TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Tarkvarateaduse instituut

Tanel Tõemets 155605IABB

INTEGRATSIOONIPLATVORMIDE VÕRDLEV ANALÜÜS JA VALIK LOGISTIKA PLUSS OÜ NÄITEL

Bakalaureusetöö

Juhendaja: Inna Švartsman
MSc
Lektor

Toomas Mändla
MSc
IT Juht

Tallinn 2018

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: Tanel Tõemets

20.05.2018

Annotatsioon

Käesoleva bakalaureusetöö eesmärkideks on leida integratsiooniplatvorm, mis sobiks hästi Logistika Pluss OÜ klienditellimuste vastuvõtmiseks ja töötlemiseks mõeldud süsteemi arhitektuuri korrastamiseks ning pakkuda osaliselt realiseeritud integratsiooniplatvormi, mis on sisendiks edasisel integratsiooniplatvormi kasutuselevõtul Logistika Pluss OÜ-s. Eesmärk tuleneb probleemidest Logistika Pluss OÜ klienditellimuste vastuvõtmiseks ja töötlemiseks mõeldud süsteemi arhitektuuris.

Eesmärgi saavutamiseks on analüüsitud Logistika Pluss OÜ klienditellimuste vastuvõtmiseks ja töötlemiseks mõeldud süsteemi arhitektuuri. Toetudes teoreetilistele allikatele on uuritud teenustele orienteeritud arhitektuuri ja integratsiooniplatvormide rolli selles. Probleemi iseloomust tulenevalt on põhjalikumaks tutvumiseks valitud Zato ja Azure Service Bus integratsiooniplatvormid. Mõlemal platvormil on realiseeritud dokumendivahetuse teenusepakkuja poolt saabuva tellimuse vastuvõtmine. Saadud kogemuste põhjal on koostatud platvormide võrdlus.

Bakalaureusetöö esimese tulemusena jõuti võrdleva analüüsi ning kaalutud mitme kriteeriumi põhise hindamismeetodi abil otsusele, et Logistika Pluss OÜ jaoks hästi sobiv integratsiooniplatvorm on Zato. Bakalaureusetöö teiseks tulemuseks on integratsiooniplatvormide kasutamist kirjeldav ja platvorme võrdlev materjal ning osaliselt realiseeritud ja konfigureeritud Zato platvorm, mis on sisendiks platvormi kasutuselevõtul Logistika Pluss OÜ-s.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 35 leheküljel, 7 peatükki, 13 joonist, 2 tabelit.

Abstract

Comparative Analysis and Choice of Integration Platform on the Example of Logistika Pluss OÜ

The aim of this bachelor's thesis is to find integration platform that is well-suited for correcting and improving the architecture of Logistika Pluss OÜ client orders receiving and processing system. Another aim is to offer partly set up integration platform that will be the input for further deployment of integration platform in Logistika Pluss OÜ.

The main problems caused by insufficiency of current architecture are following: updating some components of Logistika Pluss OÜ system is complicated, integrating Logistika Pluss OÜ system with new clients is sophisticated and inadequate orders are not checked automatically.

To achieve the goals of this thesis the architecture of receiving and processing client orders system was analysed. In addition, Service Oriented Architecture as well as integration platform place in a Service Oriented Architecture was studied with the help of theoretical materials. Due to the nature of the problem and authors wish to use different approaches to the problem two significantly different integration platforms were chosen. These platforms were Zato and Azure Service Bus. Receiving a client order was implemented with both platforms. Gathered experience was used to form comparison of these two platforms.

The first result of this thesis is the choice of Zato integration platform for Logistika Pluss OÜ. The choice was made using Multi-Factor Scoring Method and experience gathered from implementing client order receiving functionality with both platforms. The second result of this thesis is the material that was gathered to give overview of Azure Service Bus and Zato integration platforms and partly set up and configured Zato platform that will be the input for further deployment of the integration platform in Logistika Pluss OÜ.

The thesis is in Estonian and contains 35 pages of text, 7 chapters, 13 figures, 2 tables.

Lühendite ja mõistete sõnastik

AMQP	*Advanced Message Queuing Protocol*, protokoll sõnumitele orienteeritud vahevarale
API	*Application Programming Interface*, rakendusliides
CSV	*Comma Separated Values*, komaeraldusega väärtused
ESB	*Enterprise Service Bus*, standarditele toetuv integratsiooniplatvorm
FTP	*File Transfer Protocol*, failiedastusprotokoll
HTTP	*HyperText Transfer Protocol*, hüperteksti edastusprotokoll
IT	*Information Technology*, infotehnoloogia
JMS	*Java Message Service*, Java sõnumiteenus
JSON	*JavaScript Object Notation*, andmevahetusvorming
MQ	*Message Queue*, sõnumite riviloend
MS SQL	*Microsoft SQL Server*, Microsofti relatsiooniline andmebaasisüsteem
NAV	*Microsoft Dynamics Navision*, Microsofti majandustarkvara millega juhitakse ja hallatakse ettevõtte tegevust
REST	*Representational State Transfer*, tarkvaraarhitektuuri stiil
SDK	*Software Development Kit*, tarkvaraarenduskomplekt
SOA	*Service Oriented Architecture*, teenustele orienteeritud arhitektuur
SOAP	*Simple Object Access Protocol*, lihtne objektipöördusprotokoll
SQL	*Structured Query Language*, struktuurpäringukeel
URL	*Uniform Resource Locator*, universaalne ressursilokaator
XML	*Extensible Markup Language*, laiendatav märgistuskeel

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Logistika Pluss OÜ on logistikateenuse pakkumisega tegelev Eesti ettevõte. Kaasaegse logistikaettevõttena kasutatakse töö korraldamisel mitmeid IT süsteeme. Käesolevas bakalaureusetöös keskendutakse neist klienditellimuste vastuvõtmiseks ja töötlemiseks mõeldud süsteemile.

Bakalaureusetöö eesmärk on analüüsi ja võrdluse abil leida integratsiooniplatvorm, mis sobiks hästi Logistika Pluss OÜ klienditellimuste vastuvõtmiseks ja töötlemiseks mõeldud süsteemi arhitektuuri parandamiseks ning pakkuda osaliselt realiseeritud integratsiooniplatvormi, mis oleks sisendiks edasisel integratsiooniplatvormi kasutuselevõtul Logistika Pluss OÜ-s.

Klienditellimuste haldamiseks kasutab Logistika Pluss OÜ Microsoft Dynamics NAV majandustarkvara, millega on otse liidestatud suur hulk süsteeme. Sellest tulenevalt on probleemiks uute klientidega liidestuste loomise keerukus ja sisse tulevate tellimuste kontrollimine. Lisaks on keeruline ka Microsoft Dynamics NAV versioonivärskendamine, sest uue versiooni paigaldamisel tuleks kõik liidestused uuesti realiseerida. Integratsiooniplatvormi kasutuselevõtt aitab nimetatud probleeme lahendada. Seejuures on oluline valida ettevõtte vajaduste ning soovidega sobiv platvorm.

Bakalaureusetöö ülesehitus on järgmine: esmalt on antud ülevaade Logistika Pluss OÜ klienditellimuste vastuvõtmiseks ja töötlemiseks mõeldud süsteemi praegusest arhitektuurist, probleemist ja lahendusest. Seejärel on teoreetilistele allikatele toetudes uuritud teenustele orienteeritud arhitektuuri, et anda ülevaade peamistest põhimõtetest mida rakenduste integreerimisel tihti kasutatakse. Järgmisena on läbi proovitud tellimuse vastuvõtmist Zato ja Azure Service Bus integratsiooniplatvorme kasutades. Mõlemast platvormist ning nende kasutamisest tellimuse vastuvõtmiseks on antud põhjalik ülevaade. Saadud kogemustele toetudes ning Logistika Pluss OÜ jaoks oluliste näitajate põhjal on koostatud eelnimetatud integratsiooniplatvormide võrdlus. Viimasena on toodud järeldused. Järeldustes tuuakse välja analüüsi ja võrdluse tulemusena valitud integratsiooniplatvorm, mis sobib hästi klienditellimuste vastuvõtmiseks ja töötlemiseks

mõeldud süsteemis esinevate probleemide lahendamiseks. Platvormi valikul on kasutatud kaalutud mitme kriteeriumi põhist hindamismeetodit.

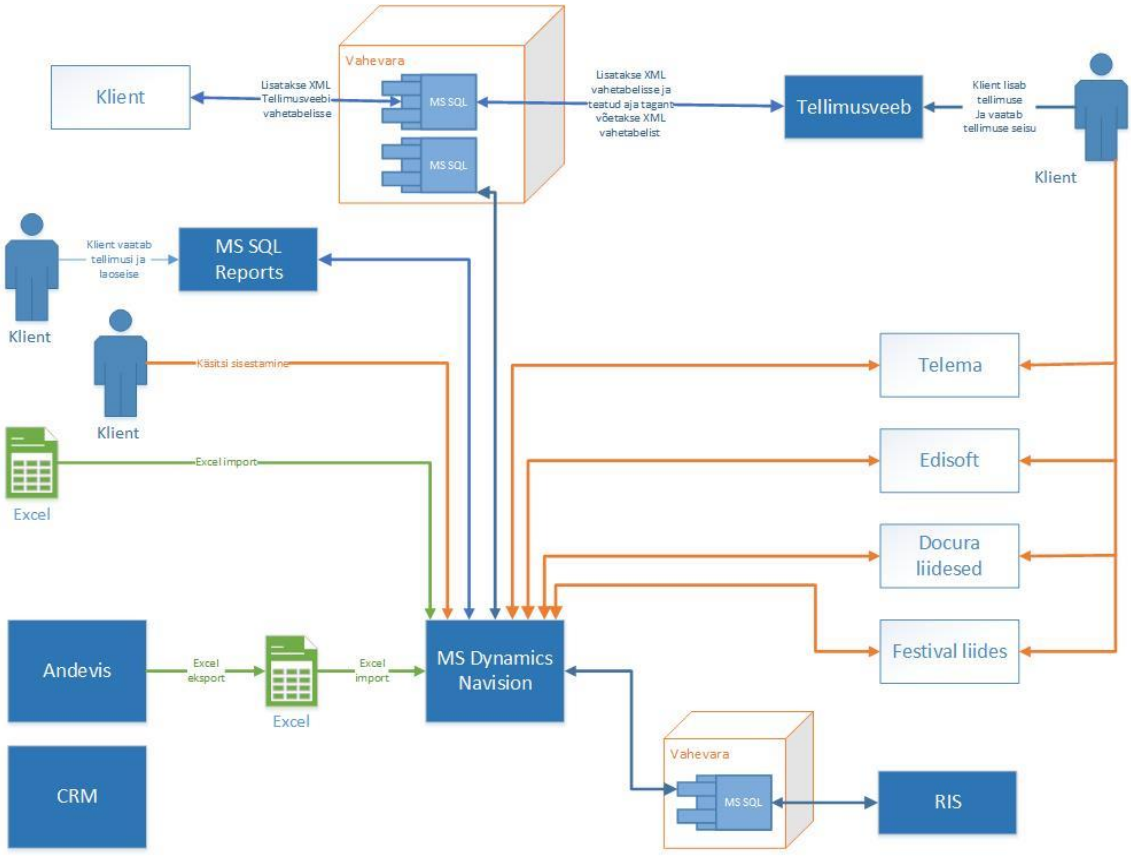
Bakalaureusetöö pakub peamiselt väärtust Logistika Pluss OÜ-le, andes ülevaate nende klienditellimuste vastuvõtmiseks ja töötlemiseks mõeldud süsteemist ning pakkudes konkreetse integratsiooniplatvormi näol välja lahenduse nimetatud süsteemi arhitektuuri parandamiseks. Lisaks leidub kindlasti teisigi ettevõtteid kelle süsteemides esineb sarnaseid arhitektuurilisi probleeme ning kellele käesolevas töös olevast informatsioonist kasu võib olla. Töös toodud põhjalik ülevaade Zato ja Azure Service Bus platvormidest on kasuks nii nendega tutvumisel kui ka platvormide kasutuselevõtul.

2 Logistika Pluss OÜ süsteemi arhitektuur

Käesolevas töös on vaatluse all Logistika Pluss OÜ klienditellimuste vastuvõtmiseks ja töötlemiseks mõeldud süsteemi arhitektuur. Antud peatükis on toodud üldisel tasemel ülevaade süsteemi praegusest arhitektuurist. Lisaks on välja toodud praegusest arhitektuurist tulenevad puudused, võimalik meetod puuduste lahendamiseks ning süsteemi parandatud arhitektuur. Arhitektuurist ülevaate andmisel toetus autor Logistika Pluss OÜ IT juhi Toomas Mändla poolt esitatud süsteemi kirjeldustele.

2.1 Praegune olukord ja probleem

Alljärgnevalt (Joonis 1) on esitatud praegune Logistika Pluss OÜ klienditellimuste vastuvõtmiseks ja töötlemiseks mõeldud süsteemi arhitektuur.



Joonis 1. Logistika Pluss OÜ arhitektuur.

Vaatluse all oleva süsteemi (Joonis 1) keskseks osaks on Microsofti majandustarka NAV (*Microsoft Dynamics Navision*). NAV-is algatatakse erinevad tehingud, näiteks ostutellimused ja müügitellimused. NAV-iga on seotud suur hulk liidestusi. Peamiselt on süsteemi klientidega sidumiseks kasutatud dokumendivahetusplatvorme nagu Telema ja Edisoft, kuid leidub ka erilahedusi. Erilahendused on realiseeritud näiteks Docura ja Festivali liideseid kasutades. Selgitused kõigi joonisel olevate komponentide kohta on esitatud tabelina lisas 1.

Kirjeldatud arhitektuuri probleem seisneb selles, et välised klientide rakendused on läbi erilahenduste või dokumendivahetuse teenusepakkujate nagu Telema ja Edisoft otse seotud NAV-iga. Välistest liidestustest jõuavad kõik tellimused kõigepealt MS SQL (*Microsoft SQL Server*) vahetabelisse, kus toimub saabunud tellimuse kontroll. Juhul kui tellimus on korrektne liigub see edasi kinnitatud tellimuste tabelisse, kuid kui tellimuse info on vigane märgitakse see vigaseks ning sellega peab tegelema hakkama kliendihaldur. Nii vahetabelit kui ka kinnitatud tellimuste tabelit saab vaadata ja hallata NAV-i vahendusel. Üks praeguse lahenduse puudustest seisneb selles, et vigase tellimuse parandamiseks peab kliendihaldur ühendust võtma tellijaga. Sellele kuluvat ressurssi oleks aga võimalik kokku hoida, kui andemete kontroll toimuks süsteemi poolt varem ning klient saaks koheselt sellekohase veateate.

