TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Rasmus Vahelaan 206118IADB

Rakendus targa hinnavõrdluse tegemiseks toidukaupadele

Bakalaureusetöö

Juhendaja: Märt Kalmo
MSc

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: Rasmus Vahelaan

15.05.2023

Annotatsioon

Lõputöös käsitletav probleem on seotud toidukaupade ostmisel raha säästmisega. Toidupoodide vaheline hinnakonkurents annab tarbijatele võimaluse raha säästa, kui teha teadlike valikuid ostukorvide koostamisel. Probleemiks on aga tasakaalu leidmine odavama ja mitte liiga ebamugava ostukorvi vahel. Kõige odavama ostukorvi saamiseks tuleb tooteid osta paljudest erinevatest poodidest, kuid see ei pruugi olla mõistlik, kui võtta arvesse ka tarbija mugavust, näiteks transpordiks kuluvat aega.

Probleemist tulenevalt on lõputöö eesmärgiks analüüsi koostamine lahendusele, mis automatiseeriks toidukaupade hinnavõrdluse, aitaks toidukaupade ostmisel raha säästa ja arvestaks seejuures ka tarbija mugavusega. Lisaks on eesmärgiks arendada valmis rakenduse prototüüp, mis vastaks analüüsis määratud nõuetele.

Lõputöö üheks tulemuseks on lahenduse analüüs, kus käsitletakse teemasid kombineeritud ostukorvide koostamisest, läbitava teekonna ja ajakulu leidmisest, andmete kogumisest ja andmete sidumisest, tehnoloogia valikust. Lisaks on lõputöö tulemuseks rakenduse prototüüp, mis võimaldab tarbijatel teha hinnavõrdlust kombineeritud ostukorvide teel ning arvestab ka tarbija eelistustega läbitava teekonna pikkuse ja ajakulu osas.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 42 leheküljel, 7 peatükki, 14 joonist, 1 tabelit.

Abstract

Application for Smart Price Comparison of Food Products

The problem addressed in the thesis is related to saving money when buying groceries. The price competition between grocery stores gives consumers the opportunity to save money by making informed choices when compiling their shopping carts. The problem, however, is finding a balance between a cheaper and not too inconvenient shopping cart. To get the cheapest shopping cart, products must be purchased from many different stores, but this may not be reasonable when considering consumer convenience.

The aim of the thesis is to create an analysis for a solution that would automate grocery price comparison, help save money when buying groceries, and take into account consumer convenience. In addition, the goal is to develop a prototype of the finished application that meets the requirements determined in the analysis.

First result of the thesis is the analysis of the solution, which covers topics such as creating combined shopping carts, finding the best routes and travel times, data collection, data integration, and technology selection. Second result of the thesis is a prototype application that allows consumers to compare prices using combined shopping carts and take into account preferences for travel distance and time. The created application collects price information from two data sources and links the same products in them together. The route length and travel time are calculated using TSP algorithms.

In the author's opinion, the thesis meets the objectives set and the created prototype application proves that such an approach could be one possible solution to the problem addressed in the thesis. However, to confidently claim this, the application should also integrate the other data sources mentioned in the analysis and perform longer-term analysis.

The thesis is in Estonian and contains 42 pages of text, 7 chapters, 14 figures, 1 tables.

Lühendite ja mõistete sõnastik

API	*Application Programming Interface* – rakendusliides
CRON	UNIXi käsurea tööriist ettemääratud tegevuste perioodiliseks täitmiseks
CSS	*Cascading Style Sheets* – keel, mida kasutatakse veebilehtede disaini ja kujunduse määratlemiseks.
EAN	*European Article Number* – rahvusvaheline standardiseeritud ribakoodisüsteem
GDPR	*General Data Protection Regulation* – Euroopa Liidu andmekaitse määrus, mis kaitseb isikuandmeid
HTML	*HyperText Markup Language* – veebilehtede loomiseks kasutatav märgistuskeel
JSON	*JavaScript Object Notation* – andmevahetusformaat, mida kasutatakse andmete edastamiseks veebis
JSX	*JavaScript XML* – JavaScripti laiendus, mis võimaldab kasutada HTML sarnast süntaksit JavaScripti koodis
REST	*Representational State Transfer* – tarkvaraarhitektuuri laad, mis määrab hulga tavasid ja juhiseid veebirakenduste loomiseks
RFC	*Request for Comments* – dokument, mida kasutatakse internetiga seotud spetsifikatsioonide kirjeldamiseks ja levitamiseks
SQL	*Structured Query Language* – keel, mida kasutatakse relatsiooniliste andmebaaside haldamiseks ja päringute tegemiseks
TSP	*Travelling Salesperson Problem* – klassikaline optimeerimisprobleem, mis keskendub minimaalse teekonna leidmisele, et läbida kõik etteantud punktid ja seejärel jõuda tagasi alguspunkti
XML	*Extensible Markup Language* – keel, mida kasutatakse andmete salvestamiseks ja vahetamiseks erinevate süsteemide vahel
XPath	*XML Path Language* – keel, mida kasutatakse XML-dokumendi elementide asukoha leidmiseks

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Kõrge inflatsioon on põhjustanud muuhulgas ka toiduainete hindade hüppelise kasvu Eestis [1]. Hinnatõus paneb tarbijaid eelistama aina enam soodsamaid pakkumisi ning muudab inimesed toitu ostes kaalutlevamaks. Seda kinnitab 2023. aasta jaanuari Postimehe ostukorvi uuring [2], kus 64% vastanutest leidis, et nad on sunnitud sügavalt mõtlema ostukorvi sisu üle. Lisaks erinevate toodete vahel valimisele on tarbijatel enamasti võimalik teha ka valikuid toidupoodide osas. Eesti toidupoodide vahel on reaalne hinnakonkurents, mida teadlik tarbija saaks kasutada oma hüvanguks.

Kõige odavama ostukorvi saamiseks tuleks tooteid osta paljudest erinevatest poodidest. Näiteks Postimehe 2022. aasta detsembrikuu hinnavõrdluse [9] põhjal oleks vajalik maksimaalseks raha säästmiseks osta tooteid kõigist seitsmest uuringus olevast toidupoest. Samas ei pruugi seitsme toidupoe külastamine olla eriti mõistlik, kui võtta arvesse ka tarbija mugavust, näiteks transpordiks kuluvat aega. Tarbijatel on keeruline koostada käsitsi optimaalseid ostukorve, mis arvestaksid nii hinna kui ka mugavusega. Samas puudub ka automaatne hinnavõrdlustarkvara, mis seda nende eest teeks.

Käesolevas lõputöös soovib autor luua lahenduse, mis aitaks tarbijatel automaatselt koostada ostukorve, mis arvestaksid nii hinna kui ka mugavusega. Selleks tehakse esmalt ülevaade probleemi taustast ja olemasolevatest lahendustest, mille põhjal pakutakse välja uue lahenduse visioon. Visiooni realiseerimiseks koostatakse lahenduse analüüs, kus leitakse sobivad meetodid ja tehnoloogiad probleemi lahendamiseks. Visiooni ja analüüsi tulemuste valideerimiseks arendatakse valmis ka rakenduse prototüüp. Viimaks hinnatakse, kas saadud lahendus vastab eesmärkidele ja aitab lahendada algselt sõnastatud probleemi.

2 Ülesandepüstitus

Käesolevas peatükis tutvustatakse lõputöö teemat ja aktuaalsust. Sõnastatakse probleemi olemus ning püstitatakse probleemist lähtuvalt lõputöö eesmärk. Samuti kirjeldatakse lõputöös kasutatavat metoodikat.

2.1 Taust ja aktuaalsus

2023. aasta kõrge inflatsioon on põhjustanud muuhulgas ka toiduainete hindade hüppelise kasvu Eestis. Statistikaameti andmetel [1] oli tarbijahinnaindeksi tõus 2023. aasta jaanuaris võrreldes 2022. aasta jaanuariga 18,6%, millest toidukaupade hinnatõus moodustas 27,4%. Taoline hinnatõus paneb tarbijaid eelistama aina enam sooduspakkumisi ning muudab inimesed toitu ostes kaalutlevamaks. Seda kinnitab ka Postimehe ostukorvi uuring [2], kus koguni 64% vastanutest leidis, et nad on sunnitud sügavalt mõtlema ostukorvi sisu üle. Lisaks erinevate toodete vahel valimisele on tarbijatel enamasti võimalik teha ka valikuid toidupoodide osas. Eesti toidupoodide vahel on reaalne hinnakonkurents, mida teadlik tarbija saaks kasutada oma hüvanguks. Delfi Ärilehe tehtud ostukorvi maksumuse uuring jaanuaris 2023 [3] näitas, et kalleima ja odavaima ostukorvi vahe ulatus koguni 30 euroni. Antud uuringus oli ostukorvide maksumuse standarthälve 10,23 € ja keskmine hind 88,25 €. Statistikaameti andmetel kulus keskmisel leibkonnaliikmel Eestis 2020. aastal aastas toidule 1300 eurot [4], mis oleks 2023. aastal 1747 eurot [5]. Siit saame hinnata, et keskmiselt on aastas võimalik ühe leibkonnaliikme pealt säästa ~202 € (1747 / 88,25 * 10,23) tehes teadlikumaid otsuseid toidupoodide valikul.

2.2 Probleemi kirjeldus

Toidukaupade hinnatõus avaldab survet tarbijatele, kes otsivad toidu ostmiseks kuluefektiivsemaid võimalusi. Toidupoodide vaheline hinnakonkurents annab tarbijatele võimaluse raha säästa, kui teha teadlike valikuid ostukorvide koostamisel. Probleemiks on aga tasakaalu leidmine odavama ja mitte liiga ebamugava ostukorvi vahel. Kõige odavama ostukorvi saamiseks tuleks tooteid osta paljudest erinevatest poodidest, kuid see

ei pruugi olla mõistlik, kui võtta arvesse ka tarbija mugavust, näiteks transpordiks kuluvat aega. Tarbijatel on keeruline koostada käsitsi optimaalseid ostukorve, mis arvestaksid nii hinna kui ka mugavusega. Samas puudub ka automaatne hinnavõrdlustarkvara, mis seda nende eest teeks.

2.3 Eesmärk

Käesoleva lõputöö eesmärgid on järgmised:

- analüüsida võimalusi lahenduse loomiseks, mis automatiseeriks toidukaupade hinnavõrdluse ning aitaks tarbijatel toitu ostes mugavalt raha säästa;
- arendada valmis analüüsist lähtuv rakenduse prototüüp.

2.4 Metoodika

Käesolev lõputöö valmib arendusuuringuna, mille käigus kirjeldatakse esmalt probleem ning püstitatakse sellest lähtuvalt eesmärgid. Valdkonnast parema ülevaate saamiseks uuritakse detailsemalt probleemi tausta ning olemasolevaid lahendusi, mille põhjal on võimalik luua visioon eesmärkide täitmiseks. Analüüsi käigus leitakse sobilikud vahendid ja tehnoloogiad, mis võimaldaksid realiseerida lahenduse visiooni. Lõpuks kirjeldatakse rakenduse teostust ning hinnatakse, kas valminud rakendus aitab lahendada algselt sõnastatud probleemi.

3 Probleemi taust ja lahenduse visioon

Antud peatükis kirjeldatakse lõputöös käsitletava probleemi tausta. Selleks vaadatakse hinnavõrdluse meetodeid ning uuritakse olemasolevaid lahendusi. Lisaks esitab autor oma visiooni käsitletava probleemi lahendamiseks.

3.1 Toidupoodide vahelise hinnavõrdluse meetodid

Toidupoodide vahelise hinnavõrdluse tegemiseks puudub standardne metoodika ning kasutusel on erinevaid lahendusi. Üldjuhul enim kasutust leidnud meetodid on ühikuhindade võrdlemine kaubagrupi tasemel, sarnaste omadustega toodete võrdlemine, toodete üks-ühele võrdlemine ja ostukorvi kogumaksumuse võrdlemine. Igal lähenemisel on oma eelised ja puudused, millega tuleks arvestada.

3.1.1 Ühikuhindade võrdlemine kaubagrupi tasemel

Vastavalt Eestis kehtivatele tarbijakaitseseadusele [6] on suuremas osas müüdavatele kaupadele kohustuslik lisada ühikuhind, mis näitab kauba lõpphinda vastavalt standardsele mõõtühikule, näiteks ühe kilogrammi või ühe liitri hind. Tarbijatel on võimalik tänu ühikuhinnale kergesti võrrelda sama kaupa, mis on pakendatud erineva suurusega pakenditesse. Hinnavõrdluse tegemine ühikuhinda arvestades aitab leida pikas perspektiivis kõige soodsama pakkumise, isegi kui algul tuleb toote eest maksta suurem summa. Ühikuhinna kasutamisel tuleb aga arvestada, et raske on võrrelda konkreetseid tooteid omavahel, kuna ühikuhind on arvutuslik hind, mille juurde ei arvestata tarbija eelistusi ega ka toodete erinevaid omadusi, nagu päritolu, kvaliteet, pakend jm. Seega kõige efektiivsem on ühikuhinna baasil hinnavõrdlust teha ühe terve kaubagrupi või tootekategooria kohta, kus eesmärgiks on leida lihtsalt kõige odavam hinna-koguse suhe ning samas ei ole nii olulised selle toote omadused või tarbija eelistused. Ühikuhinna tasemel hinnavõrdlust toidukaupadele pakub vaikimisi veebikeskkond Ostukorvid.ee ning üldist hinnainfot ühikuhinna tasemel kogub ka Eesti Konjunktuuriinstituut [7].

3.1.2 Sarnaste omadustega toodete võrdlemine

Sarnaste omadustega toodete võrdlemise meetod on kasutusel toidupoodide vahelises hinnavõrdluses mitmete Eesti juhtivate uudisteportaalide poolt, näiteks kasutatakse seda Postimehe [8] või Delfi Ärilehe [3] koostatud uuringutes. See meetod põhineb esmalt toote üldisel defineerimisel, näiteks „piim kiles, 1l“ või „pakk võid“. Seejärel otsitakse definitsioonile vastavalt kõige soodsam toode igast toidupoest ning leitud hindu võrreldakse omavahel. Sellisel viisil hinnavõrdluse tegemine on natukene täpsem kui ühikuhinna alusel hinnavõrdlus, kuna vastavalt definitsiooni täpsusele on võimalik arvestada ka mõningate toote omaduste ja tarbija eelistustega. Siiski võib paljudel juhtudel jääda definitsiooni ebamääraseks ning omavahel võrreldakse tooteid, mis on erineva kvaliteedi, päritolu, koguse või kaubamärgiga. Näiteks Delfi Ärilehe koostatud toidupoodide vahelises hinnauuringus 2023. aasta jaanuaris [3] määrati definitsiooniks eelnevalt mainitud „või pakk“ ning omavahel võrreldi 150 grammist kohalikku Eesti toodet 200 grammise importtootega Leedust. Taoliste hinnavõrdluste korral võib tarbijatel jääda ekslikult mulje, et saadud on kõige parem pakkumine, kuigi tegelikult ei pruugi leitud toode olla optimaalseim valik nii ühikuhinna arvutamisel ega ka toote omaduste, näiteks kvaliteedi, poolest. Samas on sarnaste omadustega toodete võrdlemine definitsiooni järgi kasulik, kui toidupoodide vahel puuduvad üks-ühele võrreldavad tooted.

