TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Steven Juks 164662IAPB

KASSAPÕHISEL RAAMATUPIDAMISARVESTUSEL PÕHINEV LAOSEISU HALDAMISE VEEBIRAKENDUS

Bakalaureusetöö

Juhendaja: Martin Verrev
MSc

Tallinn 2019

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: Steven Juks

05.05.2019

Annotatsioon

Käesoleva töö eesmärgiks on arendada kassapõhise raamatupidamisarvestusega FIEdele veebipõhine laohaldusrakendus, mis võimaldab laoseisu üle arvestust pidada, arveid koostada ning kuludest ja tuludest ülevaade saada.

Töös analüüsin olemasolevaid laohaldusrakendusi ja kirjeldan nende põhjal loodava rakenduse nõuded ning andmemudeli. Lisaks uurin tehnoloogiaid, mida on rakenduse realiseerimiseks võimalik kasutada.

Töö tulemusena valmis veebipõhine laohaldusrakendus, mis vastab põhilistele seatud ärinõutele ning millel on edasiarendamise võimalused. Rakendus läheb ka reaalselt kasutusse.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 37 leheküljel, 6 peatükki, 15 joonist, 12 tabelit.

Abstract

Cash Accounting Based Inventory Management Web Application

Currently, there doesn't exist a bare-bones inventory management software for invoicing and bookkeeping for sole proprietors in Estonian. The existing applications come with a number of features that are necessary for larger companies, but unneeded for sole proprietors.

The intended web application is supposed to provide an easy way for sole proprietors to manage their inventory, create invoices and obtain an overview of their revenue without any unnecessary features.

The first part of the thesis gives an analytical overview of existing solutions along with functional and non-functional requirements. The second part of the thesis discusses the technologies used for the development and describes the application's main components. The final part outlines further improvements to make the application easier to use and develop.

As a result of this thesis, an inventory management application featuring the required functionality and possibilities for further improvement was developed. The application is intended to be actually used.

The thesis is in Estonian and contains 37 pages of text, 6 chapters, 15 figures, 12 tables.

Lühendite ja mõistete sõnastik

FIE	Füüsilisest isikust ettevõtja
JSON	*JavaScript Object Notation*, andmeesitusformaat
PDF	*Portable Document Format*, failiformaat
API	*Application Programming Interface*, liides serveriga suhtlemiseks
HTTP	*Hypertext Transfer Protocol*, protokoll teabe edastamiseks arvutivõrkudes
MIT litsents	Massachusettsi Tehnikainstituudi tarkvaralitsents
ORM	*Object-relational mapping*, andmebaasi mudeli defineerimine programmikoodi kaudu
SQL	*Structured Query Language*, andmebaasisüsteemide haldamiseks kasutatav keel
HTML	*Hypertext Markup Language*, veebilehtede märgendamise keel
JWT	*JSON Web Token*, JSON-i põhine ligipääsutokenite standard
URL	*Uniform Resource Locator*, veebiressurssi aadress
callback	kood, mida kutsutakse välja pärast mingi teise protsessi lõppemist

Sisukord

1 Sissejuhatus 10
1.1 Taust 10
1.2 Eesmärk 10
1.3 Metoodika 10
1.4 Ülevaade tööst 11
2 Analüüs 12
2.1 Alternatiivsed lahendused 12
2.1.1 Zoho Inventory 12
2.1.2 TradeGecko 13
2.1.3 Wave 13
2.1.4 Alternatiivsete lahenduste analüüs 14
2.2 Nõuded 15
2.2.1 Funktsionaalsed nõuded 15
2.2.2 Mittefunktsionaalsed nõuded 22
2.3 Andmemudel 22
2.3.1 Olemite semantika 22
3 Rakenduse realisatsioon 25
3.1 Tehnoloogiate valik 25
3.1.1 Figma 25
3.1.2 React 25
3.1.3 Node.js 26
3.1.4 Express 27
3.1.5 PostgreSQL 27

3.1.6 Sequelize ... 27
3.2 Arenduskeskkonna seadistamine ... 28
3.2.1 Serveripoolse arenduskeskkonna seadistamine ... 28
3.2.2 Kliendipoolse arenduskeskkonna seadistamine ... 29
3.3 Kasutajaliides ... 29
3.3.1 Vaated ... 30
3.3.2 Olekuhaldus ... 36
3.4 Andmebaas ... 37
3.5 Server ... 38
3.5.1 REST API disain ... 38
3.5.2 Autentimine ... 39
3.5.3 Andmete valideerimine ... 40
3.5.4 Müügiarve PDF faili koostamine ... 40
4 Valideerimine ... 42
5 Võimalikud edasiarendused ... 45
5.1 Registreerimise variandid ... 45
5.2 Kasutajaliidese mugavdamine ... 45
5.3 Tootmise moodul ... 46
5.4 Refaktoreerimine ja testimine ... 46
6 Kokkuvõte ... 47
Kasutatud kirjandus ... 48

Jooniste loetelu

Joonis 1. Andmemudel. 24
Joonis 2. Päringute töötlemise järjekord blokeeriva (vasakul) ja mitteblokeeriva (paremal) sisend/väljund mudeli korral. 26
Joonis 3. env faili sisu. 29
Joonis 4. Laoseisu vaade. 31
Joonis 5. Ostuarvete vaade. 32
Joonis 6. Müügiarvete vaade. 32
Joonis 7. Laekumiste vaade. 33
Joonis 8. Kulude vaade. 33
Joonis 9. Tulude vaade. 34
Joonis 10. Partnerite vaade. 34
Joonis 11. Statistika vaade. 35
Joonis 12. Sätete vaade. 35
Joonis 13. MobXi olekuhalduse koodinäide. 37
Joonis 14. Kulu muutmise valideerimisskeem. 40
Joonis 15. Müügiarve PDF faili koostamise koodinäide. 41

Tabelite loetelu

Tabel 1. Analüüsitud rakenduste võrdlus. 14
Tabel 2. Ladude halduse allsüsteemi nõuded. 15
Tabel 3. Kontode halduse allsüsteemi nõuded. 16
Tabel 4. Ostuarvete halduse allsüsteemi nõuded. 17
Tabel 5. Müügiarvete halduse allsüsteemi nõuded. 18
Tabel 6. Kulude halduse allsüsteemi nõuded. 19
Tabel 7. Laekumiste halduse allsüsteemi nõuded. 20
Tabel 8. Tulude halduse allsüsteemi nõuded. 21
Tabel 9. Partnerite halduse allsüsteemi nõuded. 21
Tabel 10. Kasutaja töölaua nõuded. 22
Tabel 11. Kaupadega seotud API päringud. 38
Tabel 12. Implementatsiooni nõuetele vastavus. 42

1 Sissejuhatus

1.1 Taust

Raamatupidamis- ja laoarvestus on ettevõtte tegevuses väga olulised põhiprotsessid. Erinevaid laotarkvarasid on palju, kuid enamus neist on suunatud pigem tekkepõhise raamatupidamisega käibemaksukohustuslastest ettevõtetele, kelle arvestus on keerulisem kui kassapõhise raamatupidamisega ettevõtetel. Kassapõhise raamatupidamisega FIEdel ei pruugi olla ressurssi selliste tarkvarade soetamiseks ja puudub ka vajadus kõigi nende funktsionaalsuste järele, mida tekkepõhise raamatupidamisega ettevõtetel vaja on. Näiteks kodumasinate remonditeenust pakkuvale FIEle piisaks vaid laoseisu, müügiarvete, ostuarvete ning kulude ja tulude haldamisest, et võimalikult hõlpsalt laoseisust ning raamatupidamisest ülevaade saada.

1.2 Eesmärk

Bakalaureusetöö põhieesmärgiks on luua eestikeelne veebirakendus, mis võimaldab teostada laohaldust, moodustada ostuarveid ja müügiarveid ning saada raamatupidamiseks kuludest ja tuludest ülevaade. Rakenduse sihtgrupiks on peamiselt kassapõhise raamatupidamisega FIEd. Lisaeesmärkideks on anda ülevaade rakenduse arenduskäigust ja võimalikest edasiarendustest.

1.3 Metoodika

Eesmärkide saavutamiseks viin esmalt läbi intervjuud rakenduse potentsiaalse kasutajaga ja uurin teisi sarnaseid rakendusi, et ilmutada rakenduse nõuded. Kasutaja poolt saadud tagasiside ja analoogsete rakenduste võrdlusanalüüsi põhjal arendan rakenduse kasutajaliidese funktsionaalse prototüübi, mille käigus võtan samuti arvesse kasutaja tagasisidet. Seejärel arendan prototüübi põhjal rakenduse kasutajaliidese, mille andmeobjektid on esmalt realiseeritud staatiliste JSON mudelitena. Kui kasutajaliidese komponendid on realiseeritud, realiseerin rakenduse serveripoole, et asendada staatilised

andmed dünaamilistega. Rakenduse valideerimiseks kontrollin rakenduse nõuetele vastavust.

1.4 Ülevaade tööst

Töö esimeses osas analüüsin alternatiivseid lahendusi ja kirjeldan rakenduse funktsionaalsed ja mittefunktsionaalsed nõuded ning andme- ja komponentmudeli. Seejärel valin rakenduse arendamiseks kasutatavad tehnoloogiad ja kirjeldan rakenduse realiseerimisprotsessi. Valideerin rakenduse vastavust ilmutatud nõuetele ja annan hinnangu arendusprotsessi edukusele. Lõpuks kirjeldan võimalikke edasisi arendustegevusi, mis muudaksid rakenduse nii kasutajatele mugavamaks kui arendajatele kergemini hallatavamaks, kuid ei mahtunud antud töö skoopi.

2 Analüüs

2.1 Alternatiivsed lahendused

Antud peatükis uurin teisi sarnase otstarbega rakendusi. Keskendun peamiselt rakenduste laoseisu ja arvete halduse funktsionaalsusele, kuna need on loodavas rakenduses kesksel kohal.

2.1.1 Zoho Inventory

Zoho Inventory on tasuline ingliskeelne pilvepõhine veebirakendus laohalduse teostamiseks. See võimaldab laoseisu kontrollida, hallata ladusid ja tellimusi, koostada ostu- ja müügiarveid ning saada ülevaade tehingute statistikast. [1]

Rakenduse kasutamiseks tuleb luua kasutaja. Kasutaja loomiseks makseandmeid sisestama ei pea, kuna rakendusel on 14-päevane prooviversioon, mis võimaldab kasutada kõiki funktsionaalsusi. Rakendusel on ka tasuta versioon, kuid seda kasutades saab kasutada vaid ühte ladu ja luua 20 arvet kuus. Alates $39 kuus on võimalik kasutada 2 ladu ja luua 1500 arvet kuus. [1] [2]

Mitme lao kasutamiseks tuleb rakenduse sätetes aktiveerida mitme lao režiim. Uue kauba lisamiseks on kohustuslik sisestada kauba nimi, kood ja ühik. On võimalik valida, kas soovin müügi-, ostu-, ja laoseisu andmeid lisada, kuid ei saa valida, millisesse lattu kaup lisada. Kaup lisatakse automaatselt põhilattu, millest ümber tõstmiseks tuleb eraldi luua *transfer order* ehk ümbertõstmine. Kaupade nimekirjas ei saa valida, millise lao kaupa vaadata tahan, vaid kuvatakse kõik süsteemis olevad kaubad korraga.

Müügiarve lisamiseks tuleb sisestada kliendi nimi, arve number, arve kuupäev, maksetähtaeg ja ladu, millest kaupa müüa. Arvesse kaupade lisamisel ei saa valida individuaalselt, millisest laost konkreetset kaupa müüa, vaid kõik kaubad peavad olema eelnevalt valitud laost. Kui valitud laos kaupa ei ole, siis võetakse kaup maha sellest laost, kuhu see esimesena lisati. Pärast arve loomist saab arve PDF formaadis alla laadida, kuid arve on ingliskeelne. Arve malle saab küll läbi Zoho Invoice rakenduse muuta, kuid kõiki

välju siiski eestikeelseks teha pole võimalik. Ostuarvete lisamine toimib sarnasel viisil, nagu müügiarvete lisamine.