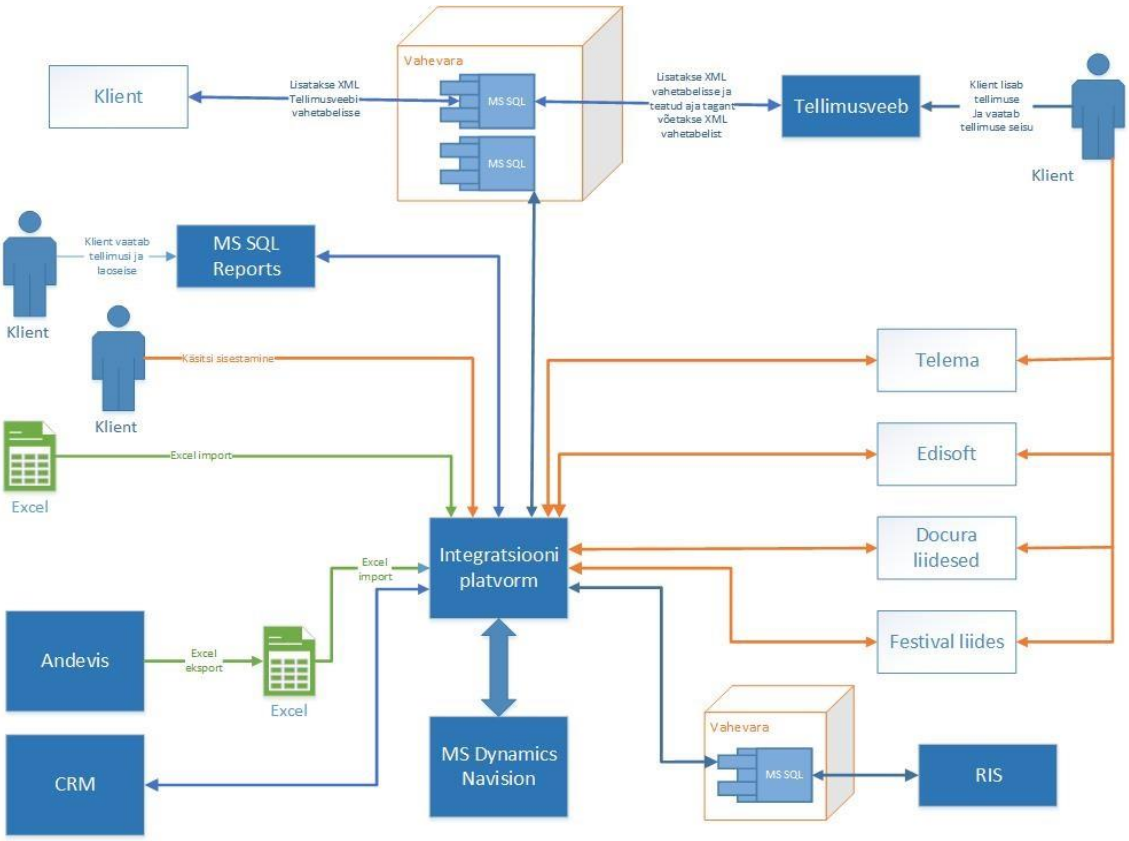
Otsene seotus väliste rakendustega muudab ka NAV-i versiooni uuendamise ning uute klientidega liidestuste loomise Logistika Pluss OÜ jaoks keerukaks. Hetkel on kasutusel Microsoft Dynamics NAV 2009 R2 versioon, mis on tulevikus plaanis uuema vastu välja vahetada. Uuele verisoonile üle minnes tuleks NAV uuesti otse liidestada kõigi väliste rakendustega, mis aga oleks ettevõtte jaoks põhjendamatult kulukas. Lisaks ei arvestaks kõnealune lahendus tulevaste versioonivärskendustega ning uute klientidega liidestuste loomine oleks Logistika Pluss OÜ jaoks endiselt kulukas.

2.2 Tulevik

Süsteemide integreerimine on keeruline ülesanne ning leidub palju erinevaid integratsioonistiile, mida arhitektuuri parandamisel kaaluda. Peamiseid integreerimisstiile on neil. Failiedastus (*File Transfer*) ja jagatud andmebaasi (*Shared Database*) meetodid võimaldavad rakendustel vahetada andmeid. Et aga võimaldada ka keerukamaid protsesse ning tagada süsteemi parem kohanemisvõime ajas muutuvate

äriliste nõuetega on parem kasutada protseduuri kaugkäivitus (*Remote Procedure Invocation*) või sõnumivahetus (*Messaging*) integratsioonistiile. Protseduuri kaugkäivituse stiili puuduseks on sõnumiside aeglus, kehv töökindlus ning asjaolu, et rakenduste sõltuvus üksteisest on kõrge. Sõnumivahetus integratsioonistiil võimaldab aga luua pakettide sagedase, kohese, usaldusväärse ning asünkroonse edastamise ning seejuures on võimalik kasutada ka mitmesuguseid andmeformaate. Seega võib sõnumivahetus integratsioonistiili pidada parimaks lahenduseks. Kogu sõnumiside korraldamine sõnumivahetus integratsioonistiili puhul on eraldi komponendi (*Message Bus*) ülesanne [1].

Tulenevalt eelnevast on uue arhitektuuri puhul hea valik sõnumivahetus integratsioonistiil. Sõnumiside korraldamiseks on mõistlik kasutusele võtta valitud integratsioonistiili realiseerimist võimaldav integratsiooniplatvorm. Leidub palju erineva funktsionaalsuse ja ülesehitusega integratsiooniplatvorme, mistõttu on oluline valida just Logistika Pluss OÜ võimalustele ja vajadustele vastav. Sobiva integratsiooniplatvormi valik on käesoleva bakalaureusetöö üheks eesmärgiks. Alljärgneval joonisel (Joonis 2) on toodud integratsiooniplatvormi abil täiendatud arhitektuur.



Joonis 2. Logistika Pluss OÜ parendatud arhitektuur.

Parandatud arhitektuur (Joonis 2) lahendab varem välja toodud arhitektuuri ülesehitusest tingitud probleemid. Uue lahenduse puhul ei jõuaks vigased tellimused enam NAV-i ning kliendihaldur ei pea nendega tegelema. Arhitektuurimuudatuste tulemusena käivad kõik tellimused enne NAV-i jõudmist läbi integratsiooniplatvormi. Enne NAV-i jõudmist kontrollib integratsiooniplatvorm tellimuse korrektsust ning vaid korrektsed tellimused saadetakse edasi. Vigase tellimuse puhul saadetakse kliendile sellekohane teade ning palutakse tellimust parandada või täiendada. Uuele arhitektuurile üle minnes saab ka NAV-i uuendamine ning uute klientidega liidestuste loomine olema Logistika Pluss OÜ jaoks lihtsam ning vähem ressursse nõudev, sest kõik komponendid pole enam otse NAV-iga seotud.

3 Teenustele orienteeritud arhitektuur ja ESB

Varasemalt kirjeldatud sõnumivahetus integratsioonistiil on lai mõiste ning pakub vaid üldiseid põhimõtteid. Et anda põhjalikum ülevaade headest integratsioonitavadest on käesolevas peatükis käsitletud teenustele orienteeritud arhitektuuri ehk SOA-d (*Service Oriented Architecture*). SOA keskendub samuti suures osas sõnumivahetusele, kuid pakub autori hinnangul lisaks ka häid arhitektuurilisi soovitusi mida süsteemide integreerimisel järgida tuleks. Sellest tulenevalt on järgnevalt toodud ülevaade teenustele orienteeritud arhitektuuri põhimõtetest ning integratsiooniplatvormi rollist selles. Teenustele orienteeritud arhitektuuri tundmine ja selle põhimõtete järgimine aitab saavutada hästi hallatavat süsteemide arhitektuuri. Integratsiooniplatvormi teenustele orienteeritud arhitektuuris nimetatakse ESB-ks (*Enterprise Service Bus*).

Lühidalt öelduna kujutab SOA endast arhitektuuri ärirakenduste ehitamiseks ning ESB on üks selle peamistest komponentidest. ESB ülesandeks on saata sõnumeid SOA eri komponentide vahel [2].

3.1 Teenusele orienteeritud arhitektuur

Teenustele orienteeritud arhitektuur ehk SOA on arhitektuur ärirakenduste ehitamiseks. SOA puhul koosneb süsteem komponentidest ning igat komponenti võib vaadelda kui „musta kasti.“ Komponentidest koosneva süsteemi töö tulemuseks on terviklik teenus, mille eesmärgiks on täita mingit äriprotsessi [2].

„Musta kasti“ komponentidest koosnemine tähendab, et SOA püüab saavutada madalat sõltuvust. Madal sõltuvus on ka üks väga oluline osa SOA arhitektuurist. Ühtlasi on madal sõltuvus ka üldteada tarkvara hea disaini põhimõte, mille kasutamine kõrgemal abstraktsioonitasemel ehk süsteemide integreerimisel toob kasu samamoodi nagu näiteks objektorienteeritud programmeerimisprojektis [2].

Täpsemalt tähendab SOA puhul madal sõltuvus, et üks rakendus ei pea teadma kuidas teine rakendus töötab või kuidas ta üles on ehitatud. Näiteks oletame, et üks komponent süsteemis saadab teisele mingeid andmeid sisaldava päringu. Teine komponent täidab

päringu ning vajaduse korral tagastab päringu tulemuse päringu teinud komponendile. Iga komponent pakub teisele väikest valikut lihtsatest teenustest. Seejuures on rõhk lihtsusel ja autonoomsusel. Mingi hulk madala kokkukuuluvusega komponente suudavad ära teha sama töö, mida teeb üks keeruka struktuuriga rakendus. Eri komponentidest koosneva süsteemi eelis aga seisneb selles, et süsteemi muutmine on tunduvalt lihtsam ja kiirem kui ühe keeruka rakenduse korral ning seega on komponentidest koosneva süsteemi lahendust kasutava ettevõtte süsteemide struktuur palju paindlikum. Mõistetavalt võimaldab nimetatu ettevõttele ka kulude kokkuhoidu, sest süsteemide hoolduskulud on väiksemad ning uute funktsionaalsuste lisamine samuti soodsam. Paindlikkus ja kiirem reageerimisvõime muutuvatele oludele võivad kaasa tuua ka konkurentsieelise ning seeläbi aidata kaasa ettevõtte edenemisele [2].

Kokkuvõtvalt võib öelda, et SOA abil on võimalik luua mingi hulga omavahel seotud komponentide abil küllaltki lihtne süsteem, mis suudab läbi viia väga keerukaid äriprotsesse. Küll aga ei sobi kõnealune arhitektuur igasuguste rakenduste ehitamiseks vaid on suunatud ärirakendustele [2].

3.1.1 SOA kui vahend äri ja IT koostoimeks

Teenustele orienteeritud arhitektuuri tuleks silmas pidada nii süsteemide ehitamisel kui ka äriliste otsuste langetamisel. Kui SOA põhimõtteid juba ärilisi otsuseid langetades arvesse võtta on tulemuseks äriprotsessid, mida tehnoloogia toetab. Seega on võimalik palju efektiivsemalt ja kiiremini äriprotsesse automatiseerida ning nende täitmiseks vajalikke IT süsteeme üles ehitada. Selleks, et SOA saaks edukalt toimida peavad ettevõtte juhid ja IT osakonnad tegema tihedat koostööd äriprotsesside kindlaks määramisel [2].

Kokkuvõtvalt on SOA poolt ärile toodud kasu järgmine: ettevõtte ärilisi eesmärke täitvate IT süsteemide ehitamine ja kasutuselevõtt muutub lihtsamaks ja kiiremaks, äril on võimalik keskenduda ärile ning IT-l on soodne keskkond arenemiseks ja muutustega sammu pidamiseks [2].

3.1.2 SOA kui vahend äriliste teenuste taaskasutamiseks

Lisaks eespool mainitule on üks peamisi SOA kasutamisest ettevõtte jaoks saadav kasu see, et olemasolevaid rakendusi taaskasutatakse. SOA koosnemine eelpool nimetatud „musta kasti“ komponentidest teeb taaskasutamise võimalikuks [2].

Paljudes suurtes ettevõtetes tekitavad probleeme sarnased programmid, mis realiseerivad sarnaseid äriprotsesse. Tihti juhtub, et mõni ettevõtte osakondadest otsustab oma äriprotsesse muuta ning seetõttu arendatakse kõnealuse osakonna jaoks uus tarkvara, mis tegelikkuses on väga suures osas olemasolevate programmidega identne. Seega võib samasugusest tarkvarast ühes ettevõttes olla palju väikeste erinevustega versioone. Sellised väikeste erinevustega versioonid tarkvarast põhjustavad selle, et ettevõtte IT süsteemide haldamine muutub väga keerukaks. Väike muudatus ärireeglites võib põhjustada suuri probleeme, kuna muudatused tuleb teha paljudes eri kohtades [2].

SOA aitab sellise olukorra tekkimist vältida. Olulisi äriprotsesse vaadeldakse SOA arhitektuuris kui teenuseid. Näiteks võib teenuseid luua tellimuse koostamise või reserveerigu koostamise protsesside läbiviimiseks. Teenuse all mõeldakse siinkohal mingit konkreetset äriprotsessi kirjeldavat programmikoodi. Kõnealune äriprotsess võib olla seotud teiste äriprotsessidega. Teenust on võimalik taaskasutada kogu organisatsiooni erinevates rakendustes. Seega pole muudatused ärireeglites enam probleemiks kuna muudatus tuleb teha vaid ühes kohas [2].

Tarkvara taaskasutamise idee ei ole uudne. Näiteks on suur rõhk koodi taaskasutamisel objektorienteeritud programmeerimiskeeltes ning ka paljude muude praktikate puhul on taaskasutamine tähtsal kohal. SOA erinevus seisneb selles, et ta aitab taaskasutada ärilisi teenuseid. Enam pole vaatluse all madala astme taaskasutatavad komponendid vaid taaskasutatakse kogu äriprotsessi kirjeldavat teenust [2].

3.2 ESB

ESB (*Enterprise Service Bus*) on standarditele toetuv integratsiooniplatvorm, mis ühendab endas intelligentset marsruutimist, sõnumivahetust, veebiteenuseid ja andmete töötlust ühest märgistuskeelset teise. ESB eesmärk on usaldusväärselt siduda suur hulk mitmesuguseid organisatsiooni ja tema äripartnerite rakendusi ning koordineerida nende rakenduste omavahelist suhtlust tagades seejuures transaktsioonide terviklikkus [3].