3.1.3 Toodete üks-ühele võrdlemine

Eesti Konjunktuuriinstituudi tehtud uuringust [9] selgub, et lisaks hinnale peavad tarbijad toidukaupade ostmisel väga tähtsaks ka kvaliteeti, värskust, maitset ning mõnevõrra tähtsaks tuttavat kaubamärki, toote välimust, koostist ja päritolu. Sellest saab järeldada, et tarbijad valivad poest tihti toodet tervikuna, mitte ei keskendu üksnes kõige odavamale pakkumisele. Kauba omaduste ja tarbija eelistustega enim arvestav hinnavõrdluse meetod on toodete üks-ühene võrdlemine. Sellisel võrdlusel võetakse aluseks konkreetse toote, näiteks „Pereviiner Rakvere, 500g“, müügihind ning leitakse täpselt sama toote hind teistes toidupoodides. Sellisel viisil hindade võrdlemine võimaldab näiteks tarbijatel kergelt leida oma lemmikkaupade kampaaniapakkumised, mida poodide kogemuse järgi [10], [11] ka aina enam otsitakse. Antud hinnavõrdlusmeetodi juures on aga probleemiks toidupoodide sortimentide erinevus ning seetõttu ei pruugi võrreldavat toodet alati igas poes leiduda. Samuti on olemas ka võimalus, et toote üks-ühele võrdlemisel on hinnavahed toidupoodide vahel pea olematud. Siiski eksisteerib üldjuhul Eesti

toidupoodide vahel reaalne hinnakonkurents nii tava- kui ka soodushindades. Toodete üks-ühele hinnavõrdlus on kasutusel portaalides Ostukorvid.ee, Hind24.ee ning ka Coopi poolt tellitud hinnavõrdluses, mis ilmus Delfi Ärilehes 2022. aasta septembris [12]. Samade toodete võrdlemisel saadud tulemused on tänu oma läbipaistvusele heaks aluseks ka toidupoodide üldiste hinnatasemete leidmisel.

3.1.4 Ostukorvide kogumaksumuse võrdlemine

Toidupoodide reaalsetest hinnatasemetest ülevaate saamiseks on vaja eelnevates peatükkides 3.1.1 – 3.1.3 kirjeldatud meetodite kasutamine agregeeritud kujul, kuna üksikute toodete või kaubagruppide tasemel hinnavõrdluse tegemine ei arvesta asjaoluga, et üldjuhul ostetakse toidupoest mitu toodet korraga. Sellise hinnavõrdluse tegemiseks on parim lahendus kasutada ostukorvide kogumaksumuse meetodit. Ostukorv on kujutletav kogum, mis koosneb üksikutest toodetest või kaubagruppidest. Ostukorvi kogumaksumus arvutatakse rakendades ühte peatükkides 3.1.1 – 3.1.3 kirjeldatud meetoditest ning seeläbi saadud tulemused koondatakse toidupoodide lõikes. Ostukorvide kogumaksumuste võrdlemine on üldjuhul tarbijatele kõige mugavam viis toidupoodide vahelise hinnavõrdluse tegemiseks. Seetõttu märgitakse ostukorvide kogumaksumus ära Eesti juhtivate uudisteportaalide poolt tehtud uuringutes, näiteks Postimehes [8] või Delfi Ärilehes [3]. Lisaks on antud meetodi põhjal saadud tulemused laialdaselt kasutusel erinevate toidupoodide reklaammaterjalides, näiteks Coopi toidupoe hüüdlause „Pere ostukorv on soodsaim Coopis“ [13]. Ostukorvi kogumaksumuse põhjal tehtud hinnavõrdlustes on aga oluline jälgida, et kasutusel võib olla erinevaid alusmeetodeid üksikutele toodetele hindade leidmiseks, mis võivad muuta hinnavõrdluse tulemust. Samuti võib tarbija ostukorv olulisel määral erineda uuringutes koostatud ostukorvidest ja seeläbi ka hinnast.

3.2 Olemasolevate lahenduste ülevaade

Toidukaupade hinnavõrdluse tegemiseks on olemas mitmeid erinevaid lahendusi. Lahenduste seas leidub nii igakuiselt käsitsi koostatud uuringuid kui ka automaatseid rakendusi, mis jälgivad reaalaja hindasid poodides. Lisaks on hinnainfo jälgimist käsitletud Eestis ka ülikoolide diplomitöödes. Samas on olemasolevad lahendused autori hinnangul tihti pealiskaudsed, ebamugavad või ei täida antud lõputöös püstitatud eesmärki.

3.2.1 Toidukaupade hinnainfo jälgimist käsitlevad diplomitööd

Autorile teadaolevalt on Eestis kaitstud vähemalt kaks diplomitööd, mille teemad on seotud toidukaupade hinnainfo jälgimise mugavamaks tegemisega.

Anastassia Rogatšova Tartu Ülikoolis 2021. aastal koostatud lõputöös „Veebirakenduse loomine kauplusketide sooduskampaaniate vahendamiseks“ [14] keskendutakse toidupoodide sooduskampaaniate koondamisele ühte keskkonda, mis peaks kampaaniate leidmise tarbija jaoks lihtsamaks tegema. Lisaks on võimalus kasutajatel näiteks kampaaniapakkumisi salvestada, kuid samas puudub funktsionaalsus pakkumiste võrdlemiseks. Antud rakenduse juures on puuduseks asjaolu, et kajastatakse ainult sooduspakkumisi, mis moodustavad toidupoodide üldistest sortimentidest marginaalse osa ning seeläbi ei pruugi tarbijat igapäevaselt piisavalt säästmisel aidata.

Ahmed Abdullajevi Tallinna Tehnikaülikoolis 2022. aastal koostatud bakalaureusetöös „Rakendus hindade jälgimiseks toidupoodides“ [15] kuvatakse toidukaupade hindasid kogudes selleks perioodilist andmeid kaupluste e-poodidest. Tarbijal on võimalik näha iga toote kohta ühiku- ja tavahinda ning samuti ka kampaaniapakkumisi. Antud lahenduse juures on probleemiks, et erinevates poodides olevad samad tooted on kuvatud dubleeritult, mis teeb tarbijale hindade võrdlemise väga ebamugavaks. Lisaks ei ole antud rakenduses ka automaatse hinnavõrdluse funktsionaalsust ning hindade võrdlemine on jäetud täielikult tarbija enda ülesandeks. Seega ei lahenda antud rakendus probleemi, mis on seotud toidukaupade hinnavõrdluse mugavaks tegemisega.

3.2.2 Meediaväljaannete koostatud ostukorvide uuringud

Ilmselt kõige suurem osa teabest, mis on seotud toidukaupade ja -poodide hinnatasemetega, jõuab Eesti tarbijateni läbi juhtivate meediaväljaannete koostatud uuringute. Postimehes ilmuvad teatud regulaarsusega „Postimehe ostukorv“ [8], milles võrreldakse ostukorvide maksumust nii e-poodides kui ka kauplustes kohapeal, ja „Postimehe e-ostukorv“ [16], kus hinnavõrdlust tehakse ainult e-poodide põhjal. Samuti ilmuvad ostukorvide hinnavõrdlused aeg-ajalt ka Delfi Ärilehes [3]. Peamiseks probleemiks meediaväljaannete poolt koostatud ostukorvide uuringus on enamasti ebatäpse hinnavõrdlusmeetodi kasutamine, kuna üksikute toodete hinnavõrdlust tehakse sarnaste omadustega toodete võrdlemise meetodil (vt peatükki 3.1.2). Seda probleemi on muuhulgas käsitletud ka Delfi Ärilehes ilmunud artiklis „Kas ostukorvi hinnavõrdlustes

võrreldakse võrreldavaid tooteid?“ [12]. Siinjuures erineb teistest taolistest uuringutest „Postimehe e-ostukorv“, kus on hinnavõrdlusesse kaasatud kolm toodet fikseeritud brändi alusel. Meediaväljaannete poolt koostatud uuringute puhul on tavaliselt probleemiks veel asjaolu, et tarbija ostukorv võib olulisel määral erineda uuringus toodud ostukorvist ning seeläbi on tarbijal raske leida just temale kõige soodsamat ostukohta [12]. Lisaks ilmuvad meediaväljaannetes ostukorvide uuringud parimal juhul korra kuus, mis ei ole aga piisav aktuaalse hinnainfo edastamiseks.

3.2.3 Toidukaupade hinnavõrdluse rakendused Eesti turul

Autor on täheldanud, et Eesti turule on tekkinud viimasel ajal uusi ettevõtteid ja keskkondi toidukaupade hinnavõrdluse tegemiseks. Näiteks 2023. aasta jaanuaris registreeritud ettevõte BeboTech OÜ [17] lubab turule tulla keskkonnaga, kus oleks võimalik otsida Eesti suurimatest toidupoodidest tooteid, need ostukorvi lisada ning seeläbi mugavalt ostukorvide hinnavõrdluse abil raha säästa [18]. Siiski töö kirjutamise hetkel on Eesti turul olemasolevaid ja töötavaid hinnavõrdluse lahendusi peamiselt kahte tüüpi:

- sooduskampaaniaid vahendavad keskkonnad;
- üksikute toodete hinnainfo jälgimist võimaldavad portaalid.

Eesti toidupoodides olevaid sooduskampaaniaid on võimalik jälgida rakendusest Saleapp.ee või portaalist Kliendilehed.ee. Mõlemad keskkonnad on mõeldud selleks, et koondada erinevates toidupoodides olevad sooduspakkumised ühte keskkonda. Seejuures ei ole aga tarbijal võimalik näha poodides kehitavaid tavahindasid, mis on ka mõlema keskkonna kitsaskohaks.

Üksikute toodete hinnainfo täpset jälgimist võimaldavad teha portaalid Ostukorvid.ee ja Hind24.ee. Mõlemas portaalis kuvatakse tarbijatele toodete hindu erinevates toidupoodides. Hind24.ee lehel ei ole aga kuvatud kaupade ühikuhinda, puudub otsing ning toodete sortiment on võrdlemisi väike. Samas on aga hinnainfot koguni 10 erineva toidupoe kohta. Ostukorvid.ee portaalis on andmeid paljude erinevaid toodete kohta, näha saab nii toote kui ka ühiku hinda, võimalik tutvuda toote hinna ajalooga ning kasutada ka otsingut. Kummaski portaali ei ole võimalik aga koostada ostukorve ning seetõttu on suur osa hinnavõrdluse tegemisest jäetud taaskord tarbija ülesandeks.

3.3 Loodava rakenduse visioon

Toidukaupade hinnavõrdlusel tasakaalu leidmine odavama ja mitte liiga ebamugava lahenduse vahel ei ole iseenesest mõistetavalt lihtne probleem. Arvestada tuleb paljude erinevate teguritega, puuduvad olemasolevad piisavalt head lahendused ning samuti ühtne hinnavõrdluse metoodika. Probleemi lahendamiseks oleks vajalik luua uus lahendus targa hinnavõrdluse tegemiseks.

Loodav targa hinnavõrdluse rakendus peaks võimaldama tarbijatel koostada personaalseid ostukorve ning näitama toidupoodide reaalaja hinnainfo põhjal, kuidas on võimalik ostukorvis olevaid tooteid odavalt ja mugavalt soetada. Targa hinnavõrdluse rakendus keskenduks ainult toodete, mitte kaubagruppide, põhistele ostukorvidele, kuna siis arvestatakse kõige täpsemalt tarbija eelistustega kauba kvaliteedi, päritolu, brändi jm osas (vt peatükki 3.1). Kirjeldatav rakendus võiks oma ülesehituselt sarnaneda modernse e-poe või toidutellimise rakendusega, et kasutuskogemus oleks tarbijale võimalikult tuttav. Esmalt valib tarbija endale ostukorvi soovitud tooted ning seejärel saab näha hinnavõrdlust.

Targa hinnavõrdluse rakendus võiks odavama hinna leidmiseks hinnavõrdlusmeetodeid kombineerida. Näiteks oleks tarbijatel võimalik toidukaupade ostmisel rohkem raha säästa kui teha hinnavõrdlust üksikute toodete tasemel. Samas võib üksikute toodete võrdlemisel saadud tulemus olla aga ebamugav, kuna on võimalik, et ostud tuleks sooritada ebamõistlikul arvul paljudest poodidest. Postimehe 2022. aasta detsembrikuu hinnavõrdluses [8] saaks maksimaalselt raha säästa, kui osta tooteid kõigist seitsmest uuringus olevast toidupoest. Tarbijatel on palju mugavam võimalus raha säästa, kui kasutada ostukorvide kogumaksumuse võrdlemist ning osta tooted kõik ühest poest. Seejuures ei pruugi ostukorvide võrdlemisel saadud hinnavõit olla aga piisavalt suur. Seega tuleks võimalusel ikkagi tooteid osta erinevatest poodidest, kuid teha seda optimaalselt. Targa hinnavõrdluse rakendus peaks koostama tarbijatele selliseid hinnavõrdlusi, kus näidatakse millistest poodidest toitu ostes saaks tarbija oma ostukorvi kõige odavamalt soetada. Seejuures peab aga tarbija saama ise valida, kui mitut toidupoodi on ta nõus maksimaalselt ühe ostukorra ajal külastama.

Tarbijad on harjunud külastama toidupoode, mis asuvad kodule või töökohale kõige lähemal [19] ning seejuures hindavad ka, et oste saaks teha võimalikult vähese ajaga [20].

Seega on mitme erineva kaupluse külastamisest tingitud ebamugavuse kompenseerimiseks väga oluline hinnavõrdlusel arvestada toidupoe asukoha ning sinna jõudmiseks kuluva ajaga. Targa hinnavõrdluse rakendus võiks arvestada tarbija hetke asukohaga ning leida hinnavõrdlusse poed tema lähiümbrusest. Seejuures peaks tarbija saama ka ise valida, kui pika teekonna on ta nõus läbima ja/või kui palju aega kulutama, et jõuda kõikidesse hinnavõrdluse poolt pakutud poodidesse.

