TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Ahto Jalak 202023IADB

Asukohapõhise kaupade hinnavõrdluse veebirakenduse väljatöötamine

Bakalaureusetöö

Juhendaja: Lembit Viilup
doktorikraad

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: Ahto Jalak

15.05.2023

Annotatsioon

Antud lõputöö eesmärgiks on luua veebirakendus, mis võimaldab kasutajal teha tarbekaupade hinnainfo otsinguid ning võrrelda erinevate poodide hinnapakkumisi. Rakenduse toimimiseks vajalik andmebaas kujuneb kasutajate tegevuse tulemusena ning sisaldab kauba triipkoodi, nime, kirjeldust, pilti, hinda ja kaupluse asukoha koordinaate.

Lõputöö raames analüüsitakse kogutud teemakohase kirjanduse baasil varasemate sarnaste lahenduste nõrku ja tugevaid külgi. Kogutud andmete tulemusel kirjeldati rakendusele omased ärinõuded ja funktsionaalsus. Lõputöö tulemusena loodi andmebaas ning pilveplatvormil töötav veebipõhine klient- ja tagarakendus.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 37 leheküljel, 5 peatükki, 16 joonist, 7 tabelit.

Abstract

Development of Location-based Product Price Comparison Web Application

The aim of this bachelor's thesis is to create a web application that allows users to search price information of consumer products and compare offers from different shops. The database of the application is formed as a result of user activity and contains product information such as barcode, name, description, image, product price and shop location coordinates.

This thesis will describe the business requirements and system specifications based on the analysis of the weaknesses and strengths of previous similar solutions on the basis of collected relevant literature. The database structure, web-based client application and back-end web application will be created and deployed to the cloud platform as a result of the thesis.

The thesis is in Estonian and contains 37 pages of text, 5 chapters, 16 figures, 7 tables.

Lühendite ja mõistete sõnastik

AKS	Azure Kubernetes Service
API	Application Programming Interface
AWS	Amazon Web Services – pilveteenuste pakkuja
CDK	Cloud Development Kit – tööriist ressursside kirjeldamiseks
CPC	Cost Per Click
CRUD	Create Read Update Delete
CSS	Cascading Style Sheets
CSV	Comma-Separated Values
DTO	Data Transfer Object – andmeedastusobjekt
ECDSA	Elliptic Curve Digital Signature Algorithm
ECR	Elastic Container Registry – hoidla
ECS	Elastic Container Service – AWS pakkutav konteinerteenus
EKS	Amazon Elastic Kubernetes Service
GPS	Global Positioning System
HMAC	Räsipõhine sõnumi autoriseerimiskood
HTML	HyperText Markup Language
JPA	Java Persistence API või Jakarta Persistence API
JSON	JavaScript Object Notation
JWT	JSON Web Token
MoSCoW	Ärinõuete prioritiseerimise metoodika
MVP	Minimum Viable Product
ORM	Object-relational mapping
REST	Representational State Transfer
RSA	Rivest–Shamir–Adleman algoritm
Scrum	Agiilse tarkvaraarendamise raamistik
SDLC	Software Development Lifecycle
UML	Unified Modeling Language
XML	eXtensible Markup Language

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Antud lõputöö eesmärk on luua rakendus, mis töötab internetiühendusega varustatud mobiilses seadmes ja võimaldab kasutajal teha tarbekaupade hinnainfo otsinguid ning võrrelda erinevate poodide hinnapakkumisi. Rakenduse toimimiseks vajalik andmebaas kujuneb kasutajate tegevuse tulemusena ning sisaldab kauba triipkoodi, nime, kirjeldust, pilti, hinda ja kaupluse asukoha koordinaate.

Lõputöö esimeses etapis analüüsitakse kogutud kirjanduse baasil varasemate sarnaste lahenduste nõrku ja tugevaid külgi. Kogutud andmete tulemusel kirjeldatakse rakendusele omased ärinõudeid, mis jagatakse loogilisteks osadeks ja kaardistatakse töö skoop. Analüüsi käigus töötatakse välja tingimused planeeritavatest arendustöödest ja protsessidest. Rakenduse väljatöötamise tulemusena valmib mobiilses seadmes kasutatav veebipõhine klient- ja tagarakendus, mis paigaldatakse pilvekeskkonda.

Lõputöö kirjalik osa koosneb sissejuhatusest, kolmest peatükist ja kokkuvõttest. Esimeses peatükis kirjeldatakse lõputöö eesmärk ja selgitatakse kasutatavaid metoodikaid. Teises peatükis püstitatakse planeeritavate arendustööde eesmärgid ja ärinõuded ning valitakse välja infosüsteemi realiseerimiseks kasutatavad tehnoloogiad. Kolmandas peatükis kirjeldatakse töö praktilist osa, milleks on andmebaasi ning klient- ja tagarakenduse loomine vastavalt püsitatud ärinõuetele.

2 Lõputöö eesmärk

Antud peatükis kirjeldatakse lõputöö eesmärk lähtuvalt probleemi taustast. Kirjeldatakse töös kasutatud metoodikad ja töö skoop.

2.1 Probleemi taust

Tarbijate huvi kasv kaasaegsete hinnavõrdlusrakenduste vastu on tingitud ühelt poolt interneti võimaluste suurenemisest ja teiselt poolt mobiilseadmete tehnoloogiliste võimaluste kiirest arengust. Tihti tehakse poes irratsionaalne tarbimisotsus, sest puudub piisav info turul toimuva kohta – paljud tarbijad ei soovi teha hinnainfo otsinguid enne ostude sooritamist või tundub see liiga keerukas ja ajakulukas.

USA-s läbiviidud tarbijate käitumisuuring tõi esile tarbijate madala hinnatundlikkuse, mis võib olla tingitud ka ebapiisavast informeeritusest turuhindade osas. Uuring tõi välja, et 37% tarbijaid külastavad ainult ühte poodi enne ostu sooritamist ja 19% tarbijatest külastab kahte poodi. Uuring näitas ka, et 32% tarbijaid eelistavad ühte kindlat kaubamärki, 26% valivad kahe kaubamärgi vahel ja 26% tarbijate valikus on kolm erinevat kaubamärki. Selgub, et paljud tarbijad ei soovi end siduda hinnavõrdlemise tegevusega, kuigi võiks eeldada, et see on tarbija enda huvides. Isegi kui tarbija soovib olla hinnateadlik ja uurib enne ostu sooritamist erinevate kauplejate pakkumisi, siis tihti puudub objektiivne arvamus mõistlikust hinnast või on tehtud otsinguid kindla kaubamärgi kohta. Uuringus tuuakse välja, et palju tarbijaid ei ole võimelised optimeerima oma ostuotsuseid olematu või raskesti kättesaadava lähteinformatsiooni tõttu. Võimaliku lahendusena tuuakse välja, et hõlpsalt kasutatav ja mobiilne hinnavõrdluse tööriist võib tarbijate eelarvamusi ja harjumusi muuta. [1]

On arusaadav, et kiiresti muutuvas ja suhteliselt killustatud infokeskkonnas on keeruline orienteeruda ja tülikas leida infot parajasti huvipakkuva kauba ning nendega kauplejate kohta piisavalt optimaalse aja jooksul. Info on tihti paljudes erinevates keskkondades laiali ja seetõttu tarbijal on keeruline saada head ülevaadet.

Eesti Konjunktuuriinstituudi poolt 2021. aastal läbi viidud uuringust selgub, et tarbijad on aja möödudes muutunud palju teadlikumaks ja informeeritumaks. Uuringus toodi esile, et 77% vastanutest uurib e-poest, sotsiaalmeediast või kataloogist kaupu ostes müüja kontaktandmeid. Valdav enamik tarbijatest (93%) uurib enne ostu sooritamist või teenuse kasutamist erinevate kauplejate ja teenuseosutajate pakkumisi ning reeglina valib neist soodsaima. Uuringus tuuakse välja, et probleemiks on eksitavad reklaamid ja ebaausad kauplemisvõtted, mille kasutamist kaupleja poolt on kogenud 73% vastanud tarbijatest. Samas hindab 80% vastanutest, et kauplejad annavad toodete ja teenuste kohta piisavalt müügieelset teavet. [2]

Olukorras, kus hinnad kasvavad kiires tempos ja järjest rohkem soovitakse leida soodsamaid pakkumisi, kasvab ka vajadus internetis toimiva mobiilse lahenduse järgi, mis võimaldab võrrelda erinevate kauplejate pakkumisi ühes keskkonnas, mille tulemusena soodustada tarbija võimet teha parim ostuotsuse ja nii säästa raha kui ka aega.

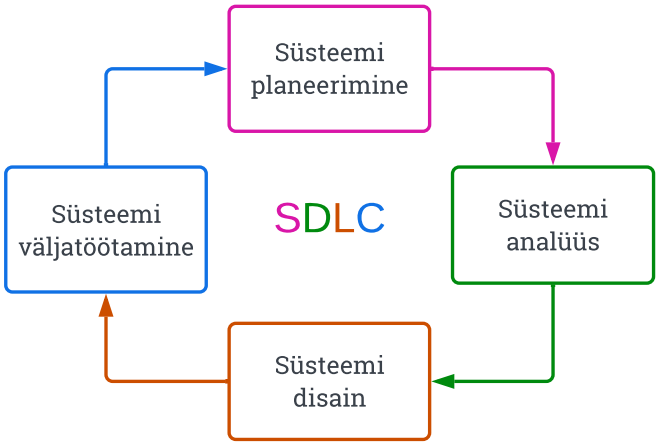
2.2 Eesmärk

Probleemi lahendamiseks luuakse internetis töötav veebirakendus, mida on mugav kasutada mobiiltelefonis. Rakendus peab olema võimeline teostama otsinguid ja pakkuma kasutajale infot läheduses asuvate poodide pakkumiste kohta. Sellise rakenduse loomiseks analüüsitakse ning kaardistatakse loodava rakenduse ärinõuded ja tingimused, mille raames kirjeldatakse rakenduse funktsionaalsus ja töö skoop eesmärgiga veebirakendus planeeritud kujul realiseerida. Antud lõputöö raames on kirjeldatud peamised tegevused, et luua lühikese aja jooksul rakenduse töötav versioon (MVP – *Minimum Viabile Product*). Rakenduse väljatöötamise käigus luuakse andmebaasi struktuur ning klient- ja tagarakendus. Rakenduse paigaldamiseks pilvekeskkonda on loodud automatiseeritud paigaldusprotsess.

2.3 Metoodika

2.3.1 Tarkvaraarenduse metoodika

Antud lõputöös kasutatakse agiilse tarkvaraarenduse põhimõtteid. Agiilsed meetodid võimaldavad paindlikku ja iteratiivset arendust ning kiiresti kohanduda muutuvate ärinõuetega. Tööprotsess, mida tuntakse ka kui tarkvaraarenduse elutsüklit (SDLC – *Software Development Life Cycle*), on efektiivne tarkvaratoote väljatöötamise protsess, mille käigus valmistatakse tarkvara järk-järgult ja võimalikult lühikeste ajaliste tsüklitena. Tarkvara arenduse iteratiivne elutsükkel koosneb neljast osast: planeerimine, analüüs, disain ja väljatöötamine (Joonis 1).



Joonis 1. Tarkvaraarenduse elutsükkel

Arendustsükli esimene osa on planeerimine, kus tuvastatakse arendatava süsteemi vajadused ja tingimused. Teine osa on süsteemi analüüs, mis koosneb ärinõuete ja üldise struktuuri kirjeldamisest – selle raames organiseeritakse kirjeldatud nõuded ja protsessid ning tehakse valik teostamisele kuuluvatest tegevustest. Kolmas osa on süsteemi disain, milles konverteeritakse analüüsi ajal loodud ärinõuded ja protsessikirjeldused süsteemi spetsifikatsiooniks. Selle osa eesmärk on kirjeldada ära kõik süsteemiga seotud aspektid, alustades süsteemi tehnoloogia valikust kuni protsessi ja andmebaasi struktuuri skeemideni. Neljas osa on süsteemi väljatöötamine, mille raames programmeeritakse funktsionaalsus vastavalt süsteemi disainis kirjeldatud spetsifikatsioonile ja sooritatakse testimine ning valminud rakendus paigaldatakse pilvekeskkonda. [3]

2.3.2 Nõuete kogumise metoodika

Ärinõuete kogumiseks on kasutatud mitmeid meetodeid ja infoallikaid. Algandmed on autori enda poolt kogutud vaatluse ja internetis leiduvate lahenduste võrdlemise tulemusena. Nõuded jagunevad funktsionaalseteks ja mittefunktsionaalseteks ning need on kirjeldatud, kategoriseeritud, järjestatud ja visualiseeritud erinevate tarkvaraarenduse analüüsitehnikate abil – analüüsi tulemusena moodustub loodava rakenduse skoop.

Kogutud andemete baasil koostati visuaalne ja interaktiivne prototüüp, mis aitab leida uusi nõudeid nii funktsionaalsuse kui ka kasutajakogemuse kohta. Interaktiivse prototüübi koostamine võtab oluliselt vähem aega, kui seda teha programmeerides, samas annab hea ettekujutuse rakendusest nii visuaalse disaini loomiseks kui ka rakenduse funktsionaalsuse väljatöötamiseks.

2.4 Töö skoop

Antud lõputöös keskendutakse infosüsteemi arendamise tehnilisele aspektidele ja valikutele, mille tulemusena valmib veebirakenduse piisava funktsionaalsusega töötav versioon (MVP – *Minimum Viabile Product*). Töös keskendutakse nõuete kogumisele, analüüsile ja rakenduse väljatöötamisele. Töö tulemuseks valmib rakenduse lähtekood (vt Lisa 3), mida saab kompileerida ja paigaldada serverisse. Rakendus võimaldab uuel kasutajal luua kasutajakonto ja turvaliselt keskkonda sisenda. Rakenduse abil saab kasutaja tooteid otsida tootekategooria järgi või sooritada otsing vabateksti kui ka triipkoodi kasutades. Kasutaja saab lisada huvipakkuvaid tooteid personaalsesse jälgimisnimekirja, seadistada hinnateavitusi ja toodetele kirjutada hinnanguid. Kasutaja saab süsteemi lisada uusi tooteid ja sisestada toote kohta hinnainfot. Kasutajal on võimalik tootepõhiselt teiste kasutajate poolt sisestatud hinnainfot näha ja pakkumisi võrrelda. Hinnapakkumisi on võimalik järjestada kaupleja asukoha järgi sõltuvalt rakenduse kasutaja enda asukohast.