Rakendusel on olemas ka mobiiliversioon iOS seadmetele App Store'is ja Android seadmetele Google Play Store'is.

2.1.2 TradeGecko

TradeGecko on ingliskeelne tasuline veebipõhine tarkvara laohalduse teostamiseks väikestele ja keskmise suurusega ettevõtetele. See võimaldab muuhulgas laoseisu kontrollida, arveid koostada, saada ülevaade finantsstatistikast, ühendada laoseis e-poe raamistikega ja makseid teostada. [2]

Rakenduse kasutamiseks tuleb luua kasutaja. Sarnaselt Zoho Inventoryle on ka TradeGeckol tasuta 14-päevane prooviversioon, kuid TradeGeckol edaspidine tasuta pakett puudub. Kõige odavama paketiga ($39/kuu) saab kasutada vaid ühte ladu ja luua 50 müügiarvet kuus. Mitme lao võimalust saab kasutada alates $199/kuu paketist. [2]

Kauba lisamisel on kohustuslik sisestada vaid kauba nimetus ja algne kogus. Erinevalt Zoho Inventoryst saab kauba lisamisel valida, millisesse lattu kaup lisada. Ühest kaubast on võimalik luua mitu varianti - näiteks saab luua kauba „T-särk" ja teha sellest mitu varianti iga t-särgi värvi jaoks, kusjuures igale variandile saab lisada oma andmed.

Müügiarve lisamisel on kohustuslik sisestada klient, arve number, müügikuupäev ja valida, millisest laost kaupa müüa. Ka TradeGeckos ei saa müügiarvele kaupade lisamisel individuaalselt valida, millisest laost kaupa müüa, st kogu kaup peab olema müüdud ühest laost. Pärast arve loomist võetakse kaup kohe laost maha. Arve saab PDF formaadis alla laadida, kuid arve on inglise keeles ja erinevalt Zoho Inventoryst pole võimalik arve väljade nimetusi muuta. Ostuarve lisamine toimib sarnaselt müügiarve lisamisega.

Ka TradeGeckol on olemas iOS seadmetele mõeldud mobiilirakendus.

2.1.3 Wave

Wave on väikeettevõtetele mõeldud tasuta ingliskeelne veebirakendus raamatupidamise teostamiseks, arvete koostamiseks, maksete jälgimiseks ja palgaarvestuseks. Erinevalt teistest käsitletavatest rakendustest ei paku Wave laohalduse võimalust. [3]

Müügiarve loomiseks peab valima kliendi ja sisestama arve numbri, müügikuupäeva ja maksetähtaja. Laoarvestust küll ei toimu, kuid kaupa arvesse sisestades jätab rakendus eelnevates arvetes sisaldunud kaubad meelde, et need edaspidi kiiresti sisestatavad oleks. Sarnaselt eelnevate rakendustega on ka Wave'iga võimalik koostada ainult ingliskeelseid arveid.

Wave'il on olemas mobiilirakendus nii iOS kui Android seadmetele.

2.1.4 Alternatiivsete lahenduste analüüs

Eespool nimetatud rakendused on pigem suunatud suurematele müügiga tegelevatele ettevõtetele ja seetõttu sisaldavad funktsionaalsusi, mida näiteks FIEdel vaja pole. Lisaks on rakendused ingliskeelsed ja ei võimalda koostada eestikeelseid arveid. Samuti peab mitme lao kasutamise võimaluse eest maksma juurde.

Tabel 1. Analüüsitud rakenduste võrdlus.

Nimi	Hind	Platvormid	Laohaldus	Eestikeelsed arved
Zoho Inventory	Tasuta – 1 ladu ja 20 arvet kuus. $39 – 2 ladu ja 1500 arvet kuus.	Veeb, iOS, Android	Jah	Ei
TradeGecko	$39 – 1 ladu ja 50 arvet kuus. $199 – mitu ladu ja 1000 arvet kuus.	Veeb, iOS	Jah	Ei
Wave	Tasuta	Veeb, iOS, Android	Ei	Ei

2.2 Nõuded

Antud peatükis kirjeldan rakenduse funktsionaalseid ja mittefunktsionaalseid nõudeid. Funktsionaalsed nõuded on parema hallatavuse mõttes jaotatud eraldi allsüsteemideks.

2.2.1 Funktsionaalsed nõuded

Ladude halduse allsüsteem

Tabel 2. Ladude halduse allsüsteemi nõuded.

Identifikaator	Nõue	Lisatingimused
fn1_1	Kasutaja peab saama lisada uut ladu.	Lao lisamisel on kohustuslik sisestada lao nimi. Lao nimi on unikaalne.
fn1_2	Kasutaja peab saama ladude nimekirja vaadata.	-
fn1_3	Kasutaja peab saama ladu kustutada.	Ladu saab kustutada vaid siis, kui seal pole ühtegi kaupa.
fn1_4	Kasutaja peab saama lattu kaupa lisada.	Kauba lisamisel on kohustuslik sisestada kauba kood, nimetus, ladu, kogus ja ühik. Kauba kood on unikaalne. Kasutaja peab saama laokaupade nimekirja vaadata. Nimekirjas on kuvatud kauba kood, nimetus, müügihind ja kogus koos ühikuga.
fn1_5	Kasutaja peab saama lisada uut ühikut.	Ühiku lisamisel on kohustuslik sisestada ühiku nimetus ja lühend. Ühiku lühend on unikaalne.
fn1_6	Kasutaja peab saama laokauba detaile vaadata.	Detailides on kuvatud kauba kood, nimetus, tarnija nimi, ladu, ostuhind, müügihind, kogus, ühik ja märkused.
fn1_7	Kasutaja peab saama laokaupa kustutada.	Kaupa saab kustutada vaid siis, kui see pole seotud ühegi arvega.

fn1_8	Kasutaja peab saama laokaupa muuta.	-
fn1_9	Kasutaja peab saama ühikut muuta.	-
fn1_10	Kasutaja peab saama ühikut kustutada.	Ühikut ei saa kustutada, kui see on mõne kaubaga seotud.
fn1_11	Kasutaja peab saama ladu muuta.	-

Kontode halduse allsüsteem

Tabel 3. Kontode halduse allsüsteemi nõuded.

Identifikaator	Nõue	Lisatingimused
fn2_1	Rakendust saab käidelda ainult autoriseeritud kasutaja.	-
fn2_2	Kasutaja peab saama konto sätteid muuta.	-
fn2_3	Kasutaja peab saama sisse logida salasõna ja parooliga või Google'i/Facebooki kontoga	-
fn2_4	Kasutaja peab saama profiili andmeid muuta.	Profiili andmetes on kasutaja ettevõtte nimi, registrikood ja kasutaja kontaktandmed.
fn2_5	Kasutaja peab saama pangakontosid lisada.	Pangakontode lisamisel on kohustuslik sisestada panga nimi ja konto number.
fn2_6	Kasutaja peab saama endaga seotud pangakontosid kustutada.	-

Ostuarvete halduse allsüsteem

Tabel 4. Ostuarvete halduse allsüsteemi nõuded.

Identifikaator	Nõue	Lisatingimused
fn3_1	Kasutaja peab saama ostuarvet luua.	Ostuarve lisamisel on kohustuslik sisestada tarnija, arve nr, ostukuupäev, maksetähtaeg ja kaubad. Kasutaja võib ostuarvele lisada ühe PDF formaadis dokumendi.
fn3_2	Kasutaja peab saama ostuarvete nimekirja vaadata.	Nimekirjas on kuvatud arve nr, tarnija nimi, ostukuupäev, maksetähtaeg, arve summa ja kas arve on makstud või mitte.
fn3_3	Kasutaja peab saama ostuarvesse kaupa lisada.	Kaubal on 3 tüüpi: laokaup, teenus ja kuluartikkel. Laokauba lisamisel on kohustuslik sisestada kauba nimetus, kood, kogus, ühik, ladu ja ostuhind. Kuluartikli ja teenuse lisamisel on kohustuslik sisestada kauba nimetus, kogus ja ostuhind. Laokaup võetakse lattu arvele, teenus ja kuluartikkel lisanduvad ainult arve summale. Kui vastava koodiga laokaup on juba laos olemas, siis uuendatakse kauba kogust.
fn3_4	Kasutaja peab saama ostuarve detaile vaadata.	Detailides on kuvatud tarnija nimi, arve nr, arve fail, ostukuupäev, maksetähtaeg, märkused ja ostetud kaubad.
fn3_5	Kasutaja peab saama ostuarvet kustutada.	Ostuarvega seotud kaubad võetakse laost ära.

		Kui ostuarve on makstud, siis ostuarvega seotud kulud kustutatakse.
fn3_6	Kasutaja peab saama ostuarveid kõigi parameetrite järgi otsida.	-
fn3_7	Kasutaja peab saama ostuarvet muuta.	-

Müügiarvete halduse allsüsteem

Tabel 5. Müügiarvete halduse allsüsteemi nõuded.

Identifikaator	Nõue	Lisatingimused
fn4_1	Kasutaja peab saama müügiarvet luua.	Müügiarve lisamisel on kohustuslik sisestada klient, arve nr, müügikuupäev, maksetähtaeg ja müüdavad kaubad.
fn4_2	Kasutaja peab saama müügiarvete nimekirja vaadata.	Nimekirjas on kuvatud arve nr, kliendi nimi, müügikuupäev, maksetähtaeg, arve summa, makstud summa ja maksmata summa.
fn4_3	Kasutaja peab saama müügiarvet PDF failina alla laadida.	Müügiarves peavad olema kuvatud dokumendi nimetus ja number, koostamise kuupäev, tehingu sisu, tehingu arvnäitajad (kogus, hind, summa), osapoolte nimed, osapoolte aadressid vastavalt Raamatupidamisseaduse §7-le [4].
fn4_4	Kasutaja peab saama müügiarvele kaupa lisada.	Kaubal on 3 tüüpi: laokaup, teenus ja kuluartikkel. Laokauba lisamisel on kohustuslik sisestada kauba nimetus, kood, kogus, ühik ja müügihind.

		Kuluartikli ja teenuse lisamisel on kohustuslik sisestada nimetus, kogus ja müügihind. Müüdav laokaup võetakse laost maha, teenus ja kuluartikkel lisanduvad ainult arve summale. Kui müüdavat laokaupa pole laos piisavalt, siis kuvatakse veateade.
fn4_5	Kasutaja peab saama müügiarve detaile vaadata.	Müügiarve detailvaates on kuvatud kliendi nimi, arve nr, arve fail, müügikuupäev, maksetähtaeg, märkused ja müüdud kaubad.
fn4_6	Kasutaja peab saama müügiarvet kustutada.	Müügiarvega seotud laokaubad pannakse lattu tagasi. Müügiarvega seotud laekumised ja tulud kustutatakse.
fn4_7	Kasutaja peab saama müügiarvet muuta.	-

Kulude halduse allsüsteem

Tabel 6. Kulude halduse allsüsteemi nõuded.

Identifikaator	Nõue	Lisatingimused
fn5_1	Kasutaja peab saama kulude nimekirja vaadata.	Nimekirjas on kuvatud kuluga seotud ostuarve nr, tarnija nimi, kulu kuupäev ja summa.
fn5_2	Kasutaja peab saama kulu detaile vaadata.	Detailides on kuvatud ostuarve nr, tarnija nimi, kulu tekkimise kuupäev, summa ja märkused.
fn5_3	Kasutaja peab saama kulu kustutada.	Kulu kustutamisel läheb kuluga seotud müügiarve maksmata seisundisse.

fn5_4	Kasutaja peab saama ostuarvet kuludesse lisada.	Kulu lisamisel on kohustuslik valida makstav ostuarve ja arve eest maksmise kuupäev.