4 Lahenduse analüüs

Antud peatükis määratakse täpsed nõuded rakenduse prototüübi loomiseks võttes aluseks autori pakutud lahenduse visiooni. Lisaks analüüsitakse lahenduse visiooniga seotud olulisemate aspektide võimalikke lahendusi.

4.1 Nõuete määramine

Loodavale lahenduse funktsionaalsete ja mittefunktsionaalsete nõuete määramisel võetakse aluseks töös püstitatud probleem ning autori pakutud lahenduse visioon. Seejuures tuleb nõuete määramisel ka arvestada, et antud lõputööle kehtivad ranged ajalised piirangud, mistõttu valmib töö käigus rakenduse prototüüp, mille peamine eesmärk on lahenduse kontseptsiooni valideerimine. Ühtlasi määravad lahendusele esitatud nõuded ka antud töö skoobi.

4.1.1 Funktsionaalsed nõuded

Loodav lahendus peab võimaldama järgnevaid funktsionaalsusi:

1. rakendus peab automaatselt jälgima toidukaupade hindasid;
2. rakendus peab siduma erinevates poodides olevad samad tooted omavahel;
3. rakendus peab kuvama tooteid kategooriatesse jaotatuna;
4. rakendus peab võimaldama otsingut toote nime järgi;
5. rakendus peab nimekirja vaates kuvama iga toote juures müügi- ja ühikuhinda kasutades „hind alates“ kontseptsiooni;
6. rakendus peab võimaldama toodete ostukorvi lisamist ja eemaldamist;
7. rakendus peab arvestama tarbija asukohaga ning hinnavõrdlusse leidma toidupoed tema ümbrusest;
8. rakendus peab võimaldama kasutajal valida limiidi, kui mitmest poest on ta nõus maksimaalselt toidukaupu ostma;
9. rakendus peab võimaldama kasutajal valida limiidi, kui pika teekonna on ta nõus toidukaupade ostmiseks läbima;

10. rakendus peab võimaldama kasutajal valida limiidi, kui palju aega on ta nõus toidupoodidesse liikumiseks kulutama;
11. rakendus peab näitama, millistest toidupoodidest on tarbijal kõige mõistlikum ostukorvis olevad tooted osta, arvestades seejuures ka kasutaja poolt määratud limiitidega;
12. rakendus peab meelde jätma kasutaja poolelioleva ostukorvi sisu ja määratud limiidid.

4.1.2 Mittefunktsionaalsed nõuded

Loodav lahendus peab toetama järgnevaid mittefunktsionaalseid nõudeid:

1. rakendus peab olema kättesaadav nii arvutist kui ka mobiilsetest seadmetest;
2. rakenduses peab olema vähemalt kahe toidupoe hinnainfo.

4.2 Kombineeritud ostukorvi koostamine

Kombineeritud ostukorv aitaks tarbijatel leida soodsaima ostukorvi võttes arvesse toidukaupade hindasid mitmes erinevas poes. Autorile teadaolevalt ei ole mitme erineva poe lõikes odavaima ostukorvi leidmiseks loodud meetodeid ega algoritme, mida saaks antud töös kasutada. Seega oleks vajalik vastav algoritm ise koostada. Antud peatükis on odavaima kombineeritud ostukorvi leidmiseks vajalikud järgnevad eeldused:

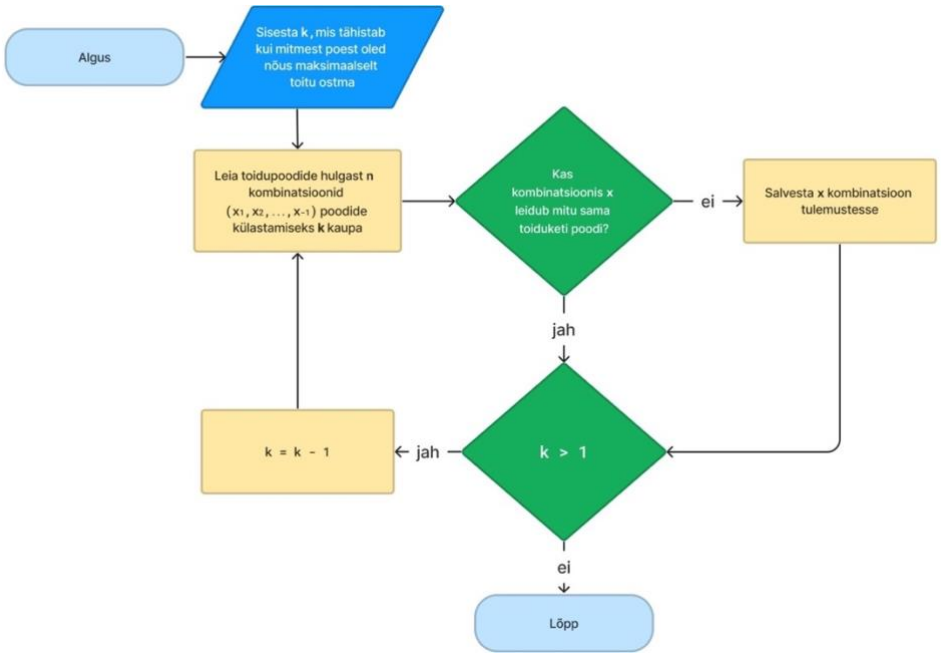
- on olemas kogum võimalikest toidupoodidest ning seejuures on antud kogum eelnevalt filtreeritud tarbija asukoha põhiselt ja võetakse arvesse ainult tarbija lähedal asuvaid toidupoode;
- on teada tarbija ostukorv;
- on teada üksikute toodete maksumus igas toidupoes.

Eeldusel, et toidupoodide kogum on eelnevalt filtreeritud, saame oletada, et võimalike toidupoodide arv on võrdlemisi väike. Seega oleks loodava algoritmi puhul võimalik kasutada jõumeetodit, kus kõige odavam ostukorv leitakse kõikide võimalike kombinatsioonide läbivaatamisel. Samuti võiks algoritmi esmasel koostamisel hoiduda enneaegset optimeerimisest, kuna tegelikult võib juba optimeerimata algoritm olla piisavalt kiire ülesande lahendamiseks. Sel juhul oleks optimeerimine ebavajalik ja raiskaks väärtusliku aega arenduseks. Kui loodud algoritm osutub aga rakenduse teostuse ajal selgelt liiga aeglaseks, siis on võimalik leida ka tagantjärele võimalusi algoritmi

optimeerimiseks, näiteks vahemälu kasutamine, lisanduvad filtrid sisendandmetele või paralleelne protsessimine.

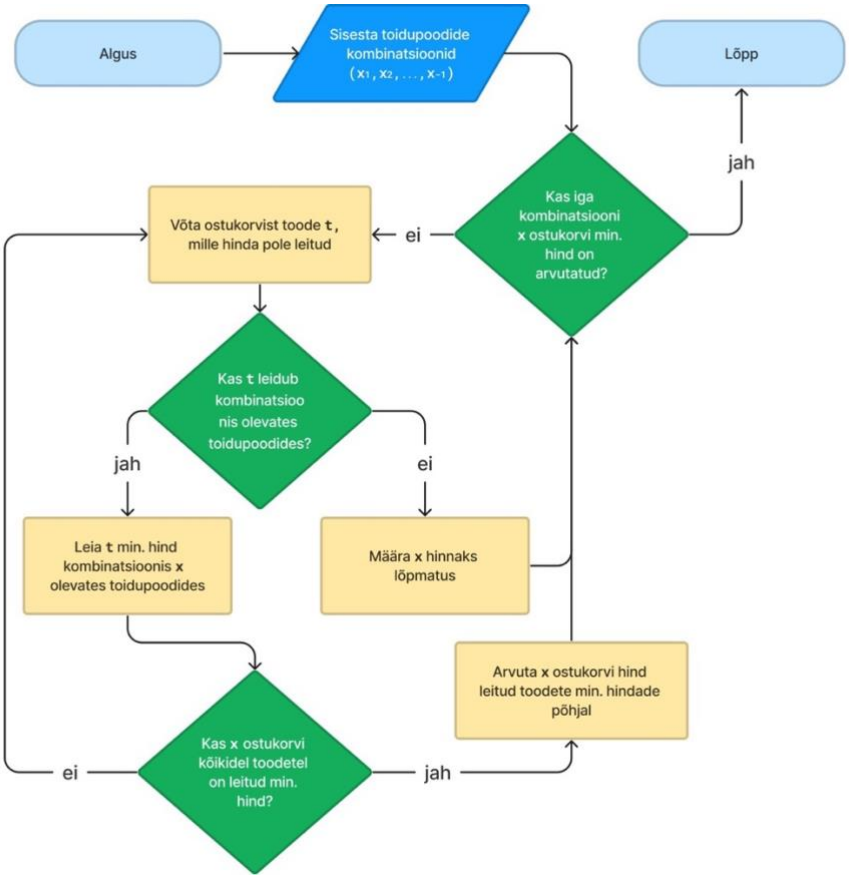
Kombineeritud ostukorvi algoritm peaks koosnema kolmest suuremast osast. Esimene osa peaks leidma kõik võimalikud kombinatsioonid toidupoodide külastamiseks, teine osa peaks leidma iga kombinatsiooni korral ostukorvi hinna ning kolmas osa leidma lõpuks kõige odavama(d) ostukorvi(d).

Toidupoe külastamiseks võimalike kombinatsioonide leidmiseks oleks esmalt vajalik tarbija sisend **k**, mis tähistab kui mitmes toidupoes on tarbija maksimaalselt nõus käima. Seejärel on võimalik toidupoodide kogumist **n** leida kombinatsioonid **k** kaupa. Kui **k** on aga suurem kui 1, siis tuleb leida kombinatsioonid ka kõikidel **k - 1** juhtudel. See on vajalik, kuna peame arvestama ka olukorraga, kus odavaim ostukorv võib koosneda vähematest poodidest kui on tarbija määratud limiit. Näiteks tarbija sisend **k** on 3, aga tarbija ostukorvis olevad tooted saab tegelikult kõige soodsamalt kahest poest ostes. Lisaks oleks vajalik iga kombinatsiooni **x** korral kontrollida, kas kombinatsioonis olevad toidupoed on kõik erinevatest toidupoekettidest. Kui kombinatsioonis leidub mitu toidupoodi samast ketist, siis ei oleks vajalik antud kombinatsiooniga arvestada, kuna ühes ketis olevates poodides on hinnad enamjaolt samad. Joonisel 1 on esitatud kombinatsioonide leidmise algoritm plokkskeemina.



Joonis 1. Kombinatsioonide leidmise algoritmi plokkskeem.

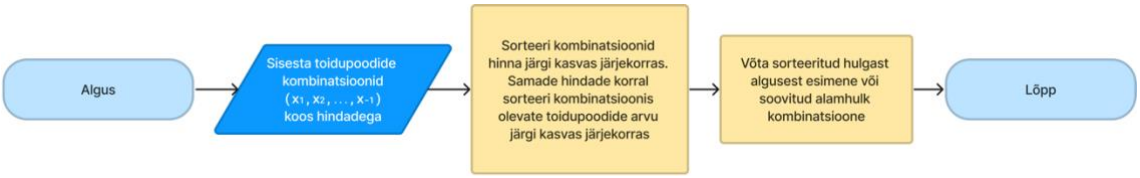
Pärast kõikide kombinatsioonide ($\mathbf{x}_1$, $\mathbf{x}_2$, ..., $\mathbf{x}_{-1}$) leidmist tuleks arvutada ostukorvi hind iga kombinatsiooni **x** korral. Selleks tuleks võtta ostukorvist üks toode **t** ning leida kombinatsioonis olevatest toidupoodidest toote **t** odavaim hind. Ostukorvist tuleks võtta tooteid ükshaaval seni, kuni ostukorvis on iga toote kohta leitud minimaalne hind. Toote hinna leidmisel tuleks aga arvestada ka olukorraga, kus kombinatsioonis olevates toidupoodides antud toodet ei leidu. Sellisel juhul ei oleks mõistlik enam antud kombinatsiooni ostukorviga arvestada, kuna see ei täidaks tarbija soove. Ebasobiva ostukorvi semantiliselt lihtsaks määramiseks oleks hea kasutada näiteks lõpmatust, kuigi algoritmi teostuse ajal ei pruugi olla see kõige optimaalsem valik. Kui aga kombinatsiooni **x** ostukorvis on kõikide toodete minimaalsed hinnad leitud, siis on võimalik leida kombinatsiooni **x** ostukorvi koguhind. Joonisel 2 on esitatud ostukorvi hinna leidmise algoritm kombinatsioonide kasutamise korral.



Joonis 2. Kombinatsiooni ostukorvi hinna leidmise algoritmi plokkskeem.

Kõige odavama või odavamate ostukorvide leidmisel oleks mõistlik arvestada kahe tingimusega. Esiteks peaks olema ostukorvi hind minimaalne ning teiseks võrdsete

hindade korral tuleks eelistada kombinatsiooni, mis koosneb vähimal arvul poodidest. Seda oleks võimalik lihtsasti saavutada näiteks kombinatsioonide sorteerimistel kasvas järjekorras võttes arvesse eelnevaid tingimusi. Taolisest sorteeritud kombinatsioonide kogumi algusest on võimalik võtta esimene kõige odavam kombinatsioon või pakkuda tarbijale valikut mitmest soodsast ostukorvist ja võtta teatud suurusega alamhulk. Joonisel 3 on toodud kõige odavama ostukorvi leidmise algoritmi plokkskeem.



Joonis 3. Odavaima ostukorvi leidmise algoritmi plokkskeem.

4.3 Läbitava teekonna ja ajakulu leidmine

Tarbija läbitava teekonna ja ajakulu leidmiseks oleks vajalik teada toidupoodide asukohti ning võimalust määrata ka tarbija asukoht. Antud peatükis eeldame, et toidupoodide asukohad on juba eelnevalt andmebaasis olemas. Tarbija asukoha määramiseks on tänapäeval nii veebi- kui ka mobiiliplatvormidel olemas vastavad rakendusliidesed: veebirakendustel Geolocation API [21], Apple mobiilirakendustel Core Location [22] ning Android mobiilirakendustel Google'i asukohateenused [23]. Seetõttu saame kindlad olla, et olenevalt rakenduse platvormi valikust on võimalik lihtsasti määrata ka tarbija asukohta.

