TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Piia-Mai Orav 184974IADB

Veebirakenduse arendamine mänguasjavahetuse keskkonna näitel

Bakalaureusetöö

Juhendaja: Meelis Antoi
MSc

Tallinn 2022

Autorideklaratsioon

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

Autor: Piia-Mai Orav

16.05.2022

Annotatsioon

Antud lõputöö eesmärgiks oli arendada veebirakendus, mille kaudu on kasutajatel võimalik üksteisega vahetada mänguasju.

Veebirakenduse arendamiseks tehti kõigepealt põhjalik nõuete analüüs. Rakenduse nõudeid vaadeldi erinevatelt abstraktsioonitasemetelt, liikudes üldisest süsteemi kontekstist spetsiifiliste nõueteni. Nõuete analüüsi käigus tuli teha ka valikuid, milliseid funktsionaalsusi minimaalse töötava toote jaoks välja arendada ja milliseid jätta esialgu välja. Lisaks on kirjeldatud arenduse jaoks kasutatud tehnoloogiad ja tööriistu ning antud kirjeldus loodud ees- ja tagarakenduset.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 41 leheküljel, 6 peatükki, 18 joonist, 2 tabelit.

Abstract

Development of a Web Application on the Example of a Toy Swapping Portal

The purpose of this paper is to demonstrate the development of a web application where users can swap toys with each other.

Firstly, a thorough analysis was conducted, in order to figure out the requirements for the system. The application was observed from differnt levels of abstractions, moving from the most generic system context to the most specific requirements. During the analysis process, some decisions had to be made as to which features to implement in the minimum viable product and which ones to leave for the future. After the analysis section, an overview of the used technologies and descriptions of created frontend and backend applications are given.

The thesis is in Estonian and contains 41 pages of text, 6 chapters, 18 figures, 2 tables.

Lühendite ja mõistete sõnastik

Aktor	Süsteemiga suhtlev inimene või inimene kindlas rollis või teine süsteem.
Andmebaasihaldur	DBMS ehk *Database Management System*.
API	*Application Programming Interface*, liides, mille kaudu kaks rakendust üksteisega suhelda saavad.
AWS	*Amazon Web Services*, Amazoni pilveteenuste platvorm.
CORS	*Cross-Origin Resource Sharing*, HTTP-päisetel põhinev mehhanism, mis võimaldab serveri poolel defineerida, milliste päritoludega päringuid veebilehitsejas lubada.
CRUD	*Create, Read, Update, Delete* – akronüüm, mis tähistab viise, kuidas andmeid käidelda (loo, loe, uuenda, kustuta).
CSS	*Cascaded Style Sheet,* standardiseeritud keel HTML'i stiliseerimiseks.
EC2	*Elastic Compute Cloud*, AWS'i server.
Featuur	*Feature*, terviklik funktsionaalsus, mida on võimalik ühe korraga teostada.
HTML	*Hypertext Markup Language*, standardiseeritud keel veebilehtede loomiseks.
HTTP	*Hypertext Transfer Protocol*, rakenduskihi protokoll tekstiliste andmete edastamiseks üle võrgu.
JWT	*Json Web Token*, allkirjastatud JSON kujul andmeblokk, mis koosneb päisest ja kehast. Seda saab lugeda igaüks, kuid muuta saab seda ainult siis, kui on teada allkirja võti.
LINQ	*Language-Integrated Query*, kirjeldab kogumikke tehnoloogiaid, mis võimaldavad C# programmeerimiskeeles teha andmepäringuid.
MVP	*Minimum Viable Product*, minimaalne töötav toode. Algne toode, millel on minimaalne funktsionaalsus ja mida saab testimiseks kasutajatele anda, et seda saadud tagasiside põhjal edasi arendada.
OTP	*One Time Pin or Password,* ühekordne PIN-kood või salasõna, mille eluiga on piiratud ja mida saab kasutada ainult ühe korra. Kasutatakse kasutaja verifitseerimiseks.

Pollimine	Tegevus, mille käigus korratakse ühte päringut mitu korda kuni saadakse vastus, või kuni korduste arv lõpeb.
S3	*Simple Storage Service*, AWS'i pilvehoidla.
Sidusrühm	*Stakeholder*, inimene või grupp inimesi, kellel on otsene huvi loodava rakenduse vastu. Nendeks võivad olla näiteks tulevased kasutajad, rakenduse tellijad kui ka tootejuhid, tarkvara arendajad jne.
SPA	*Single Page Application*, veebirakendus, mis on üles ehitatud ühele leheküljele.
SQL	*Standardized Query Language*, standardiseeritud päringukeel, mida kasutatakse enamasti relatsionniliste andmebaaside pihta päringute tegemiseks.
WebSocket	Arvutite suhtlusprotokoll, mis võimaldab kahesuunalist suhtlust ühe TCP ühenduse kaudu.

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Lapsevanemad seisavad tihti probleemi ees, et lastele on ostetud mänguasju, mis on olnud kasutuses kuu või paar ning edaspidi võtavad ainult ruumi. Samal ajal soovitakse lapse arengu toetamiseks osta aina uusi lelusid, mis võivad uuena ostes olla suhteliselt kallid. Kindlasti leidub neid lapsevanemaid, kes erinevates foorumites, platvormidel ja keskkondades neid rahaks teevad, ning vastavalt sellele leidub ka neid, kes samadest kohtadest neid otsivad ja ostavad. Käesoleva töö autori poole pöördus üks taoline lapsevanem, kes kurtis, et selline tegevus on tüütu ning kasu võiks olla keskkonnast, mis just laste mänguasjade vahetusele spetsialiseerub ning kus eesmärgiks ei pea tingimata olema maksimaalse kasumi teenimine, vaid säästev eluviis. Olles inspireeritud hästi vastuvõetud raamatuvahetus.ee platvormist, käis ta välja idee luua keskkond, kus lelusid saab vahetada sümboolse tasu eest ning kindlate reeglite alusel. Käesolev töö seabki eesmärgiks luua veebikeskkonna prototüüp, mille kaudu on võimalik seisma jäänud mänguasju müümise asemel vahetada.

Käesolev töö proovib peegeldada kogu arenduse elutsüklit, hõlmates koostöös kliendiga nõuete väljaselgitamist ja kogutud nõuete realiseerimist päris teenuseks. Etteruttavalt võib öelda, et loodav rakendus saab koosnema eesrakendusest, tagarakendusest ning andmebaasist ning on juurutatud pilveteenusesse.

Kuna idee on palju suurem kui käesoleva lõputöö maht, siis esialgu luuakse MVP ehk minimaalne töötav toode. Esialgu ei ole plaanis seda laiemale publikule kasutamiseks avada, vaid väikesele grupile testkasutajatele, kes teenust proovivad ja tagasisidet annavad. Kui tuleb välja, et pakutav teenus on tõepoolest kasulik, siis on soov ja tahe seda edasi arendada.

Käesoleva kirjatöö struktuur on järgmine. Esimeses osas kirjeldatakse metoodikat, mida töö realiseerimiseks kasutatakse. Teises osas kirjutatakse lahti loodavale rakendusele seatud ootused ning detailsem kontseptsioon. Kolmandas osas rakendatakse valitud

metoodikat analüüsiks ja nõuete väljaselgitamiseks. Neljandas osas valitakse tehnoloogiad, mille abil rakendus valmis ehitada, ning vaaldetakse lähemalt loodud rakenduse osasid.

2 Metoodika

Enne tarkvara arendamist on vaja mõista, millised on nõuded loodavale tarkvarale. Tarkvara nõuete kogumise jaoks on palju erinevaid meetodeid. Selles töös on otsitud juhiseid Klaus Pohl'i raamatust *Requirements Engineering* [15] , mis pakub põhjalikku nõuete arenduse raamistikku tarkvara nõuete väljaselgitamiseks, mistõttu tugineb see peatükk suuresti ühele konkreetsele allikale. Kuigi käesoleva töö eesmärk ei ole produtseerida võimalikult detailne ja kõikehõlmav dokumentatsioon, proovitakse selle raamatu abil luua piisav dokumentatsioon, et toote arendamisega pihta hakata.

Miks on nõuete arendus oluline ja mille jaoks on seda vaja? Pohl toob raamatus välja terve rida põhjuseid, millest mõned on väärt ka siin mainimist [15, lk 6-7]:

- Projekti läbikukkumiste üheks suurimaks põhjuseks on vale nõuete arendus.
- Nõuete arenduse defektide tagantjärele silumine hilisemates tarkvaraarenduse faasides toob kaasa suuremad kulutused.
- Süsteemi funktsioonide arvu kasv, tihedam integratsioon teiste süsteemidega ja diferentseeritud kasutus.

Mõned olulisemad mõisted, mida loetud raamatust kaasa võtta on süsteem ja selle kontekst, stsenaariumid ja nõuded. Järgnevates alapeatükkides süvenetakse neisse veidi sügavamalt.

2.1 Süsteem ja selle kontekst

Raamatus viidatakse süsteemi kontekstile, täheldades, et igat süsteemi ümbritseb mingi kontekst ning seda konteksti tuleb nõuete arenduses alati silmas pidada. Kusjuures selle konteksti moodustavad nii tehnoloogilised kui ka mitte-tehnoloogilised süsteemid,

inimesed, äriprotsessid, seadused, loodusseadused ja muu sarnane [15, lk 64]. Silmas tuleb pidada, et mitte kogu kontekst pole arenduse jaoks oluline ning tuleb vahet teha olulisel ja mitte-olulisel. Selle piiri aitab tõmmata “konteksti piir”, millest välja jäävad aspektid ei ole arenduse jaoks relevantsed. Eksisteerib ka piir süsteemi enda ja selle konteksti vahel – “süsteemi piir”, mis eraldab üksteisest konteksti need osad, mida saab arenduse käigus muuta, ja need osad, mida arenduse käigus muuta ei saa [15, lk 65]. Süsteemi konteksti aitab mudelina väljendada konteksti diagramm (loe lähemalt konteksti diagrammi kohta leheküljelt https://www.edrawmax.com/context-diagram/).

2.2 Stsenaariumid

Stsenaariumid dokumenteerivad tegevuste järjestust ja illustreerivad positiivsete ja negatiivsete näidete abil sidusrühmade eesmärke [15, lk 139]. Kuigi stsenaariumid on abstraktsemad kui päriselu ja detailsemad kui kontseptuaalsed mudelid, võib ise otsustada, millist abstraktsioonitaset kasutada. Stsenaariumid on hea tööriist dokumenteerima informatsiooni süsteemi konteksti kohta ja seega toimivad ühenduslülina konteksti aspektide ja konkreetsete nõuete vahel [15, lk 143]. Konteksti informatsiooniks on nii aktorid, rollid, eesmärgid kui ka eel- ja järeltingimused, nõutavad ressursid ja täitmiskeskkond [15, lk 144]. Nõuete arenduse käigus loodavad stsenaariumid on enamasti kasutusjuhtumite stsenaariumid, millega täpsustakse kasutajate konkreetseid tegevusi [15, lk 145-146].

Kasutusjuhtumeid saab kasutada põhi-, alternatiiv- ja erindistsenaariumite grupeerimisesks, mis seostuvad spetsiifiliste eesmärkidega [15, lk 148]. Seega on kasutusjuhtum tegude järjestuste spetsifikatsioon, mis sisaldab endas nii alternatiivseid kui ka erandlikke tegude järjestusi, mida süsteem, alamsüsteem või klass saab ellu viia, suheldes väljaspool asuvate objektidega ja pakkudes mingisugust väärtust [15, lk 164]. Kasutusjuhtum koosneb konteksti informatsioonist, põhistsenaariumist, alternatiivsetest stenaariumitest ja erindistsenaariumitest. Konteksti informatsioon sisaldab endas sidusrühmade eesmärke, mis antud kasutusjuhtumiga seostuvad, ning kasutusjuhtumi täideviimise eel- ja järeltingimusi [15, lk 164]. Kasutusjuhtumite dokumenteerimiseks

võib kasutada nii naturaalset keelt kui ka mudeleid, kuid Pohl soovitab kasutada stsenaariumite dokumenteerimiseks neid mõlemaid [15, lk 193].

2.3 Nõuded

Nõuded defineerivad süsteemi poolt realiseeritavaid funktsionaalsuseid ja featuure. Need on veel detailsemad kui stsenaariumid. Koos stsenaariumitega moodustavad nõuded süsteemi realiseerimiseks aluse. Nõuded loovad aluse nii süsteemi arhitektuurile kui ka testjuhtumitele [15, lk 214]. Nõudeid tuletatakse põhiliselt stsenaariumitest, kuid mingil määral ka teistest allikatest [15, lk 218]. Mittefunktsionaalseid nõudeid saab näiteks tuletada prototüübi põhjal. Nõudeid saab valideerida, vaadates, kas see nõue viib dokumenteeritud eesmärkide ja stsenaariumite täitmiseni või mitte [15, lk 219].

Käsitletavas raamatus kirjeldatakse kolme tüüpi nõudeid: funktsionaalsed nõuded, kvaliteedinõuded ja piirangud [15, lk 17-24]. Viimased kaks on paremini tuntud ühise nimetaja “mittefunktsionaalsed nõuded” all, kuid autor soovitab tungivalt seda terminit vältida, kuna tema sõnul on tegemist enamasti lihtsalt alamääratletud, rafineerimata nõuetega, millega kaasneb risk, et erinevad sidusrühmad mõistavad neid erinevalt.

Funktsionaalsed nõuded täpsustavad, millist funktsionaalsust süsteem kasutajale pakub. Selliste nõuete kirjeldamiseks kasutatakse kolme perspektiivi: andmete perspektiiv, funktsionaalne perspektiiv ja käitumuslik perspektiiv. Enamasti on tegemist lahendustele orienteeritud nõuetega, kuna nad on defineeritud viisil, mis toetavad süsteemi realiseerimist. Kvaliteedinõuded defineerivad süsteemi omaduste kvaliteeti, nagu näiteks jõudlus, usaldusväärsus ja stabiilsus. Seda tüüpi nõuded suunavad tihti arhitektuurilisi valikuid. Piirangud piiravad enamasti tarkvaraarenduse protsessi või arendatava süsteemi omadusi. Piiranguks võib olla näiteks aja ja ressursi jaotus, keskkond ja/või kontekst, milles süsteem hakkab toimima. Piirangu puhul on oluline märkida, et seda ei saa tüüpiliselt teiste sidusrühmade poolt mõjutada.

Nagu stsenaariume, saab ka nõudeid dokumenteerida nii mudelite kui ka naturaalses keeles. Naturaalses keeles dokumentatsioonis on erinevad perspektiivid omavahel väga seotud. Seevastu mudelite keel annab võimaluse vaadelda funktsionnalseid nõudeid

erinevatest perspektiividest: andmed, funktsionaalsus ja käitumine. Andmete perspektiivi saab kirjeldada olem-suhte diagrammi abil, funktsionaalsest perspektiivi aitavad visualiseerida andmevoo diagrammid ning käitumise perspektiivist aitab näha olekumasina diagramm [15, lk 221-223].

2.4 Rakendatav metoodika

Eelnevates alapeatükkides anti ülevaade nõuete arendamise raamistikust, mis annab tööriistad nõuete kogumiseks ja dokumenteerimiseks. Antud töös proovitaksegi nõuete väljaselgitamiseks rakendada kirjeldatud tegevusi. Kuna nõuete väljaselgitamine on pidev protsess, korraldatakse kliendi soovide väljaselgitamiseks vajaduspõhiselt kokkusaamisi ja arutelusid. Tuleb meeles pidada asjaolu, et nõuded ei ole staatilised ning võivad ajas väga palju muutuda.

Et välja selgitada nõuded süsteemi realiseerimiseks, rakendatakse meetodeid, mida eelmises peatükis tutvustati. Kõigepealt defineeritakse süsteemi kontekst, seejärel kasutusjuhtumid ja viimaks nõuded. Peale kasutusjuhtumite väljaselgitamist paigutatakse need investeeringutasuvuse tabelisse, et välja selgitada nõuded, mida MVP jaoks implementeerida, ja nõuded, mis jäävad esialgu tagalogisse.

3 Rakenduse kirjeldus

Loodava rakenduse kirjeldus selgitati välja koostöös kliendiga. Viidi läbi mitmeid intervjuusid ja mõtteharjutusi, mile käigus mõeldi koos läbi võimalikud protsessid ja kitsaskohad. Koos loodi ka prototüüp (Lisa 3), mis aitab visualiseerida ootusi loodavale keskkonnale ja hiljem valideerida dokumenteeritud nõuete vastavust ootustele. Käesolevas peatükis proovitakse rakenduse esialgne kirjeldus ja kontseptsioon naturaalses keeles lugejani tuua. Hilisemad peatükid tegelevad nende mõtete organiseerimise ja dokumenteerimisega, ning lõpuks ka realiseerimisega.

3.1 Kontseptsioon

Tegemist on veebikeskkonnaga, kuhu saavad külastajad luua endale kasutajakonto ning vahetada teiste kasutajatega lelusid. Kuivõrd tegemist on rahatu süsteemiga, saab lelusid osta vaid keskkonnas kasutusel olevate punktide eest. Esialgu saab punkte konto loomise eest, seejärel lelude müügist. Edasiarendusena võib seda võimalust laiendada reaalse punktimüügi kaudu. Nagu öeldud, jõuavad lelud pakkumisse läbi kasutajate endi, kes saavad neid portaali sisestada. Pakutavale leule tuleb määrata nimetus, kirjeldus, hind punktides ja ka muid asjakohaseid parameetreid, mille järgi on võimalik neid otsingus filtreerida. Kasutajad saavad kategoriseerimise põhjal otsida endale meelepäraseid lelusid, lisada neid ostukorvi ja piisava punktisaldo olemasolul esitada teisele kasutajale tellimuse. Pärast edukat tellimuse esitamist broneeritakse lelu väärtuses punktid kasutaja punktisaldost ning lelu omanik saab tellimuse kohta teavituse. Seejärel saab omanik otsustada, kas ta soovib tellimust täita või mitte. Kui ta ei soovi jätkata, saab ta tellimuse tühistada. Kui ta aga soovib jätkata, tuleb tal tellijaga kokku leppida üleandmise tingimused. Kokkuleppele jõudmiseks võib kasutada portaalisisest suhtlust. Kui kaup on pandud teele, peab ta portaalis seda kinnitama. Samamoodi peab kauba edukat kättesaamist kinnitama tellija, ning samamoodi saab

tellija tellimust tühistada, kuid vaid selle hetkeni, kuni kaupa pole veel teele pandud. Kui kättesaamine on kinnitatud, kaovad tellija kontolt broneeritud punktid ning omaniku punktisaldo täieneb lelu väärtuseks määratud punktide koguse võrra.

Esialgu peavad kasutajad transpordi osas ise omavahel kokku leppima ja seda ise ka korraldama. Edasiarendusena võib võimadada kasutajatele mõne postineenuse pakkuja valimist, pidades silmas just pakiautomaate. Et hõlbustada suhtlust tehingu osapoolte vahel, on kasutajatel võimalik üksteisega suhelda sisseehitatud kiirsuhtlusrakenduse abil. Viimase sammuna vahetuse protsessis saab nii tellija kui ka omanik jätta teisele osapoolele tagasiside ja anda talle viie tärni süsteemis hinde. Viimase põhjal kujuneb kasutaja skoor, mis on tema profiili juures nähtav, ning ka saadud tagasiside kuvatakse kasutaja profiilil.