3 Analüüs

Käesolevas peatükis analüüsitakse varasemaid sarnaseid lahendusi ning koostatakse loodavat veebirakendust kirjeldavad kasutajalood, kasutusjuhud, funktsionaalseid ja mittefunktsionaalsed ärinõuded. Peatükis kirjeldatakse töös kasutatud vahendeid ja metoodikaid, luuakse kasutajaliidese interaktiivne prototüüp, valitakse rakenduse loomiseks kasutatav tehnoloogia, koostatakse kasutusvoo skeem, andmeobjekti mudel ja olemi-suhete skeem.

3.1 Nõuete kogumine

Nõuete kogumiseks on kasutatud vaatluse ja eksperimenteerimise metoodikaid, mille käigus võrreldakse erinevaid varasemalt loodud lahendusi ja tuuakse esile hinnavõrdlusrakendustele omased kriteeriumid. Kasutatud on nõuete avastamise metoodikat, mille eesmärk on eksperimenteerimise käigus koguda täiendavat infot. Kogutud andmete põhjal loodi interaktiivne ja graafiline kasutajaliidese prototüüp.

Prototüüpimine on iteratiivne protsess, mille käigus koostatakse loodavat rakendust kujutavate graafiliste vaadete komplekt, mille elemendid ühendatakse loogiliseks kasutusvooks. Prototüüp võimaldab kiiresti teisendada põhinõuded soovitud rakenduse piiratud versiooniks. Esialgsete põhinõuete kogumiseks ja kujundamiseks on vaja tagasisidet potentsiaalsetelt kasutajatelt. Prototüüp võimaldab saadud tagasiside põhjal olemasolevaid nõudeid muuta ja uusi lisada – nõuete muutumisel korratakse protsessi. [4]

Loodava rakenduse põhinõuded kirjeldati kasutajalugude metoodika abil. Seda metoodikat kasutades kirjeldatakse süsteemi omadused vabas formaadis ja kasutaja vaatepunktist. Kirjeldus peab sisaldama infot kasutaja rolli, tegevuse ja eesmärgi kohta.

Seejärel kasutajalugude kirjeldamisel kogutud andmed kategoriseeriti FURPS+ metoodika abil, mis võimaldab eristada funktsionaalsed ja mittefunktsionaalsed nõuded viie omaduse järgi: funktsionaalsus, kasutatavus, töökindlus, jõudlus ja toetatavus [5]. Nõuded järjestati olulisuse järjekorras kasutades MoSCoW metoodikat, et valida välja loodava rakenduse töö skoopi sobivad nõuded. Olulisuse määramiseks jaotatakse kasutajalood nelja gruppi: „kindlasti teha“, „peaks tegema“, „võiks teha“ ja „hiljem teha“ [6].

3.1.1 Sarnaste lahenduste võrdlemine

Peamiselt võrreldi Euroopas tegutsevad ettevõtete loodud hinnavõrdluse veebirakendusi, mille vaatluse tulemusena koguti loodava rakenduse jaoks põhinõuded. Antud peatükis analüüsiti seitsme sarnase hinnavõrdlusportaali nõrku ja tugevaid külgi.

Ceneo.pl on üks suurimaid hinnavõrdlusportaale Poolas, mis loodi 2005. aastal ja on kiiresti kasvades jõudnud 23 miljoni külastajani kuus. Ceneo pakub hinnavõrdluse teenuseid erinevates kategooriates, mis võimaldab kasutajatel leida informatsiooni poodide ja toodete kohta. Ceneo keskkond võimaldab kasutajal teha ostutellimusi, mis lubab kasutajal osta tooteid erinevatest poodidest ühe tellimusena [7]. Ceneo järjestab otsingutulemusi kombineerides CPC (*Cost Per Click*) tariifi, kasutajate hinnanguid toodetele kui ka arvestades tarbekauba hinda. Otsingutulemuse positsiooni parandamiseks on vaja kauplejal tasuda kõrgemat tariifi – mida tihedam konkurents, seda kõrgem tariif. Keskkond toetab XML-põhist andmevahetust.

Idealo.de on üks suurimaid hinnavõrdlusega tegelevaid veebilehti Saksamaal, mis alustas tegevust 2000. aastal eesmärgiga aidata kasutajal teha informeeritud ostuotsus. Idealo keskkonda külastatakse keskmiselt 76 miljonit korda kuus, neil on registreeritud üle 50 000 poe ja üle 500 miljoni toote. Idealo tegutseb kuues Euroopa riigis: Saksamaal, Austrias, Inglismaal, Prantsusmaal, Itaalias ja Hispaanias. Idealo pakub hinnainfot ka lennupiletite, hotellide ja korterite kohta [8]. Keskkond võimaldab kasutajatel tooteid lisada jälgimisnimekirja, anda tootele hinnanguid ja seadistada hinnateavitusi. Tooteid järjestatakse odavama hinna järgi, kuid arvestatakse ka e-poe CPC tariifi ja keskkonnas kasutatud reklaamieelarvet.

Kaina24.lt on Leedus tegutsev hinnavõrdlusportaal, kuhu on registreeritud üle 600 e-poe ja üle 7 miljoni toote. Kaina24 keskkonda külastab iga kuu üle 2 miljoni külastaja [9]. Kaina24 on laiendanud oma tegevust ka Eestisse – Hind.ee veebipõhisest ostuportaalist leiab informatsiooni enam kui 150 e-poe ja 3 miljoni toote kohta [10].

Hinnavaaltus.ee on Eestis tegutsev ettevõte, mis keskendub peamiselt arvutitehnika ja elektroonika hinnainfo koondamisele ja võrdlemisele. Hinnavõrdlusportaal vahendab üle 1,5 miljoni toote kohta hinnainfot [11]. Keskkonnas on registreeritud üle 350 internetipoe, kelle tooteid on võimalik võrrelda. Toodete kohta on lisatud tootekirjeldus, hinnamuutuste ajalugu ja tarbijate tagasiside. Keskkond on suhteliselt lihtne ja vähese

funktsionaalsusega – mitmeid hinnavõrdluse keskkondadele omaseid kasutajatele lisandväärtust andvaid võimalusi ei ole – samas täidab keskkond oma eesmärki ja on Eestis suhteliselt populaarne.

Kainos.lt on Leedu ettevõtte poolt loodud hinnavõrdlusportaal, kuhu on registreeritud üle 650 poe ja üle 8 miljoni toote [12]. Keskkonnas on registreeritud üle 1100 internetipoe. Süsteem võimaldab tooteid lisada XML-failiga. Tooted saab lisada jälgimisnimekirja ja neile lisada hinnateavituse. Tootelehel on võimalus näha toote detailset kirjeldust, anda tootele hinnang ja näha hinnamuutuste ajalugu.

Prisjakt.nu on hinnavõrdlusportaal, mis tegutseb seitsmes riigis: Rootsis, Inglismaal, Uus-Meremaal, Norras, Prantsusmaal, Soomes ja Taanis. Ettevõte alustas tegevust Rootsis 2002. aastal hobiprojektina. Keskkond on kasvanud Rootsi üheks oma valdkonna suurimaks – ettevõttel üle 200 töötaja [13]. Keskkond võimaldab kasutajatel kirjutada poodide ja toodete kohta arvustusi ning salvestada tooteid jälgimisnimekirja ja seadistada hinnateavitusi. Otsingutulemused järjestatakse ainult soodsama hinna järgi.

PriceRunner.se on Rootsis tegutsev ettevõte, mis loodi 1999. aastal eesmärgiga aidata kasutajatel teha informeeritud ostuotsus, mis võimaldab kasutajal säästa aega ja raha. PriceRunner on üks suurimaid hinnavõrdluskeskkondi Rootsis omades andmebaasis üle 3 miljoni toote ja üle 7000 poe. Keskkonda on võimalik lisada e-poode üle maailma, kuid tingimuseks on tarnevõimalused Rootsi. Keskkonnal on üle 15 miljoni külastuse ühes kuus. Lisaks tegutseb ettevõte ka Taanis, Inglismaal ja Norras. [14]. Keskkonnal on kõik tüüpilisele hinnavõrdlusportaalile omased funktsioonid: toodete hindamine, salvestamine jälgimisnimekirja, hinnateavituste seadistamine, hinnaajalugu ja erinevad võimalused toodete võrdlemiseks.

3.1.2 Kasutajalood

Kasutajalugu (*user story*) on lihtne ja selge lühikirjeldus, mille eesmärk on kirjeldada vabas formaadis, mida peab kavandatav süsteem kasutaja jaoks pakkuma nii, et see omaks ka reaalset väärtust kasutaja jaoks. Kasutajalood aitavad luua selgemat arusaamist süsteemi kasutaja ootustest süsteemi funktsionaalsusele. Kasutajalugusid kasutakse agiilsetes tarkvaraarenduse raamistikes nagu näiteks Scrum, kus ei planeerita tööd pikalt ette, seega peab kasutajalugu olema suhteliselt lühike ja väikse aja jooksul teostatav.

Järgnevalt on välja toodud kasutajalugude kirjeldused (Tabel 1).

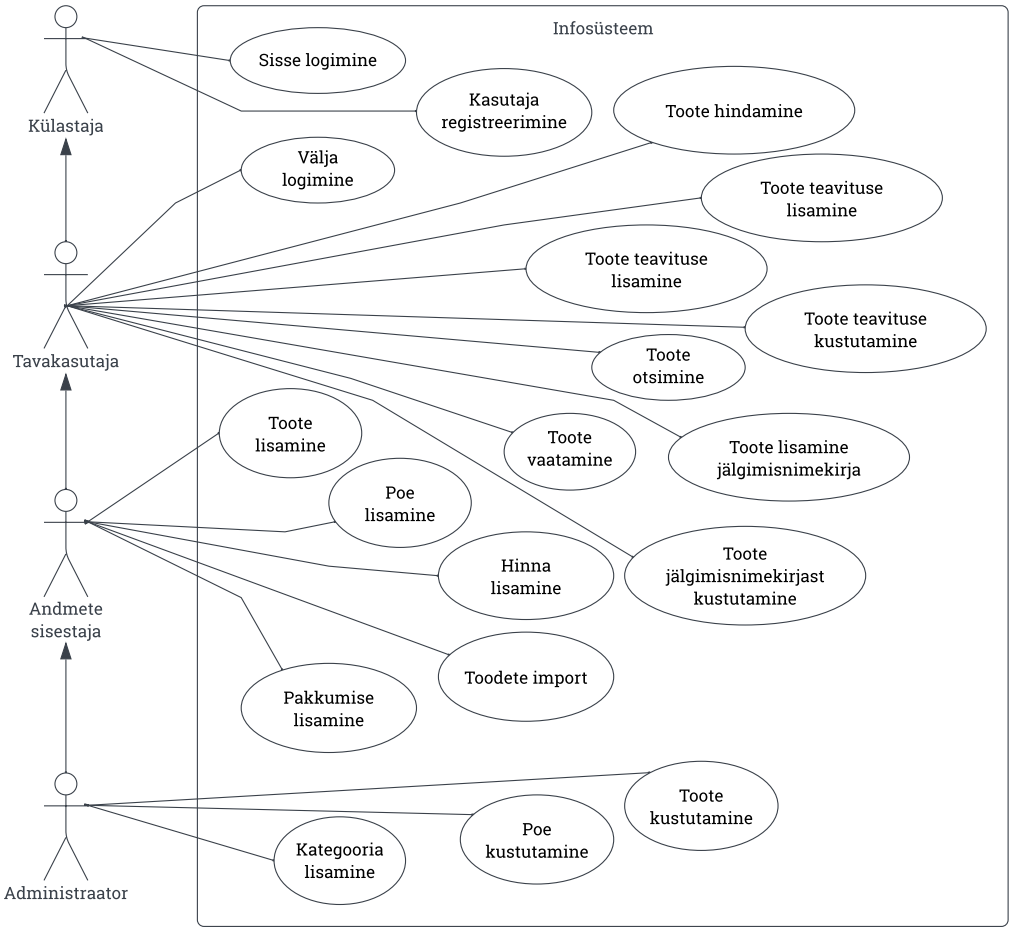
Tabel 1. Kasutajalugude kirjeldused

Kasutajana ma sooviksin ...	**Et ma saaksin saavutada ...**
Toodet otsida vabatekstina või triipkoodi abil	Leida infot toodete ja neid müüvate kauplejate kohta
Luua kasutajakonto ja sisse logida	Pääseda ligi andmetele ja teenustele
Toote hinnainfot lisada	Aidata koguda hinnainfot
Vaadata toote pakkumisi	Võrrelda toote pakkumisi
Lisada toote kohta hinnanguid	Anda tagasisidet toote kohta
Toodet lisada järgimisnimekirja	Leida huvipakkuv toode tulevikus kiiremini
Lisada tootele teavitusi hinna muutuse kohta	Saada teavitusi hinnamuutuse ja pakkumiste kohta
Süsteem toimiks mobiilses seadmes ja veebibrauseris	Kasutada rakendust mobiilses seadmetes olenemata operatsioonisüsteemist

3.1.3 Kasutusjuhtude mudel

Kasutusjuhtum (*use-case*) näitab süsteemi käitumist või funktsionaalsust, mis koosneb interaktsioonidest süsteemi ja kasutaja vahel konkreetses keskkonnas ning on seotud konkreetse eesmärgiga. Kasutusjuhtum aitab mõista süsteemi funktsionaalseid nõudeid. Kasutusjuhtude mudel (*use-case model*) kirjeldab graafiliselt süsteemi funktsioone ja näitab süsteemi käitumist koos peamiste kasutajatega, kes süsteemiga suhtlevad [15].

Kasutusjuhtude mudel kirjeldab süsteemi käitumist nii nagu näeb seda kasutaja ja see aitab valideerida funktsionaalsuse vastavust nõuetele tulenevalt kasutaja rollist süsteemis. Kasutajate rollid (*actors*) jagunevad neljaks: külastaja, tavakasutaja, andmete sisestaja ja administraator (Joonis 2).



Joonis 2. Kasutusjuhtude mudel

3.1.4 Funktsionaalsed ja mittefunktsionaalsed nõuded

Kasutajalugude põhjal on koostatud peamised nõuded, et kirjeldada loodava veebirakenduse võimekused ja piirangud. Näiteks: kas rakendus peaks töötama mobiilis; kas on vajalik interneti olemasolu; kas rakendus on maitsekalt kujundatud ja lihtne kasutada; kas rakendus on piisavalt intuitiivne; milline on rakenduse üldine kasutuskogemus.

