TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Margus Põlma 175271IDDR

Majandustarkvara laomooduli arendus

diplomitöö

Juhendaja: Toomas Lepikult
PhD

Tallinn 2021

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: Margus Põlma

07.01.2021

Annotatsioon

Käesoleva lõputöö probleem on ajendatud ühe majandustarkvara pakkuva ettevõtte vajadusest uue laohaldussüsteemi järele. Probleemi lahendamiseks tuleks leida või luua uus laosüsteem, mis vastaks ettevõtte nõudmistele ning samas oleks piisavalt universaalne, et seda oleks võimalik kasutada ka teistes tarkvarades. Seega on lõputöös käsitletav probleem laiem kui lihtsalt ühe ettevõtte vajadus.

Lõputöös uuritakse lähemalt kolme turul saadaolevat laohaldussüsteemi, mis võiksid probleemi lahendamiseks sobida ning tuuakse välja nende kasutuselevõtuga kaasnevad probleemid. Selgub, et hinna ning kaasnevate probleemide tõttu ei ole mõistlik probleemi lahendamiseks kasutada mõnda valmislahendust.

Lõputöös otsustati luua uus laohaldussüsteem REST stiilis veebiteenusena kasutades selleks skriptimiskeel PHP-d ning andmebaasi juhtsüsteemi MySQL-i. Lõputöö tulemusena valmis prototüüp, mis vastab osaliselt ettevõtte poolt seatud nõudmistele ning mis on piisavalt universaalne, et seda oleks võimalik liidestada teiste tarkvaraga, mis võimaldavad HTTP protokolli kasutamist.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 45 leheküljel, 7 peatükki, 7 joonist, 5 tabelit.

Abstract

Development of Warehouse Module for Business Management Software

A company providing a SaaS is in a need of a new inventory management system to provide additional functionalities to its customers. The company has decided that a new inventory management system should be implemented. The new system may be created from scratch or a ready-made software could be used. Other developers or companies might face a similar problem if they need to implement a simple inventory management system. Therefore, the solution of the problem should be as universal as possible.

In this thesis the author examines the company's current inventory management systems and creates some UML models in order to gain better understanding of the system. Then the author examines three commercial inventory management systems that could be integrated to its own software and finds the best of three. The author also points out some problems that arise when implementing such software. The best solution from selected three is a bit too expensive to use.

Therefore, the author decides that it would be reasonable to create a new inventory management system from scratch. Author uses scripting language PHP and database management system MySQL to build a REST style webservice as a prototype for the new inventory management system that could later be integrated with the company's SaaS as well with other software systems that support HTTP protocol.

The thesis is in Estonian and contains 45 pages of text, 7 chapters, 7 figures, 5 tables.

Lühendite ja mõistete sõnastik

API	*application programming interface* ehk programmiliides
ASP.NET	C# tarkvararaamistik
Composer	PHP paketihaldur
C#	objektorienteeritud programmeerimiskeel
DTO	*data transfer object* ehk andmeedastusobjekt
Doctrine	PHP ORM raamistik
Eloquent	PHP ORM raamistik
GIT	versioonihaldussüsteem
HTTP	*Hypertext Transfer Protocol* ehk hüperteksti edastusprotokoll
HTTPS	HTTP *Secure* ehk turvaline hüperteksti edastusprotokoll
JAX-RS	Java tarkvararaamistik
JSON	*JavaScript Object Notation* ehk andmevorming
JWT	JSON *Web Token* ehk standardiseeritud turvalise andmevahetuse viis
Java	objektorienteeritud programmeerimiskeel
LAMP-pinu	*Linux, Apache, MySQL, PHP stack* ehk komplekt arendusvahendeid
Laravel	PHP tarkvararaamistik
Lumen	PHP tarkvararaamistik
MySQL	relatsioonilise andmebaasi juhtimissüsteem
ORM	*object-relational mapping* ehk metoodika mis võimaldab ühest süsteemist saadud info teisendada teisele süsteemile arusaadavale kujule ning vastupidi
PHP	serveripoolne skriptimiskeel
PHPStorm	PHP integreeritud arenduskeskkond
POPO	*plain old PHP object* ehk lihtsad PHP objektid
PostgreSQL	objekt-relatsioonilise andmebaasi juhtimissüsteem
REST	*Representional State Transfer* ehk arhitektuuristiil andmevahetuseks
SOAP	*Simple Object Access Protocol* ehk standard andmevahetuseks

SQL	andmebaasi päringukeel
SQLite	relatsioonilise andmebaasi juhtimissüsteem
SaaS	*Software as a Service*
Slim	PHP tarkvararaamistik
Spring MVC	Java tarkvararaamistik
UML	*Unified Modeling Language* ehk standardne keel tarkvara kavandamiseks
XML	*Extensible Markup Language* ehk märgistuskeel andmevahetuseks

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Turul on saadaval suur hulk erinevaid laoprogramme. On nii tasuta kasutamiseks kui ka tasulisi lahendusi. On lahendusi mis on loodud kasutamiseks tootmisettevõtetes ning on selliseid mis on loodud keskendudes hulgimüüjate vajadustele. Kindlasti leidub erinevaid lahendusi ka muudele valdkondadele. [1]

Lõputöö probleem on ajendatud ühe pilvepõhise majandustarkvara pakkuva ettevõtte vajadusest uue laosüsteemi järele. Tuleks uurida kas turul on saadaval mõni valmis lahendus mis vastaks ettevõtte vajadustele ning samuti tuleks uurida probleeme mis kerkiksid sellise lahenduse kasutusele võtmisel.

1.1 Probleem

Ühe pilvepõhise majandustarkvara laosüsteem vajab täiendavat funktsionaalsust, et pakkuda enda klientidele paremat teenust. Lisaks on ettevõte olukorras, kus tarkvara koodibaas on kasvanud üsna suureks ning seetõttu võtavad uued arendused üha rohkem aega**.** Ettevõte on otsustanud, et mõistlik oleks täiesti uue laosüsteemi leidmine või loomine. Turul leidub küll palju lahendusi kuid enamus neist on kas liiga kulukad, keerukad või neil puuduvad vastavad liidesed mille abil oleks võimalik kogu laosüsteemi funktsionaalsust mõne teise tarkvaraga integreerida.

Sarnase probleemi ees võivad olla ka teised arendajad kui neil on vaja luua lihtne laosüsteem mille kogu funktsionaalsust oleks võimalik integreerida ükskõik millise teise tarkvara koosseisu. Seega võib lõputöös käsitletav probleem olla laiem kui lihtsalt ühe ettevõtte vajadus.

1.2 Eesmärk

Püstitatud probleemi lahendamiseks oleks vaja kas leida või luua selline laosüsteem mis vastaks esitatud nõudmistele ning mis oleks täielikult liidestatav ka teiste tarkvaradega. Selleks, et oleks võimalik uut süsteemi leida või luua tuleks esmalt uurida hetkel kasutusel olevat lahendust. Seejärel on võimalik otsida erinevaid turul saadaolevaid lahendusi, mis

võiksid probleemi lahendamiseks sobida. Kindlasti tuleb uurida ka erinevaid probleeme, mis kaasneksid mõne sellise lahenduse kasutuselevõtuga.

Lõputöö tulemusena on välja selgitatud uue laosüsteemi arendusplatvorm ning loodud selle abil prototüüp. Loodavat lahendust peab olema võimalik lihtsasti liidestada ka teiste tarkvaradega.

1.3 Lähtetingimused

Lõputöö probleemi lahendamisel tuleb lähtuda ettevõtte hetkel kasutatava süsteemi nõudmistest ning funktsionaalsusest. Selle jaoks on lõputöö sisendiks praeguse laosüsteemi kasutajalood (vt Lisa 2), mida on universaalsuse tagamiseks kohati üldistatud. Lisaks on autorile teada laosüsteemi põhifunktsionaalsused, kuid need pole dokumenteeritud.

Eelpool nimetatud sisendite põhjal on võimalik leida ning analüüsida võimalikke saadaolevaid lahendusi. Samuti aitavad need kaasa uue laosüsteemi loomisele. Siiski leiab autor, et eesmärgi saavutamiseks oleks hetkel kasutatavat süsteemi vaja täiendavalt uurida.

1.4 Töö ülesehitus

Käesoleva diplomitöö teises peatükis tutvustatakse hetkel kasutusel olevat laosüsteemi kirjeldades selle funktsionaalsust. See peatükk on oluline, sest selles kogutud info alusel on võimalik leida turult probleemi lahendavad kandidaadid ning samuti on see info uue süsteemi loomise aluseks. Teise peatüki lõpuosas pannakse paika lõputöö skoop.

Töö kolmandas peatükis uuritakse lähemalt turul saadaolevaid lahendusi. Vaatluse alla võetakse mõned laosüsteemid, mis autori arvates võiksid sobida probleemi lahendamiseks. Valitud laosüsteeme võrreldakse erinevate parameetrite alusel ning peatüki lõpus tehakse järeldused. Lisaks uuritakse lühidalt muid Eestis kasutatavaid lahendusi ning tuuakse välja suuremad probleemid, mis kaasneksid mõne sellise tarkvara kasutuselevõtuga.

Töö neljandas peatükis leitakse parim võimalik arendusplatvorm uue laosüsteemi loomiseks. Võrreldakse erinevaid arhitektuure, programmeerimiskeeli ning

andmebaasimootoreid ning valitakse nende seast parim. Valiku tegemisel lähtutakse vahendi sobivusest probleemi lahendamiseks, trendidest ning autori ja ettevõtte kogemusest.

Lõputöö viiendas peatükis kirjeldatakse uue lahenduse prototüübi loomise protsessi olulisemaid osasid. Prototüüp luuakse kasutades vahendeid, mis valiti neljandas peatükis. Viienda peatüki lõpus testitakse lühidalt loodud prototüübi põhifunktsionaalsusi.

Töö kuuendas peatükis antakse hinnang töös tehtud otsustele ning sellele kas tehtud otsused või loodud lahendus täidab peatükis 1.2 sõnastatud eesmärki. Lisaks kirjeldatakse lühidalt edasisi tegevusi.

2 Hetkel kasutatav lahendus

Selleks, et töös püstitatud probleemi lahendada tuleks esmalt uurida ettevõtte poolt kasutatava süsteemi funktsionaalsust ning tausta selle taga. Just see info on uue laosüsteemi leidmise või loomise aluseks.

2.1 Lühikirjeldus

Hetkel kasutatav lahendus võimaldab selle kasutajatel teha tüüpilise laosüsteemi toiminguid. Sellised toimingud on näiteks:

- tarnijate haldamine,
- klientide haldamine,
- laoartiklite haldamine,
- ostuarvete haldamine,
- müügiarvete haldamine,
- laoartiklite ostmine ja müümine vastava arve alusel,
- laoartiklite tellimine.

Eelnevalt kirjeldatud toimingute alusel on süsteemil võimalik genereerida ning kuvada selle kasutajatele erinevat statistikat, näiteks:

- artiklite liikumise aruannet,
- müügi- ja ostuarvete aruannet,
- tellimuste ajalugu.

Kuna tegemist on majandustarkvaraga millel on rohkem kui üks kasutaja, siis kasutatakse ka antud moodulit mitmete klientide poolt korraga. Seega on kogu teave andmebaasis eraldatud ning iga kasutaja saab käidelda ainult selliseid andmeid, mis on tema enda või tema organisatsiooni poolt loodud.

Ülal kirjeldatud omadused on ainult üks osa antud majandustarkvarast. Kogu tarkvara saab tema kihilise arhitektuuri järgi klassifitseerida monoliitseks süsteemiks. Kogu tarkvara on loodud LAMP-pinule (*Linux, Apache, MySQL, PHP stack*).

2.2 Süsteemi nõuded

Eelnevas alampeatükis kirjeldati üldistatult uuritavat laosüsteemi, et anda ülevaade selle funktsionaalsustest ning keerukusastmest. Järgnevas alampeatükis modelleeritakse süsteemi käitumist nii nagu lõppkasutaja seda näeb. Loodavate mudelite abil on võimalik leida turult lõputöö probleemi lahendavad kandidaadid ning lisaks on need sisendiks uue süsteemi loomise protsessile.

Järgnevas peatükis loodavad mudelid sünteesitakse lõputöö sisendiks olevatest kasutuslugudest (vt Lisa 2) ning autori ning ettevõtte vahel toimunud ekspertintervjuust saadud info alusel. Kogu süsteemi funktsionaalsust ei ole autori arvates mõistlik siinkohal välja tuua, sest see oleks üsna mahukas töö - keskendutakse olulisemale. Modelleerimisel kasutatakse asjakohaseid UML-i vahendeid.

UML (*Unified Modeling Language*) on standardne keel, mille abil on võimalik kavandada tarkvara või tarkvarasüsteeme luues erinevaid mudeleid, diagramme, dokumentatsiooni vms [2]. Nimetatud modelleerimiskeele versiooni 2.0 poolt defineeritud mudelid on võimalik jaotada kolme kategooriasse:

1. Struktuurimudelid (*Structure diagram*), mis kirjeldavad rakenduse staatilist struktuuri;
2. Käitumismudelid (*Behavior diagram*), mis kirjeldavad rakenduse üldistatud käitumist;
3. Suhtlusmudelid (*Interaction diagram*), mis kirjeldavad erinevat tüüpi suhtlust erinevate süsteemi osade vahel. [3]

On oluline, et uus süsteem ei erineks suurel määral praegusest lahendusest ning seda just lõppkasutaja vaatest. Sellest tulenevalt on autor arvamusel, et mõistlik on siinkohal keskenduda käitumismudelite loomisele, sest nende abil on võimalik kirjeldada tarkvara üldistatud käitumist kasutaja vaatest.

2.2.1 Kasutusmallide mudelid

Selleks, et anda kõrgetasemeline ülevaade süsteemi funktsionaalsustest loob autor kasutusmallide mudelid analüüsides kasutuslugusid (vt Lisa 2) ning ekspertintervjuust saadud infot. Kasutusmallide mudel (*use case diagram*) on UML-i poolt defineeritud käitumismudel, mis vastavalt ühele UML käsiraamatule: "näitab hulka kasutusjuhte, rolle

ning seoseid nende vahel" [2]. Seega sobivad need mudelid hästi selleks, et välja selgitada hetkel kasutatava laosüsteemi funktsionaalsused lõppkasutaja vaatest, ehk tegevused mida erinevad kasutajad teha saavad.

Kõik lühikirjelduses (vt Peatükk 2.1) nimetatud toimingud on otseselt või kaudselt seotud toote ehk artikliga. Seega võib artiklit pidada konkreetse süsteemi keskseks objektiks. Allpool oleval joonisel (vt Joonis 1) on näha kasutusmallide mudel, mis näitab kõiki võimalikke kasutusjuhte ning rolle, mis on otseselt seotud artikliga.

Süsteemi kasutajal on võimalik lisada ning muuta artiklit. Artikli kohta on võimalik salvestada erinevat infot nagu näiteks nimetus, tootekood, miinimum- ning maksimumkogused jms. Täpsem kirjeldus on leitav lisadest (vt Tabel 8). Lisaks on kasutajal võimalik määrata konkreetne artikkel mingisse kategooriasse. Ka kategooriad ning alamkategooriaid on võimalik luua eraldi (vt Tabel 12).

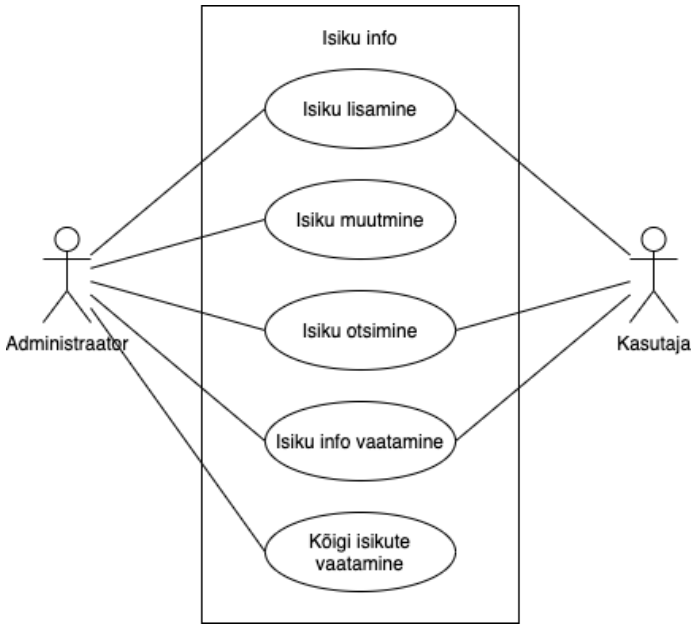
Varem lisatud artikleid on kasutajal võimalik otsida ning nende infot (omadused, laoseis, liikumised jne) vaadata. Nimetatud tegevused on täpsemalt kirjeldatud lisades (vt Tabel 9). Lisaks on ühel rollil võimalik vaadata korraga kõiki artikleid ning neid sorteerida ja filtreerida. See annab hea ülevaate kõikidest laos olevatest artiklitest ning nende laoseisust. (vt Tabel 13)

Joonis 1. Artikli kasutusmallide mudel.

Lisaks artiklitele võimaldab süsteem hallata ka isikuid. Isikuna võib süsteemis käsitleda nii eraisikuid kui ka ettevõtteid. Isikut on võimalik määrata arvete peal üheks osapooleks, kus teine osapool on tarkvara kasutav ettevõte ise. Allpool oleval joonisel (vt Joonis 2) on välja toodud kasutusjuhud, mis on seotud otseselt isikuga.

Tarkvara võimaldab isikute lisamist ning muutmist. Isiku kohta on võimalik salvestada tema kohta käivat teavet nagu näiteks nime, registrikoodi, erinevat kontaktinfot (e-maili aadress, telefoni number, aadress) ning makseinfot (vt Tabel 7).

Eelnevalt sisestatud isikut on võimalik otsida ning tema kohta käivat infot vaadata. Lisaks on administraatoril võimalik vaadata ühes kohas kõiki isikuid korraga. (vt Tabel 7)



Joonis 2. Isiku kasutusmallide mudel.

Eelnevalt kirjeldati artiklite ning isikutega seotud kasutusjuhte. Allpool oleval joonisel (vt Joonis 3) on kujutatud kasutusmallid ning rollid, mis on seotud arvetega.

Laomoodul võimaldab kasutajatel salvestada kahte tüüpi arveid - ostu- ja müügiarveid. Dokumendi kohta saab salvestada erinevat teavet, näiteks dokumendi numbrit, esitamise kuupäeva, maksetähtaega, tasumise infot jms. Lisaks on võimalik varem loodud dokumenti vaadata, printida ning mõnel kasutajal ka eksportida. Samuti saab dokumente importida. Täpsem kirjeldus on leitav töö lisadest (vt Tabel 10).

Lisaks eelnevalt kirjeldatud teabele on võimalik mistahes arve seostada konkreetse isikuga. Selle jaoks tuleb kas otsida juba eksisteerivat isikut või lisada uus.

Ostu- või müügiarve eesmärk on kas osta või müüa tooteid. Järelikult peab olema võimalik arvele lisada ostetavaid või müüdavaid artikleid. Laosüsteemi eesmärgiks on loomulikult ka artiklite laoseisu jälgimine. Seega on oluline, et ostu- või müügiarvele sisestatud rea alusel ka laoseis vastavalt suureneks või väheneks (vt Tabel 11).

Artikleid on võimalik lisada arvele ridadena, kus igal real on määratud artikkel, kogus, hinnad ning käibemaksu protsent. Artikli lisamiseks on võimalik kas otsida juba olemasolevat artiklit andmebaasist või luua uus artikkel. Kuna vahel on vaja lisada arvele ka laoväliseid tooteid, siis peab olema võimalik lisada arvele ka ridu, mis ei ole seotud laoartikliga. (vt Tabel 11)

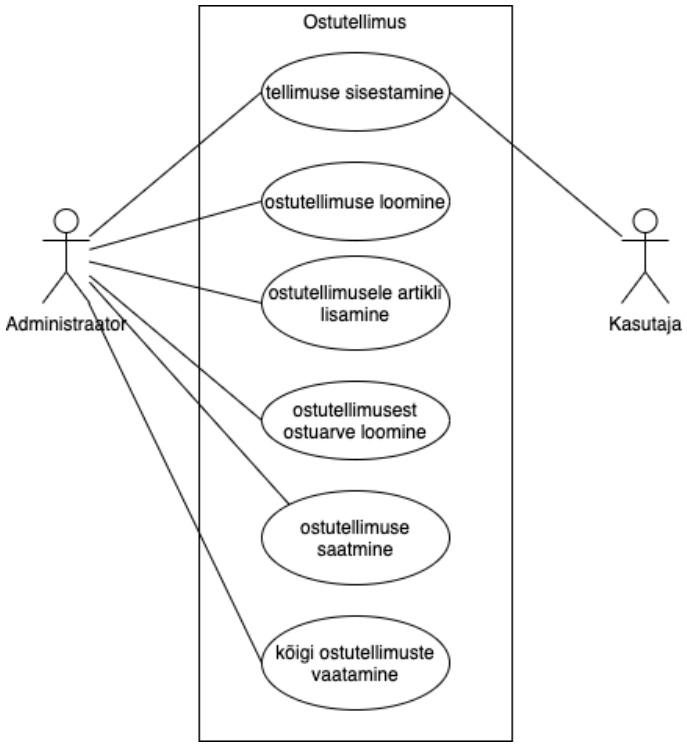
Joonis 3. Arvete kasutusmallide mudel.