Arvestades toidupoodide asukohti ja nende vahemaid üksteisest, on loodavas rakenduses probleemiks leida võimalikult lühike marsruut, mis külastab igat kombineeritud ostukorvis olevat toidupoodi täpselt ühe korra ja jõuab alguspunkti tagasi. Seejuures tuleb arvestada, et toidupoodide läbimise järjekord võib muuta teekonna pikkust ning ajakulu. Tegemist on raskelt lahenduva ülesandega, mida tuntakse ka kui *Travelling Salesperson Problem* (TSP) ehk rändmüüja probleemina [24].

Ülesande lahendamiseks tuleks esmalt leida kogu toidupoodide hulgast välja ainult need poed, mis asuvad tarbija määratud limiidi sees. Kuna on teada nii tarbija asukoha koordinaadid kui ka toidupoodide koordinaadid, siis on võimalik filtreerimiseks kasutada Haversine kauguse valemit. Haversine valem arvutab pikkus- ja laiuskraadide põhjal kahe punkti vahelise otsese kauguse meetrites [25]. Edasi lahendamiseks on vajalik aga

arvestada ka teepikkustega, mitte ainult kaugustega linnulennult. Selleks eksisteerib turul erinevaid teenuseid, kuid antud töö käigus analüüsitakse kahte valdkonda juhtivat teenusepakkujat Google'i ja Mapboxi.

Google pakub marsruutide ja vahemaade leidmiseks teenust nimega Routes API, mis võimaldab leida optimaalseid marsruute kahe või rohkema kindlaksmääratud asukohapunkti vahel. Google Routes API marsruutide leidmise funktsionaalsus eeldab, et asukohapunktide läbimise järjekord on ette teada ning seetõttu ei saa seda mõistlikult antud rakenduses kasutada. Lisaks võimaldab Google Routes API leida marsruudimaatrikseid mitme erineva asukohapunkti vahel [26]. Marsruudimaatriksi olemasolul on võimalik TSP lahendamine ise implementeerida või kasutada valmis lahendusi. Valmis lahenduste kasutamise eeliseks on üldjuhul arendusel võidetav aeg ning tõenäoliselt paremini optimeeritud algoritmid. Näiteks Google OR-Tools on eraldiseisev avatud lähtekoodiga tarkvara, mis võimaldab maatriksi põhjal just TSP probleemi lahendada [27]. Google OR-Tools'i on võimalik lisada teegina rakendustesse, mis kasutavad Python, Java või C# programmeerimiskeelt [28]. Google'il on olemas ka Routes API edasiarendus nimega Routes Preferred API, mis peaks juba teenuse tasemel toetama optimeerimisprobleemide lahendamist, kuid see ei ole hetkel veel avalikult kõigile kättesaadav [29].

Mapbox pakub sarnaselt Google'i pakutavatele teenustele võimalust leida kindlas järjekorras olevate asukohapunktide optimaalseim marsruut Directions API-ga või marsruudimaatriksid Matrix API-ga. Lisaks on Mapboxil olemas ka Optimization API, mis on mõeldud täpselt TSP probleemi lahendamiseks [30]. Optimization API võimaldab leida kuni 12 erineva sisendkoordinaadi vahel kõige optimaalsema marsruudi kõikide asukohtade läbimiseks ning alguskohta tagasi jõudmiseks. Sisendkoordinaatide arvu limiit ei tohiks seejuures loodavas rakenduses probleemiks olla, kuna Eesti jaeturu suurima osa moodustavad umbes 5-6 erinevat toiduketti [31]. Optimization API kasutamise korral tuleb aga arvestada, et lühim marsruut leitakse lähtuvalt ajakulust, mitte teepikkusest. Seega võib tekkida olukord, kus ajaliselt kõige kiirem marsruut ei ole alati teepikkuse poolest lühim ning sel juhul tarbija määratud limiitidega võrdlemine ei pruugi anda päris täpset tulemust. Autori arvates võib antud töö skoobis seda probleemi aktsepteerida.

Google ja Mapboxi poolt pakutavate teenuste analüüsil selgus, et Mapboxil on olemas täpselt TSP probleemi lahendav teenus. Seega on loodavas rakenduses mõistlik valida Mapboxi Optimization API teenus, kuna valmis teenuse kasutamine vähendab arenduseks kuluvat aega ja seda on ka arendajal mugavam rakendusse integreerida. Lisaks oleks Mapboxi teenuse kasutamine ka soodsam võrreldes Google'iga. Mapbox võimaldab teha ühes kuus kuni 100 000 päringut tasuta Optimization API teenusesse [32]. Google võimaldab maksimaalset tasuta kuus kasutada Routes API maatriksite leidmisel 40 000 elementi, seejuures üks element tähistab leitavas maatriksis ühte sissekannet [33].

4.4 Andmete kogumine

Loodava rakenduse üheks olulisemaiks osaks võib pidada toidukaupade hinnainfo automaatset kogumist. Hinnainfo saamiseks oleks võimalik teha näiteks liidestusi toidupoodide laohaldusplatvormidega või kasutada avalikult kättesaadavaid andmeid toidukettide e-poodidest. Liidestuste loomine otse laohaldusplatvormidega tundub üsna ebatõenäoline, kuna selliste liidestuste tegemiseks oleks vaja ka toidupoodide poolne panus. Seejuures on kaheldav, et toidupoodidel oleks vastav motivatsiooni ja tahtmine panustada projekti, mis ei loo nende jaoks otsest väärtust. Toidukettide e-poodides on olemas suur osa kaupluste sortimentidest ning seal olev hinnainfo on avalikult kõigile kättesaadav. Seega on antud töö kontekstis kiirem ja mõistlikum kasutada hinnainfo kogumiseks e-poode.

4.4.1 Veebikraapimise meetodid

E-poodidest andmete automaatseks kogumiseks ehk veebikraapimiseks on peamiselt kaks erinevat võimalust. Esimeseks võimaluseks on e-poe HTML-i allalaadimine ning seejärel sealt huvipakkuvate andmete eraldamine [34]. Taolist võimalust saab kasutada, kui e-poe HTML genereeritakse serveri poolel. Kuna HTML-is on andmed üldjuhul struktureerimata kujul, siis huvipakkuvate andmete kättesaamiseks on vajalik teha eeltööd. HTML-ist on võimalik andmeid ükshaaval välja sõeluda kui kasutada näiteks CSS või XPath selektoreid [35]. Selektorite kasutamise jaoks on vaja teada huvipakkuvate andmete täpset asukohta veebilehe struktuuris. Kuna selektorid on üldjuhul väga tihedalt seotud struktuuriga, siis võib veebilehe HTML-i uuenduste käigus tekkida probleem, kus määratud selektor ei leia enam otsitavaid andmeid lehelt. Seega määrab antud meetodil andmete kogumise keerukuse suuresti veebilehe struktuur ja

HTML-i kvaliteet. Teiseks andmete kogumise võimaluseks on avalikult ligipääsetavate rakendusliideste ehk API-de kasutamine [34]. Avalikult kättesaadavad võivad olla näiteks API-d, mida kasutatakse JavaScripti põhistel veebilehtedel serverist andmete pärimiseks ilma autentimiseta. Kuna API-des on andmed esitatud struktureeritud formaadis, näiteks JSON-is või XML-is, siis on antud meetodit kasutades huvipakkuvate andmete kättesaamine võrdlemisi lihtne. Samuti on API-de andmestruktuur stabiilsem ning see muutub tavaliselt vähem kui veebilehtede HTML. Seega tuleks esmalt e-poodidest andmete kogumisel eelistada API-põhiseid meetodeid ning seejärel mõelda HTML-ist huvipakkuvate andmete sõelumisele. Tavaliselt välistab aga ühe meetodi olemasolu teise kasutamise. Näiteks, kui andmed laetakse e-poodi API-st, siis serveri poolt genereeritud HTML-is neid andmeid ei ole.

4.4.2 Veebikraapimise legaalsus

Hinnainfo automaatsel kogumisel e-poodidest peaks tähelepanu pöörama asjaolule, kas selline tegevus on üldse lubatud ja seaduslik. Euroopa Liidus ja sealhulgas ka Eestis võib antud valdkonda pidada pigem halliks alaks, kus küll otseselt veebikraapimist keelavaid seaduseid või muid õigusakte pole [36]. Samas tuleks internetist andmete kogumisel olla ikkagi ettevaatlik, kuna on olemas oht rikkuda mitmeid teisi andmetega seotud seaduseid, eeskirju või kokkuleppeid, mis võivad kaasa tuua juriidilised tagajärjed. Antud peatükis analüüsitakse loodava rakenduse vastavust Euroopa Liidu andmebaasiseadusele, võimalikku ohtu e-poodide kasutustingimuste rikkumisele ning vastavust robotite välistamise standardile. Analüüsist jäetakse välja privaatsusega seotud küsimused, näiteks GDPR, kuna loodavas rakenduses ei plaanita koguda isikuandmeid.

Euroopa Liidu direktiivi 96/9/EÜ sätestab [37] andmebaaside õiguskaitse, mille alusel on võimalik andmebaase kaitsta järgnevatel alustel:

- andmebaasi struktuuri autoriõiguse kaudu;
- andmebaasi andmeid *sui generis* õiguse kaudu.

Kuna e-poe puhul on tegemist andmebaasis olevate andmete esitusega, siis tuleks veebikraapimisel ka antud direktiiviga arvestada. Eelkõige on vajalik tähelepanu pöörata andmebaaside *sui generis* kaitsele. Vähemtähtis on mõelda andmebaaside autoriõiguste kaitsele, kuna antud töös loodav rakendus ei kopeeriks ega reprodutseeriks teistes rakendustes kasutusel olevate andmebaaside struktuure. Vastavalt direktiivile on

andmebaasi sisu kopeerimist ja selle avalikult taaskasutamist võimalik keelata, kui andmebaasi looja suudab tõestada, et on teinud andmebaasi koostamiseks või esitamiseks kvalitatiivselt ja/või kvantitatiivselt olulisi investeeringuid [38]. Seega oleks loodava rakenduse tegevus vastuolus antud määratlusega ning toidukettidel oleks teoorias võimalik piirata e-poodides oleva hinnainfo veebikraapimist. Samas jääb määratluses segaseks oluliste investeeringute skoop ja suurus. Kuna antud direktiiv on loodud 1996. aastal, siis on kahtluse alla seotud selle ajakohasust tänapäeva andmepõhises maailmas. Samuti on antud direktiiviga seotud praktika erinev teooriast. Euroopa Liidu Kohus lõi 2021. aastal antud valdkonnas pretsedendi, kui CV-Online Latvia ja Melons-i vahelises kohtuprotsessis [39] otsustati, et andmebaasi loojal on direktiivist tulenev *sui generis* õigus siis, kui andmete kopeerimise ja selle taaskasutamisega seotud toimingud ohustavad andmebaasi looja suutlikust oma esialgseid investeeringuid tagasi saada. Lisaks leiti otsuses, et kui andmebaasi sisu on avalikult kättesaadav, siis võivad kasutajad ja konkurendid seda kasutada uute innovatiivsete toodete loomiseks, seejuures näiteks aidata kaasa konkurentsi sujuvale toimimisele ning hindade ja pakkumiste läbipaistvusele [40]. Euroopa Liidu Kohtu täpsustused direktiivi 96/9/EÜ osas annavad autori hinnangul alust arvata, et antud töös loodav rakendus on kooskõlas Euroopas kehtiva andmebaasiseadusega.

E-poe kasutustingimused võivad samuti keelata lehel veebikraapimise tegemist ning kasutustingimuste rikkumine võib kaasa tuua nõudeid lepingu rikkumise eest. Üldjuhul on veebilehe kasutustingimused juriidiliselt siduvad alles peale kasutaja selget nõusolekut, näiteks peale nupu vajutust või märkeruudu täitmist [41]. Seega veebilehe tavalisel sirvimisel, nagu seda teeks veebikraapimine, kasutustingimused tegelikult juriidiliselt siduvad ei pruugi olla. Samas on autori arvamusel oluline veebist andmete kogumisel eetiliselt käituda ning järgida tingimusteta ka e-poodide kasutustingimusi. Uurides Eesti toidukettide e-poode (Selver, Rimi, Barbora, Prisma, Coop), siis töö kirjutamise hetkel ei leidu ühegi e-poe kasutustingimustes veebikraapimise või muu sarnase tegevuse keeldu. Seega puudub autori hinnangul oht, et loodav rakendus läheks vastuollu Eesti e-toidupoodides määratud kasutustingimustega.

RFC 9309 ehk robotite välistamise standard [42] võimaldab veebilehtede omanikel piirata automaatsete klientide ehk robotite ligipääsu oma lehele. Samas ei paku antud standart otseselt kaitset veebirobotite eest, vaid annab robotitele juhised, milliseid osasid lehest võib kasutada ning milliseid mitte. Antud standard on kokkuleppeline ning sellel

puuduvad seaduslikud alused [43]. Samas on tegemist hea tavaga veebirobotite kasutamisel. Kuna veebikraapimise tegemiseks kasutatakse automaatseid kliente, siis tuleks antud töös loodaval rakendusel sellele standardile vastata. Robotite välistamise standardis kasutatakse juhiste määramiseks robots.txt faili, mis paigutatakse veebilehe juurkausta. Lisas 2 on esitatud ülevaade Eesti e-toidupoodide robots.txt failidest koos selgitustega. Sealt järeldub, et Rimi e-pood on ainukesena andnud kõikidele robotitele ligipääsu kogu veebilehe sisu piirangutega kasutamiseks. Nii Coop, Prisma kui ka Barbora keskkonnad on seadnud sisse piiranguid privaatsete ja parooliga kaitstud lehtede, näiteks „/profile“ või „/maksmine“, kasutamiseks. Samas pole nendel e-poodidel seatud piiranguid hinnainfo kogumise jaoks olulistele lehtedele, nagu Prisma e-poes „/products/*“, Barboras „/toode/*“ või Coopis „/tooted/*“. Seega ei sega Prisma, Coopi ja Barbora piirangud antud töös loodava rakenduse tegemist. Kõige rohkem piiranguid leiab Selveri e-poest, kus muuhulgas on piiratud ligipääsu ka hinnainfo kraapimiseks olulisele „/api“ otspunktile. Selveri e-poest automaatseks andmete kogumiseks oleks vajalik küsida vastavat luba.

4.5 Andmete integreerimine

Vastavalt nõuetele peab rakendus siduma erinevates poodides olevad samad tooted üheks tooteks. Probleemiks on aga asjaolu, et e-poodides on kuvatud toodete infot erineval määral, kujul ja vormis. Seega oleks vajalik leida võimalusi, kuidas olenemata e-poes kuvatud infost oleks võimalik andmeid siduda. Lisaks oleks vastavalt nõuetele vajalik leida kaks sobivat andmeallikat rakenduse prototüübi jaoks. Autori hinnangul peaksid olema kaks andmeallikat piisavad lahenduse visiooni valideerimiseks. Tulevikus on võimalik andmeallikaid lihtsasti juurde lisada võttes aluseks juba olemasolevad näited.