Laekumiste halduse allsüsteem

Tabel 7. Laekumiste halduse allsüsteemi nõuded.

Identifikaator	Nõue	Lisatingimused
fn6_1	Kasutaja peab saama müügiarvega seotud laekumisi lisada.	Laekumise lisamisel on kohustuslik sisestada müügiarve, laekumise kuupäev ja summa. Laekumise lisamisel lisatakse laekumisega seotud müügiarve makstud summale laekumise summa juurde. Kui laekumise lisamisel saab müügiarve täielikult makstud, siis lisatakse müügiarve ka tuludesse.
fn6_2	Kasutaja peab saama laekumiste nimekirja vaadata.	Nimekirjas on kuvatud laekumisega seotud müügiarve nr, kliendi nimi, laekumise kuupäev ja summa.
fn6_3	Kasutaja peab saama laekumise detaile vaadata.	Detailides on kuvatud müügiarve nr, kliendi nimi, laekumise kuupäev, summa ja märkused.
fn6_4	Kasutaja peab saama laekumist kustutada.	Laekumisega seotud müügiarve makstud summa väheneb laekumise summa võrra. Kui laekumisega seotud müügiarve on makstud, siis müügiarve tulu kustutatakse.
fn6_5	Kasutaja peab saama laekumist muuta.	Kui laekumisega seotud müügiarve on makstud ja laekumise summat vähendatakse, siis müügiarve tulu kustutatakse.

Tulude halduse allsüsteem

Tabel 8. Tulude halduse allsüsteemi nõuded.

Identifikaator	Nõue	Lisatingimused
fn7_1	Kasutaja peab saama tulude nimekirja vaadata.	Nimekirjas on kuvatud tuluga seotud müügiarve nr, kliendi nimi, tulu tekkimise kuupäev ja summa.
fn7_2	Kasutaja peab saama tulu detaile vaadata.	Detailides on kuvatud müügiarve nr, kliendi nimi, tulu tekkimise kuupäev, summa ja märkused.
fn7_3	Kasutaja peab saama tulu kustutada.	Tulu kustutamisel nullitakse tuluga seotud müügiarve makstud summa ja kustutatakse müügiarvega seotud laekumised.

Partnerite halduse allsüsteem

Tabel 9. Partnerite halduse allsüsteemi nõuded.

Identifikaator	Nõue	Lisatingimused
fn8_1	Kasutaja peab saama partnerit lisada.	Partneri lisamisel on kohustuslik sisestada partneri nimi ja partneri tüüp (klient või tarnija).
fn8_2	Kasutaja peab saama partnerite nimekirja vaadata.	Nimekirjas on kuvatud partneri tüüp, nimi, e-meili aadress ja telefoninumber.
fn8_3	Kasutaja peab saama partnerit kustutada.	Partnerit ei saa kustutada, kui ta on mõne arve või kaubaga seotud.
fn8_4	Kasutaja peab saama partnerit muuta.	-

Kasutaja töölaud (*dashboard*)

Tabel 10. Kasutaja töölaua nõuded.

Identifikaator	Nõue	Lisatingimused
fn9_1	Kasutaja peab saama näha kulude ja tulude statistikat.	Saab valida statistika arvestuse alguskuupäeva ja lõppkuupäeva. On kuvatud graafik, mis näitab päevade või kuude lõikes tulude ja kulude summasid. On kuvatud kulude, tulude ja kasumi kogusumma valitud perioodi jooksul.

2.2.2 Mittefunktsionaalsed nõuded

Identifikaator	Nõue
mfn_1	Rakenduse kasutajaliides on eestikeelne.
mfn_2	Rakenduse lähtekood on ingliskeelne.
mfn_3	Põhivaatest saab kõiki operatsioone teha vähem kui 3 klikiga (v.a klikid vormide täitmiseks).
mfn_4	Rakendus on kasutatav Google Chrome, Mozilla Firefox ja Microsoft Edge brauserites.
mfn_5	Kõik kasutaja poolt saadetud vormide andmed peavad olema serveri poolt valideeritud.

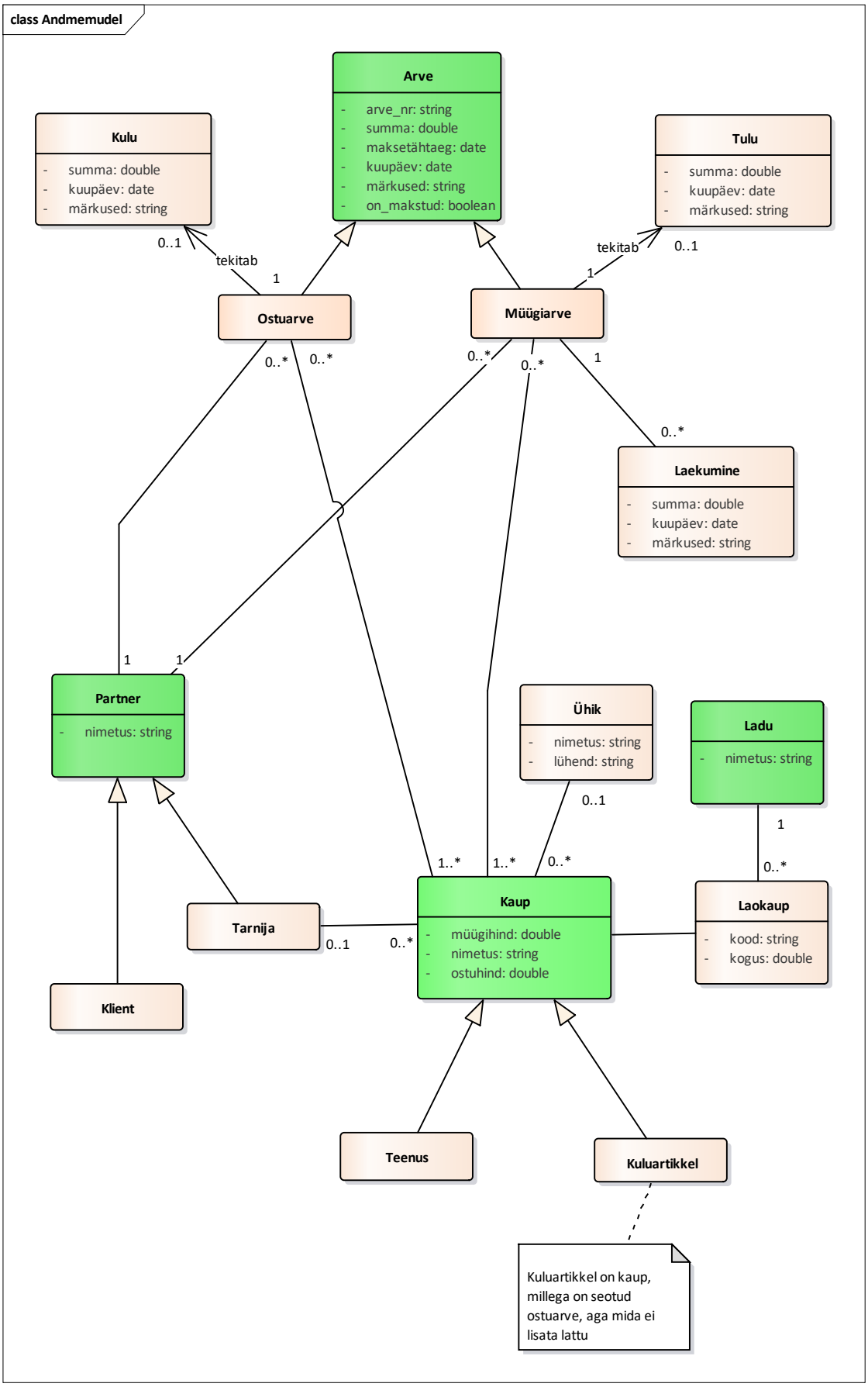
2.3 Andmemudel

Süsteemi põhiolemid on kaup, arve, partner ja ladu. Järngevalt on kirjeldatud nende suhted teiste olemitega.

2.3.1 Olemite semantika

- Kaup on laokaup, teenus või kuluartikkel.
- Laokaubal on ühik.
- Laokaup on laos.
- Teenusel ja kuluartiklil võib olla ühik.

- Kaubal võib olla tarnija.
- Partner on klient või tarnija.
- Arve on ostuarve või müügiarve.
- Müügiarve on suunatud kliendile.
- Ostuarve on saadud tarnijalt.
- Ühes arves sisaldub mitu kaupa.
- Üks kaup võib sisalduda mitmes arves.
- Klient teeb müügiarve eest laekumise.
- Makstud müügiarve tekitab tulu.
- Makstud ostuarve tekitab kulu.



Joonis 1. Andmemudel.

3 Rakenduse realisatsioon

Antud töö eesmärgiks on luua kassapõhise raamatupidamisega ettevõtetele mõeldud veebipõhine laotarkvara, mis võimaldab pidada laoarvestust, koostada arveid ja saada ülevaade raamatupidamisest.

Enne rakenduse kirjutama hakkamist konsulteerisin rakenduse potentsiaalse kasutajaga ja uurisin teisi taolisi rakendusi, et teha selgeks põhifunktsionaalsused, mida rakendus sisaldama peab. Saadud informatsiooni põhjal lõin rakenduse funktsionaalse prototüübi, mida samuti kasutaja peal testisin. Kasutasin rakenduse arendamisel *front-end-first* lähenemist ehk keskendusin esmalt kasutajaliidese arendamisele kasutades staatilisi andmeid ja alles pärast kasutajaliidese põhifunktsionaalsuse valmimist hakkasin serveripoolset funktsionaalsust arendama, et asendada staatilised andmed dünaamilistega.

3.1 Tehnoloogiate valik

3.1.1 Figma

Figma on kasutajaliideste disainimiseks ja protüüpimiseks mõeldud rakendus, mida saab kasutada nii veebis kui ka Windowsi, MacOSi või Linuxi põlisrakendusena. Sellel on olemas kõik vajalikud tööriistad veebikomponentide kujundamiseks ja klikitavate funktsionaalsete prototüüpide loomiseks. Figma on ainus disainirakendus, mis võimaldab mitmel kasutajal korraga reaalajas koostööd teha. [5]

Valisin kasutajaliidese prototüüpi loomiseks Figma, kuna sellel on olemas funktsionaalse prototüübi loomiseks kõik vajalikud tööriistad ja kasutan arendamiseks nii Windowsil kui MacOSil töötavaid arvuteid ja veebirakendusena on Figma mugavalt kasutatav mõlemal operatsioonisüsteemil. Lisaks olen Figmat ka eelnevalt kasutanud.