Eelnevalt kirjeldati kuidas on võimalik kaupa osta ja müüa. Selleks, et oleks kaupa mida müüa on vahel vaja kaupa ka juurde tellida. Järgneval joonisel (Joonis 4) on kujutatud kasutusmallid, mis on seotud ostutellimusega.

Kauba tellimise aluseks on nõudlus. Tarkvara kasutajal on võimalik sisestada tellimus mingile kaubale mingis koguses kui tal tekib töö käigus vajadus selle artikli järele ning selle artikli laoseis ei ole piisav. (vt Tabel 15) Samuti võib tellimuse sisestada ka süsteem ise, kui ta tuvastab, et konkreetse kauba laoseis on alla sellele määratud miinimumkoguse (vt Tabel 8).

Sisestatud tellimuste alusel on võimalik administraatoril luua ostutellimus, millele lisada ridadena erinevate artiklite tellimused ning määrata ostutellimusele isikute hulgast tarnija. Kõiki ostutellimusi on võimalik administraatoril näha ühes kohas. See annab hea ülevaate nii aktiivsetest tellimustest kui ka nende ajaloost. (vt Tabel 15)

Loodud ostutellimuse saab administraator edastada tarnijale. Kauba saabumisel on võimalik luua ostutellimusest ostuarve, et tarnitud kaubad laos arvele võtta (vt Tabel 15).



Joonis 4. Ostutellimuse kasutusmallide mudel.

2.2.2 Tegevusskeemid

Eelnevas peatükis loodi kasutusmallide mudelid, mis andsid hea ülevaate laosüsteemi funktsionaalsustest kasutaja vaatest. Selleks, et süsteemi paremini mõista oleks vaja teada ka seda, et kuidas ja mis järjestuses teatud protsessid toimuvad. Sellest tulenevalt loob autor mudeli ühest süsteemi olulisemast protsessist kasutades UML-i poolt defineeritud

tegevusskeemi. Loodava skeemi sisendiks on kasutajalood (vt Lisa 2), ekspertintervjuust saadud info ning eelnevas alampeatükis loodud kasutusmallide mudelid (vt 2.2.1).

Kui kasutusjuhud (*use case*) näitavad süsteemi üldistatud käitumist kirjeldades erinevaid üksikuid tegevusi, siis tegevusskeemid võimaldavad täpselt näidata, kuidas süsteem mingi kindla eesmärgi täidab. Selleks võimaldavad need modelleerida erinevaid tegevusi, tingimusi ning tegevuste järjekorda. [4]

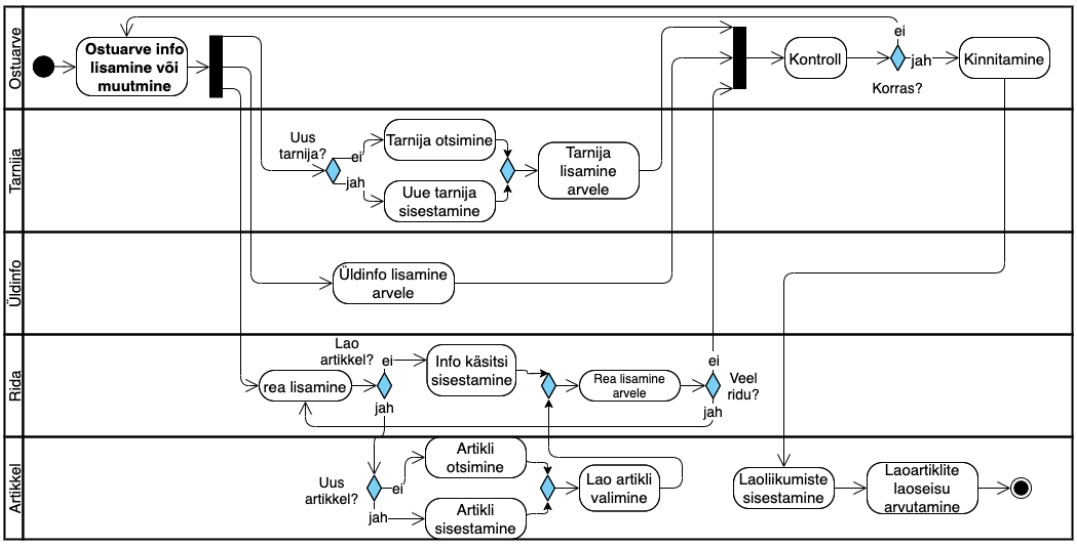
Järgmisel joonisel (vt Joonis 5) on mudel, mis kirjeldab ostuarve sisestamise protsessi. Protsessi algtingimuseks on loodud ostuarve mustand ning lõpptingimuseks kinnitatud ostuarve, mille tulemusena on arve alusel ostetud kaupade laoseis vastavalt arve ridadele suurenenud.

Kui ostuarve mustand on loodud, siis on kasutajal võimalik lisada mistahes järjekorras kolme erinevat tüüpi informatsiooni:

1. Tarnija info - kes antud arve väljastas;
2. Arve üldine info - dokumendi number, esitamise kuupäev, maksetähtaeg jms (vt Tabel 10);
3. Arve read - kauba/teenuse nimetus, seos laoartikliga (valikuline), kogus, allahindluse protsent, käibemaks (vt Tabel 11).

Kui kõik eelnev teave on ostuarvele sisestatud, siis sellele järgneb andmete korrektsuse kontroll. Kui kontrolli käigus leitakse mõni ebakõla, siis on võimalik sisestatud teavet muuta. Juhul kui kontrolli käigus ühtegi viga ei leita, siis on võimalik ostuarve kinnitada. Kinnitamise tulemusel sisestab süsteem automaatselt laoliikumised, mis dokumenteerivad lattu saabunud kaupade kogused. Kusjuures liikumised sisestatakse ainult laoartiklite kohta, sest laoväliste artiklite üle laoseisu arvestust ei peeta (vt Tabel 11).

Kirjeldatud protsess on väga sarnane müügiarve sisestamise protsessile. Seega ei pea autor vajalikuks modelleerida müügiarve sisestamise protsessi.



Joonis 5. Ostuarve sisestamise tegevusskeem.

2.3 Järeldused

Selles peatükis uuriti hetkel kasutatavat laosüsteemi. Esmalt kirjeldati praegust lahendust üldiselt. Seejärel loodi kasutusmallide mudelid, et välja selgitada põhilised tegevused, mida selle tarkvara kasutajad seda kasutades teevad. Lõpuks loodi mudel süsteemi ühe olulisema protsessi kirjeldamiseks. Selles peatükis kogutud teabe alusel on võimalik leida turult lõputöö probleemi lahendavad kandidaadid või luua uus laosüsteem. Mahu kokkuhoidmiseks ei modelleeritud selles peatükis kõiki kasutusjuhte ning tegevusi mis laomooduliga seotud on. Täiendav info on lõputöö lisades (vt Lisa 2) kasutuslugudena välja toodud.

Lähtudes kogutud teabest ning loodud mudelitest võib öelda, et laomooduli keskseks olemiks on kaup ehk artikkel. Kõik kirjeldatud tegevused on kas otseselt või kaudselt seotud artikliga:

- Tarnijate või klientide haldus - tarnija müüb kaupa, klient ostab kaupa;
- Laoartiklite haldamine - erinevate artiklite info salvestamine andmebaasi ning artiklite info vaatamine;
- Ostu- või müügiarvete haldamine - nende dokumentide alusel vastavalt kas ostetakse või müüakse kaupa.

Eelnevate punktidega on kirjeldatud laosüsteemi põhifunktsionaalsused, sest kõigi muude tegevuste (kirjeldatud peatüki jooksul kui ka lisades) sisendiks on nende tegevuste

väljundid. Näiteks ei ole võimalik artikli liikumise aruannet luua kui kauba ost ja müük ei ole dokumenteeritud.

Lisaks selgus, et laomoodulil on kahte tüüpi kasutajaid - administraator ning tavakasutaja. Administraatoril on õigus teha kõiki tegevusi kuid tavakasutajal on lubatud ainult kindlad tegevused.

2.4 Skoop

Vastavalt käesolevas peatükis kogutud infole paneb autor siinkohal paika lõputöö skoobi, ehk mida on plaanis saavutada. Kuna lõputöö eesmärgiks on kas luua või leida probleemi lahendav laosüsteem, siis töö skoop tuleb määrata lähtudes mõlemast võimalikust lahendusest.

Juhul kui lõputöös leitakse probleemi lahendav valmissüsteem ning otsustatakse selle kasutamise kasuks, siis tuleks leitud lahendus integreerida majandustarkvaraga sellises mahus, et oleksid täidetud peatükis 2.2 kirjeldatud nõuded.

Juhul kui lõputöös selgub, et mõistlik oleks luua täiesti uus laosüsteem, siis tuleks luua prototüüp, mis täidaks kasutajalugudes kirjeldatud eepikuid:

- E-1 Isikute haldus (vt Tabel 7),
- E-2 Lao artiklite haldus (vt Tabel 8),
- E-3 Lao artikli otsing (vt Tabel 9),
- E-4 Ostu- ning müügiarvete haldus (vt Tabel 10),
- E-5 Ostu- ning müügiarvete ridade haldus (vt Tabel 11),
- E-6 Lao artikli kategooriad (vt Tabel 12),
- E-7 Kõigi artiklite kuvamine (vt Tabel 13).

Eelnevalt nimetatud eepikud peaksid olema realiseeritud piisavas mahus nii, et laoarvestuse pidamine koos artiklite ning arvete haldamisega oleks võimalik. Lisaks peaks loodav prototüüp autoriseerima kõiki tegevusi nii kasutaja kui ka organisatsiooni põhiselt.

3 Saadaolevad lahendused

Nüüd, kui süsteemi olulised funktsionaalsed nõuded on teada on võimalik leida ning analüüsida erinevaid saadaolevaid lahendusi. Järgnevas peatükis uuritakse kolme erinevat turul saadaolevat lahendust, mis võiksid lõputöös käsitletavat probleemi lahendada. Esmalt kirjeldatakse laosüsteemi sobivuse hindamise metoodikat ning seejärel tutvustatakse ning analüüsitakse erinevaid lahendusi. Lisaks uuritakse lühidalt muid Eestis kasutatavaid või saadaolevaid lahendusi.

3.1 Metoodika

Selleks, et mõista kas mingi turul saadaolev laosüsteem on sobilik lõputöös käsitletava probleemi lahendamiseks tuleb uurida selle süsteemi erinevaid aspekte. Käesolevas peatükis uuritakse erinevate laotarkvarade vastavust esitatud nõuetele ning seda kui palju iga lahenduse eest maksta tuleb.

Selleks, et hinnata laosüsteemi vastavust nõuetele on autor koostanud hindamise tabeli (vt Tabel 16) vastavalt peatükis 2.2 kirjeldatud teabele. Kõik olulised nõuded on tabelis grupeeritud ning igale grupile määratakse hinne viiepalliskaalal. Kokku on võimalik ühel lahendusel saada 40 punkti. Selline hindamine annab arvulise indikatsiooni kandidaadi vastavusest nõuetele ning seetõttu on see hea metoodika lahenduse sobivuse kontrollimiseks ning erinevate lahenduste võrdlemiseks.

Lisaks nõuetele vastavusele on oluline ka konkreetse lahenduse maksumus. Maksumuse hindamise juures tuleb silmas pidada, et ettevõtte poolt pakutava majandustarkvara keskmise kliendi kuutasu on 50 eurot.

3.2 Turul saadaolevad lahendused

Brian Barry on ühes veebiartiklis öelnud, et: "Veebist laohaldussüsteemi (*inventory management system*) otsides võib leida sadu erinevaid kaubanduslikult kättesaadavaid süsteeme. Üldise otsingu tulemusena võib leida kõikvõimalike spetsialiseeritud tarkvarasid, mis ei pruugi sobida igaühele.". [1]

Selleks, et leida nende sadade võimalike lahenduste hulgast sobivaid kandidaate otsis autor eelkõige neid lahendusi, mida oleks võimalik suuremal või vähemal määral teise tarkvaraga liidestada. Leitud kandidaatide hulgast valis autor kolm tarkvara, mida tasuks edasi uurida vastavalt peatükis 3.1 kirjeldatud metoodikale. Kuna lõputöö probleemi lahendamiseks on vaja leitud tarkvara teise tarkvaraga liidestada, siis vastavuse hindamisel arvestati ainult nende funktsionaalsustega, mida on võimalik kasutada läbi programmiliidese ehk API (*application programming interface*).

3.2.1 SHARK WMS Cloud

SHARK WMS Cloud on pilvepõhine laohaldustarkvara (*warehouse management system*), mis võimaldab dokumenteerida ning planeerida järgnevaid tööprotsesse:

- kaupade vastuvõttu,
- kaupade hoiustamist (koos automaatse hoiustamise planeerimisega),
- kaupade tarnet,
- kaupade loendamist (inverteerimine),
- kaupade laosisest täiendamist,
- üldist haldust. [5]

Et hinnata uuritava tarkvara sobivust probleemi lahendamiseks funktsionaalsuse poolest kasutab autor eelnevalt kirjeldatud metoodikat. Autori poolt loodud vastavuse hindamise tabel on töö lisades (vt Tabel 17).

Uuritav tarkvara sai vastavalt kirjeldatud metoodikale 18 punkti 40-st. Arvestades kesist punktisummat võib järeldada, et antud tarkvara ei ole kõige parem lahendus töös käsitletavale probleemile. Mõned suuremad puudujäägid on:

- Isikute haldus puudub ning salvestatava teabe hulk piiratud;
- Artiklite otsing toimib ainult artikli numbri järgi;
- Artiklite liikumisi ei näe;
- Ostu- ja müügiarve kohta salvestatava teabe hulk on piiratud. (vt Tabel 17)

Tarkvara on tellimuspõhine ning selle kasutamise eest tuleb igal kasutajal maksta 520 eurot kolme kuu eest (~172.3 ühes kuus), millele lisandub käibemaks [6]. Kui ettevõtte poolt pakutava majandustarkvara kasutamise eest tuleb ühel kasutajal maksta keskmiselt

50 eurot kuus, siis vaadeldava laohaldustarkvara hind on üle kolme korra kallim. Seega ei sobi antud tarkvara ka hinna poolest lõputöö probleemi lahendamiseks.

3.2.2 Megaventory

Megaventory on pilvepõhine laohaldussüsteem (*inventory management system*), mis sobib frantsiisiettevõtetele, jae- ja hulgimüügiga tegelevatele ettevõtetele ning tootmisettevõtetele. [7]

Nimetatud tarkvara võimaldab näiteks:

- Dokumenteerida ja hallata kaupade ostu ja müüki;
- Hallata mitmeid asukohtasid, ladusid või poode;
- Hallata ning korraldada kauba vedu;
- Genereerida erinevat statistikat;
- Hallata ja dokumenteerida tootmisprotsesse. [7]

Et hinnata uuritava tarkvara sobivust probleemi lahendamiseks funktsionaalsuse poolest kasutab autor peatükis 3.1 kirjeldatud metoodikat. Autori poolt loodud vastavuse hindamise tabel on töö lisades (vt Tabel 18).

Vastavalt kirjeldatud metoodikale on antud tarkvara hindeks 28 punkti 40-st. Sellisest punktisummast võib järeldada, et uuritav tarkvara sobiks, küll mõningate mööndustega lõputöös käsitletavat probleemi lahendama. Mõned esinevad puudujäägid on:

- Arve tasumise infot ei ole võimalik salvestada;
- Dokumentide import ja eksport ei ole võimalik;
- Tegevuste piiramine rollipõhiselt puudub. (vt Tabel 18)

Puuduvad funktsionaalsused oleks teoreetiliselt võimalik juurde arendada lokaalselt. Näiteks oleks võimalik tasumise info lisamise funktsionaalsuse luua lokaalselt ning hoida seda infot kohalikus andmebaasis, kuid see oleks täiendav kulu.

Megaventory kasutamise eest tuleb igal kliendil maksta 150 USD kuus, millele lisandub käibemaks. Hinnas sisaldub lisaks tarkvara kasutusõigusele viis kasutajakontot, 20 asukohta, 20000 toodet ning 20000 klienti. [7] Tarkvara kuutasu eurodes on 126.69 eurot, kasutades lõputöö kirjutamise hetkel kehtinud kurssi. Arvestades ettevõtte poolt pakutava majandustarkvara keskmine klient maksab kasutusõiguse eest 50 eurot iga kuu, siis

uuritav tarkvara on üle kahe korra kallim. Sellist hinda on tõenäoliselt nõus maksma vaid üksikud kliendid.

3.2.3 Zoho Inventory

Zoho Inventory on pilvepõhine laohaldustarkvara, mis võimaldab kasutajatel teha selliseid toiminguid nagu näiteks:

- artiklite haldus ning jälgimine,
- mitme lao haldus,
- müügi- ja ostuarvete haldus,
- kaupade tarne,
- tarnijate haldamine,
- statistika genereerimine,
- teiste tarkvaradega integreerimine. [8]

Et hinnata uuritava tarkvara sobivust probleemi lahendamiseks funktsionaalsuse poolest kasutab autor eelnevalt kirjeldatud metoodikat. Autori poolt loodud vastavuse hindamise tabel on töö lisades (vt Tabel 19).

Zoho Inventory tarkvara sai autori poolt kirjeldatud metoodika alusel 37 punkti 40-st. Peaaegu maksimaalsest punktisummast võib järeldada, et uuritav tarkvara võiks lõputöö probleemi lahendamiseks sobida küll. Mõned üksikud puudujäägid on:

- Artikli otsing toimib ainult artikli numbri järgi;
- Artikli laoliikumisi ei näe;
- Artikleid ei saa filtreerida ega sorteerida. (vt Tabel 19)

Tarkvara kasutusõiguse eest tuleb igal kliendil maksta 49 eurot kuus, millele lisandub käibemaks [9]. Hinnas sisaldub muuseas 1500 tellimust kuus, kaks ladu, kümme kasutajat [9]. Ettevõtte poolt pakutav majandustarkvara keskmine klient maksab kasutusõiguse eest ühes kuus peaaegu sama summa. See hind on tõenäoliselt enamikele ettevõtte poolt pakutava majandustarkvara klientidele vastuvõetamatu.

3.3 Võrdlus

Eelnevalt uuriti lähemalt kolme erinevat turul saadaolevat laohaldussüsteemi ning hinnati nende sobivust lõputöö probleemi lahendamiseks. Vastavalt peatükkides 3.2.1, 3.2.2 ning 3.2.3 kirjutatud teabele lõi autor tabeli, mille abil saab lihtsasti võrrelda kolme nimetatud peatükkides uuritud laohaldussüsteemi (vt Tabel 1).

Tabelist selgub, et halvim tarkvara lõputöös käsitletava probleemi lahendamiseks on Shark WMS Cloud, sest selle funktsionaalsuse hinnang (18/40-st) on kõige madalam ning seejuures on selle kuutasu kõige kõrgem.

Teisel kohal on tarkvara Megaventory, mis sai funktsionaalsuse eest hinde 28/40. Mõningate mööndustega sobiks nimetatud tarkvara lõputöö probleemi lahendama, kuid see tähendaks ettevõtte jaoks täiendavaid kulutusi. Megaventory kasutusõiguse eest makstav kuutasu on küllaltki suur, kui võtta arvesse, et ettevõtte poolt pakutava majandustarkvara kasutav keskmine klient peab tasuma kasutusõiguse eest 50 eurot kuus.

Parim lahendus kolmest uuritud tarkvarast oleks Zoho Inventory, mis sai oma funktsionaalsuse eest 37 punkti 40-st. Kõrgest punktisummast võib järeldada, et nimetatud tarkvara sobiks töös käsitletavat probleemi lahendama väga hästi. Ka hinna poolest on nimetatud tarkvara parim kõigist kolmest võrreldud tarkvarast.