Nõuete süstematiseerimiseks kasutatakse FURPS+ klassifitseerimise mudelit. Akronüüm FURPS koosneb viiest sõnast: funktsionaalsus (*functionality*), kasutatavus (*usability*), töökindlus (*reliability*), jõudlus (*performance*) ja toetatavus (*supportability*). Esimene kategooria kirjeldab funktsionaalseid nõudeid ja kõik ülejäänud mittefunktsionaalseid nõudeid. Lisand „+“ tähistab lisanõudeid, mis kirjeldavad rakenduse disaini piiranguid, arendusmetoodikaid või muid aspekte.

Täiendavalt rakendatakse MoSCoW metoodikat, mida kasutatakse agiilses arenduses nõuete prioritiseerimiseks. Selle tulemusena paigutatakse nõuded nelja gruppi olulisuse järjekorras. Selline nimetus tuleneb neljast klassi nimetusest: „kindlasti teha“ („*must have*“), „peaks tegema“ („*should have*“), „võiks teha“ („*could have*“) ja „hiljem teha“ („*won't have*“). Antud töös keskendutakse MoSCoW metoodika klass „kindlasti teha“ nõuete realiseerimisele, mis aitab rakenduse väljatöötamisel piiritleda töö maht ja keskenduda kõige olulisemale.

Järgnevas tabelis on kirjeldatud loodava veebirakenduse planeeritavad funktsionaalsed ärinõuded, millele on lisatud ärinõude prioriteedi klassifikatsioon. (Tabel 2).

Tabel 2. Funktsionaalsed nõuded

Kategooria	Nõuded	Prioriteet
F Funktsionaalsus	F1. Kasutajal peab olema võimalik registreerida	M
	F2. Kasutaja peab saama parooliga süsteemi siseneda	M
	F3. Kasutaja peab saama sooritada toote otsingut	M
	F4. Kasutaja peab saama uusi tooteid lisada	M
	F5. Kasutaja peab saama tooteid jälgimisnimekirja lisada	M
	F6. Kasutaja peab saama tootele lisada hinnanguid	M
	F7. Kasutaja peab saama tootele lisada hinnateavitust	M
	F8. Kasutaja peab saama lisada tootele hinnainfot	M
	F9. Kasutaja peab saama teha otsinguid toote triipkoodi järgi	M
	F10. Kasutajal peab saama võrrelda toote hinnapakkumisi	M
	F11. Kasutajal peab olema võimalik näha toote detaile	M
	F12. Kasutaja peab saama sorteerida pakkumisi poe asukoha järgi	M
	F13. Kasutaja peab saama otsitava toote triipkoodi skännida	S
	F14. Kasutajal peab olema võimalik enda andmed muuta	S
	F15. Tooteleht peab sisaldama hinna muutuste ajalugu	S
	F16. Kasutaja peab saama aktiveerida enda asukoha info (GPS)	S
	F17. Kasutaja peab saama sorteerida pakkumisi hinna järgi	S
	F18. Kasutaja peab saama võrrelda tooteid nende omaduste järgi	C
	F19. Kasutaja peab saama raporteerida vigaselt sisestatud andmeid	W

Mittefunktsionaalsed ärinõuded on välja toodud alljärgnevas tabelis, mis on jaotatud kategooriatesse ja millele on lisatud ka nõuete prioriteedi klassifikatsioon (Tabel 3).

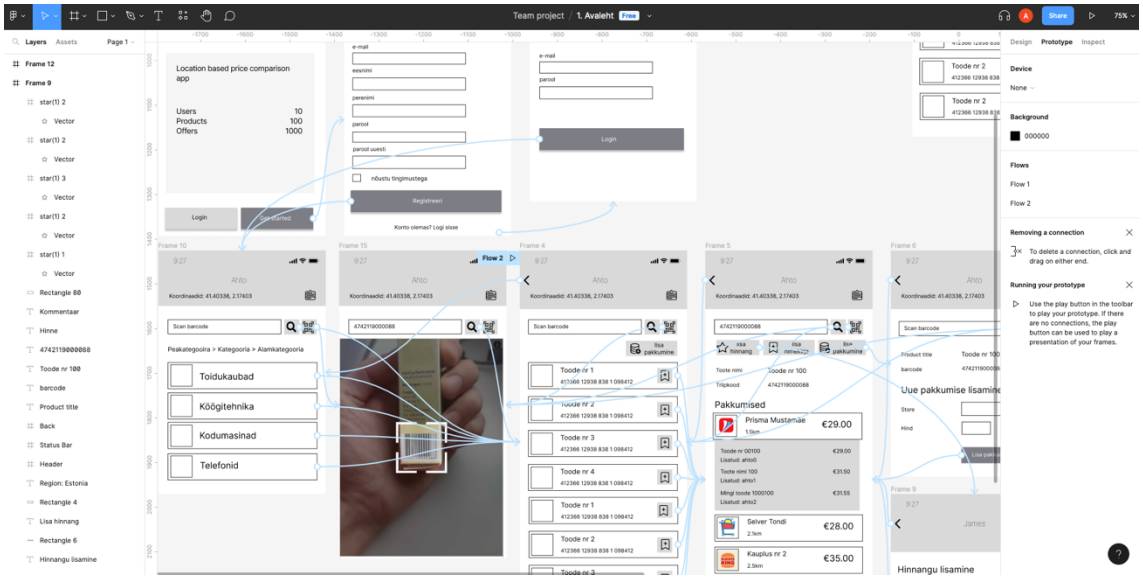
Tabel 3. Mittefunktsionaalsed nõuded

Kategooria	Nõuded	Prioriteet
U Kasutatavus	U1. Rakendus peab olema kasutatav mobiilses seadmes	M
	U2. Rakendus peab olema kasutatav erinevates veebibrauserites	M
	U3. Rakendus peab olema tasuta kasutatav	M
	U4. Rakendus peab olema kasutatav ilma juhendita	S
	U5. Rakendus peab olema dokumenteeritud	C
R Töökindlus	R1. Süsteem peab olema mõistliku aja jooksul taastatav	M
	R2. Süsteemis tekkivad vead peavad olema logitud	S
	R3. Funktsionaalsus peab töötama ettearvatavalt	C
	R4. Süsteemi peab olema kasutatav 99% ajast	W
P Jõudlus	P1. Süsteem peab töötama sujuvalt	M
	P2. Süsteem peab töötama mõistliku rahalise kuluga	S
	P3. Rakenduse päringu vastus ei tohi olla pikem kui 2 sekundit	C
	P4. Süsteem peab koormust arvestades olema skaleeritav	W
	P5. Rakendus võimaldab teenindada 1000 kasutajat samaaegselt	W
S Toetatavus	S1. Rakenduse komponendid peavad olema eraldi testitavad	M
	S2. Rakendus peab olema lihtsalt edasi arendatav	M
	S3. Rakendus peab olema lihtsalt paigaldatav	M
	S4. Rakendus peab olema lihtsalt hooldatav	S
	S5. Rakendus peab olema eesti ja inglise keelne	S
	S6. Rakendus peab olema paigaldatav erinevatel platvormidel	W
+ Piirangud	+1. Vabavaralised lahendused on eelistatud	M
	+2. Rakenduse lähtekoodi haldamiseks kasutada Git süsteemi	M
	+3. Rakenduse lähtekood peab olema avalik	M

3.1.5 Kasutajaliidese prototüüp

Kasutajaliidese abil saab kasutaja juurdepääsu rakenduse funktsionaalsusele, kuid see ei ole ainuke oluline aspekt. Kasutajaliidese disaini juures on väga oluline, et see oleks hea väljanägemisega ja intuitiivne, sest kasutajaliides peab olema kasutajale arusaadav ja ettearvatav, sest sellest sõltub, kas kasutaja jääb rakendust kasutama või mitte. Kasutajaliidese disaini prototüübi loomise protsess on tavaliselt iteratiivne – see tähendab, et pärast igat versiooni tuleb seda valideerida rakenduse kasutaja vaatenurgast ja otsese tagasiside põhjal.

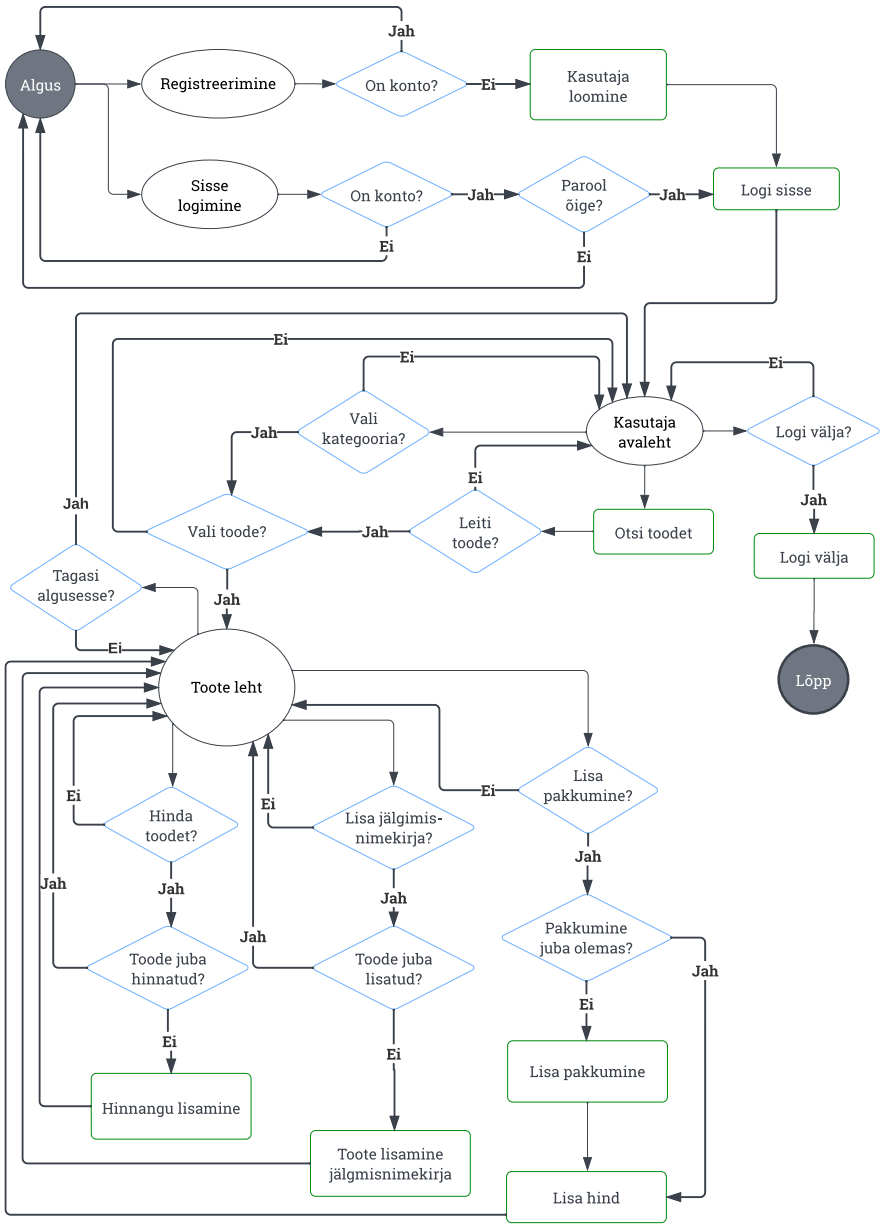
Antud lõputöös on kasutatud programmi Figma, mis on kasutajaliidese prototüüpimiseks mõeldud tööriist, millega on võimalik kujundada kasutajaliidese toimiv kasutusvoog ja luua sellest rakenduse visuaalne töötav prototüüp. Programm võimaldab luua graafilisi elemente ning neid omavahel ühendada nii, et moodustub sujuv interaktiivne kasutuskogemus. Järgneval joonisel on kujutatud prototüüpimise programmi Figma, kus on koostatud klientrakenduse vaated ja nende vahelised seosed (Joonis 3).



Joonis 3. Kuvatõmmis programmist Figma

3.2 Protsesside kaardistamine

Protsesside kaardistamine aitab luua ja analüüsida süsteemi tööprotsessi ning visualiseerida rakenduse kasutusvoogu. Järgneval joonisel on kujutatud rakenduse kasutusvoog kasutaja vaatepunktist (Joonis 4).

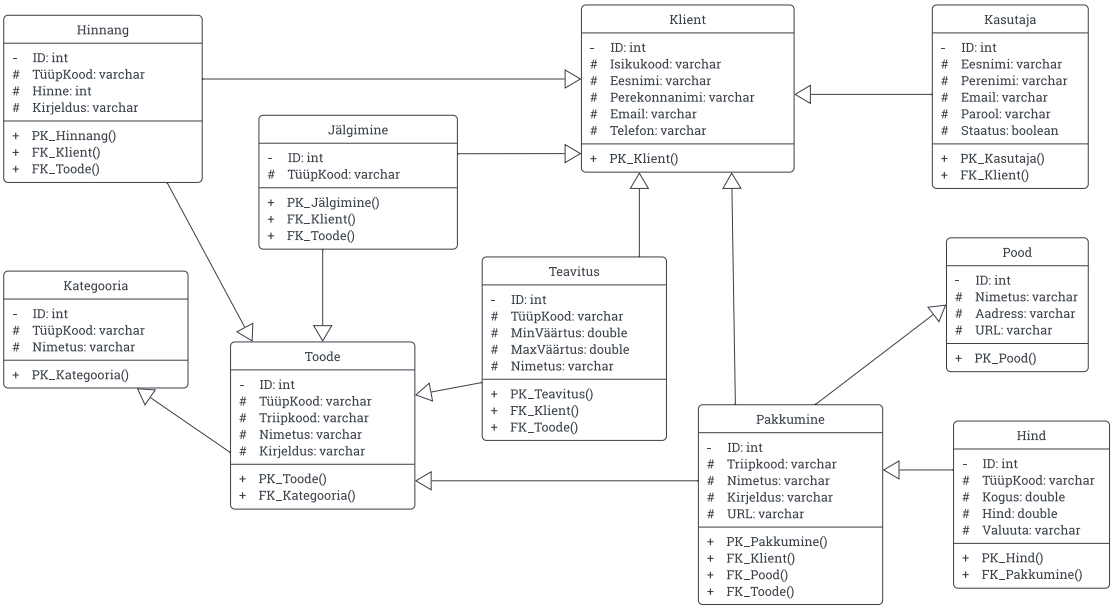


Joonis 4. Rakenduse kasutusvoog

3.3 Andmebaasi projekteerimine

Andmebaasi struktuuri loomine on üks osa infosüsteemi arenduse protsessist. Selle disainimisel tuginetakse püstitatud ärinõutele ja rakenduse funktsionaalsusele eesmärgiga katta kõik andmetega seotud aspektid nii süsteemi kui ka kasutaja vajadusi arvesse võttes.

