+
TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Nikita Saprõkin 195366IADB

Talutoodete turu rakenduse arendus

Bakalaureusetöö

Juhendaja: Meelis Antoi
Magistrikraad

Tallinn 2023

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: Nikita Saprõkin

24.04.2023

Annotatsioon

Käesoleva bakalaureusetöö eesmärk on luua rakendus talutoitude tellimiseks. Rakendus peab võimaldama kasutajatel luua oma talupoed, teenindada tellimusi ja tellida talutoite rakenduse abil. Lisaks põhifunktsionaalsusele peab rakendus võimaldama erinevate ärirollide registreerimist, Eestis toodetud talutoitude kaupade loomist, klientidele oma talutoodete ostukorvi kogumist ning näitama statistikat tellimuste teenindamise, saldo ja ostude ajaloo kohta. Seni Eesti rakendusturul puudub lahendus, mis pakuks kõiki neid funktsioone.

Arendusprotsessi käigus luuakse serveripoolne ja kasutajapoolne rakenduse MVP (minimaalne elujõuline toode).

Enne lahenduse arendamist analüüsitakse sarnaseid turul olevaid lahendusi, tuues välja nende nõrgemad kohad. Sellele järgnevalt analüüsib autor erinevaid tehnoloogiaid, mida süsteemi loomisel kasutada, tuues välja iga tehnoloogia tugevused ja nõrkused. Süsteemi arendamise käigus kirjeldatakse erinevaid kasutatavaid tehnoloogiaid ja rakenduse arhitektuuri.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 57 leheküljel, 7 peatükki, 15 joonist, 0 tabelit.

Abstract

Development of Farm Market Application

The purpose of this bachelor thesis is to create an application for ordering farm food. The application will provide an order service for a farm shop, enabling users to conveniently order food directly from the farm. Additionally, the application must incorporate several key features. Users should be able to register with different business roles, including the ability to create farm food products and specify Estonia as the origin. Customers should be able to add farm food items to their shopping basket, and the application should display farm food order service statistics, balance information, and purchase histories. Front-end application part must be done as mobile application. Until now, there was no solution in the Estonian application market that would offer all this functionality.

During the development process, server-side applications and front-end applications are analyzed to create a minimum viable product.

Before developing a solution, similar solutions on the market are analyzed, highlighting their weakest points. After that, the author analyzes the different technologies to be used in the creation of the system, highlighting the weak and strong points of each technology. During the development of the system, the different technologies used and the application architecture are described.

The thesis is in Estonian and contains 57 pages of text, 7 chapters, 15 figures, 0 tables.

Lühendite ja mõistete sõnastik

API	*Application Programming Interface* – rakendustarkvara liides
Back-end	Serveripoolne rakendus
COBRA	*Common Object Request Broker Architecture* – on tarkvaraarhitektuur, mis võimaldab erinevate rakenduste ja süsteemide vahelist suhtlust läbi erinevate platvormide ja programmeerimiskeelte.
DCOM	*Distributed Component Object Model* – hajutatud komponentobjektide mudel, mis võimaldab objektide suhtlust ja andmeedastust erinevate arvutite vahel võrgukeskkonnas.
ERD	*Entity Relationship Diagramm* – diagramm, mis näitab ära andmebaasimudelid ja nende suhted.
Front-end	Kasutajapoolne rakendus.
GPL	*GNU* üldine avalik litsents – tasuta tarkvaralitsents
HTML	*HyperText Markup Language* – hüperteksti märgistuskeel
JSON	*JavaScript Object Notation* – andmevahetusvorming
JVM	*Java Virtual Machine* – Java virtuaalmasin
LAMP	Tähistab *Linux, Apache, MySQL* ja *PHP* ning viitab tavalisele tarkvarapakendile, mida kasutatakse veebirakenduste loomiseks ja käitamiseks.
LGPL	*GNU Lesser General Public License* – on tasuta tarkvaralitsents
MVC	*Model view controller* – mudel-vaade-kotroller – arhitektuur, mida kasutatakse rakendustes.
MVP	*Minimum Viable Product* – Minimaalne elujõuline toode
NuGet	*NuGet* –tarkvara paketihaldussüsteem
REST	*Representational State Transfer* – tarkvara arhitektuur laad, mis paneb paika kindlad piiri veebirakenduse jaoks.
SDK	*Software development kits* – on tarkvarakomplekt, mis sisaldab vahendeid, teegid, dokumentatsiooni ja teised abivahendid rakenduse arendamiseks
SOAP	*Simple Object Access Protocol* – rakenduste suhtlusprotokoll
TXT	Tekstifail

WSDL	*Web Services Description Language* – XML-põhine keel, mida kasutatakse veebiteenuste kirjeldamiseks ja nendega suhtlemiseks.
XML	*Extensible Markup Language* – laiendatav märgistuskeel

Sisukord

Jooniste loetelu

1 Sissejuhatus

Tänases globaliseeritud mailmaas inimese elu sai kiire ning nende suhtlus muutus ebaehtseks. Kaasaegsed linnas elavaid inimesed kaotasid sidemeid talumehega kes toovad kohalik tooted. Kliendile on raske leida soovitud talu toodete pakkumist, talumehele on raske oma tootele stabiilse nõudlust leida. Inimesed soovivad osta ökoloogilaseid tooted eesti taludest nagu kartulit, sibulat, mee jne. Isegi kui taludel on hea pakkumis, on üsna raske nõudlust leida. Ostjate probleem selles, et kui sa ei ela maal või sul pole tuttavaid talumehed siis on raske leida hea talu, mis pakub lemmikud tooted.

Lõputöö autor analüüsib probleemi ja uurib sarnased eesti rakendusesi turul oleva lahendusi. Probleemi lahenduseks on välja pakutud mobiilirakendus, mis aitab seadustada sidemeid talumehe ja kliendi vahel. Rakenduse kaudu kliendina sisse-loginud kasutaja saab mugavalt tellida talutoitu, äri-kontona sisse-loginud kasutaja saab mugavalt oma talutooted müüja.

2 Probleemi ülevaade

Tänases globaliseerunud maailmas on inimeste elutempo muutunud kiireks ning nende suhtlus on muutunud ebaautentseks. Linnas elavad inimesed on taludest väga kaugel ning nende kogu elu toimub linnas. Enamik neist ostab kogu toidukauba suurtest kauplustest. Selle tulemusena on kohaliku talumehe nõudlus vähenenud ja linnainimesed ei tea enam, kuidas ja kust nende ostetud toit on kasvatatud. Kui tooted ostetakse otse talumeestelt, saab klient olla kindel nende kvaliteedis. Kui kliendile meeldib konkreetne talu, saab ta toetada seda, ostes nende tooteid oma raha eest.

Talumehed tihti elavad kaugelt linnast ja neil on raske hoia oma poodi, oma talutoode müümiseks. Selle tõttu nad tihti müüvad seda suur kauplustel, kus nendega kasvatud tooted, mis ostjatele need muutuvad saamaks mis on teisest riigist või suure firma toodetud tooted. Selle tõttu nende kaubad saavad lihtsam asendatud suurega firmadega mis saavad pakkuda võib olla kehvaim kvaliteetsega, aga odavam toode.

Klientidele on raske leida soovitud talu toodete pakkumist, talumehele on raske oma tootele stabiilse nõudlust leida. Talumehed elavad sageli kaugel linnast ja neil on raske säilitada oma poe tegevust ning müüa oma talutooteid. Selle tõttu nad tihti müüvad oma tooteid suurtes kauplustes, kus need võivad seguneda teiste riikide või suurte firmade toodetega, muutes ostjate jaoks keeruliseks tuvastada, mis on kohalikust talust pärit toode. Seetõttu on nende tooted sageli hõlpsasti asendatavad suurte firmade poolt pakutavate odavamate, kuid võib-olla madalama kvaliteediga toodetega.

Klientidel on raske leida soovitud talutoodete valikut ning talumeestel on keeruline leida stabiilset nõudlust oma toodetele.

2.1 Talutoote turu eksisteerivad lähendused

Käesolevas lõigus kirjeldatakse juba olemasolevaid lahendusi, mis pakuvad talutoodete müümise ja tellimise võimalusi.

2.1.1 Talutoit.ee

Talutoit on Eestimaa Talupidajate Keskliiduga loodud veebirakendus, mille kaudu saab osta või müüa Eestis kasvatatud talutoitu. Veebileht asub aadressil "pood.talutoit.ee". Veebirakenduses on võimalik luua oma konto, kus saab jälgida oma oste ja tellimusi, sisestada arve- ja tarneaadresse ning salvestada soovikorvi. Veebilehe esilehel on loodud mugav otsimis- ja filtreerimisfunktsioon, mis võimaldab otsida tooteid nii kategooria kui ka talu nime järgi. Iga toote juures on välja toodud päritolu ning sellega seotud talu informatsioon ja kontaktid. Veebilehe turu omanik saab lisada toodete ja talude kirjeldusi, mis aitavad klientidel tutvuda talu ajalooga ja toodete kirjeldustega. Toote kirjelduste all saavad kasutajad jagada oma arvustusi ostetud kaupade kohta. Toote ostmiseks valib kasutaja soovitud toote ja lisab selle korvi. Ostukorvi vaates küsitakse valitud tarne meetodi ja makseandmete valikut. Ostetud kaupade saamiseks on võimalik need ise üles korjata või tellida kulleriga koju. Tellimus saadetakse talu omanikule, kes pärast seda aktsepteerib tellimuse ning seejärel saab klient oma valitud viisil ostud kätte.

Veebilehel registreerudes saab kasutaja ennast talu omanikuna registreerida ja luua oma talupoe. Registreerimisel küsitakse taluomaniku nime, talu nime, talu aadressi ja kontaktandmeid. Pärast aktsepteerimist saab talu omanik oma tooteid lisada ja müüma hakata.

Allpool kirjeldatud lahendus on mugav ja pakub kasutajatele tuge talutoodete ostmisel ja müümisel. Suurimaks puuduseks selles lahenduses on mobiilirakenduse puudumine. Tänapäeval ei pruugi kõigil olla arvutit, kuid peaaegu kõik kasutavad mobiiltelefone. Kuigi veebirakendus on hästi kohandatud mobiilseadmetele ja see ei tekita probleeme ostjatele, kes kasutavad rakendust harva, näiteks üks kord kahe nädala jooksul, on taluomanike jaoks, kes kasutavad rakendust igapäevaselt, mugav mobiilirakendus kriitiliselt vajalik. Funktsioonid nagu kontole salvestatud tarneaadressid ei ole ostuprotsessis üldse kasutatud ega ole võimalik salvestada tellimuse tarneaadressi.

2.1.2 Talutoodang.ee

Talutoodang.ee on ühe talu veebipood, kus kliendile pakutakse võimalust osta talutooteid. Talutoodang ei ole mitme talude turg, vaid ühe talu toodete pood. Veebilehel on talutoote kategooriad, talu kontaktandmed, kasutajate arvustused ja

informatsioon talu kohta. Talutoodete lehel kuvatakse kaubad loeteluna, kus on märgitud ka toodete kättesaadavus ja tarneaeg. Tellimuse saab teha tellimisvormi kaudu, mis asub kaubaloendi kõrval. Tooteid tellides peab kasutaja sisestama toodete nimed, e-posti aadressi, telefoni numbri ja täpse tarneaadressi.

Tarne toimub kokkulepitud kuupäeval, müüja teostab tarnet ise ja toob tooted kliendile.

Veebilehe disain ja kasutajakogemus on puudulikud ning tellimine on ebamugav.

2.2 Rakenduse skoop

2.2.1 Lõputöö eesmärk

Eesmärk on lähendada autori tuletatud probleemi. Probleemi lähendamiseks loob autor MVP rakenduse, mis koosneb serveripoolest ja kliendipoolest. Rakendusele tehakse lihtne analüüs. Enne rakenduse arendamist analüüsitakse erinevaid programmeerimiskeeli, tehnoloogiaid ja mustrid ning valitakse kõige sobivamad. Toodud rakenduse MVP suunatakse edasi rakendamiseks ja selle turule toomiseks.

Lõppu töö kirjutamise käigul on skoobis:

- Eestis toodetud talutooted
- Eestis püütud kalatooted
- Serveripoolne rakenduse MVP arendamine
- Eesrakenduse rakenduse MVP arendamine
- Välja toodud testimine

Lõppu töö kirjutamise käigul ei ole skoobis:

- Reaalse raha operatsioonid
- Kullerteenuse funktsionaalsus
- Kliendi tagasiside jätmine

2.2.2 Funktsionaalsed nõuded

Administraatori funktsionaalsuse nõuded:

- Administraatori rolli määramine

- Kasutajate blokeerimine
- Teiste kasutajate vaatamine
- Talupoe aktsepteerimine
- Talupoe blokeerimine

Äri-klienti funktsionaalsuse nõuded:

- Uue talupoodi loomine
- Kaupade lisamine
- Poe statistika
- Raha saldo

Klient (ostja) funktsionaalsuse nõuded:

- Talupoe valimis list
- Ostukorv
- Ostmine
- Ostmise ajalugu

2.3 Kasutusjuhtude mudel

Rakenduse kasutusjuhtude mudel on välja toodud joonisel. Muudel on tehtud kolme kasutaja rollidele: administraator, klienti ja äri-klienti, nende vahel on kasutus juhtud.

