudu andme muudatuse kirje Qure Audit saatjasse 2. Saatja loob kirje saatmiseks sõnumi ning lisab sõnumisse kirje 3. Saatja saadab sõnumi vahendajasse 4. Vahendaja suunab marsruutimisvõtme põhjal sõnumi järjekorda 5. Vastuvõtja loeb järjekorrast sõnumi 6. Vastuvõtja teisaldab kirje objektiks 7. Vastuvõtja rikastab kirje 8. Vastuvõtja salvestab kirje andmebaasi 9. Vastuvõtja kinnitab sõnumi kätte saamisest
Alternatiivsed vood	Vastuvõtja kirje salvestamise ebaõnnestumise puhul säilib kirje järjekorras ning proovitakse uuesti kohale toimetada Kirje suunamise ebaõnnestumise korral kuvatakse kliendile vastav veateade
Lõpptulemus	Kirje on salvestatud andmebaasi

Tabel 2. Kasutusmall UC02 "Kirje auditlogi pärimine".

Nimetus	**Kirje auditlogi pärimine**
ID	**UC02**
Kirjeldus	Klient pärib kirje põhjal auditlogi
Tegutseja(d)	Klient, Qure Audit
Eeltingimus(ed)	Kliendi poolt päritav kirje ja kirje identifikaator eksisteerib Qure Audit's
Põhivoog	1. Klient saadab REST GET päringu kirje võtmega 2. Qure Audit vastuvõtja suunab päringu vastava kirje teenusesse 3. Kirje teenus pärib andmebaasist kirje 4. Qure Audit edastab kliendile kirje auditlogi
Alternatiivsed vood	Kirje või kirje identifikaatori puudumisel tagastatakse vastav veateade
Lõpptulemus	Klient saab kirje ja kirje identifikaator põhjal auditlogi

6.3 Mittefunktsionaalsed nõuded

Mittefunktsionaalsete nõuete esitamiseks kasutas autor FURPS'i, keskendudes just täpselt mittefunktsionaalsete nõuetele. Autor arvab, et tabelis olev nõuete kogum on teenuse arendamise ja valmimise jaoks ammendav, kuid jätab endale teadmiseks, et nõudeid võib lisanduda ja muutuda (Tabel 3).

Tabel 3. Mittefunktsionaalsed nõuded.

ID	Nõue	FURPS
MFN01	Järjekorrast sõnumeid vastu võttes säilib järjekord	Vastupidavus
MFN02	Vahendajad säilivad sõnumi vahendaja katkestuse korral	Vastupidavus
MFN03	Järjekorrad säilivad sõnumi vahendaja katkestuse korral	Vastupidavus
MFN04	Sõnumid säilivad sõnumi vahendaja katkestuse korral	Vastupidavus
MFN05	Sõnumid on järjekorras püsivad	Vastupidavus
MFN06	Auditlogimise teeenus on skaleeritav	Toetatavus
MFN07	Teenus on eraldi paigaldatav	Toetatavus
MFN08	Järjekorda lisamine ei võta rohkem aega kui 1 sekund	Jõudlus
MFN09	Vastuvõtja lisab kirje järjekorrast andmebaasi vähem kui 1 sekundiga	Jõudlus

7 Lahenduse realiseerimine

Peatüki esimeses pooles põhjendab autor auditlogimise teenuse arendamise jooksul kasutatuid tehnoloogilisi tööriistu. Toob esile ettevõtte poolsete piirangute tõttu teenuses kasutatavad tööriistad ning seejärel kirjeldab autori poolseid tööriistade valikuid, põhjendades nende eelistamist teenuse loomisel analüüsi tulemuste põhjal.

Peatüki teises pooles kirjeldab autor süsteemi analüüsi tulemustena valminud süsteemi arhitektuuri. Selgitab detailselt arendatud süsteemi komponente ja nende ülesandeid. Toob esile sõnumi vahendaja rolli auditlogimise teenuses ja kirjeldab, kuidas sõnumi vahendaja on liidestatud süsteemi kahe eraldiseisva teenuse vahel.