3.1.2 React

React on Facebooki poolt arendatav avatud lähtekoodiga JavaScripti teek komponendipõhiste kasutajaliideste loomiseks. See võimaldab luua üheleherakendusi kasutades korduvkasutatavaid olekuga komponente. Komponendi oleku muutuse korral renderdatakse komponent uute andmetega uuesti, mistõttu on selline struktuur dünaamiliste andmete esitamiseks väga tõhus. [6]

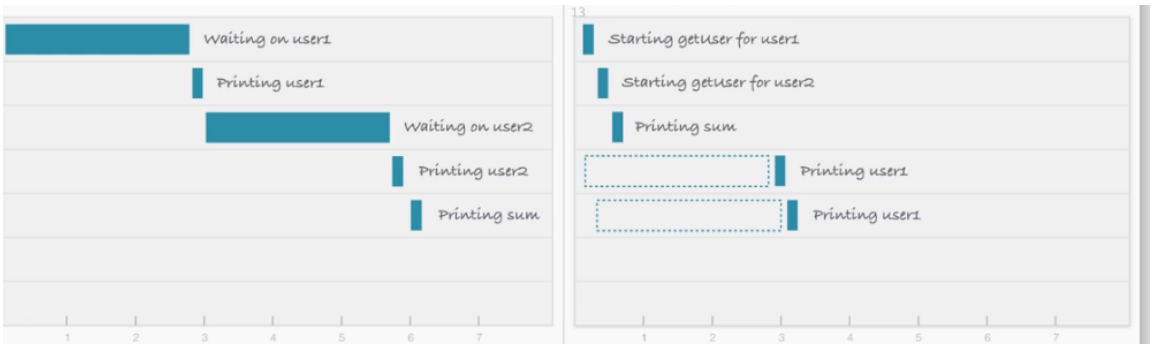
Reacti programmeerimiskeeleks on JSX (JavaScript XML), mis on laiendus JavaScripti süntaksile. See sarnaneb JavaScriptiga, kuid muudab Reacti komponentide kirjutamise mugavamaks ja loetavamaks. [6]

Valisin rakenduse loomiseks Reacti, kuna olen seda eelnevalt mitmel korral kasutanud ja oskan seda võrreldes teiste kasutajaliideste raamistikega kõige paremini kasutada.

3.1.3 Node.js

Node.js on avatud lähtekoodiga Chrome V8 mootoril töötav mitmeplatvormne JavaScripti käitussüsteem, mis võimaldab käivitada JavaScripti brauseriväliselt ja kasutada seda käsureatööriistade ja serveripoolsete skriptide kirjutamiseks. See esindab „JavaScript kõikjal“ paradigmat, võimaldades rakenduse arendamiseks kasutada JavaScripti nii serveri- kui kliendipoolses arenduses. [7]

Node.js kasutab mitteblokeerivat sisend/väljund mudelit, mis tähendab, et kui kaks kasutajat teevad serverile samaaegselt päringuid, siis teine kasutaja ei pea ootama esimese kasutaja päringu lõpuleviimist, vaid mõlemad päringud võetakse korraga vastu ja käivitatakse paralleelselt (Joonis 2). Samasugust funktsionaalsust on võimalik saavutada ka lõimtöötluse (*multi-threading*) abil, kuid see nõuab rohkem muutmälu ja protsessori jõudlust. [8]



Joonis 2. Päringute töötlemise järjekord blokeeriva (vasakul) ja mitteblokeeriva (paremal) sisend/väljund mudeli korral.

Valisin rakenduse serveripoole realiseerimiseks Node.js-i, kuna rakenduse funktsionaalsus ei hõlma protsessorit intensiivselt kasutavaid tegevusi, milleks lõimtöötluse võimalusega keel sobiks rohkem. Samuti olen JavaScriptiga rohkem tegelenud ja olen selle struktuuri ja võimalustega tuttavam kui teiste programmeerimiskeeltega.

3.1.4 Express

Express on avatud lähtekoodiga serveripoolne veebirakenduste raamistik Node.js-ile. See on disainitud veebirakenduste ja API-de loomiseks. See abstraheerib Node.js-i põliseid HTTP päringutega tegelevaid funktsioone, pakub mugava liidese API-de loomiseks ja võimaldab kasutada vahefunktsioone (*middleware*), et sissetulevaid päringuid töödelda. [9, 10]

3.1.5 PostgreSQL

PostgreSQL on avatud lähtekoodiga relatsiooniline andmebaasisüsteem, mis paneb rõhku laiendatavusele ja standarditele vastavusele. Seda arendab PostgreSQL Global Development Group - mitmekülgne grupp paljudest ettevõtetest ja individuaalsetest arendajatest. PostgreSQLi populaarsus on viimastel aastatel järsult tõusnud ja hetkel on see MySQLi järel populaarsuselt teine andmebaasisüsteem [11, 12].

Põhjused, miks eelistada PostgreSQLi MySQLile:

- PostgreSQL on avatud lähtekoodiga ja avaldatud PostgreSQL litsentsi all, mis sarnaneb MIT litsentsiga. MySQLi lähtekood on küll avalik, kuid Oracle pakub sellest ka tasulisi versioone; [13]
- PostgreSQL on kiirem keeruliste päringute puhul, MySQL on kiirem lihtsamate päringute puhul; [13]
- PostgreSQL toetab rohkem andmetüüpe; [14]
- PostgreSQL paneb rõhku SQL standardi järgimisele ja hõlmab endas peaaegu kõiki SQL standardi põhifunktsionaalsusi. MySQL järgib SQL standardit ainult osaliselt. [13]

Valisin rakenduse andmebaasisüsteemiks PostgreSQLi, kuna olen seda eelnevalt kasutanud ja oskan seda kõige paremini rakendada.

3.1.6 Sequelize

Sequelize on Node.js-ile mõeldud ORM, mida saab kasutada PostgreSQLi, MySQLi, MariaDB, SQLite ja Microsoft SQL Serveri andmebaasisüsteemidega ühendamiseks. [15] Sequelize loob liidese Node.js-i ja andmebaasi vahel. Sellega saab defineerida

andmebaasi skeemi ja teha SQL päringuid JavaScripti kaudu ilma SQLi kirjutamata. Kuna Sequelize'i süntaks jääb olenemata andmebaasist samaks, siis muudab selle kasutamine ka andmebaasisüsteemi vahetamise lihtsamaks. [16]

Kasutan rakenduse andmebaasi skeemi defineerimiseks ja päringute tegemiseks Sequelize'i, kuna see lihtsustab andmebaasi initsialiseerimist, SQL päringute kirjutamist ja nende tulemuste töötlemist.

3.2 Arenduskeskkonna seadistamine

Arenduskeskkonna seadistamiseks peavad esmalt olema arvutisse installeeritud PostgreSQL alates versioonist 10**,** Node.js alates versioonist 10 ja git [17]. Esmalt tuleb projekt oma arvutisse kloonida – selleks tuleb käsurealt liikuda `cd` käsuga kausta, kuhu soovitakse projekt kopeerida ja sisestada järgmine käsk:

```
git clone https://gitlab.cs.ttu.ee/stjuks/iapb
```

3.2.1 Serveripoolse arenduskeskkonna seadistamine

Serveripoolse arenduskeskkonna seadistamiseks tuleb teha järgmised tegevused:

- luua PostgreSQLis uus vabalt valitud nimega andmebaas;
- Google'i kontoga autentimise lubamiseks luua Google Developer Console'is [18] uus projekt, et omandada *client ID* ja *client secret. Authorized JavaScript origin*ide hulka tuleb lisada http://localhost:5000 ning http://localhost:3000 ja *Authorized redirect URI*de hulka http://localhost:5000/api/auth/google/callback;
- luua *app/api* kausta fail nimega .env, mille sisuks on Google Developer Console'ist saadud *client ID* ja *client secret*, vabalt valitud JSON Web Tokeni [19] salasõna, millega token allkirjastatakse ja andmebaasi andmed. Faili mall on näidatud allpool (Joonis 3);
- serveri käivitamiseks tuleb käsurealt liikuda `cd` käsuga *app/api* kausta ja sisestada järgmised käsud:
    - `npm install`
    - `node index`

- kuna Sequelize'iga ei ole võimalik kõiki kitsendusi JavaScripti kaudu lisada, siis peab nende lisamiseks kasutama migratsioone. Migratsioonide failid asuvad *app/api/db/migrations* kaustas. Need lisavad andmebaasi staatilistese tabelitesse andmed ja jõustavad mõned välisvõtme ja *check* kitsendused. Migratsioonide käivitamiseks tuleb avada uus käsurea aken, liikuda *app/api* kausta ja sisestada järgmised käsud:
    - `npm install -g sequelize-cli`
    - `sequelize db:migrate`

Kui kõik käsud said edukalt sisestatud, on andmebaasi skeem loodud ja server kättesaadav aadressil http://localhost:5000/api.

```
GOOGLE_CLIENT_ID=sinu_google_client_id
GOOGLE_CLIENT_SECRET=sinu_google_client_secret
JWT_SECRET=jwt_salasõna
DB_NAME=andmebaasi_nimi
DB_USER=andmebaasi_kasutajanimi
DB_PASSWORD=andmebaasi_parool
DB_HOST=andmebaasi_host
```

Joonis 3. env faili sisu.

3.2.2 Kliendipoolse arenduskeskkonna seadistamine

Kliendipoolse arenduskeskkonna seadistamiseks tuleb käsurealt liikuda app/ui kausta ja sisestada järgmised käsud:

- `npm install`
- `npm start`

Käskude edukal sisestamisel on rakendus kättesaadav aadressil http://localhost:3000.

3.3 Kasutajaliides

Rakenduse kasutajaliidese realiseerimiseks lõin esmalt Figmas rakenduse funktsionaalse prototüübi ja seejärel arendasin selle järgi Reactis kasutajaliidese. Projekti loomisel kasutasin Reacti poolt pakutavat moodulit nimega Create React App [20], mis kõigest

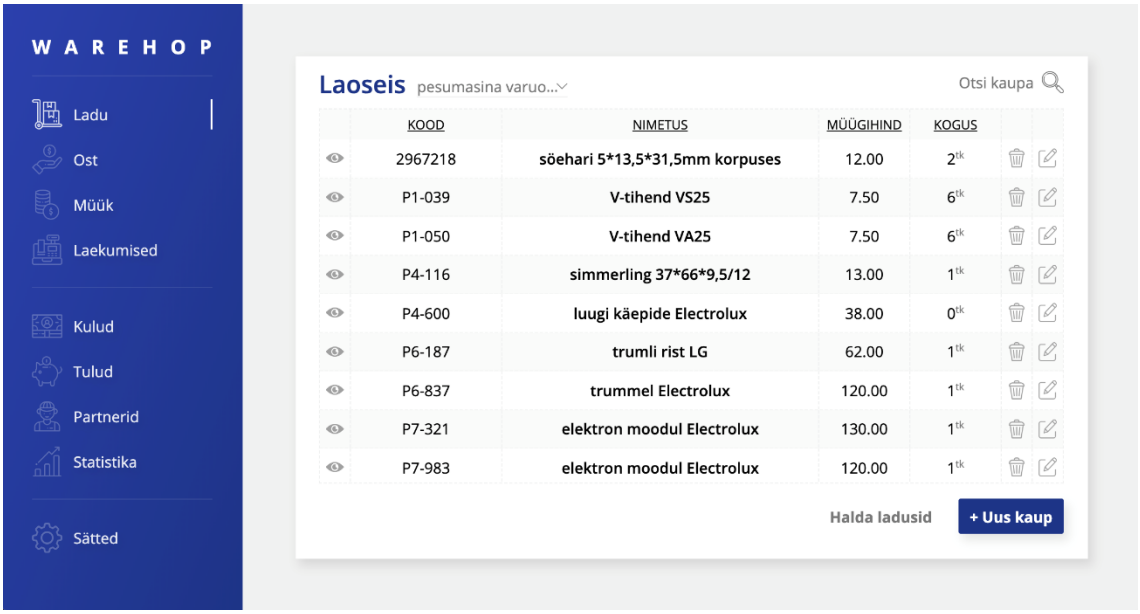
ühe käsuga loob ja konfigureerib automaatselt Reacti projekti nii, et arendaja saab kohe hakata koodi kirjutama.

3.3.1 Vaated

Rakenduse kujundamisel lähtusin põhimõttest, et iga vaate korral peab olema kohe selge, mis selles vaates näha on ja mida teha saab. Vaated peavad olema võimalikult sarnased, et erinevate vaadete korral oleks samad tegevused kiiresti ülesleitavad. Samuti peaks olema vaate põhitegevus nupuna esile tõstetud (näiteks kauba lisamine).