Administraatori rolli juhtud

- Kasutaja blokeerimine
- Uute poodi kontrollimine ja aktsepteerimine
- Turuplatvormi haldamine ja reeglite rikkuva poodi blokeerimine
- Kasutajatele abi ja tugi pakkumine

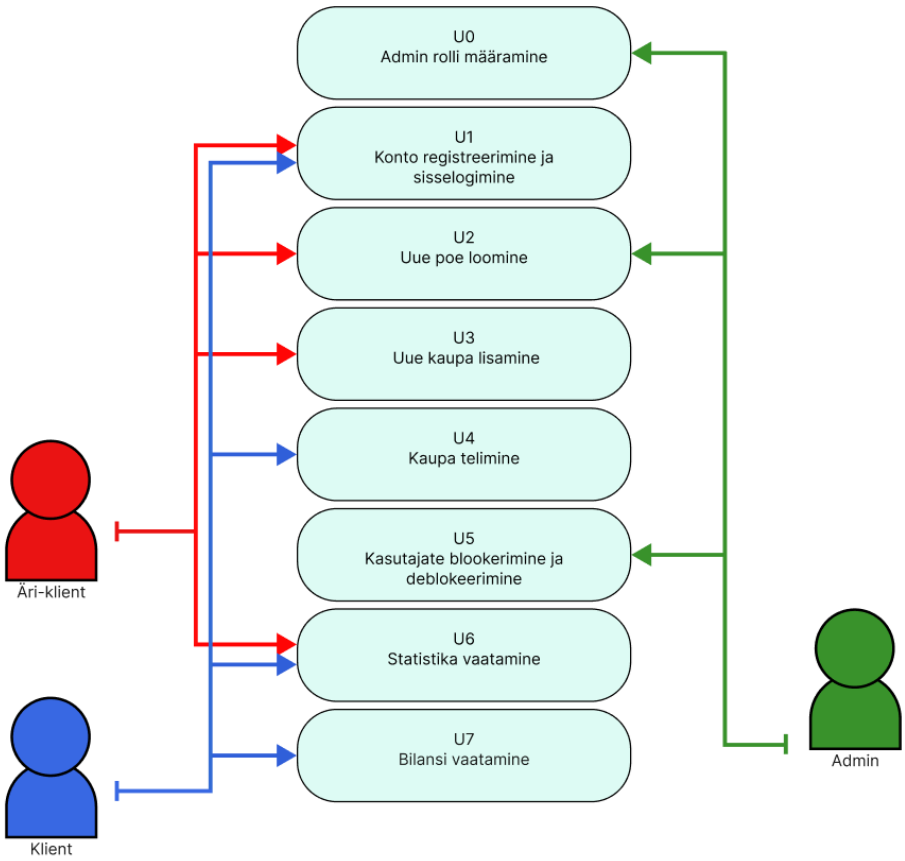
Kliendi rolli juhtud:

- Toote lisamine ostukorvi
- Kaupade otsimine
- Tellimuse esitamine
- Tellimuse ja selle staatusi jälgimine
- Oma konto ja saldo statistika vaatamine

Äri-kliendi rolli juhtud:

- Talupoe loomine ja haldamine
- Toodete lisamine ja muutmine
- Tellimuste vastuvõtmine ja töötlemine
- Poe statistika ja saldo vaatamine

Märkus: Joonisel on esitatud üldine kasutusjuhtude mudel ning täpsemad kasutusjuhud ja nende omavahelised seosed



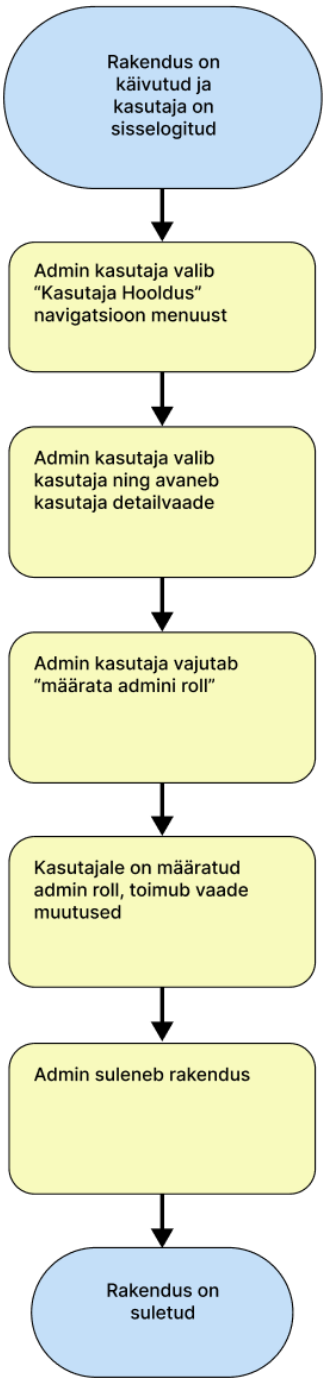
Joonis 1. Rakenduse kasutusjuhend.

2.3.1 U0 - Admin rolli määramine

Tegurid: Admin.

Ülevaade: Toimub sisse logimisega ja lõppeb rakenduse sulgemisega.

Eeltingimused: Administraatori rolliga sisse logitud kasutaja; Teised kasutajad on olemas.



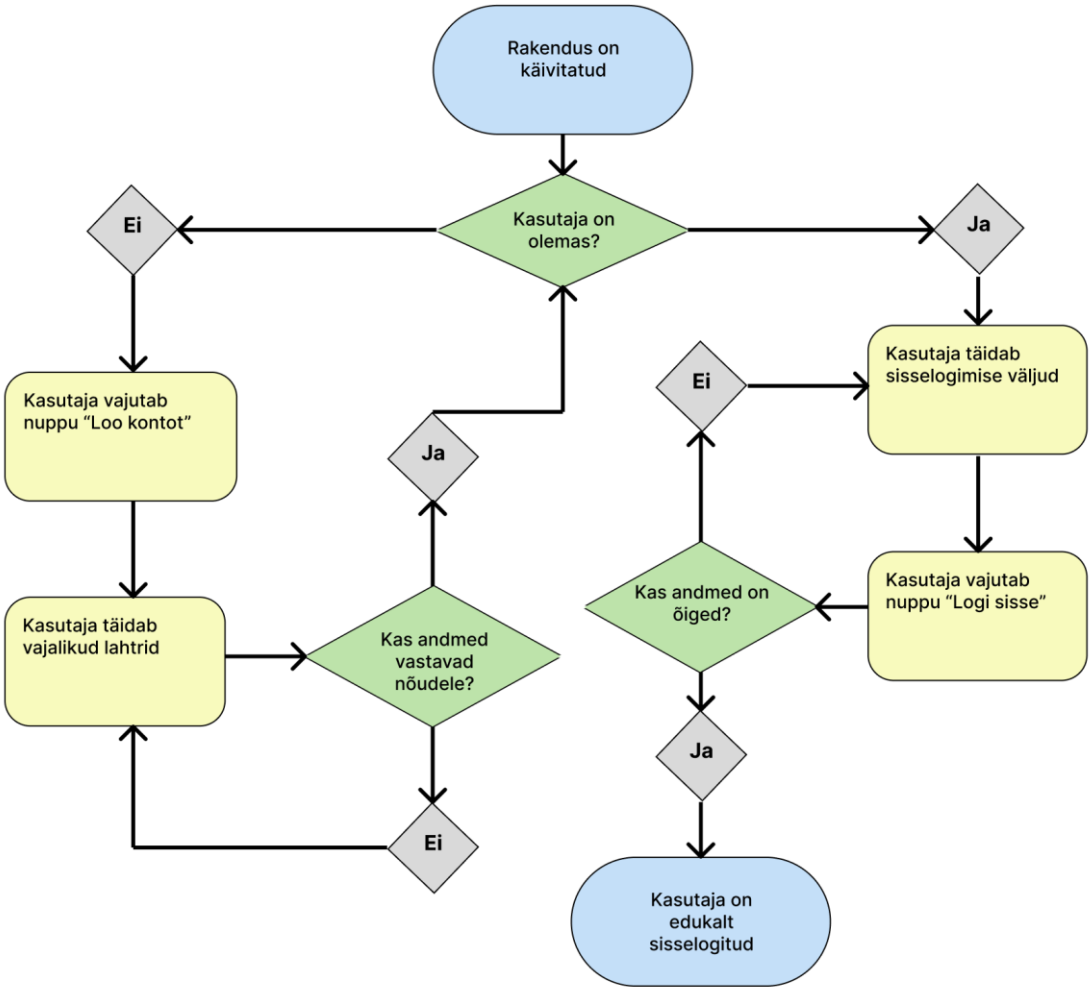
Joonis 2. Admin rolli määramine.

2.3.2 U1 - Konto registreerimine ja sisselogimine

Tegurid: Äri-klient; Klient.

Ülevaade: Toimub rakenduse avamisega ja lõppeb eduka sisselogimisega.

Eeltingimused: Kasutaja avab rakenduse.



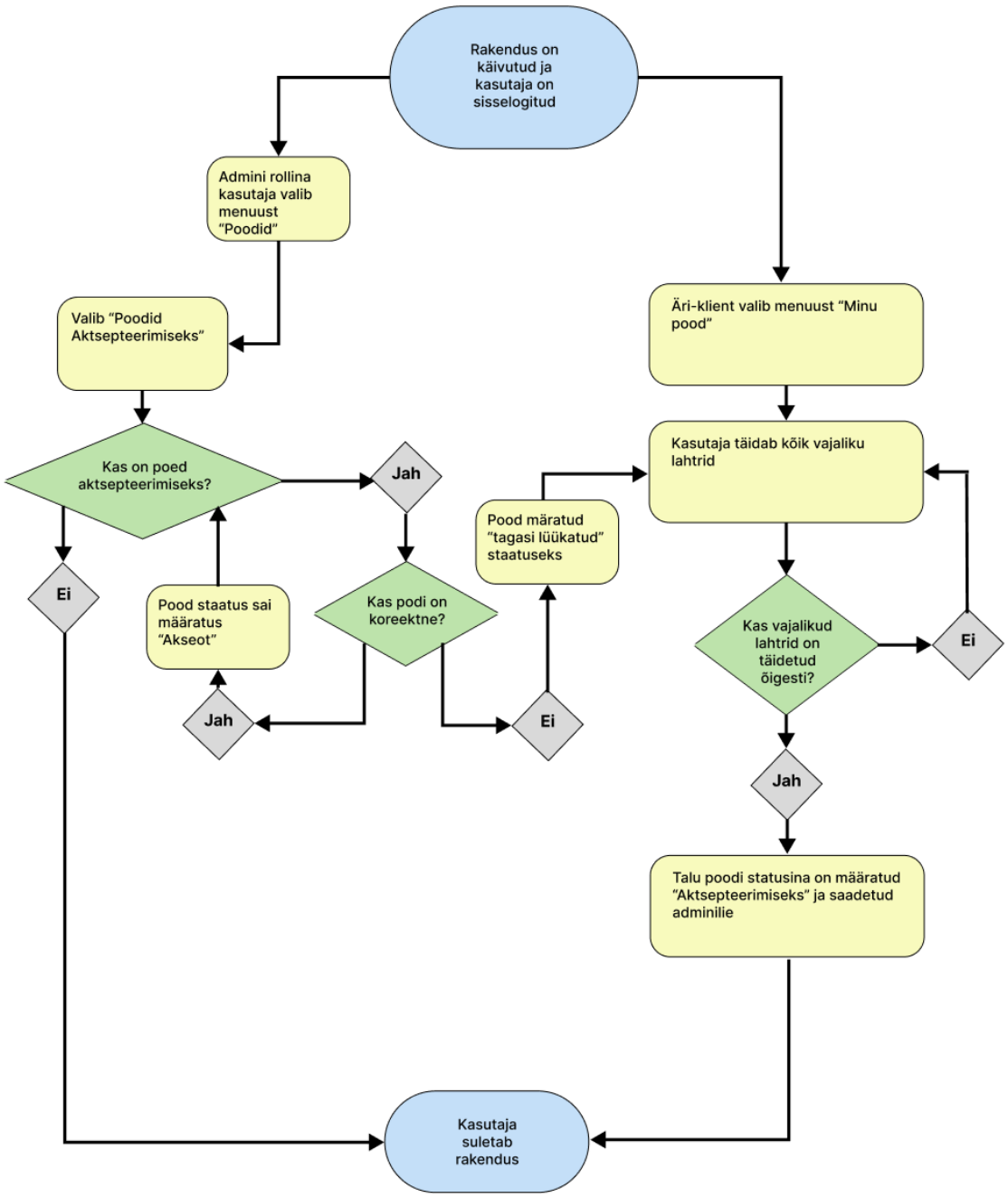
Joonis 3. Konto registreerimine ja sisselogimine.

2.3.3 U2 – Uue poe loomine

Tegurid: Äriklient ja Administraator.

Ülevaade: Kasutaja avab rakenduse ja loob eduka poodi.

Eeltingimused: Kasutaja avab rakenduse; Kasutaja on sisse logitud; Kasutaja roll on Äriklient või Administraator; Ärikliendil ei ole veel poodi.



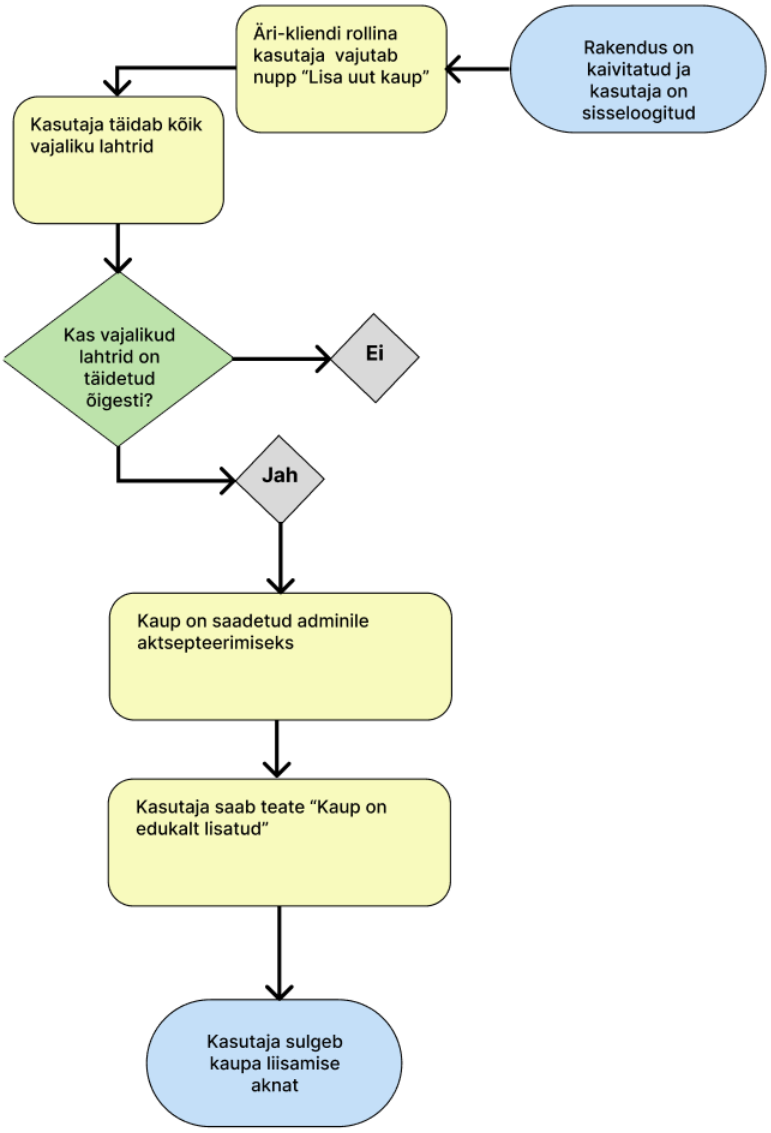
Joonis 4. Uue poe loomine.

2.3.4 U3 - Uue kaupa lisamine

Tegurid: Äriklient.

Ülevaade: Toimub „Uute toote lisamine“ nuppu vajumisel ja lõppeb teavitusega "Kaup on edukalt lisatud".

Eeltingimused: Kasutaja avab rakenduse; Kasutaja roll on Äriklient; Kasutajal on olemas pood.