4.5.1 Andmete sidumine

Loodavas rakenduses oleks mõistlik erinevates e-poodides olevate samade toodete sidumiseks kasutada lihtsasti võrreldavaid ja masinloetavaid andmeid. Sellised masinloetavad andmed on toodete ribakoodid, toodete nimed ja toodete omadused. Antud töö skoobis ei oleks mõistlik andmete sidumiseks kasutada näiteks tootefotosid, kuna need ei ole kergesti võrreldavad ja sellise lahenduse loomine oleks liialt ajakulukas.

Eestis on kaupade kodeerimiseks kasutusel EAN (*European Article Number*) ribakoodid. Ribakoodi puhul on tegemist toote identifikaatoriga, mis on terves maailmas unikaalne

[44]. Seega on ribakoodi järgi toodete sidumine väga lihtne ja täpne meetod. Probleemiks seejuures on aga asjaolu, et osades Eesti e-toidupoodides pole toodete juures ribakoode välja toodud.

Üheks alternatiiviks oleks toodete sidumine nime järgi, kuna e-toidupoodides on toodete nimed väga täpsed ja sisaldavad ka infot koguse kohta. Seega on antud meetod võimalik kõikide toodete puhul olenemata e-poest, kuna ei vaja võrdlemiseks lisainformatsiooni. Seejuures võib aga probleemiks olla, et toodete nimesid kirjutatakse e-poodides erinevalt. Toote nime järgi sidumine töötab hästi ainult siis, kui nime kirjapilt on üks-ühele sama või äärmisel juhul väga sarnane. Antud meetodit oleks võimalik täiustada, kui toodete nimesid enne võrdlemist eeltöödelda, näiteks eemaldada nimest ebaolulised sümbolid ja kirjavahemärgid. Lisaks aitaks antud meetodit täiustada ka sünonüümide kasutamine. Näiteks kui andmebaasis on juba seotud ribakoodi järgi mõned tooted, siis saab nende seotud toodete nimesid ära kasutada võrdlemisel.

Tooteid oleks võimalik siduda ka omaduste alusel. Selleks tuleks valida hulk toote omadusi ning vaadata, kas kahel võrreldaval tootel need kattuvad. Euroopa Liidus müüdavatele toidukaupadele on kohustuslik lisada toitumisalane teave. Kuna toitumisalane teave koosneb vähemalt seitsmest parameetrist (energiasisaldus, rasvad, küllastunud rasvhapped, süsivesikud, suhkrud, valgud ja sool), siis on see üpris toote spetsiifiline ning seda saaks kasutada hästi võrdlemiseks [45]. Seejuures jääb siiski võimalus, et erinevatel toodetel on täpselt sama toitumisalane teave. Seetõttu oleks mõistlik lisada võrdlusse ka toote kaubamärk ja kogus. Toote ja koostisosade kirjeldusi on raske sidumiseks kasutada, kuna need on üldjuhul esitatud vaba tekstina ning seejuures võib nende kirjapilt e-poodides päris palju erineda. Samuti ei oleks mõistlik toodete sidumiseks kasutada toote päritolu, kuna toidupoed võivad sama toodet osta sisse erinevatest riikidest. Antud meetod ei pruugi olla kõige täpsem, kuid selle kasutamine on heaks alternatiiviks, kui ribakoodi või toote nime järgi sidumine ei ole võimalik.

Toodete sidumiseks kasutatavate meetodite analüüsil selgus, et ühte ja ainukest meetodit, mida oleks võimalik alati kasutada ei leidu. Esmalt tasuks eelistada toodete sidumist kasutades ribakoode, kuna see on kõige täpsem ja lihtsam meetod. Samas kui e-poes ei ole toote ribakoodi välja toodud, siis on võimalik kasutada ka toote nime või omaduste järgi sidumist. Loodavas rakenduses tuleks suure tõenäosusega kasutusele võtta kõik kolm meetodit.

4.5.2 Andmeallikate valik

Antud töö nõuetest tulenevalt oleks vajalik leida kaks andmeallikat loodava lahenduse valideerimiseks. Andmeallikate valikul tuleks arvestada eelnevalt analüüsitud aspektidega, nagu andmeallikast andmete kogumise meetod, andmete kogumise legaalsus ja võimalused kogutud andmete sidumiseks. Lisaks toodete hinnainfole oleks vajalik leida andmeallikast ka toidupoodide asukohad, et neid kasutada läbitava teekonna ja ajakulu leidmisel.

Vaadates Eesti jaekettide turujõu järgi [46] suurimaid toidukette, siis puudub töö kirjutamise hetkel e-pood ainult Lidli toiduketil. Autori hinnangul pole antud töös Lidli puudumine suureks probleemiks, kuna Lidli kaupluste sortiment on väga erinev teistest Eesti toidupoodidest. Seega ei sobi Lidl hästi kokku kombineeritud ostukorvide loogikaga ega ka loodava rakenduse visiooniga. Samuti ei arvestata andmeallikatena antud töös toidutellimise rakendusi, nagu Bolt Food või Wolt, kuna nendes rakendustes ei kuvata toidupoodide soodushindu. Lisaks on nendes rakendustes toidupoodide sortimendid väiksemad kui vastavate kettide e-poodides. Tabelis 1 on toodud ülevaade loodava rakenduse võimalikest andmeallikatest.

Tabel 1. Ülevaade Eesti e-toidupoodide kasutamisest andmeallikatena.

Jaekett	**Andmeallikad**	**Andmete kogumise keerulisus**	**Andmete kogumise piirangud**	**Andmete sidumine**
Prisma	**Tooted:** prismamarket.ee **Asukohad:** prismamarket.ee/store/list	**Tooted ja asukohad:** Keskmine – andmeid tuleks sõeluda HTML-ist.	Puuduvad	1. Ribakood 2. Toote nimi 3. Toote omadused
Rimi	**Tooted:** rimi.ee/epood **Asukohad:** rimi.ee/kauplused	**Tooted:** Raske – andmeid tuleks sõeluda HTML-ist, kuid HTML-is on vähe täpseid selektoreid otsitavatele andmetele. **Asukohad:** Keskmine – andmeid tuleks sõeluda HTML-ist.	Puuduvad	1. Toote nimi 2. Toote omadused

Jaekett	**Andmeallikad**	**Andmete kogumise keerulisus**	**Andmete kogumise piirangud**	**Andmete sidumine**
Maxima	**Tooted:** barbora.ee **Asukohad:** maxima.ee/kauplus eketid	**Tooted:** Keskmine – andmeid tuleks sõeluda HTML-ist. **Asukohad:** Lihtne – andmeid saaks koguda API-st.	Puuduvad	1. Toote nimi 2. Toote omadused
Selver	**Tooted:** selver.ee **Asukohad:** selver.ee/kauplused	**Tooted ja asukohad:** Lihtne – andmeid saaks koguda API-st.	RFC 9309 standardi piirang „/api“ otspunktile. Vajalik Selveri nõusolek.	1. Ribakood 2. Toote nimi 3. Toote omadused
Coop	**Tooted:** vandra.ecoop.ee või hiiumaa.ecoop.ee **Asukohad:** coop.ee/kauplused	**Tooted:** Lihtne – andmeid saaks koguda API-st. **Asukohad:** Keskmine – andmeid tuleks sõeluda HTML-ist.	Puuduvad	1. Ribakood 2. Toote nimi 3. Toote omadused

Andmeallikate analüüsil selgub, et loodavas rakenduse prototüübis oleks kõige mõistlikum andmeallikateks valida Selveri ja Coopi e-poed, kuna nendes saab andmete kogumiseks kasutada API-t ja toodete sidumiseks ribakoode. Seejuures tuleks aga arvestada, et Coop erineb teistest jaekettidest, kuna Coopi puhul on andmeid võimalik koguda ainult piirkondlikest e-poodidest. Piirkondlikest e-poodidest kogutud hinnainfo ei pruugi alati olla kõige täpsem üle-eestilise hinnavõrdluse tegemiseks, kuid autori hinnangul on antud kontekstis see risk aktsepteeritav. Selveri e-poest andmete kogumiseks on vajalik küsida vastavat nõusolekut, mida autor ka teha plaanib. Kui andmete kogumiseks nõusolekut ei ole võimalik saada, siis oleks mõistlik Selveri asemel kasutada teise andmeallikana Prismat. Prismast kogutud andmeid on võimalik siduda ribakoodide alusel, mida ei ole võimalik teha näiteks Barborast või Rimist kogutud andmete korral.

4.6 Tehnoloogiate valik

Esmalt tuleks loodava rakenduse tehnoloogiate valikul lähtuda töös määratud nõuetest ning lahenduse analüüsist. Määratud nõuetest on oluline jälgida kahte järgnevat punkti:

- rakendus peab automaatselt jälgima toidukaupade hindasid;
- rakendus peab olema kättesaadav nii arvutist kui ka mobiilsetest seadmetest.

Ülejäänud nõuded ei oma tehnoloogia valikul erilist rolli, kuna nende täitmine on võimalik suvaliste programmeerimiskeelte ja raamistikega. Samuti ei selgunud ka lahenduse analüüsi käigus ühtegi otsest piirangut tehnoloogia valikute osas. Seega ei piisa ainult määratud nõuetest ja lahenduse analüüsist, et leida paljude võimalike valikute hulgast mõistlikus kogus võrreldavaid tehnoloogiaid. Seepärast otsustati tehnoloogiate valikut piiritleda lähtudes enim kasutatavatest tehnoloogiatest. Tehnoloogia populaarsus on oluline aspekt uue rakenduse arendamisel, kuna see aitab tagada rakenduse jätkusuutlikust. Kui valitakse populaarne ja ennast tõestanud tehnoloogia, siis leidub suure tõenäosusega piisavalt spetsialiste, kes saaksid vajadusel loodud rakendust ka mitme aasta möödudes edasi arendada.

4.6.1 Tagarakenduse tehnoloogia valik

TIOBE indeksi [47] andmetel on töö kirjutamise ajal viie kõige populaarsema programmeerimiskeele seas kolm tehnoloogiat, mida oleks mõistlik veebirakenduste tegemiseks kasutada: Python, Java ja C#. Ülejäänud kahte keelt, C ja C++, reeglina ei kasutata veebirakenduste arendamiseks.

Python on valikus ainukene dünaamiliselt tüübitud ja interpreteeritud keel [48]. Autor sooviks vältida dünaamiliselt tüübitud keele kasutamist, kuna autori hinnangul on arendaja kogemuse vaatenurgast dünaamiliselt tüübitud keeled pikas perspektiivis ebamugavad. Dünaamiliselt tüübitud keeled on veaohtlikumad võrreldes staatiliselt tüübitud keeltega, kuna kõik programmeerija tehtud vead selguvad alles rakenduse käivitusajal. Samuti on üldjuhul dünaamiliselt tüübitud keeltel vähem tuge programmeerimiskeskkonna poolt, näiteks on raskem koodi refaktoreerimine või ei ole nii tark koodi automaatjätkamine kui staatiliselt tüübitud keeltel [49]. Seega loodavas rakenduses Pythonit ei kasutata.

Nii Java kui ka C# on autori hinnangul mõlemad sobilikud loodava rakenduse tegemiseks, kuna need on staatiliselt tüübitud ja kompileeritud keeled ning seega tagavad hea arenduskogemuse [50]. Lisaks on C# ja Java sarnased ka tööriistade ja raamistike valikul. Mõlemale keelele leidub veebirakenduste arendamiseks korralike populaarseid raamistike, vastavalt C# korral .NET ja Java korral Spring. Kuna üldjoontes on võrreldavad tehnoloogiad omavahel väga sarnased, siis tuleks valik teha detailide põhjal. Loodava rakenduse puhul on oluliseks nõudeks automaatne hinnainfo jälgimine. Seega võiks tehnoloogia valikul eelistada raamistikku, milles on ajastatud tegevused hästi toetatud. Springis on ajastatud tegevused väga hästi toetatud juba raamistiku tasemel. Spring raamistikus on ajastatud tegevuse loomiseks vajalik lisada ainult üks annotatsioon meetodile, mida soovitakse automaatselt käivitada [51]. Seejuures .NET-is ei ole ajastatud tegevused raamistiku tasemel ühe annotatsiooniga toetatud ning arendajal tuleb nende tegemiseks rohkem koodi kirjutada. Näiteks on vaja .NET-is arendajal luua enda taimer, mis oskaks õigel ajal tegevust käivitada [52]. Lisaks selgus analüüsi peatükis 4.3, et rakenduses kasutatakse Mapboxi teenust. Mapboxi integratsiooni on võimalik teha läbi nende avaliku API nii Javas kui ka C#-is. Samas on Mapboxil olemas ka valmis tarkvaraarenduse komplekt Javale, mille kasutamine vähendaks arendaja töö hulka integratsiooni tegemisel [53]. Seega oleks mõistlik tagarakenduse tehnoloogiaks valida Java koos Spring raamistikuga, kuna see aitaks antud töö kontekstis oluliste aspektide arendusele kuluvat aega säästa.

4.6.2 Eesrakenduse tehnoloogia valik

Eesrakenduse tehnoloogia valikul tuleb arvestada nõudega, et rakendus peab olema kättesaadav nii arvutist kui ka nutitelefonist. Seega on antud töö kontekstis mõistlik eesrakenduseks teha veebirakendus, mis on erinevatele seadmetele kohanduv. Seejuures tasuks eesrakendus luua tagarakendusest eraldi ning kasutada andmevahetuseks REST API-t, kuna sellisel juhul oleks eesrakenduse muutmine paindlik ning tulevikus oleks võimalik arendada lihtsasti juurde ka näiteks mobiilirakendus.