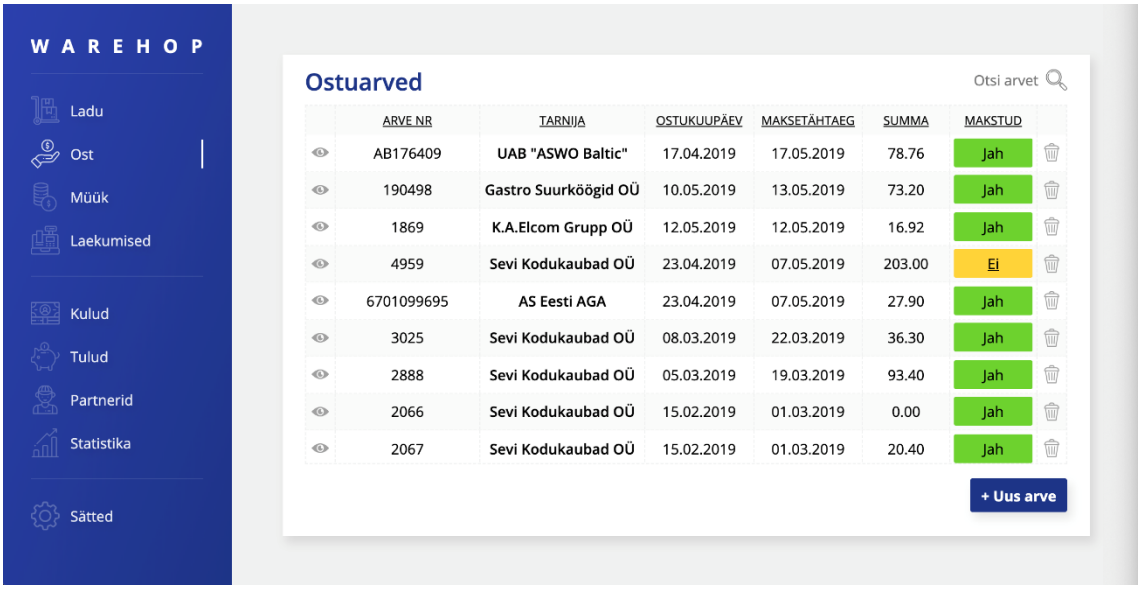
Rakenduses on kokku 9 põhivaadet: laoseis, ostuarved, müügiarved, laekumised, kulud, tulud, partnerid, statistika ehk *dashboard* ja sätted. Kõik põhivaated peale statistika ja sätete on realiseeritud sarnase ülesehitusega tabelitena. Lisaks põhivaadetele on olemas ka modaalidena realiseeritud vormide vaated, millega saab andmeid lisada ja muuta.

Laoseisu vaates (Joonis 4) on kuvatud laokaupade tabel. Tabeli kohal vasakul on rippmenüü, kust saab valida, millise lao kaupa vaadata. Paremal on otsingu väli, millega saab valitud laost kaupa otsida kauba nimetuse ja koodi järgi. Tabelis on kuvatud kauba kood, nimetus, müügihind ja kogus koos ühikuga. Lisaks on igal real olemas nupud kauba detailvaate vaatamiseks, kauba kustutamiseks ja muutmiseks. Tabelit on võimalik iga tulba väärtuste järgi kasvavas ja kahanevas järjekorras sorteerida. Tabeli jaluses on nupud ladude haldamiseks (uue lao lisamine või kustutamine) ja uue laokauba lisamiseks, mis avavad modaali vastava vormiga.

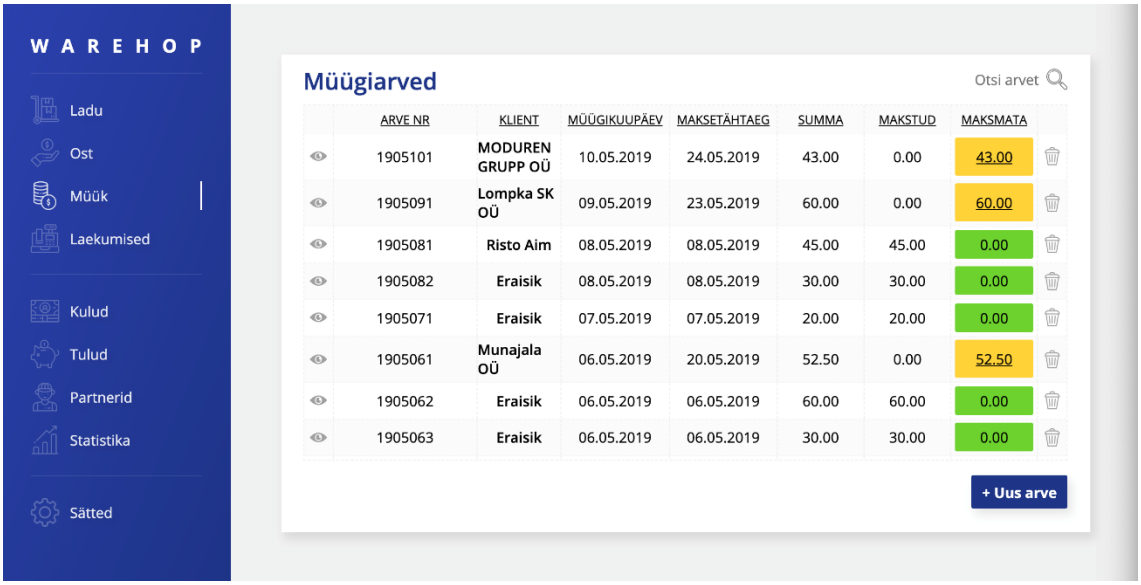


Joonis 4. Laoseisu vaade.

Ostuarvete (Joonis 5) ja müügiarvete (Joonis 6) vaadetes on kuvatud arvete nimekiri. Otsingu välja kaudu saab arveid otsida tarnija või kliendi nime ja arve numbri järgi. Ostuarvete tabelis on kuvatud ostuarve number, tarnija nimi, ostukuupäev, maksetähtaeg, arve summa ja kas arve on makstud või mitte. Müügiarvete tabelis on kuvatud müügiarve number, kliendi nimi, müügikuupäev, maksetähtaeg, arve summa, makstud summa ja maksmata summa. Makstud arved on nimekirjas märgitud rohelisega ja maksmata arved kollasega, et makstud ja maksmata arvetel lihtsamalt vahet teha. Maksmata arve real on võimalik kollasele tulbale vajutada, et avada ostuarve puhul kulu lisamise ja müügiarve puhul laekumise lisamise modaal ja mugavalt arve makstuks märkida. Arves sisalduvate kaupade vaatamiseks või arve faili allalaadimiseks peab avama arve detailvaate. Tabeli jaluses on nupp, mis avab uue arve lisamise modaali.

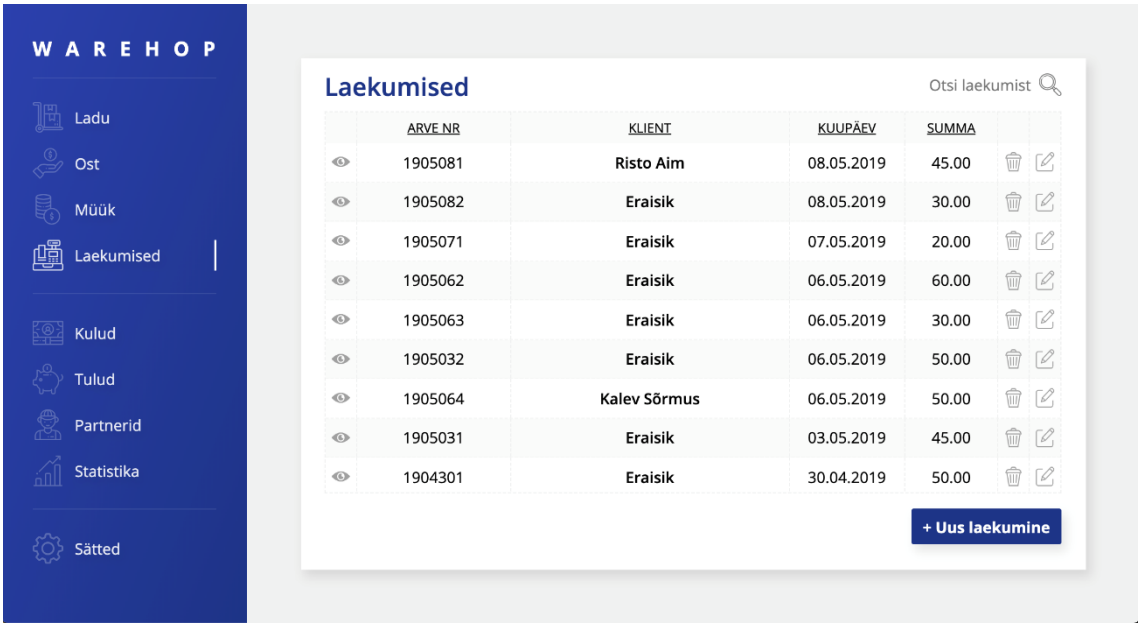


Joonis 5. Ostuarvete vaade.

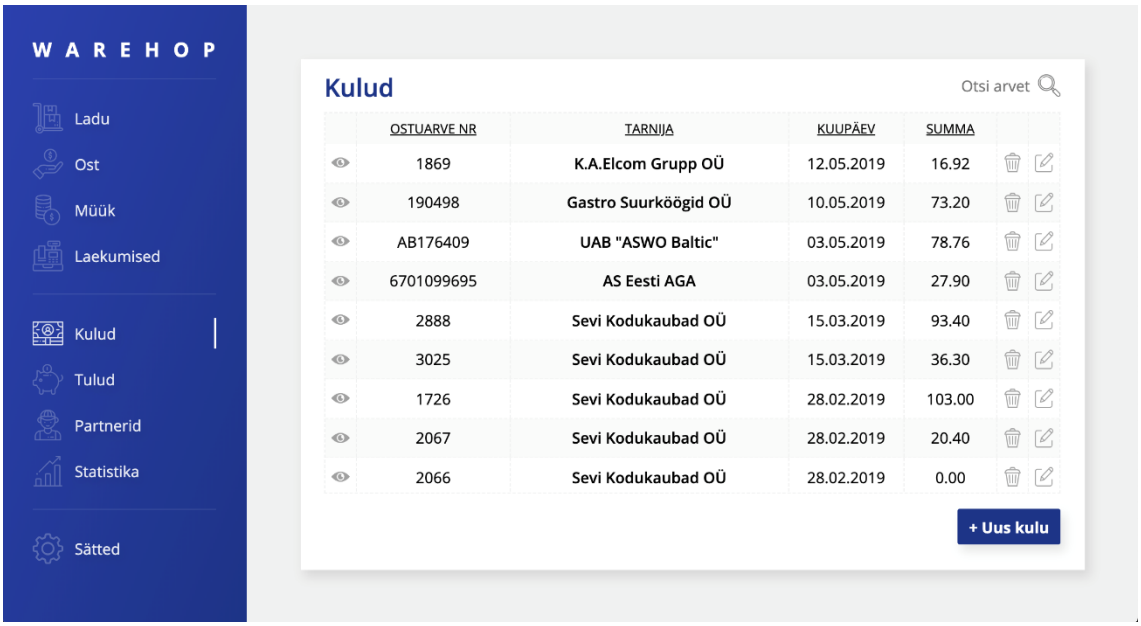


Joonis 6. Müügiarvete vaade.