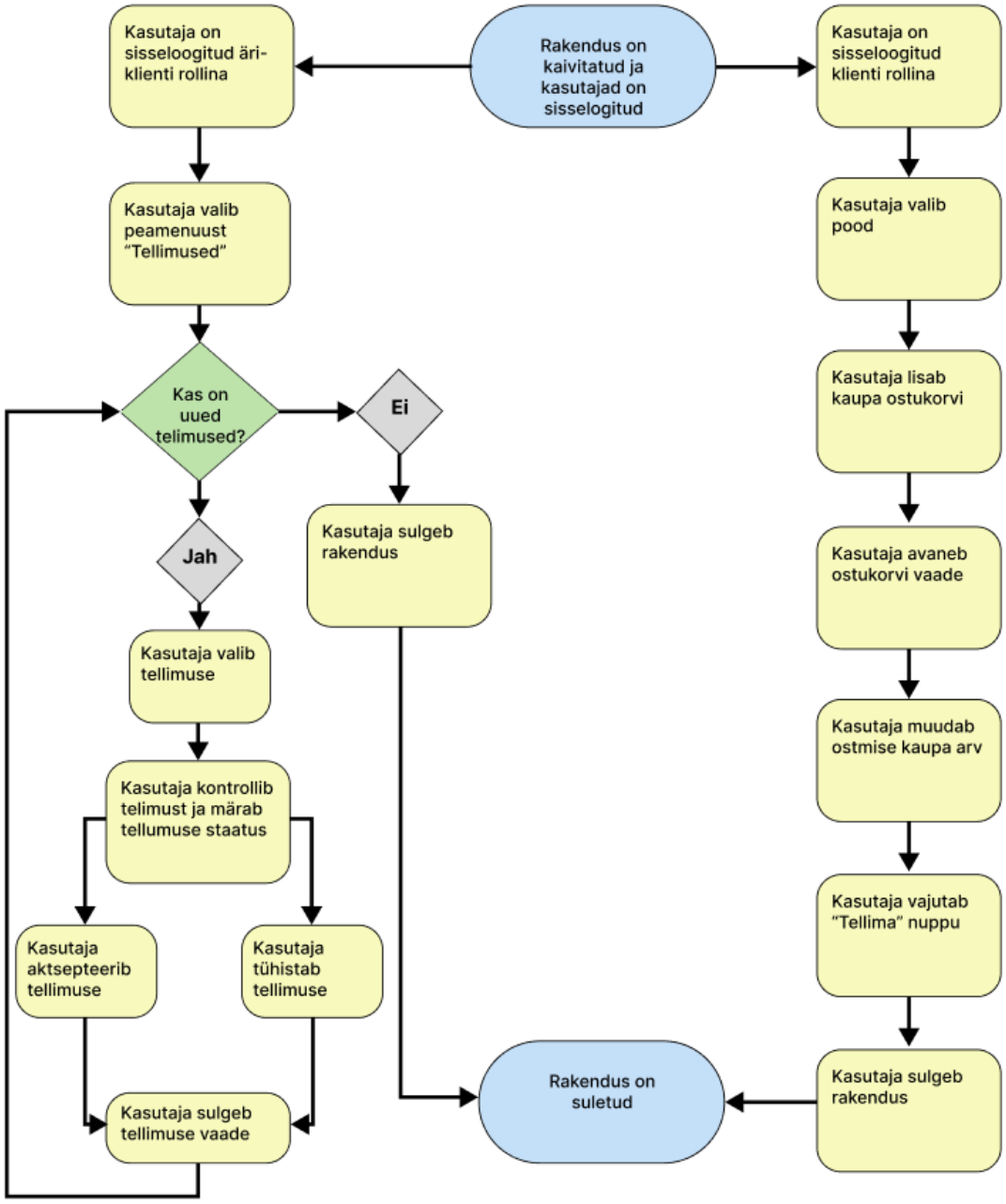
Joone 5. Uue kaupa lisamine.

2.3.5 U4 – Kaupa tellimine

Tegurid: Kasutaja roll on Äriklient; Kasutajal on olemas pood.

Ülevaade: Toimub rakenduse avamisega ja lõppeb eduka tellimuse vormistamisega

Eeltingimused: Kasutaja avab rakenduse; Kasutaja on sisse logitud; Kasutaja roll on Äriklient või Klient.



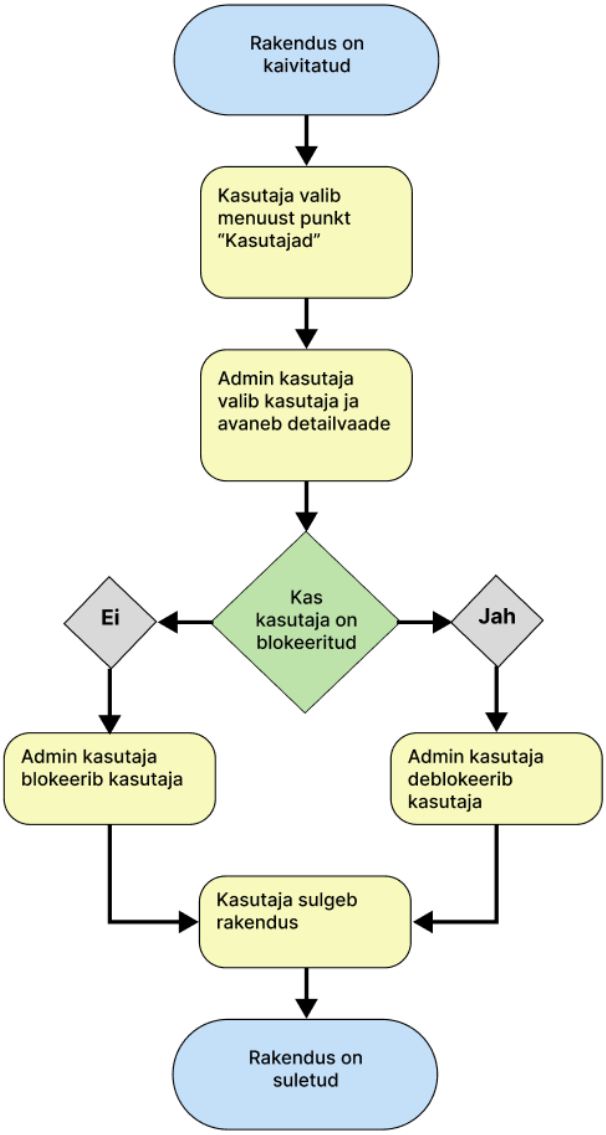
Joonis 6. U4 Kaupa tellimine.

2.3.6 U5 – Kasutajate blokeerimine ja deblokeerimine

Tegurid: Administraator.

Ülevaade: Toimub rakenduse avamisega ja lõppeb rakenduse sulgemisega.

Eeltingimused: Kasutaja on sisse logitud; On olemas kasutajad rollidega Äriklient ja Klient.



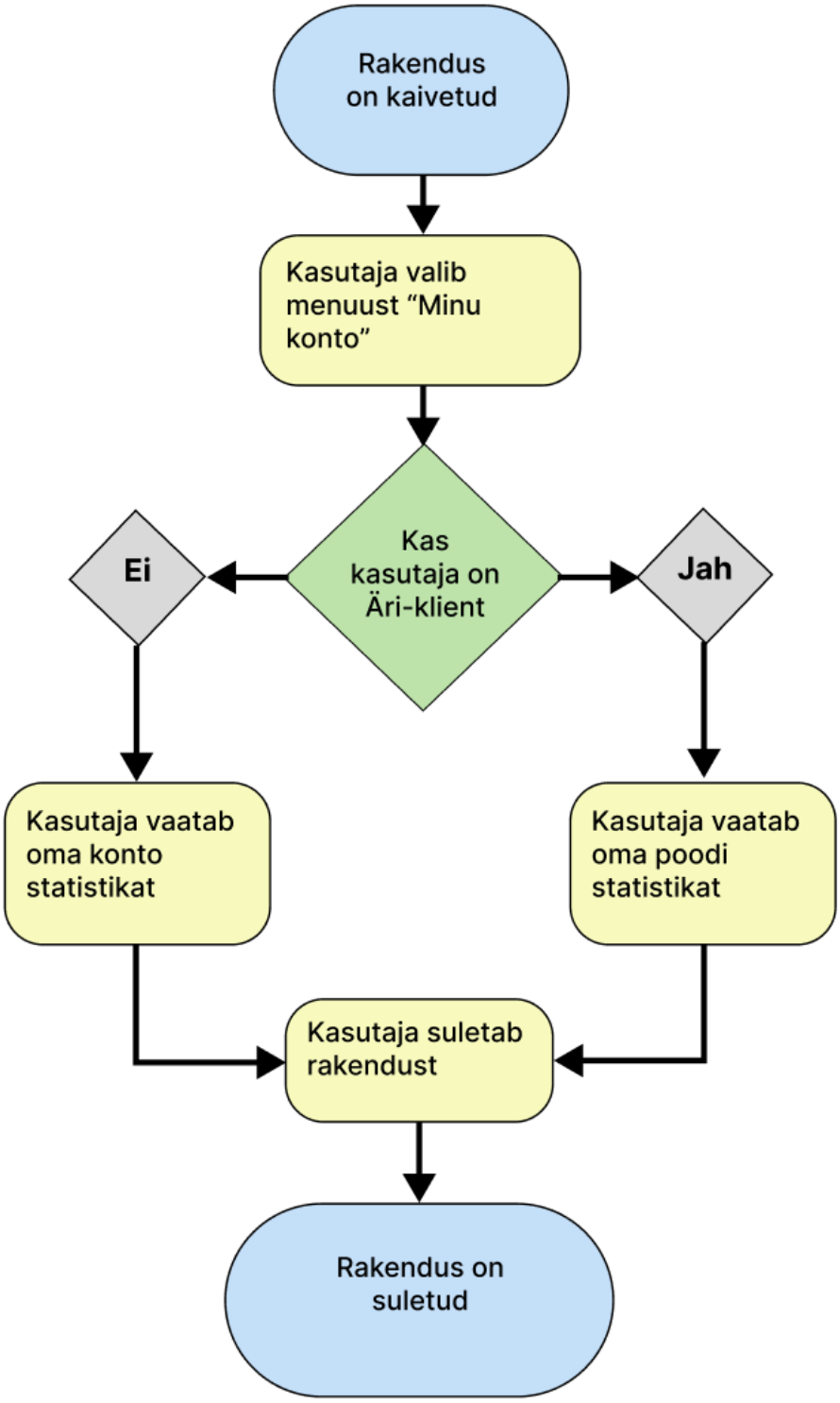
Joonis 7. U5 Kasutajate blokeerimine ja deblokeerimine.

2.3.7 U6 - Statistika vaatamine

Tegurid: Äriklient; klient.

Ülevaade: Toimub rakenduse avamisega ja lõppeb rakenduse sulgemisega.

Eeltingimused: Kasutaja avab rakenduse; Kasutaja on sisse logitud.



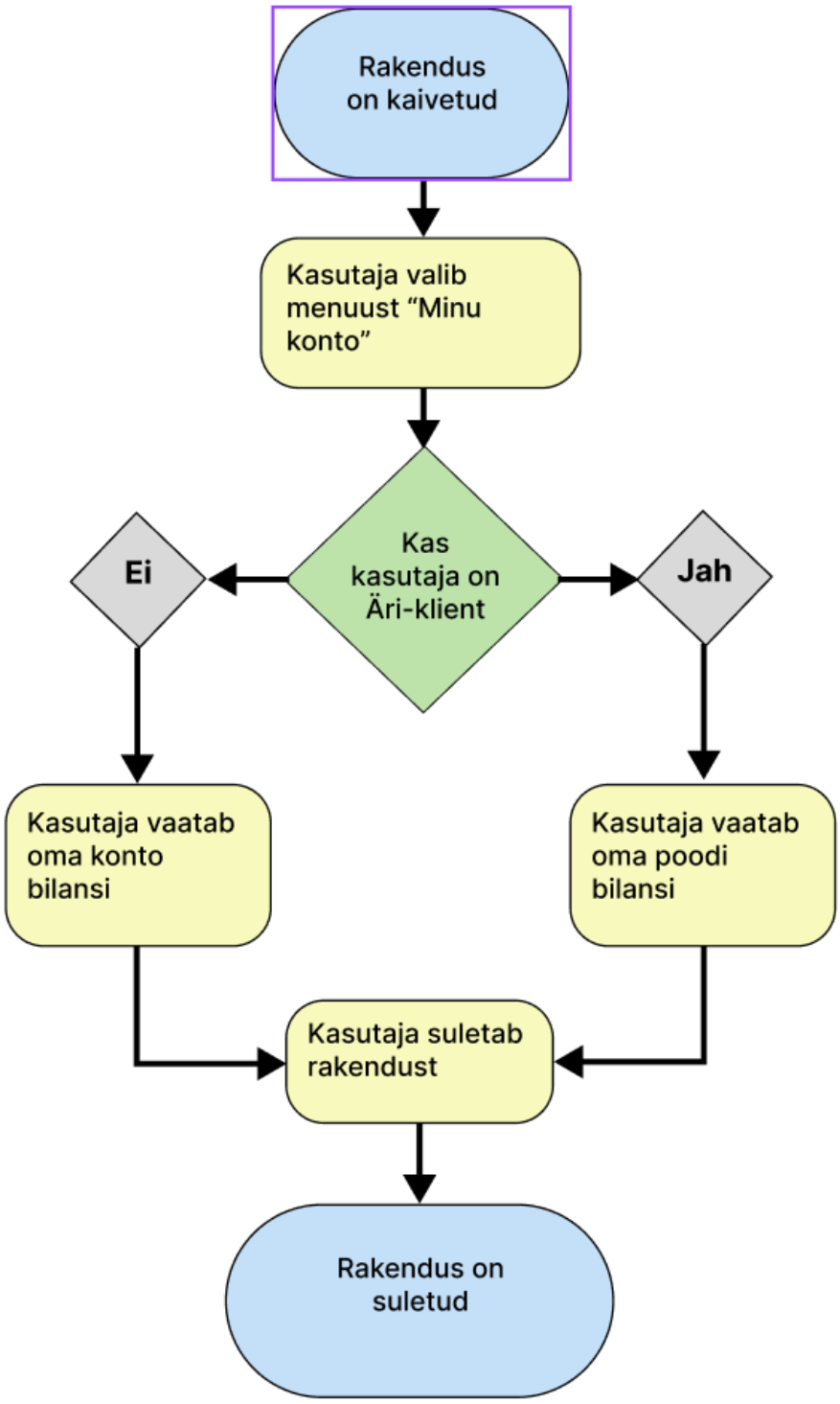
Joonis 8. U6 Statistika vaatamine.

2.3.8 U7 - Bilansi vaatamine

Tegurid: Äri-kasutaja; kasutaja.

Ülevaade: Toimub rakenduse avamisega ja lõppeb rakenduse sulgemisega

Eeltingimused: Kasutaja avab rakenduse; Kasutaja on sisse logitud.



Joonis 9. U7 Bilansi vaatamine.

3 Tehnoloogiate analüüs

Lõputöö käigus loodav rakendus on mõeldud mobiilseadmetel kasutamiseks, näiteks telefonis või tahvelarvutis. Seetõttu on tehnoloogiate valikul arvestatud, et rakendus oleks mobiilirakendus.