7.1 Tehnoloogilised tööriistad

Alampeatükis kirjeldab ja põhjendab autor auditlogimise teenuse arendamisel kasutatud tehnoloogilisi tööriistu. Autor liigitab tehniliste tööriistade valiku ettevõtte poolseteks ja autori poolseteks valikuteks. Tööriistade valiku tegemisel toetus autor analüüsi tulemusena kirjeldatud süsteemi nõuetele, ettevõtte poolt kasutavatele tööriistadele ja autori poolsetele valikutele.

7.1.1 Ettevõtte poolsed

Ettevõttes kasutatakse erinevate rakenduste ja teenuste arendamisel läbivalt sarnaseid tehnoloogilisi tööriistu. Sellel põhjusel otsustas autor lõputöö käigus arendatavas auditlogimise teenuses kasutada ettevõtte poolt läbivalt kasutatud tehnoloogilisi tööriistu. Auditlogimise teenuses kasutatavad tehnoloogilised tööriistad on vastavuses teenuse süsteemsete nõuetega. Selline lähenemisviis aitab hoida ettevõttes ühtset standardit, dokumentatsiooni ja tavasid. Ühtse standardi hoidmine auditlogimise teenuses aitab autoril efektiivsemalt töötada ning vähendab teenuse arendamise jaoks kuluvat aega ja ressurssi. Järgnevad tehnoloogilised tööriistad on ettevõttes standard ja kasutusele võetud auditlogimise teenuses: Java, PostgreSQL, Spring Boot, Hibernate, Maven ja Docker.

7.1.2 Sõnumite vahendaja

Arendatavas auditlogimise teenuses oli autoril ülesandeks valida sobiv sõnumite vahendaja vaheprogramm, mis täidaks kirjete kohale toimetamise ja säilitamise ülesannet, et oleks võimalik turvaliselt ja kindlalt kirje auditlogimiseks kohale toimetada.

Süsteemi analüüsi põhjal pandi paika peamised mittefunktsionaalse nõuded, millele sõnumite vahendaja auditlogimise teenuse raames peab vastama: sõnumi vahendaja peab olema suuteline säilitama sõnumi katkestuse korral, sõnumi vahendaja peab olema suuteline katkestuse korral säilitama järjekorrad, sõnumid peavad säilitama sisestusel tekkinud järjekorra, sõnumid on järjekorras püsivad ja sõnumi vahendaja peab olema võimalikult kiire.

Sõnumite vahendajana suudavad sõnumit säilitada nii ActiveMQ, RabbitMQ kui ka Kafka. Kõik kolm sõnumite vahendajat pakuvad võimalust hoiustada sõnumeid nii mälus kui kettal. Auditlogimise teenuse jaoks on tähtis hoiustada sõnumeid kettal, et sõnumid suudaksid säilida sõnumi vahendaja katkestuse korral. Sõnumid aga ei ole võimelised säilima, kui sõnumeid hoidev järjekord ei suuda katkestusest taastuda. Järjekordade hoidmisel on kõik kolm sõnumite vahendajat suutelised täitma süsteemi nõudmist.

Sõnumite järjekorra säilitamine on oluline auditlogimisel põhjusel, et andmed kajastaksid täpselt logitavates teenustes tekkinud andmemuudatustest sellises järjekorras nagu muutused esinesid. Sõnumite järjekorda suudavad säilitada kõik kolm sõnumite vahendajat. Sõnumite järjekorras püsimine tagab olukorra, kui sõnumi vastuvõtja poolelt tekib katkestus, säilib sõnum järjekorras, et oleks võimalik tulevikus sõnum uuesti saata, et kindlasti saaks sõnumis olev kirje logitud. Nii RabbitMQ kui ka ActiveMQ täidavad seda nõuet. Kafka garanteerib ainult ühekordset sõnumite kohale toimetamist. Kiiruse poolest on kõige kiirem sõnumite vahendajana Kafka, millele järgneb RabbitMQ ja ActiveMQ [22]. Kuna RabbitMQ täidab kõiki nõudeid ja nõuete täitmise põhjal kõige kiirem, otsustas autor seda sõnumite vahendajat kasutada enda poolt loodavas auditlogimise teenuses.