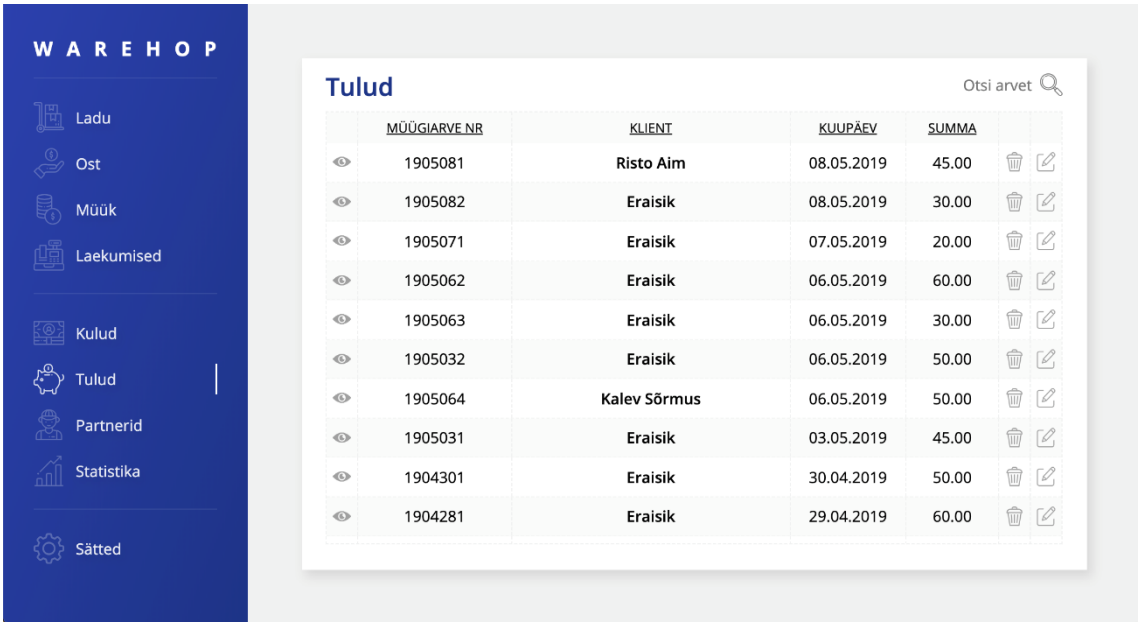
Järgnevates joonistes on näidatud laekumiste (Joonis 7), kulude (Joonis 8), tulude (Joonis 9) ja partnerite (Joonis 10) vaated. Need vaated kuvavad vastavate andmete nimekirju ja sisaldavad samu funktsionaalsusi nagu eelnevalt kirjeldatud vaated.



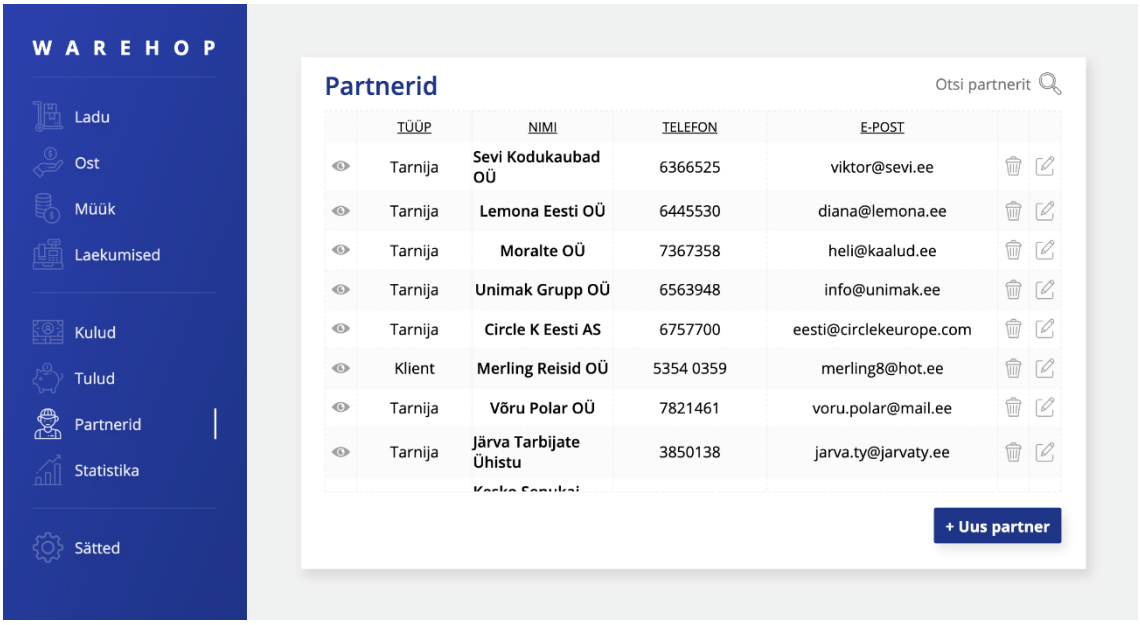
Joonis 7. Laekumiste vaade.



Joonis 8. Kulude vaade.

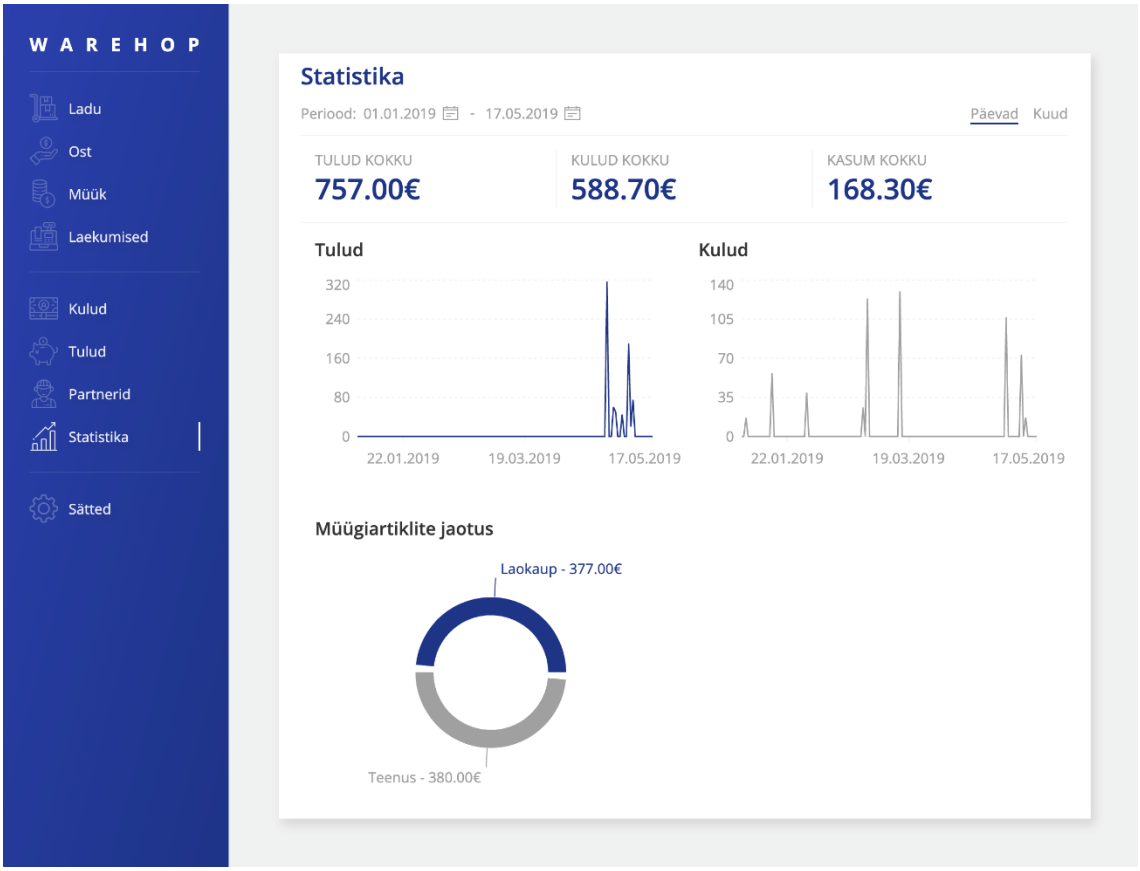


Joonis 9. Tulude vaade.



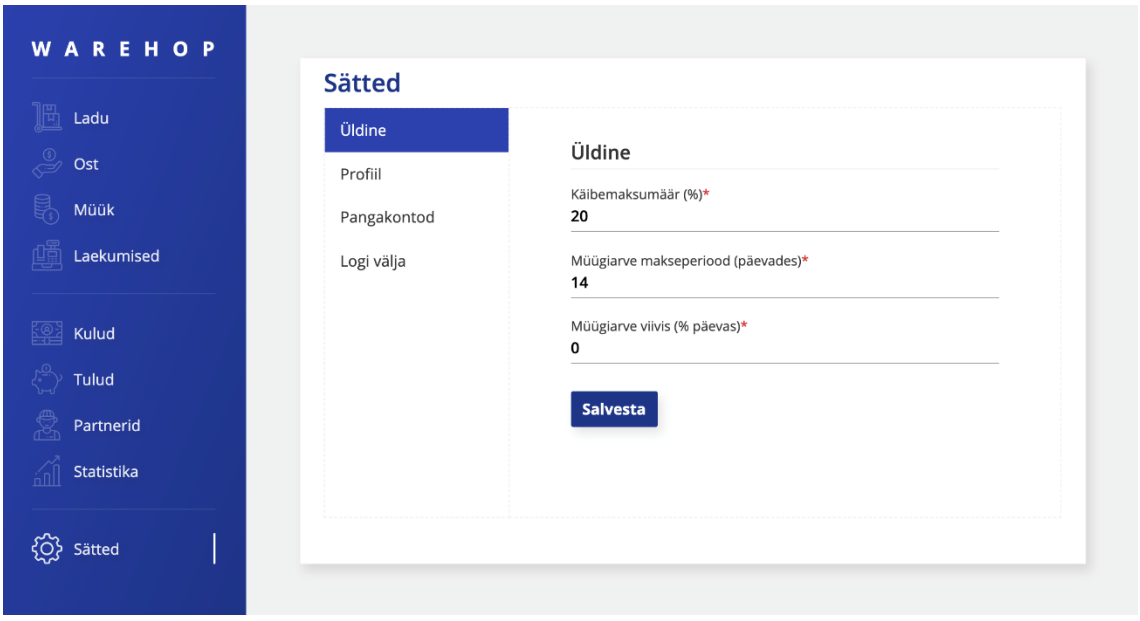
Joonis 10. Partnerite vaade.

Statistika vaates (Joonis 11) on kuvatud majandusaasta finantsstatistika. Lehe ülaosas vasakul saab valida perioodi, mille kohta statistikat kuvatakse ja paremal valida, kas graafikutel kuvatakse andmeid päevade või kuude kaupa. Statistika põhinäitajad on tulud kokku, kulud kokku ja kasum kokku. Samuti on graafikutelt võimalik näha tulusid ja kulusid päevade või kuude kaupa. Lisaks on kuvatud ka müügiartiklite jaotus ehk kui suure summa eest mis tüüpi artikleid on müüdud.



Joonis 11. Statistika vaade.

Sätete vaates (Joonis 12) on kuvatud sätete alammenüü ja valitud alammenüü komponendi sisu. Sätete vaates on võimalik muuta üldiseid sätteid, profiili andmeid, hallata pangakontosid ja logida rakendusest välja.



Joonis 12. Sätete vaade.