3.1 Serveripoolsed tehnoloogiat

Selles peatükis kirjeldatakse ja valitakse serveripoolsed tehnoloogiad, tuues välja nende eelised ja puudused.

3.1.1 Andmevahetuse protokolli tehnoloogia

Erinevad rakendused vajavad omavaheliseks suhtluseks vahendajaid. Selle jaoks luuakse rakendusliideseid, mille kasutamisel saab anda ligi teise süsteemi andmetele või funktsionaalsusele. Selleks on erinevad protokollid ja nõuded, mida kasutatakse rakenduste arendamiseks .

3.1.1.1 SOAP tarkvaraarhitektuur

SOAP (Simple Object Access Protocol) on veebiteenuse juurdepääsuprotokoll, mis on olnud kasutusel alates 1998 aastast. Microsoft algselt välja töötas SOAP-i, et asendada vanemad tehnoloogiad, mis Internetis hästi ei töötanud. Asendatavate tehnoloogia hulgal on DCOM (hajutatud komponendi objektimudel) ja CORBA (Common Object Request Broker Architecture), mis ebaõnnestuvad, kuna tuginevad kahebitti-sõnumitele. SOAP tehnoloogia kasutab sõnumid XML(Extensible Markup Language) kujul, need toimivad Internetis paremini. SOAP tugineb sõnumsideteenuste pakkumisel eranditult XML-ile, WSDL (Web Services Description Language) tüüpi kujul [1].

SOAP juurdepääsuprotokolli eelised

- SOAP on keele, platvormi ja transpordi sõltuvuse vaba. Päringu vormistamiseks on vaja ainult korrektselt kirjutatud WSDL [1].
- SOAP juurdepääsuprotokolli kasutamisel rakendused töötavad hästi hajutatud ettevõtte keskkondades[1].
- WSDL nõuab määrata saadetav andmete tüüp, mis välistab ujuvad tüübid ja nendega seotud vead

- Sisseehitatud vigade töötlemine tagastab veakoodi selgitusega[1].
- SOAP on väga laiendatav, toetab palju laiendust[1].

SOAP juurdepääsuprotokolli puudused

- SOAP võimaldab ainult XML struktuuri [1].
- Teenuse muutmine vajab WSDL muutmist, mille peale tuleb uuendada kõike klienditel oleva WSDL
- XML failid on suured, selle tõttu SOAP teenused nõuvad rohkem andmemahtu[1].

3.1.1.2 REST API tarkvaraarhitektuur

REST (Representational State Transfer) on tarkvaraarhitektuur, mis on arendatud et töötada meedia-, failide- ja teistega komponentidega. REST ei sõltu ühestki aluseks olevast protokollist ega ei ole tingimata seotud HTTP-ga. Kuid enamik levinumaid REST API rakendusi kasutavad HTTP rakendusprotokollina. REST API kasutavad ühtset liidest, mis aitab lahti siduda kliendi ja serveri pooled. REST API puhul rakendus kasutab standardsete HTTP-verbide ressursside toimingut, kõige levinumad nendest on GET, POST, PUT ja DELETE. REST tarkvaraarhitektuuri kasutamiseks pole vaja nii palju standartiseerimist nagu SOAP kasutamiseks, mis võimaldab REST kasutamine laiamaks [2].

Rest tarkvaraarhitektuur eelised:

- Kiire, pole vajalik WSDL faili kirjutama ja muuta, serveripoolne osa on eraldatud kliendipoolsest. [1]
- SOAP kasutab kõigi sõnumite jaoks XML, REST saab kasutada väiksemaid sõnumivorminguid[1]
- REST päringu töötlemine on kiirem SOAP töötlemist[1]
- REST tarkvaraarhitektuur on paindlik, see võimaldab TXT (tekstidokumendi vorming), HTML (HyperText Markup Language), JSON (JavaScript Object Notation) ja teised andmete vormingud. [1]

Rest puudused

- REST tarkvaraarhitektuur ei pakku turvalise kontrollid, arendajale on vaja ise iga päringu ligipääsetavus kontorollida.[1]

3.1.2 Serveripoolse rakenduse programmeerimiskeele ja raamistiku valimine

Selles peatükkis on kirjeldatud erinevaid programmeerimisekeeled ja raamistikud, tuues välja nende eelised ja puudused

3.1.2.1 Django raamistik koos Python programmeerimisekeelega

Python oli loodud Guido van Rossumiga ja tuli välja esialgselt skriptikeelena aastal 1991. Tänapäevaks Python on muutanud täis programmeerimiskeeleks, sellega saab arendama erinevate programmeerimisstiili kasutamisel: Objektorienteeritud programmeerimise stiil, protseduuriline programmeerimise stiil ja funktsionaalne programmeerimise stiil [3].

Python kasutamiseks on olemas väga lai standarti tegite valik, mille kasutamisel arendamise käigul ei tohi palju triviaalse koodi. See tõttu Python on sageli kirjeldatakse nagu "sisestatuga patareidega" programmeerimiskeel. Python on dünaamiliste andmetüüpidega keel, kus programmeerijal ei ole tarvis määrata muutujate tüüpe. See suurendab programmeerija võimalusi ja lühiajaliselt lihtsustab arengut. Süntaktiliselt Python programmeerimiskeeles kirjutatud kood välja näeb nagu käivitatav pseudokood, mida saab inglise keele tekstsiga segi ajada. Selle tõttu koodi arusaamine lihtne ja vajab väiksem kompetentsusi koodi kirjutamiseks. Python programmeerimiskeeles arendamine ja rakenduse hoius on kiire ja odav, sellega saab kirjutada ka rakenduse prototüüpide testimiseks.

Django on kõige populaarne Python programmeerimiskeelel põhinev avatud lähtekoodiga raamistik. Django raamistiku eesmärk on anda arendajatele mugav ja lihtne töörist veebirakenduse loomiseks. See järgib MVC (Model View Controller) mustrit ja kasutab REST mustrit API kontrolleri loomiseks, ning sobib keerukate ja funktsioonirikaste veebirakendusi arendamiseks. Django kasutab Python kõigi raamistiku toimingute jaoks ja pakub arendajale kasutajaliide operatsiooni loomiseks, lugemiseks, uuendamiseks ja kustutamiseks. Django põhjal on loodud palju tuntud veebirakendust, nagu Disqus, Mozila ja Washington Post [4].

Django raamistiku eelised:

- Django raamistik on lihtne õppida, mis aitab algajatel arendajatel kiirendada kogu arendusprotsessi algusest lõpuni [4].
- Django pakub arendajatele laia valiku funktsioonilt, mille tõttu saab täitma paljud levinud arendamise nõudeid, nagu autentimine, navigatsioon ja palju muud [5].
- Django pakub arendajale abi turvaprobleemide hoidmiseks, mille hulgas on saidiülest skriptemist, klõpsamist, SQL-i sisestamist ja võltsimise taotlemist [4].
- Django rakendused on hea skaleerimiseks, need on lihtne ja mugav mastabeerida [4].

Django raamistiku puudused:

- Django ei sobi vaikese rakenduse arendamiseks, raamistik vajab palju koodi kirjutamist, ning päringu töötlemiseaja on suur [6].
- Konventsioonide puudumine tekitab arendajatele palju probleeme, mille hulgas on madal arendamise mugavus, kontrastsed komponendid ja madal arenduskiirus [6].

3.1.2.2 ExpressJS raamistik ning NodeJS

ExpressJS on tasuta ja avatud lähtekoodiga veebirakenduste raamistik, mis on ehitatud Node.js platvormi peale. Selle eesmärk on lihtsustada serveripoolsete rakenduste loomist ja hooldamist. ExpressJS abil saate luua dünaamilisi ja interaktiivseid veebisaite, ehitada REST API ning käidelda HTTP päringuid ja vastuseid korraldatud ja tõhusal viisil [7].

ExpressJS eelised

- Lihtne õppimiseks, JavaSript on üks kõige laialdasemalt kasutatavaid programmeerimiskeeli ja peaaegu kõik arendajad oskavad seda kasutada. Arendaja õppimine ei nõua palju vaeva ega aega [8].
- Kui Node.js nõuab JavaScripti koodi, arendaja ei pea kasutama ühtegi teist serveripoolset keelt.
- Suur jõudlus – Node.js kasutab Node.js tõlgendamiseks Google'i JavaScripti V8 mootorit. Mootor muudab JavaScripti-põhise koodi masinkoodiks ja muudab koodi tõhusa rakendamise lihtsamaks. Käituskeskkond parandab ka täitmiskiirust, kuna JavaScript toetab mitteblokeerivaid I/O toiminguid [9].

ExpressJS puudused

- ExpressJS raamistik ei sisalda paljusid täiustatud funktsioone juba algusest peale, nagu näiteks turvalisuse tugi. Arendajad peavad sageli kasutama täiendavaid mooduleid või looma kohandatud funktsionaalsust, et rahuldada nende vajadusi [10].
- Express.js ei paku tugevat tüübikontrolli ega kompileerimise ajal tehtavaid kontrollimisi, mis võib viia vigade ja vigade tekkimiseni teie koodis. See on eriti oluline suuremate rakenduste puhul, kus on palju sõltuvusi ja keerukat loogikat [10].

3.1.2.3 Asp .NET core raamistik koos C# programmeerimiskeel

C# on kaasaegne programmeerimiskeel, mida saab kasutada mitmesuguste ülesannete ja eesmärkide täitmiseks erinevates erialades. C# põhikasutusala on Windowsi .NET raamistik, kuigi seda saab rakendada ka avatud lähtekoodiga platvormidel. See väga mitmekülgne programmeerimiskeel on objektorienteeritud programmeerimiskeel (OOP) ning võrreldes teiste keeltega suhteliselt uus, kuid juba saanud suur usaldusväärsus [11].

C# on tugeva tüüpi turvaline programmeerimiskeel, mis nõuab arendajatelt läbimõeldava koodi kirjutamist. Programmeerimise keel ei luba ujuva tüübi, mis võivad põhjustada andmete kadumist või muid probleeme. Sellega arenduse protsess saab aeglaseks, aga aitab kirjutama veakindlat koodi.

C# on avatud lähtekoodiga, mille alusel tegutseb ja juhib seda Microsoftist .NET Foundation. C# keele spetsifikatsioonid, kompilaatorid ja sellega seotud tööriistad on avatud lähtekoodiga ja asub Github avaliku koodi hoidjas. Kuigi C# keele omaduste kujundamisel on eestvedajaks Microsoft, see on avatud lähtekoodiga kogukond väga aktiivne keele arendamisel ja täiustamisel. C# on mitmete teiste kõrgema taseme programmeerimiskeeltega võrreldes kiire [12].

C# programmeerimisega keelega saab arendama igasuguste rakendused, mille hulgas on Windowsi kliendirakendused, komponendid ja tekid, teenuste ja API, veebirakendused, mobiilirakendused, pilverakendused ja videomängud [12].

ASP.NET Core on tasuta, avatud lähtekoodiga, kõrge jõudlusega, kerge raamistik pilvepõhiste rakenduste, näiteks veebirakenduste ja mobiilsete tagakülgede loomiseks.

.NET Core on modulaarse arhitektuuriga minimaalse ülekoormusega ning seejärel saab lisada muid täiustatud funktsioone vastavalt rakenduse nõuetele NuGet (tarkvara paketihaldussüsteem) pakettidena. Selle tulemusena on selle jõudlus kõrge, see nõuab

vähem mälu, vähem paigaldust suurust ja on lihtne hooldada.ASP.NET Core raamistiku arendatud rakendused saab käivita Windows, Linux ja Mac seadustel [13].

ASP.NET Core ja C# pakub väga hea ja lai dokumentatsioon, kui ka suur arendajate kogukond.

ASP .NET raamistiku eelised:

- ASP.NET Core toetab Microsoft. Usaldusväärse suur ettevõttete tugi tagab, et pikaajalised investeeringud ASP.NET-põhistesse rakendustesse on turvalised ja mõistlikud [14].
- C# on tugevasti tüpiseeritud keel, arendustööriistade kasutamisel, nagu IntelijIDEA või VisualStudio, saab varakult vead tuvastada ja parandada. See pärast C# kirjutatud kood on veakindel [14].
- NET Core on kavandatud modulaarseks, mis tähendab, et peate oma rakenduse jaoks lisama ainult raamistiku komponendid, mida vajate. See võib hõlbustada kergekaaluliste rakenduste loomist, mis on optimeeritud jõudluse jaoks [15].
- .NET Core on arendatud, et olema kiire ja tõhus ning seda on optimeeritud serveripoolsete rakenduste jaoks. See teeb sellest hea valiku kõrge jõudlusega rakenduste loomisel [15].

ASP .NET raamistiku puudused:

- .NET Core ei pruugi olla ühilduv vanemate platvormidega, seega võib teil olla vaja kasutada täielikku .NET raamistikku, kui soovite toetada vanemaid platvorme [15].
- .NET Core on suhteliselt uus raamistik, seega võib olla õppimiskõver neile arendajatele, kes on harjunud täieliku .NET raamistikuga [15].
- Võrreldes teistega avatutega lähtekoodiga raamistikutega ASP.NET on üsna kallis kasutamiseks. Arendamine ja hoidmine vajab paljud kulud, mille hulgas on Visual Studio litsents, SQL Serveri litsents, ning Windows Serveri litsents [16].

3.1.2.4 Spring Boot raamistik koos Java programmeerimiskeelega

Java on laialdaselt kasutatav objektorienteeritud programmeerimiskeel, mis töötab miljarditel seadmetel, sealhulgas sülearvutitel, mobiilseadmetel, mängukonsoolidel,

meditsiiniseadmetel ja paljudel teistel. Java reeglid ja süntaks põhinevad C ja C++ keeltel. Üks oluline eelis Java tarkvara arendamisel on selle lihtne ülekantavus. Kui olete Java programmi koodi kirjutanud sülearvutis, on selle koodi liigutamine mobiilseadmesse väga lihtne. Kui keel loodi 1991. aastal Sun poolt James Goslingu poolt, oli peamine eesmärk olla võimeline "kirjutama üks kord, töötama kõikjal" [17].

Java programmeeritud rakenduse käivitamiseks kasutatakse JVM (Java Virtual Machine). JVM on toodud kahega põhi funktsioonidega – käivama Java programmeerimiskeeles kirjutatud mis tahes seadmes või operatsioonisüsteemis, ning haldama ja optimeerima rakenduse kasutatavat mälu [18].

Spring on avatud lähtekoodiga üks enim kasutatavaid Java raamistikke, mis pakub tugevat programmeerimise, ning konfigureerimise võimalust. Raamistiku loomise eesmärk oli lihtsustada rakenduste arendamist Oracle populaarsel Java EE tehnoloogia pinnal, mis oli tol ajal väga keeruline ja raskesti kasutatav. Spring Boot raamistik üheks peamiseks eeliseks on sõltuvuse injektsiooni mustriga kasutamine. Sõltuvuse injektsioon muudab rakenduste vajaliku funktsionaalsuse rakendamise palju lihtsamaks ja võimaldab luua nõrgalt seotud ja üldisemaid klasside [19].