Kuigi kasutajad saavad virtuaalraha eest pakkuda kaupu, ei saa nad teenitud tulu päris rahas välja võtta. Kui keegi tellib pakutava mänguasja, saab omanik oma kontole punkte. Neid punkte saab ta omakorda kasutada teiste kasutajate pakutavate lelude tellimiseks. Sihtgrupiks ei ole inimesed, kes soovivad teenida reaalset müügitulu. Sihtgrupiks on inimesed, kes pooldavad rohelist mõtteviisi ning soovivad pakkuda oma lapsele vaheldust, suuri kulutusi tegemata.

3.2 Edasiarendused

Eelnevalt kirjeldatud kontseptsioon on piiratud antud töö skoopi. Kui esialgne toode võetakse testgrupi poolt hästi vastu, on plaanis teha süsteemi järgnevad edasiarendused: kasutaja verifitseerimine, punktisüsteemi edasiarendus piletite näol ning punkti-ja piletimüük.

Keskkond peaks veenduma ja tagama, et loodavaid kontosid ei kasutata kuritahtlikel eesmärkidel. Esmane meede on nõuda konto registreerimisel kasutaja nõusolekut veebikeskkonna tingimustega, mis sätestavad osapoolte kohustused ja õigused, mis veebikeskkonnas tegutsemisega kaasnevad. Teine meede on nõuda konto loomisel kasutaja verifitseerimist. E-maili teel verifitseerimise käigus seotakse kindel e-mail loodava kontoga. Siiski on e-maili kontot avada võrdlemisi lihtne ning ei hoia ära

petturlust. Teine variant on SMS-i teel verifitseerimine – uusi telefoninumbreid on märksa tüütum avada. Kui kasutaja loob konto, peab ta jagama oma telefoninumbrit, mida hiljem saab vahetada ainult taotluse esitamise kaudu. Antud telefoninumbrile saadetakse OTP (ühekordne PIN kood), mille kasutaja peab konto avamiseks registreerumisankeeti sisestama. Selle tulemusena on kontoga seotud kindel telefoninumber. Edaspidi sisse logides ei ole e-maili või tekstisõnumi teel verifitseerimist vaja. Piisab e-maili ja salasõnaga sisse logimisest.

Punktisüsteemi kõrvale on plaanis luua ka piletisüsteem, mis reguleerib, kui mitu tellimust kasutaja mingi aja jooksul võib esitada. Kõige tavalisem pilet annaks kasutajale õiguse esitada ühe kuu jooksul ühe tellimuse. See pilet oleks tasuta ning uueneks automaatselt iga kuu. Kui oleks soov tihedamini tellimusi esitada, siis selle jaoks oleks vaja juba osta uus pilet. Lisaks piletitele saaks osta ka punkte. Valida saaks erinevate variantide vahel, näiteks kolmene pilet, viiene pilet ja piiramatu pilet. Piletid kehtiksid ostu sooritamise hetkest alates 30 päeva. Punkte ja pileteid saaks osta vajaduse põhiselt. Tehingu läbiviimiseks saaks kasutada pangamakselinki. Kui makse on sooritatud, laekuksid punktid ja/või piletid kasutaja kontole ning ta saaks neid koheselt kasutama hakata.

3.3 Konkureerivad lahendused

Kasutatud mänguasjade taaskasutus pole kuigi uus idee ning seetõttu leidub turul palju konkurente. Kuivõrd vähemalt esialgu suuri maailmavallutusplaane ei ole, siis konkurentideks peetakse vaid Eesti turul tegutsevaid lahendusi.

3.3.1 Mänguasjalaenutused

Veebis tegutsevaid mänguasjalaenutusi on Eesti turul päris mitu. Nii LaenuLelu OÜ (https://www.laenulelu.ee), MTÜ Lelukogu (https://lelukogu.ee) kui ka HoiaEnd OÜ (https://hoialelu.ee) tegelevad mänguasjade väljalaenutamisega. Kuigi nende sihtgrupp ja tegevuse ajendid on saranased käesoleva projekti omadele – soov aidata säästa loodust ja raha, lastele vahelduse pakkumine ja mänguasjade kuhjumise vältimine,

toimivad nende süsteemid teistel põhimõtetel. Mänguasju laenutatakse välja ühe keskse pakkuja poolt ja mänguasja peab pärast laenuperioodi tagastama. Väljalaenutamise eest küsitakse kas kuutasu või laenutusperioodi pikkusest tulenevat hinda.

3.3.2 Smartswap

Seevastu on SmartSwap OÜ (https://www.smartswap.com) tegevuspõhimõtted väga sarnased planeeritava süsteemi omadele. Ka nende puhul on tegemist rahatu süsteemiga, kus kasutajad saavad omavahel vahetada erinevaid esemeid. Iga eduka vahetuse eest teenib loovutaja 1 punkti ning tellija kaotab 1 punkti. Teenitud punkte saab portaalis kasutada uute tellimuste esitamiseks. Lisaks vahetuste vahendamisele pakuvad nad ka kulleriteenust. Kuigi teenus on sarnane, ei keskendu see spetsiifiliselt mänguasjadele.

3.4 Skoop

Ülalpool on kirjeldatud lahendust, mille korralikuks väljaarendamiseks kulub väga palju rahalist ja ajalist ressurssi. Kui arvesse võtta lõputöö maht, autori rahalised võimalused ja ka oskused, siis päris sellist lahendust realiseerida ei ole esialgu võimalik. Lõputöö raames on eesmärgiks saada valmis MVP. Välja peavad jääma lahendused, mis nõuavad ettevõtte registreerimist või arvestatavat rahalist väljaminekut. Kui rääkida agiilsest tarkvaraarenduse protsessist, siis käesolevasse töösse võib läheneda kui projekti esimesse iteratsiooni. Esimese iteratsiooni eesmärk on valmis saada toode, millel on olemas põhifunktsioonid ja mida on võimalik testkasutajate poolt testida. See, mis funktsionaalsuste arendamine esimeses iternatsioonis eesmärgiks võetakse, tehakse lõplikult selgeks neljandas peatükis.

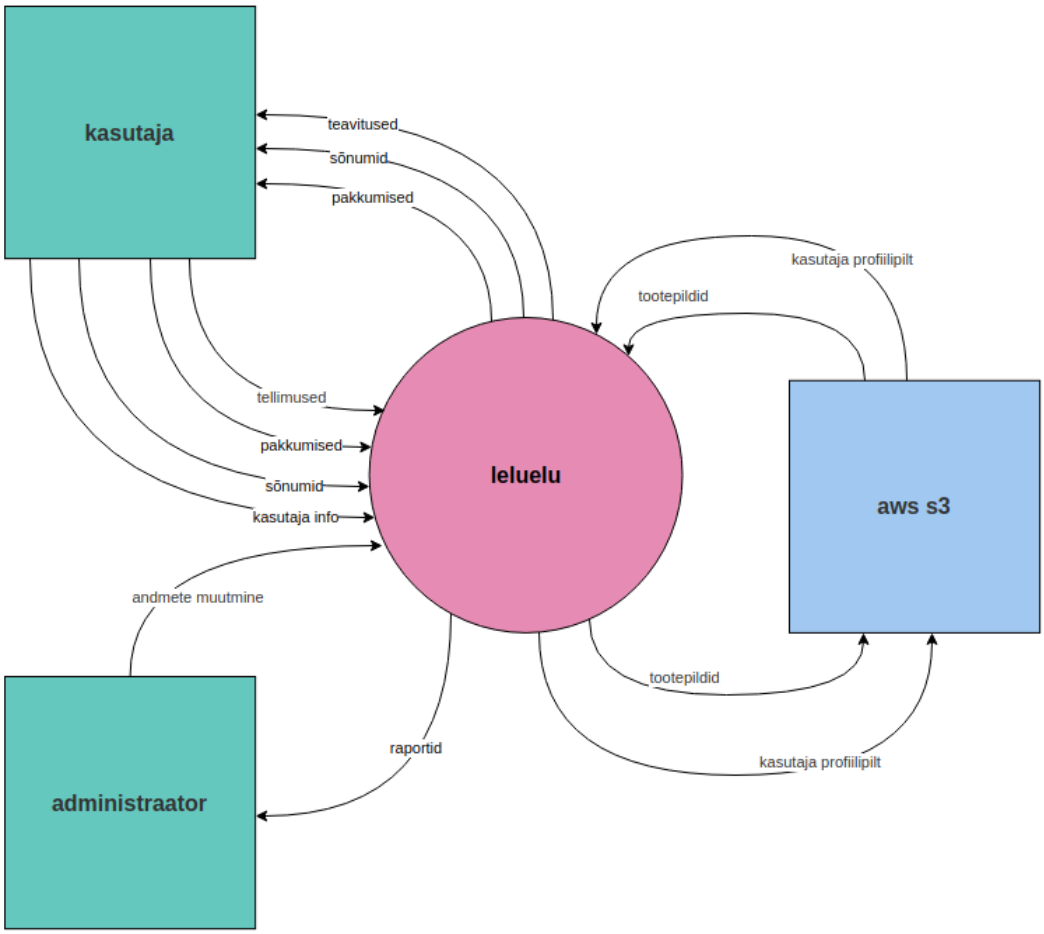
4 Nõuete analüüs

Kui teises peatükis tegeleti nõuete väljaselgitamiseks vajaminevate meetodite tutvustamisega, siis käesolevas peatükis proovitakse neid meetodeid rakendada. Edasiarendustega seonduvad nõuded jäetakse edaspidi kõrvale. Järgmistes alapeatükkides tegeletakse nõuete dokumenteerimisega, alustades kõige üldisemast konteksti diagrammist. Sellele järgneb püüd dokumenteerida stsenaariume, kasutades kasutusjuhtumite diagrammi ja kasutusjuhtumite spetsifikatsiooni. Viimaks keskendutakse nõuetele kõige spetsiifilisemal viisil. Funktsionaalseid nõudeid kirjeldatakse kolmest erinevast perspektiivist: andmed, funktsionaalsus ja käitumine. Mittefunktsionaalsed nõuded jagunevad kvaliteedinõueteks ja piiranguteks. Nõuete dokumenteerimisel on kasutatud ulatuslikult mudelite keelt ning diagrammide koostamiseks on kasutatud projektihaldustarkvara Visual Paradigm Community Edition (https://www.visual-paradigm.com). Et mudelite abstraktsust veidigi kompenseerida, on püütud neid lahti seletada ning nõudeid naturaalses keeles välja kirjutada.

4.1 Süsteemi konteksti diagramm

Süsteemi konteksti diagramm annab süsteemist kõrgetasemelise ülevaate. Joonisel 1 oleval diagrammil on näha loodava süsteemi konteksti, mis sisaldab süsteemi ennast (leluelu) ja komponente, millega see süsteem suhtleb. Nooled näitavad andmete liikumist loodava süsteemi ja nende komponentide vahel, mistõttu teatakse seda diagrammi ka tase 0 andmevoo diagrammina. Kõik, mis on diagrammil, jääb sissepoole konteksti piiri ning on süsteemi loomise juures vaja silmas pidada. Süsteemi piiri illustreerib süsteemi ja väliste komponentide eraldatus diagrammil. Välised komponendid moodustavad konteksti selle osa, mille üle arendajal kontroll puudub. Kusjuures nendeks komponentideks on nii kasutajad, kes süsteemi kasutama hakkavad, kui ka välised teenusepakkujad, kelle teenuseid on plaanis kasutama hakata. Süsteemi

konteksti diagramm ei ole tehniline joonis, vaid väga kõrgetasemeline illustratsioon, mis on eelkõige mõeldud mitte-tehniliste alade esindajatele süsteemi olemuse lihtsamaks mõistmiseks.

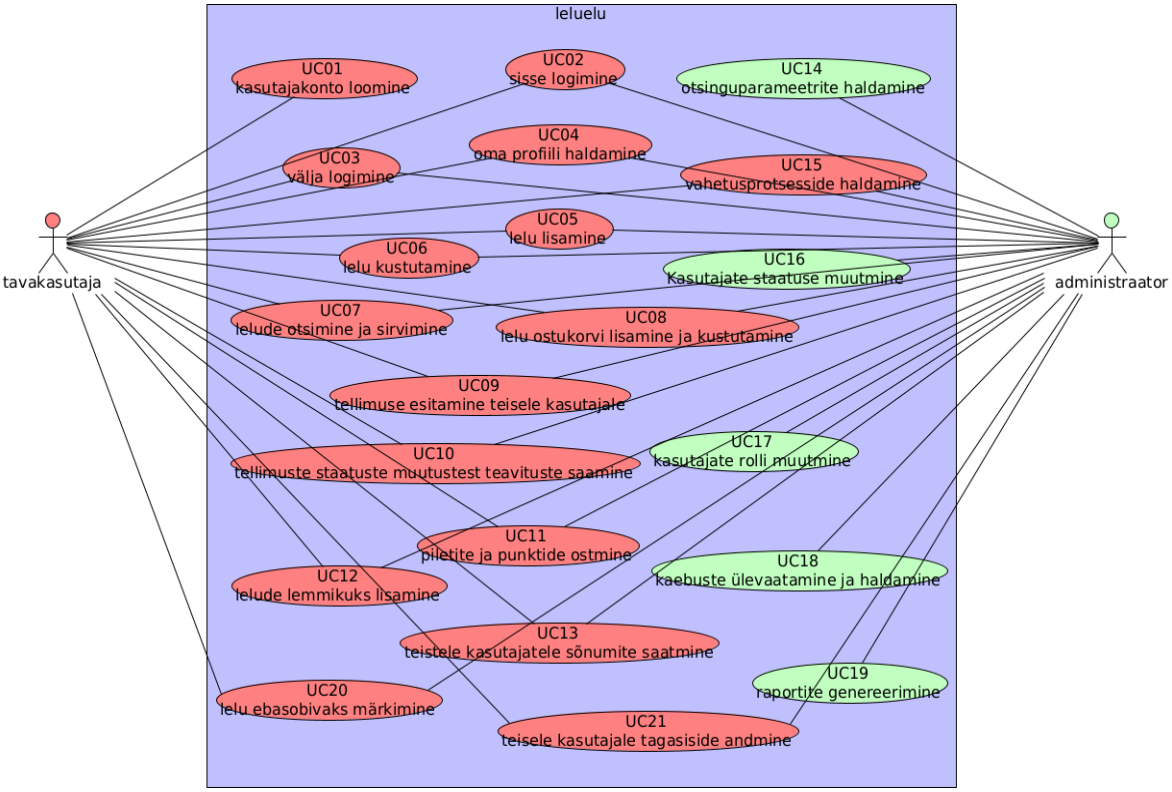


Joonis 1. Süsteemi konteksti diagramm

4.2 Stsenaariumid

Stsenaariumeid saab väljendada kasutusjuhtumitena. Selles alapeatükis kirjeldatakse kasutusjuhtumeid nii naturaalses kui ka mudelite keeles. Joonisel 2 on näha kasutusjuhtumite diagramm, mis kujutab interaktsiooni aktorite ja süsteemi vahel. Nagu diagrammilt selgub, on süsteemis kaks aktorit – tavakasutaja ja administraator. Nagu näha, siis administraator saab teha peaaegu kõiki toiminguid, mida tavakasutajagi,

samas kui tavakasutaja võimalused süsteemis on rohkem piiratud. Igal kasutusjuhtumil on oma unikaalne ID, et oleks võimalik järge pidada. Kasutusjuhtumeid võib juba käsitleda funktsionaalsete nõuetena [13, lk 48].



Joonis 2. Kasutusjuhtumite diagramm

Tabelis 1 on näha veidi selgem ülevaade tuvastatud kasutusjuhtumitest, millele on lisatud prioriteedi väljaselgitamise tarbeks hinnanguline väärtus ja maksumus. Detailsemad kasutusjuhtumite spetsifikatsioonid leiab peatüki Lisa 4 alt. Kasutusjuhtumite dokumenteerimiseks on kasutatud malli, millel on nõuete enda kirjeldamise vaatepunktist vaid olulisemad väljad: ID, pealkiri, kirjeldus, käivitav sündmus, aktorid, eeltingimus, järeltingimus, tulemus, põhistsenaarium, alternatiivstsenaarium ja erindistsenaariumid.

Tabel 1: Kasutusjuhtumid

ID	Pealkiri	Aktorid	Kirjeldus	Väärtus	Maksumus
UC01	Kasutajakonto loomine	Tavakasutaja, Administraator	Kasutaja peab saama portaalis registreerida endale konto	Kõrge	Madal
UC02	Sisse logimine	Tavakasutaja, Administraator	Kasutaja peab saama registreeritud kontoga sisse logida	Kõrge	Madal
UC03	Välja logimine	Tavakasutaja, Administraator	Sisse logitud kasutaja peab saama välja logida	Kõrge	Madal
UC04	Oma profiili haldamine	Tavakasutaja, Administraator	Sisse logitud kasutaja peab saama muuta oma profiili andmeid	Kõrge	Madal
UC05	Lelu lisamine	Tavakasutaja, Administraator	Sisse logitud kasutaja peab saama lisada pakkumisse lelusid	Kõrge	Madal
UC06	Lelu kustutamine	Tavakasutaja, Administraator	Sisse logitud kasutaja peab saama oma lisatud lekusid kustutada	Kõrge	Madal
UC07	Lelude otsimine ja sirvimine	Tavakasutaja, Administraator	Iga kasutaja peab saama otsida ja sirvida lelusid, mis on aktiivses staatuses	Kõrge	Madal
UC08	Lelu ostukorvi lisamine ja kustutamine	Tavakasutaja, Administraator	Sisselogitud kasutaja peab saama lisada lelusid ostukorvi	Kõrge	Kõrge
UC09	Tellimuse esitamine teisele kasutajale	Tavakasutaja, Administraator	Sisselogitud kasutaja peab saama esitada ostukorvis olevate lelude kohta teisele kasutajale tellimuse	Kõrge	Kõrge

ID	Pealkiri	Aktorid	Kirjeldus	Väärtus	Maksumus
UC10	Tellimuste staatuste muutustest teavituste saamine	Tavakasutaja, Administraator	Sisselogitud kasutaja peab saama teavitusi temaga seotud tellimuste staatuste muutumiste kohta	Madal	Madal
UC11	Piletite ja punktide ostmine	Tavakasutaja, Administraator	Sisselogitud kasutaja peab saama soovi korral osta pileteid ja punkte, mis võimaldavad tal portaalis vahetusi läbi viia	Madal	Kõrge
UC12	Lelude lemmikuks lisamine	Tavakasutaja, Administraator	Sisselogitud kasutaja peab saama soovi korral lisada meeldivaid lelusid lemmikutesse, et tal oleks võimalik neid hiljem lihtsamini üles leida	Madal	Madal
UC13	Teistele kasutajatele sõnumite saatmine	Tavakasutaja, Administraator	Sisselogitud kasutaja peab saama soovi korral saata teistele kasutajatele sõnumeid ja käimasolevaid vestlusi lugeda	Kõrge	Kõrge
UC14	Otsinguparameetrite haldamine	Administraator	Sisselogitud administraatori õigustega kasutaja peab saama lisada ja muuta otsinguparameetreid	Kõrge	Madal

ID	Pealkiri	Aktorid	Kirjeldus	Väärtus	Maksumus
UC15	Vahetusprotsesside haldamine	Tavakasutaja, Administraator	Sisselogitud kasutaja peab saama hallata temaga seotud käimasolevaid tellimusi. Administraatori õigustega kasutaja peab saama hallata kõiki süsteemis käimasolevaid telimusi	Kõrge	Kõrge
UC16	Kasutajate staatuse muutmine	Administraator	Sisselogitud administraatori õigustega kasutaja peab saama muuta teiste kasutajate staatust	Madal	Madal
UC17	Kasutajate rolli muutmine	Administraator	Sisselogitud administraatori õigustega kasutaja peab saama muuta teiste kasutajate rolli	Madal	Madal
UC18	Kaebuste ülevaatamine ja haldamine	Administraator	Sisselogitud administraatori õigustega kasutaja peab saama näha ja hallata kasutajate poolt tehtud kaebusi	Madal	Kõrge

ID	Pealkiri	Aktorid	Kirjeldus	Väärtus	Maksumus
UC19	Raportite genereerimine	Administraator	Sisselogitud administraatori õigustega kasutaja peab saama esitada serverisse päringuid, mille põhjal genereeritakse raport, mida on võimalik exceli failina alla laadida. Samuti peavad näha olema raportid, mis on juba genereeritud	Madal	Kõrge
UC20	Lelu ebasobivaks märkimine	Tavakasutaja, Administraator	Igal sisselogitud kasutajal peab olema võimalus raporteerida ebasobivaid tooteid	Madal	Madal
UC21	Teisele kasutajale tagasiside andmine	Tavakasutaja, Administraator	Igal sisselogitud kasutajal peab olema võimalus jätta tagasiside kasutajale, kellega on oldud vahetusprotsessis. Tagasiside saab anda peale vahetusprotsessi lõppemist – nii eduka lõpu kui ka tühistamise korral	Madal	Madal

Dokumenteeritud kasutusjuhtumid sisaldavad ka selliseid juhtumeid, mis antud töö skoopi ei mahu. Siinkohal on võimalik rakendada investeeringutasuvuse tabelit, et välja selgitada, millised kasutusjuhtumid ja mis järjekorras töösse võtta ja millised esialgu välja jätta.