Järgneval joonisel on kujutatud süsteemi andmeobjekti mudel, mis aitab paremini aru saada andmetest ja nende vahelistest seostest süsteemis, mida peab andmebaasi struktuuri loomisel arvestama (Joonis 5). Andmemudeli kirjeldamiseks kasutatakse UML (*Unified Modeling Language*) modelleerimiskeelt.



Joonis 5. Süsteemi andmeobjekti mudel

Andmebaasi projekteerimisel arvestatakse relatsiooni ja seose terviklikkuse põhimõtet, kus igal relatsiooni objektil on primaarvõti (*primary key*). Seose terviklikkuse piirang määratakse kahe relatsiooni objekti vahel – nii saab tagada, et viidatakse õigele relatsiooni objektile – sellist viidet nimetatakse välisvõtmeks (*foreign key*). Iga välisvõti viitab alati kindla relatsiooni objekti primaarvõtmele. [16]

Tagarakenduse väljatöötamise käigus koostati detailne olemi-suhete skeem (vt Lisa 2 Joonis 16), mis katab ära kõik andmetega seotud tehnilised detailid, mille baasil lõplik andmebaasi struktuur luuakse. Olemi-suhete skeem hõlmab endas kõiki rakenduse tööks vajalikke detaile, mis võetakse aluseks andmebaasiolemite programmeerimisel.

3.4 Tehnoloogia valik

Tehnoloogia valikul lähtutakse kaasaegsetest ja laialt kasutuses olevatest tehnilistest lahendustest. Valiku tegemisel arvestatakse erinevaid aspekte: kaasaegsus, populaarsus, õppematerjalide kättesaadavus, õppekõver, autori varasem kogemus, tehnoloogia hind, jõudlus ja funktsionaalsus. Tehnoloogia valikust sõltub keskkondade seadistus, arendustööriistad, arendusele kuluv aeg ja rakenduse paigaldamise protsess.

3.4.1 Programmeerimiskeel ja tagarakenduse raamistik

Tagarakenduse ülesanne on töödelda ja serveerida andmeid klientrakendusele. Enne rakenduse programmeerimist tehakse programmeerimiskeele valik ja sellest tulevalt valitakse tagarakenduse raamistik.

PYPL (*PopularitY of Programming Language*) programmeerimiskeelte populaarsuse indeks koostatakse analüüsides, kui sageli otsitakse Google otsingsüsteemist erinevate programmeerimiskeelte õppematerjale. Tulemustest selgus, et kõige populaarsemad serveripoolsed programmeerimiskeeled on Python (27.43%), Java (16.41%) ja C# (6.9%). [17]

Python on skriptimiskeel, mis on peamiselt levinud andmeteaduses ja masinõppes. Python on dünaamiliselt tüübitud objekt-orienteeritud keel. Python süntaks kasutab koodi kirjeldamisel treppimist (*indentation*), mis võib piirata keerukamate lausete kirjutamist. Näiteks Python toetab küll *lambda* funktsionaalsust, kuid laused võivad olla ainult üherealised. Python on üldiselt aeglasem kui Java ja C#, sest lähtekoodi ei ole vaja enne käivitamist kompileerida. Python keele eeliseks on selle õppimise lihtsus ja seda on lihtne kasutusele võtta ilma spetsiaalset keskkonda seadistamata. [18]

Java on laialdaselt kasutatud programmeerimiskeel veebipõhiste rakenduste loomiseks. Java on kiire, turvaline, stabiilne ja tüübikindel objekt-orienteeritud keel, mida kasutatakse nii väikeste mobiilirakendustes kui ka suurtes infosüsteemides. Java sarnaneb C, C++ ja JavaScript programmeerimiskeeltele. Kuigi Java keeles on sarnasusi JavaScriptiga, siis need on väga erinevad keeled, sest Java keelt on vaja kompileerida ja seda on võimalik käivitada kõikidel platvormidel, erinevalt JavaScript keelele, mis töötab ainult veebibrauseris ega vaja kompileerimist. [19]

C# on Microsofti poolt arendatav kaasaegne, objekt-orienteeritud ja tüübikindel programmeerimiskeel, mille juured pärinevad C-keele perekonnast ja on C, C++, Java ja JavaScript programmeerimiskeeltele sarnane. C# programmeerimiskeele võimaluste hulka on lisatud komponente ja konstruktsioone, mis muudavad selle keele kasutamise rakenduste loomise juures väga mugavaks. [20]

Järgnevas tabelis on võrreldud nimetatud programmeerimiskeelte omadusi (Tabel 4).

Tabel 4. Programmeerimiskeelte võrdlus

Omadus / Keel	**Python**	**Java**	**C#**
Kogemus	Keskmine	Keskmine	Keskmine
Materjalid	Hea	Hea	Keskmine
Õppekõver	Sujuv	Keskmine	Keskmine
Raamistikud	Flask, Django	Spring, Spark	ASP.NET
Tüübikindel	Ei	Jah	Jah
Jõudlus	Keskmine	Hea	Väga hea

Programmeerimiskeeleks valiti Java, sest tegemist on tugevalt tüübitud ja laialdaselt kasutatud objekt-orienteeritud programmeerimiskeelega. Java raamistiku valiku puhul arvestati, et see võimaldab kasutada kaasaegseid tarkvaraarenduse mustreid, samas peab olema piisavalt lihtne õppida ja kasutusele võtta. Spring raamistik on Java-põhine ja aitab oluliselt kiirendada rakenduse väljatöötamise protsessi, sest sisaldab laial valikul valmiskomponente. Näiteks, raamistik lihtsustab rakenduse seadistamist ja Java sõltuvuste haldamist ning sisaldab veebiserveri funktsionaalsusi (Spring Boot). [21]

3.4.2 Klientrakenduse raamistik

Klientrakenduse raamistiku valiku eeltingimuseks on TypeScript programmeerimiskeele põhisus, mis tagab klientrakenduse toimimise veebibrauseris ilma täiendava tugisüsteemita. Raamistike omaduste võrdlemiseks koostatakse koondtabel (Tabel 5).

StackOverflow poolt 2022. aastal läbi viidud küsitlus „Developer Survey“ leidis, et uuringus osalenute hinnangul klientrakenduste loomiseks on kõige populaarsemad raamistikud React.js (42.62%), Angular (20.39%) ja Vue.js (18.82%). [22]

Vue on vabavaraline JavaScript keelel põhinev raamistik, mida kasutatakse peamiselt kasutajaliideste loomiseks. Raamistik põhineb tehnoloogiatel HTML, CSS ja JavaScript. Vue pakub deklaratiivset ja komponendipõhist programmeerimismudelit, mis aitab

tõhusalt arendada kasutajaliideseid. Vue raamistik sobib kasutamiseks nii väikestes kui ka suure mahulistes projektides. [23]

Angular on komponendipõhine raamistik, mis põhineb tehnoloogitel HTML ja TypeScript. Angular sisaldab laial valikul teeke ja tööriistu, mis aitavad rakendust efektiivselt arendada ning testida. Angular projektid on peamiselt eraldatud mooduliteks, komponentideks ja teenusteks. Iga komponent sisaldab HTML malli, äriloogikat ja metaandmeid (*decorators*). Teenused võimaldavad eraldada äriloogika nii, et see oleks taaskasutatav mitmetes komponentides samaaegselt. [24]

React.js on populaarne JavaScript põhine teek, mis sisaldab laia valiku kasutajaliidese loomise jaoks loodud funktsioone ja komponente. React.js järgib komponendipõhist loogikat, kuid ei sisalda kõiki tüüpilisele raamistikule omaseid funktsionaalsusi nagu näiteks marsruutimine (*routing*) või andme pärimine (*data fetching*), kuid neid saab lisada eraldi arendatud moodulitena. [25]

Tabel 5. Klientrakenduse raamistike võrdlus

Omadus / Raamistik	**Vue**	**Angular**	**React**
Populaarsus	Keskmine	Hea	Hea
Jõudlus/skaleeritavus	Hea	Hea	Hea
Adapteeritavus	Hea	Keskmine	Keskmine
Kiirus	Hea	Hea	Keskmine
Paindlikus	Hea	Keskmine	Keskmine
Materjalid	Hea	Keskmine	Keskmine
Õppekõver	Väga hea	Keskmine	Hea

Antud lõputöö raames tehti valik Vue kasuks, sest antud raamistik kombineerib nii Angular kui ka React omadusi. Vue on kiire, paindlik ja lihtsalt õpitav ning võimaldab luua kihilise struktuuriga klientrakenduse.

3.4.3 Andmebaas

Andmebaasi valikul eeltingimuseks on see, et andmebaas peab võimaldama relatsioonide seadistamist, mis sobib kõige paremini planeeritava rakenduse andmete struktuuriga. Erinevate andmebaasisüsteemide võrdlemiseks koostatakse koondtabel (Tabel 6).

TOPDB (*Top Database*) populaarsuse indeks, mis analüüsib populaarsust Google otsingusüsteemi andmete põhjal, on kindlaks teinud, et populaarsemad relatsioonilased andmebaasid on Oracle (26.41%), MySQL (19.06%), SQL Server (12.25%) ja PostgreSQL (6.65%). [26]

Oracle on laialt levinud suurtes ärirakendustes. Oracle pakub kasutajale laia valiku erinevaid optimaalseid ja hea jõudlusega andmebaasitooteid. See relatsiooniline andmebaasisüsteem sisaldab kõiki olulisi funktsioone, mida andmebaasisüsteemi juures võib vaja minna. Oracle puuduseks on see, et andmebaasi kasutamine ei ole tasuta, seega on pigem ärikliendile suunatud lahendus. [27]

SQL Server (Microsoft SQL Server) on relatsiooniline andmebaasisüsteem, mida arendab Microsoft. See andmebaasisüsteem on küll tasuta kättesaadav, kuid teatud mahupiirangutega – mahtude suurenedes on vajalik osta litsents. SQL Server andmebaasi teeb populaarseks maailmas laialt kasutuses olevad teised Microsoft tooted, seega on sellistel kasutajatel lihtne teha valik antud süsteemi kasuks, sest varasemalt juba kasutatakse Microsofti tooteid. SQL Server on parim valik, kui andmebaasi hallatakse Windows operatsioonisüsteemis. [28]

MySQL on laialt levinud andmebaasi tehnoloogia. MySQL alustas vabavaralisena, kuid on loodud ka tasulisi versioone, mis on suunatud äriklientidele. MySQL on ilmselt kõige tuntum andmebaasisüsteem maailmas, sest sellel on pikk ajalugu ja seda kasutakse laialdaselt jagatud veebimajutust (*shared hosting*) pakkuvatel platvormidel. [29]

PostgreSQL on vabavaraline andmebaasisüsteem, mis on väga tugev alternatiiv MySQL andmebaasisüsteemi kõrval. PostgreSQL sobib nii väikeste kui ka suurte infosüsteemide loomiseks, sest see on stabiilne ja väga hea jõudlusega, mis saab hakkama ka väga suurte andmemahtudega. PostgreSQL andmebaasi eelis on, et see tuleb hästi toime olukordades, kus võib andmebaas peab tegelema suurel arvul samaaegselt saadetud päringutega (*concurrent transactions*). PostgreSQL on objekt-relatsiooniline (*object-relational*)

andmebaas, mis võimaldab kasutada mitmeid funktsioone nagu näiteks pärilikkus (*inheritance*) ja samanimelisi funktsioone erinevate sisenditega (*overloading*). [30]

Tabel 6. Andmebaasisüsteemide võrdlus

Omadus / Süsteem	**Oracle**	**MySQL**	**SQL Server**	**PostgreSQL**
Funktsionaalsus	Hea	Hea	Hea	Hea
Materjalid	Keskmine	Hea	Keskmine	Hea
Kiirus	Väga hea	Hea	Hea	Väga hea
Hind	Tasuline	Tasuta	Tasuta*	Tasuta
Jõudlus	Väga hea	Hea	Hea	Väga hea
Turvalisus	Hea	Keskmine	Keskmine	Hea

Andmebaasisüsteemiks valiti PostgreSQL, sest see on tasuta kättesaadav, stabiilne ja väga hea jõudlusega. See andmebaas on laialdaselt levinud ja selle kohta leidub piisavalt õppematerjale. Tuleviku perspektiivis on PostgreSQL hea valik, sest süsteem on võimeline toime tulema ka suurte andmemahtudega – seega on vähe tõenäoline, et mahtude suurenedes on vaja andmebaasisüsteem välja vahetada.

3.4.4 Veebirakenduse serveerimine konteinertehnoloogia abil

Rakenduse serveerimiseks on kasutatakse pilveplatvormi konteinerteenuseid, mis aitab efektiivselt kulusid hallata. Konteiner võimaldab pakendada rakendus iseseisvasse virtuaalkeskkonda, mis tagab keskkonna konfiguratsiooni ja sujuva paigaldamise. Lisaks jätab konteinerite kasutamine võimaluse tulevikus pilveplatvormi vahetada.

Valitakse kolme ettevõtte vahel, kes olid lõputöö koostamise ajal populaarseimad: Amazon AWS, Microsoft Azure ja Google Cloud [31]. Kõik nimetatud pakkujad toetavad oma platvormil Kubernetes konteinerite orkestreerimise mehhanismi, mida kasutatakse konteinerite paigaldamiseks, skaleerimiseks ja haldamiseks [32]. Valiku tegemisel võetakse arvesse, et nimetatud pakkujatel on arendatud ka arvestatavad alternatiivid, mis on piiratud võimalustega, kuid oluline tegur kulude optimeerimisel.

AWS (*Amazon Web Services*) on alustas 2002. aastal ja loodi ettevõtte Amazon poolt. AWS on antud hetkel enim kasutatud pilveplatvorm, mis sisaldab suurel hulgal teenuseid üle maailma. AWS võimaldab valida EKS (*Elastic Kubernetes Service*) ja ECS (*Elastic Container Service*) konteinerteenuse vahel [33]. ECS on AWS poolt arendatud toode, mis on oluliselt kulusõbralikum ning ei sisalda litsentsitasusid – hinnastamine toimib tunnipõhiselt ainult kasutatud ressursi eest.