Töö kirjutamise hetkel on kolm kõige laialdasemalt kasutatavat tehnoloogiat eesrakenduste loomiseks React, Vue ja Angular [54]. Eelnevas tagarakenduse valiku peatükis 4.6.1 selgitatakse, et autor sooviks kasutada rakenduses staatiliselt tüübitud programmeerimiskeeli. TypeScript, mis lisab muidu dünaamiliselt tüübitud JavaScriptile juurde staatilised tüübid, on vaikimisi programmeerimiskeel Angulari rakendustes [55],

kuid on hästi toetatud ka Reactis ja Vues [56], [57]. Seega täidavad kõik kolm valitud tehnoloogiat selle olulise nõude. Antud lõputöös loodav eesrakendus ei ole tehnoloogiliselt eriti keeruline ning seetõttu ei oma erilisi nõudeid, mida oleks võimalik saavutada ainult kindla tehnoloogiaga. Seega on suures pildis kõik valitud tehnoloogiad sobilikud eesrakenduse loomiseks. Vaadates tehnoloogiaid arendaja kogemuse vaatenurgast, siis on React erinev teistest valikus olevatest keeltest, kuna selles kasutatakse komponentide kirjutamiseks JSX-i, mis võimaldab kasutada HTML sarnast süntaksit JavaScripti/TypeScripti koodis. Seega kui arendaja on tuttav JavaScriptiga, siis JSX-i kasutamisel ei ole tal enamasti vaja uut süntaksit õppida. Vues või Angularis vaikimisi kasutatav mallide loogika nõuab aga üldjuhul arendajalt spetsiaalse süntaksi õppimist. Seepärast oleks mõistlik eesrakenduse tehnoloogiaks valida React.

4.6.3 Andmebaasisüsteemi valik

Andmebaasisüsteemi valikul tuleks esmalt teha otsus relatsiooniliste ja mitterelatsiooniliste andmebaaside vahel. Relatsioonilased andmebaasid hoiavad andmeid struktureeritult kindlate skeemade järgi ning tagavad kõrgel tasemel andmete terviklikkuse. Seejuures relatsiooniliste andmebaaside peamiseks probleemiks on skaleeritavus. Mitterelatsioonilised andmebaasid võimaldavad andmete salvestamist struktureerimata dünaamiliselt muutuvate skeemade alusel [58]. Seejuures on mitterelatsioonilised andmebaasid üldjuhul hästi skaleeruvad. Loodavas rakenduses on plaanitud andmete hoidmine struktureeritud kujul. Seega sobiks rakenduses paremini kasutada relatsioonilisi andmebaase. Autori hinnangul ei ole põhjust muretseda ka rakenduse skaleeruvuse üle, kuna rakendus on mõeldud üksnes Eesti turule.

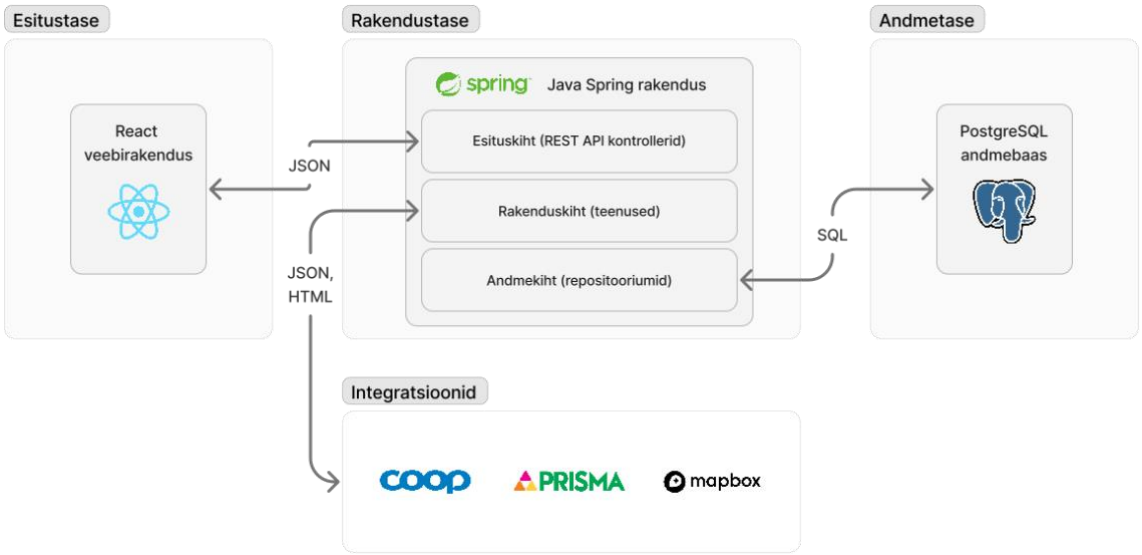
Relatsioonilise andmebaasi valikul on antud rakenduse seisukohast oluline, et tegemist oleks vaba ja tasuta tarkvaraga. Kaks kõige populaarsemat sellist valikut turul on MySQL ja PostgreSQL [59]. MySQL-i puhul on tegemist lihtsama ning traditsiooniliselt ka natukene kiirema andmebaasiga kui PostgreSQL. Samas on PostgreSQL paindlikum ja omab rohkem funktsionaalsusi kui MySQL-i [60]. Autori hinnangul tasuks antud rakenduses kasutada paindlikumat andmebaasisüsteemi, mis võimaldaks vajadusel salvestada ka XML, JSON andmetüüpe ning teha andmete pärimisel keerukaid päringuid vahemaade arvutamiseks. Seega oleks mõistlik andmebaasisüsteemiks valida PostgreSQL.

5 Teostus

Antud peatükis tutvustatakse lõputöö praktilist osa. Praktilises osa arendati peatükis 4.1 määratud nõuetele vastav rakenduse prototüüp, mis valideeriks analüüsi käigus leitud lahenduse kontseptsiooni.

5.1 Rakenduse arhitektuur

Rakenduse loomisel kasutati kolmetasemelist arhitektuuri, kus kasutajaliides, äriloogika ning andmed asuvad üksteisest eraldatult. Ühelt poolt oli kasutajaliidese ja äriloogika eraldamine ning seega kolmetasemelise arhitektuuri kasutamine tingitud analüüsi peatükist 4.6.2, kuid teiselt poolt on tegemist ka väga laialdaselt levinud arhitektuurimustriga veebirakenduste tegemiseks, mis võimaldab tarkvara erinevaid osi üksteisest sõltumatult ja mugavalt arendada. Joonisel 4 on kujutatud loodud rakenduse arhitektuur.



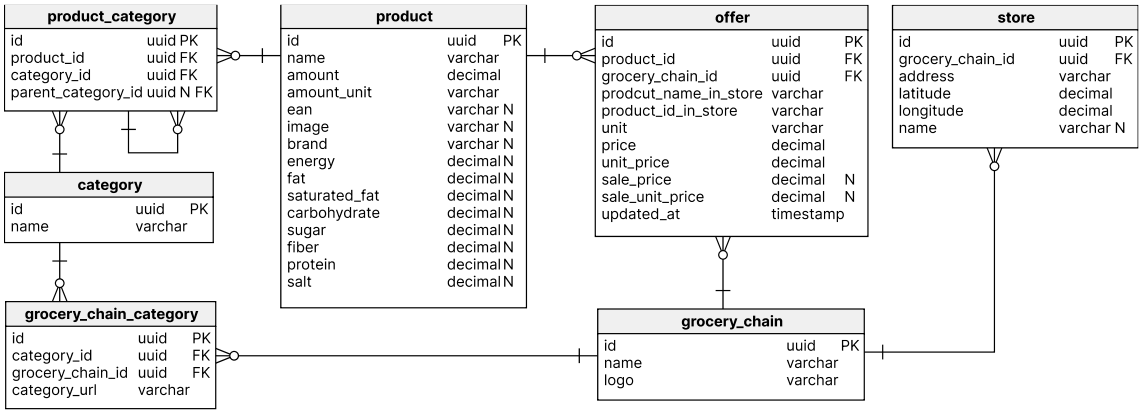
Joonis 4. Rakenduse arhitektuur.

Joonisel on näha rakenduse kolmetasemelist arhitektuuri, mille moodustavad esitus-, rakendus- ja andmetase. Rakenduse esitustasemes on React veebirakendus (edaspidi ka *eesrakendus*), mille eesmärgiks on kasutajaliidese kuvamine tarbijatele. Esitustase

suhtleb andmete saamiseks rakendustasemega kasutades selleks REST API-t ja JSON andmevormingut. Rakendustasemes on Java Spring rakendus (edaspidi ka *tagarakendus*), mis sisaldab äriloogikat, sealhulgas näiteks protsesse hinnavõrdluse ning kombineeritud ostukorvide koostamiseks. Java Spring rakendus on omakorda koostatud kolmekihilist arhitektuuri järgides. Kolmekihiline arhitektuur on oma olemuselt väga sarnane kolmetasemelisele arhitektuurile, kuid antud töö kontekstis on kasutatud IBM-i [61] terminoloogiat, kus kiht (ingl *layer*) viitab rakenduse funktsionaalsele jaotusele, kuid tase (ingl *tier*) on tarkvara funktsionaalne osa, mis töötab teistest osadest eraldi infrastruktuuril. Rakendustasemega on seotud ka tarkvara erinevad integratsioonid väliste organisatsioonide või teenustega, näiteks antud rakenduses hinnainfo kogumine Coop ja Prisma e-poodidest või Mapboxi teenuse kasutamine läbitava teekonna ja ajakulu leidmiseks. Antud rakenduses toimub andmevahetus integratsioonidega nii JSON kui ka HTML vormingus. Rakenduse kõiki andmeid, näiteks hinnainfot, hoitakse andmetasemel, milleks on loodud rakenduses PostgreSQL andmebaasisüsteem. Rakendustaseme ja andmetaseme vaheliseks suhtluseks kasutatakse SQL päringukeelt.

5.2 Andmemudel

Enne ees- ja tagarakenduse arendust projekteeriti loodava süsteemi andmemudel. Autor pidas andmemudeli loomist oluliseks, et tekiks ülevaade rakenduses salvestatavatest andmetest ning seega vältida vigu, mis kanduksid andmekihilt edasi arendatavasse rakendusse. Joonisel 5 on kujutatud loodud rakenduse olemi-suhte diagramm.



Joonis 5. Rakenduse olemi-suhte diagramm.

Rakenduse andmebaasis on kokku 7 tabelit. Rakenduses olevate toidukaupade andmeid hoitakse andmebaasi tabelis *product*, tootekategooriad tabelis *category*, toidukettide

andmed tabelis *grocery_chain* ning toidupoodide asukohad tabelis *store*. Tabelis *offer* salvestatakse toidukaupade hinnad kindlal ajahetkel kindlas toiduketis. Lisaks on andmebaasis kaks seosetabelit *product_category* ja *grocery_chain_category*. Seosetabelis *product_category* hoitakse toidukaupade kuuluvused tootekategooriatesse ning *grocery_chain_category* tabelis kaardistatakse toidukettides olevad tootekategooriad loodava rakenduse kategooriatega.

5.3 Tagarakenduse arendus

Süsteemi tagarakendus arendati vastavalt analüüsi peatükis 4.6.1 valitule Java Spring tehnoloogias. Projektis kasutati Spring Boot tööriista, mis muudab Spring raamistikul põhinevate rakenduste algse seadistamise kiiremaks ja arendajale mugavamaks, kuna vähendab arendaja manuaalse töö hulka. Antud töö kontekstis oli Spring Booti kasutamine mõistlik, kuna aeg rakenduse arenduseks oli väga piiratud. Lisaks kasutati tagarakenduse arendusel Spring Web raamistikku REST API otspunktide loomiseks, Hibernate Validatorit sisendandmete valideerimiseks, Spring Data JPA raamistikku rakenduse olemite ja andmebaasiga suhtluseks ning Flyway tööriista andmebaasi versioonihalduseks. Järgnevalt kirjeldatakse täpsemalt antud töö kontekstis olulisemate tagarakenduse osade arendust.

5.3.1 Ajastatud veebikraapimiste arendus

Vastavalt analüüsi peatükile 4.4 kogutakse rakenduses andmeid veebikraapimise teel. Veebikraapimise tegemiseks on vajalik teha rakendusest HTTP päringuid andmeallikatesse. HTTP päringute tegemiseks oleks saanud kasutada juba kasutusel olevat Spring Web raamistikku, mis sisaldab RestTemplate nimelist klassi. Autor otsustas RestTemplate-i mitte kasutada, kuna see on märgitud aegunuks ning seega ei pruugi antud klassi enam uutes Springi versioonides olla [62]. Springi poolt soovitatud alternatiiviks on kasutada Spring Webfluxi raamistikust klassi nimega WebClient, mis on asünkroonne ja modernsem versioon RestTemplate-ist. Kuigi antud rakenduses ei kasutata asünkroonset protsessivoogu, siis on tegemist tulevikukindlama lahendusega. WebClienti kasutusulevõtul ilmnes probleem rakenduse arenduseks kasutatava masinaga. Nimelt on WebClient sisemuses kasutusel Reactor Netty HTTP klient [63], mis ei töötanud aga operatsioonisüsteem macOS ARM64 versiooniga. Autor leidis probleemile lahenduse GitHubi diskussioonist [64], kus toodi välja, et Reactor Nettyt on võimalik macOS

ARM64 operatsioonisüsteemiga kasutada, kui projektile lisada juurde spetsiaalne sõltuvus. Joonisel 6 on toodud näide antud sõltuvuse lisamisest rakendusse.

```
def isMacOS = System.getProperty('os.name').startsWith('Mac OS X')
def architecture = System.getProperty('os.arch').toLowerCase()
if (isMacOS && architecture == 'aarch64') {
    implementation('io.netty:netty-resolver-dns-native-
macos:4.1.90.Final:osx-aarch_64')
}
```

Joonis 6. Gradle kood Reactor Netty DNS sõltuvuse lisamiseks macOS ARM64 operatsioonisüsteemile.

WebClienti seadistamisel lisati kõigi HTTP päringute päisesse info, mis võimaldaks andmeallikatel tuvastada päringu teinud rakenduse. See on oluline, et tagada veebikraapimise läbipaistvus. Rakenduse identifitseerimiseks kasutati HTTP päist User-Agent ning sinna lisati rakenduse domeen, lühikirjeldus ning kontakt e-mail. Joonisel 7 on toodud näide WebClienti loomisest koos rakendust kirjeldava User-Agentiga.

```
final WebClient client = WebClient
        .builder()
        .defaultHeader("User-Agent", "Tarkostukorv.ee (Collecting
Publicly Available Price Information; ravahe@taltech.ee)")
        .build();
```

Joonis 7. Javas WebClienti loomine koos rakendust kirjeldava päisega.