Investeeringutasuvuse tabel aitab prioritiseerida tööde järjekorda. Loodud maatriksist (Tabel 2) on see järjekord hästi näha. Hinnangute andmisel võeti arvesse juhtumi olulisust MVP'le, seega kuigi osad juhtumid leidsid oma lõpu „jäta välja" kastis, ei tähenda, et need on täiesti maha kantud ideed. Prioriteedid on ajas muutuvad ning samuti ideed, mis loovad väärtust. Edasisest nõuete dokumenteerimisest selles töös jäetakse kõige vähem prioriteetsesed kasutusjuhtumid aga välja.

Tabel 2: Kasutusjuhtumite paiknemine investeeringutasuvuse tabelis

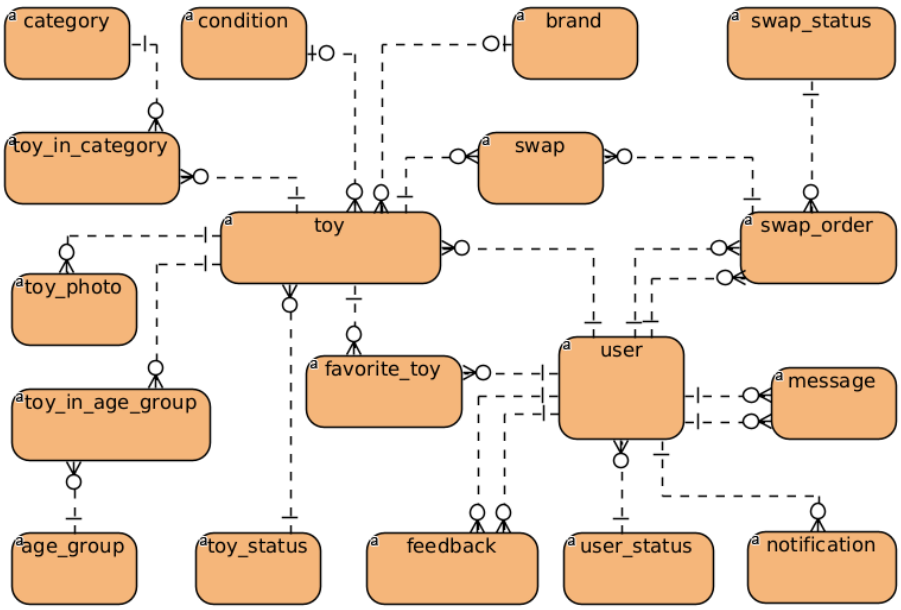
	Kõrge maksumus	**Madal maksumus**
Kõrge väärtus	*Tee hiljem:* UC08, UC09, UC13, UC15	*Tee kohe:* UC01, UC02, UC03, UC04, UC05, UC06, UC07,UC14
Madal väärtus	*Jäta välja:* UC11, UC18, UC19	*Tee palju hiljem:* UC10, UC12, UC16, UC17, UC20, UC21

4.3 Funktsionaalsed nõuded

Funktsionaalseid nõudeid on võimalik vaadata kolme pidi: andmete perspektiivist, funktsionaalsest perspektiivist ja käitumuslikust perspektiivist. Selles alapeatükis kirjeldatakse neid perpektiive põhiliselt mudelite abil, kuid lõppu lisatakse identifitseeritud funktsionaalsetest nõetest ka kirjalik loend.

Andmete perspektiivi parimaks kirjeldusviisiks on olem-suhte diagramm. Joonisel 3 on kujutatud loodava süsteemi relatsioonilise andmebaasi mudeli lihtsustatud versiooni. Täielik versioon on lisatud peatükki Lisa 6. Andmebaasimudeli loomisel on peetud silmas, et see vastaks vähemalt esimesele normaliseeritud kujule [4] . Et seda saavutada, on igal real oma unikaalne ID, on loodud eraldi tabelid mitu-mitmele relatsioonide jaoks ning kantud hoolt, et üheski tulbas ei hoitaks rohkem kui üht informatsioonikildu. Andmebaasi normaliseerimisel on palju häid omadusi, kuid kaks neist väärivad eraldi välja toomist. Esiteks aitab see vähendada andmete kordumist, vähendades andmete hoiustamiseks vajamineva ruumi vajadust. Teiseks aitab see struktureerida andmeid

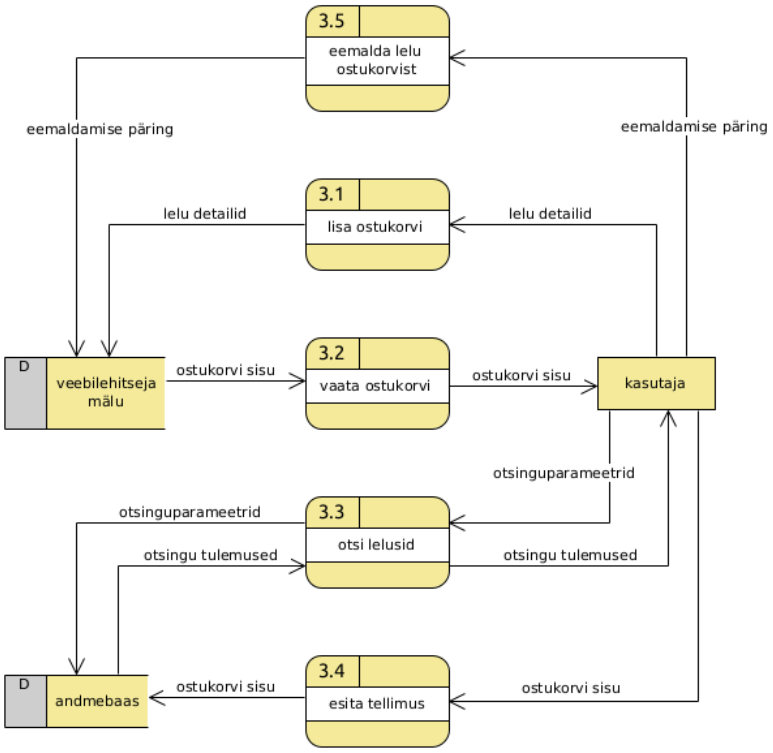
loogiliselt, vähendades oluliselt nii andmemudeli kui ka andmete enda muutmisega seonduvaid probleeme ning muutes päringud kiiremaks [1] .



Joonis 3: Lihtsustatud olem-suhte diagramm

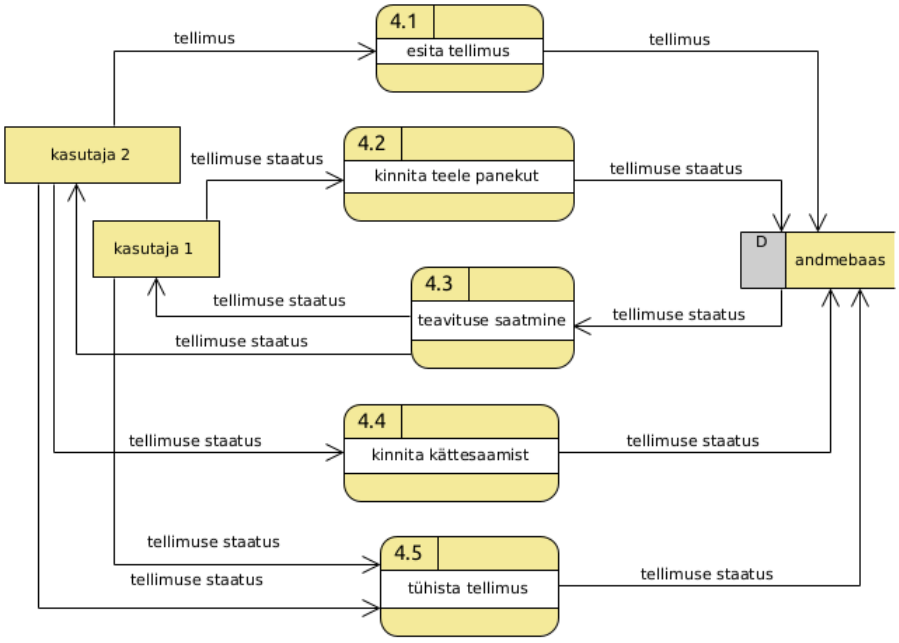
Andmevoo diagrammid näitavad info liikumist läbi süsteemi või protsessi [12] . Nende abil on lihtne mõista, kust andmed pärinevad, kuidas need läbi süsteemi liiguvad, kuidas neid muudetakse ja kuhu need lõpuks välja jõuavad. Andmevoo diagramme saab kujutada väga üldiselt, luues järjest detailsemaid aladiagramme. Seetõttu saab jaotada need diagrammid erinevates tasemeteks [8] . Tase 0 andmevoo diagramm aitab mõista süsteemi konteksti ning on kasutatud ka käesolevas töös süsteemi konteksti illustreerimiseks (Joonis 1). Tase 1 andmevoo diagramm on veidi detailsem, kuid kirjeldab protsesse siiski väga üldiselt. Tase 2 diagrammid on tase 1 diagrammide alamdiagrammid, mis kujutavad protsesse juba detailsemalt. Niimoodi võib minna aina detailsemaks ja aina sügavamale, kuid antud süsteemi funktsionaalsete nõuete kirjeldamiseks piisab täiesti ka kuni tase 2 diagrammidest. Lisast 7 leiab ülevaate kõikidest tase 1 diagrammidest ning vaid keerulisemate protsesside tase 2 andmevoo diagramme vaadeldakse lähemalt siin alapeatükis.

Joonisel 4 on näha tellimuse esitamise tase 2 andmevoo diagramm. Diagrammilt leiab andmehoidlad – veebilehitseja mälu ja andmebaas – mida kasutatakse andmete hoiustamiseks. Vahetusprotsessi alamprotsessid on nummerdatud viitega tase 1 andmevoo diagrammile, mille alamdiagramm see on. Välise aktorina on näha kasutaja, kes süsteemiga suhtleb. Diagrammilt on näha, et kasutaja saab lisada lelusid ostukorvi ja sealt neid ka eemaldada. Ostukorvi sisu salvestatakse esialgu veebilehitseja mällu. Andmebaasi salvestatakse juba esitatud tellimus.



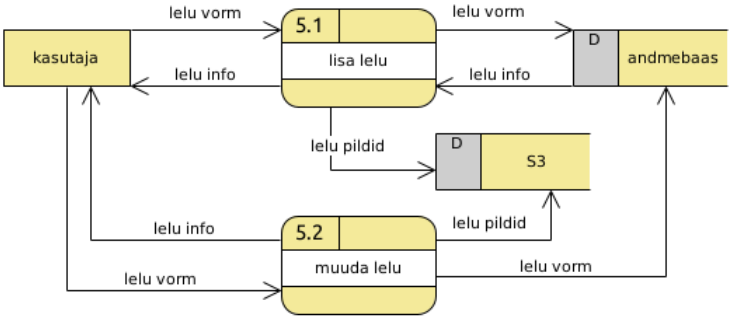
Joonis 4: Tellimuse esitamise tase 2 andmevoo diagramm

Joonisel 5 on kirjeldatud andmete liikumist vahetusprotsessi korral. Diagrammilt on näha, et kui üks kasutaja esitab teisele kasutajale tellimuse, salvestatatakse see tellimus andmebaasi ning tellimuse saajale edastatakse selle kohta teavitus. Kui kasutaja võtab tellimuse vastu ja paneb lelu tellija poole teele, saab ta süsteemis seda kinnitada. Seevastu saab tellija kinnitada, kui ta on lelu kätte saanud. Mõlemad osapooled saavad tellimust ka tühistada. Iga kord, kui tellimuse staatus muutub, saadetakse teisele osapoolele selle kohta ka teavitus.



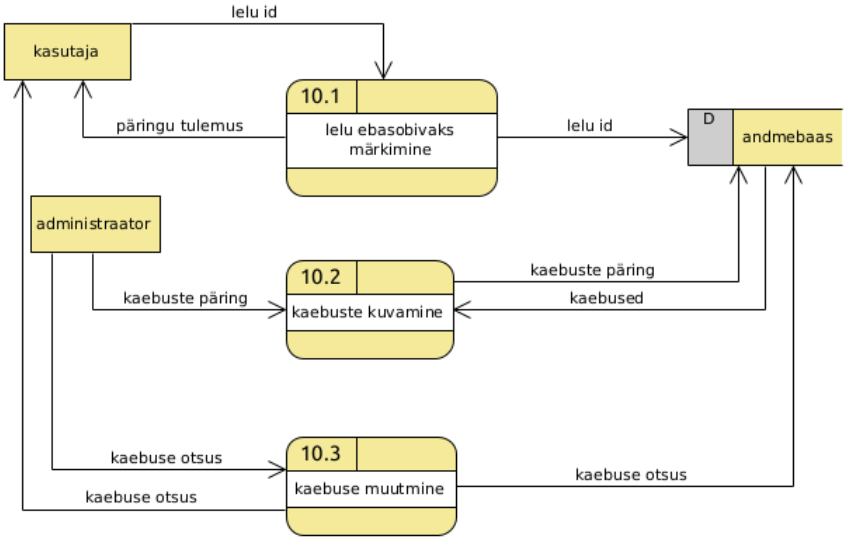
Joonis 5: Vahetusprotsessi tase 2 andmevoo diagramm

Et aga oleks midagi, mille kohta esitada tellimusi, peavad kasutajad saama lisada uusi lelusid pakkumisse. Samuti peavad kasutajad saama muuta oma lisatud lelusid. Joonisel 6 on kujutatud just seda protsessi. Kasutaja saadab süsteemi uue lelu ankeedi. Lelu kohta sisestatud andmed salvestatakse andmebaasi ning leluga seotud pildid salvestatakse pilvehoidlasse. Sarnane protsess toimub ka lelu muutmise ajal, kuid siis muudetakse olemasolevat andmebaasi kirjet.



Joonis 6: Lelu lisamise ja muutmise tase 2 andmevoo diagramm

Üks lahtiseletamist vajav protsess on ka kaebuste esitamine ja haldamine. Joonisel 7 on näha, et kasutajad saavad märkida nende meelest ebasobivaid lelusid mittesobivaks. Kaebustega tegeleb administraator, kellel on võimalus saada kaebustest ülevaade ning sisestada nende kohta süsteemi otsus. Otsusest teavitatakse ka kaebuse esitajat.



Joonis 7: Ebasobivate lelude haldamise tase 2 andmevoo diagramm

Käitumusliku perspektiivi illustreerimiseks võetakse appi olekumasina diagramm. Kogu veebikeskkonda illustreeriv olekumasina diagramm on pandud peatükki Lisa 8.

Funktsionaalsed nõuded saab kokku võtta järgmiselt:

Kasutajad peavad saama:

- registreerida kasutajakonto
- registreeritud kontoga sisse logida ja välja logida
- lisada pakkumisse lelusid
- otsida ja sirvida lelusid
- märkida tooteid ebasobivaks
- lisada lelusid ostukorvi

- esitada tellimus teisele kasutajale
- jätta teisele kasutajale tagasiside pärast vahetusprotsessi edukat või mitteedukat lõppemist
- muuta tellimuse staatust
- muuta oma konto andmeid
- teenida edukate tellimuste pealt punkte
- saada teavitusi neile oluliste muudatuste kohta süsteemis
- saata teistele kasutajatele sõnumeid
- lugeda sõnumeid, mis teised kasutajad neile saatnud on

Administraatorid peavad saama:

- teha kõiki samu toiminguid, mis tavakasutajagi
- manageerida otsinguparameetreid
- manageerida teisi kasutajaid, muu hulgas muuta vajadusel nende staatust (ajutiselt blokeerida), lisada nende kontole punkte jne
- manageerida vahetusi
- manageerida mänguasju, muu hulgas eemaldada ebasobivaid tooteid
- ülevaate kasutajate poolt tehtud kaebustest, et tal oleks võimalik üles leida ebasobivaid tooteid või kasutajaid
- esitada süsteemi kaebuse kohta otsus

4.4 Kvaliteedinõuded

Koos kliendiga töötati välja veebirakenduse prototüüp, mis aitab visualiseerida soovitava süsteemi kasutatavust ja väljanägemist. Prototüübi kuvatõmmiseid on peatükis Lisa 3. Kuigi mobiilivaatele prototüüpi ei loodud, tuleb siiski tagada, et rakendus oleks piisavalt hästi kasuatav erineva ekraanilaiustega seadmetel. Hea kasutajakogemuse loomiseks on oluline ka kasutaja tegevustele tagasiside andmine. Kui andmete sisestamisel tekkis viga, siis peab kasutaja saama teada, milles viga seisnes. Näiteks kasutajakonto loomisel tuleb kasutajale teada anda, kui konto loomine ebaõnnestus, ning mis põhjusel see ebaõnnestus. Sama kehtib ka teiste ankeetide kohta. Kasutaja peab aru saama, mis väljad on kohustuslikud. Lisaks sellele tuleb tagada, et kui kasutaja vajutab mingil nupul, mille tagajärjel kasutaja vaatepunkist midagi ei muutu, peab kasutajale teada andma, kas nupu vajutamise soovitud tagajärg saavutati või mitte [16] .