Microsoft Azure on pilveplatvorm, mis pakub üle 200 erineva toote ja teenuse, mis võimaldavad efektiivselt luua, paigaldada ja hallata rakendusi nii pilvekeskkonnas kui ka klassikalistes serverites. Sarnaselt teistele pilveteenuse pakkujatele võimaldab Azure ressursipõhist hinnastamist – see tähendab, et kasutaja maksab tunnipõhiselt ainult kasutatud ressursi eest [34]. Azure pakub kahte konteinerteenust, mille vahel valida: AKS (*Azure Kubernetes Service*) ja Azure Container Instance.

Google Cloud on pilveplatvorm, mis pakub üle 100 erineva infrastruktuuri teenuse. Google Cloud eeliseks on paindlik hinnastamine, mis on sekundipõhine erinevalt AWS ja Azure tunnipõhisest kuluarvestusest. Samuti pakutakse erinevaid soodustusi pikaajalistele klientidele. Google Cloud pakub kolme erinevat konteinerteenust: GKE (*Google Kubernates Engine*), Cloud Run ja GCE (*Google Compute Engine*).

Järgnevalt on võrreldud valikus olnud pilveteenusepakkujate omadusi (Tabel 7).

Tabel 7. Pilveteenusepakkujate võrdlus

Omadus / Süsteem	**AWS**	**Azure**	**Google Cloud**
Funktsionaalsus	Hea	Keskmine	Hea
Materjalid	Hea	Keskmine	Keskmine
Kiirus	Hea	Hea	Hea
Hinnastamine	Keskmine	Keskmine	Hea
Turvalisus	Hea	Hea	Hea

Valik tehti AWS kasuks, sest AWS on suurim ja populaarseim pilveteenuste pakkuja. Võrreldes teistega on AWS loonud kõige rohkem andmekeskusi üle maailma. Valiku tegemist mõjutas ka lõputöö autori varasem kogemus AWS teenustega. Oluliseks osutus ka AWS loodud CDK (*Cloud Development Kit*) raamistik, mis võimaldab infrastruktuuri ressursse defineerida Python programmeerimiskeelt kasutades.

3.5 Arendusvahendid

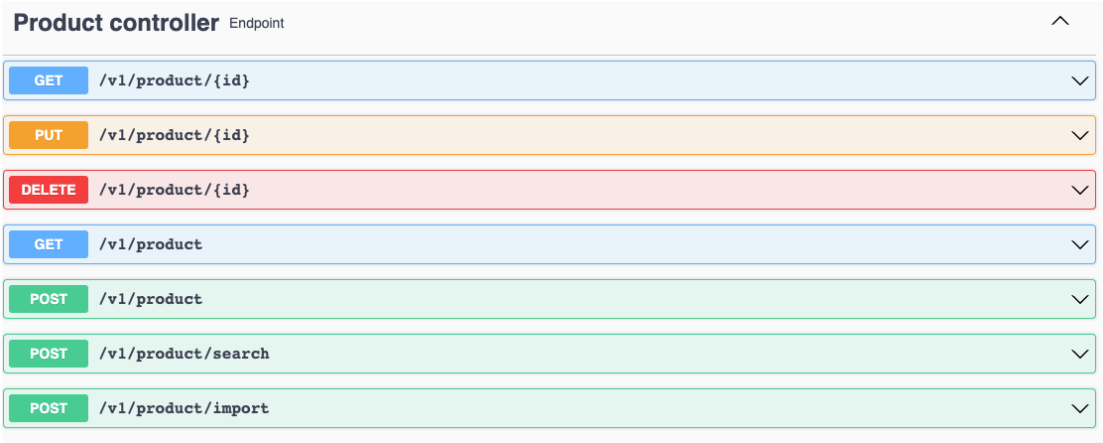
Rakenduse väljatöötamiseks on kasutatud levinud kaasaegseid tööriistu ja agiilseid tööprotsesse, et tagada arenduse kvaliteet, madal tarkvaravigade arv ja arendustööde teostamise kiirus. Rakenduse jaoks koostatud dokumentatsioon säästab aega ja energiat ning on kvaliteedi- ja protsessikontrolli jaoks hädavajalik. Dokumentatsioon on kriitilise tähtsusega, sest tulevikus uute inimeste töösse kaasamisel on vajadus selgitada süsteemi omadusi ja protsesse ning võib juhtuda, et dokumentatsioon on sellel hetkel ainuke tõeallikas (*single source of truth*).

3.5.1 Dokumentatsioon

Rakenduse dokumenteerimiseks on kasutatud erinevaid vahendeid, mille kombineerimisel on võimalik saada hea ülevaade rakenduse erinevatest aspektidest.

Toetavaks infoallikaks on Github Wiki keskkond, kuhu on koondatud täiendav materjal rakenduse tehniliste andmete kohta. Kasutajaliidese disaini ja struktuur on dokumenteeritud programmides Zeplin ja Figma. Erinevate skeemide koostamiseks on kasutatud Lucid Chart programmi.

Swagger on Java komponent, mis võimaldab automaatselt genereerida HTML-põhise API dokumentatsiooni REST kontrollerite baasil. Swagger kasutajaliides on mugavalt kasutatav ja lihtsasti loetav. Swagger võimaldab teha autoriseeritud API päringuid otse REST kontrollerite vastu veebipõhise kasutajaliidese kaudu, mis muudab API testimise väga mugavaks. Swagger leiab üles automaatselt kõik vajalikud kirjeldused, mis on API dokumentatsiooni koostamiseks oluline. Järgneval joonisel on kujutatud Swagger kasutajaliidese kuvatõmmis (Joonis 6)



Joonis 6. Swagger kasutajaliidese kuvatõmmis

3.5.2 Tööriistad ja süsteemid

Rakenduse väljatöötamise raames on kasutatud erinevat tarkvara, mis on oluline veebirakenduse disainimiseks, lähtekoodi kirjutamiseks, testimiseks, andmebaasi haldamiseks ja dokumentatsiooni hoidmiseks.

Järgnevas tabelis on välja toodud antud töös kasutatud tööriistad, mis on jaotatud nelja teema vahel. (Tabel 7).

Tabel 7. Töös kasutatud tööriistad

Klientrakendus	WebStorm IDEA, Vue, Zeplin, Figma
Tagarakendus	Intellij IDEA, Java 17, Spring Boot, Hibernate, DBeaver
Testimine	Junit, Selenium
Tugisüsteemid	Github, Gradle, Github Actions, Docker, Lucid chart

4 Rakenduse väljatöötamine

Antud peatükis kirjeldatakse lõputöö raames loodud veebirakenduse väljatöötamise protsessi ning selle juures kasutatud metoodikaid ja tehnoloogilisi lahendusi, et anda põhjalik ülevaade rakenduse arhitektuurist, ülesehitusest ja komponentidest.

4.1 Infosüsteemi arhitektuur

Käesolevas lõputöös kasutati veebirakenduse loomiseks kihilise struktuuriga arhitektuuri, mis on tarkvaraarenduses laialdaselt levinud ja enamikes rakendustes kasutatud arhitektuurimuster ning sellepärast on tundud enamike arhitektide, disainerite ja arendajate seas. Kihiline arhitektuurimuster kattub üldiselt traditsioonilise organisatsiooni IT-struktuuriga, seega on see loomulik valik enamike ärirakenduste väljatöötamisel.

Rakenduse komponendid kihilise arhitektuuris on paigutatud kihtidesse, kus igal kihil on rakenduses oma roll. Kuigi kihiline arhitektuurimuster ei määratle kihtide tüüpe ega nende arvu, siis enamik rakendusi kasutab nelja kihti: esitluskiht (*presentation layer*), talitusloogikakiht (*business layer*), andmetele juurdepääsukiht (*persistence layer* või *data access layer*) ja andmebaasikiht (*database layer* või *domain layer*). Kui väiksemate rakenduste puhul võib kasutatud olla ainult kolm kihti, siis keerukamate rakenduste arhitektuuris võib kihtide arv olla viis ja rohkem. [35]

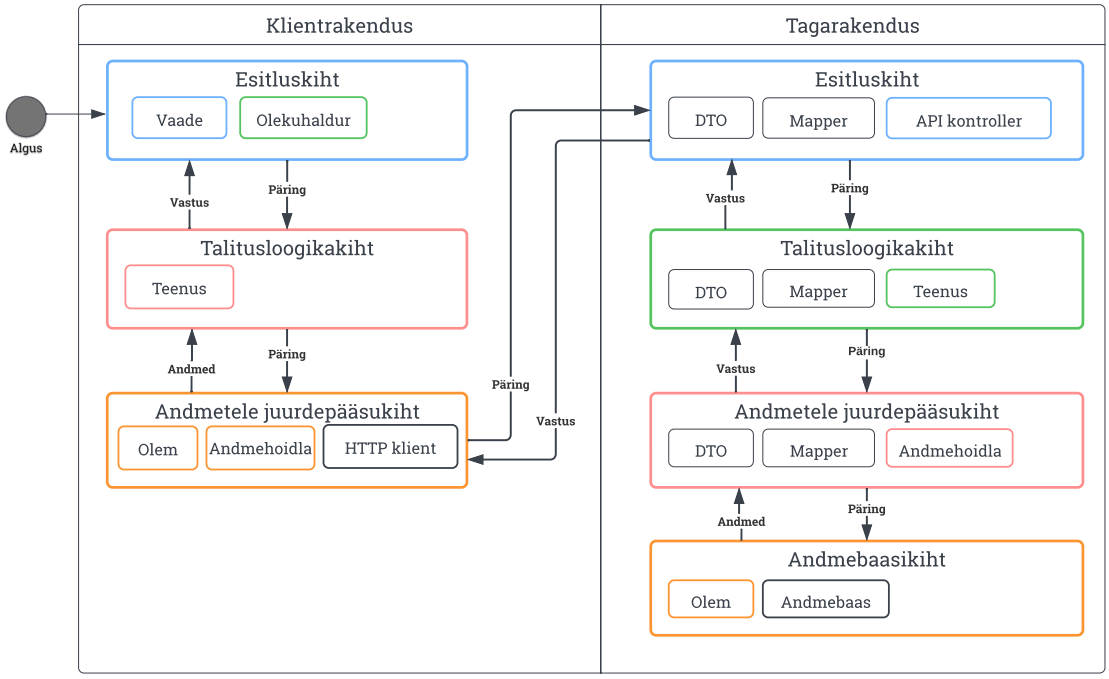
Esitluskiht on rakenduse väline kiht, millega suhtlevad otseselt kasutajad ja teised välised süsteemid. Esitluskihis kirjeldatakse rakenduse kasutajaliides, mis on kõige olulisem rakenduse kasutamiseks vajalik komponent lõppkasutaja vaatepunktist.

Talitusloogikakiht on sisaldab rakenduse toimimiseks vajalikku äriloogikat. Selline äriloogika on jaotatud komponentideks, mida nimetatakse teenuseks (*service*). Selle kihiga suhtleb ainult esitluskihis olevad komponendid. Teenus on vastutav äri spetsiifilise funktsionaalsuse ja andmete töötlemise eest.

Andmetele juurdepääsukihi ülesanne on valmistada ette andmebaasi päringuid, mida kasutatakse talitusloogikakihis asuvate teenuste poolt. Selles kihis on loogika kirjeldamiseks kasutatud andmehoidlamustrit (*repository pattern*), mis aitab jaotada komponendid nii, et päringuid on võimalik taaskasutada erinevates teenustes.

Andmebaasikihis kirjeldatakse andmebaasiga suhtlemiseks vajalikud olemid ja nende vahelised seosed. Selle kihi abiga luuakse otsene seos reaalse andmebaasi ja rakenduse lähtekoodis kirjeldatud olemite vahel.

Antud lõputöö raames loodav veebirakendus koosneb kahest rakendusest, kus tagarakendusele on planeeritud neli ja klientrakendusele kolm kiht. Tagarakenduse komponendid jagunevad nelja kihi vahel, milleks on esitluskiht, talitusloogikakiht, andmetele juurdepääsukiht ja andmebaasikiht. Klientrakendus koosneb kolmest kihist, milleks on esitluskiht, talitusloogikakiht ja andmetele juurdepääsukiht. Klientrakendusel puudub andmebaas, seega ei ole andmebaasikiht vajalik. Klientrakenduse andmetele juurdepääsukiht sisaldab API päringute funktsionaalsust, mis on oluline tagarakendusega suhtlemiseks. Klient- ja tagarakenduse kihtide vahelist suhtlust illustreerib allpool toodud joonis, kus on näha kuidas lõppkasutaja algatatud tegevused liiguvad läbi erinevate kihtide (Joonis 7).



Joonis 7. Klient- ja tagarakenduse vaheline suhtlus

4.2 Tagarakenduse arendus

Antud lõputöö raames loodav tagarakendus koosneb neljast loogilisest kihist, milleks on esitluskiht, talitusloogikakiht, andmetele juurdepääsukiht ja andmebaasikiht.

Esitluskihis on kirjeldatud REST kontrollerid, mille eesmärk on võimaldada klientrakendusel ja teistel välistel süsteemidel suhelda tagarakendusega. Antud lõputöö raames on selleks klientrakendus, kuid selleks võib tulevikus olla ka mõni teine süsteem või integratsioon. See avalik kiht defineerib rakenduse kasutamiseks olulised andmed ja funktsionaalsuse. REST kontrolleritest väljastatud ja sisestatud andmeid reguleerivad andmeedastusobjektid ehk DTO (*data transfer object*), millega on võimalik filtreerida ja kombineerida erinevaid API päringutes kasutatud andmeid.

Talitusloogikakihis on tagarakenduse teenused, mida kasutavad peamiselt teised teenused ja API kontrollerid. Teenus on komponent, kus kirjeldatakse ärirakenduse omadused ja funktsionaalsused. Teenusest päritavaid andmeid reguleeritakse andmeedastusobjektide abil.

Andmetele juurdepääsukiht koondab andmebaasi päringute koostamiseks vajaliku loogika ühte kohta. Selles kihis on kasutatud andmehoidlamustrit (*repository pattern*), mis võimaldab kirjeldada andmebaasi päringuid objekt-orienteeritud viisil kasutades JPA meetodeid.