Rakenduses implementeeriti ka lihtne päringute kiiruse piiraja. Tegemist on eraldi klassiga, mis omab ühte meetodit, mida kutsutakse välja WebClientis enne igat päringut. Antud meetod kontrollib, kas eelneva päringu ajatemplist on möödunud rakenduse konfiguratsioonis määratud aeg. Kui on möödunud vähem aega, siis pannakse selleks ajaks programmi lõim ootele. Rakenduses seadistati päringute kiiruse piirang umbes ühele päringule sekundis. Lisaks lähtutakse andmete kogumisel põhimõttest, et teha võimalikult vähe päringuid. Esiteks teeb see rakenduse tööd kiiremaks ning seejuures koormatakse ka vähem andmeallikaid. Selleks tehakse kõigepealt päring toodete nimekirja vaatesse kategooriate kaupa. Eelistatakse võimalikult suurt nimekirja, kuna siis on võimalik ühe päringuga saada rohkem andmeid. Toodete nimekirja vaatest võetakse välja toodete nimed, hinnad ja võimalusel ka metaandmed, nagu tootekood. HTTP päringu vastuse konverteerimiseks teksti kujult Java klassidesse kasutatakse JSON vastuse korral Jackson Objectmapper teeki ning HTML vastuse korral Jsoup teeki. Seejärel vaadatakse leitud andmete põhjal, kas selline toode juba eksisteerib andmebaasis. Esmalt otsitakse andmebaasist vastet toiduketi pakkumiste seast (*offer* tabel andmebaasis) kasutades selleks toiduketi spetsiifilist identifikaatorit, näiteks tootekood või toote nimi.

Kui vaste leitakse, siis uuendatakse antud pakkumise hindasid. Samas kui vastet ei leitud, siis on vajalik teha enamjaolt päring toote detailinfo saamiseks. Seejärel on võimalik otsida rakenduse üldiste toodete hulgast (*product* tabel andmebaasis) vastavust EAN koodi või toote omaduste alusel. Vaste leidmise vaadatakse, kas leitud tootel on väljasid, mille kohta on andmeid puudu ning võimalusel lisatakse need. Kui vastet ei leitud, siis lisatakse andmebaasi uus toode. Sellisel viisil loodud lahendus andmete kogumiseks vähendab peale esmast sessiooni tunduvalt tehtavate päringute hulka.

Rakenduses kasutatakse veebikraapimise ajastatud käivitamiseks Spring raamistiku *@Scheduled* annotatsiooni. Annotatsioon lisatakse meetodile, mida soovitakse ajakava alusel automaatselt käivitada ning sobiva ajakava määramiseks kasutatakse rakenduses CRON-i laadset avaldist. Veebikraapimise meetodid käivitatakse rakenduses iga päev kell 01.00, et võimalikult vähe segada andmeallikate tööd päevasel ajal, kui veebis on liiklus suurem. Joonisel 8 on toodud näide *@Scheduled* annotatsiooni kasutamisest rakenduses.

```
@Scheduled(cron = "0 0 1 * * *", zone = "Europe/Tallinn")
private void updateCoopPrices() {
    coopIntegrationService.updatePrices();
}
```

Joonis 8. Meetodi ajastatud käivitamine Spring raamistikus.

Vaikimisi on Springis ajastatud tegevuste jaoks kasutusel üks lõim, mis tähendab, et korraga saaks rakendus andmeid koguda ühest andmeallikast. Kuna andmete kogumine võib olla aeganõudev protsess ning tegelikult on erinevatest andmeallikatest andmete kogumine sõltumatu, siis oleks mõistlik andmeid koguda samaaegselt. Spring Booti kasutamine teeb ajastatud tegevuste jaoks lõimede arvu suurendamise väga lihtsaks, kuna vajalik on ainult konfiguratsioonis soovitava lõimede arvu määramine.

Loodud rakenduses kasutatakse andmeallikatena Coop ja Prisma e-poode, mille valik tehti analüüsi peatükis 4.5.2. Töö käigus küsiti luba avalike andmete kogumiseks ka Selverilt, kuid autori saadetud e-mailile ei vastatud ning seega ei olnud töös võimalik Selverilt andmeallikana kasutada.

5.3.2 Teepikkuse ja ajakulu leidmise arendus

Peatükis 4.3 selgus, et rakenduses toidupoodide külastamiseks võimalikult lühikese marsruudi leidmiseks oleks võimalik kasutada kombinatsiooni Haversine valemist ja

Mapboxi TSP-d lahendavast teenusest. Haversine valemit plaaniti kasutada andmebaasi päringus filtrina, et leida ainult kasutaja läheduses asuvad toidupoed. Arenduse ajal selgus, et rakenduse andmebaasisüsteemiks valitud PostgreSQL-il on juba olemas tööriistad kauguste arvutamiseks koordinaatide järgi ning seega ei olnud vajadust Haversine valemit ise andmebaasi funktsioonina implementeerida. Rakenduses eelistati PostgreSQL-i tööriistu, kuna üldjuhul on valmis lahendus töökindlam ja paremini optimeeritud kui arendaja enda kirjutatud lahendus. Seega kasutati andmebaasi päringus filtrina PostgreSQL-i funktsiooni *earth_distance*, mis võimaldab arvutada vahemaad kahe maakera punkti vahel. Koordinaatpunkti konverteerimiseks maakera punktiks kasutati funktsiooni *ll_to_earth*. Nende funktsioonide kasutamiseks oli vajalik andmebaasis sisse lülitada ka *earthdistance* moodul [65]. Joonisel 9 on toodud näide *earth_distance* ja *ll_to_earth* funktsioonide kasutamisest Spring Data JPA andmebaasi päringust, mis leiab kõik poed etteantud asukoha ja kauguse piires.

```
@Query("SELECT s FROM StoreEntity s WHERE
earth_distance(ll_to_earth(s.latitude, s.longitude), ll_to_earth(:lat,
:lon)) <= :maxDistance")
List<StoreEntity> findWithinDistance(@Param("lat") BigDecimal
latitude, @Param("lon") BigDecimal longitude, @Param("maxDistance")
Integer maxDistance);
```

Joonis 9. Spring Data JPA päring poodide leidmiseks etteantud asukoha ja kauguse piires.

Samuti selgus arenduse ajal, et tarbija optimaalset marsruuti ei ole võimalik leida kasutades üksnes Mapboxi TSP teenust. Toidupoodide külastamiseks on võimalikke kombinatsioone lihtsalt liiga palju, et iga kombinatsiooni teepikkuse ja ajakulu leidmiseks teha päring välisesse teenusesse. Seda ka siis kui eelnevalt kasutatakse toidupoodide andmebaasi päringul kauguse filtrit. Seetõttu oli vajalik teha täiendavalt rakenduse sisest filtreerimist, et leida ligikaudse arvutuse teel kõige optimaalsemad kombinatsioonid ning Mapboxi TSP teenusesse teha ainult limiteeritud koguses päringuid täpse teepikkuse ja ajakulu leidmiseks. Rakenduses toidupoodide kombinatsioonide filtreerimiseks oli vajalik leida iga kombinatsiooni lühim ligikaudne teepikkus. Selleks tuli teha rakenduses lokaalne TSP lahendus ja toidupoodide omavahelised kaugused arvutada koordinaatide põhjal Haversine valemit kasutades. Graafide loomiseks rakenduses kasutati Java teeki JGraphT. TSP lahenduseks valiti JGraphT-s implementeeritud Held-Karpi algoritm, kuna antud algoritm leiab alati optimaalse lahenduse. Kuigi Held-Karpi algoritm on eksponentsiaalse keerukusega [66], siis antud rakenduses ei ole see probleemiks, kuna

ühes kombinatsioonis on toidupoodide arv väike. Joonisel 10 on toodud näide ligikaudse teepikkuse arvutamisest rakenduses.

```
public Double findApproxShortestTripBetweenStores(
        Coordinates consumerLocation,
        List<Coordinates> stores
) {
    final var graph = new SimpleWeightedGraph<Coordinates,
DefaultWeightedEdge>(DefaultWeightedEdge.class);

    // Add consumer location and stores to graph
    graph.addVertex(consumerLocation);
    for (Coordinates store : stores) {
        graph.addVertex(store);
    }

    // Find all possible pairs of vertexes inside graph
    List<List<Coordinates>> pairs = Generator
            .combination(graph.vertexSet())
            .simple(2)
            .stream()
            .toList();

    // Calculate distance between each pair
    for (List<Coordinates> pair : pairs) {
        Coordinates a = pair.get(0);
        Coordinates b = pair.get(1);
        DefaultWeightedEdge edge = graph.addEdge(a, b);
        graph.setEdgeWeight(edge, Haversine.calculateDistance(a, b));
    }

    // Find the shortest path that visits all the vertexes
    final var tsp = new HeldKarpTSP<Coordinates,
DefaultWeightedEdge>();
    final var tour = tsp.getTour(graph);
    return tour.getWeight();
}
```

Joonis 10. Java kood ligikaudse vahemaa arvutamiseks rakenduses.

Täpse teepikkuse ja ajakulu arvutamiseks filtreeritakse välja kolm lühima ligikaudse teepikkusega kombinatsiooni ning nendega tehakse päringud Mapboxi TSP teenusesse.

5.4 Eesrakenduse arendus

Süsteemi eesrakendus arendati vastavalt analüüsi peatükis 4.6.2 valitule React tehnoloogias. Eesrakenduse arendamisel kasutati programmeerimiskeelena TypeScripti, mis lisab muidu dünaamiliselt tüübitud JavaScriptile juurde staatilised tüübid. Eesrakenduse arhitektuur on kahekihiline ning koosneb andme- ja esituskihist. Andmekihis olevates repositooriumites tehakse andmete saamiseks päringuid

tagarakendusse, kasutades selleks Axios HTTP klienti. Esituskiht koosneb vaadetest ja komponentidest. Esituskihi vaated määravad rakenduse üldise struktuuri ja koosnevad tavaliselt mitmetest komponentidest. Komponendid on korduvkasutatavad elemendid, mis vastutavad rakenduse teatud funktsionaalsuse eest. Repositooriumist andmete küsimine tehakse üldjuhul samuti komponentide sees. Korduvkasutatavate elementide lihtsamaks kujundamiseks kasutati Styled Components teeki. Rakenduse erinevate vaadete vahel navigeerimiseks kasutati React Router teeki. Eesrakenduses hoitakse ühte globaalset olekut ostukorvis sisu haldamiseks. Lisaks kasutatakse vormide kuvamisel ka komponendipõhiseid olekuid. Rakenduse olekute haldamiseks kasutati Reacti sisseehitatud *useState* funktsionaalsust. Järgnevalt kirjeldatakse täpsemalt antud töö kontekstis olulisemate eesrakenduse osade arendust.

5.4.1 Asukoha määramise arendus

Peatükis 4.3 selgus, et veebirakendustes on tarbija asukoha koordinaatide määramiseks võimalik kasutada Geolocation API-t. Võttes arvesesse kasutatava seadme võimalusi valib veebibrauser ise kõige optimaalsema viisi asukoha määramiseks. Seejuures on arendajal võimalik määrata, kas soovitakse suure täpsusega asukoha määramist või mitte [67]. Antud rakenduses eelistatakse täpset asukoha määramist, kuna asukoha ebatäpsusest tulenevad eksimused võivad otseselt mõjutada ostukorvi tulemust. Rakenduse esmakordsel kasutamisel on vajalik ka kasutaja nõusolek tema asukoha määramiseks. Geolocation API kasutamisel küsib kasutaja nõusolekut veebibrauser automaatselt. Joonisel 11 on toodud näide kasutaja asukoha määramisest kasutades Geolocation API-t.

```
const savePosition = (position: GeolocationPosition) => {
    setUserlocation({
        lat: position.coords.latitude,
        lon: position.coords.longitude
    })
}
const errorHandler = () => {
    setErrorMessage("Unable to locate the user.")
}
const options = {enableHighAccuracy: true}

if (navigator.geolocation) {
    navigator.geolocation.getCurrentPosition(savePosition,
errorHandler, options);
}
```

Joonis 11. Asukoha määramine Geolocation API-ga.

Seejuures peab aga rakenduses arvestama ka olukordadega, kus erinevatel põhjustel ei ole võimalik tarbijate asukohti määrata, näiteks ei toeta asukoha määramist kasutatav seade või ei anna kasutaja selleks nõusolekut. Nende olukordade lahendamiseks loodi rakendusse ka teksti kujul asukoha sisestamise võimalus. Seejuures kasutatakse vigade vältimiseks ning parema kasutusmugavuse tagamiseks Google Places API välist teenust, mis pakub sisestamise hetkel kasutajale soovitusi korrektsetest aadressidest ja asukohanimedest. Lisaks võimaldab Google Places API konverteerida teksti kujul asukohanimed koordinaatideks, mida on võimalik kasutada tagarakenduses. Arenduse ajal kaaluti ka Mapboxi sarnast teenust, kuna siis oleks nii ees- kui tagarakenduses kasutuses sama väline teenusepakkuja. Lõpuks eelistati ikkagi Google teenuse kasuks, kuna sellele oli valmis Reacti komponent, mille kasutamine aitas hoida kokku arendusele kuluvat aega [68]. Mapboxi Reacti komponent asukohtade otsinguks oli töö kirjutamise hetkel veel suletud beetaversioonis [69].