Kvaliteedinõuded saab kokku võtta järgmiselt:

- Rakendus peab olema meeldiva väljanägemisega
- Rakendus peab olema kasutatav erineva ekraanilausega seadmetel
- Kasutaja tegevustele peab alati andma mingisuguse tagasiside, et nad saaksid aru, et midagi toimus
- Kui kasutaja sisestab mittevaliidset infot, tuleb talle sellest teada anda
- Kui serveris juhtus viga, ei tohi kasutajakogemus sellest kannatada

4.5 Piirangud

Selle projekti juures tuleb arvestada teatud kitsendustega. Antud projekti rahaline ja ajaline ressurss on äärmiselt piiratud. Eesmärgiks on võetud MVP loomine ilma suuri kulutusi tegemata, mistõttu ei ole võimalik kõiki planeeritud featuure kohe implementeerida. Üheks suureks piiranguks on ka asjaolu, et mõnede väliste teenuste

kasutamiseks on vaja registreerida ettevõte, mis esialgu ei ole veel autori soov. See tingimus välistab integratsiooni nii kasutaja verifitseerimist pakkuvate teenustega, kui ka maksekeskusega, mis tähendab, et esialgu ei ole võimalik võimaldada kasutajatele tasulisi teenuseid – pileti- ja punktimüüki. Rahalise ressursi puudus seab teatud piirangud ka infrastruktuurile – kasutada saab vaid tasuta või kõige odavamaid pilveteenuseid.

Piirangud saab kokku võtta järgmiselt:

- Rakendus ei tohi kasutada tasulisi teenuseid
- Rakendus ei tohi kasutada teenuseid, mis nõuavad ettevõtte registreerimist

5 Veebirakenduse arendus

Selles peatükis antakse ülevaade veebirakenduse arendamisest. Põgusalt põhjendatakse tehnoloogiate valikut ning kirjeldatakse detailsemalt nii ees- kui tagarakenduse ülesehitust. Lõpuks esitletakse ka rakenduse juurutamise skeemi, et lugejale tekiks ettekujutus, kuidas rakendus juurutatud on.

5.1 Tehnoloogiate valik

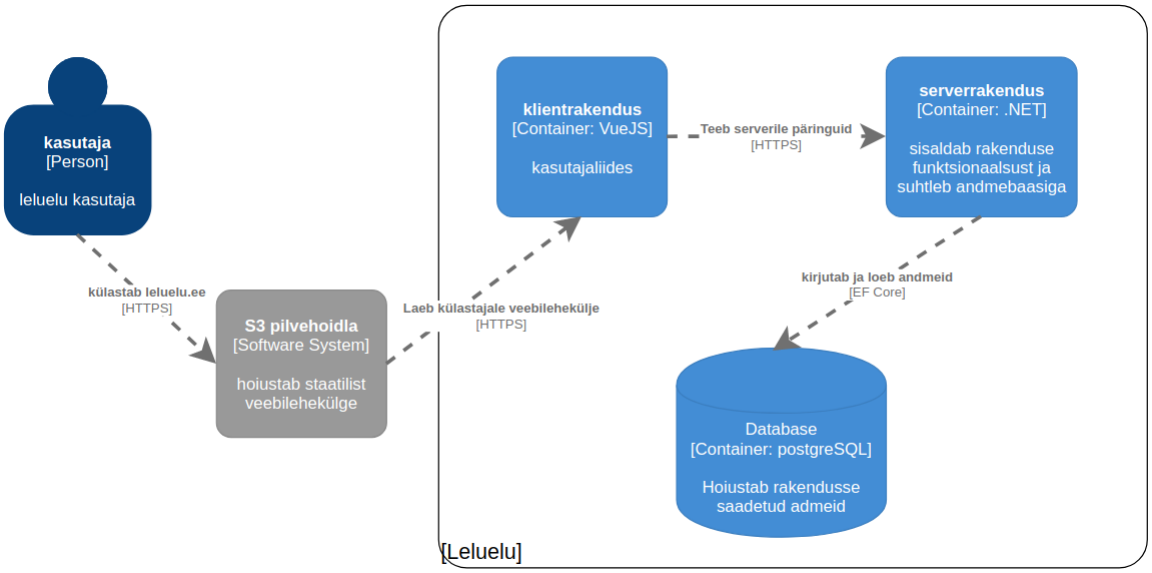
Tagarakenduse arendamiseks kaaluti nii Java programmeerimiskeelt kui ka .NET raamistikku, mis baseerub programmeerimiskeelel C#. Kuivõrd töö autoril on rohkem kogemust Java programmeerimiskeelega, otsustati lõpuks .NET'i kasuks, kuna sooviti saada selles vallas põhjalikum kogemus.

Eesrakenduse arendamiseks valiti JavaScript'i raamistike React ja VueJS vahel. Need raamistikud on üpris sarnased, olles üles ehitatud taaskasutatavate komponentide kasutamisele, millest veebilehekülg kokku pannakse. Kuigi mõlemad raamistikud on väga laialt levinud ja seega hea toega, otsustati lõpuks VueJS'i kasuks, kuna selle dokumentatsioon on põhjalikum ning süntaks kergemini loetav – HTML, javascript ja CSS on komponentides selgesti eraldatud blokkides (Joonis 9). Kuna javascript ise on nõrgalt tüübitud keel, mis tähendab, et muutujate tüübid ei ole ette antud, otsustati kasutusele võtta ka TypeScript (https://www.typescriptlang.org). Typescript annab võimaluse kirjutada javascripti koodi, kasutades tüüpe, klasse ja liideseid, mis teeb koodi paremini loetavaks ja ka veakindlamaks. Selle otsuse kasuks rääkis ka VueJS raamistiku valik, kuna Vue ise on kirjutatud typescriptis [5] .

Andmebaasihalduri valikul toetuti autori senisele kogemusele PostgreSQL'iga (https://www.postgresql.org). Tegemist on laialt levinud ja toetatud vabavaralise objekt-relatsioonilise andmebaasi halduriga, mis on oma võimsuse ja töökindluse tõttu kõrgelt

hinnatud. Kuigi vabavaraliste lahenduste seas on laialt levinud ka MySQL, on PostgreSQL viimasega võrreldes võimsam [14] , [17] .

Kuna tegemist on veebirakendusega, peab rakendus olema juurutatud serverisse. Kuna autor ei soovi ise oma serverit hallata, otsustati pilveteenuse kasuks. Kuigi tagarakendus on kirjutatud kasutades Microsofti loodud vahendeid, ei soovitud kasutada Microsofti Azure'i pilveteenust tema piiratud tasuta võimaluste tõttu. Amazon pakub seevastu aastast *free-tier* lahendust (https://aws.amazon.com/free), mis võimaldab kasutada paljusid teenuseid terve aasta tasuta. Server, millel jookseb andmebaas, on küll DigitalOcean'is (https://www.digitalocean.com), kuna töö autoril oli varasemate katsetuste tarbeks seal sobiv server olemas. Joonisel 8 on näha rakenduse skeem koos tehnoloogiatega, mida on plaanis kasutada.



Joonis 8: Leluelu tase 2 C4 diagramm

5.2 Eesrakenduse ülesehitus

Eesrakendus on üles ehitatud SPA rakendusena (üheleherakendus), mis tähendab, et teekonna muuutmisel ei ole veebilehitsejal vaja kogu rakendust uuesti laadida. Andmete muutumisel arvutatakse vaade dünaamiliselt ümber [10] . Olulisemad osad loodud

eesrakenduse juures on ruuter, mis seob teekonnad kindlate komponentidega, teenused, mis suhtlevad tagarakendusega, teenused, mis salvestavad andmeid veebilehitseja mällu, komponendid, millest vaated üles on ehitatud, ja VueJS'i sisse ehitatud mälu (Vuex store), kus hoitakse andmeid, millele on vaja komponentideülest ligipääsu. Lisaks nendele mängivad olulist rolli ka erinevad teegid, mida on kasutatud mittefunktsionaalsete nõuete täitmiseks. Selles alapeatükis tutvustatakse lähemalt komponentide kasutamist, suhtlemist tagarakendusega ja mittefunktsionaalsete nõuete täitmist.

5.2.1 Komponendid

VueJS rakendused on üles ehitatud taaskasutatavatele komponentidele, mis tähendab, et rakenduse elemendid on justkui objektid. Iga vaade on eraldi komponent, mis sisaldab omakorda teisi komponente. Komponente saab mugavalt taaskasutada, mistõttu ei ole vaja näiteks ühte sama nuppu mitu korda koodis defineerida. See aitab vältida koodi dubleerimist ning teeb rakenduse arendamise kiiremaks ja lihtsamaks. Kui veebilehitsejas avatakse rakenduse lehekülg, laetakse põhikomponent *App* (Joonis 9). Seejärel vaadatakse URLi järgi, mis komponent antud teekonnale vastab ning see komponent renderdatakse *App* komponendi *router-view* elementi. Seejärel vaadatakse, mis komponendid on kasutusel selle komponendi sees ja niiviisi ehitataksegi komponentidest kokku veebilehekülg. Seda võib ette kujutada komponentide puuna, kus *App* on tüvi ja selles sisalduvad komponendid moodustavad harulise võra. Komponendid saavad oma lapskomponentidele edasi pärandada ka andmeid.

VueJS'i saab kirjutada kahel erineval viisil – objekti ja klassi stiilis. Kuigi mõlemad stiilid on funktsionaalselt samaväärsed, on klassi stiil loetavam ja intuitiivsem [9] . Üheks miinuseks selle lähenemise juures on see, et enamus koodinäiteid ja juhendeid on kirjutatud objekti stiilis. Sellest hoolimata on autor eelistanud just klassi stiili.

```
<template>
  <div id="app">
    <navbar></navbar>
    <div class="page-content container">
      <router-view />
    </div>
  </div>
</template>

<script lang="ts">
import { Component, Vue } from 'vue-property-decorator'
import Navbar from '@/components/navbar/Navbar.vue'
import { namespace } from 'vuex-class'

const AppInfo = namespace('AppInfo')

@Component({
  components: { Navbar }
})
export default class App extends Vue {
  @AppInfo.Action
  private fetchData: () => void;

  created (): void {
    this.fetchData()
  }
}
</script>

<style>
...
<style/>
```

Joonis 9. Koodinäide põhikomponendist App

5.2.2 Bootstrap-Vue

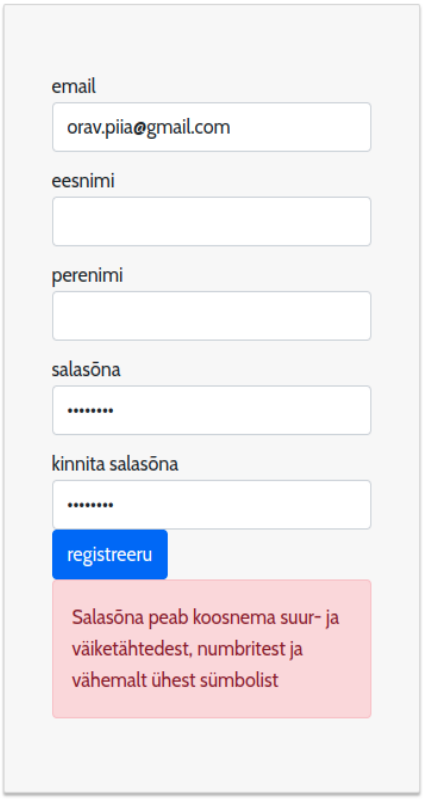
Rakenduse vaadete loomisel on palju kasutatud Bootstrap-Vue teegi võimalusi (https://bootstrap-vue.org). Bootstrap-Vue pakub palju erinevaid juba valmis VueJS

komponente, mida saab tasuta oma lehekülje ehitamisel kasutada. Selle hulka kuuluvad näiteks erinevad ankeediväljad, teadete paneelid, modaalaknad, pildikaardid ja palju muud. Kõige rohkem on aga kasutatud võresüsteemi[1], mis aitab hallata elementide paigutust ekraanil sõltumata ekraanilaiusest. Võresüsteem reageerib erinevatele ekraanilaiustele ning organiseerib vaate vastavalt sellele ümber.

5.2.3 Ankeetide valideerimine

Ankeetide valideerimiseks on kasutatud populaarset sihtotstarbelist teeki VeeValidate (https://vee-validate.logaretm.com/v2). Selle abil saab hõlpsasti määrata kliendi poolseid valideerimisreegleid ja lokaliseerida veateateid. Joonisel 10 on näidatud registreerimisankeedi valideerimist, kus antakse kasutajale teada, et valitud salasõna ei vasta seatud reeglitele. Siiski ei tohi ankeetide valideerimisel jääda lootma ainult sellele, kuna nii HTML'i kui ka päringuid on võimalik üpris lihtsasti modifitseerida. Seetõttu on peetud oluliseks valideerida andmeid ka tagarakenduses.

1 Võresüsteemist loe lähemalt siit: https://bootstrap-vue.org/docs/components/layout#layout-and-grid-system.



Joonis 10: Valideerimine vee-validate teegi abil

5.2.4 Suhtlemine tagarakendusega

Enamus suhtlusest ees- ja tagarakenduse vahel toimub üle HTTP protokolli, kus eesrakendus teeb GET, POST, PUT ja ka DELETE päringuid tagarakenduse kontrollerite pihta. Päringute tegemiseks on kasutatud Axios'e teeki (https://axios-http.com). Kuigi javascriptil on päringute saatmise võimekus natiivselt olemas, pakub see teek kasutajasõbralikku liidest ning on seetõttu lihtsam kasutada. Joonisel 11 on näide POST päringust. Selle POST päringuga saadetakse tagarakendusele lelu lisamise ankeedist saadud andmed, mistõttu on päises ka öeldud, et tegemist on ankeedi kujul andmetega. Kuna tegemist on funktsionaalsusega, mis on kättesaadav vaid sisselogitud kasutajale, tuleb päisesse lisada ka kasutaja autentimisel loodud JWT, mis pärast autentimist veebilehitseja mällu on salvestatud.

```
const API_URL = ServiceUtility.getApiBaseUrl()

class UserService {
  ...
  createToy (toy: Toy) {
    const fd = this.composeToyForm(toy)

    return axios
      .create({
        headers: {
          'Content-Type': 'multipart/form-data'
        }
      })
      .post(API_URL + 'Toys/CreateToy', fd, { headers: authHeader() })
      .then(
        response => {
          return response.data
        }
      )
  }
  ...
}
export default new UserService()
```

Joonis 11. Koodinäide API päringust

Kuna HTTP päringud ei võimalda kahesuunalist suhtlemist ees- ja tagarakenduse vahel, on kiirsuhtlusrakenduse tarbeks kasutusele võetud SignalR teek. See teek võimaldab tagarakendusel käivitada eesrakenduse meetodeid, eemaldades vajaduse teha HTTP päringuid uue seisu teada saamiseks.

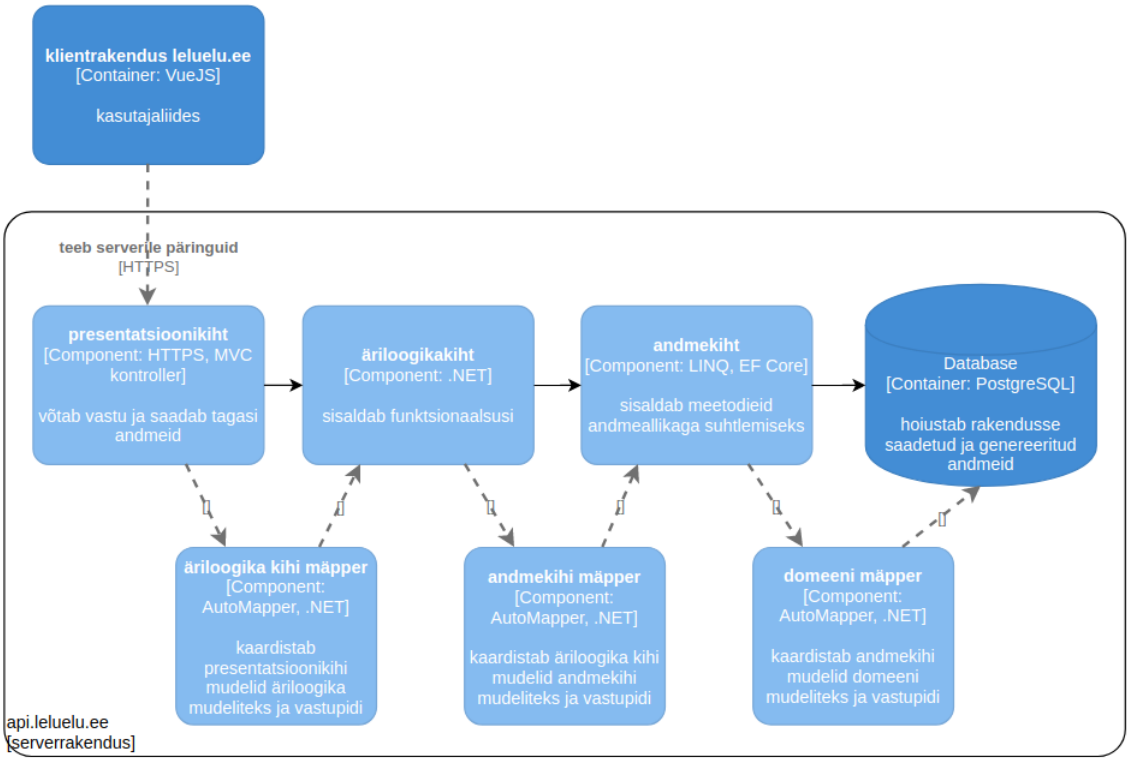
5.2.5 Loodud eesrakendus

Selle töö käigus loodi eesrakendus, kasutades VueJS raamistikku ja erinevaid abistavaid teeke. Prototüübi (Lisa 3) alusel loodi rakenduse vaated ning taaskasutatavad komponendid. Kui võrrelda prototüüpi arendatud rakendusega, siis on näha, et järgitud on prototüübi stiile ja värvilahendusi. Siiski ei vasta päris rakendus täielikult

prototüübile, kuna prototüübil on esitletud funktsionaalsusi, mis otsustati esialgse toote arenduse käigus välja jätta. Olulisemate välja arendatud funktsionaalsuste hulka kuuluvad navigatsioonimenüü, registreerimis- ja sisselogimisankeet, lelu lisamise ja muutmise ankeet, kasutaja koduleht, kasutaja avalik profiil, lelu leht, ostukorv, tellimuste leht ja suhtlusrakendus. Loodud eesrakenduse kuvatõmmised on paigutatud peatüki Lisa 5 alla.