Tabel 1. Turul saadaolevate lahenduste võrdlus.

	Hinnang funktsionaalsusele	**Kuutasu**
Shark WMS Cloud	18/40	~172.3 eurot + KM
Megaventory	28/40	~126.69 eurot + KM
Zoho Inventory	37/40	49 eurot + KM

3.4 Kaasnevad probleemid

Eelnevalt uuriti lähemalt kolme turul saadaolevat laohaldussüsteemi ning leiti nende seast lõputöös käsitletava probleemi lahendamiseks parim. Enne sellise lahenduse kasutuselevõtmist tuleks uurida ka sellega kaasnevaid probleeme.

Mõned autori arvates olulisemad probleemid, mis kaasneksid mõne SaaS (*software as a service*) põhise laohaldussüsteemi kasutuselevõtuga on:

- andmeid töötlevad kolmandad osapooled [10],
- teenusepakkuja kontrollib koodibaasi ning tarkvara funktsionaalsusi [11],
- teenusepakkuja võib lõpetada teenuse pakkumise [12].

Kuna andmeid (sealhulgas isikuandmeid) töötlevad kolmandad osapooled, siis ei saa kunagi kindel olla, kui turvaliselt neid säilitatakse. Seega võib mõne sellise lahenduse kasutuselevõtuga minna vastuollu mõne õigusaktiga. Lisaks ei pruugi olla teada kuidas teenusepakkuja andmeid varundab ning seetõttu võivad olulised andmed kaotsi minna. [10] Andmete turvalisus on oluline majandustarkvara pakkuva ettevõtte jaoks ning seetõttu on see üsna suur probleem mis takistab mõne muu SaaS lahenduse liidestamist enda tarkvaraga.

Kuna SaaS lahenduse koodibaasi kontrollib seda arendav osapool, siis ei pruugi olla võimalik soovitud funktsionaalsuste lisamine sellesse. Samuti võib arendaja lisada selliseid funktsionaalsusi mida kasutajal vaja ei lähe. [11] Ettevõtte jaoks on see suur probleem, sest see ei võimalda laosüsteemi kohandada vastavalt enda klientide nõudmistele. Kui ettevõte ei suuda tarnida klientide poolt nõutud lahendusi, siis võib klient lihtsalt loobuda pakutava tarkvara kasutamisest.

Kui SaaS teenusepakkuja peaks lõpetama teenuse pakkumise, siis tuleb majandustarkvara pakkuval ettevõttel asuda koheselt uut lahendust otsima. See tähendab aga suuri ning potentsiaalselt ka ootamatuid kulusid. [12]

3.5 Muud Eestis kasutatavad lahendused

Järgnevas peatükis uurib lühidalt muid Eestis kasutatavaid või pakutavaid lahendusi ning hindab nende sobivust käesoleva lõputöö probleemi lahendamiseks. Esmalt uuritakse kahte varasemalt kaitstud laoteemalist lõputööd ning seejärel ühte muud Eestis pakutavat lahendust.

3.5.1 Varasemad lõputööd

Sarnase temaatikaga kuid erineva probleemiga lõputöid on kirjutatud ka varem. Autor uurib lähemalt kahte hiljutisemat TTÜ-s kaitstud laosüsteemi temaatikaga lõputööd ning teeb kindlaks, kas need võiksid käesoleva lõputöö probleemi lahendamiseks kasvõi osaliselt sobida.

Kõige hiljutisem lao temaatikaga töö on Kai Kabini 2020-nda aasta kevadel kaitstud magistritöö "Laoarvestuse analüüs tootmisettevõtte näitel" [13]. Nimetatud magistritöö eesmärgiks oli "kirjeldada tootmisettevõtte laoarvestuse tarkvara rätsepalahenduse jaoks funktsionaalsed ja mittefunktsionaalsed nõuded ning koostada andmebaasi ja arhitektuuri kavandid" [13]. Töös käsitleti konkreetsele tootmisettevõttele omaseid tööprotsesse nagu:

- ostutegevus,
- tootmisplaani koostamine,
- tootmistegevus / tootmise seire,
- valmistoodangu lõppkontroll,
- valmistoodangu üleandmine valmistoodangu lattu,
- väljastamine laost / kohaletoimetamine. [13]

Kuna käesoleva lõputöö eesmärgiks ei ole leida laosüsteemi tootmisettevõttele ning tootmisprotsesside toetamine ei ole vastavalt peatükile 2 üldse vajalik, siis eelnevalt kirjeldatud magistritöö tulemus ei lahenda käesolevas lõputöös käsitletavat probleemi.

Andreas Jürimäe on 2017-ndal aastal kaitsnud bakalaureusetöö "Rahvatantsuansambli laohaldustarkvara", mille eesmärk oli "lihtsustada ansambli tööd muutes ansambli lao halduse lihtsamaks ja selgemaks" [14]. Andreas Jürimäe poolt loodud rakendus võimaldab rahvatantsuansamblitel hallata ansambli rühmasid ning selle inventari rahvatantsurõivaste näol ning selle põhifunktsionaalsuseks on rõivaste või rõivakomplektide väljastamine ning vastuvõtmine [14].

Eelnevalt kirjeldatud lahendus on üsna spetsiifiline keskendudes rahvatantsuansambli inventari jälgimisele. Nimetatud töö kasutusjuhud erinevad oluliselt peatükis 2 kirjeldatud laosüsteemile, sest A. Jürimäe poolt loodud lahendus on keskendunud artiklite (rõivaesemete) rendile ning mitte ostule ja müügile [14]. Seega ei sobi eelnevalt nimetatud bakalaureuse töö tulemus käesoleva lõputöö probleemi lahendamiseks.

3.5.2 Muud turul saadaolevad lahendused

Lisaks eelnevalt kirjutatud lõputöödele on Eestis saadaval ka muid lahendusi. Autor toob siinkohal välja neist ühe ning kirjeldab seda lühidalt. Loomulikult on Eestis saadaval rohkem erinevaid lahendusi, kuid suur osa neist on käesoleva lõputöö probleemis

kirjeldatud ettevõtte lahendusega liiga sarnaseid ning autor otsustas neid siinkohal mitte välja tuua.

Näiteks võiks lõputöö probleemi lahendada laotarkvara Taavi Ladu, mis "on mõeldud müügitegevuse organiseerimiseks firmades, kes tegelevad hulgi - ja jaemüügiga." [15]. Vastavalt nimetatud tarkvara kasutusjuhendis [16] olevale infole võib järeldada, et see võiks funktsionaalsuse poolest probleemi lahendada küll. Kuid ei juhendis [16], ega laoprogrammi tutvustuses [15] ei leia autor infot liidestamise kohta. Seega ei ole tõenäoliselt võimalik antud lahendust teiste tarkvaradega piisavas mahus liidestada ning seetõttu ei sobi see lõputöös käsitletava probleemi lahendamiseks.

3.6 Kokkuvõte

Selles peatükis uuriti kolme erinevat turul saadaolevat laohaldussüsteemi ning hinnati nende vastavust lõputöös käsitletava probleemi lahendamiseks. Analüüsi tulemusel selgus, et pilvepõhine laohaldustarkvara Zoho Inventory sobiks funktsionaalsuse poolest probleemi lahendama väga hästi (vt 3.2.3). Nimetatud tarkvara kasutusõiguse eest tuleks igal kliendil tasuda 49 eurot kuus, millele lisandub käibemaks (vt 3.2.3). Arvestades seda, et ettevõtte poolt pakutava majandustarkvara keskmine kasutaja maksab kasutuseõiguse eest 50 eurot kuus on tegemist siiski üsna kalli lahendusega. Sellist hinda oleksid tõenäoliselt nõus maksma ainult mõned kliendid ning seetõttu ei oleks antud tarkvara integreerimine majanduslikult otstarbekas.

Lisaks uuriti peatükis 3.5 põgusalt kahte hiljutisemat TTÜ-s kaitstud laohaldussüsteemi temaatikaga lõputööd kuid uuritud töödes loodud lahendused olid liiga spetsiifilised ning ei sobi käesolevas lõputöös käsitletava probleemi lahendamiseks. Lisaks kirjeldati samas peatükis ka ühte muud Eestis saadaolevat lahendust ning leiti, et ka see ei ole probleemi lahendamiseks sobiv.

Peatükis kirjeldati ka mõne SaaS tarkvara kasutuselevõtuga kaasnevaid probleeme. Autori arvates on suurimateks probleemideks andmete turvalisus, koodibaasi mitteomamine ning ebakindlus SaaS pakkuja tuleviku osas.

Võttes arvesse tarkvara hinna ning selle kasutuselevõtuga kaasnevad probleemid arvab autor, et mõistlikum on sobiv laosüsteem nullist arendada.

4 Arendusplatvormi valik

Eelmises peatükis leidis autor, et lõputöös käsitletava probleemi lahendamiseks oleks mõistlik luua uus laosüsteem. Esimese sammuna uue süsteemi poole tuleks välja selgitada vahendid mida kasutada. Esmalt tuleks uurida kuidas loodavat süsteemi kõrgetasemeliselt struktureerida ning seejärel leida sobivad vahendid süsteemi loomiseks.

Selles peatükis leitakse sobiv programmeerimiskeel, raamistik ning andmebaas. Lisaks kirjeldatakse muid vahendeid mida plaanitakse kasutada. Tehtavad otsused tuginevad kirjandusallikatele, trendidele ning ettevõtte ja autori kogemustele. Otsuste tegemisel tuleb silmas pidada ka lõputöö ühte eesmärki - et lahendus oleks võimalikult universaalne ning lihtsasti liidestatav ka teiste tarkvaradega.

4.1 Üldine arhitektuur

Enne konkreetsete arendusvahendite leidmist on vaja välja selgitada kuidas tuleks loodavat tarkvara struktureerida. Valitud struktureerimisviis peab tagama loodava tarkvara universaalsuse, ehk seda peab olema võimalik lihtsasti liidestada ka teiste tarkvaradega.

4.1.1 Osa monoliidist

Erinevad arhitektuuri stiilid on võimalik jaotada kahte kategooriasse: monoliitsed ning hajutatud [17]. Nagu peatükis 2.1 öeldud, on ettevõtte poolt pakutava majandustarkvara kihilise arhitektuuriga, mis tähendab, et tegu on monoliitse süsteemiga.

Kihilise arhitektuuriga tarkvara puhul on tarkvara erinevad komponendid jaotatud erinevatesse kihtidesse nii, et iga kiht täidab mingit konkreetset rolli. Näiteks esitlusloogika kiht (*presentation layer*) vastutab kasutajaliidese kuvamise eest ning äriloogika kiht (*business layer*) ärireeglite täitmise eest. Seejuures on kihid üksteisest eraldatud. [17]

Seega oleks võimalik uus laosüsteem luua eraldi komponentidena, mis lisatakse praeguse tarkvara erinevatesse kihtidesse. Selline lahendus on kasutusel ka hetkel kuid lõputöös käsitletava probleemi lahendamiseks see siiski ei sobi.

Üks peamine mittesobivuse põhjus on programmeerimiskeel. Kuna ettevõtte majandustarkvara on loodud kasutades PHP-d, siis tuleb ka komponendid luua kasutades PHP-d. Sellisel juhul ei oleks loodavat lahendust võimalik kasutada mõnes teises projektis kus kasutatakse mõnda muud programmeerimiskeelt. Seega selliselt loodud lahendus ei oleks just kõige universaalsem.

Selline lahendus teeks loodava laosüsteemi veel vähem universaalsemaks, sest kihilise arhitektuuriga tarkvarad ei ole just kõige modulaarsemad [17]. See tähendab, et on vaja teha palju tööd selleks, et see lahendus mõne teise tarkvara koosseisus tööle saada. Seega ei oleks täidetud lõputöö eesmärk leida lihtsasti integreeritav lahendus.

4.1.2 Eraldi rakendus

Kuna uue laosüsteemi loomine majandustarkvara osana ei taga lahenduse universaalsust ning ei võimalda loodud lahendust tulevikus kergesti kasutada, siis tuleks uus süsteem eraldiseisva rakendusena.

Seega tuleks luua selline tarkvarasüsteem, mis on eraldiseisev kuid mis omab vastavaid liideseid, mis võimaldaksid teistel tarkvaradel sellega suhelda. Kaks võimalikku lahendust sellise liidese loomiseks on SOAP ning REST.

SOAP (*Simple Object Access Protocol*) on standardiseeritud protokoll mille järgi saavad veebiteenused andmeid vahetada kasutades andmeformaadina XML (*Extensible Markup Language*) märgistuskeelt [18].

REST (*Representational State Transfer*) on arhitektuuristiil mis võimaldab samuti veebiteenustel andmeid vahetada [19]. Nimetatud stiil ei kohusta andmevahetusel kasutama kindlat vormingut kuid tüüpiliselt kasutatakse selleks JSON (*JavaScript Object Notation*) andmevormingut [20]. Samuti ei ole REST puhul tegemist standardiga vaid tegemist on arhitektuuri stiiliga, mille kirjeldas Roy Thomas Fielding enda 2000-ndal aastal kirjutatud doktoritöös [21].

SOAP eeliseks võib pidada seda, et tegemist on standardiga, millesse on sisse ehitatud hulganisti reegleid mis tagavad andmete terviklikkuse ning turvalisuse [22]. REST eelisteks võib pidada selle lihtsust ning seda, et REST võimaldab andmevahetuseks erinevaid andmevorminguid [20].

Arvestades, et autor on varasemalt kokku puutunud ainult REST stiilis teenustega ning et REST stiilis teenust on lihtsam implementeerida, siis otsustab ta uue laosüsteemi loomisel luua selle programmiliidese kasutades just seda stiili. Kui luua uus laosüsteem REST stiilis veebiteenusena, siis see tagab tarkvarale ka universaalsuse. Seda seetõttu, et sellist veebiteenust on tulevikus võimalik liidestada mistahes teise tarkvaraga, mis toetab HTTP (*Hypertext Transfer Protocol*) protokolli.

4.2 Programmeerimiskeel

Eelnevalt leiti, et uus laosüsteem tuleks luua REST stiilis veebiteenusena. Järgnevalt leitakse veebiteenuse loomiseks sobiv programmeerimiskeel ning raamistik. Sobiva vahendi leidmiseks võrreldakse kolme erinevat õppe käigus kasutatud programmeerimiskeelt.

4.2.1 Metoodika

Selleks, et hinnata programmeerimiskeele sobivust veebiteenuse loomiseks tuleb uurida, kas selleks on olemas tugi ja vahendid. Näiteks kui leidub mõni raamistik, mis toetab veebiteenuse loomist, siis võib pidada antud keelt sobivaks.

Lisaks uuritakse ning võrreldakse valitud keelte või nende keelte raamistike populaarsust. Populaarsel programmeerimiskeelel on tõenäoliselt suur kogukond ning seetõttu ka suurem spetsialistide arv, kes oleksid suutelised tulevikus loodavat tarkvara täiendama.

Populaarsuse hindamiseks kasutatakse TIOBE indeksit, mis järjestab erinevad programmeerimiskeeled vastavalt otsingu tulemuste arvule kasutades erinevaid otsingumootoreid [23]. Kirjeldatud indeks annab indikatsiooni sellest, kui palju konkreetsest programmeerimisest veebis kirjutatakse kuid ei anna täpset ülevaadet sellest, et kui palju seda keelt reaalselt kasutatakse.

Kõige kaalukam faktor keele valimise juures on nii töö autori kui ka ettevõtte eelnev kogemus konkreetse keele kasutamises, sest mida rohkem on varasemaid kogemusi seda kiiremini uus lahendus valmib.

4.2.2 C#

Üks võimalik programmeerimiskeel mida veebiteenuse loomiseks võiks kasutada on C#. See on Microsofti poolt arendatud üldotstarbeline objektorienteeritud programmeerimiskeel [24].

Veebiteenuse loomiseks kasutades C# programmeerimiskeelt on võimalik kasutada ASP.NET raamistikku. ASP.NET on avatud lähtekoodiga platvormiülene raamistik, mis võimaldab luua veebirakendusi [25]. Seega on veebiteenuse loomine kasutades programmeerimiskeelt C# võimalik.

C# populaarsus TIEBO indeksi alusel on 4.20% 2020-nda aasta detsembri seisuga [26].

Autor hindab enda kogemust antud programmeerimiskeele kasutamises viiepallisüsteemis hindega 3. Ettevõttes pole varasemalt C# kasutatud, seega hinnatakse ettevõtte kogemust hindega 0.

4.2.3 Java

Teine õppekava läbimisel õpitud programmeerimiskeel, mis sobiks veebiteenuse loomiseks on Java. Java on Sun Microsystemsi poolt arendatud platvormiülene objektorienteeritud programmeerimiskeel, mida on võimalik kasutada erineva suuruse ja keerukusega tarkvarade loomiseks [27].

Java abil REST stiilis veebiteenuse loomiseks on võimalik kasutada mitmeid erinevaid raamistikke ning teeke nagu näiteks JAX-RS või Spring MVC [28]. Näiteks JAX-RS on Java API, mis lihtsustab REST stiilis rakenduste arendamist [29] samas Spring MVC on raamistik veebirakenduste loomiseks [30]. Seega on olemas eeldused veebiteenuse loomiseks kasutades Java programmeerimiskeelt.

Java populaarsus TIEBO indeksi alusel on 12.53% 2020-nda aasta detsembri seisuga [26].

Autor hindab enda kogemust antud programmeerimiskeele kasutamises viiepallisüsteemis hindega 1. Ettevõttes pole varasemalt Java programmeerimiskeelt kasutatud, seega hinnatakse ettevõtte kogemust hindega 0.

4.2.4 PHP

Kolmas võimalik programmeerimiskeel veebiteenuse loomiseks on PHP. PHP on nii staatiliste kui ka dünaamiliste veebilehtede ning -rakenduste loomiseks kasutatav serveripoolne skriptimiskeel [31].

Kuna veebirakendus, mis on kirjutatud kasutades PHP-d töötab niikuinii veebiserveris [32], siis põhimõtteliselt ei ole veebiteenuse loomiseks vaja ühtegi raamistikku kasutada. Sellest hoolimata on olemas erinevad raamistikud mis toetaksid REST stiilis veebiteenuse loomist. Mõned raamistikud, mille kasutamist tasuks kaaluda on näiteks Slim, Lumen või Laravel [33]. Seega on olemas eeldused veebiteenuse loomiseks kasutades PHP skriptimiskeelt.

PHP populaarsus TIEBO indeksi alusel on 2.12% 2020-nda aasta detsembri seisuga [26].

Autor hindab enda kogemust PHP kasutamises viiepallisüsteemis hindega 4. Ettevõttes peamiselt kasutatav programmeerimiskeel ongi PHP, seega hinnatakse ettevõtte kogemust hindega 5. Seejuures on ettevõtte poolt pakutav majandustarkvara samuti kirjutades kasutades PHP-d.

4.2.5 Valik

Eelnevalt uuriti kolme erinevat programmeerimiskeelt. Allpool olevas tabelis (vt Tabel 2) on kolme uuritud keele võrdlus. Tabelist selgub, et kõige populaarsem keel on Java ning kõige vähem populaarsem on PHP. Seejuures omab autor ning ettevõte kõige vähem kogemust Java kasutamises ning kõige rohkem kogemust PHP kasutamises. Lisaks selgus, et kõigi programmeerimiskeelte abil on võimalik luua veebiteenuseid.

Tabel 2. Keelte võrdlus. Populaarsus TIEBO indeksil 2020-nda aasta detsembri seisuga [26].

	Hinnang autori kogemusele (0-5)	**Hinnang ettevõtte kogemusele (0-5)**	**Populaarsus (TIEBO indeks)**	**Sobib veebiteenuse loomiseks**
C#	3	0	4.20%	jah
Java	1	0	12.53%	jah
PHP	4	5	2.12%	jah

Kuna kõik võrreldud keeled on küllaltki populaarsed ning kõik võimaldavad luua veebiteenuseid, siis otsustab autor siinkohal teha valiku vastavalt enda ning ettevõtte kogemustele. Arvestades seda, et nii autoril kui ka ettevõttel on kõige rohkem kogemust PHP kasutamises ning ka seda, et ettevõtte poolt pakutav majandustarkvara on kirjutatud kasutades PHP-d, otsustab autor et kõige mõistlikum on ka uus laosüsteem luua kasutades just PHP-d.