3.2.1 ESB koht teenustele orienteeritud arhitektuuris

ESB kujutab endast teenustele orienteeritud arhitektuuri ehk SOA realiseerimise peamist komponenti. ESB ülesandeks teenustele orienteeritud arhitektuuris on saata sõnumeid teenusetele orienteeritud arhitektuuri eri komponentide vahel [3].

SOA realisatsioonide puhul kasutatakse peaaegu alati mõnd ESB platvormi. Küll aga tuleb silmas pidada, et SOA järgimine ei tähenda tingimata ESB kasutamist ning ESB kasutamine ei tähenda, et ettevõtte IT arhitektuur järgib tingimata SOA põhimõtteid. Samas on ESB kasutamine mõistlik ja hea vahend SOA realiseerimiseks ning selleks, et komponendid saaksid omavahel sõnumeid vahetada [2].

4 Integratsiooniplatvormide kasutamine

Käesolev peatükk annab ülevaate Zato ja Azure Service Bus integratsiooniplatvormidest. Nimetatud platvormide valikul võttis autor arvesse ka Logistika Pluss OÜ poolt tehtud eelvalikut. Zato platvorm võeti vaatluse alla kuna tegu on avatud lähtekoodiga modernse lahendusega, mis võimaldab realiseerida juba pikalt kasutatud ning ennast tõestanud teenustele orienteeritud arhitektuuri. Microsofti poolt pakutav Azure Service Bus ostus aga valikuks kuna Logistika Pluss OÜ kasutab ka mitmeid teisi Microsofti tooteid. Sama pakkuja integratsiooniplatvormi kasutamine võib seega olla mugavam ja efektiivsem, kui mõne teise pakkuja lahenduse valimine. Lisaks oli oluline, et platvormid pakuksid probleemile erinevat lähenemist ning võimaldaksid ellu viia eelnevalt kirjeldatud teenustele orienteeritud arhitektuuri või sõnumiside integratsioonistiili.

Mõlemast platvormist on antud ülevaade ning seejärel on läbi proovitud Telema (dokumendivahetuse teenusepakkuja) vahendusel saabunud XML (*Extensible Markup Language*) kujul tellimuse vastuvõtmist FTP (*File Transfer Protocol*) ühenduse kaudu. Telema struktuuriga näidistellimus on toodud lisas 2.

4.1 Zato

Zato puhul on tegu teenustele orienteeritud arhitektuuri järgiva ja avatud lähtekoodiga ESB ning rakenduste serveriga. Peamiseks kasutatavaks programmeerimiskeeleks on Python [4].

Zato koosneb paljudest eri komponentidest mille ülesandeid tuleb antud ESB süsteemi kasutamisel tunda. Alljärgnevalt on seega toodud lühike ülevaade olulisematest Zato't puudutavatest mõistetest.

Klastri all mõeldakse Zato puhul serverite gruppi. Serverites jooksevad sisemised ja kasutaja defineeritud teenused. Klastris olevate serverite arv pole limiteeritud, küll aga peab igasse klastrisse kuuluma vähemalt üks server. Kõikides serverites jooksevad alati samad teenused ehk kõik teenused on dubleeritud [5]. Igas klastris peab olema oma Redise võti-väärtus andmebaas [6].

Teenuseks nimetati eelpool kirjeldatud teenustele orienteeritud arhitektuuri puhul mingit olulist äriprotsessi, mida kirjeldas programmikood [2]. Täpsemalt on teenus programmikood, mis kirjeldab äriprotsessi süsteemi ning temaga integreeritud süsteemi vahel. Zato puhul kirjutatakse teenused programmeerimiskeeles Python kasutades Zato pakutud API-t (*Application Programming Interface*) [5]. Antud platvormi dokumentatsiooni põhjal saab autori hinnangul öelda, et väga suurt rõhku on pandud eelpool kirjeldatud teenustele orienteeritud arhitektuuri põhimõtetele.

4.1.1 Zato keskkonna üles seadmine

Zato'st ülevaate saamiseks on see installeeritud Ubuntu 16.04 LTS versioonile. Lisaks Zato'le on installeeritud ka Redis võti-väärtus andmebaas. Redis't kasutab Zato mitmesuguste andmete hoidmiseks, mis oma struktuurilt ei sobi relatsioonilisele andmemudelile toetuvasse andmebaasisüsteemi. Näiteks hoitakse seal andmeid statistika ning jagatud lukkude kohta [6]. Antud realisatsiooni puhul on kasutatud Zato versiooni 2.0.8 ning Redise versiooni 4.0.7.

Kogu Zato keskkonda ehk nii Zato platvormi kui ka esimest klastrit on võimalik üles seada kasutades Docker'i paigaldustarkvara [7]. Teine võimalus, mida on kasutatud ka antud realisatsioonis, on jooksutada käsureal alljärgnevat käsku (Joonis 3). Käsk on mõeldud juba installeeritud Zato keskkonda esimese Zato klastri kiireks loomiseks. Enne kõnealuse käsu jooksutamist peaks kindlasti olema paigaldatud ja käivitatud ka eelpool mainitud Redise andmebaas (*redis-server* käsuga). Operatsiooniliste andmete jaoks kasutab Zato vaikimisi SQLite andmebaasisüsteemi, mille jaoks eraldi üles seadmine pole vajalik kuna SQLite paigaldatakse koos Zato'ga. Soovi korral saab kasutada ka MySQL, PostgreSQL või Oracle andmebaasisüsteemi, kuid nende kasutamisel tuleb teostada täiendavat konfigureerimist [8].

```
zato@tanel-VirtualBox:~$ zato quickstart create ~/env/qs-1 sqlite
localhost 6379 \ --kvdb_password ’’ --verbose
```

Joonis 3. Zato komponentide loomise lause [8].

Joonisel (Joonis 3) toodud käsuga luuakse kõik tervikliku Zato keskkonna loomiseks vajalikud komponendid etteantud asukohta (antud juhul /env/qs-1). Määratakse ka ära operatsiooniliste andemete hoidmiseks kasutatav andmebaasisüsteem. Võimalikud variandid on „mysql,“ „oracle,“ „postgresql,“ ja „sqlite.“ Järgmisena määratakse ära

Redis andmebaasisüsteemi host ja port (antud juhul localhost ja 6379). Viimasena on vaja sisestada Redise parool. Antud juhul on parool jäetud tühjaks kuna vaikimisi seadetes pole Redisele parooli määratud ning antud realisatsioonis ongi kasutatud Redise vaikimisi seadeid. Lisaks antud lauses kasutatule on võimalik kõnealuses käsus konfigureerida veel mitmeid parameetreid, näiteks kui kasutatakse PostgreSQL andmebaasisüsteemi saab määrata millist skeemi kasutada [8]. Kõik erinevad võimalused on toodud Zato dokumentatsioonis.

Eelneva käsu tulemusena luuakse klaster koos järgmiste komponentidega (Joonis 4).

```
[1/8] Certificate authority created
[2/8] ODB schema created
[3/8] ODB initial data created
[4/8] server1 created
[5/8] server2 created
[6/8] Load-balancer created
Superuser created successfully.
[7/8] Web admin created
[8/8] Management scripts created
Quickstart cluster quickstart-257279 created
Web admin user:[admin], password:[ihen-neni-home-rbac]
Start the cluster by issuing the /opt/zato/env/qs-1/zato-qs-start.sh
command
Visit https://zato.io/support for more information and support
options
```

Joonis 4. Zato komponendid.

Sellega on Zato keskkond koos kõigi sinna kuuluvate komponentidega loodud. keskkonna käivitamiseks tuleb jooksutada skripti zato-qs-start.sh. Skripti asukoht on näha ka eelpoolt toodud joonisel (Joonis 4). Skripti jooksutamisel käivitatakse kõik klastri komponendid, mis on vajalikud Zato keskkonna kasutamiseks (Joonis 5).

```
zato@tanel-VirtualBox:~/env/qs-1$ ./zato-qs-start.sh
Starting the Zato cluster quickstart-257279
Running sanity checks
[1/6] Redis connection OK
[2/6] SQL ODB connection OK
[3/6] Load-balancer started
[4/6] server1 started
[5/6] server2 started
[6/6] Web admin started
Zato cluster quickstart-257279 started
Visit https://zato.io/support for more information and support
options
```

Joonis 5. Zato komponentide käivitamine.

Nagu joonisel (Joonis 5) näha loodi ühendus Redise võti-väärtus andmebaasisüsteemiga ja operatsiooniliste andmete hoidmiseks mõeldud SQL (*Structured Query Language*) andmebaasisüsteemiga. Lisaks pannakse tööle ka koormuse tasakaalustaja (*Load-balancer*), kaks serverit ning veebihalduskeskkond (*Web-admin*).

Koormuse tasakaalustaja peab olemas olema igas klastris ning selle ülesandeks on jaotada sissetulevat liiklust võrdselt kõigi serverite vahel [9]. Veebihalduskeskkond kujutab endast brauserist kasutatavat graafilise kasutajaliidesega töövahendit administraatorile, mis võimaldab hallata olemasolevaid klastreid. Näiteks saab veebihaldus keskkonnas luua ühendusi (*Connections*), kanaleid (*Channels*) ning laadida serverisse üles kasutaja defineeritud teenuseid [10]. Veebihalduskeskkonna kasutajaliidest võib autori hinnangul pidada küllatki vanamoeliseks, kuid selle olemasolu muudab mitmete toimingute toestamise tunduvalt mugavamaks.

Oluline Zato omadus on ka kohene kasutuselevõtt (*Hot-deployment*). See tähendab, et enamus tegevusi on Zato's võimalik teha ilma taaskäivitust sooritamata. Muudatused rakendatakse koheselt ning muudatused sünkroniseeritakse automaatselt klastris olevates serverites [5]. Palju kasu toob see omadus näiteks teenuste kasutuselevõtul, mille toestamiseks pakub Zato kaht võimalust.

Esimene võimalus on laadida teenuse koodi sisaldav Python'i fail Zato installeerimisel loodud spetsiaalsesse kausta, kuhu kopeeritud failid koheselt kasutusele võetakse ja teenusena kättesaadavaks muutuvad. Selle kausta nimi on „pickup-dir“ ning see asub iga serveri kaustas. Pole vahet millist serveri kausta kasutada, sest muudatused sünkroniseeritakse kõigis serverites [11].

Teine võimalus teenuse kättesaadavaks tegemiseks on kasutada veebihalduskeskkonna kasutajaliidest [11]. Viimane on mõnevõrra mugavam eelpool kirjeldatud käsurea abil faili kopeerimisest.

4.1.2 XML kujul tellimuse vastuvõtmise realisatsioon

Et saada põhjalik ülevaade Zato võimalustest ning pakkuda antud töö näol sisendit konkreetse integratsiooniplatvormi kasutuselevõtuks Logistika Pluss OÜ-s on läbi

proovitud XML failide vahendamine kasutades Zato't. Täpsemalt on proovitud läbi Telema (dokumendivahetuse teenusepakkuja) vahendusel saabunud XML kujul tellimuse vastuvõtmist FTP ühenduse kaudu. Telema näidistellimus on toodud lisas 2.

XML failiga tegeleva teenuse kirjutamiseks on Zato puhul erinevaid võimalusi. Näiteks on olemas võimalus nimega SimpleIO. Antud lahendus võimaldab arendada teenuseid mida saab taaskasutada paljude eri rakenduste jaoks kuna andmete formaat pole kindlaks määratud. Seega on SimpleIO abil võimalik arendada teenuseid, mida saab kasutada samade andmete eri formaadis edastamiseks üle erinevate transpordi protokollide. Näiteks on võimalik sama teenust kasutades edastada andmeid eri rakendustele: ühele rakendusele XML formaadis ja AMQP (*Advanced Message Queuing Protocol*) protokolli kasutades ning teisele JSON (*JavaScript Object Notation*) formaadis HTTP (*HyperText Transfer Protocol*) protokolli kasutades [12]. Antud realisatsioonis on kasutatud Zato tavapärast XML failidele teenuste kirjutamise funktsionaalsust, mis võimaldab korraga sõnumeid saata vaid ühte andmeformaati ja protokolli kasutades [13]. Järgnevalt (Joonis 6) on toodud lihtne teenus Telema tellimuse vahendamiseks.

```
# Zato
from zato.server.service import Service

class GetOrderDetails(Service):
    def handle(self):
        conn = self.outgoing.ftp.get('TelemaOrders')
        contents = conn.getcontents('/orders/order-IR475639.xml')
        self.logger.info(contents)
```

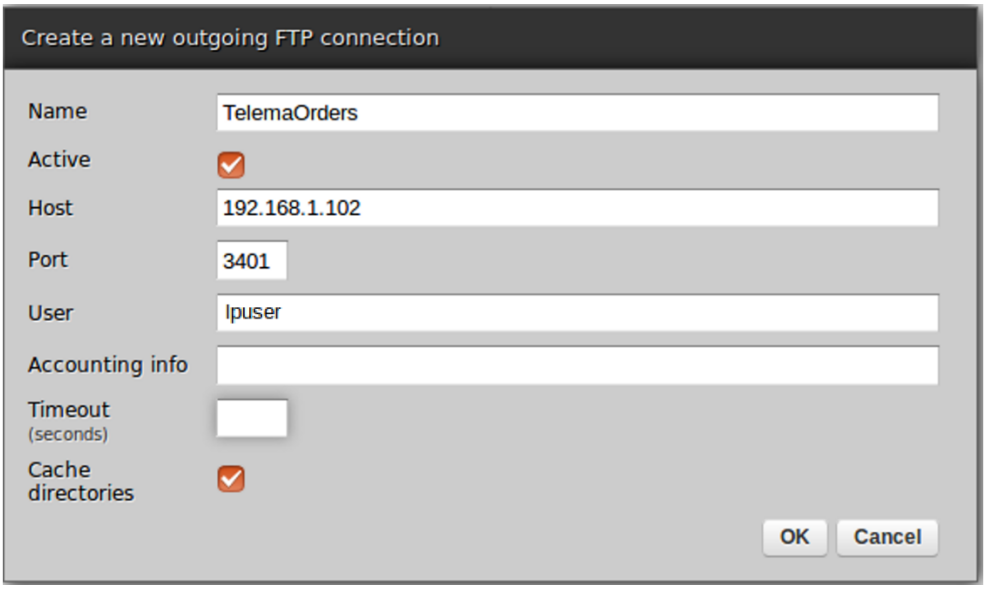
Joonis 6. Telema teenus Zato's [13].