5.3 Tagarakenduse ülesehitus

Tagarakendus on jaotatud kihtideks, millest igaühel on oma kindel eesmärk. Esitluskiht tegeleb päringute vastuvõtmise ja vastuste saatmisega. Selle kihi kaudu toimub kogu rakenduse suhtlus eesrakendusega. Siin kihis asuvad tagarakenduse kontrollerid, mille kaudu võetakse vastu HTTP päringuid. Tagarakenduse API dokumenteerimiseks on kasutatud Open API Spetsifikatsiooni versiooni 3.0.1. API spetsifikatsioon on avalik ja leitav aadressilt *https://api.leluelu.ee/swagger/index.html*. Äriloogikakiht tegeleb rakenduse funktsionaalsusega. Siin kihis asuvad teenused, mis tegelevad andmete muutmisega. Samuti asub siin kihis teenus, mis tegeleb piltide üles laadimisega pilvehoidlasse. Andmekiht tegeleb admete vahendamisega andmeallika ja äriloogika kihi vahel. Selle kihi eesmärk on eemaldada andmete vahendamise loogika sõltuvus andmeallika tehnoloogiast. Iga kihi vahel muundatakse ühe kihi objektid teise kihi objektideks. Joonisel 12 on lihtsustatud skeem kirjeldatud kihtide toimimisest. Edasistes alapeatükkides vaadeldakse veidi lähemalt mõningaid tagarakenduse erilisi funktsionaalsusi.



Joonis 12: Leluelu tagarekenduse tase 3 C4 diagramm

5.3.1 Kontrollerid

Konrollerid on klassid, mis sisaldavad meetodeid, mis käivituvad, kui tuleb sisse mõni HTTP päring. Antud rakenduses on *BaseApiController* klass, mis sisaldab CRUD meetodeid. Seda baasklassi on võimalik vajadusepõhiselt üle kirjutada, kui on vaja rakendada karmimat autoriseerimispoliisi või kui on vaja teise käitumisega meetodeid. Joonisel 13 on näidatud üht baaskontrolleri meetodit. Selle meetodi juures on näha erinevaid annotatsioone, mis on vajalikud autoriseerimise, OpenAPI spetsifikatsiooni ja päringu valideerimise jaoks. Kontrolleri meetod kutsub välja äriloogikakihi meetodit, mis edastab päringu andmekihile, ning tagastab lõpuks päringu tegijale vastava tulemuse.

```
// GET: api/v{version}/Entities/5
/// <summary>
/// Gets entity with given ID.
/// </summary>
/// <param name="id">ID of entity</param>
/// <returns><see cref="OkResult"/>(200) and entity with given
ID.</returns>
[Produces("application/json")]
[ProducesResponseType(StatusCodes.Status200OK)]
[ProducesResponseType(StatusCodes.Status404NotFound)]
[AllowAnonymous]
[HttpGet("{id:long}")]
public virtual async Task<ActionResult<TPublicEntity>> Get(long
id)
{
      var entity = await Service.FirstOrDefaultAsync(id);
      if (entity == null) return NotFound(new
ErrorMessage(ErrorCode.Err0));
      return Ok(Mapper.Map(entity));
}
```

Joonis 13. Baaskontrolleri GET meetod

5.3.2 CORS

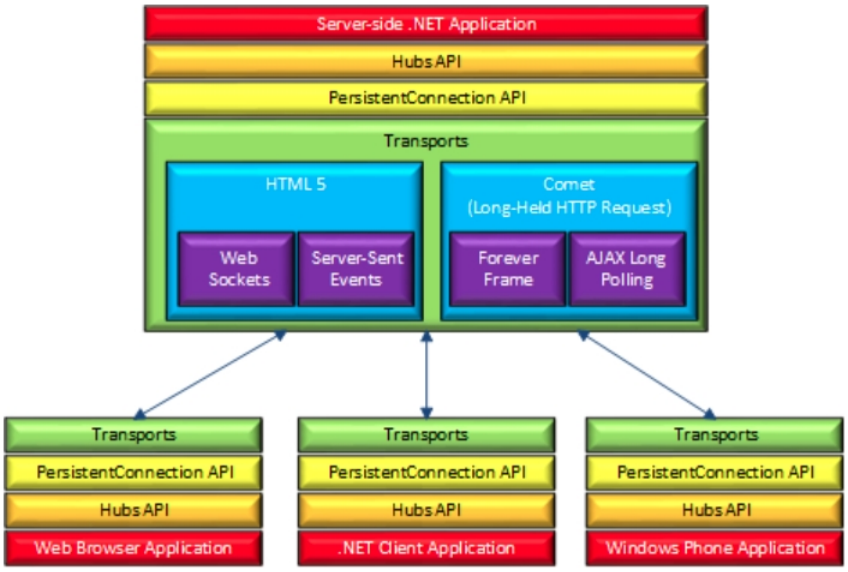
CORS on HTTP-päisetel põhinev mehhanism, mis võimaldab serveri poolel defineerida, milliste päritoludega päringuid veebilehitsejas lubada. Selle jaoks teeb veebilehitseja kõigepealt eelpäringu (*preflight request*), ning kui päritolu on lubatud, lubatakse teha ka tegelik päring [3] . Muu hulgas saab piirata ka meetodeid ja päiseid, lubades läi näiteks ainult GET päringuid, millel on olemas mingi kindel päis. Joonisel 14 on näidatud, kuidas on tagarakenduses CORS kofigureeritud ja millise päritoluga päringud on lubatud.

```
app.UseCors(builder =>
{
    var allowedOrigins = new []{"https://www.leluelu.ee",
"https://leluelu.ee"};
    builder.WithOrigins(allowedOrigins)
        .AllowAnyMethod()
        .AllowAnyHeader()
        .SetIsOriginAllowed((host) => true)
        .AllowCredentials();
});
```

Joonis 14. CORS'i konfiguratsioon

5.3.3 Otseühenduse loomine SignalR abil

Et võimaldada kiirsuhtlusrakenduse tööd, on kasutatud ASP.NET SignalR teeki nii taga- kui ka eesrakenduses. SignalR loob eesrakenduse ja tagarakenduse vahel otseühenduse ning eesrakendus ei pea uue seisu teada saamiseks eraldi päringut tegema. SignalR proovib esmajärjekorras luua WebSocketi ühenduse ning kui see ei õnnestu, proovib teisi ühenduse tüüpe, nagu näiteks pollimine. SignalR kasutab suhtlemiseks nii madalatasemelisi püsiühendusi kui ka püsiühenduste liidestena kõrgetasemelisi huube. Kuna enamasti on soovitatav kasutada abstraktsemaid huube, kasutatakse neid ka selles rakenduses. Lihtsustades saab huubi tööd kirjeldada järgmiselt: kui serverist kutsutakse välja eesrakenduses olev meetod, saadetakse üle aktiivse transpordiühenduse kliendile pakett, mis sisaldab väljakutsutava meetodi nime ja selle parameetreid. Seejärel teab klient, millist meetodit millise parameetritega käivitada [18] . Joonisel 15 on kujutatud SignalR'i arhitektuuridiagrammi, mis on laenatud Microsofti dokumentatsioonist [18] .



Joonis 15: SignalR arhitektuuridiagramm

5.3.4 Kasutaja autentimine ja autoriseerimine

Kuna rakenduse kasutamise üheks nõudeks on kasutajakonto olemasolu, siis on vaja tegeleda kasutajate autentimise ja autoriseerimisega. Microsofti ametlikus dokumentatsioonis on kirjeldatud autentimist ja autoriseerimist järgnevalt. Autentimine on protsess, mille käigus kasutaja sisestab oma kasutaja andmed, milleks on enamasti kasutajatunnus ja salasõna, ning neid andmeid võrreldakse andmebaasi salvestatud andmetega. Kui sisestatud andmed vastavad salvestatud andmetele, siis on kasutaja autenditud ning ta saab teha toiminguid, mille jaoks on vaja olla sisse logitud. Autoriseerimine viitab protsessile, mille käigus otsustatakse, milliseid toiminguid kasutaja teha võib [2] .

Joonis 16: ASP.NET Core Identity liidese lisatud andmebaasitabelid ja relatsioonid

Kasutajakontode haldamiseks on kasutatud *ASP.NET Core Identity* liidest [6] . Selle liidesega tulevad kaasa andmebaasi tabelid (Joonis 16), kus hoitakse informatsiooni rakenduse kasutajate ja rollide kohta, ning vahendid, millega kasutajaid autoriseerida. Kasutajate autoriseerimiseks kasutatakse JWT põhist poliisi ning kui on vaja piirata tegevusi vaid administraatori rolliga kasutajale, saab selleks kasutada ka rollipõhist poliisi.

5.3.5 Andmebaasi päringud

Andmebaasi päringute tegemiseks on välditud toore SQLi kasutamist ning on eelistatud LINQ päringukeelt. Põhiline argument LINQ poolt on see, et see ei hooli andmeallika tüübist – LINQ keelega on võimalik teha päringuid samamoodi nii SQL andmebaaside, kollektsioonide kui ka XML failide pihta. Kuigi loodud rakenduses on andmeallikana kasutusel vaid PostgreSQL'i andmebaas, välistab LINQ eelistamine tulevikus võimaliku andmeallika muutumisel päringute ümberkirjutamise [7] . Samuti on autori hinnangul see hea tööriist vältimaks SQL'i süstimist. LINQ kõrval on võimalik kasutada ka paljast SQL'i, kuid seda võiks teha ainult erandjuhtumeil – näiteks kui päringut on vaja optimiseerida ning LINQ päringukeelega ei ole see võimalik [11] . Joonisel 17 on näide veidi keerukamast andmebaasipäringust LINQ abil.

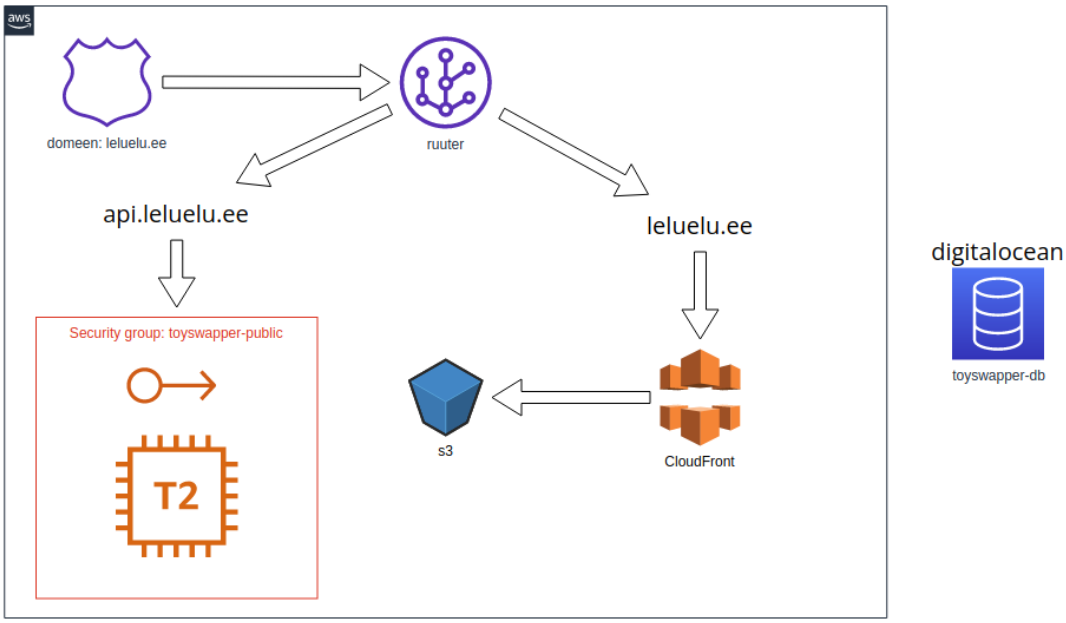
```
public async Task<Dictionary<long, IEnumerable<Message>>>
GetNewMessages(long userId, bool noTracking)
{
    var query = InitializeQuery(userId, noTracking);
    var resultQuery = query
        .Where(m => m.ToApplicationUserId == userId && m.MessageStatus
== "new")
        .Include(u => u.FromApplicationUser)
        .Include(u => u.ToApplicationUser)
        .OrderBy(m => m.CreatedAt)
        .GroupBy(m => m.FromApplicationUserId);

    return (await resultQuery
        .Select(messageGroup => messageGroup.Select(message =>
Mapper.Map(message)))
        .ToDictionaryAsync(d => d.First()!.FromApplicationUserId))!;
}
```

Joonis 17. Koodinäide andmebaasi päringust

5.4 Juurutamine

Joonisel 18 on näha rakenduse juurutamise skeemi. Tagarakendus on juurutatud AWS'i EC2 serverisse dockeri konteinerisse (https://www.docker.com/resources/what-container). Eesrakenduse staatilised failid on laetud AWS'i pilvehoidlasse S3. Kui

tehakse päring leluelu.ee pihta, suunatakse külastaja S3 kaustas olevale veebirakenduse index.html'ile. Server, kus jookseb tagarakendus, on kättesaadav api.leluelu.ee kaudu. Andmebaas asub eraldi DigitalOcean'is ja on ligipääsetav üle võrgu.



Joonis 18: Rakenduse juurutamise skeem

6 Kokkuvõte

Käesolev töö tegeles ühe veebirakenduse arenduse protsessi dokumenteerimisega. Töö sai alguse ideest, millega pöörduti abi saamiseks autori poole. Ideeks oli portaali loomine, kus kasutajad saaksid oma laste vanu mänguasju teiste kasutajatega vahetada. Inspiratsiooni saadi olemasolevast raamatuvahetuse portaalist, kus kasutajad saavad omavahel vahetada raamatuid, kasutades rahatut punktisüsteemi.

Kuna pelgalt idee põhjal on raske midagi teha, selgitati koos kliendiga välja soovid ja ootused loodavale portaalile. Nõuete kogumine sai alguse vestlustest kliendiga. Väga suureks abiks oli ka kliendi soovide visualiseerimine prototüübi kujul. Sellest prototüübist sai peamine mittefunktsionaalsete kvaliteedinõuete allikas. Järgnes nõuete analüüsi faas, mille käigus vaadeldi loodava süsteemi konteksti, selgitati välja kasutusjuhtumid ning nii mudelite kui naturaalse keele abil pandi kirja nõuded.

Arenduse käigus loodi ees- ja tagarakendus, mille juurutamiseks kasutati AWS'i pilveteenuseid. MVP vajadusi arvesse võttes jäi esialgu välja arendamata automaatne pideva arenduse protsess, mistõttu toimub uuenenud rakenduse koodi üles laadimine serverisse ja pilvehoidlasse manuaalselt. Olgu siinkohal mainitud, et analüüsi ja arenduse faas toimusid käsikäes. Rakenduse arendamisega hakati pihta juba väga varajases staadiumis ning projekti edenedes selgusid ka detailsemad nõuded.

Kuna idee on suurem, kui teostuseks võimaldatud ajaline ja rahaline ressurss, tuli teha kompromisse, mille tõttu ei suudetud välja arendada kõiki soovitud funktsionaalsusi. Sellest hoolimata sai valmis üks käegakatsutav lahendus, mida on võimalik lasta testgrupil testida. Kuigi nõuete arendamise faasis seati eesmärgiks välja arendada kõik prioriteetsemad kasutusjuhtumid, tuli kompromisse teha veelgi. Planeeritud funktsionaalsustest jäid välja arendamata süsteemiteavitused, lelude lemmikuks lisamine, teisele kasutajale tagasiside andmine, lelude ebasobivaks märkimine ja erinevad administraatori toimingud.

Selle projektiga kavatsetakse ka edaspidi edasi tegeleda. Töö autor ja klient moodustavad Leluelu meeskonna, kes valiti osalema NULA inkubaatorisse (https://nula.kysk.ee). Inkubatsiooniperioodi ajal on kavas testida toodet testgrupi peal, osaleda erinevates töötubades, viimistleda äriplaani ning lõpuks hakata tegutsema registreeritud ettevõttena.

Kasutatud kirjandus

[1] "*What is Data Normalization and Why Is It Important?*" 2021. [Võrgumaterjal]. Saadaval:https://www.geeksforgeeks.org/what-is-data-normalization-and-why-is-it-important. [Kasutatud 16.05.2022].

[2] *"ASP.NET Core security topics"* 2022. [Võrgumaterjal]. Saadaval: https://docs.microsoft.com/en-us/aspnet/core/security/?view=aspnetcore-6.0. [Kasutatud 16.04.2022].

[3] *"Cross-Origin Resource Sharing (CORS)"*. [Võrgumaterjal]. Saadaval:https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS. [Kasutatud 20.04.2022].

[4] "*First Normal Form (1NF)*" 2022. [Võrgumaterjal]. Saadaval:https://www.geeksforgeeks.org/first-normal-form-1nf. [Kasutatud 16.05.2022].

[5] *"Frequently Asked Questions"*. [Võrgumaterjal]. Saadaval:https://vuejs.org/about/faq.html#should-i-use-javascript-or-typescript-with-vue. [Kasutatud 17.04.2022].

[6] *"Introduction to Identity on ASP.NET Core"* 2022. [Võrgumaterjal] Saadaval:https://docs.microsoft.com/en-us/aspnet/core/security/authentication/identity?view=aspnetcore-6.0&tabs=visual-studio. [Kasutatud: 16.04.2022].

[7] *"Language Integrated Query (LINQ) (C#)"* 2022. [Võrgumaterjal]. Saadaval:https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/linq. [Kasutatud 19.04.2022].

[8] "*Levels in Data Flow Diagrams (DFD)"* 2020. [Võrgumaterjal]. Saadaval:https://www.geeksforgeeks.org/levels-in-data-flow-diagrams-dfd. [Kasutatud 24.04.2022].

[9] *"Overview"*. [Võrgumaterjal]. Saadaval:https://class-component.vuejs.org. [Kasutatud: 19.04.2022].

[10] *"SPA (Single-page application)"*. [Võrgumaterjal]. Saadaval:https://developer.mozilla.org/en-US/docs/Glossary/SPA. [Kasutatud: 19.04.2022].

[11] *"Why LINQ beats SQL"*. [Võrgumaterjal]. Saadaval:https://www.linqpad.net/WhyLINQBeatsSQL.aspx. [Kasutatud 19.04.2022].

[12] C. Clifford, *" A Beginner's Guide to Data Flow Diagrams"* 2021. [Võrgumaterjal]. Saadaval:https://blog.hubspot.com/marketing/data-flow-diagram. [Kasutatud 19.02.2022].

[13] C. Larman, *“Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process”*, 2nd Edition, Pearson (2001)

[14] D. Tobin, *“Which Modern Database Is Right for Your Use Case?”* 2021. [Võrgumaterjal]. Saadaval: https://www.integrate.io/blog/which-database. [Kasutatud 17.04.2022].

[15] K. Pohl, "*”Requirements Engineering. Fundamentals, Principles and Techniques”*, Berlin: Springer, 2010.

[16] M. Rae, *“What Makes a Good User Experience?”* 2020. [Võrgumaterjal]. Saadaval:https://xd.adobe.com/ideas/principles/web-design/what-makes-good-ux. [Kasutatud 24.04.2022].

[17] M. Smallcombe, *“ PostgreSQL vs MySQL: The Critical Differences”* 2020. Võrgumaterjal]. Saadaval:https://www.integrate.io/blog/postgresql-vs-mysql-which-one-is-better-for-your-use-case. [Kasutatud: 17.04.2022].

[18] P. Fletcher, *“ Introduction to SignalR”* 2020. [Võrgumaterjal]. Saadaval:https://docs.microsoft.com/en-us/aspnet/signalr/overview/getting-started/introduction-to-signalr. [Kasutatud 29.01.2022].

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

Mina, Piia-Mai Orav

1 Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Veebirakenduse arendamine mänguasjavahetuse keskkonna Leluelu näitel" mille juhendaja on Meelis Antoi.