Spring Boot raamistiku eelised:

- Spring raamistik pakub automaatne konfigureerimise funktsionaalsusi mis toimub konfigureerimise failide kaudu [20].
- Spring Boot kasutab mälu säästmiseks kinnitamise tehnikat, mille kaudu kompileeritakse lähtekeel alglaadija abil. Tulemusena rakendused laadivad palju kiiremini [20].
- Spring Boot rakendused on platvormiüldised ja ei vaja lisa arendust erinevate operatsioonisüsteemidel käivitamiseks [20].

Spring Boot raamistiku puudused

- Edukalt arendamiseks on vaja kogemusi ja teadmusi Spring ökosüsteemist ja OOP programmeerimisest [21].
- Ei sobi monoliitprojektidele, Spring Boot töötab mikroteenuste loomisel väga hästi, aga ei ole parim valik monoliitsete rakenduste arendamiseks [22].

3.1.3 Andmebaasi tehnoloogia

Analüüsitakse erinevaid andmebaaside tehnoloogiad tuues välja nende eelised ja puudused. Andmebaasi haldussüsteemile on sellised nõed:

- Andmebaasi haldussüsteem on relatsioonilane
- Andmebaasi haldussüsteem põhinenud SQL keeles
- Andmebaasi haldussüsteem on tasuta
- Andmebaasi haldussüsteem on ajaproovitud ja on olemas dokumentatsioon

3.1.3.1 MySQL

MySQL on võimsas relatsioonilane andmebaasi haldussüsteem, mis kasutab SQL päringukeelena. Algselt MySQL oli mõeldud töötamiseks väikese ja keskmise suurusega andmebaasidega, kuid nüüd hiljutiste jõudluse ja mastaapsuse täiustuste pärast saab seda kasutada peaaegu kõikjal ja mis tahes suurusega rakenduses. MySQL on kirjutatud C ja C++ programmeerimiskeele kasutamisega ja tänaks see on enamasti SQL standardiga kooskõlas.

MySQL põhiversiooni levitab Orcacle Corporation, versioon on tasuta ja tuleb avatud lähtekoodi litsentsiooniga. MySQL saab kasutada Linux põhinev operatsiooni süsteemil ja enamikul muudel platvormidel. See on avatud lähtekoodiga LAMP (Linux, Apache, MySQL, PHP/Perl/Python) pinu oluline komponent koos Linuxi, Apache ja PHP programmeerimiskeelega. LAMP-pinn on avatud lähtekoodiga veebirakenduste arendamise pinn Linux operatsioonisüsteemile. MySQL saab kasutada kui serveri, kui ka kliendi süsteemi osana või manustatud süsteemi osana. Administraatorid ja kasutajad määravad seosed andmebaasi tabelite sees ja vahel. Erinevaid veerge saab märkida kohustuslikuks või valikuliseks ning need võivad toimida primaarvõtmena või osutada teisele tabelile. MySQL on stabiilne, usaldusväärne ja hõlpsasti kasutatav [23].

Seda on kasutusel paljudel suurtel ettevõtjal, mille hulgas on Uber, Airbnb ja Shopify[24].

MySql andmebaasi haldussüsteemi eelised:

- Rakenduste jõudluse suurendamine. Selle üheks peapõhjuseks on see, et MySQL-i salvestatud 9 protseduuri koostatakse pigem nõudmisel kui kompileeritakse ja muudel meetoditel andmebaasi salvestatakse [24].
- MySQL on loodud platvormiüldine andmebaasi server, see tähendab et server töötab erinevates operatsiooni süsteemides mille hulgas on Linux, MacOS ja Windows [24].
- MySQL-i andmete turvalisus on uks tema tugevast eelistest. Seda on tunnistatud üheks turvalisemaks ja usaldusväärsemaks andmebaasi haldussüsteemiks [24].
- MySQL-serveriga ühenduse loomiseks on saadaval erinevad turvalised ja sujuvad ühendusmehhanismid. Need ühendused hõlmavad nimega torusid, TCP/IP-pesasid ja UNIX-i pistikupesasid. Mis on väga vajalik funktsioon, eriti arvestades veebirakendusi[24].

MySql andmebaasi haldussüsteemi puudused:

- MySql stabiilsus on kehv ja kaotab teiste andmebaasi haldussüsteemi puhul [24]
- Kuigi MySql on halvasti skaleeritav haldussüsteem, mis suudab käsitleda suuremahulisi ja suure hulga andmebaase, ei saa seda teha tõhusalt. Samuti ei saa tehinguid teha tõhusalt ja tõhusalt [24].
- MySQL niidibasseini toetus ei tule minimaalsega versiooniga. Seda kasutamiseks tuleb osta vähemalt Enterprise taseme versiooni [25].
- MySQL ei toeta CASCADE funktsiooni [27].

3.1.3.2 MariaDb

MariaDB on populaarne avatud lähtekoodiga relatsioonilane andmebaasi haldussüsteem, mis oli loodud 2009. aastal. See on GNU General Public litsentsi alusel ja kavatseb jääda tasuta ja avatud lähtekoodiga. MariaDB looja nimetas projekti oma teise tütre järgi ja mõelnud et see saab hea MySQL asendamiseks [28].

Kõik MariaDB komponendid on GPL, LGPL või BSD litsentsi alla, mis tähendab et tehnoloogia kasutamine on tasuta. MariaDB ei toeta skeemi kontseptsiooni, kasutaja ühenduse MariaDB loomiseks, ta ei loo ühendust konkreetse andmebaasiga, vaid selle asemel pääsevad juurde tabelile, mille jaoks neil on õigused. MariaDB oskab automaatselt parandama sissetulevaid andmetabeli skeemale järgi andmetüübid. MariaDB pakub tasuta niidibasseini toetus. Niidibassein on nagu ühendused, mis üritavad andmebaasiga linkida. Ühendus valib andmebaasi päringu tegemiseks lõime[29].

On võimas töötada erinevatega andmehoiustega mootoritega, mida saab valida iga tabeli kohta [30].

Virtuaalsete veergude tugi on MariaDB üks peamine funktsioon, seda saab kasutada arvutuste tegemiseks andmebaasi tasemel. Kui mitu rakendust pääseb juurde ühele andmeveerule, ei pea igasse päringu kohta eraldi kirjutama, andmebaas teeb seda ise[30].

MariaDB andmebaasi haldussüsteem eelised

- MariaDB ühendusprotokoll on tagasiühilduv. Mille tõttu ei pea vanemat versiooni uuendama, et saata ühendust luua uuema MariaDB versiooniga [31].
- MariaDB põhi on hargnenud MySQL haldussüsteemist, enamasti kõik MySQL-i funktsioonid töötavad ilusti. Migratsioonid MySQL haldussüsteemist MariaDB haldussüsteemile on kiire ja mugav [32].
- MariaDB toob tugi salvestada harvemini kasutatavaid andmeid eraldi sektsioonis, mis aitab kiirendama päringu täitmine [33].
- Alates versioonilt 10.0 saab täita mitte päringut üheaegselt ilma jõudluse halvenemiseta, mille tõttu tööd andmebaasiga saab kiiremaks [26].

MariaDB andmebaasi haldussüsteem puudused

- MariaDB-l on piiratud vahemälu, mis võib suuremate andmekogumite korral põhjustada pikemaid läbilaskeid [34]
- Kuna MariaDB-l puudub võime suuremaid andmekogusid hallata, ei sobi see suuremahulise andmetöötluse toetamiseks [34].

3.1.3.3 PostgresSQL

PostgresSQL on relatsiooniandmebaasi andmebaasi haldussüsteem, mis kasutab ja laiendab SQL-i keelt koos paljude funktsiooniga. PostgreSQL päritolu ulatub 1986. aastasse California ülikooli Berkeley projekti POSTGRES osana ja sellel on rohkem kui 35 aastat aktiivset arendustööd. PostgreSQL lähtekood on vabalt saadaval avatud lähtekoodiga litsentsi alusel. Mis annab vabaduse seda kasutada, muuta ja rakendada vastavalt vajava ettevõtte vajadustele. PostgreSQL haldussüsteem oskab kasutada sisemist vahemälu ja serveri vahemälu sageli kasutatavate andmete toomiseks, mille tõttu sageli kasutatavad päringud saab kiirem ja jõudsem. PostgreSQL toob tugi

geograafilisi objekte salvestamiseks, mida lihtsustab asukohapõhiste teenuste töötamist[35].

PostgresSQL eelised

- PostgreSQL haldussüsteemi logide ettekirjutamine kindlustab, et arendaja teeb vähem vigu [36].
- Dokumentatsioon on saadaval ja hästi kirjutatud [36].
- PostgreSQL haldussüsteemil on hea ühilduvus teise andmebaasi hooldussüsteemidega[33].
- On võimsas JSON ja JSONB andmetüüpidega töötamiseks [33].
- PostgresSQL haldussüsteemi arendajad suhtuvad turvalisusesse väga tõsiselt ja sageli toovad turvaparandamise uuendused [33].
- PostgreSQL on täielikult SQL-iga ühilduv [27].
- PostgreSQL toob SSL kasutamise tugi [27].

PostgresSQL puudused

- PostgresSQL haldussüsteemi töö kiiruse parandamine nõuab rohkem pingutust võrreldes teistega andmebaasi haldussüsteemidega [27].
- PostgresSQL haldussüsteem versioon uuendamisega ei toob tugi automaatseks andmebaasi uuendamist. Mille tõttu arendajale tuleb ise migratsiooni teha [27].

3.2 Kliendipoole tehnoloogiat (Front-end)

Selles peatükis kirjeldatakse ja valitakse kliendipoolsed tehnoloogiad, toes välja nende eelised ja pooled.

3.2.1 Kliendipoolse programmeerimiskeel ja raamistik

Selles peatükkis kirjeldatakse ja valitakse tehnoloogiat kliendipoole rakenduse arendamiseks, toes välja nende eelised ja poolesed. Valimise nõue raamistikule on tugi toomine Androidi ja IOS põhitatud operatsioonisüsteemidega mobiiliseadmetel käivitamiseks.

3.2.1.1 Ionic

Ionic on avatud lähtekoodiga SDK (Software development kit) raamistik platvormiüldiste rakenduste loomiseks, sellega saab arendama rakendused arvuti ja mobiiliseadmete käivitamiseks. Algversioon sai elu 2013 aastal, mis oli põhjendatud AngularJS ja Apache Cardova peale. Ionic oli üks esimestest raamisikutest mis tõi tugi hübriid-mobiilirakenduse arendamisiks. Ionic pakub arendajatele kliendipoolse arendamise tööriist, mis on üles ehitatud Cordova platvormile [37].

Cordova pakub töörist JavaScripti koodi arusaadava seadmetele, ning Ionic raamistik on tegeleb välja nähtemisega. Ionic nõuab arendajatelt HTML5, CSS, JavaScript ja AngularJS oskuseid rakenduse arendamiseks. AngularJS raamistiku asemel saab kasutada ka VueJS või ReactJs raamistikud. Ionic hübriidrakendused kasutavad WebView funktsionaalsusi, ehk mobiilikujuline veebivaade leht, mis kasutatakse kasutajaliide koostamiseks. Ionic töötamine toimub sama nagu veebileht, see kuvab HTML vaade kasutamises kõike leevitud veebi tehnoloogiat [38].

Ionic raamistiku eelised:

- Ionic kasutamiseks ei vaja spetsiaalse keeli oskusi, see kasutab kõige leevimaid veebipõhine arenduse tehnoloogiad. Selle tõttu arendajate leidmine ei ole raske [38].
- Ionic toob ühe koodibaasi kasutamise tugi, arendaja saab kasutada VueJs, AngularJs ja ReactJs raamistiku abil kirjutatud rakenduse koodi [39].
- Ionic võimaldab rakenduse kasutajaliide vaadete muutmine käivitatud operatsioonisüsteemi standartide järgi [40].

Ionic raamistiku puudused

- Ionic silumisetöörist ning testimisetöörist ei ole kõige usaldusväärsed, see sageli ei anna veateated mitte töötava koodi kohta [38].
- Ionic rakendused nõuab käivitamiseks palju tekid, mis sageli tekkivad ühilduvusprobleemid [38].
- Ionic rakendused töötavad palju aeglasem kui natiivsed mobiilirakendused [38].
- Ionic ei too tugi tugeva graafiliste elementide joonistamiseks, mis on vajalik videomängu või teiste 3D elementide kasutava rakenduste arendamiseks [38].
- JavaScript on mitte usaldusväärne programmeerimiskeel, mis tekkib palju raskusi ja probleemi arendamisekäigul.

3.2.1.2 React Native

2015 a. Facebook avaldas esimene React Native versiooni avatud lähtekoodiga projektina, vaid paari aastaga on sellest saanud ühest parimast mobiilirakenduse arendamise raamistikuna. React native on populaarne JavaScript programmeerimiskeele kasutav mobiilirakenduste arendamiseks raamistik. Sellega saab arendama platvormiüldiste rakendused, mis käivitab iOS ja Android seadmetel. Raamistik võimaldab luua saama koodibaasi kasutades rakendusi erinevatele platvormidele.

ReactJS raamistikuga kirjutatud koodi üks kord luuatud, saab kasutada rakenduse toiteks iOS ja Androidi operatsioonisüsteemiga seadmetele. Mille tõttu võimaldas raamistik esiotsa arendajatel, kes varem said töötada ainult veebitehnoloogiatega, luua mobiilplatvormidele usaldusväärseid ja kasutusvalmis rakendusi [41].

React Native on kirjutatud JavaScripti ja JXL-i segu abil, mis on XML-ile sarnane spetsiaalne märgistuskood. React Native teiste muude platvormide üldiste raamistiku unikaalsus seisneb selles, et React Native vaadete ehitamiseks kasutab mitte WebView, vaid oma vaated ja komponendid [42].

React Native raamistik on kasutamisel suure ettevõttega nagu FaceBook, Instagram, Skype, Soundcloud, Pinterest, Discord [41].

React Native raamistiku eelised:

- React Native raamistik kirjutatud JavaScriptis, mis on populaarne ja arendajatel ei ole vaja õppida spetsiaalne programmeerimisekeelt et raamistikuga sellega saama hakkama[41].
- React nativile komponenditel on hea ühilduvus erinevatega raamistikuga, üles 90% sisseehitatud koodi saab taaskasutada [41].
- ReactJS ja React Native raamistikude ühilduvus. Juba oleva ReactJS veebirakenduse koodi saab kasutada mobiilirakenduse arendamiseks [41].
- React Native raamistikul on suur arendajate kogukond annab kindlust leia appi probleemi tekitamisel, 2020. aasta keskpaiga seisuga on Stack Overflow märgendis React Native umbes 50 000 aktiivset liiget [41].
- Kuigi JavaScript ei tööta nii kiiresti kui algkood, on erinevus nähtamatu. Selle täiendavaks tõestamiseks oli läbi viietud testi, milles võrreldakse React Native ja Swift

kasutamisel kirjutatud lihtsa rakenduse kahte versiooni, mis mõlemad saavutasid sarnased jõudlustulemused [41].