Teenus (Joonis 6) on kirjutatud Pythonis ning kasutab Zato API-t. Kõigepealt luuakse FTP ühendus *TelemaOrders* nimelise väljuva FTP ühendusega (*Outgoing connection*). Seejärel määratakse kust tuleb võtta andmed. Viimaks antakse saadud andmed väljundisse. Selleks, et teenus muutuks kasutatavaks tuleb see kopeerida mõne serveri spetsiaalsesse kausta (Joonis 7), kust teenus koheselt kasutusele võetakse ning info uue teenuse kohta koheselt ka kõigi teiste serveritega sünkroniseeritakse [11].

```
tanel@tanel-VirtualBox:~$ sudo cp telema_service.py
/opt/zato/env/qs-1/server1/pickup-dir
```

Joonis 7. Teenuse kasutuselevõtt Zato's [11].

Selleks, et kõnealust teenust oleks nüüd võimalik ka kasutada on vaja luua eelpool koodis (Joonis 6) toodud (*TelemaOrders* nimeline) väljuv FTP ühendus (*Outgoing connection*), mis loob ühenduse Telema dokumendivahetusplatvormiga. Selleks, et Zato platvorm ja Telema saaksid sõnumeid vahetada on vaja luua ka nende vaheline kanal (*Channel*) [11]. Telema dokumendivahetusplatvormiga suhtlemiseks sobib FTP protokolli kasutav väljuv ühendus. FTP sobib ka enamuste teiste välisete süsteemidega ühenduse loomiseks Logistika Pluss OÜ-s. Seetõttu ongi järgmiseks toodud FTP väljuva ühenduse loomine (Joonis 8). Samas võiks ettevõte tulevikus kaaluda ka teiste protokollide kasutusele võtmist, mis pakuvad Zato platvormi puhul rohkemaid võimalusi. Üleminekuperioodil oleks hea kasutada eelpool kirjeldatud SimpleIO vahendit.

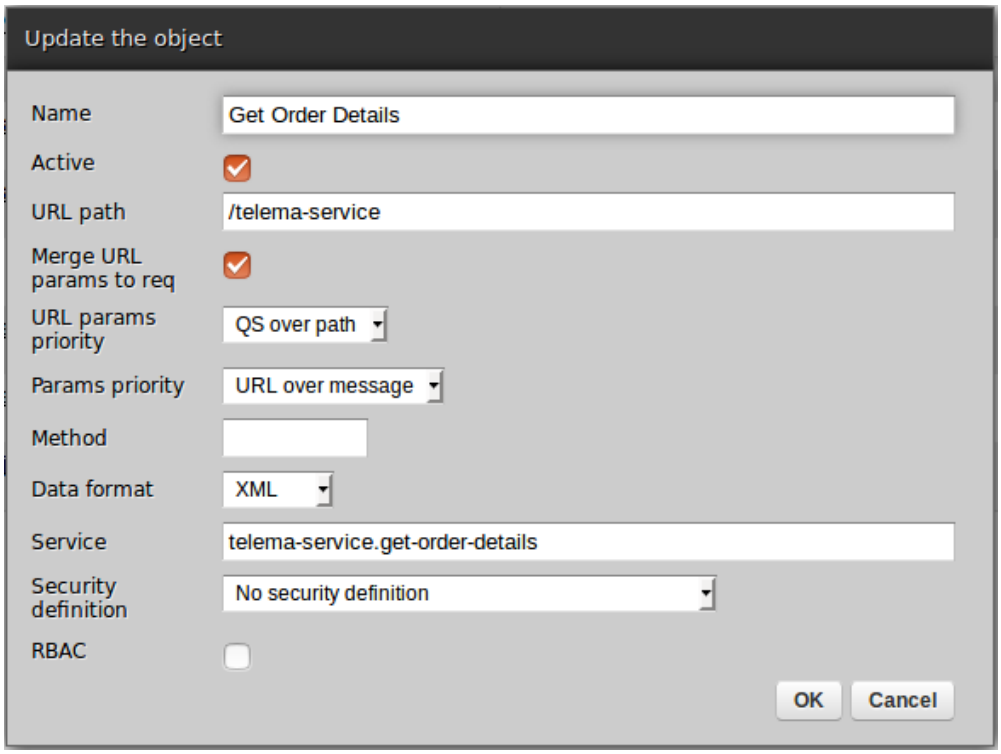


Joonis 8. FTP ühenduse loomine Zato's.

Jooniselt (Joonis 8) on näha, et FTP ühenduse loomiseks on vaja määrata ühenduse nimi (*Name*). Nime kasutatakse teenuse koodis ühenduse loomiseks. Aktiivsuse (*Active*) valiku lahtris saab määrata kas teenus saab ühendust kasutada või mitte. Host lahtris tuleb määrata FTP serveri host ning port lahtris port millega ühendamine toimub. Lisaks saab määrata kasutajanime (*User*), arvepidamis infot (*Accounting Info*), aegumis aega (*Timeout*) ja viimase valikuna ka vahemälu kasutamist. Parooli FTP ühenduse esmakordsel loomisel koheselt määrata ei saa [14]. Selleks tekib veebihaldus keskkonnas võimalus kohe peale uue ühenduse loomist.

Lisaks FTP ühendusele pakub Zato muidugi ka mitmeid teisi võimalusi väljuva ühenduse (*Outgoing connection*) loomiseks mingi välise süsteemi või rakendusega. Ühendada on võimalik kasutades AMQP, FTP, JMS (*Java Message Service*) WebSphere MQ (*Message Queue*), Plain HTTP ehk REST (*Representational State Transfer*), Odoo ehk OpenERP, SOAP (*Simple Object Access Protocol*), SQL ja ZeroMQ protokolle [15].

Viimaks tuleb luua ka kanal (*Channel*) mis ühendaks omavahel Telemaga loodud FTP ühenduse ja teenuse. Ka selle loomine käib veebihaldus keskkonnas [11]. Kanaleid on erinevaid, täpsemalt pakub Zato võimalusi AMQP, JMS WebSphere MQ, Plain HTTP, SOAP ja ZeroMQ protokolli kasutavate kanalite loomiseks. Lisaks aktsepteerivad teenused ka päringuid mis on tulnud plaanuri (*Scheduler*) vahendusel [16]. Alljärgnevalt on toodud HTTP kanali loomine veebihalduskeskkonnas (Joonis 9).



Joonis 9. HTTP kanali loomine Zato's.

Jooniselt (Joonis 9) on näha, et kanali loomiseks tuleb määrata kanali nimi (*Name*). Aktiivsuse (*Active*) lahtris on võimalik määrata, kas kanal on hetkel aktiivne ja kasutatav. Järgmisena saab määrata URL (*Uniform Resource Locator*) tee (*URL path*), vaatluse all oleva kanali ühendamiseks. Teises linnukesega märgitavas lahtris (*Merge URL params to req*) on võimalik määrata, kas tuleks võtta arvesse ka URL-is olevaid parameetreid. Kaks järgmist valikut võimaldavad määrata kuidas süsteem käitub parameetritega

erinevates olukordades (*URL params priority ja Params priority*). Meetodi lahtris saab määrata millise HTTP meetodi puhul saab kõnealust kanalit kasutada. Kui lahter tühjaks jätta saab kasutada kõiki meetodeid. Järgmisena saab määrata saadetavate andmete formaadi (*Data format*). Teenuse (*Service*) lahtrist tuleb määrata millist olemasolevat teenust päringu saabudes välja kutsuda. Turvalisuse kirjelduste lahtris (*Security definition*) saab määrata millega turvalisust tagatakse. Näiteks pakub Zato lahendusi HTTP Basic Auth ja OAuth. Viimasena saab määrata kas kasutada juurdepääsu võimaldamisel erinevate õigustega rolle (*RBAC ehk Role-Based Access Control*) [17].

Väljuvat ühendust ja kanalit nimetatakse koos konnektoriks (*Connector*). Kui konnektoris kasutada HTTP, FTP ja SQL protokolle luuakse rakenduste vahel otseühendus. AMQP, JMS WebSpahere MQ ja ZeroMQ protokollide kasutamisel saab luua asünkroonse ühenduse. Viimase puhul suhtlevad teenused ja ühendused omavahel läbi Redise (*Redis broker*). Näiteks kui kanalisse jõuab sõnum siis kõigepealt jõuab see Redis võti-väärtus andmebaasi. Sealt võtab selle vastu üks serveritest ning teenus kutsutakse välja [18].

Teenust peab olema võimalik NAV-ist välja kutsuda. Zato ühendamisel NAV-iga tuleks arvestada, et NAV peaks iga teatud aja tagant tegema päringuid, et uued saabunud tellimused kokku koguda. Hetkel on testimise eesmärgil kasutatud Curl tarkvara, et kutsuda välja Telema teenust (Joonis 10).

```
curl localhost:11223/telema-service -d '<root/>'
```

Joonis 10. Zato's loodud FTP ühenduse testimine Curl'iga [11].

Eeltoodud väljakutse tulemusena jõuab päring teenuseni. Teenus loob FTP ühenduse Telemaga ning kogub sealt varem defineeritud HTTP kanali kaudu andmed. Andmed esitatakse päringu tulemuseks XML dokumendina.

4.2 Azure Service Bus

Azure näol on tegemist Microsofti poolt pakutava vahendiga, mis pakub IT lahenduste arendajatele suurt hulka erinevaid pilveteenuseid rakenduste ehitamiseks, haldamiseks ja kasutuselevõtuks. Azure Service Bus on üks neist pakutavatest pilveteenustest [19].

Täpsemalt on Azure Service Bus puhul tegu Microsofti poolt pakutava sõnumite asünkroonseks saatmiseks mõeldud ning pilves asuva platvormiga, mis võimaldab vahetada sõnumeid eri süsteemide vahel [20].

Microsoft ei nimeta oma sõnumiedastamise süsteemi ESB-ks (*Enterprise Service Bus*) nagu Zato vaid ütleb selle kohta lihtsalt *Service Bus.* Ärile ja ettevõtlusele viitava osa (*Enterprise*) oma nimest eemaldamise üheks põhjuseks võib olla see, et ESB süsteeme seostatakse tugevalt eelpool kirjeldatud SOA ehk teenusetele orienteeritud arhitektuuriga. Teenustele orienteeritud arhitektuuri põhimõtteid pole autori hinnangul Azure Service Bus'i puhul järgitud ning vahendi mitmekülgsus muudab teenustele orienteeritud arhitektuuri realiseerimise keerukaks. Samas on vahendi mitmekülgsus ning võimalus kasutada ka teisi Azure pilveteenuseid mõningate projektide jaoks positiivne ning vahend sobib hästi sõnumiside integratsioonistiili järgmiseks. Ühe näitena on dokumentatsioonis välja toodud, et Azure Service Bus'i kasutusalaks võib olla kodumasinate, sensorite ja teiste seadmete ühendamine keskse rakendusega [20]. Seega on see sobilik vahend ka asjade interneti loomiseks.

Tulenevalt eesmärgist pakkuda erinevate olukordade jaoks erinevaid sõnumite saatmise võimalusi pakub Azure Service Bus kolme lahendust: riviloendit (*Queue*), teemapõhist riviloendit (*Topic*) ja releed (*Relay*). Nende täpsem kirjeldus on toodud järgnevalt.

4.2.1 Riviloend (*Queue*)

Riviloendid võimaldavad ühesuunalist sõnumivahetust saatja ja vastuvõtja vahel. Saatja saadab sõnumi riviloendisse ning vastuvõtja võib selle vastu võtta kohe või ka mingil muul ajal. Vastuvõtja pidev kättesaadavus pole oluline kuna sõnumit säilitatakse riviloendis kuni see vastuvõtja poolt loetud on. Riviloendisse saabunud sõnumid loetakse sealt välja saabumise järjekorras ehk esimesena saabunud sõnum loetakse välja esimesena. Riviloendil võib olla üks või mitu vastuvõtjat, kuid juhul kui neid on mitu loetakse sõnum vaid ühe vastuvõtja poolt. Selleks, et sõnum jõuaks mitmesse vastuvõtjasse tuleks kasutada teemapõhist riviloendit [21].

Vastuvõtjal on sõnumi vastuvõtmisel kaks võimaliku toimimise viisi. Vaikimisi võimaluse puhul mille nimetuseks on *ReceiveAndDelete* kustutatakse sõnum koheselt peale vastuvõtmist. Selliselt toimimise puudseks on võimalus, et vastuvõtjas toimub sõnumi töötlemise ajal krahh ning sõnum läheb seetõttu kaduma. Teine võimalus mille nimetus on *PeekLock* lahendab selle probleemi. Sõnumi kohese kustutamise asemel see lukustatakse teiste vastuvõtjate jaoks. Juhul kui sõnumi töötlemine õnnestub ning vastuvõtjalt tuleb sellekohane teade (kutsutakse välja *Complete()*) kustutatakse sõnum riviloendist. Juhul kui vastuvõtja mõistab, et sõnumi töötlemine ebaõnnestub saadab ta sellekohase teate (kutsutakse välja *Abandon()*) ning sõnumilt vabastatakse lukk ja teised vastuvõtjad saavad seda uuesti vastu võtta. Juhul kui vastuvõtja ei saada teatud aja jooksul (vaikimisi 60 sekundit) riviloendisse sõnumit siis vabastatakse lukk automaatselt [21].

4.2.2 Teemapõhine riviloend (*Topic*)

Ka teemapõhised riviloendid võimaldavad ühesuunalist liiklust saatja ja vastuvõtja vahel. Erinevus võrreldes riviloendiga seisneb selles, et teemapõhise riviloendiga on võimalik ühendada mitmeid eri süsteeme ehk tegu on üks mitmele seosega. Ka riviloendis võis olla mitu vastuvõtjat, kuid juhul kui neid oli mitu loeti sõnum vaid ühe vastuvõtja poolt. Seega sai riviloendiga siduda vaid ühe süsteemi. Teemapõhise riviloendi puhul jõuab sõnum kõikidesse vastuvõtjatesse ning seega saab seda kasutada mitme süsteemi ühendamiseks. Teemapõhise riviloendi puhul jõuavad kõikidesse eri süsteemidesse samad andmed [21].

Teine oluline teemapõhise riviloendi erinevus on võimalus soovi korral andmeid filtreerida. Sellisel juhul jõuavad vastuvõtjasse vaid need andmed, mis filtri põhjal sobilikud olid [21].

Muus osas on teemapõhine riviloend sarnane riviloendiga. Sõnumeid hoitakse alles seni kuni vastuvõtja selle vastu võtab. Seega ei pea sarnaselt riviloendiga vastuvõtja pidevalt kättesaadav olema. Ka sõnumi vastuvõtmisel toimimise viisid on riviloendiga identsed ehk kasutatakse eelpool kirjeldatud *ReceiveAndDelete* ja *PeekLock* võimalusi [21].