3.3.2 Olekuhaldus

Kuna loodavas rakenduses on palju dünaamilisi andmeid, mida serverist laadima ja rakenduse erinevates osades samaaegselt kasutama peab, siis ei piisa olekute haldamiseks Reacti komponendisiseste olekute kasutamisest, vaid on mõttekam rakendada olekuhalduse teeke nagu näiteks MobX [21] või Redux [22]. Olekuhalduse teegid hoiavad kogu rakenduse olekud ühes kohas koos ja võimaldavad neid muuta nii, et iga muutuse korral renderduvad ka konkreetset olekut kasutavad HTML komponendid uuesti. Kuna kogu rakenduse olekud on ühes kohas koos, siis on ka rakendust kergem hallata, kuna eksisteerib vaid üks „tõe allikas“. [23]

MobX on mõjutatud objektorienteeritud programmeerimise põhimõtetest ja reaktiivsest programmeerimisest, samas Redux järgib funktsionaalse programmeerimise põhimõtteid. Kuna mul on eelnev kogemus objektorienteeritud Java programmeerimisega, siis valisin ka realiseeritava rakenduse olekuhalduse teegiks MobXi, kuna selle struktuur on mulle käepärasem. Samuti on MobXiga olekuhaldust lihtsam teostada, kuna olekut võib otse muuta, samas kui Reduxis peab oleku muutmiseks väljastama *action*i, mille võtab vastu *reducer*, mis omakorda määrab olekule uue väärtuse. [23]

MobXil on kolme tüüpi olemite väärtuseid: *action*, *observable* ja *computed*. *Action* annotatsiooniga märgitakse funktsioonid, mis muudavad rakenduse olekut, *observable* annotatsiooniga märgitakse olekumuutujad ja *computed* annotatsiooniga funktsioonid, mis tagastavad oma väärtuse mingi olekumuutuja põhjal. Et Reacti komponent oleku muutustele reageeriks, peab komponendi märkima *observer*iks. MobX-i kasutamise koodinäide on toodud Joonis 13.

```
// TodoStore.js
class TodoStore extends Store {
    todos = [];

    addTodo = todo => {
        this.todos.push(todo);
    };
}

export default decorate(AppStore, {
    todos: observable,
    addTodo: action
})

// TodoList.js
class TodoList extends Component {
    render() {
        const { store } = this.props;

        return (
            <div>
                <button onClick={() => store.addTodo({ name: "todo" })}>
                    Add todo
                </button>
                {store.todos.map(todo => (
                    <div>{todo.name}</div>
                ))}
            </div>
        );
    }
}

export default observer(TodoList);

// index.js
const store = new TodoStore();

ReactDOM.render(<TodoList store={store} />, document.getElementById('root'));
```

Joonis 13. MobXi olekuhalduse koodinäide.

3.4 Andmebaas

Rakenduse andmebaas on loodud PostgreSQL andmebaasisüsteemiga ja andmebaasiga suhtlemine toimub Node.js-i ja Sequelize ORMi kaudu.

Andmebaasi skeemi loomisel tuli arvesse võtta seda, et põhiolemite primaarvõtmed ja nendele viitavad välisvõtmed peavad sisaldama lisaks olemi ID-le ka nendega seotud kasutaja ID-d, et vältida olukorda, kus näiteks uue kauba lisamisel valib kasutaja kauba ühikuks mõne teise kasutaja loodud ühiku. Lisaks tuli arvestada sellega, et üks kaup võib sisalduda samaaegselt mitmes laos, ostuarves ja müügiarves, kusjuures kui müügiarve ära kustutada, peab laokaup minema tagasi sinna lattu, kust ta müüdi.

3.5 Server

3.5.1 REST API disain

Serveri API aadress on http://localhost:5000/api. API disainimisel lähtusin põhimõttest, et päringut ei kirjelda mitte aadress, vaid HTTP meetod, mida päringu tegemisel kasutatakse. Selle põhimõtte järgimine aitab API aadresse lihtsamini hallata ja võimaldab GET, POST, PUT ja DELETE meetodeid kasutades teha samale aadressile erineva otstarbega päringuid. HTTP meetodite valimisel lähtusin järgmistest põhimõtetest:

- GET meetod on andmete pärimiseks, s.t päringu tulemusel andmebaasis andmed ei muutu;
- POST meetod on andmebaasi uute andmete lisamiseks;
- PUT meetod on andmebaasis juba eksisteerivate andmete muutmiseks;
- DELETE meetod on andmebaasi andmete kustutamiseks.

Näited kaupadega seotud API päringutest on toodud Tabel 11.

Tabel 11. Kaupadega seotud API päringud.

Meetod	Aadress	Kirjeldus	Parameetrid
GET	/api/items	Tagastab kõik vastavas laos olevad kaubad.	warehouseId: Integer limit: Integer offset: Integer
POST	/api/items	Lisab uue kauba.	name: String unit: Object itemType: Object partner: Object code: String description: String purchasePrice: Number retailPrice: Number
PUT	/api/items	Muudab kauba andmeid.	name: String unit: Object

			partner: Object code: String description: String purchasePrice: Number retailPrice: Number quantity: Number oldWarehouse: Object
DELETE	/api/items	Kustutab kauba laost.	itemId: Integer warehouseId: Integer
GET	/api/items/search	Otsib kõik parameetritele vastavad kaubad.	warehouseId: Integer name: String code: String itemTypeId: Integer limit: Integer offset: Integer

3.5.2 Autentimine

Rakenduses on esialgu implementeeritud ainult Google'i konto kaudu autentimine, kuna töö eesmärgiks on rõhku panna rakenduse ärinõuete implementeerimisele ning põhjaliku kasutajanime ja salasõnaga autentimissüsteemi implementeerimine ei mahuks antud töö skoopi.

Autentimine on realiseeritud kasutades Node.js-ile mõeldud Passport teeki. Sellel teegil on erinevad „starteegiad", mida saab autentimiseks kasutada. Antud töös on kasutatud Google Token ja JSON Web Token (JWT) strateegiaid.

Sisselogimisel avaneb kasutajale aken, kuhu ta saab sisestada oma Google'i konto andmed. Andmete edukal sisestamisel saadetakse kasutaja ligipääsutoken ja profiili andmed arendaja poolt Google Developer Console'is seadistatud *callback* URLile, mis kasutab Google Token strateegia vahefunktsiooni, et ligipääsutokeni õigsust kinnitada. Kui token on õige, siis leitakse andmebaasist vastava Google ID-ga kasutaja (kui kasutajat pole, siis luuakse uus kasutaja), signeeritakse kasutaja andmed JWT salasõnaga ja tagastatakse kliendile signeeritud token koos kasutaja andmetega. Järgnevate päringute autentimiseks on vaja saadud token lisada päringu „*Authentication*" päisesse kujul

„*Bearer* {token}“. Autentimist vajavatele päringutele on serveri poolel JWT strateegia vahefunktsioon, mis kontrollib päisesse lisatud tokeni õigsust. Kui token on õige, viiakse päring lõpuni, kui vale, siis tagastatakse „401 *Unauthorized*“ HTTP staatus.

3.5.3 Andmete valideerimine

Kliendi poolt saadetud andmete valideerimiseks kasutan Joi teeki. See võimaldab defineerida JavaScripti objekti skeemi ja sisseantavat objekti skeemi vastu valideerida. Implementeerisin päringute sisu valideerimise POST ja PUT meetodeid vastu võtvatele aadressidele vahefunktsioonidena. Kui valideerimine on edukas, siis viiakse päring lõpuni, vastasel juhul päring katkestatakse ja tagastatakse kliendile „400 *Bad Request*“ HTTP staatus.

Allolevas koodinäites (Joonis 14) on toodud kulu muutmise päringu valideerimisskeem, mis määrab ära, et päringus peavad sisalduma numbrina kulu ID ja õigesti vormistatud kuupäev, kuid kulu kirjeldus on mittekohustuslik string.

```
editExpense: (req, res, next) => {
        const schema = Joi.object().keys({
            id: Joi.number().required(),
            date: Joi.date().required(),
            description: Joi.string().allow(null, '')
        });
        validate(req.body, schema, res, next);
}
```

Joonis 14. Kulu muutmise valideerimisskeem.

3.5.4 Müügiarve PDF faili koostamine

Müügiarvete PDF formaadis faile saab alla laadida saates autoriseerimispäisega päringu API aadressile /api/sales?saleId={müügiarve ID}. PDF faili koostamiseks kasutan Pug ja Pdf-puppeteer teeke.

Pug on teek, mis võimaldab luua muutujaid sisaldavaid HTML malle ja JavaScripti kaudu muutujatele väärtused anda. Pdf-puppeteer on teek, mis konverteerib HTML lehe PDF failiks. See kasutab taustal Puppeteer teeki, mis võimaldab „peata“ ehk käsurealt juhitavat Chrome’i kasutada JavaScripti kaudu.

Hetkel on PDF arve allalaadimine realiseeritud nii, et PDF fail pole kuhugi salvestatud, vaid iga päringu korral leiab server andmebaasist vastava müügiarve andmed, kompileerib Pug malli, käivitab Puppeteeri kaudu Chrome'i ja seejärel koostab PDF faili (Joonis 15). Kuna iga arve allalaadimise korral Chrome'i käivitamine ja HTMLi PDFiks konverteerimine nõuab suhteliselt palju aega (üle 2 sekundi) ja protsessori jõudlust, siis oleks tulevikus mõttekas PDF arve koostada vaid müügiarve loomisel ja see edasiseks allalaadimiseks salvestada.

```
const sale = await models.Sale.findOne({ ... });
const invoiceTemplate = pug.compileFile('./templates/saleInvoice.pug');

htmlToPDF(invoiceTemplate(sale), pdf => {
    res.setHeader('Content-Type', 'application/pdf');
    res.send(pdf);
});
```

Joonis 15. Müügiarve PDF faili koostamise koodinäide.

4 Valideerimine

Valideerin rakenduse implementatsiooni peatükis 2.2 toodud nõuetele vastavuse kontrollimise teel.

Tabel 12. Implementatsiooni nõuetele vastavus.

Identifikaator	Nõue	Implementeeritud
fn1_1	Kasutaja peab saama lisada uut ladu.	Jah
fn1_2	Kasutaja peab saama ladude nimekirja vaadata.	Jah
fn1_3	Kasutaja peab saama ladu kustutada.	Jah
fn1_4	Kasutaja peab saama lattu kaupa lisada.	Jah
fn1_5	Kasutaja peab saama lisada uut ühikut.	Jah
fn1_6	Kasutaja peab saama laokauba detaile vaadata.	Jah
fn1_7	Kasutaja peab saama laokaupa kustutada.	Jah
fn1_8	Kasutaja peab saama laokaupa muuta.	Jah
fn1_9	Kasutaja peab saama ühikut muuta.	Ei
fn1_10	Kasutaja peab saama ühikut kustutada.	Ei
fn1_11	Kasutaja peab saama ladu muuta.	Ei
fn2_2	Kasutaja peab saama konto sätteid muuta.	Jah
fn2_3	Kasutaja peab saama sisse logida salasõna ja parooliga või Google'i/Facebooki kontoga	Osaliselt. Sisse saab logida ainult Google'i kontoga.
fn2_4	Kasutaja peab saama profiili andmeid muuta.	Jah
fn2_5	Kasutaja peab saama pangakontosid lisada.	Jah
fn2_6	Kasutaja peab saama endaga seotud pangakontosid kustutada.	Jah
fn3_1	Kasutaja peab saama ostuarvet luua.	Jah
fn3_2	Kasutaja peab saama ostuarvete nimekirja vaadata.	Jah
fn3_3	Kasutaja peab saama ostuarvesse kaupa lisada.	Jah