4.2.6 Raamistik

Tarkvararaamistik võimaldab lihtsustada ning kiirendada tarkvara arendamist, sest see hoolitseb madala taseme funktsionaalsuse eest ning seeläbi võimaldab arendajal keskenduda kõrgema taseme funktsionaalsuste loomisele [34]. Mõned olulised omadused, mida uue laosüsteemi loomiseks sobiv tarkvararaamistik peaks autori arvates omama on:

- Marsruutide (*routes*) ning neile vastavate kontrollerite defineerimise funktsionaalsus, sest see lihtsustab REST stiilis veebiteenuse loomist;
- Sõltuvuste süstimise (*dependency injection*) funktsionaalsus, kuna see võimaldab vähendada kõrgetasemeliste komponentide vahelist otsest sõltuvust kasutades selleks erinevaid abstraktsioone ning seeläbi lihtsustab erinevate komponentide taaskasutamist [35];

Muid funktsionaalsusi ei pea autor hetkel oluliseks, sest ta plaanib uue laosüsteemi loomisel hoida selle äriloogika raamistikust võimalikult eraldi. Nii on võimalik tulevikus kasutatav raamistik vähese vaevaga välja vahetada, kui selleks peaks vajadus tekkima.

Kolm populaarsemat PHP raamistikku REST stiilis veebiteenuse loomiseks on Slim, Lumen ning Laravel [33] ning autor omab kogemust nende kõigi kasutamises. Lisaks on

kõigil kolmel olemas ülalpool nimetatud omadused [36], [37], [38]. Kuna kõik kolm raamistikku sobivad autori arvates uue laosüsteemi loomiseks ning autor kavatseb hoida uue süsteemi äriloogika raamistikust võimalikult eraldi, siis otsustab autor kasutada Lumen raamistikku.

4.3 Andmebaas

Eelnevalt valitud skriptimiskeel PHP võimaldab kasutada 17 erinevat andmebaasi juhtimissüsteemi [39]. Autor uurib sellest hulgast kolme erinevat lahendust, mis on töö kirjutamise hetkel kõige populaarsemad avatud lähtekoodiga andmebaasi juhtimissüsteemid: PostgreSQL, MySQL ning SQLite [40].

4.3.1 PostgreSQL

PostgreSQL on avatud lähtekoodiga objekt-relatsioonilise andmebaasi juhtimissüsteem, mis on California Ülikoolis Berkeleys (*University of California, Berkley*) aastatel 1986 kuni 1993 arendatud projekti POSTGRES edasiarendus [41]. PostgreSQL töötab klient/server arhitektuuril [42]. Vastavalt litsentsile on PostgreSQL tasuta kasutamiseks [43].

Relatsioonilises andmebaasis seostatakse andmeid erinevate tabelite vahel kasutades selleks seotavates tabelites olevaid sama semantikaga andmevälju [44]. Objekt-relatsioonilistes andmebaasides salvestatakse andmeid samamoodi nagu relatsioonilises andmebaasis kuid andmeid on võimalik käsitleda kui objekte [45].

Andmete käitlemiseks PostgreSQL andmebaasis kasutatakse päringukeelt SQL [46]. Uuritav andmebaasi juhtimissüsteem vastab SQL-i 2016-nda aasta standardile üsna hästi - 177-st standardi järgi nõutud funktsionaalsusest on täidetud 170 [47].

Mõned PostgreSQL tugevamad küljed on:

- Vastavus SQL standardile nagu eelnevas lõigus kirjeldati;
- Laiendatav - võimalik defineerida uusi andmetüüpe [46] ning luua ise teeke, et andmebaasi juhtimissüsteemi protseduure lisada või muuta [48];
- Arendamisel on peetud tähtsaks andmete terviklikkust [46].

PostgreSQL mõned negatiivsed omadused on näiteks:

- Kohati aeglasem kui MySQL [49];
- Andmete replikatsiooni on keeruline seadistada [49].

Autor hindab nii enda kui ka ettevõtte PostgreSQL kasutamise kogemust viiepalliskaalal hindega 1.

4.3.2 MySQL

MySQL on avatud lähtekoodiga relatsioonilise andmebaasi juhtimissüsteem, mida saab kasutada nii klient/server süsteemina kui ka manustatuna (*embedded*) [50]. MySQL-i loodi 1995-ndal aastal ning hetkel arendab ja haldab seda Oracle [51]. MySQL-i on lubatud vastavalt GPLv2 litsentsile tasuta kasutada [52].

Ka MySQL andmebaasis kasutatakse andmete käsitlemiseks SQL päringu keelt. MySQL ei vasta SQL-i standardile täielikult, kuid võimaldab käivitada serverit erinevates režiimides, kui peaks tekkima vajadus täpsemalt standardit järgida [53].

MySQL parimad küljed on:

- Enim kasutatud andmebaasi juhtimissüsteem PHP rakendustes [54];
- Populaarseim avatud lähtekoodiga relatsioonilise andmebaasi juhtimissüsteem [40];
- Kiirus [50];
- Erinevad viisid andmete replikatsiooniks [55].

Mõned MySQL puudused on:

- Vastab SQL-i standardile puudulikult [56];
- Aeglasti arenev [49].

Autor hindab nii enda kui ka ettevõtte MySQL kasutamise kogemust viiepalliskaalal hindega 5.

4.3.3 SQLite

Erinevalt eelnevalt nimetatud kahest andmebaasi juhtsüsteemist ei ole SQLite puhul tegemist klient/server mudelit järgiva andmebaasiga vaid see andmebaas töötab manustatuna, ehk andmebaasi jaos ei ole eraldi serveri protsessi tarvis. [57] SQLite on

avatud lähekoodiga ning tasuta kasutamiseks [58]. SQLite vastab SQL-i standardile hästi - puudu on vaid mõned funktsionaalsused [59].

SQLite kasutamise eelised on näiteks:

- Kompaktsus - teegi suurus vähem kui 600KiB [60];
- Jõudlus [57];
- Ei vaja serverit [61];
- Laialt kasutatud [62];
- Ei vaja konfigureerimist [63].

Mõned puudused on näiteks:

- Samaaegselt saab andmebaasis olevaid kirjeid muuta või lisada vaid üks kasutaja [64];
- Pole võimalik skaleerida [64].

Autor hindab nii enda kui ka ettevõtte SQLite kasutamise kogemust viiepalliskaalal hindega 1.

4.3.4 Valik

Käesolevas peatükis uuriti lähemalt kolme PHP poolt toetatud andmebaasi juhtimissüsteemi. Järgnevalt selgitatakse välja parim süsteem, mida uue laosüsteemi loomiseks kasutada. Arvestada tuleb sellega, et andmebaasi hakkavad korraga kasutama paljud kliendid ning silmas tuleb pidada ka lõputöö eesmärki luua laosüsteem mis oleks universaalne ning lihtne üles seada.

SQLite sobiks uue süsteemi loomiseks hästi sellepoolest, et see ei vaja töötamiseks serverit [54] ning seda ei ole vaja konfigureerida [63]. See võimaldaks loodavat laosüsteemi kerge vaevaga üles seada ning seeläbi vähendaks vajaliku töö hulka laosüsteemi integreerimisel. Aga kuna SQLite ei võimalda samaaegselt mitmel kasutajal andmebaasi kirjutada [64] ning andmebaasi ei saa hästi skaleerida [64], siis ei pruugi see lahendus sobida üha kasvava kasutajaskonnaga tarkvarale.

MySQL-i eelised PostgreSQL ees on tema kiirus, paremad vahendid andmete replikatsiooniks ning tema populaarsus (vt 4.3.2). Kuna MySQL on lõputöö kirjutamise hetkel populaarseim avatud lähtekoodiga relatsioonilise andmebaasi juhtimissüsteem

[40], siis tagab see loodavale tarkvarale universaalsuse, sest populaarsusest võib järeldada kui palju on spetsialiste kes on valmis andmebaasi üles seadma, haldama ning vajadusel muudatusi tegema. Lisaks on MySQL enim kasutatud andmebaasi juhtimissüsteem PHP rakendustes [54] ning nii ettevõttel kui ka töö autoril on enim kogemust just selle kasutamisel.

PostgreSQL eelised MySQL ees on parem vastavus SQL-i standardile, laiendatavus ning andmete terviklikkusele keskendumine (vt 4.3.1). Autori arvates on need eelised vähemtähtsamad kui MySQL-i omad, sest:

- MySQL vastab SQL-i standardile küll puudulikult, kuid selle nimel tehakse pidevalt tööd [65];
- Andmete terviklikust on võimalik heal tasemel tagada ka kasutades MySQL-i näiteks kasutades varundusmootorit InnoDB [66].

Kuna eelnevalt kirjeldatu põhjal on MySQL parem lahendus, siis autor otsustab uue laosüsteemi loomisel kasutada andmebaasi juhtimissüsteemina just seda.

4.4 Muud vahendid

Eelnevalt leiti uue laosüsteemi loomiseks sobiv programmeerimiskeel ning andmebaasi juhtimissüsteem. Käesolevas peatükis tutvustatakse muid vahendeid, mida autor plaanib kasutada.

4.4.1 Versioonihaldus

Versioonihaldussüsteemi ülesandeks on failide muutuste salvestamine ning see võimaldab tulevikus tagasi vaadata failide erinevatele versioonidele. Tarkvarasüsteemi mõttes tähendab see seda, et versioonihaldus võimaldab salvestada koodibaasi erinevaid versioone ning seejuures tulevikus vanemaid versioone ka kasutada. [67] Autor kasutab versioonihalduseks GIT-i, mis on Linuxi arenduskogukonna poolt arendatud versioonihaldussüsteem [68].

4.4.2 Arenduskeskond

Integreeritud arenduskeskkonna eesmärgiks on lihtsustada tarkvara loomist toetades arendajat erinevate funktsioonidega. Autor kasutab arenduskeskkonnana PHPStormi, mis on JetBrainsi poolt loodud tarkvara mis on loodud PHP projektide arendamiseks [69].

4.5 Kokkuvõte

Käesolevas peatükis selgitati välja vahendid, mida plaanitakse uue laosüsteemi loomiseks kasutada. Valiku tegemisel lähtuti kirjandusallikatest saadud infost, trendidest ning autori ja ettevõtte kogemustest. Samuti peeti valikute tegemisel oluliseks lõputöö ühte eesmärki - luua võimalikult universaalne ning lihtsasti liidestatav lahendus.

Esmalt selgus, et uus laosüsteem tuleks luua eraldiseisva rakendusena, mis omab teiste tarkvaradega suhtlemiseks vastavaid liideseid. Nii on võimalik loodavat lahendust ka tulevikus lihtsasti teiste tarkvaradega liidestada. Autor otsustas andmevahetuse liidese loomiseks kasutada REST stiili.

Seejärel võrdles autor kolme erinevat programmeerimiskeelt: C#, Java ning PHP. Uuriti iga keele sobivust veebiteenuse loomiseks, populaarsust ning autori ja ettevõtte varasemat kogemust nende kasutamisel. Kuna kõik keeled on sobivad veebiteenuse loomiseks, siis otsustas autor teha valiku enda ning ettevõtte kogemuse alusel ning valis uue süsteemi loomiseks skriptimiskeele PHP. Uue laosüsteemi loomiseks kasutatakse Lumen tarkvararaamistikku kuid seejuures hoitakse äriloogika raamistikust võimalikult eraldi.

Kolmandaks võrreldi kolme erinevat andmebaasi juhtimissüsteemi: PostgreSQL, MySQL ning SQLite tuues välja iga süsteemi head ning halvad omadused. Autor otsustas valida MySQL-i, sest see oli töö kirjutamise hetkel populaarseim avatud lähtekoodiga andmebaasijuhtimissüsteeme ning seda kasutatakse PHP-s kirjutatud rakenduste loomisel enim.

Lisaks kirjeldati lühidalt muid vahendeid, mida autor plaanib uue laosüsteemi loomisel kasutada.

5 Uue lahenduse loomine

Nüüdseks on teada kuidas loodav laosüsteem töötab ning mis arendusvahendeid plaanitakse kasutada. Prototüübi loomiseks kasutatakse programmeerimiskeelena PHP-d kasutades seejuures Lumen raamistikku ning andmebaasijuhtimissüsteemina MySQL-i. Järgnevas peatükis antakse ülevaade uue lahenduse prototüübi loomise protsessi olulistest punktidest. Sealjuures kirjeldatakse loodava tarkvarasüsteemi arhitektuuri ning andmemudelit. Peatüki lõpus testitakse loodud prototüüpi kasutades käsitsi testimise metoodikat.

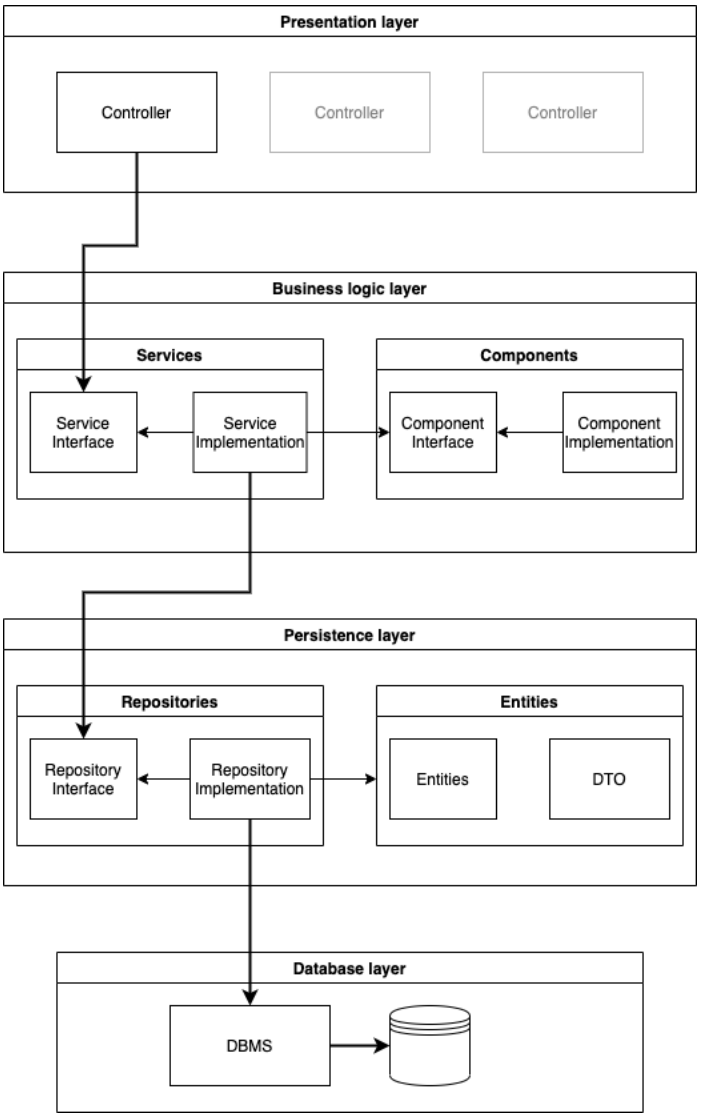
5.1 Arhitektuur

Uue laosüsteemi prototüübi loomiseks kasutatakse kihilist arhitektuuri stiili. Sellise arhitektuuri stiili puhul jaotatakse tarkvara komponendid erinevatesse kihtidesse nii, et igal kihil on kindel ülesanne. Kusjuures iga kiht on üldjuhul teadlik ainult temast allpool olevast kihist [17].

Autor valis selle arhitektuuri stiili, sest:

- Seda on küllaltki lihtne arendada [70];
- See võimaldab programmikoodi hästi eraldada [70];
- Seda võib pidada heaks valikuks juhul kui täpne arhitektuuri stiil pole veel välja selgitatud [17].

Allpool oleval joonisel (vt Joonis 6) on kujutatud loodava laosüsteemi arhitektuuri kus on kuvatud kõik erinevad kihid ning on üldistatult näidatud kuidas käib suhtlus kihtide vahel.



Joonis 6. Autori koostatud loodava süsteemi arhitektuuri mudel.

Järgnevalt kirjeldab autor kõiki joonisel kujutatud kihte ning annab ülevaate nende ülesannetest loodava laosüsteemi kontekstis.

5.1.1 Esitluskiht

Kõige ülemine kiht loodavas tarkvaras on esitluskiht (*presentation layer*). Kihilise arhitektuuriga tarkvaras on esitluskihi või esitlusloogikakihi ülesandeks lõppkasutajaga suhtlemine [71]. Nii on see ka autori poolt loodavas prototüübis. Esitluskihi komponentideks on kontrollerid. Kontrollerite ülesandeks on tarkvara lõppkasutaja päringute vastuvõtmine, päringute esmane valideerimine ning seejärel ülesande delegeerimine temast allpool asuvale äriloogikakihile. Saades äriloogikakihilt vastuse ülesande täitmise õnnestumise või mitteõnnestumise kohta, tagastab see vastava info lõppkasutajale.

5.1.2 Äriloogikakiht

Kihilises arhitektuuris on äriloogikakihi ülesandeks äriprotsesside ning loogika täitmine [71]. Nagu öeldud, delegeerib esitluskiht kasutaja päringule vastava ülesande äriloogikakihile (*business logic layer*), mis valideerib ning töötleb seda. Kui kasutaja päringut ei saa mingil põhjusel täita, siis tagastatakse esitluskihile asjakohane info.

Kasutaja päringu täitmiseks võivad selles kihis olevad komponendid suhelda ka omavahel. Selles kihis toimub andmeobjektide loomine või töötlemine ning selle põhilised komponendid on teenused (*services*). Igal teenusel äriloogikakihis on oma eesmärk. Näiteks võib üheks teenuseks olla laoartikli info pärimise teenus, mille ülesandeks on vastavalt kasutaja päringule artikli leidmine, täiendava info leidmine ja töötlemine ning seejärel info tagastamine. Seega saab esitlusloogikakiht kasutaja päringule vastava ülesande delegeerida sobivale teenusele.

5.1.3 Andmekiht

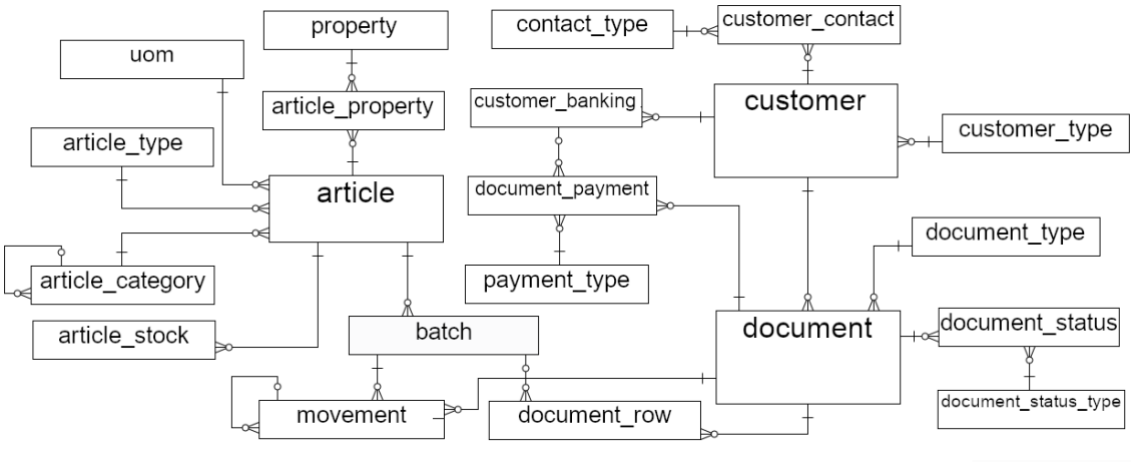
Andmekihi ülesandeks kihilise arhitektuuriga tarkvaras on andmete salvestamine ning lugemine andmebaasist [71]. Andmeobjektide salvestamiseks, muutmiseks või kustutamiseks pöördub äriloogikakiht temast allpool oleva kihi - andmekihi (*persistence layer*) poole, mis suhtleb andmebaasi juhtimissüsteemiga.

Andmebaasiga suhtlus toimub läbi hoidlate (*repository*), mis kujutavad endast abstraktsiooni andmebaasi ning rakenduse vahel. Hoidlal on hulk meetodeid kindla andmeobjektiga seotud andmebaasitoimingute tegemiseks ning rakenduses on võimalik nende meetodite abil neid toiminguid teostada. [72] Näiteks on loodavas süsteemis olemas hoidla lao artiklite jaoks, mis omab meetodeid artikli lisamiseks, muutmiseks, kustutamiseks, otsimiseks jne. Äriloogikakiht saab mõne nimetatud toimingu teostamiseks pöörduda hoidla poole, mis seejärel hoolitseb andmebaasiga suhtluse eest ning tagastab päringule vastava tulemuse.

5.2 Andmebaasi struktuuri loomine

Loodava laosüsteemi andmebaasi kavandamiseks kasutab autor olemi-suhte diagrammi metoodikat. See on enim levinud metoodika andmemudelite koostamiseks, mille abil on võimalik kirjeldada süsteemi olemeid, vaateid, seoseid, olemite atribuute, piiranguid jms [44].