7.2 Teenuse arhitektuur

Teenuse arhitektuur tekkis süsteemi nõuete ja ettevõtte poolt etteantud tarkvara arhitektuuri tüübi põhjal. Süsteem koosneb kahest eraldi seisvast teenusest, mis on omavahel ühendatud sõnumi vahendajaga, milleks on RabbitMQ. Sõnumeid loov ja RabbitMQ'sse saatev teenus, *producer*, on liidestatud kliendiga kasutades REST API otspunkti, vastu võtmaks kliendi poolt saadetud kirjeid auditlogimiseks. Kliendi kirjeid salvestav pool ehk *consumer* on kliendiga liidestatud auditlogi pärimise eesmärgil. Samuti on liidestatud klassifikaatori teenusega, mille eesmärgiks on küsida andmeid auditkirje rikastamise hetkel (Lisa 2, Joonis 4).

7.2.1 Sõnumi vahendaja

Sõnumi vahendajas otsustas autor sõnumite vahetusprotokollina kasutada AMQP'd. Kuna RabbitMQ kasutab vaikesättena AMQP'd ja Spring pakub valmis kujul liidestust sellele protokollile ja mõlema tehnoloogia põhjal on valmis kirjutatud põhjalik dokumentatsioon [23], siis kiirendab see teenuse arendamist. Vahetustüübina otsustas autor kasutada otsest vahetust. Autor ei soovi kasutada *fanout* vahetust kuna ei ole tarvis kirjete duplikatsiooni. Otsese vahetuse eelis *topic* ja päise põhise vahetuse ees on kiirus, vahetaja ei pea kulutama aega sõnumi marsruutimisele, kuna kasutab ainult ühte marsruutimisvõtit, mis on selgelt etteantud ja ei pea otsima marsruutimisvõtmele vastet. Autor otsustas lisada sõnumi vahendajasse kirjete põhised vahetajad. Autor usub, et see suudab hoida lahus erinevad kirjed, tõstes sellega sõnumi vahendaja hallatavaust ja hõlpsam skaleerida ja liidestada. Teguvuspõhiseid vahetajaid on raskem hallata ja liidestada, kuna puudub täpne ülevaade, millist tüüpi kirjet kandev sõnum vastuvõtja vastu võtab ja kuidas kirjega edasi käituda.

7.2.2 Producer

Kliendid saavad saata kirjeid auditlogimiseks *producer* teenusele. Autor kirjutas *producer*'le kontrolleri, mis omab otspunkti *logMessage*, mis võtab vastu POST päringuid. *Producer* seejärel saadab kirje asünkroonsesse teenusesse *ProducerService*'sse. Teenus kasutab kirjet sisaldava sõnumi loomiseks, vahetaja tüübi saamiseks ja marsruutimisvõtme loomiseks teenust *MessageService*, mis saab kõik vajalikud parameetrid kirjest, mis on esitatud JSON kujul. Seejärel saadab asünkroonselt

ProducerService loodud sõnumi vahetaja tüübi ja marsruutimisvõtme abil sõnumi vahendajasse.

7.2.3 Consumer

Consumer teenuses kirjutas autor vastavalt logitavale kirjele järjekorra kuulajad, *MessageListener*. Järjekorra kuulajad registreeritakse järjekorra nime järgi. Seejärel suunatakse sõnum vastava kirje teenusesse. Kirjele põhinevas teenusele kirjutas autor abistava teenuse *MessageService*, mille ülesandeks on sõnumis olev JSON tüüpi kirje teisaldada POJO'ks. Seejärel päritakse logitava kirje klassifikaatori id põhiselt *ClassifierService* abil kirjele kuuluv klassifikaator ja lisatakse kirjele. Lõpuks salvestatakse kirje objekt detailandmete andmebaasist eraldiseisvasse andmebaasi läbi *EntitiyDao*. Samuti kirjutas autor teenusele kontrolleri, millel on otspunkt *getAuditLog*, mis võtab vastu GET päringuid, mille abil saavad kliendid pärida kirje tüübi põhjal kirjele vastava auditlogi.

8 Tulemus

Peatükis kirjeldab autor lõputöö käigus loodud mikroteenusel põhinevat kontrolljälje teenust. Lisaks toob välja lõputöö tulemused ja järeldused ning kirjeldab lühidalt teenuse tulevikku.