fn3_4	Kasutaja peab saama ostuarve detaile vaadata.	Jah
fn3_5	Kasutaja peab saama ostuarvet kustutada.	Jah
fn3_6	Kasutaja peab saama ostuarveid kõigi parameetrite järgi otsida.	Osaliselt. Otsida saab vaid tarnija nime ja arve numbri järgi.
fn3_7	Kasutaja peab saama ostuarvet muuta.	Ei
fn4_2	Kasutaja peab saama müügiarvete nimekirja vaadata.	Jah
fn4_3	Kasutaja peab saama müügiarvet PDF failina alla laadida.	Jah
fn4_4	Kasutaja peab saama müügiarvele kaupa lisada.	Jah
fn4_5	Kasutaja peab saama müügiarve detaile vaadata.	Jah
fn4_6	Kasutaja peab saama müügiarvet kustutada.	Jah
fn4_7	Kasutaja peab saama müügiarvet muuta.	Ei
fn5_1	Kasutaja peab saama kulude nimekirja vaadata.	Jah
fn5_2	Kasutaja peab saama kulu detaile vaadata.	Jah
fn5_3	Kasutaja peab saama kulu kustutada.	Jah
fn5_4	Kasutaja peab saama ostuarvet kuludesse lisada.	Jah
fn6_1	Kasutaja peab saama müügiarvega seotud laekumisi lisada.	Jah
fn6_2	Kasutaja peab saama laekumiste nimekirja vaadata.	Jah
fn6_3	Kasutaja peab saama laekumise detaile vaadata.	Jah
fn6_4	Kasutaja peab saama laekumist kustutada.	Jah
fn6_5	Kasutaja peab saama laekumist muuta.	Jah
fn7_1	Kasutaja peab saama tulude nimekirja vaadata.	Jah
fn7_2	Kasutaja peab saama tulu detaile vaadata.	Jah
fn7_3	Kasutaja peab saama tulu kustutada.	Jah
fn8_1	Kasutaja peab saama partnerit lisada.	Jah

fn8_2	Kasutaja peab saama partnerite nimekirja vaadata.	Jah
fn8_3	Kasutaja peab saama partnerit kustutada.	Jah
fn8_4	Kasutaja peab saama partnerit muuta.	Jah
fn9_1	Kasutaja peab saama näha kulude ja tulude statistikat.	Jah
mfn_1	Rakenduse kasutajaliides on eestikeelne.	Jah
mfn_2	Rakenduse lähtekood on ingliskeelne.	Jah
mfn_3	Põhivaatest saab kõiki operatsioone teha vähem kui 3 klikiga (v.a klikid vormide täitmiseks).	Jah
mfn_4	Rakendus on kasutatav Google Chrome, Mozilla Firefox ja Microsoft Edge brauserites.	Jah
mfn_5	Kõik kasutaja poolt saadetud vormide andmed peavad olema serveri poolt valideeritud.	Jah

5 Võimalikud edasiarendused

Kuna loodud rakenduse arendamise aeg oli piiratud, siis jõudsin implementeerida ainult rakenduse baasfunktsionaalsuse, et põhiline äriloogika töötaks ja rakendust oleks võimalik ka praktikas kasutada. Kasutajakogemuse parandamiseks saab rakendusele mitmeid edasiarendusi teha.

5.1 Registreerimise variandid

Hetkel on realiseeritud ainult Google'i kontoga registreerimine. See on küll hea variant kiireks registreerimiseks ja ei nõua uue kasutajanime ja parooli meeldejätmist, kuid ei ole hea kasutajatele, kellel näiteks Google'i kontot pole või ei soovi mingil põhjusel oma kontot rakendusega siduda. Seetõttu peaks tulevikus implementeerima ka kasutajanime ja salasõnaga registreerimise ja lisada võimaluse ka muude sotsiaalplatvormidega registreerimiseks.

5.2 Kasutajaliidese mugavdamine

Hetkel võimaldab kasutajaliides küll enamikke rakendusele seatud nõudeid täita, kuid mõned tegevused võiksid siiski kasutajasõbralikumad olla.

Näiteks ei saa arve loomisel arvesse lisatud kaupu muuta, vaid peab kauba ära kustutama ja uuesti sisestama. Lisaks peab kauba lisamise vormil *scroll*ima, et lisamise nupuni jõuda. Arvete loomise vormi peaks ümber disainima nii, et arvesse lisatud kaubad on esitatud tabelina ja kauba lisamise vorm on näiteks tabelisse reana sisse ehitatud. Nii on arvesse lisatavaid kaupasid kergem hallata.

Lisaks peaks rakenduse ümber disainima nii, et kasutajaliides oleks ekraani suurusega kohalduv ehk *responsive*. Nii saaks rakendust kasutada ka telefonis, kui on näiteks kiiresti vaja mõni kulu või tulu lisada. Kogu rakenduse funktsionaalsust pole siiski mõtet kohalduvaks teha, kuna rakenduses on vaja esitada palju vorme ja andmeid, mille kõigi kohandamine läheks keeruliseks. Rakenduse kogufunktsionaalsuse kasutamiseks telefonis oleks mõttekas luua eraldi mobiilirakendus.

Hetkel on rakendusel igal tabelil üks otsingu väli, millega saab andmeid ainult teatud parameetrite järgi otsida. Sellele lisaks oleks kasulik teha ka põhjalikum otsingusüsteem, millega oleks võimalik valida, milliste parameetrite järgi andmeid otsida.

5.3 Tootmise moodul

Tulevikus peaks rakendusele lisama ka tootmise mooduli, millega saab laosolevast materjalist koostada uue kauba ja mis arvutaks materjalikulu põhjal ka toote omahinna ja võimaldaks lisada juurdehindlust. Selline funktsionaalsus tuleks kasuks ettevõtetele, kes tegelevad tootmisega - näiteks riideid tootev ettevõte saaks laosolevast niidist ja kangast toota särke ja kulunud materjali põhjal automaatselt arvutada särgi omahinna.

5.4 Refaktoreerimine ja testimine

Kuna rakendust on plaanis ka pärast töö valmimist edasi arendada ja tulevikus võib rakendust arendama hakata ka mõni teine arendaja, siis peaks rakenduse lähtekoodi refaktoreerima ja muutma selle kergemini hallatavamaks. Näiteks hetkel on rakenduse serveripoolsed päringuid vastuvõtvad funktsioonid üsna pikad ja raskesti arusaadavad. Need võiks jagada eraldi osadeks ja lisada kommentaarid, et ka teistele arendajatele oleks kood lihtsasti arusaadav. Samuti võiks kliendi- ja serveripoolse koodi katta ühiktestidega, et vigu võimalikult varakult tuvastada.

6 Kokkuvõte

Antud töö eesmärgiks oli luua veebipõhine laohaldusrakendus, mis võimaldab moodustada arveid ning kuludest ja tuludest ülevaade saada. Loodud rakendus on eelkõige mõeldud kassapõhise raamatupidamisarvestusega FIEdele, kellel puudub vajadus olemasolevate laotarkvarade funktsionaalsuse järgi, mis on suunatud pigem tekkepõhise raamatupidamisega ettevõtetele.

Töö käigus analüüsisin olemasolevaid rakendusi ja suhtlesin rakenduse potentsiaalse kasutajaga. Analüüsi ja kasutaja tagasiside põhjal prototüüpisin ja implementeerisin *full-stack* laohaldusrakenduse. Rakenduse prototüüpimiseks kasutasin Figmat ja implementeerimiseks Reacti, Node.js-i koos Express raamistikuga ja PostgreSQLi. Olin nende tehnoloogiatega küll tuttav, kuid töö tegemise käigus õppisin uusi viise nende kasutamiseks.

Seatud eesmärk sai täidetud ning töö tulemusena valmis laohaldusrakendus, mis võimaldab laoseisu üle arvet pidada, arveid koostada ning kuludest ja tuludest ülevaade saada.

Rakendust on plaanis ka edasi arendada, kuna kõik täiendused ei mahtunud antud töö skoopi. Näiteks võib rakendusele lisada peale Google'i konto veel registreerimisviise, muuta kasutajaliidest mugavamaks ja kohaldada see ka mobiilivaatele, lisada uut funktsionaalsust ja muuta lähtekood paremini hallatavamaks.

Valminud rakendus on ligipääsetav aadressil https://warehop.tk ja lähtekoodi repositoorium on saadaval aadressil https://gitlab.cs.ttu.ee/stjuks/iapb.

Kasutatud kirjandus

[1] „Zoho Inventory,“ [Võrgumaterjal]. Available: https://www.zoho.com/inventory/. [Kasutatud 13 mai 2019].

[2] „TradeGecko,“ [Võrgumaterjal]. Available: https://www.tradegecko.com/. [Kasutatud 13 mai 2019].

[3] „Wave Financial,“ [Võrgumaterjal]. Available: https://www.waveapps.com/. [Kasutatud 13 mai 2019].

[4] „Raamatupidamise seadus,“ 20 november 2002. [Võrgumaterjal]. Available: https://www.riigiteataja.ee/akt/125052012016. [Kasutatud 20 mai 2019].

[5] „UX Tricks,“ 29 november 2018. [Võrgumaterjal]. Available: https://uxtricks.design/blogs/ux-design/best-ui-design-tools/. [Kasutatud 13 mai 2019].

[6] „React (JavaScript library),“ [Võrgumaterjal]. Available: https://en.wikipedia.org/wiki/React_(JavaScript_library). [Kasutatud 13 mai 2019].

[7] „Node.js,“ [Võrgumaterjal]. Available: https://en.wikipedia.org/wiki/Node.js. [Kasutatud 13 mai 2019].

[8] P. Patel, „What exactly is Node.js?,“ 18 aprill 2018. [Võrgumaterjal]. Available: https://medium.freecodecamp.org/what-exactly-is-node-js-ae36e97449f5. [Kasutatud 13 mai 2019].

[9] „Express.js,“ [Võrgumaterjal]. Available: https://en.wikipedia.org/wiki/Express.js. [Kasutatud 13 mai 2019].

[10] „Node.js - Express Framework,“ [Võrgumaterjal]. Available: https://www.tutorialspoint.com/nodejs/nodejs_express_framework.htm. [Kasutatud 13 mai 2019].

[11] „Stack Overflow Developer Survey 2019,“ 2019. [Võrgumaterjal]. Available: https://insights.stackoverflow.com/survey/2019#technology. [Kasutatud 13 mai 2019].

[12] „PostgreSQL,“ [Võrgumaterjal]. Available: https://en.wikipedia.org/wiki/PostgreSQL. [Kasutatud 13 mai 2019].

[13] S. Bhatia, „PostgreSQL vs. MySQL: [2019] Everything You Need to Know,“ 31 01 2019. [Võrgumaterjal]. Available: https://hackr.io/blog/postgresql-vs-mysql#PostgreSQL_vs_MySQL_Key_Differences. [Kasutatud 13 mai 2019].

[14] „PostgreSQL vs. MySQL,“ [Võrgumaterjal]. Available: http://www.postgresqltutorial.com/postgresql-vs-mysql/. [Kasutatud 13 mai 2019].

[15] „Sequelize,“ [Võrgumaterjal]. Available: http://docs.sequelizejs.com/. [Kasutatud 16 mai 2019].

[16] A. Mittal, „Getting Started with Sequelize for Nodejs Applications,“ [Võrgumaterjal]. Available: https://hackernoon.com/getting-started-with-sequelize-for-nodejs-applications-2854c58ffb8c. [Kasutatud 16 mai 2019].

[17] „Git,“ [Võrgumaterjal]. Available: https://git-scm.com/. [Kasutatud 13 mai 2019].

[18] „Google Developer Console,“ [Võrgumaterjal]. Available: https://console.developers.google.com/. [Kasutatud 16 mai 2019].

[19] „Introduction to JSON Web Tokens,“ [Võrgumaterjal]. Available: https://jwt.io/introduction/. [Kasutatud 16 mai 2019].

[20] „Create React App,“ [Võrgumaterjal]. Available: https://facebook.github.io/create-react-app/. [Kasutatud 13 mai 2019].

[21] „MobX,“ [Võrgumaterjal]. Available: https://mobx.js.org/index.html. [Kasutatud 17 mai 2019].

[22] „Redux,“ [Võrgumaterjal]. Available: https://redux.js.org/. [Kasutatud 17 mai 2019].

[23] R. Wieruch, „Redux or MobX: An attempt to dissolve the Confusion,“ 28 03 2017. [Võrgumaterjal]. Available: https://www.robinwieruch.de/redux-mobx-confusion/. [Kasutatud 17 mai 2019].