Allpool oleval joonisel (vt Joonis 7) on kujutatud autori loodud laosüsteemi andmemudel infoloogilisel tasemel, ehk olemite seesmist struktuuri ei ole kuvatud [44]. Samuti on nimetatud jooniselt ära jäetud autentimise ning autoriseerimisega seotud olemid. Olemite sisemine struktuur ning muu täpsem info on leitav lõputöö lisades (vt Lisa 7).



Joonis 7. Autori koostatud laomooduli andmemudel infoloogilisel tasemel.

Loodud andmemudel koosneb 21-st olemist ning on sobiv peatükis 2.4 kirjeldatud funktsionaalsusega laosüsteemi loomiseks, sest see võimaldab salvestada lao artikleid, artiklite kategooriaid, isikuid ning ostu- ja müügiarveid ning nende kohta käivat muud informatsiooni.

5.3 Andmemudelid

Nagu töö peatükis 5.1.2 öeldud, töötleb loodava laosüsteemi äriloogikakiht andmeobjekte. Andmeobjektid on andmemudelile vastavad objektid, mis hoiustavad erinevat tüüpi infot. Need objektid esitavad tüüpiliselt elulisi olemeid või andmebaasi tabeleid. [73] Näiteks loodava laosüsteemi kontekstis on olemas andmemudelid, mis kirjeldavad laoartikleid, isikuid, arveid, arve ridu jne. Uue artikli loomisel luuakse äriloogikakihis objekt vastavalt artikli andmemudelile, omistatakse objektile uue artikli omadused ning edastatakse loodud objekt andmekihile salvestamiseks.

5.3.1 ORM raamistik

Kuna teavet kõigi olemite kohta hoiustatakse andmebaasis tabelite kirjetena ning äriloogikakiht käitleb ärireeglite ning -protsesside täitmisel andmeobjekte, siis oleks vaja mingit mehhanismi, mis tõlgendaks andmebaasi kirjed andmeobjektideks ning vastupidi. Selline tehnoloogia on tuntud kui ORM (*object-relational-mapping*). ORM (*object-*

relational-mapping) on metoodika, mis võimaldab ühest süsteemist saadud info teisendada teisele süsteemile arusaadavale kujule ning vastupidi [74].

ORM raamistikud võimaldavad ORM metoodika kasutamist. Autor kirjeldab järgnevalt kahte PHP ORM-i: Eloquent ning Doctrine.

Eloquent on ORM raamistik, mis on üks osa Lumen raamistikust [75]. Genet Schneider on enda 2017-ndal aastal kirjutatud bakalaureusetöös Eloquent mudelite kohta öelnud, et: "Rakenduse mudelid User ja Post käituvad teatud mõttes teenustena, mille kaudu saab andmebaasi tabelitest pärida andmeid." [76]. Teenusteks neid mudeleid siiski pidada ei saa, sest tegemist on lihtsalt mudelitega, mis on teadlikud andmebaasist. Nimelt töötab Eloquent kasutades "aktiivse kirje" (*active record*) mustrit [75]. Aktiivse kirje mustri puhul kirjeldab andmeobjekt otseselt andmebaasi kirje sisu ning seejuures omab ka meetodeid andmebaasiga suhtlemiseks [77].

Selline lähenemine ei sobi autori poolt valitud arhitektuuri stiiliga (vt 5.1), sest see annaks äriloogikakihile otsese ligipääsu andmebaasile ning siis ei oleks andmekihil mingit eesmärki. Lisaks puuduvad Eloquent mudelitel rangelt defineeritud andmeväljad (*property*) ning andmevälja väärtuse küsimiseks või muutmiseks kasutatakse "*_get*" ning "*_set*" meetodeid [78], mida PHP nimetab "maagilisteks meetoditeks" [79]. Sellised mudelid võivad tekitada arendajates segadust või põhjustada vigasid.

Doctrine on PHP ORM raamistik, mis töötab kasutades "andme kaardistaja" (*data mapper*) mustrit [80]. Andme kaardistaja muster teeb andmeobjekti andmebaasist täielikult sõltumatuks teisendades andmebaasi kirje töömälus olevaks objektiks ning vastupidi [77]. Doctrine olemid on lihtsad PHP objektid ehk POPO-d (*plain old PHP objects*), mille andmeväljad on konkreetse andmemudeli poolt ära määratud [80]. Need kaks aspekti võimaldavad kihilise arhitektuuriga tarkvaral jääda kihiliseks - rakenduse äriloogikakiht töötleb POPO-sid seejuures teadmata kas ja kuidas neid salvestatakse ning andmekihis paiknev ORM hoolitseb andmete pärimise, uuendamise jms eest. Seetõttu otsustas autor prototüübi loomisel kasutada Doctrine ORM-i.

5.3.2 DTO objektid

Martin Fowler defineerib DTO (*data transfer object*) kui: "Objekt, mis kannab andmeid eri protsesside vahel eesmärgiga vähendada meetodi kutseid" [81]. Loodava laosüsteemi

kihilise arhitektuuri kontekstis kasutab autor neid objekte eesmärgiga vahetada infot erinevate kihtide vahel. See võimaldab äriloogikakihil esitluskihile edastada või sellelt vastu võtta infot, mis ei vasta rangelt andmemudelitele. Näiteks võib DTO koosneda andmeväljadest, mis pärinevad erinevatelt andmeobjektidelt ning mida on võibolla modifitseeritud.

5.4 Turvalisus

Kuna loodavat laosüsteemi hakkavad kasutama paljud erinevad kliendid, siis on turvalisus oluline. Seetõttu peab loodav prototüüp võimaldama kasutajapõhist autentimist ning kasutaja- ja organisatsioonipõhist autoriseerimist. Autentimine on protsess, mille käigus tehakse kindlaks kasutaja identiteet ning autoriseerimine on protsess, mille käigus selgitatakse kas kasutajal on õigus mingit toimingut teostada [82].

5.4.1 Autentimine

Kasutaja autentimiseks REST stiilis veebiteenuses on mitmeid võimalusi. Autor otsustas kasutada JWT baasil autentimist. JWT ehk JSON Web Token on standard mis defineerib kompaktse ning turvalise andmevahetuse viisi kasutades JSON (*JavaScript Object Notation*) objekte [83]. JWT koosneb kolmest osast: objekti päis, objekti sisu ning signatuur [83].

Autentimiseks tuleb kasutajal teha HTTP POST päring otspunktile *"/auth/login*". Päringu kehas peab olema kasutajanimi ning parool, mille alusel saab veebiteenus kasutaja isiku kindlaks teha.

Kui kasutaja autentimine õnnestub, siis luuakse ning tagastatakse kasutajale väärtus, mis kujutab endast signeeritud sõnumit JWT kujul ning mis sisaldab endas infot kasutaja ning JWT enda kohta. Sõnum on signeeritud kasutades mõnda krüpteerimisalgoritmi ning see tagab sõnumi terviklikkuse - kuna ainult veebiteenus teab krüpteerimisel kasutatud salajast võtit, siis ainult veebiteenus ise saab sõnumile õige allkirja arvutada [83]. Seetõttu ei ole kolmandatel osapooltel võimalik enda loodud JWT-it õigesti allkirjastada ning veebiteenus neid ei aktsepteeri.

Nüüd kui kasutajal on olemas JWT, mis on väljastatud ning allkirjastatud veebiserveri poolt ning mis tõestab kasutaja identiteeti, peab kasutaja iga järgneva päringuga edastama

veebiserverile sellesama väärtuse. Tüüpiliselt edastatakse JWT serverile lisades selle HTTP päringu päisesse autoriseerimise väljana [83]. Kasutaja poolt esitatud JWT põhjal saab veebiserver teha kindlaks kasutaja isiku või keelata tundmatul isikul ressursile ligipääsu.

JWT baasil autentimise implementeerimiseks kasutas autor Sean Tymoni poolt loodud "jwt-auth" paketti [84], mida on võimalik Composer paketihalduri abil paigaldada käsuga: "*composer require tymon/jwt-auth*". Paketi seadistamine dokumentatsiooni järgi on lihtne ning autor ei pea vajalikuks selle dokumentatsiooni dubleerimist käesolevasse töösse [84].

5.4.2 Autoriseerimine

Kasutaja identiteedi kindlaks tegemisel välistatakse andmete ligipääs kolmandatel isikutel. Kuna loodavat lahendust hakkavad kasutama erinevad majandustarkvara kasutavad organisatsioonid ning igal organisatsioonil võib olla palju erinevaid kasutajaid, siis on konfidentsiaalsuse tagamiseks vaja implementeerida ka autoriseerimine ehk õiguste kontrollimine. Loodavas laosüsteemis on vaja kasutaja tegevusi autoriseerida kahel viisil: kasutaja- ning organisatsioonipõhiselt.

Kasutajapõhiselt autoriseerimiseks on süsteemis defineeritud erinevad õigused ning igal kasutajal on hulk õiguseid. Kui kasutaja soovib teha mingit tegevust, siis kontrollitakse kas kasutajale on antud selleks õigus. [85] Näiteks, kui süsteemis on defineeritud õigus "artikli lisamine" ning see õigus on nõutud uue artikli lisamisel, siis kasutaja kellele sellist õigust lisatud ei ole, seda tegevust ka teha ei saa. Autor autoriseerib kasutaja päringud esitluskihis olevate kontrollerite sees kasutades selleks Lumen raamistiku võimalusi. See ei ole küll esitluskihi ülesanne, kuid prototüübi loomiseks on see lahendus piisav.

Organisatsioonipõhise autoriseerimise mõte on see, et üks organisatsioon saab käidelda ainult enda andmeid ning mitte mingil juhul ei tohi olla ligipääsu mõne teise organisatsiooni andmetele. [85] Autor teeb loodava prototüübi andmekihi teadlikuks organisatsioonist. See võimaldab andmekihil andmebaasist pärida andmeid, mis on seotud ainult konkreetse organisatsiooniga. Selle tulemusel ei ole võimalik äriloogikakihil töödelda mõne muu organisatsiooni andmeid ega edastada neid läbi esitluskihi lõppkasutajale. Selleks, et organisatsioonipõhine autoriseerimine oleks

võimalik on vaja andmebaasi olemitele lisada väli, mis välisvõtmena (*foreign key*) viitab sellele organisatsioonile, kes on konkreetse kirje omanik.

Lumen raamistik võimaldab autoriseerimist ning autentimist teostada lihtsasti ning kõik vajalik on juba raamistikku sisse ehitatud [86]. Kahjuks vajab sisse ehitatud autoriseerimise ning autentimise funktsionaalsus töötamiseks Eloquent ORM-i [87], mida autor otsustas mitte kasutada.

Selleks, et saada raamistikusisene autoriseerimise ning autentimise funktsionaalsus tööle kasutades Eloquent ORM-i asemel Doctrine ORM-i on võimalik kasutada "Laravel Doctrine ACL" paketti [88]. Nimetatud paketi paigaldamiseks saab kasutada paketihaldurit Composer ning selle käsku "*composer require "laravel-doctrine/acl:1.0.*'"*" ning seadistamiseks on olemas lihtne dokumentatsioon [88].

5.4.3 Valitud lahenduste kitsaskohad

Eelnevalt autentimiseks ning autoriseerimiseks valitud vahendid on autori arvates head kuid nende kasutamisel on mõned kitsaskohad, kuhu peaks tähelepanu pöörama.

Nagu peatükis 5.4.1 öeldud, tuleb kasutajal enda autentimiseks teha HTTP päring veebiserverile ning päringu kehasse tuleb lisada kasutajanimi ning parool. Kui klient saadab veebiserverile päringu, siis on võimalik kolmandatel isikutel spetsiaalsete tööriistadega tehtud päringut näha ning seeläbi kasutaja kasutajanime ning parooli teada saada [89]. Selle probleemi lahendamiseks tuleks HTTP asemel kasutada HTTPS-i (*HTTP Secure*), mis võimaldab muuseas ka kliendi ning serveri vahelise suhtluse krüpteerimist [89]. Kui kasutaja autentimispäringus olev info on krüpteeritud, siis ei ole võimalik kolmandatel isikutel tehtuid päringuid lihtsa vaevaga lugeda ning seetõttu saab kasutaja ennast turvaliselt autentida.

Lisaks on peatükis 5.4.1 kirjeldatud, et kasutajale tagastatakse eduka autentimispäringu korral JWT, mille lisab kasutaja igale järgnevale päringule, et enda isikut tõestada. Teoreetiliselt on võimalik pahatahtlikul kolmandal kasutajal see JWT varastada ning seda ise kasutada, esitledes end tavalise kasutajana [90]. Selle vastu võiks aidata JWT eluea vähendamine [90].

5.5 Lahenduse kasutamine

Selleks, et laosüsteemi kasutada tuleb esmalt lõppkasutajal ennast autentida nii nagu on kirjeldatud peatükis 5.4.1. Seejärel on kasutajal võimalik teha erinevaid päringuid veebiteenusele. Nagu peatükis 5.1.1 öeldud, on prototüübi kasutajaliideseks esitluskihis asuvad kontrollerid, mis võtavad vastu kasutaja päringu, delegeerivad ülesande ning kõige lõpus tagastavad kasutajale saadud tulemuse.

5.5.1 REST otspunktid

Peatükis 4.1 otsustati, et laosüsteem tuleks luua kasutades REST stiili. REST stiili defineeris Roy Thomas Fielding enda 2000-ndal aastal kaitstud doktoritöös "*Architectural Styles and the Design of Network-based Software Architectures*", kus ta väitis muuseas, et REST stiili põhiliseks informatsiooni abstraktsiooniks on ressurss ning, et erinevad ressursid on kättesaadavad ettemääratud ning ühtlase liidese kaudu [21].

Seega on ka loodava laosüsteemi kasutajal võimalik käidelda erinevaid ressursse kasutades selleks ettemääratuid ning ühtlaseid liideseid - REST otspunkte. Kõik laosüsteemi prototüübis loodud otspunktid on välja toodud lõputöö lisades (vt Lisa 8). Näiteks, kui laosüsteemi kasutaja soovib laoartikleid kuvada, lisada, muuta või kustutada, siis tuleb tal kasutada selleks sobivat otspunkti (vt Tabel 3). Kui kasutaja teeb päringu mõnele otspunktile, siis veebiteenus teeb vastava toimingu ning tagastab kasutajale kindlaksmääratud ressursi.

Tabel 3. Laoartikli põhilised REST otspunktid.

HTTP meetod	**Ressursi aadress**	**Kirjeldus**
GET	/articles	tagastab laoartiklite kogumi
POST	/articles	loob uue laoartikli
GET	/articles/{id}	tagastab konkreetse laoartikli info
PUT	/articles/{id}	uuendab konkreetse laoartikli infot
DELETE	/articles/{id}	kustutab konkreetse laoartikli

5.5.2 Liidestamine majandustarkvaraga

Kuna loodava laosüsteemi näol on tegemist veebiteenusega mille kasutajaliideseks on REST otspunktid ning mis võtab vastu ja tagastab masinloetavat informatsiooni, siis

tegemist ei ole lahendusega, mida hakkab otseselt kasutama lõppkasutaja. Lõppkasutaja eest hakkab loodava laosüsteemiga suhtlema majandustarkvara ise tehes päringuid laosüsteemile kasutades HTTPS protokolli.

Kuna laosüsteemi liidestamine majandustarkvaraga ei kuulu käesoleva lõputöö skoopi, siis autor seda protsessi ei kirjelda.

5.6 Lahenduse testimine

Loodud laosüsteemi prototüübi testimiseks loob autor mõned testid, eesmärgiga kontrollida süsteemi põhiliste funktsioonide toimimist. Kuna loodud lahendus on küllaltki mahukas, siis autor ei plaani lõputöö käigus testida kogu loodud lahendust vaid selle olulisemaid osasid. Lõputöö lisades (vt Lisa 9) on viis testi, millest kolm kontrollivad loodud lahenduse turvalisust ning kaks kontrollivad laoarvestuse toimimist.

Allpool olevas tabelis (vt Tabel 4) on välja toodud viis testi ning nende tulemused. Nagu näha, siis kõikide testide tulemus on positiivne.

Tabel 4. Lahenduse testimise tulemus.

Testi number ja nimi	**Tulemus**
T01 - Autentimine (vt Tabel 22)	korrektne
T02 - Autoriseerimine - kasutaja õiguse kontrollimine (vt Tabel 23)	korrektne
T03 - Autoriseerimine - kasutaja organisatsiooni kontrollimine (vt Tabel 24)	korrektne
T04 - Artikli sisestamine (vt Tabel 25)	korrektne
T05 - Artikli ostmine ostuarvega (vt Tabel 26)	korrektne

Neid teste on küll vähe, kuid need annavad ülevaate loodud prototüübi põhifunktsionaalsuste toimimisest. Tulevikus tuleks neid teste samas stiilis rohkem koostada või testimine hoopis automatiseerida.

6 Saavutatud tulemus

Lõputöö eesmärgiks oli kas leida või luua laosüsteem, mis vastaks ettevõtte poolt esitatud nõudmistele ning mida oleks võimalik liidestada ka teiste tarkvaradega. Lõputöö tulemusena leiti, et Zoho Inventory laohaldussüsteem vastab enamusele ettevõtte nõuetele ning on liidestatav ka teiste tarkvaradega. Siiski osutus nimetatud süsteem üsna kalliks ning seetõttu ei oleks seda mõistlik kasutada. Lisaks kerkiksid mõne SaaS lahenduse kasutuselevõtuga erinevad probleemid, mille tulemusel võivad kunagi tekkida ootamatud kulutused.

Seega otsustati luua uus laosüsteem REST stiilis veebiteenusena. See otsus võimaldas täita lõputöö ühte eesmärki - luua lahendus mida saab liidestada ka teiste tarkvaradega, sest sellist veebiteenust on võimalik liidestada mistahes teise tarkvaraga, mis toetab HTTP protokolli kasutamist.

Uus laosüsteem otsustati luua kasutades skriptimiskeelt PHP ning andmebaasijuhtsüsteemi MySQL. Need on küllaltki populaarsed vahendid ning seetõttu ei tohiks tekkida probleeme spetsialisti leidmisel, kes oskaks loodud lahendust seadistada või vajadusel selles muudatusi teha. Lisaks võib PHP-d pidada heaks valikuks ka seetõttu, et ka ettevõtte poolt pakutav majandustarkvara on kirjutatud kasutades sama keelt, seega ei ole ettevõttel vaja leida eraldi spetsialiste kahe erineva süsteemi haldamise ja arendamise jaoks. Lõputöö tulemusena valmis prototüüp kasutades kihilist arhitektuuri stiili, mis võimaldab tarkvara erinevaid osasid suhteliselt hästi eraldada ning seetõttu peaks see olema lihtsasti mõistetav ka teiste arendajate poolt.

Loodud prototüüp vastab peatükis 2.4 kirjeldatud skoobile osaliselt. Loodud lahenduse turvalisus on tagatud hetkel piisaval tasemel - on implementeeritud kasutajate autentimine ning kasutaja- ja organisatsioonipõhine autoriseerimine.

Funktsionaalsete nõuete koha pealt on täielikult täitmata ainult isikute halduse funktsionaalsus (vt Tabel 7). Muud skoobis kirjeldatud funktsionaalsused on täidetud täpselt sellises mahus, et loodud süsteem oleks kasutatav ehk artiklite haldamine, ostmine ning müümine ja selle alusel laoseisu arvestamine on võimalik. Täitmata jäid peamiselt sellised funktsionaalsused, mis hõlmavad endas erinevate ressursside otsimist või kõigi sarnaste ressursside filtreerimist ning sorteerimist. Lisades (vt Lisa 8) on välja toodud

kõik prototüübis loodud otspunktid ning nende kirjeldused, mis näitavad täpselt, mis funktsionaalsused implementeeritud said.

Allpool olevas tabelis (vt Tabel 5) on välja toodud arvud, mis näitavad lõputöö kasutuslugudes kirjeldatud funktsionaalsuste täidetavust loodud prototüübi poolt. Nagu näha, siis prototüüp ei ole veel täielikult valmis, kuid 23 funktsionaalsust 36-st on kas osaliselt või täielikult implementeeritud. Siiski arvab autor, et tulemus on hea, sest suur osa prototüübist on valmis ning see töötab piisavalt heal tasemel.

Tabel 5. Loodud prototüübi nõuetele vastavus.