8.1 Loodud lahendus

Loodud lahenduseks on mikroteenusel põhinev auditlogimise teenus Qure Audit, mis koosneb kahest eraldi seisvast teenusest ja sõnumi vahendajast, milleks valiti RabbitMQ. Sõnumi vastuvõttev teenus *Producer* võtab vastavalt rakendamisele vastu kliendi (teine mikroteenus) poolt logimist vajava kirje seisundi ja luues kirjet kandva sõnumi ning vastutab sõnumi kohale toimetamise sõnumi vahendajale. Sõnumi vahendaja võtab vastu saatja poolt saadetud sõnumi ning suunab sõnumi vahetajale, mis on püsiv kliendi auditkirjele spetsiifiline sõnumi suunamise vahend. Vahetaja seejärel suunab sõnumi, vastavalt sõnumi marsruutimisvõtmele ettenähtud vastupidavasse järjekorda. Järjekord hoiustab sõnumit kuni sõnum on valmis vastuvõtja poolt vastu võtmiseks. Vastuvõtja, *Producer*, on Qure Audit teenuse teine osa, mis vastutab sõnumis transporditava kirje andmetega rikastamise ja kirje andmebaasi salvestamise eest. Samuti on vastuvõtja nõue kuvada kliendile ette antud kirje tüübi põhjal kirje terve olemasolev auditlogi.

8.2 Tulemused ja järeldused

Lõputöö tulemuseks on edukalt analüüsitud ja arendatud mikroteenuse arhitektuuriga auditlogimise teenus Qure Audit. Analüüsi tulemusena vastab teenus süsteemi nõuetele, andes ülevaate, mis on lahenduse juures tugevused ja nõrkused. Qure Audit omadused on standardne lahend, mis sobib igale juhtumile. Lahendus soosib auditlogimisel keerukuse eraldamist ja andmebaaside kergendamist. Teenusena valmis prototüüp, mida on võimalik edaspidiselt arendada ja tulevikus ettevõttes kasutusele võtta.

8.3 Lahenduse tulevik

Autor näeb ette tulevikus arendada teenusele juurde kirjete kontroll ja täpsem veateadete haldamine ja kuvamine. Lahenduse edasises arenduses on võimalik lisada teenusele auditlogi pärimises piiranguid nagu kirjete limiit, kirjete nihe, kirjete filtreerimine kuupäevade järgi ja teisi kirjete filtreerimise meetodeid. Samuti on võimalik veel piirata teenusele ligipääsu. Siiski jääb see osa arendustest paigaldaja mureks. Ettevõttes on tulevikus plaan loodud lahendus kasutusele võtta.

9 Kokkuvõte

Mikroteenuste arhitektuur on tänapäeval järjest enam kasutatav infosüsteemides. Mikroteenustele on tihti nõutud auditlogi säilitamine. Ühes infosüsteemis võib olla mitmeid mikroteenuseid, kus mikroteenused peavad säilitama kirjete auditlogi. Kõigile mikroteenustele tuleb rakendada auditlogimise funktsionaalsus, mis tekitab korduvat funktsionaalse arendamist, nõudes lisaaega ja ressurssi.

Lõputöö eesmärgiks oli arendada prototüüp auditlogimise mikroteenus, kasutades sõnumite vahendajaid, mida kliendid (teised mikroteenused) saavad kasutada, saates kirjetes toimunud andmemuudatused salvestamiseks või pärida kirjega seotud auditlogi.

Töös selgitati auditlogimise põhimõtteid ja eesmärke. Seejärel anti ülevaade sõnumitele orienteeritud vahevarast, keskendudes töö kõigus arendatavale teenuses kasutatavale alamliigile, milleks on sõnumite vahendajad. Nende alampeatükis võrreldi erinevaid sõnumite vahendajaid, keskendudes loodava teenuse vajadustele. Analüüsi tulemusena kirjeldati lahendus ning liigitati funktsionaalsed ja mittefunktsionaalsed nõuded. Lahenduse realiseerimises põhjendati tehnoloogiate valikut ja kirjeldati nõuete põhjal loodud teenuse arhitektuuri. Tulemuse peatükis pandi kirja loodud lahendus ja analüüsiti vastavust süsteemi nõuetele, mille põhjal jõuti tulemuste ja järeldusteni. Samuti anti ülevaade lahenduse tulevikust.