4.2.3 Relee (*Relay*)

Relee sobib saatja ja vastuvõtja vahel otseühenduse loomiseks. Sõnumeid on võimalik saata mõlemas suunas. Seega on mõlemad osapooled nii saatja kui ka vastuvõtja rollis. Erinevalt riviloendist ja teemapõhisest riviloendist sõnumeid ei säilitata vaid võetakse koheselt vastu [21].

Kõnealune lahendus on sobiv kui rakenduste ühendamine muul viisil on liialt keerukas. Näiteks kui soovitakse ühendada kahe eri ettevõtte süsteeme, kuid mõlemad süsteemid asuvad turvalises võrgus ning otseühenduse loomine rakenduste vahel on keerukas [21].

4.2.4 Ülevaade maksumusest

Antud realisatsioonis on Azure Service Bus'iga tutvumiseks kasutatud tasuta kasutajat, mis sobib küll testimiseks ning tutvumiseks, kuid lahenduse lõplikul kasutuselevõtul tuleb arvestada, et tegu tasulise teenusega.

Azure Service Bus'i pakutakse kahel peamisel hinnatasemel, milleks on Standard hinnatase ja Premium hinnatase [22]. Lisaks on olemas ka väheste võimalustega primitiivne hinnatase (*Basic tier*) [23].

Azure Service Bus kasutab maksustamisel kahte mõõdikut. Esimeseks mõõdikuks on operatsioonid. Operatsiooni all mõeldakse kõiki Azure Service Bus API vahendusel riviloendi või teemapõhise riviloendi poole tehtud pöördumisi. Seega on maksustatud lisaks sõnumite saatmisele ja vastuvõtmisele ka haldamis operatsioonid (näiteks riviloendi loomine ja kustutamine) ning ka seisundi oleku päringud [22].

Teine maksustatavaks mõõdikuks on pidevalt ühenduses olevate ühenduste arv. Pidevalt ühendatud on need vastuvõtjad ja/või saatjad, kes kasutavad ühendamiseks AMQP protokolli või HTTP protokolli, mille aegumise (*timeout*) väärtus on märgitud nulliks [22].

Järgnevalt on esitatud ülevaade peamistest Azure Service Bus hinnatasemetest (Tabel 1).

Tabel 1. Azure Standart ja Premium hinnatasemete võrdlus [23].

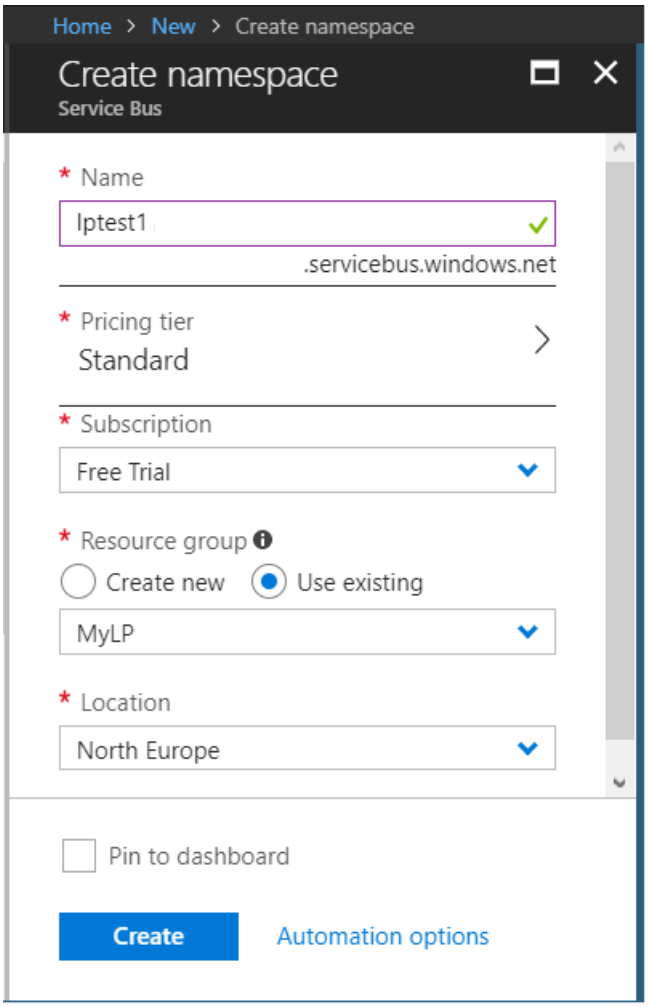
Mõõdik	**Standard hinnatase**	**Premium hinnatase**
Baashind	8,433 eurot kuus	18.78 eurot päevas
Operatsiooni hind	13 miljonit operatsiooni kuus kuuluvad baashinna sisse. Järgmised 87 miljonit operatsiooni maksavad 0.675 eurot miljoni operatsiooni kohta. Järgmised 2400 miljonit operatsiooni 0.422 eurot miljoni operatsiooni kohta ja üle 2500 miljoni operatsiooni 0.169 eurot miljoni operatsiooni kohta.	Kuuluvad baashinna sisse
Püsivad ühendused (*Brokered connections*)	Tuhat ühendust kuus kuuluvad baashinna sisse. Järgmised 99 tuhat ühendust maksavad 0.026 eurot ühenduse kohta. Järgmised 400 tuhat ühendust 0.022 eurot ühenduse kohta ja üle 500 ühenduse 0.013 eurot ühenduse kohta.	Kuuluvad baashinna sisse

Nagu tabelist (Tabel 1) näha pakub Standard hinnatase astmelist hinnastamist ehk mida rohkem operatsioone või püsivaid ühendusi tehakse seda suurem on pakutav soodustus. Premium hinnataseme puhul kuulub kõik baashinna sisse. Seejuures on oluline, et Premium hinnataseme puhul pakutakse ressursside isoleerimist ehk isoleeritud arvutusvõimsust ja mäluruumi. Igat sellist konteinerit nimetatakse sõnumiside üksuseks (*messaging unit*). Tabelis toodud Premium hinnataseme hind kehtib ühe sõnumiside üksuse kohta. Võimalik on osta ka mitu üksust ühte nimeruumi ning siis on ka teenuse hind selle võrra kallim. [22].

Eelpool mainitud primitiivsel hinnatasemel maksab miljon operatsiooni 0.043 eurot. Sõnumiside üksuseid primitiivsel hinnatasemel luua pole võimalik. Releed on maksustatud tunni ja sõnumi põhiselt. Sada tundi relee kasutamist maksab 0.085 eurot ning iga saadetud kümme tuhat sõnumit maksab 0.009 eurot [23].

4.2.5 Azure Service Bus keskkonna üles seadmine

Azure Service Bus-i kasutamiseks tuleb kõigepealt logida sisse Microsoft Azure portaali (https://portal.azure.com) ning teha kaasaegsest kasutajaliidesest valik Service Bus-i loomiseks [24]. Seejärel kuvab süsteem akna nimeruumi loomiseks (Joonis 11).



Joonis 11. Azure Service Bus nimeruumi loomise kuvatõmmis.

Avanenud aknas tuleb sisestada soovitud nimeruumi (*Name*) unikaalne nimetus. Seejärel tuleb valida sobiva hinna ja võimalustega pakett (*Pricing tier*). Lisaks tuleb valida ka tellimus (*Subscription*). Tellimuse valimise võimalus on kasulik kui kasutajal on mitmeid erinevaid tellimusi. Näiteks erinevate projektidega seotud tellimused. Järgmiseks tuleb valida olemasolev või luua uus ressursirühm (*Resource group*) mille sisse paigutub varem määratud nimeruum. Viimasena tuleb määrata asukoht (*Location*) kus äsjaloodud nimeruumi majutama hakatakse [24]. Sellega on nimeruum loodud. Ühtegi rakendust Azure Service Bus'i kasutamiseks installeerida pole vaja kuna teenus asub pilves.

Azure Service Bus-i kasutamiseks pakutakse mitmeid võimalusi. Windowsi rakenduste puhul on võimalik kasutada Microsofti poolt pakutavat spetsiaalset API-t. Lisaks pakub Microsoft SDK-d (*Software Developer Kit*) ka paljudele programmeerimiskeeltele. Dokumentatsioonis on juhendid riviloendite, teemapõhiste riviloendite ja releede

kasutamiseks .NET, Java, Node.js, PHP, Python ja Ruby programmeerimiskeeltes ning lisaks on võimalik kasutada REST API-t ja HTTP(S) protokolli [24].

Käesolevas töös on Azure Service Bus'i kasutamine realiseeritud Javas. Javas Azure Service Bus-i kasutamiseks tuleb enne alustamist installeerida Microsofti poolt pakutav Azure SDK (*Software Development Kit*) Java rakendustele. Kui kasutada Eclipse programmeerimiskeskkonda on võimalik sellele lisada Azure tööriistakomplekt (*Azure Toolkit for Eclipse*), mis sisaldab kohe ka Azure SDK-d Java rakendustele [25].

4.2.6 XML kujul tellimuse vastuvõtmise realisatsioon

Sarnaselt eelpool kirjeldatud Zato realisatsioonile on ka Azure Service Bus-iga läbi proovitud Telema (dokumendivahetuse teenusepakkuja) vahendusel saabunud XML kujul tellimuse vastuvõtmist FTP ühenduse kaudu. Telema näidistellimus on toodud lisas 2. Realisatsiooni koostamisel on toetutud peamiselt Azure Service Bus-i dokumentatsioonis toodud materjalidele ja koodinäidetele [25].

Antud realisatsiooni puhul on kasutatud riviloendit. Esialgu oleks riviloendi kasutamine Logistika Pluss OÜ jaoks sobiv. Pikemas perspektiivis oleks ettevõtte jaoks mõistlikum valik siiski teemapõhine riviloend, kuna need võimaldavad kasutada filtreid ning toetavad paremini süsteemide arengut ajas. Näiteks kui soovitakse sõnumeid saata korraga mitmesse süsteemi. riviloendite kasutamise üks põhjustest käesolevas analüüsis oli Microsofti poolsed piirangud Azure tasuta kontole. Küll aga on riviloendi olemus ja kasutamine väga sarnane teemapõhise riviloendiga ning nad sobivad hästi käesoleva töö eesmärgiga saada ülevaade Azure Service Bus-i poolt pakutavatest võimalustest ja sobivusest Logistika Pluss OÜ-le.

Antud realisatsiooni puhul on kasutatud Java programmeerimiskeelt. Programmikoodi redigeerimiseks on kasutatud Eclipse programmeerimiskeskkonda koos sellele lisatud Azure tööriistakomplektiga (*Azure Toolkit for Eclipse*).

Azure Service Bus-iga ühenduse loomiseks tuleb määrata ära konfiguratsioon (Joonis 12), mida edaspidi kasutatakse *ServiceBusContract* objektis Azure pilveteenusega ühenduse loomiseks. *ServiceBusContract* on ka ainus klass, mida Azurega suhtlemiseks on vaja kasutada [25].

```
package ee.lp.com;

import com.microsoft.windowsazure.Configuration;
import
com.microsoft.windowsazure.services.servicebus.ServiceBusConfiguration;

public class AzureConnectionConfig {

	public Configuration createConfig(){

		Configuration config =
			ServiceBusConfiguration.configureWithSASAuthentication(
				"lptest1",
				"RootManageSharedAccessKey",
				"G70/dC3gqEgPQFCZ64KtnPOPBka0q3Nps/QSoBIezlQ=",
				".servicebus.windows.net"
				);

		return config;
	}
}
```

Joonis 12. Azure Service Bus-i konfiguratsioon Javas [25].

Konfiguratsioonis (Joonis 12) tuleb määrata varem loodud nimeruum (*lptest1*), võtme tüübi nimetus (*RootManageSharedAccessKey*) ja peamine ühendamise võti (*Primary Connection String*). Nendest osadest pannakse kokku URL, mille abil luuakse ühendus. Konfiguratsiooni kasutatakse edaspidi nii riviloendite loomiseks, sõnumi saatmiseks kui ka vastuvõtmiseks [25].

Eelpool mainitud peamine ühendamise võti on leitav Azure portaalist. Vahendi dokumentatsioonis on autori hinnangul häiriv mitmete erinevate nimetuste kasutamine sama võtme kohta. Näiteks on dokumentatsioonis kasutatud järgmisi nimetusi: *RootManageSharedAccessKey*, *SAS_key_value*, *Connection string-primary key*, *Primary Connection String* ja *SAS token* [25].

Riviloendite loomiseks pakub Azure Service Bus eri võimalusi. Esimene võimalus on luua riviloend Azure portaali kasutades, tehes sobiva nimeruumi alt vastav valik. Teine võimalus on luua riviloend koodis. Näide selle võimaluse kohta on toodud lisas 3. Lisaks saab koodis määrata ka mitmeid riviloendi omadusi nagu näiteks selle maksimaalset suurust kilobaitides [25].

Kui riviloend on loodud on võimalik sinna sõnumeid saata. Telema dokumendivahetusplatvormi poolt varasemalt vastu võetud tellimuse Azure Service Bus'i

pilve saatmist võimaldav Java klass on toodud lisas 4. Sõnumi saatmiseks tuleb luua ühendus Azure Service Bus-iga (Joonis 12). Lisaks on vaja sisse lugeda tellimuse XML fail, mille realisatsioon on toodud lisas 5. Et tellimust oleks võimalik sisse lugeda tuleb luua ka FTP ühendus ning tellimus serverist alla laadida. FTP ühenduse loomine ja tellimuse allalaadimine on toodud lisas 6. FTP ühendus on loodud kasutades Apache Commons Net API-it ning ühenduse loomiseks on kasutatud Codejava veebilehekülje toodud koodinäiteid [26]. Kui tellimus on sisse loetud tõstetakse äsja loetud tellimuse fail Telema imporditud tellimuste arhiivi. Arhiivi tõstmine on toodud lisas 7.

Eelpool kirjeldatud toimingute tulemusena saadetakse tellimus Azure Service Bus'i riviloendisse. Riviloendist tellimuse vastuvõtmiseks NAV-is tuleb luua uus ühendus Azure Service Bus'iga ja realiseerida sõnumi vastuvõtmine *ReceiveAndDelete* või *PeekLock* meetodit kasutades. Antud realisatsioonis on läbi proovitud realisatsioon veakindlamat *PeekLock* meetodit kasutades. Sõnumi vastuvõtmine nimetatud meetodit kasutades on esitatud lisas 8.