Eepik	**Täidetud**	**Osaliselt täidetud**	**Täitmata**
E-1 (vt Tabel 7)	0	0	5
E-2 (vt Tabel 8)	5	0	1
E-3 (vt Tabel 9)	2	0	3
E-4 (vt Tabel 10)	3	3	2
E-5 (vt Tabel 11)	5	2	2
E-6 (vt Tabel 12)	2	0	0
E-7 (vt Tabel 13)	0	1	0
Kokku	17	6	13

6.1 Edasised tegevused

Nagu eelnevalt selgus, siis loodud prototüüp ei vasta täielikult peatükis 2.4 määratud skoobile. Seega tuleb prototüüpi veel edasi arendada ning seejärel on võimalik see liidestada ettevõtte poolt pakutava majandustarkvaraga. Lõputöö probleemi loodud prototüüp veel ei lahenda kuid see on üks suur samm selles suunas. Autor toob järgnevalt välja veel mõned olulised sammud prototüübi edasisel arendamisel.

Lõputöös kontrollis autor prototüübi toimimist testides seda käsitsi. Tulevikus tuleks kindlasti kirjutada automaattestid, mis teeksid seda palju kiiremini ning põhjalikumalt. See võimaldab leida erinevaid vigu tarkvaras.

Prototüüp ei ole valmis rakendus. Seega tuleb enne uue laosüsteemi kasutuselevõttu implementeerida kõik nõutud funktsionaalsused. Sealhulgas on vaja luua

administreerimisliides, mis võimaldaks majandustarkvara administraatoritel läbi viia erinevaid toiminguid, mis on vajalikud et selle klientide töö toimiks sujuvalt.

7 Kokkuvõte

Lõputöö eesmärgiks oli leida või luua laosüsteem, mis vastaks ettevõtte poolt esitatud nõudmistele ning mis oleks täielikult liidestatav ka teiste tarkvaradega.

Selleks, et põhilist eesmärki täita oli vaja esmalt uurida täpsemalt ettevõtte praegust laosüsteemi. Selleks lõi autor erinevad UML-i käitumismudelid lähtudes lõputöö lähtetingimuseks olnud kasutuslugudest ning ekspertintervjuust saadud infost. Käitumismudelite loomine oli vajalik, sest see võimaldas paremini mõista ettevõtte poolt kasutatavat laosüsteemi. Lisaks kasutati seda infot turul saadaolevate lahenduste leidmiseks ning uue lahenduse loomiseks.

Vastavalt käitumismudelitele ning muule infole leiti kolm turul saadaolevat valmislahendust ning hinnati nende sobivust probleemi lahendamiseks. Leiti küll üks hea lahendus kuid selle kuutasu on liiga kõrge. Lisaks selgus, et mõne sellise valmislahenduse kasutuselevõtuga kaasneksid mõned tõsised probleemid.

Seega otsustati luua täiesti uus lahendus REST stiilis veebiteenuse näol kasutades selle loomiseks skriptimiskeelt PHP ning andmebaasijuhtimissüsteemi MySQL. Kuna loodud laosüsteemi prototüüp on eraldiseisev süsteem, mis omab REST stiilis liidest, siis on võimalik seda liidestada mistahes teise tarkvaraga, mis toetab HTTP protokolli. PHP ning MySQL on küllaltki populaarsed, seega ei tohiks olla probleem spetsialisti leidmisega, kes loodud laosüsteemi paigaldada suudaks.

Lõputöö tulemusena valmis laosüsteemi prototüüp kasutades kihilise tarkvara arhitektuuristiili ning mis omab REST stiilis andmevahetuse liidest. Loodud prototüüp on piisavalt funktsionaalne, et artiklite haldamine, ostmine ning müümine ja selle alusel laoseisu arvestamine oleks võimalik.

Kasutatud kirjandus

[1] B. Barry, „Top 6 Features to Look For In An Inventory Management System,“ [Võrgumaterjal]. Available: https://www.fcbco.com/blog/top-6-features-to-look-for-in-an-inventory-management-system. [Kasutatud 4 12 2020].

[2] G. Booch, J. Rumbaugh ja I. Jacobson, Unified Modeling Language User Guide, The, Second Edition, Addison-Wesley Professional, 2005.

[3] O. M. Group, „WHAT IS UML,“ [Võrgumaterjal]. Available: https://www.uml.org/what-is-uml.htm. [Kasutatud 4 12 2020].

[4] R. Miles ja K. Hamilton, Learning UML 2.0, O'Reilly Media, Inc, 2006.

[5] Logiware ApS, „SHARK WMS Specifications,“ [Võrgumaterjal]. Available: https://sharkwms.com/en/spec_shark_wms_cloud.html. [Kasutatud 4 12 2020].

[6] Logiware ApS, „SHARK WMS Cloud Pricing,“ [Võrgumaterjal]. Available: https://sharkwms.com/en/shark_prices.html. [Kasutatud 4 12 2020].

[7] Megaventory Inc, „Megaventory koduleht,“ [Võrgumaterjal]. Available: https://www.megaventory.com.

[8] Zoho Corporation Pvt. Ltd, „Inventory management software for growing businesses.,“ [Võrgumaterjal]. Available: https://www.zoho.com/inventory/features/. [Kasutatud 4 12 2020].

[9] Zoho Corporation Pvt. Ltd, „Zoho Inventory hinnakiri,“ [Võrgumaterjal]. Available: https://www.zoho.com/inventory/pricing/. [Kasutatud 4 12 2020].

[10] A. Sirianni, „Should I Use SAAS or Custom Software? The Pro's and Con's,“ [Võrgumaterjal]. Available: https://www.dcodegroup.com/blog/cloud-apps-vs-custom-software. [Kasutatud 4 12 2020].

[11] Loveleen, „Software As A Service vs Custom Build,“ 28 2 2017. [Võrgumaterjal]. Available: https://blog.leadformance.com/software-as-a-service-vs-custom-build. [Kasutatud 4 12 2020].

[12] A. Henson, „ Using SaaS software vs creating your own bespoke software,“ [Võrgumaterjal]. Available: https://foxsoft.net/using-saas-software-vs-creating-your-own-bespoke-software/. [Kasutatud 4 12 2020].

[13] K. Kabin, *Laoarvestuse analüüs tootmisettevõtte näitel,* TTÜ, 2020.

[14] A. Jürimäe, *Rahvatantsuansambli laohaldustarkvara,* TTÜ, 2017.

[15] Taavi Tarkvara OÜ, „Taavi Ladu,“ [Võrgumaterjal]. Available: https://www.taavi.ee/et/tooted/taavi-ladu/tutvustus. [Kasutatud 3 12 2021].

[16] Taavi Tarkvara OÜ, „Taavi Ladu juhend,“ [Võrgumaterjal]. Available: https://www.taavi.ee/images/juhendid/taavi_ladu.pdf. [Kasutatud 3 1 2021].

[17] M. Richards, Ford ja Neal, Fundamentals of Software Architecture, O'Reilly Media, Inc, 2020.

[18] Guru99, „SOAP Web Services Tutorial: Simple Object Access Protocol EXAMPLE,“ [Võrgumaterjal]. Available: https://www.guru99.com/soap-simple-object-access-protocol.html. [Kasutatud 4 12 2020].

[19] Codecademy, „What is REST?,“ [Võrgumaterjal]. Available: https://www.codecademy.com/articles/what-is-rest. [Kasutatud 4 12 2020].

[20] Guru99, „SOAP Vs. REST: Difference between Web API Services,“ [Võrgumaterjal]. Available: https://www.guru99.com/comparison-between-web-services.html. [Kasutatud 4 12 2020].

[21] R. T. Fielding, *Architectural Styles and the Design of Network-based Software Architectures.,* Irvine: University of California, 2000.

[22] Red Hat, Inc, „REST vs. SOAP,“ [Võrgumaterjal]. Available: https://www.redhat.com/en/topics/integration/whats-the-difference-between-soap-rest. [Kasutatud 4 12 2020].

[23] TIOBE Software BV, „TIOBE Programming Community Index Definition,“ [Võrgumaterjal]. Available: https://www.tiobe.com/tiobe-index/programming-languages-definition/. [Kasutatud 4 12 2020].

[24] Ecma International, *C# Language Specification, ECMA-334,* 2017.

[25] Microsoft, „What is ASP.NET?,“ [Võrgumaterjal]. Available: https://dotnet.microsoft.com/learn/aspnet/what-is-aspnet. [Kasutatud 4 12 2020].

[26] TIOBE Software BV, „TIOBE Index for December 2020,“ [Võrgumaterjal]. Available: https://www.tiobe.com/tiobe-index/. [Kasutatud 4 12 2020].

[27] A. Johari, „What Is Java? A Beginner’s Guide to Java and Its Evolution,“ 26 11 2020. [Võrgumaterjal]. Available: https://www.edureka.co/blog/what-is-java/. [Kasutatud 4 12 2020].

[28] J. Paul, „Why Use Spring to develop Java Web Services?,“ 20 8 2018. [Võrgumaterjal]. Available: https://hackernoon.com/why-use-spring-to-develop-java-web-services-ba0dcb2cafbf. [Kasutatud 4 12 2020].

[29] M. Hadley, „The Java API for RESTful Web Services (JAX-RS) -- Rapidly Build Lightweight Web Services,“ 6 2010. [Võrgumaterjal]. Available: https://www.oracle.com/technical-resources/articles/java/jax-rs.html. [Kasutatud 4 12 2020].

[30] M. Behler, „What is Spring MVC: @Controllers & @RestControllers,“ 22 11 2020. [Võrgumaterjal]. Available: https://www.marcobehler.com/guides/spring-mvc. [Kasutatud 4 12 2020].

[31] Guru99, „What is PHP? Write your first PHP Program,“ [Võrgumaterjal]. Available: https://www.guru99.com/what-is-php-first-php-program.html. [Kasutatud 4 12 2020].

[32] S. Suehring ja J. Valade, „How PHP Works,“ [Võrgumaterjal]. Available: https://www.dummies.com/programming/php/how-php-works/. [Kasutatud 4 12 2020].

[33] Slant.co, „What are the best PHP frameworks for building a RESTful API?,“ [Võrgumaterjal]. Available: https://www.slant.co/topics/6956/~php-frameworks-for-building-a-restful-api. [Kasutatud 4 12 2020].

[34] V. Singh, „What is Frameworks?,“ 9 4 2020. [Võrgumaterjal]. Available: https://hackr.io/blog/what-is-frameworks. [Kasutatud 26 12 2020].

[35] T. Janssen, „Design Patterns Explained – Dependency Injection with Code Examples,“ 19 6 2018. [Võrgumaterjal]. Available: https://stackify.com/dependency-injection/. [Kasutatud 27 12 2020].

[36] J. Lockhart, A. Smith, R. Allen, P. Bérubé ja Slim Framework Team, „Slim 4 Documentation,“ [Võrgumaterjal]. Available: https://www.slimframework.com/docs/v4/. [Kasutatud 27 12 2020].

[37] Laravel LLC, „Installation - Lumen - PHP Micro-Framework By Laravel,“ [Võrgumaterjal]. Available: https://lumen.laravel.com/docs/8.x. [Kasutatud 27 12 2020].

[38] Laravel LLC, „Installation - Laravel - The PHP Framework For Web Artisans,“ [Võrgumaterjal]. Available: https://laravel.com/docs/8.x. [Kasutatud 27 12 2020].

[39] The PHP Group, „PHP Manual - Database Extensions,“ [Võrgumaterjal]. Available: https://www.php.net/manual/en/refs.database.php. [Kasutatud 4 12 2020].

[40] solid IT gmbh, „DB-Engines Ranking of Relational DBMS,“ [Võrgumaterjal]. Available: https://db-engines.com/en/ranking/relational+dbms. [Kasutatud 4 12 2020].

[41] The PostgreSQL Global Development Group, „A Brief History of PostgreSQL,“ [Võrgumaterjal]. Available: https://www.postgresql.org/docs/current/history.html. [Kasutatud 4 12 2020].

[42] The PostgreSQL Global Development Group, „Architectural Fundamentals - PostgreSQL,“ [Võrgumaterjal]. Available: https://www.postgresql.org/docs/13/tutorial-arch.html. [Kasutatud 4 12 2020].

[43] The PostgreSQL Global Development Group, „License - PostgreSQL,“ [Võrgumaterjal]. Available: https://www.postgresql.org/about/licence/. [Kasutatud 4 12 2020].

[44] P. Raspel, „Kursuse Andmebaaside alused loengumaterjalid,“ [Võrgumaterjal]. Available: https://enos.itcollege.ee/~priit/1.%20Andmebaasid/1.%20Loengumaterjalid/. [Kasutatud 4 12 2020].

[45] B. Udaranga, „Introduction to PostgreSQL,“ 17 3 2019. [Võrgumaterjal]. Available: https://medium.com/@buddhimau/introduction-to-postgresql-71887c29982e. [Kasutatud 4 12 2020].

[46] The PostgreSQL Global Development Group, „What is PostgreSQL?,“ [Võrgumaterjal]. Available: https://www.postgresql.org/about/. [Kasutatud 4 12 2020].

[47] The PostgreSQL Global Development Group, „Appendix D. SQL Conformance - PostgreSQL dokumentatsioon,“ [Võrgumaterjal]. Available: https://www.postgresql.org/docs/current/features.html. [Kasutatud 4 12 2020].

[48] Packt, „PostgreSQL as an Extensible RDBMS,“ 3 3 2015. [Võrgumaterjal]. Available: https://hub.packtpub.com/postgresql-extensible-rdbms/. [Kasutatud 4 12 2020].

[49] M. Drake ja osterez, „SQLite vs MySQL vs PostgreSQL: A Comparison Of Relational Database Management Systems,“ 19 3 2019. [Võrgumaterjal]. Available: https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-management-systems. [Kasutatud 4 12 2020].

[50] Oracle Corporation, „What is MySQL?,“ [Võrgumaterjal]. Available: https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html. [Kasutatud 4 12 2020].

[51] „History of MySQL,“ [Võrgumaterjal]. Available: https://databasefriends.blogspot.com/2014/02/history-of-mysql.html. [Kasutatud 4 12 2020].

[52] Oracle Corporation, „Commercial License for OEMs, ISVs and VARs - MySQL,“ 7 2010. [Võrgumaterjal]. Available: https://www.mysql.com/about/legal/licensing/oem/. [Kasutatud 4 12 2020].

[53] Oracle Corporation, „The Most Important SQL Modes,“ [Võrgumaterjal]. Available: https://dev.mysql.com/doc/refman/8.0/en/sql-mode.html#sql-mode-important. [Kasutatud 4 12 2020].

[54] S. Verma, „Top 5 Databases for PHP Web Application Development,“ 19 5 2014. [Võrgumaterjal]. Available: https://www.veonconsulting.com/top-databases-for-php/. [Kasutatud 4 12 2020].

[55] Oracle Corporation, „Replication - MySQL,“ [Võrgumaterjal]. Available: https://dev.mysql.com/doc/refman/8.0/en/replication.html. [Kasutatud 4 12 2020].

[56] J. Mack, „Five Advantages & Disadvantages Of MySQL,“ 28 5 2014. [Võrgumaterjal]. Available: https://www.datarealm.com/blog/five-advantages-disadvantages-of-mysql/. [Kasutatud 4 12 2020].

[57] „About SQLite,“ [Võrgumaterjal]. Available: https://www.sqlite.org/about.html. [Kasutatud 4 12 2020].

[58] „SQLite Copyright,“ [Võrgumaterjal]. Available: https://www.sqlite.org/omitted.html. [Kasutatud 4 12 2020].

[59] „SQL Features That SQLite Does Not Implement,“ [Võrgumaterjal]. Available: https://www.sqlite.org/omitted.html. [Kasutatud 4 12 2020].

[60] „Size Of The SQLite Library,“ [Võrgumaterjal]. Available: https://www.sqlite.org/footprint.html. [Kasutatud 4 12 2020].

[61] „SQLite Is Serverless,“ [Võrgumaterjal]. Available: https://www.sqlite.org/serverless.html. [Kasutatud 4 12 2020].

[62] „Most Widely Deployed and Used Database Engine,“ [Võrgumaterjal]. Available: https://www.sqlite.org/mostdeployed.html. [Kasutatud 4 12 2020].

[63] „SQLite Is A Zero-Configuration Database,“ [Võrgumaterjal]. Available: https://www.sqlite.org/zeroconf.html. [Kasutatud 4 12 2020].

[64] „Appropriate Uses For SQLite,“ [Võrgumaterjal]. Available: https://www.sqlite.org/whentouse.html. [Kasutatud 4 12 2020].

[65] Oracle Corporation, „MySQL Standards Compliance,“ [Võrgumaterjal]. Available: https://dev.mysql.com/doc/refman/8.0/en/compatibility.html. [Kasutatud 4 12 2020].

[66] Oracle Corporation, „Introduction to InnoDB,“ [Võrgumaterjal]. Available: https://dev.mysql.com/doc/refman/8.0/en/innodb-introduction.html. [Kasutatud 4 12 2020].

[67] „Getting Started - About Version Control,“ [Võrgumaterjal]. Available: https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control. [Kasutatud 4 12 2020].

[68] „Getting Started - A Short History of Git,“ [Võrgumaterjal]. Available: https://git-scm.com/book/en/v2/Getting-Started-A-Short-History-of-Git. [Kasutatud 4 12 2020].

[69] JetBrains s.r.o., „PHPStorm: The Lightning-Smart PHP IDE,“ [Võrgumaterjal]. Available: https://www.jetbrains.com/phpstorm/. [Kasutatud 4 12 2020].

[70] M. Richards, Software Architecture Patterns, O'Reilly Media, Inc, 2015.

[71] A. F. Tappenden, T. Huynh, J. Miller, A. Geras ja M. Smith, *Agile Development of Secure Web-Based Applications,* Idea Group Inc, 2006.

[72] DevIQ, „Repository Pattern,“ [Võrgumaterjal]. Available: https://deviq.com/repository-pattern/. [Kasutatud 27 12 2020].

[73] „Application Architecture Guide - Chapter 11 - Business Layer Guidelines,“ [Võrgumaterjal]. Available: http://www.guidanceshare.com/wiki/Application_Architecture_Guide_-_Chapter_11_-_Business_Layer_Guidelines. [Kasutatud 4 12 2020].

[74] „NHibernate - ORM,“ [Võrgumaterjal]. Available: https://www.tutorialspoint.com/nhibernate/nhibernate_orm.htm. [Kasutatud 4 12 2020].

[75] Laravel LLC, „Eloquent,“ [Võrgumaterjal]. Available: https://laravel.com/docs/8.x/eloquent. [Kasutatud 4 12 2020].

[76] G. Schneider, *Populaarsemate PHP mikroraamistike võrdlus ja nende kasutamine tudengi õppetöös,* TTÜ, 2017.

[77] M. Fowler, Patterns of Enterprise Application Architecture, Addison-Wesley Professional, 2002.

[78] S. Yung, „Laravel Greatest Trick Revealed: Magic Methods,“ 15 4 2019. [Võrgumaterjal]. Available: https://dev.to/nvio/laravel-greatest-trick-revealed-magic-methods-31om. [Kasutatud 4 12 2020].

[79] The PHP Group, „PHP: Magic Methods,“ [Võrgumaterjal]. Available: https://www.php.net/manual/en/language.oop5.magic.php. [Kasutatud 4 12 2020].

[80] „Getting Started with Doctrine,“ [Võrgumaterjal]. Available: https://www.doctrine-project.org/projects/doctrine-orm/en/2.7/tutorials/getting-started.html. [Kasutatud 4 12 2020].

[81] M. Fowler, „Data Transfer Object,“ [Võrgumaterjal]. Available: https://martinfowler.com/eaaCatalog/dataTransferObject.html. [Kasutatud 4 12 2020].

[82] Auth0, Inc, „Authentication and Authorization,“ [Võrgumaterjal]. Available: https://auth0.com/docs/authorization/authentication-and-authorization. [Kasutatud 4 12 2020].

[83] Auth0 Inc, „Introduction to JSON Web Tokens,“ [Võrgumaterjal]. Available: https://jwt.io/introduction/. [Kasutatud 4 12 2020].

[84] S. Tymon, „JSON Web Token Authentication for Laravel & Lumen,“ [Võrgumaterjal]. Available: https://jwt-auth.readthedocs.io/en/develop/. [Kasutatud 4 12 2020].

[85] B. Childress, „Securing Applications With Better User Authorization,“ 20 11 2018. [Võrgumaterjal]. Available: https://medium.com/capital-one-tech/securing-applications-with-better-user-authorization-625ec07a7001. [Kasutatud 3 1 2021].

[86] T. Otwell, „Authorization - Lumen,“ [Võrgumaterjal]. Available: https://lumen.laravel.com/docs/5.3/authorization. [Kasutatud 4 12 2020].