JPA (*Java Persistence API*) on Java programmeerimiskeeles kasutatav ORM (*Object Relational Mapping*) spetsifikatsioon, mis sisaldab kogumiku eeldefineeritud meetodeid ja vahendid uute andmebaasipäringute loomiseks.

Andmebaasikihis kirjeldatakse olemid ehk andmeobjektid ja nende vahelised seosed. ORM metoodikat rakendades on võimalik luua seos koodis kirjeldatud andmeobjekti ja andmebaasi asuva andmetabeli vahel.

4.2.1 Rakenduse ülesehitus

Antud tagarakendus kasutab Java programmeerimiskeele põhist Spring raamistiku edasiarendatud versiooni Spring Boot ning sellest lähtuvalt on loodud järgnev struktuur:

- `src` – rakenduse lähtekood
  - `main`
    - `java` – rakenduse Java lähtekood
      - `bll` – talitusloogikakiht
        - `dto` – andmeedastusobjektid
        - `filter` – HTTP päringute/vastuste filtrid
        - `mapper` – talitusloogikakihi sidumine andmetega suhtlemise kihiga
        - `service` – äriloogikat sisaldavad teenused
        - `util` – abistavad funktsioonid
      - `config` – komponentide konfiguratsioon
      - `controller` – esitluskiht, REST kontrollerid, API meetodid
      - `dal` – andmetega suhtlemise kiht
        - `dto` – andmeedastusobjektid
        - `mapper` – andmetega suhtlemise kihi sidumine andmebaasi kihiga
        - `repository` – andmehoidlad
      - `domain` – andmebaasikiht, olemid
      - `dto` – avalikud andmeedastusobjektid
        - `mapper` – esitluskihi sidumine talitusloogikakihiga
    - `resources` – staatilised ressursid
  - `test` – rakenduse testide kood
- `build.gradle` – Gradle seadistus ja Java sõltuvuste haldamine
- `Dockerfile` – konteineri loomise kirjeldus

4.2.2 Andmeedastusobjektid

Käesoleva tagarakenduse kihilise arhitektuuri kihtide vahelises suhtluses andmete edastamiseks on kasutatud andmeedastusobjekte. Andmeedastusobjekt ehk DTO on laialt kasutatud disainimuster, mis võimaldab edastada andmeobjekte, mis ei sisalda äriloogikat. Antud tagarakenduses on iga kihi jaoks loodud eraldi andmeedastusobjektid.

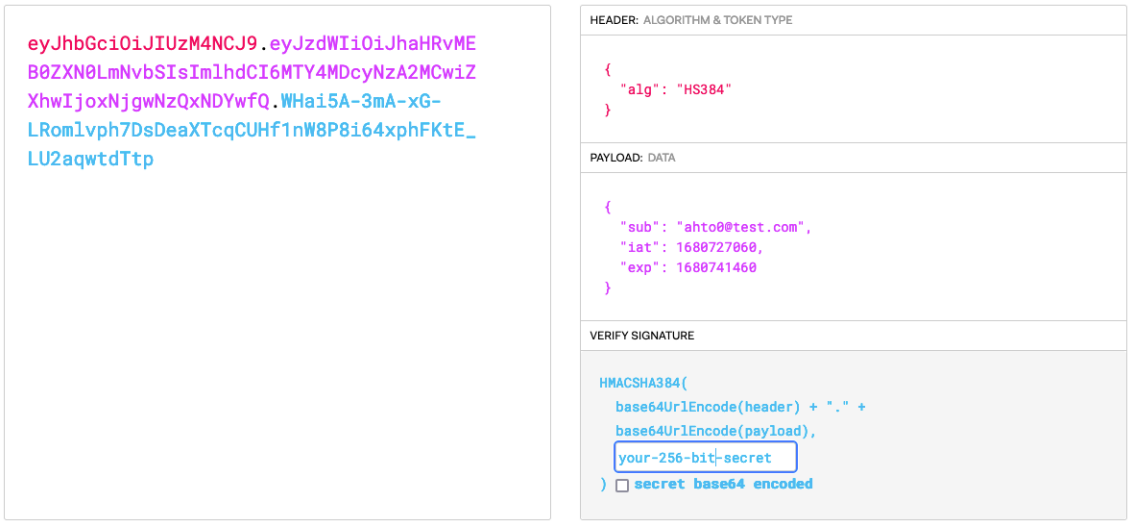
4.2.3 Objekt-relatsioonivastendus

Antud tagarakenduse andmebaasi ja Spring raamistiku sidumiseks on kasutatud objekt-relatsioonivastendust ehk ORM spetsifikatsiooni, mis võimaldab objekt-orienteeritud viisil siduda andmeobjektid andmebaasiobjektidega.

4.2.4 Kasutaja tuvastamine

Andmetele turvalise juurdepääsu tagamiseks on kasutatud REST API päringu päisesse lisatavat JWT-põhine autoriseerimiskoodi, mis tagab päringu teinud kasutaja õigsuse. Allkirjastatud ja kindla kehtivusajaga autoriseerimiskood luuakse kasutaja sisselogimise käigus. Sellise koodi koostamisel kasutatakse krüpteeritud JSON formaati.

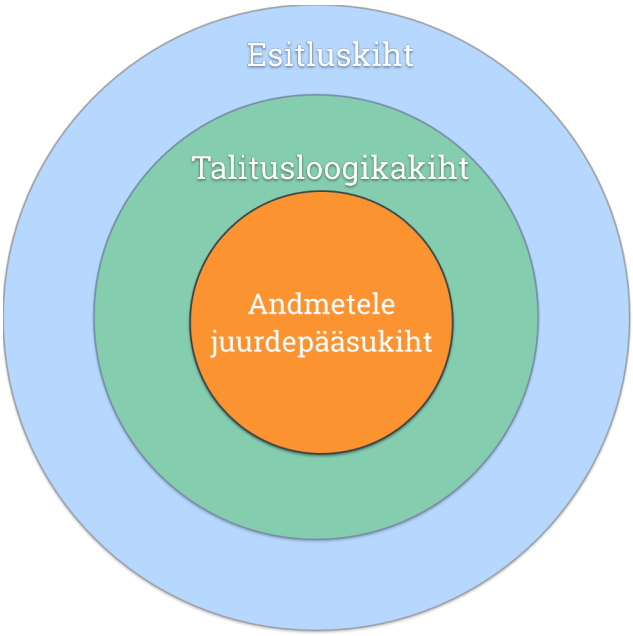
JWT (*JSON Web Token*) on JSON-objektil põhinev kompaktne viis turvaliselt edastada osapooltevahelist teavet, mida saab kontrollida ja usaldada, kuna andmed digitaalselt allkirjastatud. JWT objekti saab allkirjastada salajase HMAC-algoritmil põhineva räsiga või avalik-privaatvõtmepaari abil kasutades RSA või ECDSA algoritmi. JWT objekt võimaldab ka osapooltevahelise salastatuse tagamiseks andmeid krüpteerida [36]. Järgneval joonisel on illustreeritud JWT objekti koostamist, mis koosneb kolmest Base64 formaadis osast: päis, andmeobjekt ja allkiri (Joonis 8).



Joonis 8. JWT struktuur (https://jwt.io)

4.3 Klientrakenduse arendus

Klientrakenduse arhitektuur koosneb kolmest kihist: esitluskiht, talitusloogikakiht ja andmetele juurdepääsukiht (Joonis 9).



Joonis 9. Klientrakenduse kihiline struktuur

Esitluskiht – see kiht sisaldab kasutajaliidese disaini ja struktuuri. See kiht on kasutajale visuaalselt nähtav. Selles kihis vahendatakse kasutaja ja tagarakenduse vahel andmeid. Esitluskihis asub oleku-hoidla, mis võimaldab hallata andmeid, mis on kogutud rakenduse kasutamise käigus.

Talitusloogikakiht sisaldab teenuseid ja komponente, mille abil kirjeldatakse klientrakenduse funktsionaalsus. Teenused väljastavad infot esitluskihile ja vajalikud andmed pärinevad andmetega suhtlemise kihist.

Andmetega suhtlemise kihis kirjeldatakse tagarakendusega suhtlemiseks vajalik loogika kasutades hoidla-mustrit (*repository pattern*). Selles kihis pannakse kokku ja käivitatakse API päringud tagarakendusega suhtlemiseks ning töödeldakse vastuseks saadud JSON formaadis andmeobjektid.

4.3.1 Rakenduse ülesehitus

Klientrakenduse loomisel on kasutatud Vue raamistikku, mis on seadistatud kasutama TypeScript programmeerimiskeelt, millest lähtuvalt on loodud rakenduse ülesehitus.

Klientrakenduse ülesehitus koosneb järgmistest osadest:

- `node_modules` – Node.js moodulid
- `public` – staatilised ressursid, millega käivitatakse rakendus
- `src` – klientrakenduse lähtekood
  - `assets` – staatilised ressursid
  - `bll` – talitusloogikakiht
    - `service` – äriloogikat sisaldavad teenused
  - `components` – taaskasutatavad komponendid
  - `dal` – andmetele juurdepääsukiht
    - `domain` – olemid, andmeedastusobjektid
    - `repository` – andmehoidla funktsionaalsus, HTTP integratsioon
  - `router` – rakenduse vaadete struktuur ja viited komponentidele
  - `stores` – andmeladu, oleku haldamine
  - `views` – vaated koos skriptide ja stiili-komponentidega
  - `App.vue` – rakenduse algus
  - `main.ts` – rakenduse käivitamine
- `tests` – rakenduse testid
- `Dockerfile` – konteineri loomise kirjeldus
- `package.json` – projekti seadistus ja sõltuvuste haldamine
- `tsconfig.json` – TypeScript konfiguratsioon
- `vue.config.js` – Vue konfiguratsioon

4.3.2 Suhtlus tagarakendusega

Antud klientrakenduses on kasutatud REST API tarkvaraarhitektuuri standardit, et tagada klientrakenduse ja tagarakenduse vaheline suhtlus. REST rakendused võimaldavad juurde pääseda ressurssidele kasutades eelnevalt kindlaks määratud ilma olekuta päringuid. Klientrakenduse andmetele juurdepääsu kihis on kasutatud andmehoidla mustrit (*repository pattern*), mis on kombineeritud HTTP päring-vastus mustriga (*HTTP request/response pattern*). HTTP päringute loomiseks on kasutatud Axios moodulit.

Axios on JacaScript programmeerimiskeele põhine HTTP rakendus, mis võimaldab klientrakenduses sooritada asünkroon-andmeedastus operatsioone ja lihtsustada HTTP päringutega opereerimist veebibrauseris. [37]

4.3.3 Olekuhoidla

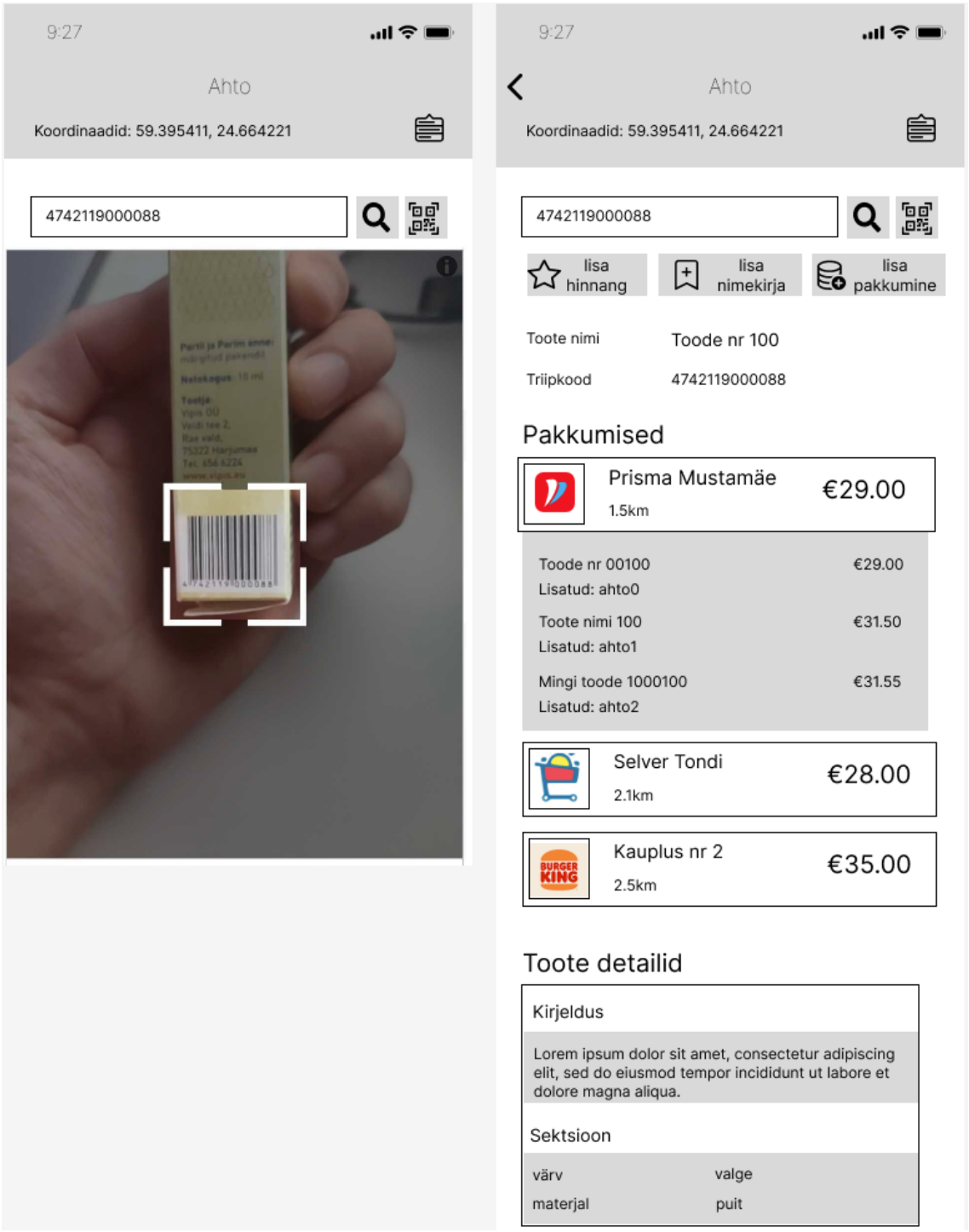
Olekuhoidla on esitluskihis paiknev komponent, mille eesmärk on klientrakendusel salvestada ja hallata rakenduse töö käigus loodud andmeid. Olekuhoidlad on kasulikud ka tagarakendustelt vastuvõetud andmete hoidmiseks, et korduvaid andmeid ei oleks pidevalt vaja uuesti pärida. Olekuhoidla peamine eesmärk on rakenduses olevate andmete jagamine erinevate teenuste ja komponentide vahel. Näiteks võimaldab olekuhoidla hõlpsalt salvestada rakenduses JWT objekt, mis loodi kasutaja sisselogimisel ja hiljem kasutatakse REST API päringute koostamisel kogu rakenduse kasutamise vältel. Antud töös on kasutatud Pinia olekuhaldusmoodulit. Pinia on olekuhoidla teek, mis võimaldab komponentide ja vaadete vahel olekuandmeid jagada. [38]

4.3.4 Kasutajaliidese vaated

Rakendus eeldab kasutajalt funktsionaalsusele ligipääsemiseks kasutajakonto loomist ja sisselogimist. Tuvastatud kasutajale võimaldatakse rollipõhine ligipääs veebirakenduse funktsionaalsusele, mille hulka kuuluvad jälgimisnimekirja loomine, toote arvustuste kirjutamine, tootehindade lisamine ja hinnateavituste seadistamine.