5 Zato ja Azure Service Bus integratsiooniplatvormide võrdlus

Zato ja Azure Service Bus integratsiooniplatvormid on oma olemuselt erinevad. Töö eesmärgiks on leida Logistika Pluss OÜ-le hästi sobiv integratsiooniplatvorm. Eesmärgi saavutamiseks on koostatud võrdlus, mis toetub autori kogemustele nimetatud integratsiooniplatvormidega. Võrreldavad parameetrid on valitud koostöös Logistika Pluss OÜ-ga ning lähtuvad ettevõtte vajadustest. Parameetrite valikul on arvestatud nii tehnilisi kui ma mittetehnilisi näitajaid.

5.1 Platvormi arhitektuur

Zato ja Azure Service Bus's on oma arhitektuurilt küllaltki erinevad. Zato järgib tugevalt teenusetele orienteeritud arhitektuuri. Kõnealusest arhitektuurist on antud ülevaade Zato dokumentatsioonis ning läbivalt on kogu dokumentatsioonis kasutatud mitmeid teenustele orienteeritud arhitektuuriga seotud termineid. Kogu platvorm on üles ehitatud viisil, et ta sobiks hästi teenustele orienteeritud arhitektuuris ESB rolli. Näiteks vastavad Zato-s realiseeritud teenused täpselt teenustele orienteeritud arhitektuuris kirjeldatud teenustele. Zato koosnemine suurest hulgast komponentidest teeb platvormi mõistmise keerukaks. Samas peaks platvormi arhitektuur olema kiiresti mõistetav kui teenustele orienteeritud arhitektuur on eelnevast tuttav.

Azure Service Bus-i dokumentatsioonis teenustele orienteeritud arhitektuurile tähelepanu ei pöörata ning kõnealuse arhitektuuri põhimõtteid järgitud pole. Sellest tulenevalt on autori hinnangul Azure Service Bus'i puhul suurem tõenäosus, et valesid valikuid tehes muutub süsteem keeruliseks ja raskesti hallavatavaks. Azure Service Bus'is saab rakenduste vaheliseks suhtluseks kasutada kolme lahendust: riviloendit, teemapõhist riviloendit ja releed. Seega koosneb Azure Service Bus vähestest komponentidest ning rakenduse arhitektuuri mõistmisele palju aega ei kulu.

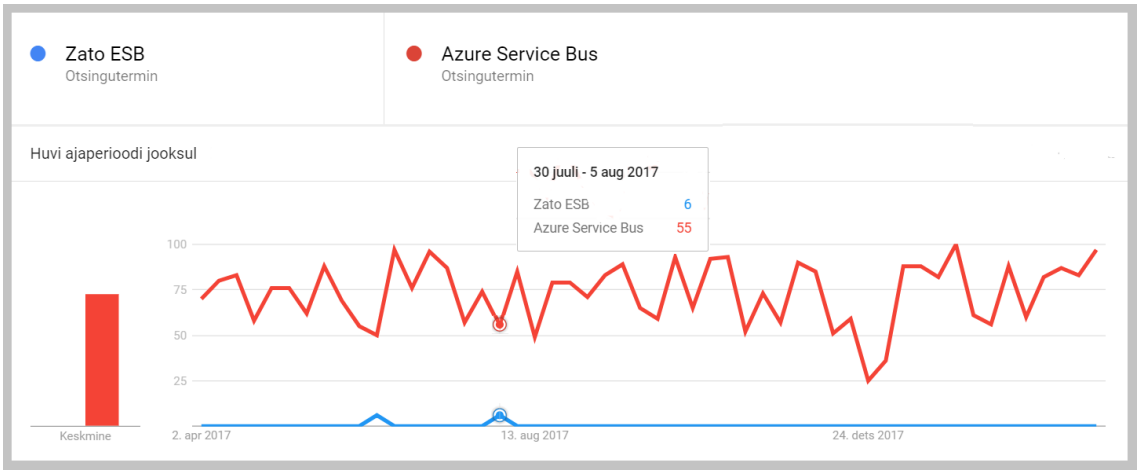
5.2 Platvormi õpitavus

Zato küllaltki keeruka arhitektuuri tõttu on platvormi tundma õppimine aeganõudev protsess. Samas on platvormi dokumentatsioon autori hinnangul väga hea ja ülevaatlik. Programmeerimisoskus pole Zato kasutamisel niivõrd oluline nagu Azure Service Bus'i puhul ning pigem on Zato kasutamisel rõhk õige konfiguratsiooni loomisel. Zato platvormi õppimisel ja kasutamisel on hea omada eelteadmisi teenustele orienteeritud arhitektuurist, arvutivõrkudest ja andmebaasisüsteemidest.

Azure Sevice Bus on kiiresti õpitav kuna koosneb vähestest komponentidest. Samas on Azure Service Bus'i dokumentatsioon autori hinnangul üsna kehv ja kohati eksitav. Kindlasti on Azure Service Bus'i puhul vajalikud eelteadmised programmeerimisest mõnes Azure Service Bus'i poolt toetatud programmeerimiskeeles, sest mitmed funktsionaalsused mida Zato puhul oli võimalik konfigureerida veebihaldus keskkonnas tuleb Azure Service Bus'i puhul realiseerida programmikoodis (näiteks FTP ühenduse loomine).

5.3 Platvormi tugi

Platvormide kasutamisel tekib paratamatult probleeme mille lahendamisel on abiks internetis saadaolevad materjalid või tuge pakkuvad ettevõtted. Suure tõenäosusega on mõistlikum võtta kasutusele platvorm, mis on populaarne ning mille kohta on võimalik leida erinevaid õpetusi ja juhendeid ning mille jaoks leidub paigaldust ja tuge pakkuvaid ettevõtteid. Selgus, et Zato platvorm on tunduvalt väiksema populaarsusega kui Azure Service Bus. Seda kinnitab näiteks Google Trends, mis võrdleb otsinguterminite suhet (Joonis 13) [27].



Joonis 13. Google Trends Zato ESB ja Azure Service Bus võrdlus [27].

Jooniselt (Joonis 13) on näha, et terminit „Azure Service Bus“ otsiti ligikaudu kümme korda rohkem kui terminit „Zato ESB.“ Ka autor koges oma töö käigus, et Azure Service Bus-i kohta info leidmine oli tunduvalt lihtsam kui Zato platvormi puhul. Azure Service Bus-i kohta leidus lisaks dokumentatsioonile mitmesugust informatsiooni erinevates foorumites ja veebisaitidel. Näiteks oli tuntud arendajate kogukonna leheküljel Stack Overflow esitatud 3263 Azure Service Bus-i puudutavat küsimust [28]. Zato puhul on lisaks dokumentatsioonile info allikaks Zato ametlik blogi [29]. Platvormi kasutamisel vigade avastamisel on võimalik sellest teada anda Zato Githubi repositooriumisse probleeme postitades. Peamiselt on seal probleemidena välja toodud kasutajate poolt avastatud platvormi vead või puudused, kuid leidus ka mõningaid platvormi kasutamist puudutavaid küsimusi. Postituste hulgas oli 599 suletud probleemi ja üle saja lahendust ootava probleemi [30]. Lisaks on Zatol olemas 510 postitusega aktiivne foorum, mis pakub kasutajate probleemide korral tuge [31]. Stack Owerflow leheküljel on Zato'ga seotud küsimusi esitatud vaid 35 [32].

5.3.1 Partnerettevõtte olemasolu

Logistika Pluss OÜ jaoks on lisaks eelmainitule oluline Eestis tegutseva partnerettevõtte olemasolu, kes tegeleks platvormi paigaldamise ja hilisema toega. Zato platvormile pakub Eestis tuge Avatud Lahendused OÜ, kellega on Logistika Pluss OÜ ka lahenduse osas suhelnud. Kuna Zato on oma olemuselt spetsiifilisem vahend kui Azure Service Bus siis on partneri leidmine keerulisem. Hetkel on teada vaid üks Eesti turul Zato platovormiga tegelev ettevõte, mis on Zato oluliseks miinuseks. Azure Service Bus'i puhul pole tegu nii spetsiifilise tootega ning suur osa funktsionaalsusi tuleks realiseerida

programmikoodis. Seetõttu sobivad partneriks mitmed Eesti turul tegutsevad ja Microsofti toodetega tegelevad arendusettevõtted.

Oluline on ka valitud vahendi tulevikukindlus ehk tulevikus pakutav tugi. Kuna Azure Service Bus on Microsofti toode võib eeldada, et ta on ka tulevikukindlam kui avatud lähtekoodiga Zato. Samas on tulevikukindluse osas võimalik teha vaid prognoose ning midagi kindlat väita pole võimalik.

5.4 Vigade avastamine

Zato platvormi kasutamisel tuleb vigade avastamiseks uurida logifaile. Infot on neis palju ning kasutamine autori hinnangul ebamugav. Ka teenuste kirjutamine Zato't kasutades on küllaltki ebamugav kuna programmeerimiskeskkond ei abista programmeerijat Zato API kasutamisel sama hästi nagu seda tegi Eclipse programmeerimiskeskkond koos sinna paigaldatud Azure tööriistakomplektiga (*Azure Toolkit for Eclipse*). Tänu nimetatud tööriistakomplekti ja programmeerimiskeskkonna koos kasutamise võimalusele on vigade avastamine Azure Service Bus'i kasutamisel lihtsam ja mugavam.

5.5 Referents

Platvormi valikul on oluline arvesse võtta, kas platvormi kasutavad ka Logistika Pluss OÜ-ga analoogilises valdkonnas tegutsevad ettevõtted. Autor leidis informatsiooni nii Zato kui ka Azure Service Bus platvorme kasutavate ettevõtete kohta. Mõlemat vaatluse all olevat platvormi kasutavad väga erinevates valdkondades tegutsevad ettevõtted kuid logistikaettevõtteid nende hulgas polnud [4], [33].

Sellegipoolest saab väita, et Azure Service Bus-i kasutavad Logistika Pluss OÜ-ga tegevusala poolest sarnasemad ning tuntumad ettevõtted. Näiteks kasutab Azure Service Bus platvormi teiste hulgas Toyota Material Handling Europe [33]. Zato platvormi kasutavate ettevõtete hulgast nii laialdaselt tuntud ettevõtteid ei leia. Küll aga räägib Zato platvormi kasuks asjaolu, et seda kasutavad mitu ülikooli ning kaasaegne tarkvaraarendusettevõte [4]. Eelnevast tulenevalt pole kummalgi platvormil antud võrdlusparameetri põhjal olulist eelist.

5.6 Maksumus

Zato puhul on tegu avatud lähtekoodiga tasuta platvormiga. Küll aga tuleb Zato puhul arvestada, et andmed asuksid selle platvormi valimisel lokaalses serveris. Seega lisanduvad Zato platvormi valimisel kulud serverile, näiteks amortisatsioon. Azure Service Bus on suletud lähtekoodiga tasuline teenus, mis pakub erinevaid hinnatasemeid. Küll aga pole Azure Service Bus'i kasutamine kulukas kuna maksustamine toimub API-ga loodud ühenduste arvu pealt. Logistika Pluss OÜ sõnul oleks nende hinnangul eelpool kirjeldatud Standard hinnatasemel teenuse kasutamise eest makstav tasu marginaalne. Seega ei anna tasuta platvormiks olemine Zato'le olulist eelist. Küll aga tekivad olulised kulutused Azure Service Bus'i kasutuselevõtul, sest paljud funktsionaalsused, mida Zato puhul on võimalik kasutada konfiguratsiooni muutes tuleb Azure Service Bus'i puhul realiseerida programmikoodis.

Seega kujuneks suure tõenäosusega Azure Service Bus-i kasutuselevõtu maksumus Zato omast kõrgemaks. Ka hilisemad muudatused tooksid Zato puhul kaasa väiksema kulu kuna platvorm pakub suure funktsionaalsusega veebihalduskeskkonda. Paljud muudatused oleks seetõttu võimalik teha Logistika Pluss OÜ enda poolt. Azure Service Bus'i kasutamisel tuleks muudatuste tegemiseks teha muudatusi programmikoodis ning seetõttu tuleks kasutada partnerina mõnd arendusettevõtet. Kokkuvõtvalt võib öelda, et Zato valimine integratsiooniplatvormiks kujuneks soodsamaks kui Azure Service Bus-i valik.

6 Järeldused

Autori hinnangul sobivad mõlemad platvormid klienditellimuste vastuvõtmiseks ja töötlemiseks mõeldud süsteemi arhitektuuri korrastamiseks. Seejuures on mõlemal platvormil tugevaid ja nõrku külgi. Et võtta platvormi valikul arvesse Logistika Pluss OÜ jaoks kõige olulisemaid kriteeriumeid ning anda platvormidele võimalikult objektiivne hinnang on valiku tegemisel kasutatud kaalutud mitme kriteeriumi põhist hindamismeetodit (*Weighted Multi-Factor Scoring Method*).

Nimetatud meetodi tulemused on toodud tabelis (Tabel 2). Kaalutud mitme kriteeriumi põhist hindamismeetodit kasutatakse valiku tegemiseks mitme alternatiivi vahel. Antud juhul on võrreldavateks alternatiivideks Zato ja Azure Service Bus integratsiooniplatvormid. Valiku tegemiseks hinnatakse platvorme erinevate kriteeriumite põhjal andes platvormidele punkte vahemikus 1-9. Saadud punktid korrutatakse läbi kriteeriumi kaaluga, sest kõikide kriteeriumite olulisus ettevõtte jaoks pole ühesugune. Kriteeriumi kaal väljendab protsentides näitaja olulisust ettevõtte jaoks [34]. Kaalude ja kriteeriumite määramisel lähtus autor Logistika Pluss OÜ poolt antud sisendist.

Tabel 2. Zato ja Azure Service Bus platvormide hindamine

Kriteerium	**Kaal**	**Zato**	**Azure Service Bus**
Toode	25%	0.25 x 8 = **2**	0.25x 4 = **1**
Partner ja platvormi tugi	25%	0.25 x 5 = **1.25**	0.25 x 7 = **1.75**
Referents	20%	0.2 x 1 = **0.2**	0.2 x 2 = **0.4**
Maksumus	30%	0.3 x 8 = **2.4**	0.3 x 3 = **0.9**
Kokku	**100%**	**5.85**	**4.05**

Nagu tabelist (Tabel 2) näha osutus punktide summeerimisel tulemusega 5.85 paremaks valikuks Zato platvorm. Kuna antud meetodi puhul võib subjektiivsest hindamisest tekkida hindamisviga on tulemuse usaldusväärsuse tagamiseks autor läbi viinud ka

tulemuste tundlikkuse analüüsi (*Sensitivity analysis of multi-factor scoring method*). Tulemuste tundlikkuse analüüs kujutab endast hinnangute muutmist, et selgitada proovimise teel välja kas hinnangute muutmine vastavalt võimalikule veale hinnangu andmisel mõjutab tulemust. Kõige olulisem on seejuures läbi proovida nende hinnangute muutmine, mis võivad kõige tõenäolisemalt olla ebatäpsed või liialt subjektiivsed [34]. Kirjeldatud analüüsi läbiviimise tulemusena saab väita, et platvormi valik on usaldusväärne. Hinnangute muutmisel tuli välja, et tulemus pole tundlik ning on vähetõenäoline, et tegu on väära valikuga. Tulemuse muutumiseks peaks antud hinnangute viga olema väga suur.