1.1 reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;

1.2 üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.

2 Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.

3 Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

16.05.2022

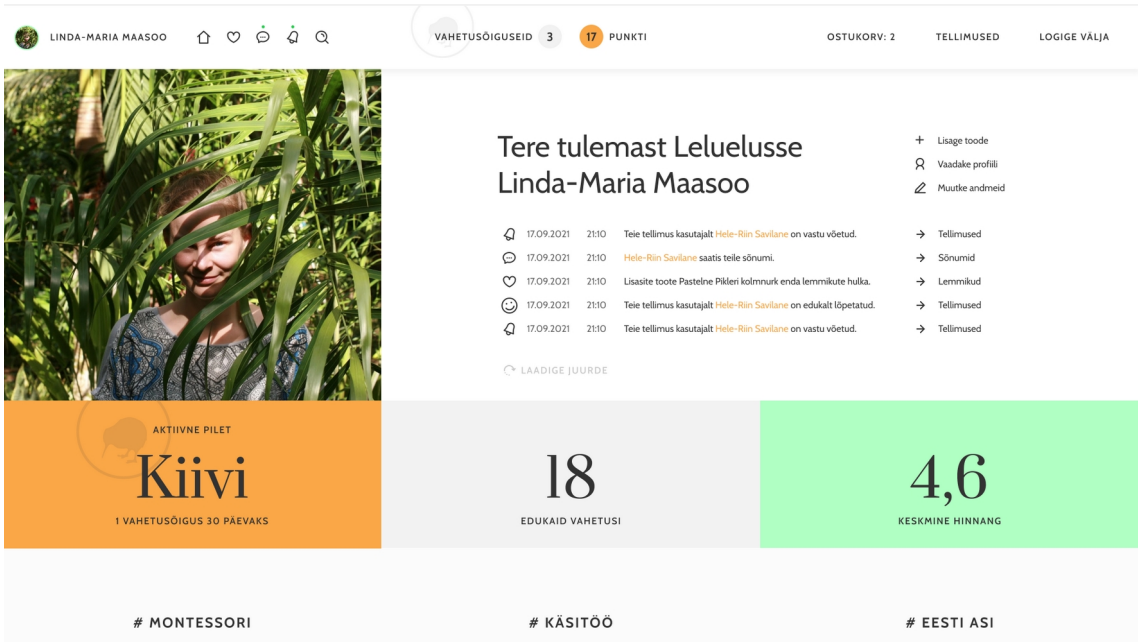
1 Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või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

Lisa 2 – Lähtekood

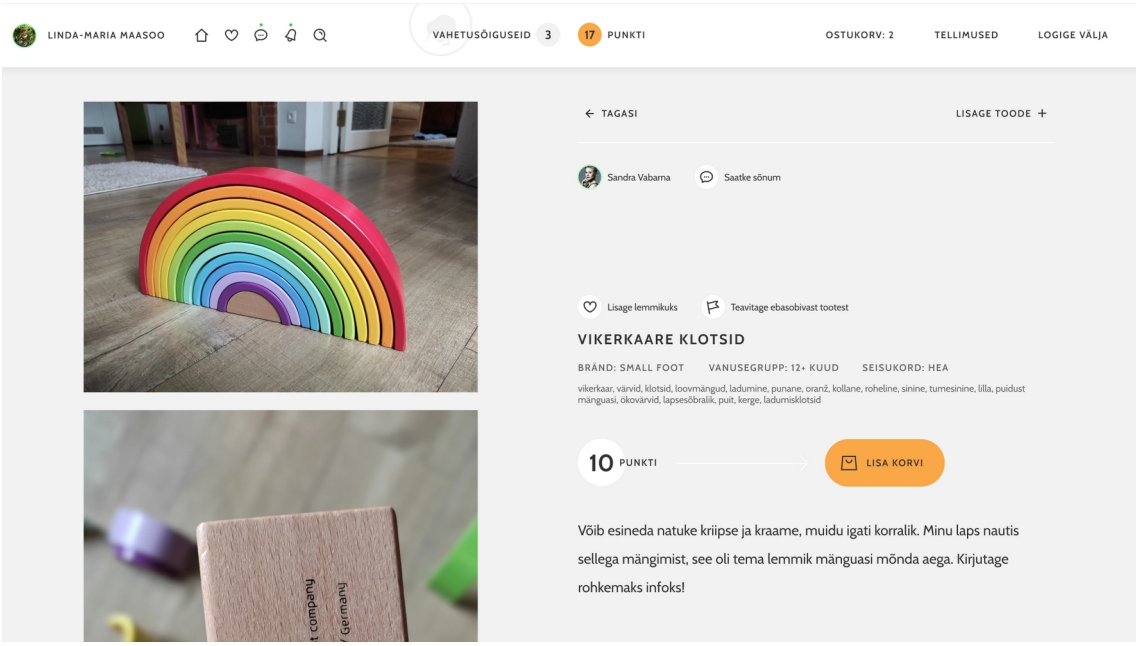
Tagarakenduse kood: https://bitbucket.org/PiiaMiia/toyswapper/src/master/.

Eesrakenduse kood: https://bitbucket.org/PiiaMiia/toyswapper-client/src/master/.

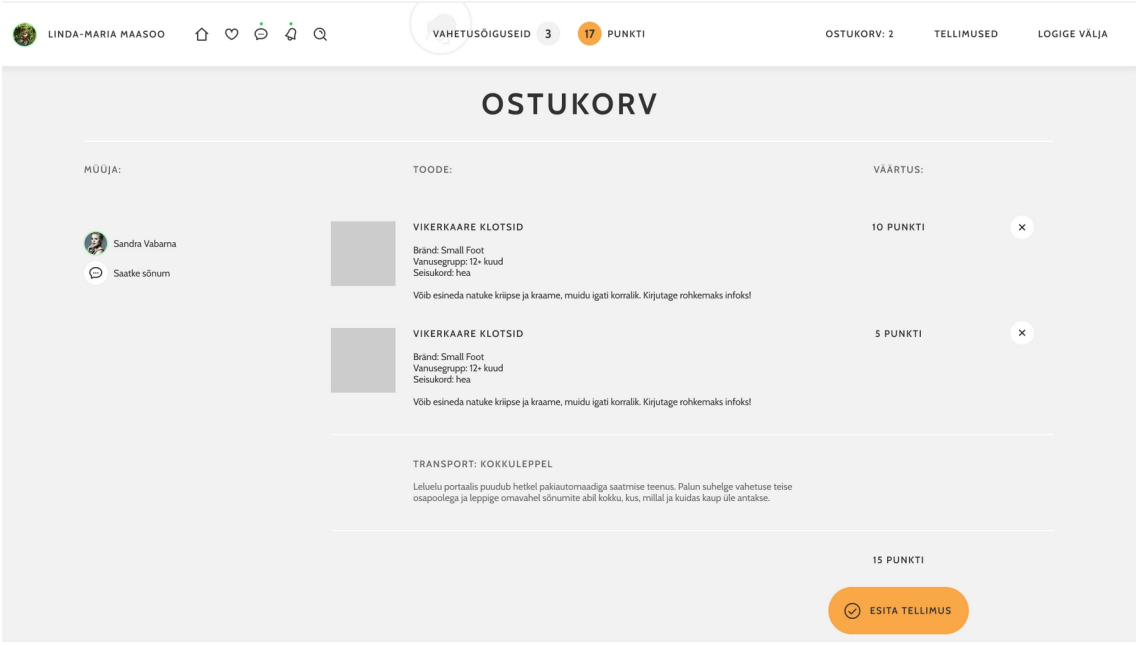
Lisa 3 – Mänguasjavahetuse keskkonna prototüüp



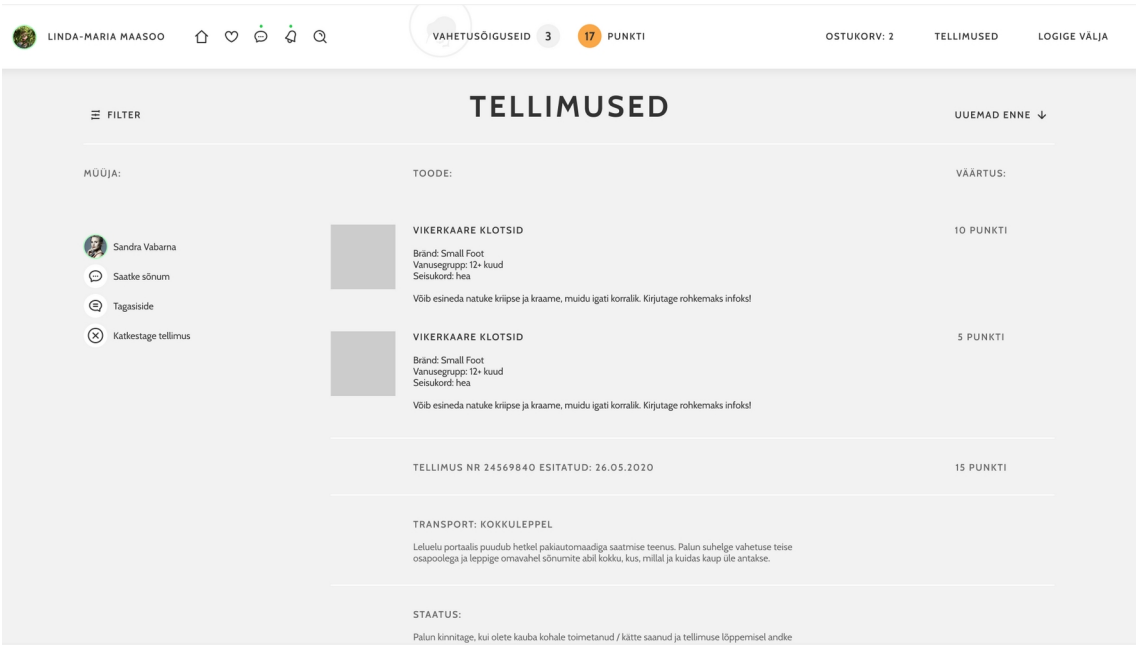
Pealehe vaade sisselogitud kasutajale



Lelu leht



Ostukorvi leht



Tellimuste leht

Lisa 4 – Kasutusjuhtumite spetsifikatsioon

ID	UC01
Nimi	Kasutajakonto loomine
Kirjeldus	Kasutaja peab saama portaalis registreerida endale konto
Käivitav sündmus	Kasutaja soovib registreerida konto
Aktorid	Tavakasutaja
Eeltingimus	• Kasutaja avab portaali veebilehitsejas ja vajutab „registreeru” nupule
Järeltingimus	• Kasuta konto on registreeritud
Tulemus	Kasutaja konto andmed on sisestatud andmebaasi
Põhistsenaarium	1. Kasutaja navigeerib veebilehitsejas portaali lehele 2. Kasutaja vajutab „looge konto” nupule 3. Kasutaja sisestab konto loomiseks vajalikud andmed: e-mail, salasõna, salasõna kinnitus 4. Kasutaja vajutab nupule „registreeru” 5. Kasutaja andmed saadetakse serverisse ja lisatakse andmebaasi 6. Kasutaja logitakse portaali sisse
Alternatiivstsenaarium	4a. Kasutaja navigeerib registreerimislehelt ära 4a1. Registreerimisprotsess tühistatakse 4a2. Kasutaja andmeid ei saadeta serverisse 5a. Kasutaja sisestatud andmed ei läbinud validatsiooni 5a1. Tehtud vigade kohta kuvatakse teavitused
Erindiststenaariumid	Käivitavad sündmused: • Kasutaja sisestatud andmed ei vasta validatsioonireeglitele • Kasutaja proovib registreerida e-mailiga, mis on juba registreeritud • Serveris juhtus viga

ID	UC02
Nimi	Sisse logimine
Kirjeldus	Kasutaja peab saama registreeritud kontoga sisse logida
Käivitav sündmus	Kasutaja soovib sisse logida
Aktorid	Tavakasutaja, administraator
Eeltingimus	• Kui kasutaja avab portaali veebilehitsejas, pole veel sisse logitud ja vajutab „logi sisse” nupule
Järeltingimus	• Kasutaja on portaali sisse logitud
Tulemus	Kasutajale genereeritakse kood, mis tõendab tema sisse logitud olekut ja kasutaja info salvestatakse veebilehitsejasse
Põhistsenaarium	1. Kasutaja navigeerib veebilehitsejas portaali lehele 2. Kasutaja vajutab „logi sisse” nupul 3. Kasutaja sisestab sisselogimisankeedi e-maili ja salasõna 4. Kasutaja vajutab „logi sisse” nupule 5. Sisestatud andmed saadetakse serverisse, kus kontrollitakse, kas see konto on olemas ja salasõna õige 6. Kasutaja logitakse portaali sisse
Alternatiivstsenaarium	4a. Kasutaja navigeerib sisselogimislehelt ära 4a1. Sisselogimisprotsess tühistatakse 4a2. Sisestatud andmeid ei saadeta serverisse 6a. Kasutaja sisestatud andmed ei ole õiged 6a1. Tehtud vigade kohta kuvatakse teavitus
Erindiststenaariumid	Käivitavad sündmused: • Kasutaja sisestatud e-mail ei ole registreeritud või salasõna ja e-mail ei sobi kokku • Serveris juhtus viga

ID	UC03
Nimi	Välja logimine
Kirjeldus	Kasutaja peab saama välja logida

Käivitav sündmus	Kasutaja soovib välja logida
Aktorid	Tavakasutaja, administraator
Eeltingimus	• Kui kasutaja avab portaali veebilehitsejas, on sisse logitud ja vajutab „logi välja” nupule
Järeltingimus	• Kasutaja on portaalist välja logitud
Tulemus	Veebilehitsejasse salvestatud kasutaja info kustutatakse
Põhistsenaarium	1. Kasutaja navigeerib veebilehitsejas portaali lehele 2. Kasutaja vajutab „logi välja” nupul 3. Kasutaja logitakse portaalist välja
Alternatiivstsenaarium	-
Erindiststenaariumid	-

ID	UC04
Nimi	Oma profiili haldamine
Kirjeldus	Kasutaja peab saama muutma oma profiili andmeid
Käivitav sündmus	Kasutaja soovib muuta oma profiili andmeid
Aktorid	Tavakasutaja, administraator
Eeltingimus	• Kui kasutaja avab portaali veebilehitsejas, on sisse logitud ja vajutab „muuda andmeid” nupule
Järeltingimus	• Kasutaja andmed on muudetud • Kasutaja uuenenud andmed kajastuvad igal pool, kus neid kuvatakse
Tulemus	Andmebaasi kirjet kasutaja kohta muudetakse
Põhistsenaarium	1. Kasutaja navigeerib veebilehitsejas portaali lehele 2. Kasutaja navigeerib oma profiili lehele 3. Kasutaja vajutab „muuda andmeid” nupul 4. Kasutaja uuendab andmeid 5. Kasutaja vajutab „salvesta” nupul 6. Kasutaja sisestatud muudetud andmed saadetakse serverisse 7. Kasutaja andmed on uuendatud
Alternatiivstsenaarium	2a. Kasutaja navigeerib portaali kodulehele

	6a. Kasutaja sisestatud andmed ei vasta validatsioonireeglitele 6a1. Tehtud vigade kohta kuvatakse teavitus
Erindiststenaariumid	Käivitavad sündmused: • Kasutaja sisestatud andmed ei vasta validatsioonireeglitele • Serveris juhtus viga

ID	UC05
Nimi	Lelu lisamine
Kirjeldus	Sisse logitud kasutaja peab saama lisada pakkumisse lelusid
Käivitav sündmus	Kasutaja soovib lisada uue lelu
Aktorid	Tavakasutaja, administraator
Eeltingimus	• Kui kasutaja avab portaali veebilehitsejas, on sisse logitud ja vajutab „lisa toode” nupule
Järeltingimus	• Uus lelu on portaali sisestatud • Uus lelu on aktiivses olekus
Tulemus	Andmebaasi lisatakse kirje uue lelu kohta
Põhistsenaarium	1. Kasutaja navigeerib veebilehitsejas portaali lehele 2. Kasutaja navigeerib portaali kodulehele 3. Kasutaja vajutab „lisa toode” nupul 4. Kasutaja valib oma seadmest tootepildid 5. Kasutaja täidab ankeedi, kus on kohustuslikud väljad: nimetus, punktid, asukoht, seisukord, ja mittekohustuslikud väljad: kirjeldus, vanusegrupid, kategooriad 6. Kasutaja vajutab „sisesta” nupul 7. Kasutaja sisestatud andmed saadetakse serverisse 8. Lelu on süsteemi lisatud
Alternatiivstsenaarium	4a/5a. Kasutaja navigeerib toote lisamise ankeedilt ära 4a1/5a1. Juba sisestatud andmed unustatakse 7a. Kasutaja sisestatud andmed ei vasta validatsioonireeglitele 7a1. Tehtud vigade kohta kuvatakse teavitus

Erindiststenaariumid	Käivitavad sündmused: • Kasutaja sisestatud andmed ei vasta validatsioonireeglitele • Serveris juhtus viga

ID	UC06
Nimi	Lelu kustutamine
Kirjeldus	Sisse logitud kasutaja peab saama oma lisatud lekusid kustutada
Käivitav sündmus	Kasutaja soovib kustutada enda lisatud lelu
Aktorid	Tavakasutaja, administraator
Eeltingimus	• Kui kasutaja avab portaali veebilehitsejas, on sisse logitud ja on oma lisatud lelu lehel, mis ei ole veel vahetatud või parasjagu vahetuses
Järeltingimus	• Lelu ei ole enam teistele kasutajatele nähtav, kuid on andmebaasis siiski olemas • Lelu staatust on võimalik muuta „aktiivseks”
Tulemus	Andmebaasi kirjet lelu kohta muudetakse, muutes lelu staatuse „mitteaktiivseks”
Põhistsenaarium	1. Kasutaja navigeerib portaalis mõne oma sisestatud lelu lehele. 2. Kasutaja vajutab nupul „deaktiveerige toode” 3. Serverisse saadetakse päring toote staatuse muutmise kohta 4. Kasutajale kuvatakse teavitus edukast deaktiveerimisest ja kuvatakse nupp, mis võimaldab lelu uuesti aktiveerimist
Alternatiivstsenaarium	4a. Serveris juhtus viga ja lelu ei õnnestunud deaktiveerida. 4a1. Kasutajale kuvatakse vastavasisuline teavitus
Erindiststenaariumid	Käivitavad sündmused: • Serveris juhtus viga

ID	UC07
Nimi	Lelude otsimine ja sirvimine

Kirjeldus	Iga kasutaja peab saama otsida ja sirvida lelusid, mis on aktiivses staatuses
Käivitav sündmus	Kasutaja soovib otsida ja sirvida lelusid
Aktorid	Tavakasutaja, administraator
Eeltingimus	• Kui kasutaja avab portaali veebilehitsejas ja navigeerib otsingulehele
Järeltingimus	• Kasutaja leiab oma valitud parameetrite järgi üles lelud, mis nendele parameetritele vastavad
Tulemus	Kasutajale kuvatakse kõik otsinguparameetritele vastavad lelud, mis on aktiivses staatuses
Põhistsenaarium	1. Kasutaja navigeerib otsingulehele 2. Kasutajale kuvatakse kõik aktiivsed lelud 3. Kasutaja valib vanusegrupi, kategooria ja/või seisukorra 4. Automaatselt toimub aktiivsete lelude filtreerimine 5. Otsinguparameetritele vastavad lelud kuvatakse kasutajale
Alternatiivstsenaarium	2a. Aktiivseid lelusid pole 2a1. Kasutajale kuvatakse teade, et hetkel pole lelusid 5b. Otsinguparameetritele vastavaid lelusid ei ole 5b1. Kasutajale kuvatakse teade, et hetkel pole otsinguparameetritele vastavaid lelusid
Erindiststenaariumid	Käivitavad sündmused: • Serveris juhtus viga