React Native raamistiku puudused:

- Mäluhaldus on kehv, nii et kui rakendused nõuavad suurepärast jõudlust, siis seda siin React Native ei täideta [43].
- Kehv dokumentatsioon, mis tekitab arendus raskemaks ja võtab rohkem aja seda õpetamiseks [41].
- Silumine, React native on Beta versioonis, ning arendamise käigul võib tekkida palju raskused ja probleemid [43].
- React Native rakenduse arendus on vajab natiiv rakenduse arendaja arendatud rakendusi ülevaatamiseks, kui mõned funktsionaalsused vajavad selle platvormi üksikasjalikke teadmisi ja kontseptsioone [43].
- Litsentside ja patentide vaidlus, kasutate avatud lähtekoodiga projekte patendi väljastamiseks, saab Facebook teie juurdepääsu blokeerida [43].
- React Native on uuemate funktsioonide lisamisel aeglane. Praegu on tehnoloogiliste edusammude aeg ja me näeme seadme iga uue mudeli puhul mõnda funktsiooni. Uue seadme turule toomine lisas palju tarkvaraga seotud uusi funktsioone. React Native lisab nende funktsioonide toe, kuid see võtab kaua aega või vajate selle konkreetse funktsiooni kasutamiseks natiiv tuge [43].

3.2.1.3 Flutter

Flutter on Google'i loodud tasuta ja avatud lähtekoodiga mobiilrakenduse arendamiseks raamistik, mis ilmus 2017. aasta mais. Flutter raamistik on ehitatud Dart programmeerimise keeles, mis Google toetas oktoobris 2011 [44].

Flutter on mobiilirakenduse SDK suure jõudlusega ja täpsusega iOS ja Androidi seadmetele rakenduste loomiseks ühest koodibaasist. Flutter sisaldab kaasaegset reageerimisstiilis raamistikku, 2D-renderdusmootorit, valmis vidinaid ja arendustööriistu. Need komponendid töötavad koos, et aidata arendajale rakendusi kujundada, luua, testida ja siluda. Flutter raamistikus kõik komponendid on Widget. Widget on Flutter raamistiku peamised vaadete ehitusplokid, ning iga Widget on vaadete osa muutumatu deklaratsioon. Erinevalt teistest raamistikust, vaatekontrollereid, paigutusi ja muid omadusi eraldavatest raamistikest on Flutter raamistikul on ühtne ja ühtne objektimudel: Widget. Widget komponendid moodustavad kompositsiooni alusel

hierarhia. Iga Widget pesitseb sees ja pärib omadused oma vanemalt. Eraldi "rakenduse" objekti pole. Selle asemel täidab seda rolli põhine Widget. Sündmustele reageerimiseks, näiteks kasutaja interaktsioonile, käskides raamistikul asendada hierarhias olev vidin mõne teise vidinaga. Seejärel võrdleb raamistik uusi ja vanu vidinaid ning värskendab tõhusalt kasutajaliidest. Widget komponendi muutmisel Flutter üle ehitab kõik vaated, oma kiiruse ja jõudu tõttu see toimub väga kiire ja saab kasutajatele nähtumata [46].

Widget komponente saab ise arendama ja kasutata.

Flutter raamistiku eelised:

- Oma viimases väljaandes on Flutter pakub Windowsi, macOS, Linuxi ja veebi rakendusi tuge ühe koodibaasi kasutamisel [47].
- Flutter raamistik aastate jooksul arendas arendajate toetamiseks tööriistu, rakenduse «hot-reload», Flutter Inspector ja teised töristad muudab silumise ja arendamise käik palju lihtsamaks [45].
- Flutter raamistiku õpperessursid ja platvormi dokumentatsioon konkureerivad mõne parimaga, et aidata platvormi uutel arendajatel väikese viivitusega tööle saada [47].
- Kui tavapärased raamistikud ehitavad vaadet tavaliselt Native koodiks, siis Flutter loob platvormi jaoks oma kasutajaliidese komponendid, muutes selle kasutajaliidese koodi usaldusväärsemaks ja kiiremaks kui enamik konkureerivaid rakendusraamistikke [45].
- Flutter toob väga hea ühildus erinevatele platvormidele, ekraani suurustele ja seadmetele ühe koodibaasiga. Paljudel juhtudel Flutter võimaldab nii hästi, et enamik raamistikke sellega konkureerida ei suuda [47].

Flutter raamistiku puudused:

- Flutter on veel uus ja suhteliselt testimata platvorm. Palju Flutter raamistiku toetavad raamistikud ja tekid on veel varasemas arenduses, mis ei ole väga veakindel [45].
- Ei saa rakendada Flutter raamistikuga rakendust ilma Dart programmeerimiskeelena, Dart suhteliselt ebaküps tehnoloogia. Kuigi keel on nüüdseks kümme aastat vana, areneb ja muutub see endiselt kiires tempos. Kiired muutused ja ebastabiilsuse tunne keeles võivad muuta pikaajalise hoolduse olulisemaks väljakutseks [45].
- Kuigi Flutter raamistik ehitab vaadet oma komponentidega, see toob kaasa mõned suurimad tugevused, põhjustab see et rakenduste ootuspärase välimuse ja tunde, eriti mobiilsete platvormide puhul [47].

4 Tehnoloogiate valimine

Tehnoloogia valimisel tuleb võtta arvesse järgnevad tegureid:

- Autori tehnoloogia kasutamise kogemus
- Nõude täitmine
- Edasiarendamise võimalused
- Dokumentatsioon kvaliteet

4.1 Serveripoole tehnoloogiate valimine

Selles peatükis valitakse serveripoolsed tehnoloogiad arendamiseks, ning iga valik põhjendatakse.

4.1.1 Andmevahetuse viisi valimine

Veebitarkvaraarhitektuurina on valitud REST (Representational State Transfer) arhitektuur, mis võimaldab serveripoolse ja kliendipoolse rakenduse vahel infot edastada. Andmevahetuse formaadina on valitud JSON (JavaScript Object Notation), kuna see on paremini sobiv edasirakenduseks ja skaleeritavuseks. REST arhitektuur on valitud, kuna see on lihtsam ja kiirem arendamiseks võrreldes SOAP (Simple Object Access Protocol) põhise juurdepääsuprotokolliga. Lisaks sellele, REST on laialdaselt kasutusel uutes projektides, mis tähendab, et tööjõu leidmine on lihtsam ja odavam. Autor omab ka rohkem kogemusi REST arhitektuuriga rakenduste arendamisel võrreldes SOAP juurdepääsuprotokollipõhiste rakendustega.

4.1.2 Raamistiku ja programmerimise keele valimine

Serveripoolse rakenduse raamistiku ja keelena on valitud Spring Boot koos Javaga. Autori arvates ei sobi Node.js raamistik oodatava mahuga rakenduse edasiseks arendamiseks, kuna see on pigem mõeldud väikeste rakenduste jaoks. Lisaks on JavaScript programmeerimiskeel tuntud oma ebausaldusväärsuse ja kõrgete hoolduskulude poolest. Suuremahuliste projektide arendamine JavaScripti keeles võib olla keeruline, aeganõudev ja ebausaldusväärne tulevikus.

Django raamistik koos Pythoniga ei sobi autorile, kuna tal puudub sellega seotud kogemust. Valiku tegemine .NET ja Spring Boot raamistike vahel oli raske, kuna need tehnoloogiad on põhimõtteliselt võrdsed ning kasutavad kaasaegseid, usaldusväärseid programmeerimiskeeli ning autoril on nendega võrdne kogemus. Siiski on Spring Boot raamistikuga arendus lihtsam ja kiirem paigaldamiseks, ning on kiirem API ressursside arendamise osas võrreldes .NET raamistikuga. Spring Boot kaasas olev Spring Data, mis võimaldab andmebaasiga suhtlemist, on mugavam ja kergem kasutada kui .NET raamistikuga pakutav Entity Framework. Java programmeerimiskeel on Eestis laialdaselt kasutusel, mis tähendab, et arendaja leidmine on lihtsam ja odavam. Lisaks on .NET raamistik noorem kui Spring Boot ning seetõttu võib see olla vähem usaldusväärne ja halvemini testitud.

4.1.3 Andmebaasi haldussüsteemi valimine

Kuna MySQL, MariaDB ja PostgreSQL andmebaasi haldussüsteemid on võrdselt sobivad kirjeldatud rakenduse arendamiseks, on autor otsustanud valida PostgreSQL. See valik on tingitud autori vähemate kogemustega MySQL ja MariaDB haldussüsteemidega.

PostgreSQL andmebaasi haldussüsteemid on usaldusväärne ja laialdaselt kasutatav avatud lähtekoodiga andmebaasihaldussüsteem, mis pakub võimsaid funktsioone ja hea jõudluse. Autor otsustas valida PostgreSQL haldussüsteem, et tagada parim võimalik tugi rakenduse arendamisel andmebaasiga seotud funktsionaalsuse ja nõuete jaoks.

4.2 Kliendipoolsete tehnoloogiate valimine

Selles peatükis valitakse klienti poolsed tehnoloogiad arendamiseks, ning iga valik põhjendatakse.

4.2.1 Mobiilirakenduse arendamise raamistiku valimine

Vaatamata sellele, et Ionic raamistik on vanem kui teised kirjeldatud raamistikud, pole see saavutanud sama suurt populaarsust. Kuna Ionic võimaldab luua rakendusi, mis töötavad erinevate JavaScripti programmeerimiskeeltega ühe koodibaasi kasutamisel, võib seda mõistlikuks pidada olemasoleva veebirakenduse puhul. Kuna veebirakendust

pole ja tehnoloogia ise on aeglane; ning kasutab JavaScripti programmeerimiskeelena, mis on ebausaldusväärne ja raskendab arendustööd, otsustas autor sellest loobuda.

React Native ja Flutter raamistike vahel valiti Flutter, kuna autoril on rohkem kogemusi Flutter raamistiku kasutamisega rakenduste arendamisel võrreldes React ja React Native raamistikega.

5 Süsteemi arendus

Käesolevas peatükkis on kirjeldatud kasutavate tehnoloogiad ja rakenduse struktuur. Arendus on jagatud kahe peamise peatükki: serveripoolse rakenduse arendus ja kliendipoolse rakenduse arendus

5.1 Serveripoolse rakendus

Serveripoolse rakenduse ülesanne on võimelda äriloogikat, andmete töötlemist, andmete hoidmist ja HTTP päringu töötlus.

Käesolevas peatükkis on kirjeldatud kliendipoolse rakenduse arendamine.

5.1.1 Rakenduse loomine

Esialgne projekt loodi IntelliJ IDEA rakenduse genereerimisfunktsiooni abil. Projekti generaatormeenusena valiti Spring Initializr ning programmeerimiskeelena valiti Java koos Gradle projekti mootoriga. Projekti JDK versioonina valiti Oracle poolt pakutav OpenJDK 15, milleks on Java 19 versioon. Projekti kasutuselevõtuks valiti JAR fail.

Spring Boot versiooniks valiti 3.0.3. Algse projekti teekidena paigaldati Lombok, Spring Web, Spring Data JPA, JDBC API, PostgresSQL Driver, Rest Repositories ja Liquibase Migration.

Genereerimise pärast on loodud demo rakendus.

5.1.2 Andmebaasi struktuur ja selle mudelid

Andmebaasimudelite koostamiseks ja kirjeldamiseks on valitud ERD (Entity Relationship Diagramm) kuju. Selle koostatamiseks on valitud SqlDBM, mis asub aadressil „sqldbm.com“, veebilehega pakutatud töörist.

ERD tähistab Entity-Relationship Diagram ehk E-R diagramm. See on visuaalne esitus andmebaasi struktuurist, kus esitatakse seosed erinevate entiteetide (andmeobjektide) vahel. ERD kasutatakse andmebaasi kavandamisel ja disainimisel, et näidata andmebaasi skeemi, entiteetide omavahelisi suhteid, atribuute ja võtmeid [51].

Joonis 10. Andmebaasi struktuur.

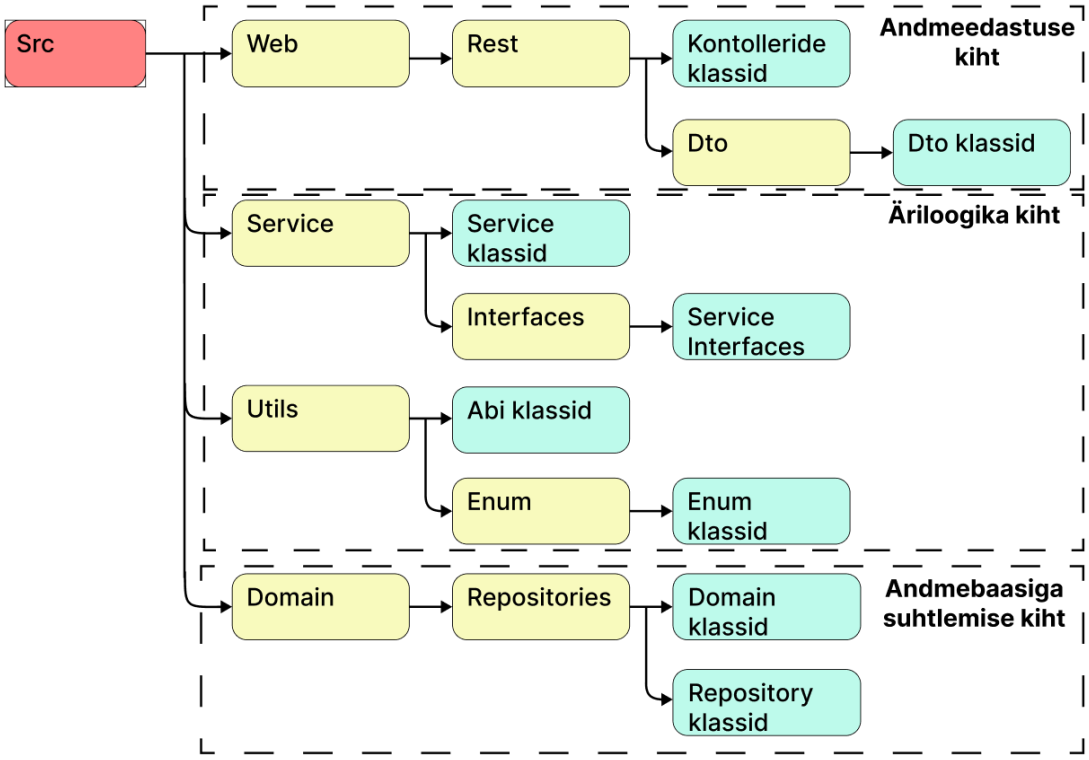
Serveripoolse rakenduse jaoks andmebaasiga suhtlemiseks on kasutusel Spring Data on abiraamistik, mis muudab andmetele juurdepääsu tehnoloogiate, relatsiooniliste ja mitterelatsiooniliste andmebaaside, kaartide vähendamise raamistike ja pilvepõhiste andmeteenuste kasutamise lihtsamaks [49].