Tabelis (Tabel 2) toodud toote kriteeriumi all on hinnatud nii platvormide arhitektuuri, õpitavust kui ka vigade avastamist. Zato platvormi arhitektuur on autori hinnangul Logistika Pluss OÜ klienditellimuste saatmise ja vastuvõtmise süsteemi arhitektuuri korrastamiseks tunduvalt sobilikum kui Azure Service Bus'i arhitektuur. Teenustel orienteeritud arhitektuuri realiseerimiseks mõeldud Zato platvormi valimine aitab saavutada hästi hallatava arhitektuuriga klienditellimuste vastuvõtmiseks ja töötlemiseks mõeldud süsteemi. Azure Service Bus'i puhul on suurem tõenäosus, et valesid valikuid tehes muutub süsteem keeruliseks ja raskesti hallavatavaks. Seega tasub Zato platvormi keerukama arhitektuuri õppimine ennast ära. Vigade avastamine on aga parem Azure Service Bus platvormil. Zato puhul tuleb vead leida logifailidest samas kui Azure Service Bus'i puhul on võimalik kasutada programmeerimiskeskkonda ning sellele mõeldud Azure tööriistakomplekti, mis teeb vigade avastamise lihtsamaks. Lisaks eelmainitule on Logistika Pluss OÜ jaoks oluline, et integratsiooniplatvormi haldamisega tuleks toime ettevõtte praegused töötajad ning iga muudatuse jaoks poleks vaja kasutada välist partnerit. Ettevõtte töötajate oskuseid arvesse võttes tuleksid nad paremini toime Zato platvormi haldamisega. Kõike eelnimetatut võeti arvesse toote kriteeriumile hinnanguid andes.

Partneri ning platvormi toe kriteeriumi puhul on hinnatud eelnevalt analüüsitud platvormi kasutamist puudutavate materjalide hulka ning Eestis tuge pakkuvate ettevõtete olemasolu. Azure Service Bus on nii partnerite kui ka vahendi kohta leiduva materjali poolest paremini toetatud platvorm, kuid ka Zato kohta leidub häid materjale ja Eestis tuge pakkuv ettevõte.

Referentsi kriteeriumi puhul on hinnatud, kas platvorme kasutavad Logistika Pluss OÜ-ga analoogses valdkonnas tegutsevad ettevõtted. Azure Service Bus-i kasutavad Logistika Pluss OÜ-ga pisut sarnasemad ning tuntumad ettevõtted. Siiski pole antud kriteeriumi puhul olulist eelist kummalgi platvormil. Mõlema platvormi madala hinnangu tingis platvorme kasutavaid logistikaettevõtteid puudutava informatsiooni vähesus.

Logistika Pluss OÜ jaoks on kõige olulisem kriteerium maksumus. Zato puhul on paljud funktsionaalsused realiseeritud valmis kujul ning veebihaldus keskkonna abil kasutatavad. Azure Service Bus'i puhul tuleb samad funktsionaalsused realiseerida programmikoodis. See on peamiseks põhjuseks miks kujuneks Zato üles seadmise ja hilisema haldamise maksumus nii autori kui ka Logistika Pluss OÜ hinnangul soodamaks kui Azure Service Bus'i puhul.

Kaalutud mitme kriteeriumi põhise hindamismeetodi, analüüsi, võrdluse, ning autori isikliku liidestamiskogemuse põhjal on Logistika Pluss OÜ jaoks sobivam integratsiooniplatvorm Zato. Töö käigus kogutud materjal teenustele orienteeritud arhitektuuri ja Zato integratsiooniplatvormi kasutamise kohta ning üles seatud ja osaliselt konfigureeritud Zato platvorm koos Redis võti-väärtus andmebaasiga saab olema ettevõtte jaoks sisendiks Zato platvormi kasutuselevõtul.

7 Kokkuvõte

Logistika Pluss OÜ klienditellimuste vastuvõtmiseks ja töötlemiseks mõeldud süsteemi arhitektuur vajab korrastamist. Praeguses süsteemis esinevad järgmised probleemid: uute klientidega liidestuste loomine on keerukas, sisse tulevate tellimuste nõuetele vastavuse kontrolli teostatakse liiga hilja ning vigade korral tellimuses tuleb kliendiga ühendust võtta. Lisaks on praeguse arhitektuuri tõttu raskendatud Microsoft Dynamics NAV tarkvara versiooniuuenduse teostamine, kuna uue versiooni kasutuselevõtul tuleb kõik liidestused uuesti realiseerida. Nimetatud probleeme aitab lahendada sobiva integratsiooniplatvormi kasutuselevõtt.

Eelnevast tulenevalt oli käesoleva bakalaureusetöö põhieesmärgiks analüüsi ja võrdluse abil leida integratsiooniplatvorm, mis sobiks hästi Logistika Pluss OÜ klienditellimuste vastuvõtmiseks ja töötlemiseks mõeldud süsteemi arhitektuuri korrastamiseks ning pakkuda osaliselt realiseeritud integratsiooniplatvormi, mis on sisendiks edasisel integratsiooniplatvormi kasutuselevõtul Logistika Pluss OÜ-s.

Töö käigus uuris autor heade integratsioonipõhimõtete esitamise eesmärgil teenustel orienteeritud arhitektuuri ning proovis läbi tellimuse vastuvõtmise kaht Logistika Pluss OÜ-le huvipakkuvat integratsiooniplatvormi kasutades. Saadud kogemuste põhjal koostati platvormide võrdlus ning kaalutud mitme kriteeriumi põhise hindamismeetodi abil valiti välja paremini sobiv integratsiooniplatvorm.

Autori ja Logistika Pluss OÜ poolt püstitatud bakalaureusetöö eesmärk sai täidetud. Töö tulemusena jõuti otsusele, et Logistika Pluss OÜ jaoks hästi sobiv integratsiooniplatvorm on teenustele orienteeritud arhitektuuri elluviimiseks mõeldud Zato platvorm. Töö käigus kogutud materjal teenustele orienteeritud arhitektuuri ja Zato integratsiooniplatvormi kasutamise kohta ning üles seatud ja osaliselt konfigureeritud Zato platvorm koos Redis võti-väärtus andmebaasiga saab olema ettevõtte jaoks sisendiks Zato platvormi kasutuselevõtul. Lisaks pakub bakalaureusetöö väärtust ka teistele integratsiooniplatvormi abil süsteemi arhitektuuri korrastamisest huvitatud ettevõtetele.

Kasutatud kirjandus

[1] Enterprise Integration Patterns: Desgning, Building and Deploying Messaging Solutions. / G. Hohpe, B. Woolf, K. Brown, C. F. D'Cruse, M. Fowler, S. Neville, M. J. Retting, J. Simon. Massachusetts: Addison-Wesley Professional, 2011.

[2] Service Oriented Architecture for Dummies. / J. Hurwitz, R. Bloor, M. Kaufman, F. Halper. Indianapolis: Wiley Publishing, 2009.

[3] Chappel, D. A. Enterprise Service Bus. Sebastopol: O'Reilly Media, 2004.

[4] What makes Zato unique. [WWW] https://zato.io/index.html (02.03.2018)

[5] Nomenclature. [WWW] https://zato.io/docs/intro/nomen.html (02.03.2018)

[6] Redis. [WWW] https://zato.io/docs/architecture/redis.html (04.03.2018)

[7] Installation under Docker. [WWW] https://zato.io/docs/admin/guide/install/docker.html (17.03.2018)

[8] Zato quickstart create. [WWW] https://zato.io/docs/admin/cli/quickstart-create.html (04.03.2018)

[9] Load-balancer. [WWW] https://zato.io/docs/architecture/load-balancer.html (04.03.2018)

[10] Web admin. [WWW] https://zato.io/docs/architecture/web-admin.html (04.03.2018)

[11] Tutorial - part 1/2. [WWW] https://zato.io/docs/tutorial/01.html (17.03.2018)

[12] SimpleIO. [WWW] https://zato.io/docs/progguide/sio.html (04.03.2018)

[13] FTP usage examples. [WWW] https://zato.io/docs/progguide/examples/ftp.html (07.03.2018)

[14] Outgoing connections - FTP. [WWW] https://zato.io/docs/web-admin/outgoing/ftp.html (17.03.2018)

[15] Open-source ESB, SOA, REST, APIs and Cloud Integrations in Python. [WWW] https://zato.io/docs/index.html (19.03.2018)

[16] Channels. [WWW] https://zato.io/docs/progguide/channels.html (20.03.2018)

[17] Channels - Plain HTTP (REST). [WWW] https://zato.io/docs/web-admin/channels/plain-http.html (20.03.2018)

[18] Servers. [WWW] https://zato.io/docs/architecture/servers.html (20.03.2018)

[19] What is Azrue? [WWW] https://azure.microsoft.com/en-us/overview/what-is-azure/ (12.03.2018)

[20] Service Bus FAQ. [WWW] https://docs.microsoft.com/en-us/azure/service-bus-messaging/service-bus-faq (12.03.2018)

[21] Azure Service Bus. [WWW] https://docs.microsoft.com/en-us/azure/service-bus-messaging/service-bus-fundamentals-hybrid-solutions (12.03.2018)

[22] Service Bus pricing and billing. [WWW] https://docs.microsoft.com/en-us/azure/service-bus-messaging/service-bus-pricing-billing (25.03.2018)

[23] Service Bus pricing. [WWW] https://azure.microsoft.com/en-us/pricing/details/service-bus/ (29.03.2018)

[24] Create a Service Bus namespace using the Azure portal. [WWW] https://docs.microsoft.com/en-us/azure/service-bus-messaging/service-bus-create-namespace-portal (12.03.2018)

[25] How to use Service Bus queues with Java. [WWW] https://docs.microsoft.com/en-us/azure/service-bus-messaging/service-bus-java-how-to-use-queues (12.03.2018)

[26] Java FTP file download tutorial and example. [WWW] http://www.codejava.net/java-se/networking/ftp/java-ftp-file-download-tutorial-and-example (15.03.2018)

[27] Google Trends. [WWW] https://trends.google.com/trends/explore?q=Zato%20ESB,Azure%20Service%20Bus (03.04.2018)

[28] Results containig Azure Service Bus. [WWW] https://stackoverflow.com/search?q=Azure+Service+Bus (03.04.2018)

[29] Zato Blog. [WWW] https://zato.io/blog/ (03.04.2018)

[30] Zato Issues. [WWW] https://github.com/zatosource/zato/issues (03.04.2018)

[31] Zato Forum. [WWW] https://forum.zato.io (21.04.2018)

[32] Results containg Zato. [WWW] https://stackoverflow.com/search?q=Zato (29.03.2018)

[33] See the amazing things people are doing with Azure. [WWW] https://azure.microsoft.com/en-gb/case-studies/?service=service-bus (22.04.2018)

[34] Schniederjans, M. J., Hamaker, J. L., Schniederjans, A. M. Information Technology Investment Decision-Making Methodology. Singapore: World Scientific Publishing, 2005.

Lisa 1 – Arhitektuuri jooniste selgitused

Süsteemi nimetus	Kirjeldus
Tellimusveeb	Veebileht mille abil on Logistika Pluss OÜ klientidel võimalik esitada tellimusi, jälgida tellimusi ja vaadata aruandeid.
Ülemine vahevara	Tegu on vahekihiga Tellimusveebi ja NAV-i vahel. Tellimusveeb lisab XML-i vahetabelisse ja mingi teatud aja tagant loetakse see fail sisse. Analoogiliselt toimib andmevahetus ka NAV-iga.
NAV	NAV-is (*Microsoft Dynamics Navision)* algatatakse tehingud, näiteks ostutellimused ja müügitellimused.
Alumine vahevara	Vahekiht NAV-i ja RIS-i vahel. Juhul kui tehing kinnitatakse võetakse vahevara päringu peale andmed NAV-ist ja kopeeritakse vastavasse vahevara tabelisse.
RIS	RIS on laojuhtimise tarkvara. RIS-i päringu peale võetakse vahevarast andmed ja kopeeritakse RIS-i tabelisse. Seda tehakse enne noppimist või ladustamist. RIS raporteerib vahevarasse, et vastav tehing on alustatud ja hiljem ka lõpetatud. RIS võib algatada ka inventuuri ja töödeldava kauba saldo tehinguid. Nende info saabub NAV-i üksnes infoks.
MS SQL	Liidestused on realiseeritud kasutades SQL vahetabeleid või otse NAV-i tabeleid. Kasutusel on andmebaasisüsteem MS SQL 2014.
Andevis	Tööriist palga, personali ja tööaja arvestamiseks.
CRM	Tegu on klienditoe keskkonnaga. Klientide andmed on aga ebatäielikud, sest liidestus NAV-iga puudub.
Telema	Dokumendivahetuse teenusepakkuja
Edisoft	Dokumendivahetuse teenusepakkuja
Docura	Arenduspartner kellega koostöös on realiseeritud erilahendused mõningate klientide jaoks.
Festival liides	Kliendiga liidestamiseks loodud erilahendus. Festivali abil saabub tellimus XML kujul sama struktuuriga nagu Telemat kasutades.