Järgenvalt on välja toodud neli rakenduse funktsionaalsust iseloomustavat vaadet:

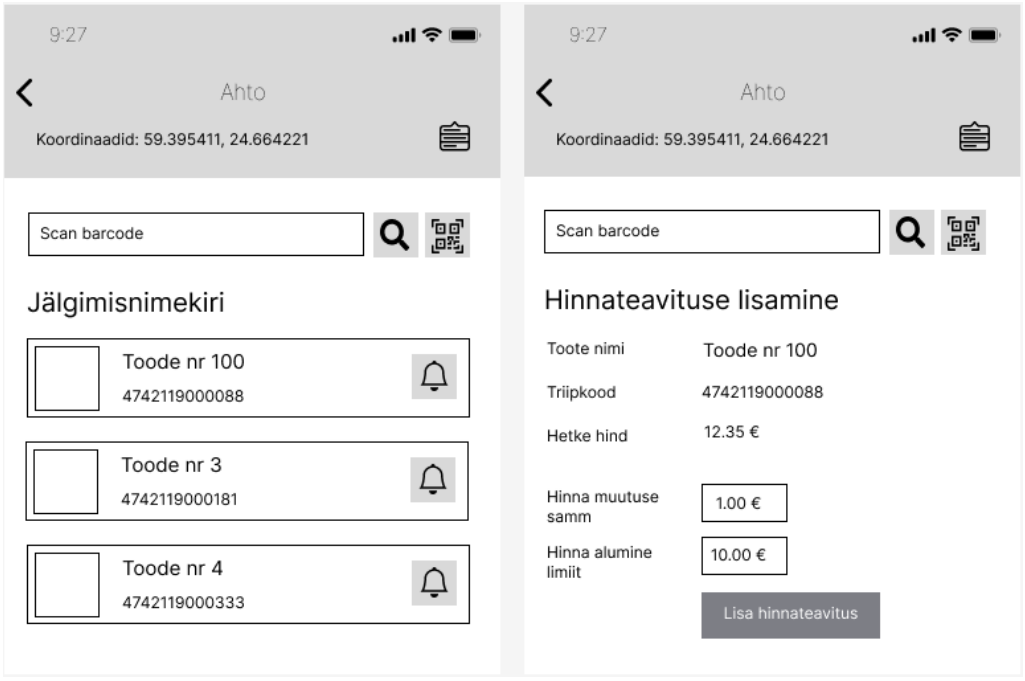
a) Tavakasutaja vaade võimaldab tooteni jõuda kasutades tootekategooria valikuid või sooritades otsing kasutades otsingusõna. Otsingusõna asemel võib kasutada ka triipkoodi, mille sisestamise lihtsustamiseks võimaldatakse kasutada mobiilse seadme kaamera baasil loodud triipkoodilugeja võimekust. Pärast toote valimist avaneb toote detailne vaade, mis sisaldab infot toote kui ka erinevate poodide hinnapakkumiste kohta. Poodide nimekiri on järjestatud kaupleja asukoha järgi arvestades kasutaja enda koordinaate. Tooteleht võimaldab luua kasutajale tervikliku ülevaate toote kohta kogutud andmetest. Tootelehel on võimalik kasutajal lisada toode personaalsesse jälgimisnimekirja, tootele hinnangu kirjutada või lisada info uue pakkumise kohta (Joonis 10).



Joonis 10. Tooteotsing ja tooteleht

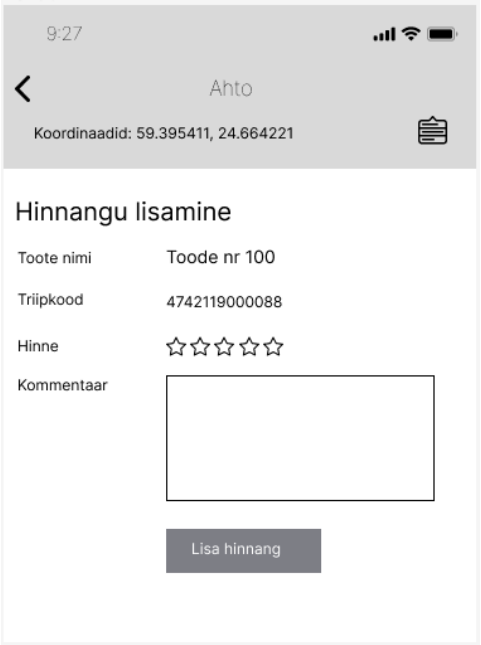
b) Järgneval joonisel on kujutatud jälgimisnimekirja vaadet, kus on iga lisatud toote juures on võimalus seadistada tootele hinnateavitus ja välja on toodud ka hinnateavituse vaade (Joonis 11). Hinnateavituse seadistamise jaoks on vajalik määrata hinnapiiri limiit, millest alla poole langemise korral saadetakse kasutajale teavitus. Lisaks on võimalus määrata teavitamise samm, mille eesmärk on saadetud

teavituste arvu piirata – see tähendab, et korduvteavitus saadetakse pärast kindla suurusega hinnamuutust.



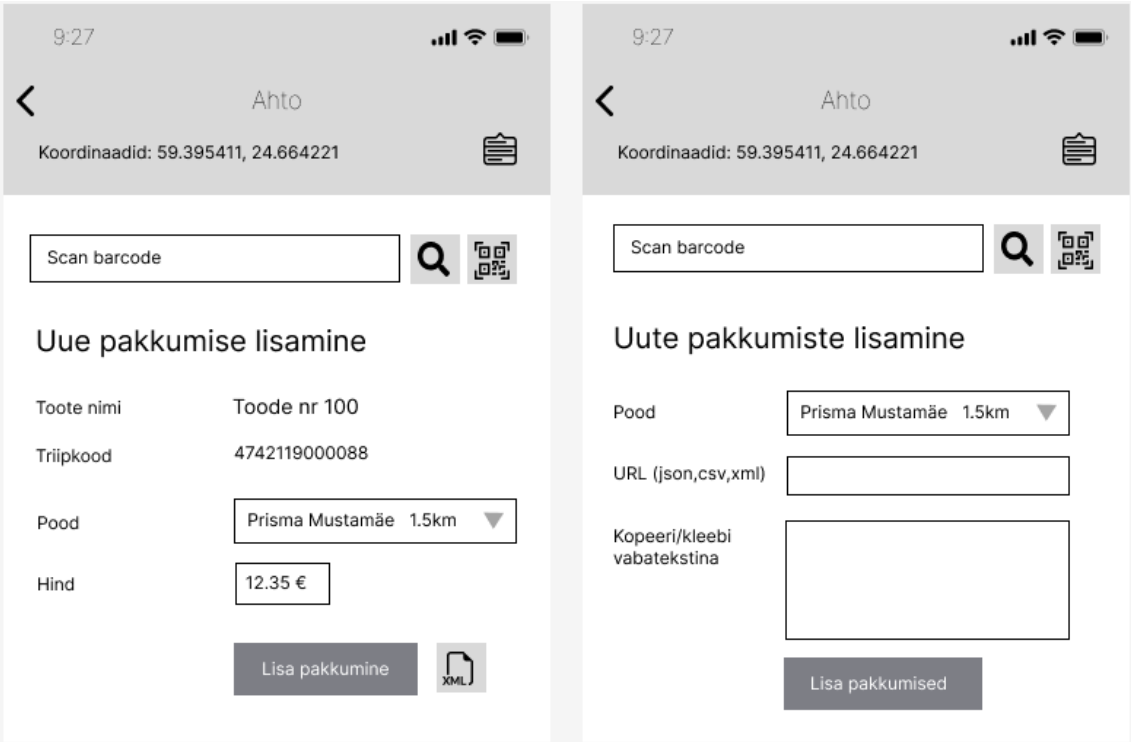
Joonis 11. Jälgimisnimekiri ja hinnateavituse lisamine

c) Järgneval joonisel on kujutatud tootele hinnangu lisamise vaade (Joonis 12). Kasutajal on võimalik hiljem enda poolt koostatud hinnanguid kustutada.



Joonis 12. Hinnangu lisamine tootele

d) Järgneval joonisel on kujutatud tootele uue hinnapakkumise lisamine, milleks on kaks võimalust: hinna lisamine ühele kindlale tootele või pakkumiste lisamine mitmele tootele korraga (Joonis 13). Pakkumise lisamisel on vajalik valida pood, mille kohta pakkumine soovitakse lisada. Kui soovitakse lisada mitmeid tooteid korraga, siis on võimalik seda teha viidates internetis juurdepääsetavale JSON, CSV või XML failile või sisestades CSV formaadis vabatekstina.



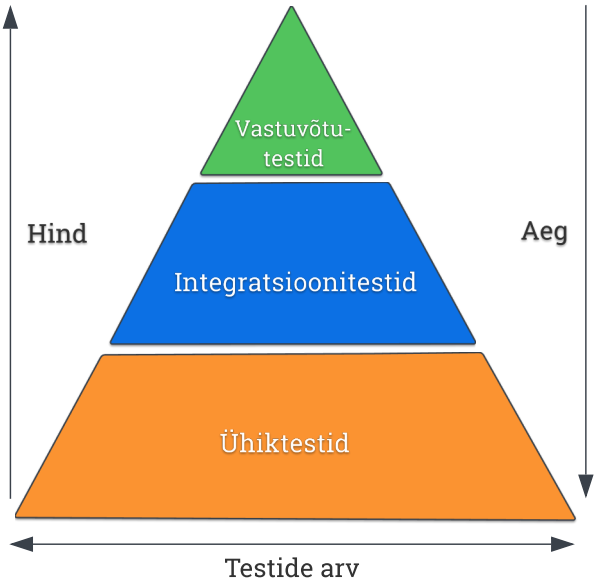
Joonis 13. Uue pakkumise lisamine

4.4 Testimine

Antud lõputöös lähtutakse veebirakenduse funktsionaalsuse testimisel testimispüramiidi loogikat, mis koosneb peamiselt kolme tüüpi testidest: ühiktestid (*unit tests*), integratsioonitestid (*integration tests*) ja vastuvõtutestid (*accepetance tests*).

Testimispüramiidi raamistik toetab kvaliteetse tarkvara loomist ja vähendab arenduseks kulunud aega. Testide rakendamine aitab vähendada arendaja kulutatud aega, et veenduda tehtud muudatuste ärinõuetele vastavuses. [39]

Järgnev joonis näitab, kuidas püramiid kujuneb sõltuvalt testide käivitusajast ja arenduse rahalisest kulust erinevate testitüüpide lõikes (Joonis 14).



Joonis 14. Testimispüramiid

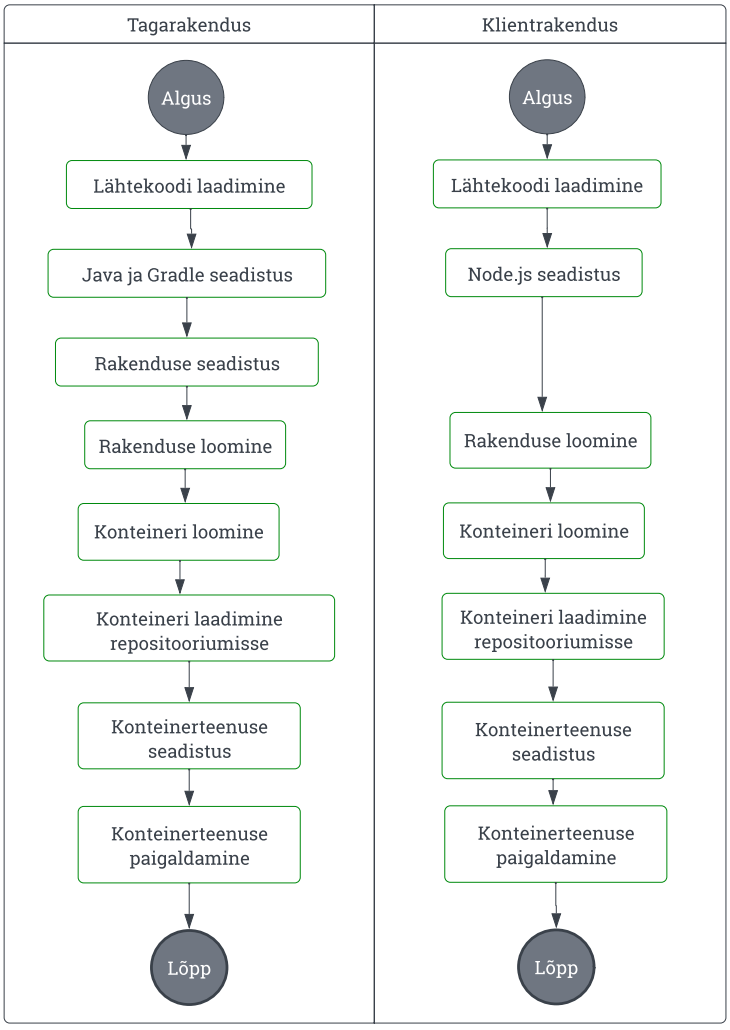
Ühiktestid on testid, millega kontrollitakse üksikute funktsioonide, meetodite või klasside toimimist. Ühiktestimine tagab, et iga kirjutatud meetod toimib vastavalt ettenähtud eesmärgile. Ühikteste on rakenduses loodud kõige rohkem, sest need on kiired, lühikesed ning testivad ühte kindlat olukorda.