Tulemusena valmis süsteemi nõutele vastav prototüüp auditlogimise teenus Qure Audit, mis on standardne lahend, vähendades seega funktsionaalsuse kordumist ja tõstes detailandmeid omava andmebaasi jõudlust ja mahtu.

Kasutatud kirjandus

[1] „What is as Audit Log? Audit Trails and How to Use Audit Logs“, 20.04.2020. [WWW] https://www.dnsstuff.com/what-is-audit-log (25.04.2022).

[2] „Data Enrichment“, *Trifacta*. [WWW] https://www.trifacta.com/7-ways-data-enrichment-boosts-your-business/ (25.04.2022).

[3] C. Richardson, *Microservice patterns*, Shelter Island, New York: Manning, 2018.

[4] E. Curry, “Middleware for Communications“, Q. H. Mahmoud, Ed., Guelph, Kanada: Wiley, 2004.

[5] N. Pubudu, “Introduction to Message Brokers”, *Medium*, 15.05.2021. [WWW] https://medium.com/nerd-for-tech/introduction-to-message-brokers-bb5cacf9c70b (21.04.2022).

[6] “Ultimate Message Broker Comparison”, 05.04.2018. [WWW] https://ultimate-comparisons.github.io/ultimate-message-broker-comparison/ (21.04.2022).

[7] B. Snyder, D. Bosanac, R. Davies, *ActiveMQ in Action*, Stamford, USA: Manning, 2011.

[8] “The Java EE 6 Tutorial”. [WWW] https://docs.oracle.com/javaee/6/tutorial/doc/bncdr.html (21.04.2022).

[9] R. Monson-Haefel, “5 Minutes or less: ActiveMQ with JMS Queues and Topics“, *Tomitribe*, 07.05.2019. [WWW] https://www.tomitribe.com/blog/5-minutes-or-less-activemq-with-jms-queues-and-topics/ (21.04.2022).

[10] “Connectivity”, *Apache ActiveMQ*. [WWW] https://activemq.apache.org/connectivity (21.04.2022).

[11] O. Gabor, “An Introduction to RabbitMQ – What is RabbitMQ?”, *Erlang Solution*, 03.04.2020. [WWW] https://www.erlang-solutions.com/blog/an-introduction-to-rabbitmq-what-is-rabbitmq/ (21.04.2022).

[12] L. Johansson, “Part 1: RabbitMQ for beginners – What is RabbitMQ?“, *CloudAMQP*, 23.09.2019. [WWW] https://www.cloudamqp.com/blog/part1-rabbitmq-for-beginners-what-is-rabbitmq.html (21.04.2022).

[13] N. Narkhede, R. Sivaram, T. Palino, G. Shapira, *Kafka: The Definitive Guide, 2nd Edition*, Sebastopol, USA: O’Reilly Media, 2021.

[14] IBM Cloud Education, “Apache Kafka”, *IBM*, 18.02.2020. [WWW] https://www.ibm.com/cloud/learn/apache-kafka#:~:text=Apache%20Kafka%20(Kafka)%20is%20an,time%2C%20event%2Ddriven%20applications (21.04.2022).

[15] „Introduction“, *Apache Kafka*. [WWW] https://kafka.apache.org/intro (21.04.2022).

[16] M. McCormick, *Waterfall vs. Agile Methodology*, MPCS, 09.08.2012. [WWW] http://mccormickpcs.com/images/Waterfall_vs_Agile_Methodology.pdf (25.04.2022).

[17] J. Satzinger, R. Jackson, S. Burd, *Systems Analysis and Design in a Changing World*, Boston, United States: Cengage Learning, 2015.

[18] F. Paetsch, A. Eberlein, F. Maurer, „Requirements engineering and agile software development," *Wet Ice 2003. Proceedings. Twelfth IEEE International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises,* Linz, Austria: IEEE, 15. sept. 2003, doi: 10.1109/ENABL.2003.1231428.

[19] N. Daly, "What Is a Use Case?", 19.01.2021. [WWW] https://www.wrike.com/blog/what-is-a-use-case/ (25.04.2022).

[20] M. Rehkopf, „User stories with examples and a template", *Atlassian*. [WWW] https://www.atlassian.com/agile/project-management/user-stories (25.04.2022).

[21] „Nonfunctional Requirements", 10.02.2021. [WWW] https://www.scaledagileframework.com/nonfunctional-requirements/ (25.04.2022).