Spring Data võimaldab arendajatel hõlpsalt töötada andmebaasiga, pakkudes automatiseeritud andmetabelite loomist, muutmist, lugemist, loomist ja kustutamist ilma SQL-päringute käsitsi kirjutamiseta. Kui arendajatel on vaja kasutada oma kohandatud SQL-päringuid, pakub Spring Data selleks ka tuge.

5.1.3 Rakenduse arhitektuur

Rakendus koosneb kolmest osast: andmeedastuse kiht, äriloogika kiht ja andmebaasiga suhtlemise kiht.

Allpool on kirjeldatud rakenduse struktuur:



Joonis 11. Serverpoolse rakenduse arhitektuur.

5.1.3.1 Andmebaasiga suhtlemise kiht

Andmebaasi kiht on rakenduse osa, mis vastutab kõigi andmebaasipäringute käsitlemise eest. Selle kihi ülesanne on andmete salvestamine ning andmebaasist päringute tegemine, tagades samal ajal, et kasutatavad andmed ja toimingud oleksid lubatud. See kiht tagab kindluse, et kui on vajadus muuta andmebaasi süsteemi, saab seda teha arendades vastavalt konkreetsele andmebaasi süsteemile.

Rakenduses kasutatakse Repository mustrit, mis on strateegia andmetele juurdepääsu saamiseks. Seega koosneb andmetele juurdepääs rakenduse koodist, mis tegeleb andmete salvestamise ja hankimisega [48].

Repository tugi on toodud Spring Data JPA tekkega.

Spring Data JPA on moodul, mis lihtsustab JPA-põhiste hoidlate rakendamist Spring raamistikus. See moodul pakub täiustatud tuge JPA-põhiste andmepääsukihtidele, muutes seeläbi andmetele juurdepääsu tehnoloogiaid kasutavate Spring raamistikuga rakenduste arendamise lihtsamaks. Spring Data JPA eesmärk on oluliselt parandada andmetele juurdepääsu kihtide rakendamist, vähendades vajalikku vaeva miinimumini. Arendajana saate kirjutada oma hoidlate liidesed, sealhulgas kohandatud otsingumeetodid, ning Spring pakub automaatset juurutamist [50].

5.1.3.2 Service kiht

Service kiht, ehk äriloogika kiht, on loodud selleks, et hoida rakenduse kogu äriloogikat. See kiht sisaldab kõiki rakendusega seotud loogilisi toiminguid. Äriloogika kiht kasutab andmebaasiga suhtlemise kiht, vastava Repository kaudu, et küsida andmeid, kontrollida nende õigsust, luua uusi või muuta olemasolevaid objekte ning teostada arvutusi ja anda moodustatud andmeid teistele rakenduse osadele. Läbi äriloogika kihi tehakse kõik rakenduse äriloogika operatsioonid.

5.1.3.3 Andmeedastuse kiht

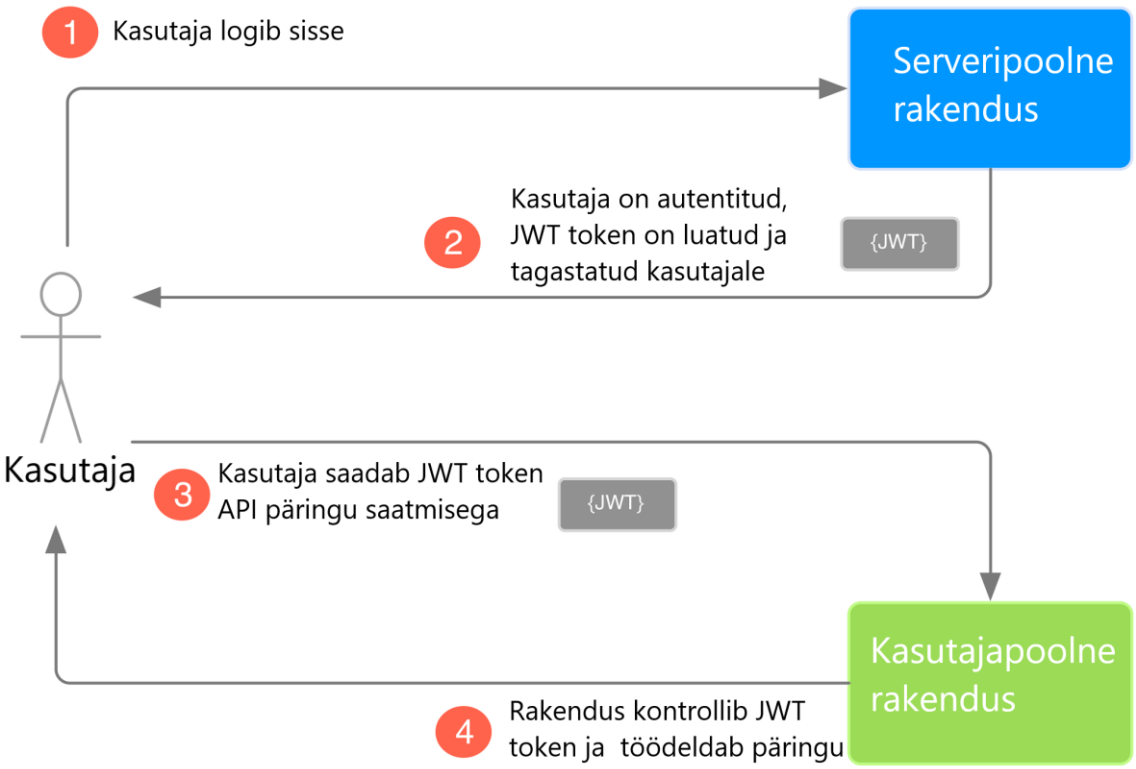
Andmeedastuse kiht on rakenduse kõige välisem kiht, mis pakub tuge rakenduse kasutajaga suhtlemisele. Rakendus on loodud REST API arhitektuuri põhjal ning selleks on loodud erinevad kontrollerid, mis esindavad erinevaid API lõpp-punkte. Klient saadab HTML päringu URI (ühtne ressursiidentifikaator) alusel kontrolleri meetodisse, kus päringu saadetud andmed võetakse vastu, kontrollitakse andmete õigsust ja edastatakse äriloogika kihile. Äriloogika kiht töötleb andmeid ja tagastab vastuse, millele kontroller teostab vastavat kujundamist ning saadab selle tagasi kliendile. Andmeedastuse kiht ei sisalda äriloogikat, vaid võtab vastu päringuandmed, kontrollib nende õigsust, edastab need äriloogika kihile ning saadab tagasi töödeldud vastuse. Loodetavas rakenduses andme vastu võtmiseks ja tagasi saatmiseks on kasutud JSON formaadil kujulise päringud, mis kaartitatakse vajalikku objektiks

5.1.4 JSON Web märk

Rakenduse autentimise tugi on seadistatud JWT (JSON Web Token) läbi. See on standard, mis kasutatakse rakenduse juurdepääsulubade loomiseks. Seda peetakse üheks

turvalisemaks viisiks teabe edastamiseks kahe osaleja vahel. Selle loomiseks tuleb määratleda päis, kus on üldine teave märgi kohta, kasulikud andmed, näiteks kasutaja ID, roll ja allkirjad.

1. Esiteks logib kasutaja autentimisserverisse sisse autentimisvõtmega.
2. Seejärel loob autentimisserver JWT ja saadab selle kasutajale.
3. Kui kasutaja esitab taotluse rakenduse API-le, lisab ta sellele eelnevalt saadud JWT.
4. Kui kasutaja esitab API päringu, saab rakendus kontrollida päringuga edastatud JWT-d, et näha, kas kasutaja on see, kes ta end olevat.



Joonis 12. JSON Web Token töötamise printsiip.

5.2 Kliendipoolse rakenduse arendus

Kliendipoolse rakenduse ülesanne on võimelda kasutajaga ja serveripoolse rakendusega vahel suhtlemist, HTTP päringu saatmist, ning andmete näitamist

Käesolevas peatükkis on kirjeldatud kliendipoolse rakenduse arendamine.

5.2.1 Rakenduse loomine

Rakenduse põhjuse oli loodud Flutter käsuliidu kaudu.

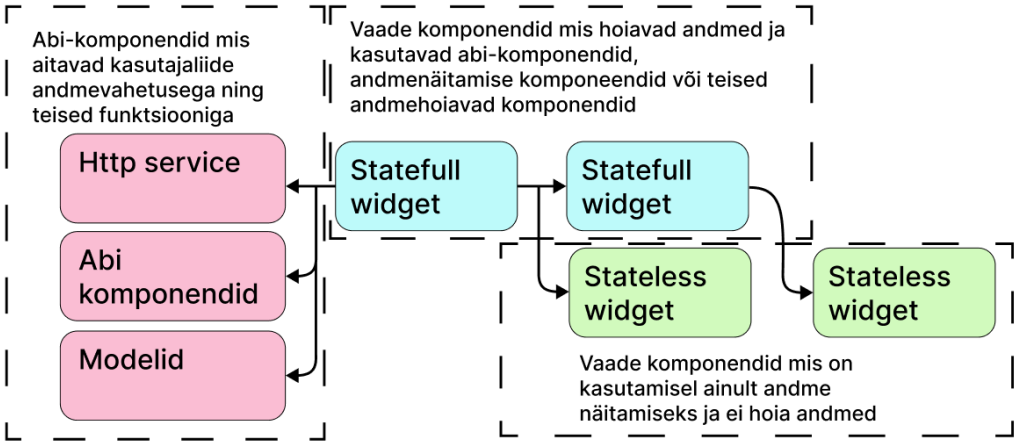
5.2.2 Kliendipoolse rakenduse struktuur

Käesolevas peatükkis on kirjeldatud kliendipoolse rakenduse struktuur

Kliendipoolne rakenduse struktuur on modulaarne. Rakendus on arendatud kasutades erinevad funktsioonitüübi moodulid: vaadeehitamise ja äriloogika moodulid. Vaade ehitamiseks on kasutatud Widget klassid, mis loob ja näitab kasutajatele kasutajaliide vaated, need võivad olla kas Stateless, ehk seisulik, või Statefull, ehk seisulikuta. Stateless vaade ja Statefull vaade erinevus on selles, et esimesed ei hõivavad andmed, vaid lihtsalt ette näitavad nendele antud andmed. Aga Statefull vaated pakkuvad tugi andme hoidmiseks ja seda muutmiseks. Kasutajaliide vaated on ehitatud nende kahe tüüpi kombineerimas, seal kus andmed ja vaade tuleb muudeta kasutaja tegemise tõttu - kasutati Statefull, seal kui vaade ainult näitab andmed – kasutati Stateless. Enamik Stateless vaadetes on saab kasutada erinevates Statefull vaadetes, selle mugavuse ja laiendamise tõttu ei tuleb iga korda uut vaadet loa, vaid saab kasutata juba oleva komponendi.

Rakendus sisaldab vaade ehitamise klasside peale, on abi klassid mis toob REST päringu saatmise tugi. Rakenduse abi komponendid on Enum klassid, abi klassid ja teised ehitatud laiendused mis saab kasutada mis tähes milles rakenduse osas. Mille tõttu rakendusel on väiksem koodi dublikaati ja selle abi komponendi funktsionaalsusi saab lihtsalt ühes kohas muuta.

Vaade ehitamiseks on kasutatud Widgeti struktuur. Vaade struktuuri kirjeldusena on toodud näide.



Joonis 13. Kliendipoolse rakenduse struktuur.

5.2.3 Asünkroonsete päringute andmete kuvamine

Flutter pakub tulevikuobjektina objekti kuvamiseks asünkroonsetest funktsioonidest kasutada FutureBuilder klassi.

FutureBuilder on Flutteri raamistikus pakutav Widget, mida kasutatakse dünaamilise sisu kuvamiseks, mis on saadaval tulevikuobjektina, ehk objekt mis ei ole kohe saadaval, vaid saab mõni ajapärast. FutureBuilder võimaldab reageerida Future raamistiku olekutele, nagu "ootamine", "valmis" või "veaga lõpetatud", ning vastavalt sellele värskendada kasutajaliidest. FutureBuilder võtab sisendiks tulevikuobjekti ja funktsiooni, mis võtab vastu BuildContext'i ja AsyncSnapshot. AsyncSnapshot sisaldab teavet tulevikuobjekti oleku kohta, nagu andmed, veateated või muud olekuga seotud andmed. Funktsioon võimaldab defineerida, millist kasutajaliidese osa tuleks kuvada erinevate tulevikuobjekti olekute korral. Näiteks saab määrata, millist laadimisikooni kuvada, kui tulevikuobjekt on ootel, või millist sisu kuvada, kui Future on valmis [52].

Selleks, et standardiseerida REST päringute kuvamist, otsustati luua ühtne lahendus, mis näitaks laadimise olekut, ning käsitleks vigu ühtemoodi. Selle saavutamiseks, kuna erinevate päringute jaoks on vaja erinevat kuvamist, otsustati luua üldine üldine FutureBuilder klassi, mis saaks lisaks tulevikuobjektile võtta vastu ka tulevikuobjekti tüübi ning funktsiooni vaade loomiseks juhul, kui päring õnnestub.

```dart
import 'package:flutter/material.dart';

class CustomFutureBuilderWidget<T> extends StatelessWidget {
  const CustomFutureBuilderWidget(
      {Key? key, required this.futureCall, required this.createWidgetFunc})
      : super(key: key);

  final Future<T> futureCall;
  final Widget Function(T obj) createWidgetFunc;

  @override
  Widget build(BuildContext context) {
    return FutureBuilder<T>(
      future: futureCall,
      builder: (BuildContext context, snapshot) {
        if (snapshot.hasData) {
          return createWidgetFunc(snapshot.data!);
        } else if (snapshot.hasError) {
          return Text(snapshot.error.toString());
        } else if (snapshot.connectionState == ConnectionState.waiting) {
          return const Center(child: CircularProgressIndicator());
        } else if (snapshot.connectionState == ConnectionState.none) {
          return const Text('cannot establish connection with the server.');
        }
        return const Center(child: CircularProgressIndicator());
      },
    );  // FutureBuilder
  }
}
```

Joonis 14. Üldine FutureBuilder klass

Enne klassi kasutamist tuleb määrata olekuks tulevikuobjekt, Flutteri puhul need tuleb määrata initState() funktsioonis, mis kuulub State klassi. Samuti tuleb arendada vaate ehitamise funktsioon, mis annatakse Üldise FutureBuilderi klassile käivitamiseks, kui päring on edukalt täidetud.