[87] Laravel LLC, „Authentication - Laravel,“ [Võrgumaterjal]. Available: https://laravel.com/docs/8.x/authentication#introduction. [Kasutatud 4 12 2020].

[88] „[acl] Introduction - Laravel Doctrine,“ [Võrgumaterjal]. Available: http://laraveldoctrine.org/docs/1.4/acl. [Kasutatud 4 12 2020].

[89] Cloudflare, Inc., „What is HTTPS?,“ [Võrgumaterjal]. Available: https://www.cloudflare.com/learning/ssl/what-is-https/. [Kasutatud 2 1 2021].

[90] R. Degges, „What Happens If Your JWT Is Stolen?,“ [Võrgumaterjal]. Available: https://developer.okta.com/blog/2018/06/20/what-happens-if-your-jwt-is-stolen. [Kasutatud 2 1 2021].

[91] Logiware ApS, „SHARK WMS Documentation,“ [Võrgumaterjal]. Available: https://sharkwms.com/en/host_link_restapi_ref.html. [Kasutatud 14 12 2020].

[92] Megaventory Inc, „Megaventory API Documentation,“ [Võrgumaterjal]. Available: https://api.megaventory.com/v2017a/documentation/index.html. [Kasutatud 14 12 2020].

[93] Zoho Corporation Pvt. Ltd, „Zoho Inventory API Documentation,“ [Võrgumaterjal]. Available: https://www.zoho.com/inventory/api/v1/. [Kasutatud 14 12 2020].

Lisa 1 - Lihtlitsents

Mina, Margus Põlma

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "Majandustarkvara laomooduli arendus" , mille juhendaja on Toomas Lepikult

    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.

Lisa 2 – Kasutajalood

Tabel 6. Laosüsteemi kasutajalugude eepikud.

Number	**Kirjeldus**
E-1	Administraatorina soovin ma lisada ning hallata isikuid/ettevõtteid, et määrata nad ostjaks või müüjaks erinevate arvete peal.
E-2	Administraatorina soovin ma lisada ning hallata lao artikleid, et saada ülevaade kõikidest artiklitest ning et oleks võimalik artikleid erinevate arvete ridade peal kasutada.
E-3	Kasutajana/administraatorina soovin otsida lao artiklit, et näha selle kohta teavet.
E-4	Administraatorina soovin sisestada ja hallata ostu- ning müügiarveid/saatelehti, et laovarude ost/müük/liikumine oleks dokumenteeritud.
E-5	Kasutajana/administraatorina soovin sisestada arvele/saatelehele ridu, et arvega osta, müüa või liigutada tooteid.
E-6	Administraatorina soovin lisada lao artiklite kategooriaid, et artiklite grupeerimine oleks võimalik.
E-7	Administraatorina soovin näha kõiki lao artikleid, neid filtreerida ning sorteerida mistahes parameetri järgi, et saada ülevaade artiklitest.
E-8	Administraatorina soovin näha erinevat statistikat artiklite kohta, et saada ülevaade artiklite liikumistest, käibest vms.
E-9	Kasutajana/administraatorina soovin tellida tooteid, et täiendada laovaru.

Tabel 7. Kasutajalood kirjeldamaks eepikut E-1.

Number	**<kasutajana>**	**<tegevus>**	**<tulemus/põhjus>**
U-1	Administraatorina	soovin salvestada isiku/ettevõtte kontaktandmed sh. nime, e-maili aadressi, telefoni numbri ning aadressi	et kasutada seda infot arvete loomisel või kliendiga kontakteerumiseks
U-2	Administraatorina	soovin salvestada ettevõtte registrikoodi, KMKR numbri	et kasutada seda infot arvete loomisel
U-3	Administraatorina	soovin salvestada isiku/ettevõtte makseinfo sh. IBAN, panga nimi vms	et kasutada seda infot arve loomisel või arve tasumisel
U-4	Administraatorina/ Kasutajana	soovin otsida isikut/ettevõtet mistahes parameetri järgi	et leida kliendi kohta infot või siduda klient mõne arvega
U-5	Administraatorina	soovin näha kõiki salvestatud isikuid ühes kohas koos isiku nime, registrikoodi, e-maili aadressi ning telefoni numbriga	et saada ülevaade kõikidest isikutest

Tabel 8. Kasutajalood kirjeldamaks eepikut E-2.

Number	**<kasutajana>**	**<tegevus>**	**<tulemus/põhjus>**
U-1	Administraatorina	soovin salvestada artikli kohta põhiinfot sh. nimetus, tootekood, mõõtühik	et seda teavet edaspidi kasutada
U-2	Administraatorina	soovin salvestada artikli kohta täiendavaid omadusi nt toote värvus	et oleks võimalik samalaadseid tooteid eristada
U-3	Administraatorina	soovin salvestada artikli miinimum ja maksimum kogused	et süsteemil oleks võimalik tellimise vajadusest teavitada
U-4	Administraatorina	soovin kustutada artiklit	et eemaldada andmebaasist vanad artiklid
U-5	Administraatorina	soovin salvestada artikli kohta selle liigi (toode, teenus, mittevähenev)	sest laos olevad artiklid võivad olla erineva iseloomuga
U-6	Administraatorina	soovin lisada artikli mingisse kategooriasse	et artikleid oleks võimalik grupeerida kategooria alusel

Tabel 9. Kasutajalood kirjeldamaks eepikut E-3.

Number	**<kasutajana>**	**<tegevus>**	**<tulemus/põhjus>**
U-1	Administraatorina/ kasutajana	soovin otsida artiklit nimetuse, tootekoodi või mistahes artikli omaduse järgi	et leida teavet otsitava artikli kohta
U-2	Administraatorina/ kasutajana	soovin näha korraga konkreetse artikli põhiinfot	et tutvuda konkreetse artikliga
U-3	Administraatorina	soovin näha artikli liikumisi valitud perioodi jooksul	et saada ülevaade konkreetse artikli kasutamisest
U-4	Administraatorina	soovin näha statistikat artikli kohta valitud perioodi jooksul sh. käive	et oleks võimalik sellest järeldusi teha
U-5	Administraatorina/ kasutajana	soovin näha artikli laoseisu	et saada infot konkreetse artikli laoseisu kohta

Tabel 10. Kasutajalood kirjeldamaks eepikut E-4.

Number	**<kasutajana>**	**<tegevus>**	**<tulemus/põhjus>**
U-1	Administraatorina	soovin salvestada dokumendi kohta põhiinformatsiooni sh. dokumendi number, esitamise kuupäev, maksetähtaeg, arve esitaja, arve saaja	et arve põhiinfo oleks salvestatud ning tulevikus leitav
U-2	Administraatorina	soovin salvestada dokumendi kohta tasumise infot sh. seda, et mitmes osas, millal ning mis makseviisi kasutati	et arve makseinfo oleks salvestatud ning tulevikus leitav ning et oleks võimalik lihtsasti leida tasumata arved
U-3	Administraatorina	soovin määrata arvele staatuse (loomisel, kinnitatud, makstud)	et oleks võimalik erinevas staatuses arveid lihtsasti eristada
U-4	Administraatorina	soovin näha kõiki ostuarveid ning nende kohta käivat põhiinfot (arve number, arve saaja, arve staatus, arve kogusumma, arve väljastamise aega ning maksetähtaega) ühes kohas ning eelnimetatud parameetrite alusel filtreerida ning sorteerida	et saada ülevaade kõikidest ostuarvetest ning nende põhiinfost
U-5	Administraatorina/ kasutajana	soovin näha kõiki müügiarveid/saatelehti ning nende kohta käivat põhiinfot (arve number, arve esitaja, arve staatus, arve kogusumma, arve väljastamise aega ning maksetähtaega) ühes kohas ning eelnimetatud parameetrite alusel filtreerida ning sorteerida	et saada ülevaade kõikidest müügiarvetest/saatelehtedest ning nende põhiinfost
U-6	Administraatorina	soovin näha detailset infot ostuarve kohta sh. ostuarve põhiinfo ning selle read	et näha infot konkreetse ostuarve kohta
U-7	Administraatorina/ kasutajana	soovin näha detailset infot müügiarve/saatelehe kohta sh. dokumendi põhiinfo ning selle read	et näha infot konkreetse dokumendi kohta
U-8	Administraatorina	soovin importida ning eksportida erinevaid arveid	et vähendada käsitsi tehtavat tööd

Tabel 11. Kasutajalood kirjeldamaks eepikut E-5.

Number	**<kasutajana>**	**<tegevus>**	**<tulemus/põhjus>**
U-1	Administraatorina	soovin lisada ostuarvele ridu mis on seotud konkreetse laoartikliga	et ostu korral konkreetse artikli laoseis suureneks
U-2	Administraatorina/ kasutajana	soovin lisada müügiarvele/saatelehele ridu, mis on seotud konkreetse laoartikliga	et müügi korral konkreetse artikli laoseis väheneks
U-3	Administraatorina/ kasutajana	soovin sisestada mistahes arvele ridu, mis ei ole seotud ühegi laoartikliga	sest vahepeal müüakse ka selliseid tooteid/teenuseid mille üle laoarvestust ei peeta
U-4	Administraatorina	soovin muuta/kustutada arvel rida ka siis, kui arve on juba kinnitatud	sest vahel erineb algselt sisestatud ostuarve tegelikust
U-5	Administraatorina	soovin sisestada arvele ümarduse summa	sest vahel arvutavad erinevad süsteemid summasid erinevalt
U-6	Administraatorina/ kasutajana	soovin sisestada arve reale käibemaksu protsendi	sest erinevatel ridadel võib olla erinev käibemaks
U-7	Administraatorina/ kasutajana	soovin näha arve ridade koondsummasid sh. summa ilma maksudeta, maksude summa, kogusumma	sest see ma ei viitsi seda käsitsi arvutada
U-8	Administraatorina/ kasutajana	soovin sisestada arve reale ostetava kauba koguse ning allahindluse protsendi	et dokumenteerida ostetava kauba kogus ning ostuhind
U-9	Administraatorina/ kasutajana	soovin et arve kinnitamisel sisestaks süsteem laoliikumised ning arvutaks kõigi liikumiste järgi laoseisu	et artiklite ost ja müük oleks jälgitav ning et kauba laoseis oleks teada

Tabel 12. Kasutajalood kirjeldamaks eepikut E-6.

Number	**<kasutajana>**	**<tegevus>**	**<tulemus/põhjus>**
U-1	Administraatorina	soovin salvestada lao kategooria nime	et artiklite grupeerimine kategooria alusel oleks võimalik
U-2	Administraatorina	soovin salvestada lao kategooriale alamkategooria	et muuta grupeerimine täpsemaks

Tabel 13. Kasutajalood kirjeldamaks eepikut E-7.

Number	**<kasutajana>**	**<tegevus>**	**<tulemus/põhjus>**
U-1	Administraatorina	soovin näha kõiki laos olevaid artikleid ühes kohas koos nende põhiinfoga (nimetus, tootekood, laoseis, mõõtühik, kategooria) ning neid sorteerida ja filtreerida eelnimetatud parameetrite järgi	saada ülevaatlikut infot artiklite kohta

Tabel 14. Kasutajalood kirjeldamaks eepikut E-8.

Number	**<kasutajana>**	**<tegevus>**	**<tulemus/põhjus>**
U-1	Administraatorina	soovin näha detailset infot kas kõigi või ühe konkreetse artikli kohta valitud perioodil	et saada infot artikli(te) liikumistest

Tabel 15. Kasutajalood kirjeldamaks eepikut E-9.

Number	**<kasutajana>**	**<tegevus>**	**<tulemus/põhjus>**
U-1	Administraatorina/ kasutajana	soovin sisestada tellimuse mingile lao artiklile mingis koguses	et oleks võimalik antud artikli laovaru täiendada ning seda artiklit müüa
U-2	Administraatorina	soovin luua tellimuspäringu ühele konkreetsele tarnijale ning vajadusel tellimust muuta	et tellida tarnijalt vajalikku kaupa
U-3	Administraatorina	soovin lisada konkreetse artikli tellimuse loodud tellimuspäringule	et grupeerida tellitavat kaupa erinevatele tellimuspäringutele
U-5	Administraatorina	soovin luua tellimispäringu alusel ostuarve	et tellimuspäringuga tellitud ning tarnija poolt müüdud kaup laos arvele võtte
U-6	Administraatorina	soovin et süsteem kuvaks tellimuste all ka neid artikleid, mille laoseis hakkab minema kriitiliselt madalaks	et täiendada aegsasti laovarusid

Lisa 3. Funktsionaalsete nõuete täitmise hindamise tabel

Tabel 16. Funktsionaalsete nõuete täitmise hindamise tabel.

	Vastavus	**Hinne**
Isikute haldus	Kas on võimalik salvestada: nime, e-maili, aadressi, telefoni numbrit, aadressi, registrikoodi, makseinfot (IBAN, panga nimi vms)? Kas on võimalik isiku infot vaadata ning kõiki isikuid ühes kohas vaadata ning filtreerida/sorteerida?	
Artiklite haldus	Kas on võimalik salvestada: nimetust, tootekoodi, mõõtühikut, miinimum- ja maksimumkoguseid, kategooriat, muud infot?	
Artikli otsimine	Kas on võimalik: otsida artiklit mistahes parameetri järgi, näha artikli põhiinfot, näha artikli liikumisi valitud perioodi jooksul, näha statistikat, näha laoseisu? Kas artikleid on võimalik näha ühes kohas?	
Ostu- ja müügiarve info	Kas on võimalik salvestada dokumendi kohta: number, esitamise kuupäev, maksetähtaeg, arve esitaja ja saaja, tasumise info. Kas on võimalik dokumentide kuvamine ning filtreerimine ühes kohas ning detailse info vaatamine? Kas on võimalik importida/eksportida arveid?	
Ostu- ja müügiarve read	Kas on võimalik lisada ridu mis on seotud lao artikliga kui ka ridu mis on laovälised? Sisestada reale koguse, allahindluse ning käibemaksu protsendi? Kas rea alusel arvestatakse laoseisu?	
Kategooriad	Kas saab luua kategooriaid ning artikleid kategooriatesse määrata?	
Tellimused	Kas on võimalik sisestada tellimusi laoseisu täiendamise eesmärgil? Kas süsteem lisab ka ise tellimusi vastavalt määratud miinimum- ja maksimumkogustele? Kas on võimalik tellimusarvest luua ostuarve?	
Rollid	Kas on võimalik kõiki tegevusi rollipõhiselt piirata?	

Lisa 4. SHARK WMS Cloud tarkvara sobivus

Tabel 17. SHARK WMS Cloud vastavus nõuetele. Info pärineb toote dokumentatsioonist [91].

	Vastavus	**Hinne**
Isikute haldus	Saab sisestada ainult isiku telefoni numbri ning nime. Halduse funktsionaalsus puudub.	1
Artiklite haldus	Vastab nõuetele.	5
Artikli otsimine	Võimalik otsida artikli numbri järgi. Põhiinfot ning laoseisu on võimalik näha. Kõiki artikleid näeb ühes kohas. Lao liikumised ei näe. Sorteerida ega filtreerida ei saa.	3
Ostu- ja müügiarve info	Võimaldab sisestada ostu ja müügiarveid tellimuste näol. Muudele nõuetele ei vasta.	1
Ostu- ja müügiarve read	Ridu saab sisestada ainult koos arvega, hiljem ridu lisada ei saa. Allahindlust, ostu-/müügihinda, käibemaksu protsenti lisada ei saa. Laoväliseid tooteid sisestada ei saa. Laoseisu arvestatakse.	1
Kategooriad	Kategooriaid saab luua vabavormis artikli loomisel/muutmisel.	4
Tellimused	Tellimusi saab sisestada. Süsteem vastaval miinimumkogusele tootele tellimust ei sisesta. Tellimusarvest ostuarvet luua ei saa, kuid laoseis suureneb ilma ostuarvetagi.	3
Rollid	Ei vasta nõuetele.	0
Kokku		18/40

Lisa 5. Megaventory tarkvara sobivus

Tabel 18. Megaventory vastavus nõuetele. Info pärineb toote dokumentatsioonist [92].

	Vastavus	**Hinne**
Isikute haldus	Isiku kohta on võimalik salvestada nõutud info. Isiku infot saab vaadata. Kõiki isikuid saab vaadata ning filtreerida kuid mitte sorteerida.	4
Artiklite haldus	Miinimum- ja maksimumkogust ei ole võimalik määrata. Muus osas vastab nõuetele.	5
Artikli otsimine	Artiklit saab otsida mistahes parameetri järgi. Artikli põhiinfot saab vaadata. Artikli liikumisi ei saa vaadata. Kõiki artikleid näeb ühes kohas.	4
Ostu- ja müügiarve info	Dokumendi kohta ei saa salvestada tasumise infot. Eksport ja import pole võimalik. Muus osas vastab nõuetele.	3
Ostu- ja müügiarve read	Vastab täielikult nõuetele.	5
Kategooriad	Vastab täielikult nõuetele.	5
Tellimused	Vastavalt miinimumkogusele automaatselt tellimusi ei sisestata. Tellimusarveid saab sisestada. Laoseisu täiendamine vastavalt tellimusele ei ole teada.	2
Rollid	Ei vasta nõuetele.	0
Kokku		28/40

Lisa 6. Zoho Inventory tarkvara sobivus

Tabel 19. Zoho Inventory vastavus nõuetele. Info pärineb toote dokumentatsioonist [93].

	Vastavus	**Hinne**
Isikute haldus	Vastab täielikult nõuetele.	5
Artiklite haldus	Maksimumkogust ei saa määrata. Muus osas vastab täielikult nõuetele.	5
Artikli otsimine	Otsida saab ainult artikli id järgi. Näeb põhiinfot ja laoseisu aga mitte liikumisi. Artikleid näeb ühes kohas aga sorteerimine/filtreerimine puudub.	3
Ostu- ja müügiarve info	Vastab täielikult nõuetele.	5
Ostu- ja müügiarve read	Vastab täielikult nõuetele.	5
Kategooriad	Vastab täielikult nõuetele.	5
Tellimused	Vastavalt miinimumkogusele automaatselt tellimusi ei sisestata. Tellimusarveid saab sisestada vastavalt nõuetele. Tellimusarvest saab luua arve. Laoseisu täiendamiseks tuleb luua dokument, mille alusel tooted lattu võetakse.	4
Rollid	Vastab täielikult nõuetele.	5
Kokku		37/40

Lisa 7. Andmebaasi struktuur

Tabel 20. Autori loodud andmebaasi tabelite kirjeldus.

ARTICLE			
Tabel laoartiklite põhiinfo hoidmiseks.			
Väli	**Tüüp**	**Omadused**	**Kirjeldus**
id	int	PK, auto increment	Tabeli primaarvõti.
name	varchar(256)		Artikli nimetus.
product_code	varchar(20)		Artikli tootekood.
min_amount	decimal(18,5)	NULL	Artikli miinimumkogus, mida saab tulevikus kasutada.
max_amount	decimal(18,5)	NULL	Artikli maksimumkogus, mida saab tulevikus kasutada.
category_id	int	FK	Välisvõti artikli kategooriate tabelitesse.
type_id	int	FK	Välisvõti artikli tüüpide tabelisse.
uom_id	int	FK	Välisvõti mõõtühikute tabelisse.
organisation_id	int	FK	Välisvõti organisatsioonide tabelisse, mis näitab kellele, konkreetne artikkel kuulub.
ARTICLE_STOCK			
Tabel laoartikli laoseisu hoidmiseks, kus salvestatakse artikli laoseis konkreetsel ajahetkel.			
Väli	**Tüüp**	**Omadused**	**Kirjeldus**
id	int	PK, auto increment	Tabeli primaarvõti.
quantity	decimal(18,5)		Laoartikli laoseis.
date_added	timestamp		Laoseisu arvutamise aeg.
article_id	int	FK	Välisvõti, mis viitab artiklile, mille kohta antud rida käib.
organisation_id	int	FK	Välisvõti, mis viitab organisatsioonide tabelisse.