Integratsioonitestid on testid, millega kontrollitakse rakenduse kihtide vahelist toimimist. Integratsioonitestimise eesmärk on testida erinevate komponentide koostööd.

Vastuvõtutestide eesmärk on kontrollida toote või teenuse vastavust ärinõuetele ja kriteeriumitele. Testimine aitab veenduda, et süsteem töötab arvestades tervet kasutusvoogu ja on valmis paigaldamiseks toodangukeskkonda.

4.5 Pakendamine ja paigaldamine

Rakenduse serverisse paigaldamiseks on loodud automaatne tööprotsess, mille raames kompileeritakse lähtekood, pakendatakse rakendus konteinerisse ja paigaldatakse pilvekeskkonda (Joonis 15).



Joonis 15. Automatiseeritud paigaldus

Protsessi automatiseerimiseks loodud töövoog on kirjeldatud kasutades Github Actions teenust. Töövoog käivitatakse automaatselt pärast seda, kui lähtekoodi muudatused on lisatud Git repositooriumisse (vt Lisa 3).

5 Kokkuvõte

Antud bakalaureusetöö käigus teostati analüüs rakenduse ärinõuete leidmiseks ja valiti metoodika ning tehnoloogia rakenduse väljatöötamiseks. Lõputöö raames analüüsitakse teemakohase kirjanduse baasil varasemate sarnaste lahenduste nõrku ja tugevaid külgi, valiti välja rakenduse loomiseks kasutatav arhitektuur, kaardistati rakenduse ärinõuded ja funktsionaalsus, loodi kasutusvooskeem, andmemudel, kasutajaliidese prototüüp ning koostati andmebaasi olemi-suhete skeem.

Valminud veebirakendus võimaldab registreeritud kasutajatel sooritada tarbekaupade otsinguid kasutades otsingusõna või triipkoodi ning võrrelda erinevate kauplejate hinnapakkumisi arvestades kaupleja asukohta ja toote hinda. Rakendus annab võimaluse kasutajal tooteid ja uusi hinnapakkumisi andmebaasi lisada ning toodete kohta arvustusi kirjutada. Tooteid on võimalik lisada personaalsesse jälgimisnimekirja ja seadistad hinnateavitusi. Rakenduse asukohapõhisus seisneb peamiselt toodete hinnapakkumiste järjestamises arvestades kasutaja ja kaupleja asukoha vahelist kaugust, mis aitab esile tuua tarbijale kõige lähemal asuvad poed.

Tulevikus planeeritakse edasi arendada lõputöö raames valminud veebirakenduse funktsionaalsusi, mis rakenduse minimaalse töötava versiooni (MVP – *Minimum Viabile Product*) skoopi ei mahtunud. Pikemas perspektiivis planeeritakse luua integratsioone kauplustega, internetipoodide ja muude tooteinfot pakkuvate platvormidega, mis võimaldavad parandada rakenduses leitava info kvaliteeti ja kasvatada kasuliku info mahtu andmebaasis üldisemalt. Täiendavat arendusressurssi vajab ka veebirakenduse olemasolevate komponentide edasiarendamine, et parandada rakenduse töökindlust ja kasutusmugavust. Antud lõputöö tulemusena on loodud hea raamistik, mis võimaldab tarkvaraarenduse eesmärke efektiivselt saavutada.

Bakalaureusetöö on täies ulatuses autori poolt tehtud ja töös püstitatud eesmärgid saavutati. Antud lõputöö tulemusel töötati välja terviklik infosüsteem, mis sisaldab andmebaasi ja pilvekeskkonda paigaldatud veebirakendust.

Kasutatud kirjandus

[1] P. a. G.-K. A. Broeckelmann, „Usage of mobile price comparison sites at the point of sale and its influence on consumers" shopping behaviour," 2008. [Võrgumaterjal]. Available: http://dx.doi.org/10.1080/09593960701868266 (https://www.researchgate.net). [Kasutatud 2023].

[2] E. Konjunktuuriinstituut, „Uuring: Eesti tarbijate teadlikkus on viimase 10 aasta jooksul märgatavalt tõusnud," [Võrgumaterjal]. Available: https://ttja.ee/uudised/uuring-eesti-tarbijate-teadlikkus-viimase-10-aasta-jooksul-margatavalt-tousnud. [Kasutatud 05.2023].

[3] J. F. G. Joseph S. Valacich, Modern Systems Analysis and Design, 8th edition (lk 32-35), 2017.

[4] J. F. G. Joseph S. Valacich, Modern Systems Analysis and Design, 8th edition (lk 189), 2017.

[5] R. Al-Qutaish, Quality Models in Software Engineering Literature: An Analytical and Comparative Study. Journal of American Science, 6(3). (lk 170-171), 2010.

[6] S. Hatton, „Choosing the Right Prioritisation Method (lk 518)," 2008. [Võrgumaterjal]. Available: https://ieeexplore.ieee.org/document/4483241. [Kasutatud 2023].

[7] „O ceneo - Kim jestesmy? - Ceneo Info," [Võrgumaterjal]. Available: https://info.ceneo.pl/o-nas. [Kasutatud 2023].

[8] „Über idealo," [Võrgumaterjal]. Available: https://www.idealo.de/unternehmen/ueber-idealo. [Kasutatud 2023].

[9] „Kaina24.lt - prekių paieška ir kainų palyginimas," [Võrgumaterjal]. Available: https://www.kaina24.lt/apie-projekta. [Kasutatud 2023].

[10] „Hind.ee," [Võrgumaterjal]. Available: https://www.hind.ee/. [Kasutatud 2023].

[11] „Hinnavaatlus - Tehnikakaupade hinnavõrdlus," [Võrgumaterjal]. Available: https://www.hinnavaatlus.ee. [Kasutatud 2023].

[12] „Kainos.lt," [Võrgumaterjal]. Available: https://kainos.lt. [Kasutatud 2023].

[13] „Om Prisjakt," [Võrgumaterjal]. Available: https://www.prisjakt.nu/info/om-prisjakt--i9. [Kasutatud 2023].

[14] „Om PriceRunner," [Võrgumaterjal]. Available: https://www.pricerunner.se/info/about-pricerunner. [Kasutatud 2023].

[15] J. F. G. Joseph S. Valacich, Modern Systems Analysis and Design, 8th edition (lk 242), 2017.

[16] P. Raspel. [Võrgumaterjal]. Available: https://enos.itcollege.ee/~priit/1.%20Andmebaasid/1. Loengumaterjalid/06/66988683001_6-6.htm. [Kasutatud 2023].

[17] „PYPL PopularitY of Programming Language index," [Võrgumaterjal]. Available: https://pypl.github.io/PYPL.html. [Kasutatud 2023].

[18] „About Python,“ [Võrgumaterjal]. Available: https://www.python.org/about. [Kasutatud 2023].

[19] „What is Java? | IBM,“ [Võrgumaterjal]. Available: https://www.ibm.com/topics/java. [Kasutatud 2023].

[20] „A tour of C# - Overview,“ [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/dotnet/csharp. [Kasutatud 2023].

[21] „What is Java Spring Boot? | IBM,“ [Võrgumaterjal]. Available: https://www.ibm.com/topics/java-spring-boot. [Kasutatud 2023].

[22] „Stack Overflow Developer Survey 2022,“ [Võrgumaterjal]. Available: https://survey.stackoverflow.co/2022. [Kasutatud 2023].

[23] „Introduction | Vue.js,“ [Võrgumaterjal]. Available: https://vuejs.org/guide/introduction.html. [Kasutatud 2023].

[24] „Angular,“ [Võrgumaterjal]. Available: https://angular.io/guide/what-is-angular. [Kasutatud 2023].

[25] „React,“ [Võrgumaterjal]. Available: https://react.dev. [Kasutatud 2023].

[26] „TOPDB Top Database index,“ [Võrgumaterjal]. Available: https://pypl.github.io/DB.html. [Kasutatud 2023].

[27] „Database | Oracle,“ [Võrgumaterjal]. Available: https://www.oracle.com/database/. [Kasutatud 2023].

[28] „Microsoft Data Platform | Microsoft,“ [Võrgumaterjal]. Available: https://www.microsoft.com/en-us/sql-server. [Kasutatud 2023].

[29] „MySQL 8.0 Reference Manual - What is MySQL?,“ [Võrgumaterjal]. Available: https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html.

[30] „PostgreSQL: About,“ [Võrgumaterjal]. Available: https://www.postgresql.org/about/. [Kasutatud 2023].

[31] „Top 10 Cloud Service Providers In 2023,“ [Võrgumaterjal]. Available: https://www.c-sharpcorner.com/article/top-10-cloud-service-providers. [Kasutatud 2023].

[32] „Kubernetes Documentation | Kubernetes,“ [Võrgumaterjal]. Available: https://kubernetes.io/docs/home/. [Kasutatud 2023].

[33] „Comparing Cloud Container Services Across AWS, Azure, and GCP,“ [Võrgumaterjal]. Available: https://blogs.vmware.com/cloudhealth/cloud-container-services-aws-azure-gcp. [Kasutatud 2023].

[34] „What is Azure,“ [Võrgumaterjal]. Available: https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-azure/. [Kasutatud 2023].

[35] M. Richards, Software Architecture Patterns (lk 1-2), O’Reilly Media, 2015.

[36] „JSON Web Token Introduction - jwt.io,“ [Võrgumaterjal]. Available: https://jwt.io/introduction. [Kasutatud 2023].

[37] „Axios“ [Võrgumaterjal]. Available: https://axios-http.com/. [Kasutatud 2023].

[38] „Introduction | Pinia,“ [Võrgumaterjal]. Available: https://pinia.vuejs.org/introduction.html. [Kasutatud 2023].

[39] „The Practical Test Pyramid,“ [Võrgumaterjal]. Available: https://martinfowler.com/articles/practical-test-pyramid.html. [Kasutatud 2023].

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

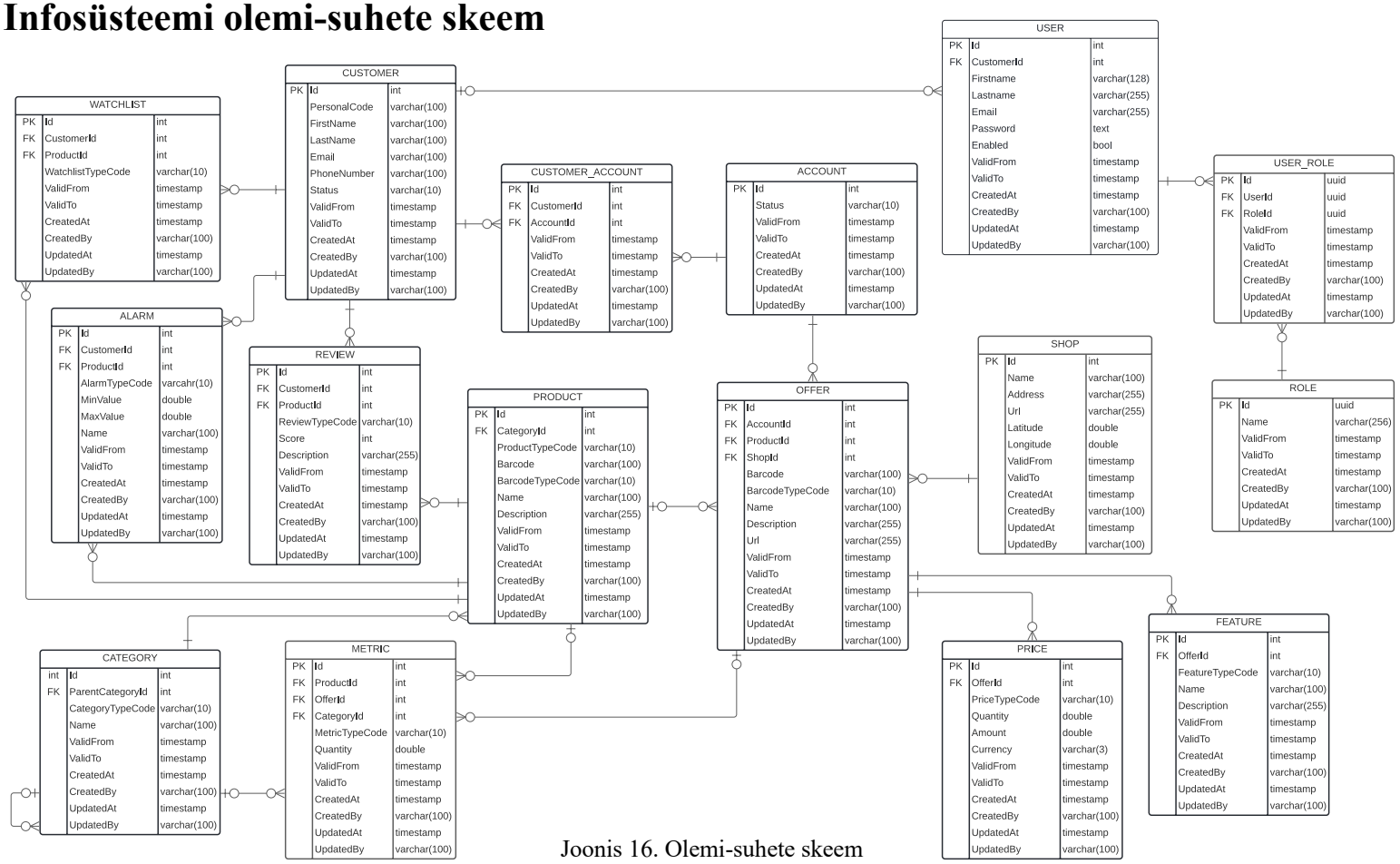
Mina, Ahto Jalak

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Asukohapõhise kaupade hinnavõrdluse veebirakenduse väljatöötamine“, mille juhendaja on Lembit Viilup
   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 – Infosüsteemi olemi-suhete skeem



Joonis 16. Olemi-suhete skeem

Lisa 3 – Viide veebirakenduse lähtekoodile

Antud lõputöös on väljatöötatud veebirakenduse lähtekoodi jaoks loodud avalikult juurdepääsetav Git repositooriumi aadressil: https://github.com/ahtoj/price-comparison

Lähtekood on jaotatud kolmeks osaks: infrastruktuur, klientrakendus ja tagarakendus.