ID	UC08
Nimi	Lelu ostukorvi lisamine ja kustutamine
Kirjeldus	Sisselogitud kasutaja peab saama lisada lelusid ostukorvi
Käivitav sündmus	Kasutaja on leidnud meeldiva lelu ja tahab seda lisada ostukorvi
Aktorid	Tavakasutaja, administraator
Eeltingimus	• Kui kasutaja vajutab leitud lelu juures nuppu „lisa korvi”
Järeltingimus	• Kasutaja ostukorvi lisatud lelu kuvatakse ostukorvis
Tulemus	Ostukorvi lisatud lelud on grupeeritud müüjate järgi. Iga eseme kohta on välja toodud selle nimetus, kirjeldus, ja väärtus

	punktides. Iga grupi all on tellimuse kogumaksumus ja nupp „esita tellimus”. Grupeeritud tellimus ostukorvis moodustab edaspidi ühe tellimuse, mida käsitletakse ühe tervikuna.
Põhistsenaarium	1. Kasutaja on mõne lelu lehel, mis ei kuulu temale ja mis on aktiivses staatuses 2. Kasutaja vajutab nupul „lisa ostukorvi” 3. Lelu lisatakse ostukorvi sellesse tellimusse, kus on sama müüja 4. Kasutajale kuvatakse teavitus, et lelu on edukalt ostukorvi lisatud
Alternatiivstsenaarium	3a. Samalt müüjalt on juba ostukorvi lisatud mõni lelu 3a1. Lelu lisatakse ostukorvi samasse gruppi, kus on teised lelud samalt müüjalt 4b. Lelu on juba ostukorvi lisatud 4b1. Kasutajale kuvatakse teade, et lelu on juba ostukorvi lisatud
Erindiststenaariumid	-

ID	UC09
Nimi	Tellimuse esitamine teisele kasutajale
Kirjeldus	Sisselogitud kasutaja peab saama esitada ostukorvis olevate lelude kohta teisele kasutajale tellimuse
Käivitav sündmus	Kasutaja tahab ostukorvi lisatud lelude kohta esitada teisele kasutajale tellimuse
Aktorid	Tavakasutaja, administraator
Eeltingimus	• Kui kasutaja vajutab ostukorvis nuppu „esita tellimus”
Järeltingimus	• Müüjale esitatakse tellimus ja tellimuses sisalduvate lelude staatus muutub „mitteaktiivseks”
Tulemus	Müüjale saadetakse teavitus sissetulnud tellimuse kohta. Lelud, mis tellimuses sisalduvad, ei ole enam nähtavad teistele kasutajatele. Sissetulnud tellimus kuvatakse nii ostja kui ka müüja tellimuste lehel.
Põhistsenaarium	1. Kasutaja on oma ostukorvis, kus on tema poolt lisatud lelud. Lelud on grupeeritud müüja järgi erinevateks tellimusteks.

	2. Kasutaja vajutab nupul „esita tellimus” 3. Tellimus saadetakse serverisse, kus tehakse selle kohta andmebaasi vastavasisuline kirje 4. Tellimuses sisalduvate lelude staatus muudetakse „mitteaktiivseks” 5. Müüjale saadetakse tellimuse kohta teavitus 6. Kasutaja suunatakse edasi oma tellimuste lehele, kus ta näeb kõiki tellimusi, milles ta osaleb kas ostja või müüjana
Alternatiivstsenaarium	3a. Tellimust ei õnnestunud esitada kas ebapiisava punktisaldo, serveri vea või vahepeal muutunud staatuste pärast 3a1. Kasutajale kuvatakse vastavasisuline teavitus
Erindiststenaariumid	Käivitavad sündmused: • Serveris juhtus viga

ID	UC10
Nimi	Tellimuse staatuse muutustest teavituse saamine
Kirjeldus	Sisselogitud kasutaja peab saama teavitusi temaga seotud tellimuste staatuste muutumiste kohta
Käivitav sündmus	Kasutajaga seotud tellimuse staatus muutus
Aktorid	Tavakasutaja, administraator
Eeltingimus	• Vahetuse teine osapool muutis vahetuse staatust
Järeltingimus	• Kasutaja saab temaga seotud tellimuse staatuse kohta teavituse
Tulemus	Kasutajale saab teada, et temaga seotud vahetuse staatus on muutunud
Põhistsenaarium	1. Kasutaja esitab ostukorvist tellimuse, kinnitab tellimuste lehel tellimuse teele panemist, tellimuse kättesaamist või tühistab tellimuse
Alternatiivstsenaarium	-
Erindiststenaariumid	-

ID	UC11
Nimi	Piletite ja punktide ostmine
Kirjeldus	Sisselogitud kasutaja peab saama soovi korral osta pileteid ja punkte, mis võimaldavad tal portaalis vahetusi läbi viia
Käivitav sündmus	Kasutaja soovib osta pileteid ja/või punkte
Aktorid	Tavakasutaja, administraator
Eeltingimus	• Sisselogitud kasutaja navigeerib piletite ja punktide ostmise lehele
Järeltingimus	• Sisselogitud kasutaja kontole lisatakse ostetud piletid ja/või punktid
Tulemus	Kasutaja pileti ja punktisaldo muutub.
Põhistsenaarium	1. Sisselogitud kasutaja navigeerib punktide ja/või piletite ostmise lehele 2. Kasutaja valib punktid ja/või piletid, mida osta soovib 3. Kasutaja vajutab nupul „osta” 4. Kasutaja suunatakse pangalingile 5. Kasutaja viib ostmise lõpule 6. Kasutaja suunatakse tagasi punktide ja/või piletite ostmise lehele 7. Kasutaja punkti ja/või piletisaldole lisatakse ostetud punktid ja/või piletid
Alternatiivstsenaarium	5a. Kasutajal pole piisavalt vahendeid, et ostmine lõpule viia 5a1. Kasutaja suunatakse tagasi punktide ja/või piletite ostmise lehele 5a2. Kasutajale kuvatakse teavitus, et ostmine ebaõnnestus
Erindiststenaariumid	Käivitavad sündmused: • Pangalingil juhtus viga • Serveris juhtus viga

ID	UC12
Nimi	Lelude lemmikuks lisamine
Kirjeldus	Sisselogitud kasutaja peab saama soovi korral lisada meeldivaid lelusid lemmikutesse, et tal oleks võimalik neid hiljem lihtsamini

	üles leida
Käivitav sündmus	Kasutaja soovib lisada lelu lemmikutesse
Aktorid	Tavakasutaja, administraator
Eeltingimus	• Sisselogitud kasutaja navigeerib lelu lehele
Järeltingimus	• Lelu on lisatud kasutaja lemmikutesse
Tulemus	Kasutaja lemmikute hulgas on lemmikuks märgitud lelu
Põhistsenaarium	1. Sisselogitud kasutaja navigeerib lelu lehele 2. Kasutaja vajutab nupul „lisa lemmikuks” 3. Kasutajale kuvatakse teavitus, et lelu on lemmikutesse lisatud
Alternatiivstsenaarium	3a. Serveris toimub viga ja lelu ei õnnestunud lemmikutesse lisada 3a1. Kasutajale kuvatakse teavitus, et lelu ei õnnestunud lemmikutesse lisada
Erindiststenaariumid	Käivitavad sündmused: • Serveris juhtus viga

ID	UC13
Nimi	Teistele kasutajatele sõnumite saatmine
Kirjeldus	Sisselogitud kasutaja peab saama soovi korral saata teistele kasutajatele sõnumeid ja käimasolevaid vestlusi lugeda
Käivitav sündmus	Kasutaja soovib saata teisele kasutajale sõnumi
Aktorid	Tavakasutaja, administraator
Eeltingimus	• Sisselogitud kasutaja vajutab nupule „saada sõnum”
Järeltingimus	• Saadetud sõnum ilmub sisselogitud kasutaja ja teise kasutaja vestlusesse
Tulemus	Sisselogitud kasutaja ja teise kasutaja vahel toimub vestlus
Põhistsenaarium	1. Sisselogitud kasutaja navigeerib teise kasutaja lehele 2. Kasutaja vajutab nupul „saada sõnum” 3. Kasutaja suunatakse tema ja teise kasutaja vestluse lehele 4. Kasutaja sisestab sõnumi teksti sisestamise alasse

	5. Kasutaja vajutab nupul „saada” 6. Sõnum salvestatakse andmebaasi koos viitega saata ja saaja kohta 7. Saadetud sõnum ilmub koheselt vestluse lehele
Alternatiivstsenaarium	6a. Serveris juhtus sõnumi salvestamisel viga 6a1. Kasutajale kuvatakse veateade, et sõnumit ei õnnestunud saata
Erindiststenaariumid	Käivitavad sündmused: • Serveris juhtus viga

ID	UC14
Nimi	Otsinguparameetrite haldamine
Kirjeldus	Sisselogitud administraatori õigustega kasutaja peab saama lisada ja muuta otsinguparameetreid
Käivitav sündmus	Administraator soovib muuta otsinguparameetreid
Aktorid	Administraator
Eeltingimus	• Administraator navigeerib vaid administraatorile nähtavale lehele, kus saab otsnguparameetreid hallata
Järeltingimus	• Otsinguparameetrid muutuvad
Tulemus	Otsinguparameetrite nimekiri, mida kasutajad saavad lelu lisamisel määrata, muutub vastavalt tehtud muudatustele
Põhistsenaarium	1. Sisselogitud administraatori õigustega kasutaja navigeerib navigatsioonimenüüs lehele, kus saab hallata otsinguparameetreid 2. Administraator näeb valikut, kus ta saab valida, milliseid otsinguparameetreid ta muuta soovib 3. Administraator valib otsinguparameetrid, mida ta muuta soovib 4. Administraator näeb ankeeti, kust ta saab lisada uusi parameetreid ja nimekirja hetkestest parameetritest. Iga otsinguparameetri juures on vastavalt parameetri staatusele nupp „aktiveeri” või „deaktiveeri” 5. Administraator vajutab nupule „aktiveeri” või „deaktiveeri” 6. Otsinguparameetri staatus muutub

Alternatiivstsenaarium	5a. Administraator sisestab ankeeti uue otsinguparameetri nimetuse ja kirjelduse ja valib, kas parameeter on aktiivne või mitte 5a1. Administraator vajutab nupul „lisa” 5a2. Otsinguparameeter lisatakse andmebaasi 5a2. Lisatud otsinguparameeter ilmub otsinguparameetrite nimekirja
Erindiststenaariumid	Käivitavad sündmused: • Serveris juhtus viga

ID	UC15
Nimi	Vahetusprotsesside haldamine
Kirjeldus	Sisselogitud kasutaja peab saama hallata temaga seotud käimasolevaid tellimusi. Administraatori õigustega kasutaja peab saama hallata kõiki süsteemis käimasolevaid telimusi.
Käivitav sündmus	Kasutaja soovib muuta käimasoleva tellimuse staatust
Aktorid	Tavakasutaja, administraator
Eeltingimus	• Tavakasutaja navigeerib oma tellimuste lehele • Administraator navigeerib vaid administraatorile nähtavale lehele, kus saab tellimusi hallata
Järeltingimus	• Tellimuse ja sellega seotud lelude staatused muutuvad
Tulemus	Tellimuse staatus muutub ja kõik asjaosalised saavad selle kohta vastavasisulise teavituse
Põhistsenaarium	1. Sisselogitud kasutaja navigeerib oma tellimuste lehele 2. Kasutaja näeb nimekirja temaga seotud tellimustest ja nende staatustest 3. Vastavalt tellimuse staatusele näeb kasutaja nuppu, mis võimaldab tal: tellimus tühistada, kinnitada tellimuse teelesaatmist, kinnitada tellimuse kättesaamist 4. Kasutaja vajutab kuvatud nupul 5. Tellimuse staatus muutub 6. Vahetuse teine osapool saab tellimuse staatuse muutumise kohta teavituse
Alternatiivstsenaarium	1a. Administraator navigeerib navigatsioonimenüüs vaid administraatori õigustega kasutajale nähtavale lehele, kus

	saab hallata kõiki süsteemis käimasolevaid tellimusi, mis ei ole veel lõpule viidud. 1a1. Administraator saab otsida tellimusi tellimuste osapoolte ja staatuste järgi 1a2. Administraator valib tellimuse, mida ta muuta soovib 1a3. Administraator muudab tellimuse staatust 1a4. Tellimusega seotud kasutajad saavad staatuse muutumise kohta vastavasisulise teavituse
Erindiststenaariumid	Käivitavad sündmused: • Serveris juhtus viga

ID	UC16
Nimi	Kasutajate staatuste muutmine
Kirjeldus	Sisselogitud administraatori õigustega kasutaja peab saama muuta teiste kasutajate staatust
Käivitav sündmus	Administraator soovib muuta teise kasutaja staatust
Aktorid	Administraator
Eeltingimus	• Administraator navigeerib vaid administraatorile nähtavale lehele, kus saab teisi kasutajaid hallata
Järeltingimus	• Kasutaja staatus muutub
Tulemus	Kasutaja staatus muutub ja ta saab selle kohta vastavasisulise teavituse
Põhistsenaarium	1. Administraator navigeerib lehele, kus ta saab teisi kasutajaid hallata 2. Administraator saab otsida kasutajaid e-maili, eesnime, perenime ja süsteemi poolt määratud id järgi 3. Administraator valib kasutaja, kelle staatust ta muuta soovib 4. Administraator saab vastavalt kasutaja hetkestaatusele muuta tema staatuse kas aktiivseks või mitteaktiivseks 5. Administraator muudab kasutaja staatust 6. Kasutaja saab muudetud staatuse kohta teavituse
Alternatiivstsenaarium	-

Erindiststenaariumid	Käivitavad sündmused: • Serveris juhtus viga

ID	UC17
Nimi	Kasutajate rolli muutmine
Kirjeldus	Sisselogitud administraatori õigustega kasutaja peab saama muuta teiste kasutajate rolli
Käivitav sündmus	Administraator soovib muuta teise kasutaja rolli
Aktorid	Administraator
Eeltingimus	• Administraator navigeerib vaid administraatorile nähtavale lehele, kus saab teisi kasutajaid hallata
Järeltingimus	• Kasutaja roll muutub
Tulemus	Kasutaja roll muutub ja ta saab selle kohta vastavasisulise teavituse
Põhistsenaarium	1. Administraator navigeerib lehele, kus ta saab teisi kasutajaid hallata 2. Administraator saab otsida kasutajaid e-maili, eesnime, perenime ja süsteemi poolt määratud id järgi 3. Administraator valib kasutaja, kelle rolli ta muuta soovib 4. Administraator muudab kasutaja rolli 5. Kasutaja saab muudetud rolli kohta teavituse
Alternatiivstsenaarium	-
Erindiststenaariumid	Käivitavad sündmused: • Serveris juhtus viga

ID	UC18
Nimi	Kaebuste ülevaatamine ja haldamine
Kirjeldus	Sisselogitud administraatori õigustega kasutaja peab saama näha ja hallata kasutajate poolt tehtud kaebusi
Käivitav sündmus	Administraator soovib näha ja hallata kasutajate poolt tehtud kaebusi
Aktorid	Administraator

Eeltingimus	• Administraator navigeerib vaid administraatorile nähtavale lehele, kus saab hallata kasutajate poolt tehtud kaebusi
Järeltingimus	• Administraator näeb kaebust ja märgib selle ülevaadatuks
Tulemus	Administraator näeb sissetulnud kaebusi ja otsustab, kuidas neid lahendada.
Põhistsenaarium	1. Administraator navigeerib lehele, kus ta saab sissetulnud kaebusi hallata 2. Administraator näeb esmajärjekorras kaebusi, mida pole veel ülevaadatuks märgitud, ja seejärel kaebusi, mis on juba ülevaadatuks märgitud 3. Administraator teeb nähtud info põhjal järeldused, mida ette tuleb võtta ja teeb vajalikud toimingud 4. Administraator sisestab kaebuse juurde otsuse ja märgib kaebuse ülevaadatuks 5. Kaebuse esitanud kasutaja saab selle kohta vastavasisulise teavituse, kus on kirjas sisestatud otsus
Alternatiivstsenaarium	-
Erindiststenaariumid	Käivitavad sündmused: • Serveris juhtus viga

ID	UC19
Nimi	Raportite genereerimine
Kirjeldus	Sisselogitud administraatori õigustega kasutaja peab saama esitada serverisse päringuid, mille põhjal genereeritakse raport, mida on võimalik exceli failina alla laadida. Samuti peavad näha olema raportid, mis on juba genereeritud.
Käivitav sündmus	Administraator soovib genereerida uue raporti andmebaasi andmete põhjal
Aktorid	Administraator
Eeltingimus	• Administraator navigeerib vaid administraatorile nähtavale lehele, kus saab näha ja genereerida raporteid
Järeltingimus	• Administraator laeb alla genereeritud raporti
Tulemus	Administraatori esitatud päringu põhjal on genereeritud raport, mis on nähtav genereeritud raportite tabelis ja mida saab alla

	laadida
Põhistsenaarium	1. Administraator navigeerib lehele, kus ta saab näha ja genereerida raporteid 2. Administraator valib, millist raportit ja mis perioodi kohta ta genereerida soovib 3. Administraator vajutab nupule „genereeri raport” 4. Serverisse saadetakse päring valitud parameetritega raporti genereerimiseks 5. Administraatorile antakse teada, et raportit genereeritakse ja ta võib soovi korral lehelt lahkuda ja hiljem tagasi tulla 6. Kui raport on genereeritud, ilmub see genereeritud raportite tabelisse 7. Administraator saab sealt tabelist genereeritud raporti nime peal vajutades selle alla laadida
Alternatiivstsenaarium	4a. Raportit ei ole võimalik mingil põhjusel genereerida 4a1. Administraatorile kuvatakse teavitus koos põhjendustega, et raportit pole võimalik genereerida
Erindiststenaariumid	Käivitavad sündmused: • Serveris juhtus viga