ARTICLE_TYPE			
Tabel laoartikli tüübi kirjeldamiseks, kuhu on salvestatud kõikvõimalikud artiklite tüübid (nt. "toode", "teenus" jms).			
Väli	**Tüüp**	**Omadused**	**Kirjeldus**
id	int	PK, auto increment	Tabeli primaarvõti.
name	varchar(96)		Artikli tüübi nimetus.
ARTICLE_CATEGORY			
Tabel laoartikli kategooriate hoidmiseks.			
Väli	**Tüüp**	**Omadused**	**Kirjeldus**
id	int	PK	Tabeli primaarvõti.
name	varchar(256)		Kategooria nimetus.
parent_id	int	FK, NULL	Välisvõti, mis viitab samasse tabelisse, konkreetse kategooria ülemkategooria juurde.
organisation_id	int		Välisvõti, mis viitab organisatsioonide tabelisse.
UOM			
Tabel laoartiklite mõõtühikute hoidmiseks.			
Väli	**Tüüp**	**Omadused**	**Kirjeldus**
id	int	PK, auto increment	Tabeli primaarvõti.
name	varchar(20)		Mõõtühiku nimetus.
organisation_id	int	FK	Välisvõti, mis viitab organisatsioonide tabelisse.
PROPERTY			
Tabel kõikvõimalike artikli omaduste hoidmiseks (nt. kaal, mõõdud, värvus).			
Väli	**Tüüp**	**Omadused**	**Kirjeldus**
id	int	PK, auto increment	Tabeli primaarvõti
name	varchar(96)		Artikli omaduse nimetus.
description	varchar(256)	NULL	Artikli omaduse täiendav kirjeldus.
organisation_id	int	FK	Välisvõti, mis viitab organisatsioonide tabelisse.
ARTICLE_PROPERTY			
Tabel artiklile omistatud omaduste ning nende väärtuste hoidmiseks.			

Väli	Tüüp	Omadused	Kirjeldus
id	int	PK, auto increment	Tabeli primaarvõti.
prop_value	varchar(256)		Omaduse väärtus. Näiteks kui omadus on värvus, siis võib sellele väljale olla salvestatud väärtus "must".
article_id	int	FK	Välisvõti, mis viitab artiklile, millele see omadus kuulub.
article_property_id	int	FK	Välisvõti, mis viitab konkreetsele artikli omadusele.
organisation_id	int	FK	Välisvõti, mis viitab organisatsioonide tabelisse.
BATCH			
Tabel kaubapartiide hoidmiseks, mille abil seotakse arvega ostetud või müüdud kaup laoliikumiste ja artiklitega.			
Väli	**Tüüp**	**Omadused**	**Kirjeldus**
id	int	PK, auto increment	Tabeli primaarvõti.
name	varchar(96)		Kaubapartii nimi/number.
date_added	timestamp		Kaubapartii lisamise aeg.
article_id	int	FK	Välisvõti, mis viitab konkreetsele artiklile, mille kohta see partii käib.
organisation_id	int	FK	Välisvõti, mis viitab organisatsioonide tabelisse.
MOVEMENT			
Tabel artiklite liikumise hoidmiseks, kus on salvestatud info kauba, partii, alusdokumendi kohta. Selle tabeli read on laoarvestuse aluseks - summeerides kõik ühe kauba liikumised saadakse kauba laoseis.			
Väli	**Tüüp**	**Omadused**	**Kirjeldus**
id	int	PK, auto increment	Tabeli primaarvõti.
quantity	decimal(18,5)		Liigutatud kogus. Võib olla positiivne (ostu korral) kui ka negatiivne (müügi korral).
date_added	timestamp		Laoliikumise lisamise aeg.
batch_id	int	FK	Välisvõti, mis viitab kaubapartiile.

Väli	Tüüp	Omadused	Kirjeldus
document_id	int	FK	Välisvõti, mis viitab dokumendile, mille alusel kaupa liigutati.
parent_id	int	FK, NULL	Välisvõti, mis viitab samasse tabelisse ülemliikumisele.
organisation_id	int	FK	Välisvõti, mis viitab organisatsioonide tabelisse.
DOCUMENT			
Tabel ostu- ja müügiarvete päisete hoidmiseks. Päises on salvestatud ka arve koondsummad, et vältida korduvat arvutamist, mille tulemusel võivad koondsummad kohati erineda.			
Väli	**Tüüp**	**Omadused**	**Kirjeldus**
id	int	PK, auto increment	Tabeli primaarvõti.
number	varchar(32)		Dokumendi number.
date_issued	timestamp		Dokumendi väljastamise kuupäev.
date_due	timestamp		Dokumendi maksetähtaeg.
document_sum	decimal(12,5)		Dokumendi ridade summa, kust on maha arvestatud allahindlused kuid millele pole lisatud maksu.
document_tax	decimal(12,5)		Dokumendi (käibe-) maksu summa.
document_total	decimal(12,5)		Dokumendi kogusumma koos käibemaksuga.
type_id	int	FK	Välisvõti dokumendi tüüpide tabelisse, mis kirjeldab konkreetse dokumendi tüübi (ostu-/müügiarve).
organisation_id	int	FK	Välisvõti, mis viitab organisatsioonide tabelisse.
DOCUMENT_TYPE			
Tabel kõikvõimalike dokumenditüüpide hoidmiseks (nt. ostuarve, müügiarve ning tulevikus ka kreeditarve vms).			
Väli	**Tüüp**	**Omadused**	**Kirjeldus**
id	int	PK, auto increment	Tabeli primaarvõti.
name	varchar(32)		Dokumenditüübi nimetus.

DOCUMENT_STATUS_TYPE			
Tabel kõikvõimalike dokumendi staatuste hoidmiseks. Näiteks võib dokumendi staatus olla "loodud", kui dokument on loodud ning sellele lisatakse aktiivselt infot. Dokumendi kinnitamisel võib dokumendi staatus olla "kinnitatud".			
Väli	**Tüüp**	**Omadused**	**Kirjeldus**
id	int	PK, auto increment	Tabeli primaarvõti.
name	varchar(32)		Dokumendi staatuse nimetus.
DOCUMENT_STATUS			
Tabel dokumendi staatuste ning nende ajas muutumise salvestamiseks.			
Väli	**Tüüp**	**Omadused**	**Kirjeldus**
id	int	PK, auto increment	Tabeli primaarvõti.
date_added	timestamp		Aeg, mil konkreetsele dokumendile määrati see staatus.
document_id	int	FK	Välisvõti, mis viitab dokumendile, mille kohta konkreetne käib.
status_id	int	FK	Välisvõti, mis viitab staatusele, mida konkreetse kirjega konkreetsele dokumendile omistati.
organisation_id	int	FK	Välisvõti, mis viitab organisatsioonide tabelisse.
DOCUMENT_ROW			
Tabel dokumendil olevate ridade hoidmiseks.			
Väli	**Tüüp**	**Omadused**	**Kirjeldus**
id	int	PK, auto increment	Tabeli primaarvõti.
description	varchar(64)	NULL	Täiendav kirjeldus rea kohta, mida saab vajadusel kasutada.
quantity	decimal(18,5)		Real oleva toote kogus.
discount_pct	int		Allahindluse protsent.
discount_amount	decimal(12,5)		Allahindluse summa kogu rea summalt.
price	decimal(12,5)		Real oleva toote tükihind ilma allahindlust arvestamata.

Väli	Tüüp	Omadused	Kirjeldus
price_total	decimal(12,5)		Rea summa, kus on arvestatud kogust ning allahindlust.
tax_pct	int		(Käibe-) maksu protsent.
tax_amount	decimal(12,5)		(Käibe-) maksu summa kogu rea summalt.
document_id	int	FK	Välisvõti dokumendi päisele.
organisation_id	int	FK	Välisvõti, mis viitab organisatsioonide tabelisse.
batch_id	int	FK, NULL	Välisvõti, mis viitab kaubapartiile. Võib olla tühi juhul kui arve rida ei ole seotud ühegi artikliga.
customer_id	int	FK	Välisvõti, mis viitab konkreetse dokumendi teisele osapoolele (ostja/müüja).
DOCUMENT_PAYMENT			
Tabel dokumendi tasumiste salvestamiseks.			
Väli	**Tüüp**	**Omadused**	**Kirjeldus**
id	int	PK, auto increment	Tabeli primaarvõti.
date_added	timestamp		Tasumise lisamise kuupäev.
amount_paid	decimal(12,5)		Tasutud summa.
document_id	int	FK	Välisvõti, mis viitab konkreetsele dokumendile.
payment_type_id	int	FK	Välisvõti, mis viitab makseviisile, mida tasumisel kasutati.
customer_banking_id	int	FK, NULL	Välisvõti, mis viitab konkreetse isiku makseinfole.
organisation_id	int	FK	Välisvõti, mis viitab organisatsioonide tabelisse.
PAYMENT_TYPE			
Tabel kõikvõimalike makseviiside hoidmiseks (nt sularaha, kaardimakse, arve).			
Väli	**Tüüp**	**Omadused**	**Kirjeldus**
id	int	PK, auto increment	Tabeli primaarvõti.
name	varchar(32)		Makseviisi nimetus.

CUSTOMER_BANKING			
Tabel isiku makseviiside hoidmiseks. Näiteks saab siin hoida ühele kliendile kuuluvaid erinevate pankate rekvisiite.			
Väli	**Tüüp**	**Omadused**	**Kirjeldus**
id	int	PK, auto increment	Tabeli primaarvõti.
name	varchar(128)		Makseviisi nimetus.
description	varchar(256)	NULL	Vajadusel täiendav makseviisi kirjeldus.
IBAN	varchar(64)		IBAN kood.
BIC	varchar(64)		BIC/SWIFT kood.
customer_id	int	FK	Välisvõti isikute tabelisse, mis viitab konkreetsele kliendile.
organisation_id	int	FK	Välisvõti, mis viitab organisatsioonide tabelisse.
CUSTOMER			
Tabel isikute info hoidmiseks. Siin võivad olla salvestatud nii klientide kui tarnijate info.			
Väli	**Tüüp**	**Omadused**	**Kirjeldus**
id	int	PK, auto increment	Tabeli primaarvõti.
name	varchar(256)		Isiku nimi.
reg_no	varchar(64)		Isiku registrikood - eraisikul isikukood, ettevõttel reg. nr.
vat_no	varchar(64)	NULL	Ettevõtte KMKR number.
type_id	int	FK	Välisvõti kliendi tüüpide tabelisse, millega määratakse konkreetse kliendi tüüp (nt. klient, tarnija vms).
organisation_id	int	FK	Välisvõti, mis viitab organisatsioonide tabelisse.
CONTACT_TYPE			
Tabel kõikvõimalike kontaktiviiside hoidmiseks. Näiteks telefoni või mobiili number, e-maili aadress, Skype-i tunnus.			
Väli	**Tüüp**	**Omadused**	**Kirjeldus**
id	int	PK, auto increment	Tabeli primaarvõti.
name	varchar(32)		Kontaktiviisi nimetus.

CUSTOMER_CONTACT			
Tabel isiku kontaktiviiside hoidmiseks.			
Väli	**Tüüp**	**Omadused**	**Kirjeldus**
id	int	PK, auto increment	
value	varchar(256)		Konkreetse kontaktiviisi väärtus. Näiteks kui kontaktiviis on telefoni number, siis siin on salvestatud isiku telefoni numbri väärtus (nt. "+372 55555555").
customer_id	int	FK	Välisvõti klientide tabelisse.
contact_type_id	int	FK	Välisvõti kontaktiviiside tabelisse.
organisation_id	int	FK	Välisvõti, mis viitab organisatsioonide tabelisse.
CUSTOMER_TYPE			
Tabel kõikvõimalike klientide/isikute liikide hoidmiseks. Näiteks klient, tarnija vms.			
Väli	**Tüüp**	**Omadused**	**Kirjeldus**
id	int	PK, auto increment	Tabeli primaarvõti.
value	varchar(32)		Kliendi/isiku liigi nimetus.
ORGANISATION			
Tabel organisatsioonide/ettevõtete hoidmiseks. Siia tabelisse sisestatakse kõik majandustarkvara kasutavad ettevõtted, ning kõik olulised tabelid omavad välisvõtit siia tabelisse.			
Väli	**Tüüp**	**Omadused**	**Kirjeldus**
id	int	PK, auto increment	Tabeli primaarvõti.
name	varchar(256)		Organisatsiooni/ettevõtte nimi.
USER			
Tabel kasutajate autentimiseks vajaliku info hoidmiseks.			
Väli	**Tüüp**	**Omadused**	**Kirjeldus**
id	int	PK, auto increment	Tabeli primaarvõti
username	varchar(32)	UNIQUE	Unikaalne kasutajanimi, mida kasutatakse isiku autentimiseks.
name	varchar(96)		Kasutaja nimi.

Väli	Tüüp	Omadused	Kirjeldus
password	varchar(255)		Kasutaja parooli räsi.
PERMISSION			
Tabel kõikvõimalike õiguste hoidmiseks.			
Väli	**Tüüp**	**Omadused**	**Kirjeldus**
id	int	PK, auto increment	Tabeli primaarvõti.
name	varchar(255)		Õiguse nimi.
description	varchar(255)	NULL	Vajadusel õiguse täpsem kirjeldus.
USER_PERMISSION			
Tabel, kus hoitakse infot selle kohta, et mis kasutajal on mis õigused.			
Väli	**Tüüp**	**Omadused**	**Kirjeldus**
user_id	int	FK	Välisvõti, mis viitab kasutajate tabelisse.
permission_id	int	FK	Välisvõti, mis viitab õiguste tabelisse.

Lisa 8. Lahenduse otspunktid

Tabel 21. Prototüübis loodud otspunktid.

Artikkel			
Meetod	**URL**	**Kirjeldus**	**Vastus**
GET	/articles	Kõigi artiklite kuvamine.	Kõigi artiklite list koos põhiinfoga.
GET	/articles/{id}	Konkreetse artikli info kuvamine.	Konkreetne artikli info.
POST	/articles/{id}	Uue artikli loomine.	Loodud artikli id.
PUT	/articles/{id}	Artikli info uuendamine.	Uuendatud artikkel.
Artiklitüüp			
Meetod	**URL**	**Kirjeldus**	**Vastus**
GET	/articles/types	Kõigi artiklitüüpide kuvamine.	Artiklitüüpide list.
Kaubapartii			
Meetod	**URL**	**Kirjeldus**	**Vastus**
GET	/articles/{artikli_id} /batches	Konkreetse artikliga seotud kaubapartiide kuvamine.	Kaubapartiide list.
GET	/articles/{artikli_id} /batches/{partii_id}	Konkreetse artikliga seotud konkreetse kaubapartii kuvamine.	Konkreetne kaubapartii.
Mõõtühik			
Meetod	**URL**	**Kirjeldus**	**Vastus**
GET	/articles/unit-of-measures	Kõigi artikli mõõtühikute kuvamine.	Mõõtühikute list.
GET	/articles/unit-of-measures/{id}	Konkreetse mõõtühiku kuvamine.	Konkreetne mõõtühik.
POST	/articles/unit-of-measures	Uue mõõtühiku loomine.	Loodud mõõtühiku id.
PUT	/articles/unit-of-measures/{id}	Mõõtühiku info uuendamine.	Uuendatud mõõtühik.
Kategooria			
Meetod	**URL**	**Kirjeldus**	**Vastus**
GET	/articles/categories	Kõigi kaubakategooriate kuvamine.	Kaubakategooriate list.

Meetod	URL	Kirjeldus	Vastus
GET	/articles/categories /{id}	Konkreetse kaubakategooria kuvamine.	Konkreetne kaubakategooria.
POST	/articles/categories	Uue kaubakategooria loomine.	Loodud kaubakategooria id.
PUT	/articles/categories /{id}	Kaubakategooria info uuendamine.	Uuendatud kaubakategooria.
Dokument			
Meetod	**URL**	**Kirjeldus**	**Vastus**
GET	/documents	Kõigi arvete kuvamine.	Kõigi arvete list koos põhiinfoga.
GET	/documents/{id}	Konkreetse arve kuvamine.	Konkreetne arve.
POST	/documents	Uue arve lisamine.	Loodud arve id.
PUT	/documents/{id}	Arve info muutmine.	Uuendatud arve.
POST	/documents/{id} /finish	Arve lõplik kinnitamine ning laoliikumiste sisestamine.	HTTP staatuskood 200.
Dokumendi read			
Meetod	**URL**	**Kirjeldus**	**Vastus**
GET	/document /{dokumendi_id} /rows	Kõigi konkreetse dokumendi ridade kuvamine.	List dokumendile lisatud ridadest.
GET	/document /{dokumendi_id} /rows/{rea_id}	Konkreetsel dokumendil oleva konkreetse rea kuvamine.	Konkreetne rida dokumendil.
POST	/document /{dokumendi_id} /rows	Uue rea lisamine dokumendile.	Loodud dokumendi rea id.
PUT	/document /{dokumendi_id} /rows/{rea_id}	Dokumendi rea info muutmine.	Uuendatud dokumendi rida.
Autentimine			
Meetod	**URL**	**Kirjeldus**	**Vastus**
POST	/auth/login	Autentimispäring, millele tuleb lisada kasutajanimi ning parool.	Eduka autentimise korral JWT, mis tuleb lisada igale järgnevale päringule.

Lisa 9. Testid

Tabel 22. Test T01 - autentimine.

Testi osa	**Kirjeldus**
Number ja nimi	T01 - Autentimine
Eesmärk	Kontrollida kas kasutaja pääseb rakenduse osadele ligi ilma autentimata.
Eeltingimus	-
Testi käik	1. Tee mistahes päring suvalisele rakenduse otspunktile (näiteks */articles*) ilma HTTP autoriseerimise päiseta.
Oodatav tulemus	HTTP veakood 401 "Unauthorized".

Tabel 23. Test T02 - autoriseerimine.

Testi osa	**Kirjeldus**
Number ja nimi	T02 - Autoriseerimine - kasutaja õiguse kontrollimine
Eesmärk	Kontrollida kas autenditud kasutaja pääseb rakenduse osadele ligi omamata selle jaoks õigust.
Eeltingimus	Kasutaja on autenditud. Kasutajal ei ole õigust kõikide artiklite vaatamiseks.
Testi käik	1. Tee GET päring rakenduse otspunktile "*/articles*" koos HTTP autoriseerimise päisega.
Oodatav tulemus	HTTP veakood 401 "Unauthorized".

Tabel 24. Test T03 - autoriseerimine.

Testi osa	**Kirjeldus**
Number ja nimi	T03 - Autoriseerimine - kasutaja organisatsiooni kontrollimine
Eesmärk	Kontrollida kas autenditud kasutaja pääseb mõne teise organisatsiooni andmetele ligi.
Eeltingimus	Kasutaja on autenditud. Kasutaja kuulub organisatsiooni. Mõne teise organisatsiooni poolt on loodud artikkel ning selle id on teada.
Testi käik	1. Tee GET päring rakenduse otspunktile "*/articles/{id}*" koos HTTP autoriseerimise päisega.
Oodatav tulemus	HTTP veakood 401 "Unauthorized".

Tabel 25. Test T04 - artikli sisestamine.

Testi osa	**Kirjeldus**
Number ja nimi	T04 - Artikli sisestamine
Eesmärk	Kontrollida kas süsteem võimaldab uute artiklite sisestamist.
Eeltingimus	Kasutaja on autenditud. Kasutaja kuulub organisatsiooni. Kasutajal on õigus artikli loomiseks.
Testi käik	1. Tee POST päring rakenduse otspunktile "*/articles*" koos HTTP autoriseerimise päisega ning otspunktile vastava päringu kehaga; 2. Õnnestunud päringu korral tagastatakse HTTP kood 201 ning päringu vastuse kehas on väli "*created_id*", mida on vaja kasutada järgmisel sammul; 3. Tee GET päring rakenduse otspunktile "*/articles/{id}*", kirjutades "*{id}*" asemele eelmisel sammul saadud id
Oodatav tulemus	HTTP kood 200 ning kasutajale kuvatakse loodud artikli info.

Tabel 26. Test T05 - artikli ostmine.

Testi osa	**Kirjeldus**
Number ja nimi	T05 - Artikli ostmine ostuarvega
Eesmärk	Kontrollida kas süsteem võimaldab artiklite ostmist ostuarvega.
Eeltingimus	Kasutaja on autenditud. Kasutaja kuulub organisatsiooni. Kasutajal on õigus ostuarvele ridade lisamiseks ning ostuarve kinnitamiseks. On loodud ostuarve mustand ning on loodud artikkel.
Testi käik	1. Tee POST päring rakenduse otspunktile "*/document/{dokumendi_id}/rows*" koos HTTP autoriseerimise päisega ning otspunktile vastava päringu kehaga lisades sellele loodud kaubaartikli kasutades välja "*batch*"; 2. Õnnestunud päringu korral tagastatakse HTTP kood 201; 3. Tee POST päring rakenduse otspunktile "*/document/{dokumendi_id}/submit*" koos HTTP autoriseerimise päisega; 4. Õnnestunud päringu korral tagastatakse HTTP kood 200; 5. Tee GET päring rakenduse otspunktile "/articles/{artikli_id}";
Oodatav tulemus	HTTP kood 200 ning kasutajale kuvatakse artikli info ning artikli laoseis on suurenenud vastavalt testi käigus sisestatud ostuarve reale.