Üldine FutureBuilder klass võtab vastu tulevikuobjekti, selle tüübi ja funktsiooni vaate ehitamiseks. Tulevikuobjektist vastuse saamiseks kasutab klass FutureBuilderit. Seni, kuni päringut pole täidetud, kuvatakse vaateks laadimisikooni, vea korral kuvatakse veasõnumi. Kui päring õnnestub, siis täidab klassiile antud funktsiooni, mis loob ja tagastab vaate.

```dart
class _CustomerDeliveryWidgetState extends State<CustomerDeliveryWidget> {
  late Future<List<MarketOrderListDto>> _marketOrderListFuture;

  @override
  void initState() {
    String token = Provider.of<AppProvider>(context, listen: false).token;
    _marketOrderListFuture = OrderFetch.fetchOrderList(token);

    super.initState();
  }

  @override
  Widget build(BuildContext context) {
    return CustomFutureBuilderWidget(
        futureCall: _marketOrderListFuture, createWidgetFunc: buildView);
  }

  Widget buildView(List<MarketOrderListDto> dto) {
    return RefreshIndicator(
      onRefresh: update,
      child: Container(
        margin: EdgeInsets.all(10),
        width: double.infinity,
        child: MarketOrderListWidget(
          orderList: dto,
          itemCreatingFunc: (MarketOrderListDto orderDto) {
            return CustomerMarketOrderCardWidget(orderDto: orderDto);
          },
        ),  // MarketOrderListWidget
      ),  // Container
    );  // RefreshIndicator
  }
```

Joonis 15. Üldise FutureBuilder klassi kasutamine

6 Testimine

Enamik rakenduse testimisest on läbi viidud arendusprotsessi käigus. Uute funktsionaalsuste loomise järel on tehtud nende testimist arendaja poolt. Uute serveripoolsete API lõpppunktide funktsionaalsus on enne kliendipoolse toe loomist testitud Postman rakenduse abil, mis aitab päringuid saata. Pärast edukat testimist Postmaniga ja kasutajaliidese funktsionaalsuse loomist on tehtud ka kasutajaliidese testimine, kus erinevaid situatsioone on testitud käsitsi. Igasuguse vigu ilmnemisel on need tagastatud vastavatele koodifragmentidele parandamiseks.

Käsitsi testimise kõrval on läbi viidud ka testimine teiste inimeste abiga. Testis osales 10 inimest, kellele anti rakendus koos seadmetega ülesannete testimiseks. Enamikul testijatest ei tekkinud ülesande täitmisel raskusi, kuid märgiti, et vanematele inimestele, kes ei kasuta sageli toidu tellimise rakendusi nagu Wolt ja Bolt Food, võib rakenduse kasutajaliidese navigeerimine võtta aega selle mõistmiseks. Suuri probleeme ei tekkinud ja kõik osalejad suutsid ülesanded edukalt lõpule viia.

Tagasiside oli positiivne ja saadi palju kommentaare rakenduse parandamiseks, alates lihtsustatud disainist kuni uute funktsionaalsuste lisamiseni. Kuigi rakendus on alles MVP vormis, on juba küsitud kasutamise alustamise võimalust. Tagasiside põhjal saab rakendust edasi arendada ja parandada.

7 Kokkuvõte

Käesoleva lõputöö eesmärk oli luua lihtne MVP (minimaalse elujõulise toote) rakendus talutoodete müümiseks ja ostmiseks ning seda prototüüpi testimise eesmärgil kasutada.

Töö autor lõi MVP rakenduse, mille kaudu saavad kasutajad administreerida talupoode, lisada kaupu, vormistada tellimusi, jälgida tellimuste staatust, vaadata oma kontoandmeid ja statistikat. Lõputöö eesmärgi saavutamiseks teostati analüüs, kus autor uuris olemasolevaid lahendusi, et määrata rakenduse funktsionaalsus ja ulatus. Lisaks loodi kasutusjuhtude mudel, kus kirjeldati igat kasutusjuhtu ja selle toiminguid. Analüüs tehti ka erinevate tehnoloogiate kohta, mida süsteemi arendamisel kasutada. See tagas, et loodav rakendus täidab kirjeldatud ülesannet ja on tulevikus edasi arendatav.

Loodud rakendus on alus edasiseks arenduseks. Autor soovib tulevikus parandada olemasolevat ja lisada uut funktsionaalsust, lähtudes saadud tagasisidest ning proovida rakendust turule viia.

Kasutatud kirjandus

[1] "Comparison Between Web Services", [Võrgumaterjal]. Saadaval: [https://www.guru99.com/comparison-between-web-services.html]. [14 veebruar 2023].

[2] "Best Practices for API Design in Azure Architecture", [Võrgumaterjal]. Saadaval: [https://learn.microsoft.com/en-us/azure/architecture/best-practices/api-design]. [14 veebruar 2023].

[3] "OMG/DARPA/MCC Position Paper on the Integration of Different Distributed Object Computing Systems", [Võrgumaterjal]. Saadaval: [https://www.python.org/doc/essays/omg-darpa-mcc-position/]. [17 veebruar 2023].

[4] "Advantages of Django Framework", [Võrgumaterjal]. Saadaval: [https://blog.back4app.com/backend-frameworks/#Advantages_Of_Django]. [17 veebruar 2023

[5] "Pros and Cons of Django Framework for App Development", [Võrgumaterjal]. Saadaval: [https://dzone.com/articles/pros-and-cons-of-django-framework-for-app-developm]. [17 veebruar 2023].

[6] "Django Framework Guide: How to Build a Web Application with Django", [Võrgumaterjal]. Saadaval: [https://careerfoundry.com/en/blog/web-development/django-framework-guide/]. [17 veebruar 2023].

[7] "What is Express.js? A Beginner's Guide", [Võrgumaterjal]. Saadaval: [https://intellipaat.com/blog/what-is-express-js/?US]. [17 veebruar 2023].

[8] "What Are the Benefits of Using Express.js for Backend Development?", [Võrgumaterjal]. Saadaval: [https://www.techomoro.com/what-are-the-benefits-of-using-express-js-for-backend-development/]. [19 veebruar 2023].

[9] "What is Express.js? An In-Depth Overview", [Võrgumaterjal]. Saadaval: [https://www.besanttechnologies.com/what-is-expressjs]. [19 veebruar 2023].

[10] "Express.js Advantages and Disadvantages", [Võrgumaterjal]. Saadaval: [https://data-flair.training/blogs/expressjs-advantages-and-disadvantages/]. [19 veebruar 2023].

[11] "Everything You Need to Know About C#", [Võrgumaterjal]. Saadaval: [https://www.pluralsight.com/blog/software-development/everything-you-need-to-know-about-c-]. [1 marts 2023].

[12] "What is C#?", [Võrgumaterjal]. Saadaval: [https://www.c-sharpcorner.com/article/what-is-c-sharp/]. [1 marts 2023].

[13] "Introduction to ASP.NET Core", [Võrgumaterjal]. Saadaval: [https://dotnettutorials.net/lesson/introduction-to-asp-net-core/]. [1 marts 2023].

[14] "ASP.NET Core 8: Pros and Cons", [Võrgumaterjal]. Saadaval: [https://ukad-group.com/blog/aspnet-core-8-pros-and-cons/]. [1 marts 2023].

[15] "Pros and Cons of .NET Core", [Võrgumaterjal]. Saadaval: [https://medium.com/@darshanunadkat67/pros-and-cons-of-net-core-37ec451edd0]. [1 marts 2023].

[16] "Advantages and Disadvantages of ASP.NET", [Võrgumaterjal]. Saadaval: [https://www.software-developer-india.com/advantages-and-disadvantages-of-asp-net/]. [3 marts 2023].

[17] "Java", [Võrgumaterjal]. Saadaval: [https://www.ibm.com/topics/java]. [4 marts 2023].

[18] "What is the JVM?", [Võrgumaterjal]. Saadaval: [https://topjava.ru/blog/what-is-the-jvm]. [6 marts 2023].

[19] "Pros and Cons of Using Spring Boot", [Võrgumaterjal]. Saadaval: [https://bambooagile.eu/insights/pros-and-cons-of-using-spring-boot/]. [6 marts 2023].

[20] "6 Advantages and Disadvantages/Limitations/Benefits of Spring Boot", [Võrgumaterjal]. Saadaval: [https://www.hitechwhizz.com/2022/08/6-advantages-and-disadvantages-limitations-benefits-of-spring-boot.html]. [14 marts 2023].

[21] "5 Advantages and Disadvantages/Drawbacks/Benefits of Spring Framework", [Võrgumaterjal]. Saadaval: [https://www.hitechwhizz.com/2022/08/5-advantages-and-disadvantages-drawbacks-benefits-of-spring-framework.html]. [14 marts 2023].

[22] "Advantages and Disadvantages of Using Spring Boot", [Võrgumaterjal]. Saadaval: [https://diliru.medium.com/advantages-and-disadvantages-of-using-spring-boot-a1debab70d5e]. [14 marts 2023].

[23] "An Overview of MySQL", [Võrgumaterjal]. Saadaval: [https://www.linode.com/docs/guides/an-overview-of-mysql/]. [15 marts 2023].

[24] "Advantages and Disadvantages of Using MySQL", [Võrgumaterjal]. Saadaval: [https://diliru.medium.com/advantages-and-disadvantages-of-using-mysql-36f6ffce3fa3]. [20 marts 2023].

[25] "5 Advantages and Disadvantages/Limitations/Benefits of MySQL", [Võrgumaterjal]. Saadaval: [https://www.hitechwhizz.com/2022/10/5-advantages-and-disadvantages-limitations-benefits-of-mysql1.html]. [20 marts 2023].

[26] "Open Source Databases", [Võrgumaterjal]. Saadaval: [https://opensource.com/article/19/1/open-source-databases]. [20 marts 2023].

[27] "MySQL vs PostgreSQL", [Võrgumaterjal]. Saadaval: [https://cloudinfrastructureservices.co.uk/mysql-vs-postgresql/]. [20 marts 2023].

[28] "MariaDB Overview", [Võrgumaterjal]. Saadaval: [https://www.openlogic.com/blog/mariadb-overview]. [20 marts 2023].

[29] "Introduction of MariaDB", [Võrgumaterjal]. Saadaval: [https://www.geeksforgeeks.org/introduction-of-mariadb/]. [20 marts 2023].

[30] "Understanding MariaDB Architecture", [Võrgumaterjal]. Saadaval: [https://mariadb.com/kb/en/understanding-mariadb-architecture/]. [20 marts 2023].

[31] "MariaDB vs PostgreSQL: Which Database Should You Use?", [Võrgumaterjal]. Saadaval: [https://kinsta.com/blog/mariadb-vs-postgresql/]. [20 marts 2023].

[32] "MariaDB vs MySQL: What's the Difference?", [Võrgumaterjal]. Saadaval: [https://www.hostinger.com/tutorials/mariadb-vs-mysql]. [25 marts 2023].

[33] "PostgreSQL vs MariaDB: Which Database Should You Choose?", [Võrgumaterjal]. Saadaval: [https://www.openlogic.com/blog/postgresql-vs-mariadb]. [27 marts 2023].

[34] "MariaDB vs MySQL: Choosing the Right Database", [Võrgumaterjal]. Saadaval: [https://www.singlestore.com/blog/mariadb-vs-mysql/]. [27 marts 2023].

[35] "About PostgreSQL", [Võrgumaterjal]. Saadaval: [https://www.postgresql.org/about/]. [27 marts 2023].

[36] "What is PostgreSQL? An Introduction, Advantages, and Disadvantages", [Online]. Saadaval: [https://www.linkedin.com/pulse/what-postgresql-introduction-advantages-disadvantages-ankita-sharda/]. [27 marts 2023].

[37] "Ionic Framework: Pros and Cons", [Võrgumaterjal]. Saadaval: [https://hackr.io/blog/ionic-framework]. [27 marts 2023].

[38] "Introduction to Ionic", [Võrgumaterjal]. Saadaval: [https://www.c-sharpcorner.com/article/introduction-to-ionic/]. [27 marts 2023].

[39] "Ionic Documentation", [Võrgumaterjal]. Saadaval: [https://ionicframework.com/docs#:~:text=Ionic%20is%20an%20open%20source,to%20learn%20the%20main%20concepts]. [2 aprill 2023].

[40] "What is Ionic and How Does It Work?", [Võrgumaterjal]. Saadaval: [https://appstronauts.co/blog/what-is-ionic-and-how-does-it-work/]. [2 aprill 2023].

[41] "Flutter vs React Native: сравнение фреймворков для разработки мобильных приложений", [Võrgumaterjal]. Saadaval: [https://habr.com/ru/articles/596183/]. [5 aprill 2023].

[42] "Learning React Native", [Võrgumaterjal]. Saadaval: [https://www.oreilly.com/library/view/learning-react-native/9781491929049/ch01.html]. [5 aprill 2023].

[43] "React Native Disadvantages You Should Consider Before Using It", [Võrgumaterjal]. Saadaval: [https://blog.back4app.com/react-native-disadvantages/]. [5 aprill 2023].

[44] "What is Flutter and Why You Should Learn It in 2020", [Võrgumaterjal]. Saadaval: [https://www.freecodecamp.org/news/what-is-flutter-and-why-you-should-learn-it-in-2020/]. [5 aprill 2023].

[45] "What is Flutter and What Are Its Advantages?", [Võrgumaterjal]. Saadaval: [https://www.adservio.fr/post/what-is-flutter-and-what-are-its-advantages]. [5 aprill 2023d].

[46] "What is Flutter? A Brief Introduction About Flutter", [Võrgumaterjal]. Saadaval: [https://medium.com/mobiosolutions/what-is-flutter-a-brief-introduction-about-flutter-40532f8af809]. [5 aprill 2023]

[47] "Best Practices for Flutter Development", [Võrgumaterjal]. Saadaval: [https://www.pangea.ai/dev-flutter-development-resources/best-practices/]. [5 aprill 2023

[48] "Repository Design Pattern", [Võrgumaterjal]. Saadaval: [https://medium.com/@pererikbergman/repository-design-pattern-e28c0f3e4a30]. [6 aprill 2023].

[49] "Spring Data", [Võrgumaterjal]. Saadaval: [https://spring.io/projects/spring-data]. [15 aprill 2023].

[50] "Spring Data JPA Reference Documentation", [Võrgumaterjal]. Saadaval: [https://docs.spring.io/spring-data/jpa/docs/current/reference/html/]. [15 aprill 2023].

[51] "What is Entity Relationship Diagram (ERD)? ", [Võrgumaterjal] Saadaval: [https://www.visual-paradigm.com/guide/data-modeling/what-is-entity-relationship-diagram/]. [15 mai 2023]

[52] "FutureBuilder<T> class ", [Võrgumaterjal] Saadaval: [https://api.flutter.dev/flutter/widgets/FutureBuilder-class.html]. [15 mai 2023]

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

Mina, Nikita Saprõkin

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "Talutoodete turu rakenduse arendus" , 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.

15.05.2023

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 – Klientrakenduse ja serveripoolse rakenduse versioonihaldus

Front-end: https://github.com/peqpeq/TTTurg/tree/main/ttturg_mobileBack-end

Back-end: https://github.com/peqpeq/TTTurg/tree/main/TTTurg