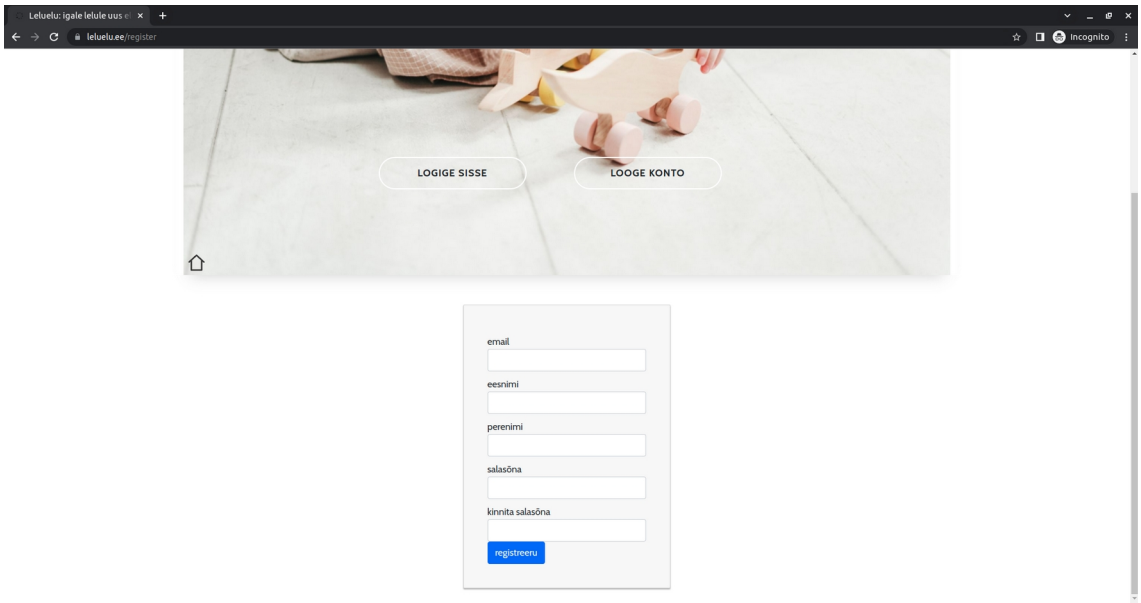
ID	UC20
Nimi	Lelu ebasobivaks märkimine
Kirjeldus	Igal sisselogitud kasutajal peab olema võimalus raporteerida ebasobivaid tooteid
Käivitav sündmus	Sisse logitud kasutaja näeb pakkumiste hulgas ebasobivat toodet ja tahab seda raporteerida
Aktorid	Tavakasutaja, administraator
Eeltingimus	• Sisselogitud kasutaja on ebasobiva toote lehel
Järeltingimus	• Süsteemi saadetakse kaebus ebasobiva toote kohta
Tulemus	Ebasobivad tooted salvestatakse andmebaasi. Administraatorid näevad administraatoritele mõeldud lehel kaebusi.
Põhistsenaarium	1. Kasutaja märkab pakkumiste hulgas ebasobivat toodet 2. Kasutaja navigeerib toote lehele

	3. Kasutaja vajutab nupul „teata ebasobivast tootest” 4. Kasutajale kuvatakse teavitus, et tema kaebus on esitatud ja sellega tegeletakse esimesel võimalusel
Alternatiivstsenaarium	4a. Serveri vea tõttu polnud võimalik kaebust edastada 4a1. Kasutajale kuvatakse teavitus, et tema kaebust polnud võimalik hetkel esitada ja ta peaks mingi hetk uuesti proovima
Erindiststenaariumid	Käivitavad sündmused: • Serveris juhtus viga

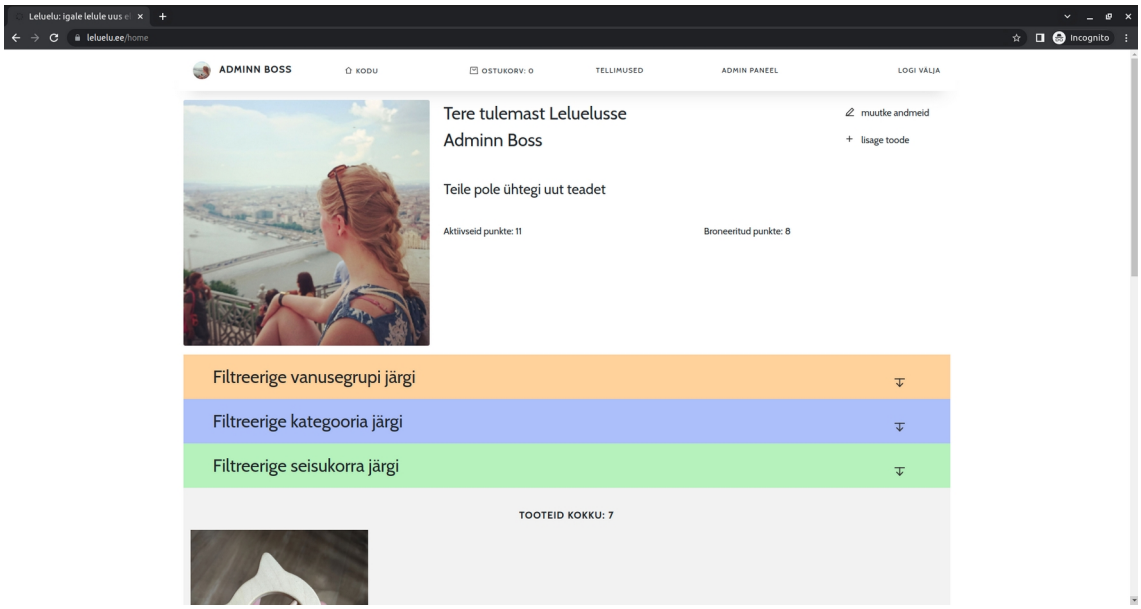
ID	UC21
Nimi	Teisele kasutajale tagasiside andmine
Kirjeldus	Igal sisselogitud kasutajal peab olema võimalus jätta tagasiside kasutajale, kellega on oldud vahetusprotsessis. Tagasiside saab anda peale vahetusprotsessi lõppemist – nii eduka lõpu kui ka tühistamise korral.
Käivitav sündmus	Sisse logitud kasutaja soovib jätta tagasisidet teisele kasutajale, kellega on osaletud vahetusprotsessis
Aktorid	Tavakasutaja, administraator
Eeltingimus	• Vahetusprotsess on edukalt lõppenud • Vahetusprotsess on lõppenud kummagi osapoole tühistamisega
Järeltingimus	• Tagasiside kuvatakse tagasiside saaja lehel
Tulemus	Tagasiside saajale on antud punktiline hinnang ja jäetud soovi korral ka tagasiside kirjalikul kujul. Saadud punktiliste hinnangute põhjal arvutatakse kasutaja skoor.
Põhistsenaarium	1. Kasutaja läheb tellimuste lehele ja kinnitab, et on tellimuse kätte saanud 2. Mõlemale vahetuses osalenud kasutajale kuvatakse tellimuse juures võimalus anda vahetuse teisele osapoolele punktiline hinnang ja jätta kirjalik tagasiside 3. Kasutaja kasutab võimalust ja annab hinde ja kirjutab tagasiside 4. Kasutaja vajutab nupule „esita tagasiside” 5. Kasutajale kuvatakse teade, et tagasiside on edukalt saadetud

	6. Punktilise hinnagu põhjal arvutatakse hinde saaja skoor ümber 7. Tagasiside saaja lehel kuvatakse kirjalik tagasiside ja kuvatakse kasutaja skoor
Alternatiivstsenaarium	1a. Kasutaja läheb tellimuste lehele ja tühistab tellimuse 3a. Kasutaja annab hinde, kuid ei kirjuta tagasisidet 5a. Serveri vea tõttu ei õnnestunud tagasisidet saata. 5a1. Kasutajale kuvatakse teade, et hetkel ei õnnestunud tagasisidet saata ja ta võiks hiljem uuesti proovida
Erindiststenaariumid	Käivitavad sündmused: • Serveris juhtus viga

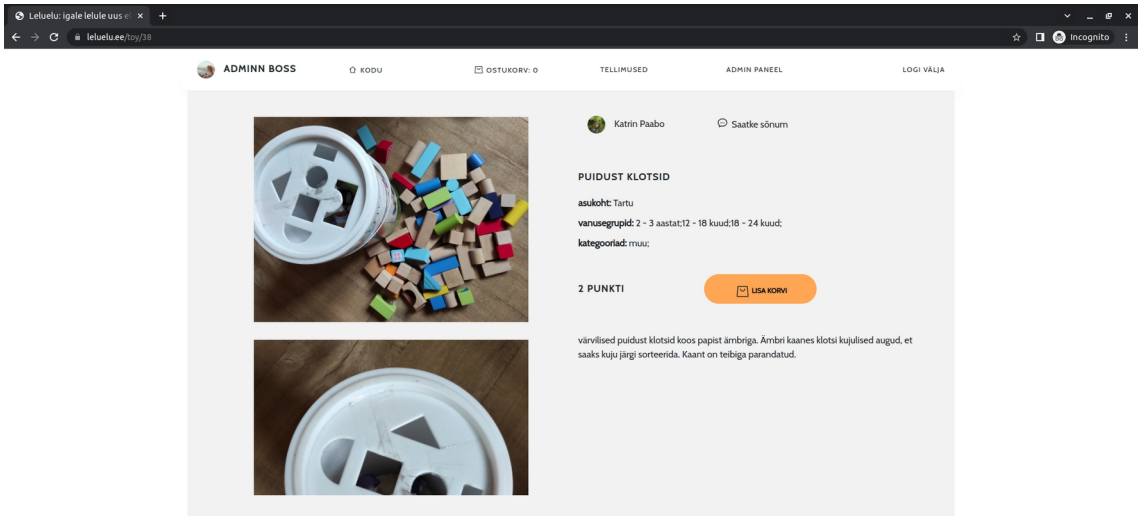
Lisa 5 – Loodud rakenduse vaated



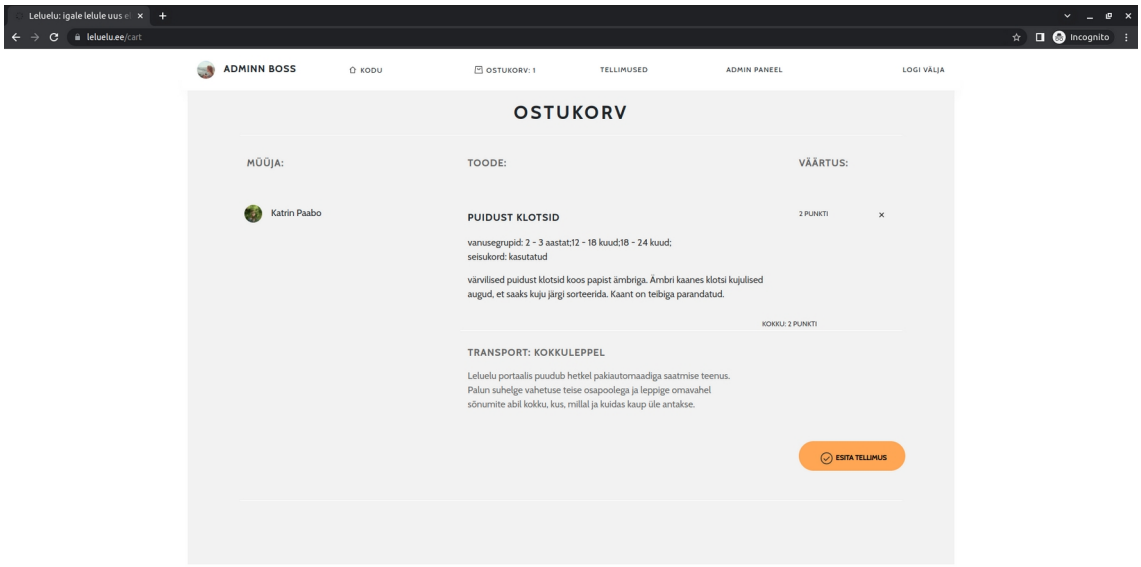
Registreerimisankeet



Kasutaja koduleht

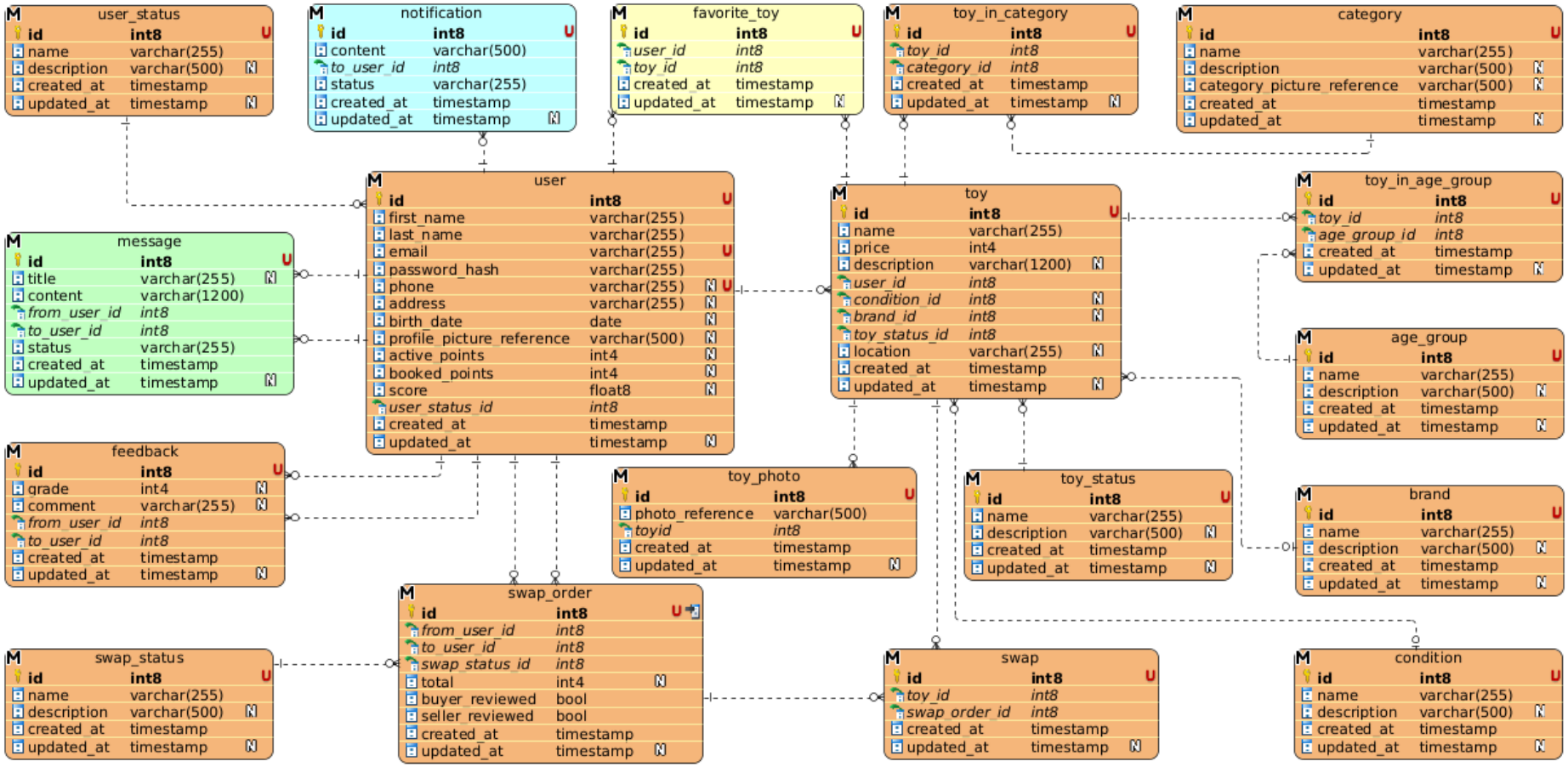


Lelu leht

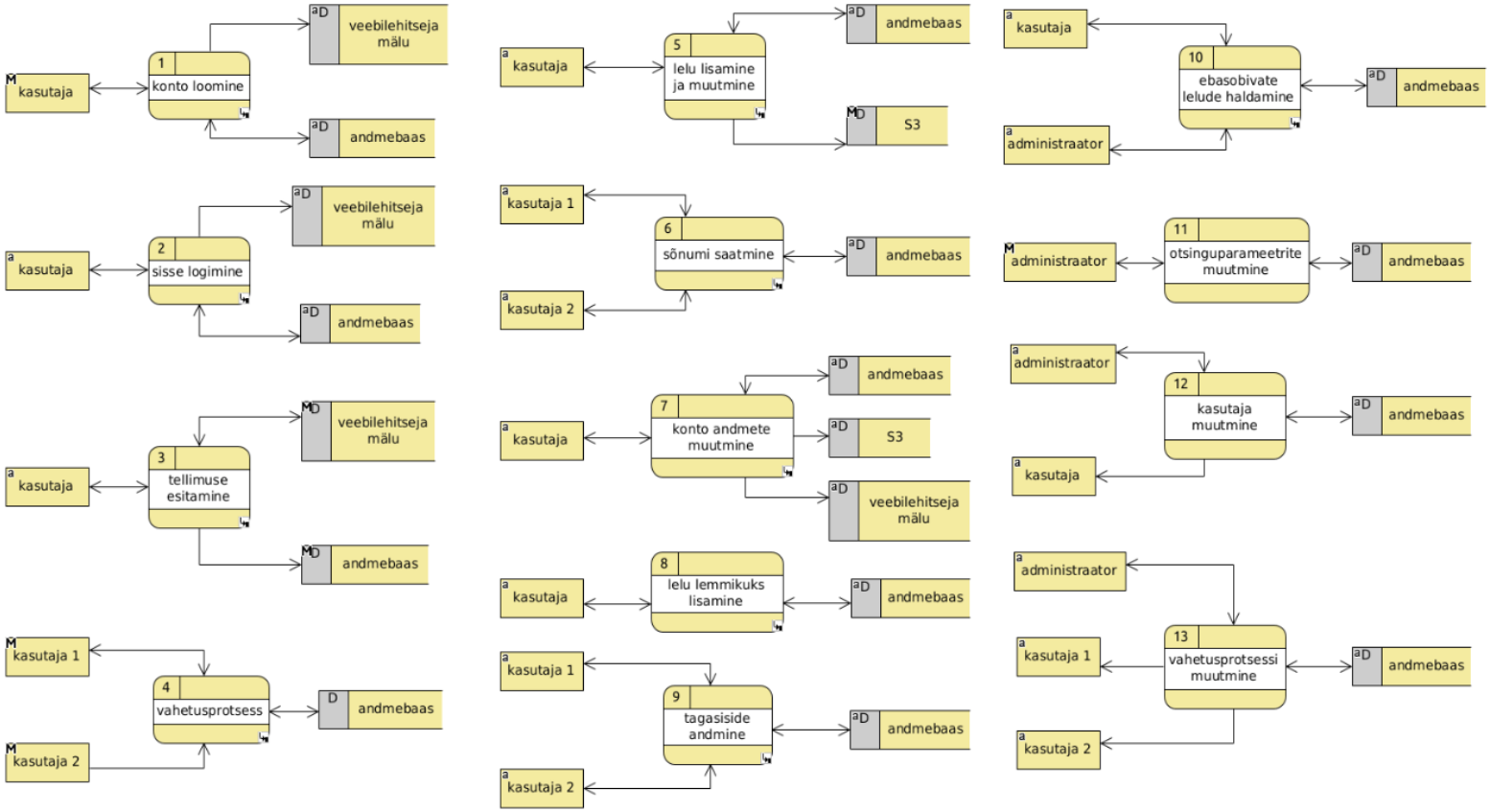


Ostukorv

Lisa 6 – Olem-suhte diagramm



Lisa 7 – Tase 1 andmevoo diagrammid



Lisa 8 – Olekumasina diagramm