[22] S.N. Raje, „Performance Comparison of Message Queue Methods". [WWW] https://digitalscholarship.unlv.edu/cgi/viewcontent.cgi?article=4749&context=thesesdissertations (25.04.2022)

[23] M. Pollack *et al*., „Spring AMQP," *Spring*. [WWW] https://docs.spring.io/spring-amqp/docs/current/reference/html/ (25.04.2022).

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

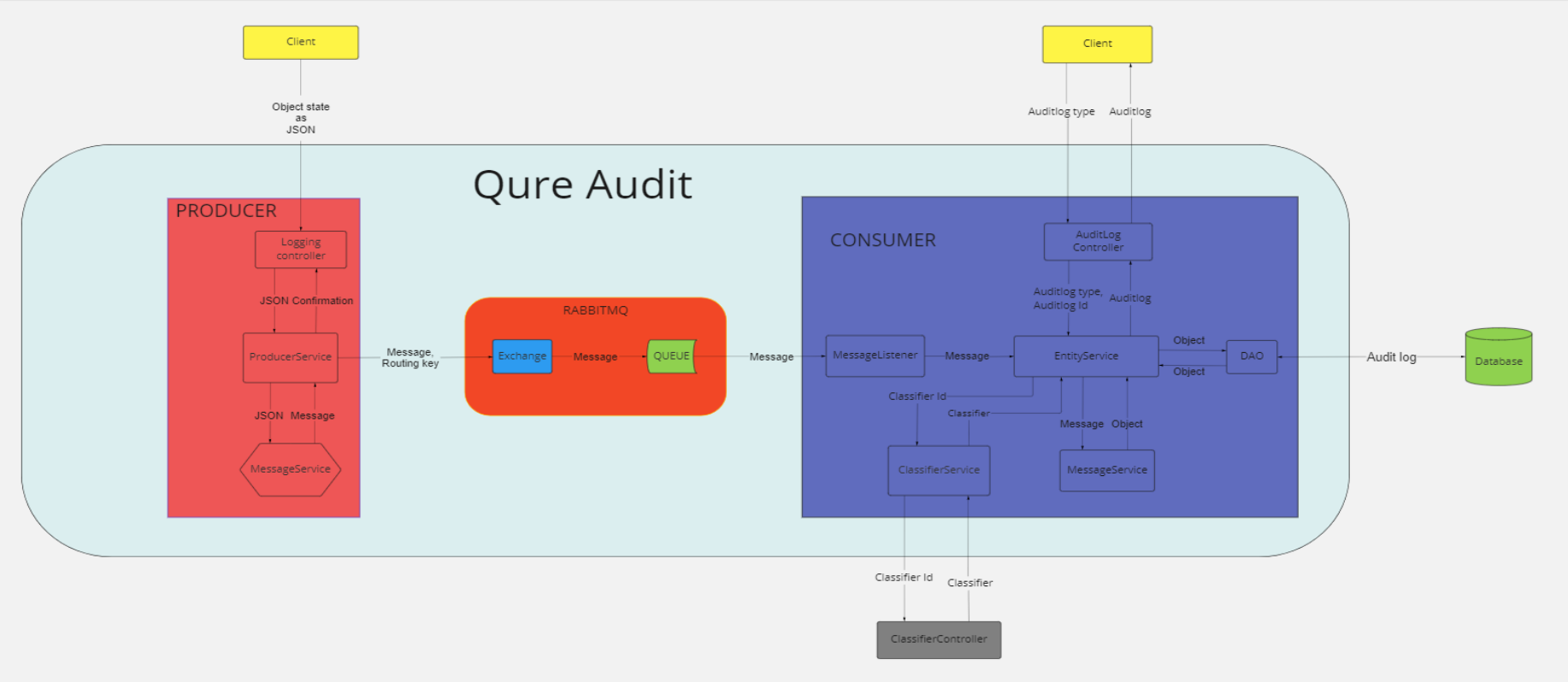
Mina, Kristjan Kattus

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Andmemuudatuste kontrolljälje teenus mikroteenuste arhitektuuriga“, mille juhendajad on Margus Jäger ja Mart Simisker
   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 - Loodud süsteemi arhitektuur



Joonis 4. Loodud süsteemi arhitektuur