Lisa 2 – Tellimuse dokumendi näide

```
<?xml version="1.0" encoding="UTF-8"?>
<E-Document>
   <Header>
      <DateIssued>2018-02-06</DateIssued>
      <SenderGLN>4832727232892</SenderGLN>
      <ReceiverID>LP</ReceiverID>
   </Header>
   <Document>
      <DocumentType>order</DocumentType>
      <DocumentFunction>original</DocumentFunction>
      <DocumentParties>
         <BuyerParty><PartyCode>20352311</PartyCode><Name>Toidupood
AS</Name><RegNum>20352311</RegNum><VATRegNum>EE100560972</VATRegNum><G
LN>4830013525232</GLN><ContactData>
               <PhoneNum>663 1100</PhoneNum>
               <FaxNum>663 1101</FaxNum>
               <EmailAddress>mari.kuusepuu@toidupood.eu</EmailAddress>
               <ActualAddress>
                  <Address1>Narva mnt 234</Address1>
                  <City>Tallinn</City>
                  <PostalCode>11646</PostalCode>
                  <County>harjumaa</County>
                  <CountryCode>EE</CountryCode>
               </ActualAddress>
            </ContactData><AccountInfo>
               <AccountNum>223480740849</AccountNum>
               <IBAN>EE134330022348074 0849</IBAN>
               <BIC>AAAASS9Y</BIC>
               <BankName>Swedbank AS</BankName>
            </AccountInfo></BuyerParty>
         <DeliveryParty><PartyCode>5555-33198</PartyCode><Name>Saku
Toidupood</Name><RegNum>20352311</RegNum><VATRegNum>EE100560972</VATRe
gNum><GLN>4832727232892</GLN><ContactData>
               <PhoneNum>6631110</PhoneNum>
               <EmailAddress>saku@toidupood.ee</EmailAddress>
               <ActualAddress>
                  <Address1>Jaama 32</Address1>
                  <City>Saku</City>
                  <PostalCode>11192</PostalCode>
                  <County>Harjumaa</County>
                  <CountryCode>EE</CountryCode>
               </ActualAddress>
            </ContactData></DeliveryParty>
         <OrderParty><PartyCode>5555-33198</PartyCode><Name>Saku
toidupood</Name><RegNum>20352311</RegNum><VATRegNum>EE100560972</VATRe
gNum><GLN>4832727232892</GLN><ContactData>
```

```
                <PhoneNum>6631110</PhoneNum>
                <EmailAddress>saku@toidupood.ee</EmailAddress>
                <ActualAddress>
                   <Address1>Jaama 32</Address1>
                   <City>Saku</City>
                   <PostalCode>11192</PostalCode>
                   <County>Harjumaa</County>
                   <CountryCode>EE</CountryCode>
                </ActualAddress>
             </ContactData></OrderParty>
          <SellerParty><PartyCode>7132</PartyCode><Name>Deof Trade
OÜ</Name><RegNum>23985545</RegNum><VATRegNum>EE24589123</VATRegNum><GL
N>9653429987872</GLN><ContactData>
                <PhoneNum>+372 5123212</PhoneNum>
                <EmailAddress>triin.tamm@deoftrade.com</EmailAddress>
                <ActualAddress>
                   <Address1>Liivalaia 123</Address1>
                   <City>Tallinn</City>
                   <PostalCode>11466</PostalCode>
                   <County>Harjumaa</County>
                   <CountryCode>EE</CountryCode>
                </ActualAddress>
             </ContactData></SellerParty>
       </DocumentParties>
       <DocumentInfo>
          <DocumentName>Tellimus</DocumentName>
          <DocumentNum>OT011230021</DocumentNum>
          <PaymentTerm>20</PaymentTerm>
          <DateInfo>
             <OrderDate>2018-02-06</OrderDate>
             <ProcessingDate>2018-02-06</ProcessingDate>
             <DeliveryDateRequested>2018-02-07</DeliveryDateRequested>
          </DateInfo>
          <CreatedByContact>
             <PhoneNum>56253785</PhoneNum>
             <EmailAddress>jaanus.kivi@toidupood.ee</EmailAddress>
          </CreatedByContact>
       </DocumentInfo>
       <DocumentSumGroup>
          <Currency>EUR</Currency>
       </DocumentSumGroup>
       <DocumentItem>
          <ItemEntry>
             <LineItemNum>1</LineItemNum>
             <BuyerItemCode>H000055111</BuyerItemCode>
             <SellerItemCode>6862630004736</SellerItemCode>
             <GTIN>6862630004736</GTIN>
             <ItemDescription>Huulekreem Lip Relief
tuub</ItemDescription>
             <ItemInfo>
                <BrandName lang="EN">BLISTEX</BrandName>
             </ItemInfo>
             <ItemUnitRecord>
                <ItemUnit>tk</ItemUnit>
```

```
   </ItemUnitRecord>
   <BaseUnit>tk</BaseUnit>
   <AmountOrdered>24</AmountOrdered>
   <ItemReserve>
      <ItemReserveUnit>
         <Size>6 g</Size>
         <ItemUnit>tk</ItemUnit>
         <AmountActual>24</AmountActual>
      </ItemReserveUnit>
   </ItemReserve>
</ItemEntry>
<ItemEntry>
   <LineItemNum>2</LineItemNum>
   <BuyerItemCode>T000022539</BuyerItemCode>
   <GTIN>7676665295008</GTIN>
   <ItemDescription>Linaseemned</ItemDescription>
   <ItemInfo>
      <BrandName lang="EN">REFORMI</BrandName>
   </ItemInfo>
   <ItemUnitRecord>
      <ItemUnit>tk</ItemUnit>
   </ItemUnitRecord>
   <BaseUnit>tk</BaseUnit>
   <AmountOrdered>16</AmountOrdered>
   <ItemReserve>
      <ItemReserveUnit>
         <Size>500 g</Size>
         <ItemUnit>tk</ItemUnit>
         <AmountActual>16</AmountActual>
      </ItemReserveUnit>
   </ItemReserve>
</ItemEntry>
<ItemEntry>
   <LineItemNum>3</LineItemNum>
   <BuyerItemCode>T000045009</BuyerItemCode>
   <GTIN>5670350009432</GTIN>
   <ItemDescription>Pan sool</ItemDescription>
   <ItemInfo>
      <BrandName lang="EN">REFORMI</BrandName>
   </ItemInfo>
   <ItemUnitRecord>
      <ItemUnit>tk</ItemUnit>
   </ItemUnitRecord>
   <BaseUnit>tk</BaseUnit>
   <AmountOrdered>8</AmountOrdered>
   <ItemReserve>
      <ItemReserveUnit>
         <Size>1,1 kg</Size>
         <ItemUnit>tk</ItemUnit>
         <AmountActual>8</AmountActual>
      </ItemReserveUnit>
   </ItemReserve>
</ItemEntry>
<ItemEntry>
```

```
            <LineItemNum>4</LineItemNum>
            <BuyerItemCode>T000039701</BuyerItemCode>
            <SellerItemCode>277607</SellerItemCode>
            <GTIN>9836000844386</GTIN>
            <ItemDescription>Saiapulber</ItemDescription>
            <ItemInfo>
               <BrandName lang="EN">DR.SCHÄR</BrandName>
            </ItemInfo>
            <ItemUnitRecord>
               <ItemUnit>tk</ItemUnit>
            </ItemUnitRecord>
            <BaseUnit>tk</BaseUnit>
            <AmountOrdered>10</AmountOrdered>
            <ItemReserve>
               <ItemReserveUnit>
                  <Size>1 kg</Size>
                  <ItemUnit>tk</ItemUnit>
                  <AmountActual>10</AmountActual>
               </ItemReserveUnit>
            </ItemReserve>
         </ItemEntry>
         <ItemEntry>
            <LineItemNum>5</LineItemNum>
            <BuyerItemCode>T000055564</BuyerItemCode>
            <SellerItemCode>180065</SellerItemCode>
            <GTIN>8008698003152</GTIN>
            <ItemDescription>Ciabatta Rustica eelküpsetatud
mitmeviljasaiake</ItemDescription>
            <ItemInfo>
               <BrandName lang="EN">DR.SCHÄR</BrandName>
            </ItemInfo>
            <ItemUnitRecord>
               <ItemUnit>tk</ItemUnit>
            </ItemUnitRecord>
            <BaseUnit>tk</BaseUnit>
            <AmountOrdered>5</AmountOrdered>
            <ItemReserve>
               <ItemReserveUnit>
                  <Size>200 g</Size>
                  <ItemUnit>tk</ItemUnit>
                  <AmountActual>5</AmountActual>
               </ItemReserveUnit>
            </ItemReserve>
         </ItemEntry>
      </DocumentItem>
   </Document>
</E-Document>
```

Lisa 3 – Riviloendi loomine Javas Azure Service Bus-iga

```
package ee.lp.com;

import com.microsoft.windowsazure.exception.ServiceException;
import com.microsoft.windowsazure.services.servicebus.*;
import com.microsoft.windowsazure.services.servicebus.models.*;
import com.microsoft.windowsazure.Configuration;

public class QueueCreator {
      public void createQueue(){

      AzureConnectionConfig config = new AzureConnectionConfig();
      Configuration conf = config.createConfig();

      ServiceBusContract service = ServiceBusService.create(conf);
      QueueInfo queueInfo = new QueueInfo("telemaqueue");

        try
        {
            CreateQueueResult result = service.createQueue(queueInfo);
        }
        catch (ServiceException e)
        {
            System.out.print("ServiceException encountered: ");
            System.out.println(e.getMessage());
            System.exit(-1);
        }
    }
}
```

Lisa 4 – Sõnumi saatmine Javas Azure Service Bus-iga

```
package ee.lp.com;

import com.microsoft.windowsazure.Configuration;
import com.microsoft.windowsazure.exception.ServiceException;
import com.microsoft.windowsazure.services.servicebus.ServiceBusContract;
import com.microsoft.windowsazure.services.servicebus.ServiceBusService;
import com.microsoft.windowsazure.services.servicebus.models.BrokeredMessage;

public class MessageSender {

	public void sendMessage(){

		AzureConnectionConfig config = new AzureConnectionConfig();
		Configuration conf = config.createConfig();
		ServiceBusContract service = ServiceBusService.create(conf);

			try{
			    XmlOrderReader xmlReader = new XmlOrderReader();
			    String telemaXml = xmlReader.getXmlFromFtpServer();

			    BrokeredMessage message =
			            new BrokeredMessage(telemaXml);
			    service.sendQueueMessage("telemaqueue", message);
			}
			catch (ServiceException e)
			{
			    System.out.print("ServiceException encountered: ");
			    System.out.println(e.getMessage());
			    System.exit(-1);
			}
	}
}
```

Lisa 5 – XML kujul tellimuse lugemine Javas

```
package ee.lp.com;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

public class XmlOrderReader {
	private String location = "C:\\Users\\LPuser\\Telema\\"
				+ "import\\LP-order-xml-example.xml";
	private String destination ="C:\\Users\\LPuser\\Telema\\imported\\";

	public String getXmlFromFtpServer() throws IOException{

		FtpOrderDownloader ftpOrderDownloader = new FtpOrderDownloader();
		ftpOrderDownloader.downloadFile(location);

		BufferedReader br = new BufferedReader(new FileReader(location));

		try {
		    StringBuilder sb = new StringBuilder();
		    String line = br.readLine();
		    while (line != null) {
		        sb.append(line);
		        sb.append(System.lineSeparator());
		        line = br.readLine();
		    }
		    String orderDataXml = sb.toString();
		    return orderDataXml;

		} finally {
			FileCopier fileCopier = new FileCopier();
			fileCopier.moveAndDeleteFile(location, destination);

		    br.close();
		}
	}
}
```

Lisa 6 – Tellimuse allalaadimine FTP ühenduse kaudu Javas

```
package ee.lp.com;

import java.io.BufferedOutputStream;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;
import org.apache.commons.net.ftp.FTP;
import org.apache.commons.net.ftp.FTPClient;

public class FtpOrderDownloader {
    private String server = "192.168.1.11";
    private int port = 21;
    private String user = "LPuser";
    private String pass = "secretpassword";

    public void downloadFile(String location){
      FTPClient ftpClient = new FTPClient();

        try {
            ftpClient.connect(server, port);
            ftpClient.login(user, pass);
            ftpClient.enterLocalPassiveMode();
            ftpClient.setFileType(FTP.BINARY_FILE_TYPE);

            String remoteFile = "/orders/LP-order-xml-example.xml";

            File downloadFile = new File(location);
            OutputStream outputStream =
                     new BufferedOutputStream(
                             new FileOutputStream(downloadFile));
            boolean success =
                    ftpClient.retrieveFile(remoteFile, outputStream);
            outputStream.close();

            if (success) {
                System.out.println("File download successful.");
            }

        } catch (IOException ex) {
            System.out.println("Error: " + ex.getMessage());
            ex.printStackTrace();
        } finally {
            try {
                if (ftpClient.isConnected()) {
                    ftpClient.logout();
                    ftpClient.disconnect();
```

```
                }
            } catch (IOException ex) {
                ex.printStackTrace();
            }
        }
    }
}
```

Lisa 7 – Imporditud tellimuse arhiivi tõstmine Javas

```
package ee.lp.com;

import java.io.File;
import java.io.IOException;
import org.apache.commons.io.FileUtils;

public class FileCopier {
      private String importedFileName = "LP-order-xml-example-imported.xml";

      public void moveAndDeleteFile(String location, String destination)
                throws IOException{
            File oldFile = FileUtils.getFile(location);
            File newFile = new File(destination + importedFileName);
            if(!newFile.exists()){
               newFile.createNewFile();
            }
            FileUtils.copyFile(oldFile, newFile);
            FileUtils.deleteQuietly(oldFile);
      }
}
```

Lisa 8 – Sõnumi vastuvõtmine Javas Azure Service Bus-iga

```
package ee.lp.com;

import com.microsoft.windowsazure.exception.ServiceException;
import com.microsoft.windowsazure.services.servicebus.*;
import com.microsoft.windowsazure.services.servicebus.models.*;
import com.microsoft.windowsazure.Configuration;
import com.microsoft.windowsazure.core.*;
import com.microsoft.windowsazure.exception.ServiceException;
import javax.xml.datatype.*;

public class MessageReceiver {
            public void ReceiveMessages(){
                  try
                  {
                        AzureConnectionConfig config =
                                    new AzureConnectionConfig();

                        Configuration conf = config.createConfig();
                        ServiceBusContract service =
                                    ServiceBusService.create(conf);

                      ReceiveMessageOptions opts =
                             ReceiveMessageOptions.DEFAULT;
                      opts.setReceiveMode(ReceiveMode.PEEK_LOCK);

                      while(true)  {
                        ReceiveQueueMessageResult resultQM =
                            service.receiveQueueMessage("telemaqueue",
                                     opts);
                          BrokeredMessage message =
                                    resultQM.getValue();
                        if (message != null &&
                                    message.getMessageId() != null)
                        {
                            byte[] b = new byte[200];
                            String s = null;
                            int numRead = message.getBody().read(b);
                            while (-1 != numRead)
                            {
                                s = new String(b);
                                s = s.trim();
```

```
                            System.out.print(s);
                            numRead = message.getBody().read(b);
                        }
                        service.deleteMessage(message);
                    }
                    else
                    {
                        System.out.println();
                        System.out.println("No more messages.");
                        break;
                    }
                }
            }
            catch (ServiceException e) {
                System.out.print("ServiceException: ");
                System.out.println(e.getMessage());
                System.exit(-1);
            }
            catch (Exception e) {
                System.out.print("Generic exception: ");
                System.out.println(e.getMessage());
                System.exit(-1);
            }
        }
    }
```