5.4.2 Kasutaja valikute salvestamise arendus

Analüüsi peatükis 4.1 määratud nõuetele vastavalt peab rakendus meelde jätma poolelioleva ostukorvi sisu ning ka tarbija määratud limiidid. Kuna rakenduse prototüübis ei ole kasutajate ega sessioonide identifitseerimist, siis on tarbija valikute salvestamine võimalik ainult eesrakenduses. Eesrakenduses andmete salvestamiseks kasutati Web Storage API-t, mis võimaldab salvestada lihtsaid võti-väärtus paare veebibrauseri mälus. Alternatiiviks oli kasutada HTTP küpsiseid, kuid tänapäeval pole see enam soovitatav viis kliendipoolel andmete salvestamiseks. Esiteks saadetakse küpsistes salvestatud andmed tagarakendusse iga päringu korral, mis ei pruugi olla vajalik. Lisaks on küpsistes salvestatud andmete kättesaamine ebamugavam võrreldes Web Storage API-st andmete kättesaamisega. Web Storage API pakub võimalust salvestada andmeid veebibrauseri *sessionStorage* või *localStorage* mälus. Rakenduses otsustati kasutada *localStorage* mälu, kuna sinna salvestatud andmed püsivad alles ka siis, kui brauser sulgeda ja hiljem taasavada [70]. Poolelioleva ostukorvi salvestamiseks jälgib rakendus ostukorvi sisu muutusi kasutades Reacti meetodit *useEffect*. Iga muutuse korral uuendatakse veebibrauseri mälus olevat kirjet. Veebibrauseri mälus hoitakse ostukorvis olevate toodete identifikaatorid ja kogused, kasutades selleks JSON vormingut. Joonisel 12 on toodud näide ostukorvi sisu salvestamisest veebibrauseri *localStorage* mälus.

```
useEffect(() => {
    const storageCartItems = cartItems.map((item) => {
        return {
            id: item.product.id,
            quantity: item.quantity
        } as IStorageCartItem
    });
    localStorage.setItem('cart', JSON.stringify(storageCartItems))
}, [cartItems])
```

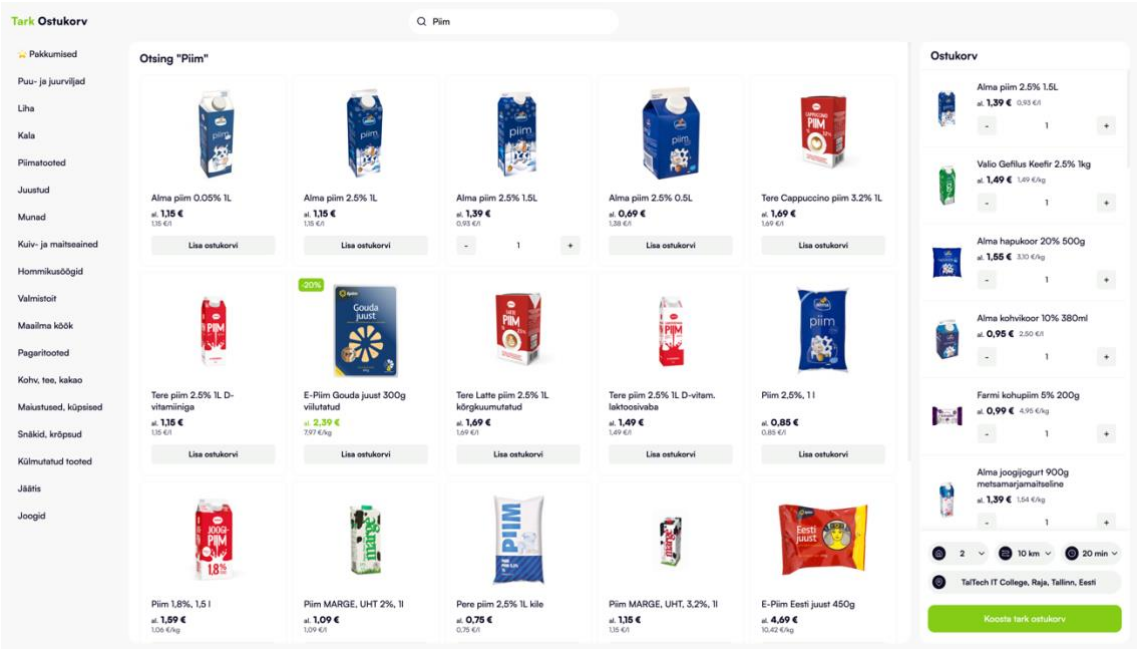
Joonis 12. Ostukorvi seisu salvestamine veebibrauseri mälus.

Veebibrauseri mälus ei salvestata aga toodete hindasid, kuna need võivad ajas muutuda. Rakenduse avades kontrollitakse, kas veebibrauseri mälus leidub kirje ostukorvi toodete kohta. Kui taoline kirje leidub, siis tehakse päring tagarakendusse värske hinnainfo saamiseks ning seejärel on võimalik kasutajale kuvada pooleliolevat ostukorvi. Tarbija valitud limiidid salvestatakse sarnaselt ostukorvi sisule, kuid limiitide korral ei ole vajalik rakenduse avades teha päringuid tagarakendusse, vaid on võimalik kohe salvestatud sätteid kasutajale kuvada.

6 Tulemused

Antud lõputöö tulemusteks on lahenduse analüüs ja selle põhjal arendatud rakenduse prototüüp. Rakenduse prototüübile seatud nõuded said kõik küll täidetud, kuid selleks, et hinnata, kas lõputöö tulemused aitavad ka lahendada algselt sõnastatud probleemi, viidi läbi rakenduse erinevate osade testimine.

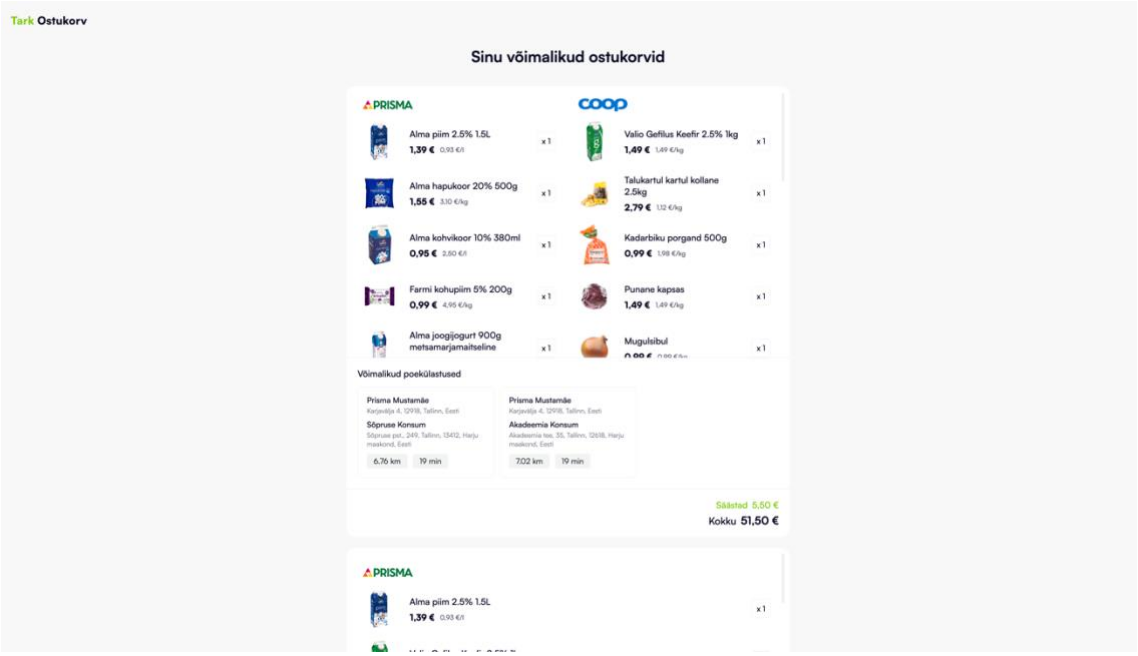
Rakenduse põhifunktsionaalsuse testimiseks tehti katsed ostukorvide koostamiseks ning jälgiti kui palju oleks võimalik loodud lahendust kasutades raha säästa. Esimesel katsel võeti aluseks Postimehe ostukorvi uuringus [8] olevad kaubagrupid. Ostukorvi valiti tooted, mis on olemas rakenduse prototüübi mõlemas andmeallikas ehk Prismas ja Coopis. Joonisel 13 on näha ostukorvi koostamist rakenduse prototüübis.



Joonis 13. Kuvatõmmis ostukorvi koostamisest rakenduses.

Rakenduses määrati ostukorvi koostamisel limiidiks kaks poodi, maksimaalseks läbitavaks vahemaaks 10 kilomeetrit ning maksimaalseks transpordile kuluvaks ajaks 20 minutit. Asukohaks määrati TalTechi IT Kolledži õppehoone. Sellisel otsingul leiti, et kõige mõistlikum oleks toidukaupa osta nii Prismast kui ka Coopist korraga. Rakendus leidis, et määratud limiitide sisse jääks Mustamäe Prisma ja Sõpruse Konsumi või

Mustamäe Prisma ja Akadeemia Konsumi koos külastamine. Taoline kombineeritud ostukorv aitaks säästa tarbijal 5,50 eurot võrreldes kõige kallima ostukorviga ehk kõikide toodete ostmisega Coopist. Samas kui kõik tooted osta Prismast, siis oleks see 2,03 eurot kallim kui kombineeritud ostukorvi hind. Lisas 3 on näha esimese katse ostukorvi sisu koos detailsete tulemustega. Joonisel 14 on näha rakenduse tulemuste lehte, kus kuvatakse rakenduse poolt leitud võimalused esimeses katses olevate toidukaupade soetamiseks.



Joonis 14. Kuvatõmmis ostukorvi tulemuste kuvamisest rakenduse.

Kuna esimesel katsel saavutatud säästmine ei pruugi tunduda piisavalt suur, siis koostati ka teine katse, kus ostukorvi valiti juhuslikult sooduskampaaniaga tooteid. Teisel katsel saadud tulemusteks oli 11,72 euro säästmine võrreldes kõige kallima ostukorviga ja 5,42 euro säästmine võrreldes kaupade ostmisega ühest poest. Lisas 4 on toodud teise katse ostukorvi sisu ja täpsed tulemused. Seejuures tuleb aga arvestada, et rakenduse prototüübis on kasutusel ainult kaks andmeallikat. Samuti sõltub potentsiaalse säästmise summa ka konkreetse ostukorvi sisust, mis on igal tarbijal personaalne.

Lisaks testiti rakenduse funktsionaalsust, mis leiab tarbija võimalikke poekülastusi vastavalt tema määratud limiitidele. Selleks viidi rakenduses läbi 3 erinevat katset. Esmalt otsiti Google Mapsi abil käsitsi välja kõik võimalikud poekülastused, mis jäävad katses määratud asukoha ja limiitide sisse. Seejärel võrreldi käsitsi otsingu teel saadud tulemusi

rakenduse poolt leitud tulemustega ning hinnati, kas rakendus leidis iga kombinatsiooni korral kuni kolm kõige optimaalsemat tulemust. Esimesel kahel katsel, mis viidi läbi Tallinnas ja Tartus, olid rakenduse poolt leitud poekülastused igati korrektsed. Kolmandal katsel, mis tehti Tartu maakonnas, ei leidnud rakendus kahte oodatud tulemust. Antud olukordades olid rakenduse poolt leitud teepikkused suuremad kui katse limiit oleks lubanud. Selle põhjuseks oli asjaolu, et rakenduses kasutatav Mapboxi teenus leiab lühima marsruudi lähtuvalt ajakulust. Seega erines Mapboxi leitud marsruut Google Mapsiga käsitsi leitud marsruudist ning osutus ka juhtumisi määratud limiidist pikemaks. Samas ei ole kolmandal katsel tekkinud probleem üllatav ning antud lõputöös on sellise olukorra tekkimisega aktsepteeritud (vt peatükki 4.3). Tehtud kolme katse täpsed kirjeldused ja tulemused on toodud välja lisas 5.

Viimaks testiti ka rakenduse prototüübi töökiirust kombineeritud ostukorvide koostamise korral reaalses olukorras. Peamine erinevus prototüübi ja reaalse olukorra vahel, mis võib mõjutada rakenduse töökiirust, on võimalike toidupoodide arv. Selleks suurendati prototüübis maksimaalse teepikkuse ja ajakulu limiiti piisavalt, et selle sisse jääks hinnanguliselt sama arv toidupoode, mis on reaalses olukorras Tallinna linnas. Hinnanguliselt on Tallinnas suurusjärgus 100 toidupoodi (Rimi 34 [71], Maxima 30 [72], Selver 30 [73], Prisma 7 [74], Coop 5 [75]), mis kuuluvad andmeallikate valiku peatükis 4.5.2 toodud toidukettidele. Oodatav tulemus peaks jääma alla 10 sekundi, mis on Nielsen Norman Groupi andmetel [76] maksimaalne limiit, et hoida kasutaja tähelepanu. Katsetel mõõdeti aega, mis kulus kombineeritud ostukorvi päringule vastuse saamiseks. Reaalse olukorra katsel saadi viie mõõtmise keskmiseks tulemuseks 1,68 sekundit, mis oli täpselt võrdne prototüübi baasolukorra tulemusega. Seepärast otsustati testida rakendust ka 200 toidupoe ja 200 erineva tootega ostukorvis. Taolisel katsel saadi viie mõõtmise keskmiseks tulemuseks 2,09 sekundit, mis on umbes 24% aeglasem võrreldes eelneva katse ja baasolukorraga. Siit võib järeldada, et rakenduse töökiirust mõjutab rohkem ostukorvis olevate erinevate toodete arv kui võimalikud toidupoed. Siiski jäävad katsete tulemused ilusti oodatud 10 sekundi piiri sisse ning näitavad, et rakendus on piisavalt kiire ka suuremate andmemahtude korral. Lisas 6 on toodud rakenduse töökiiruse testi katsete täpsemad kirjeldused ja tulemused.

Rakenduse testimise tulemused näitavad, et lõputöös loodud lahendus aitab algselt sõnastatud probleemi lahendada, kuna rakendus leiab automaatselt võimalusi toidukaupade ostmisel raha säästmiseks ning arvestab seejuures ka tarbija mugavusega.

Seega said lõputöö eesmärgid täidetud. Samas eksisteerib mitmeid võimalusi loodud lahenduse edasi arendamiseks. Nendest kõige olulisem on lisada rakendusele juurde rohkem andmeallikaid ning seejärel rakendus avalikustada. Samuti on autoril ideid lisafunktsionaalsuste osas, näiteks hinnateavitused, personaalsed ostusoovitused, võimalus valida transpordiks erinevaid liikumisviise ja palju muud. Seejuures tuleks aga loodud lahendust põhjalikumalt analüüsida ka majanduslikust aspektist.

7 Kokkuvõte

Lõputöös käsitletud probleem oli seotud toidukaupade ostmisel raha säästmisega ja seejuures tasakaalu leidmisega odavama ja mitte liiga ebamugava ostukorvi vahel. Probleemi lahendamiseks püstitati lõputöös eesmärk analüüsi koostamiseks lahendusele, mis automatiseeriks toidukaupade hinnavõrdluse, aitaks toidukaupade ostmisel raha säästa ja arvestaks seejuures ka tarbija mugavusega. Lisaks oli eesmärgiks arendada valmis rakenduse prototüüp, mis vastaks analüüsis määratud nõuetele.

Töös tehti esmalt ülevaade probleemi taustast ja olemasolevatest lahendustest. Selle alusel kirjeldati uue lahenduse visioon. Visiooni teostamiseks koostati lahenduse analüüs, kus määrati rakenduse prototüübi nõuded ja käsitleti teemasid kombineeritud ostukorvide koostamisest, teekonna ja ajakulu leidmisest, veebikraapimise meetoditest ja legaalsusest, andmete sidumisest, andmeallikate ja tehnoloogiate valikust. Seejärel kirjeldati töös rakenduse arendust. Teostuse peatükis tehti ülevaade rakenduse arhitektuurist, andmemudelist ning kirjeldati ees- ja tagarakenduse arendust ning arendusel tekkinud probleeme ja nende lahendusi.

Lõputöö tulemusteks on lahenduse analüüs ja selle põhjal arendatud rakenduse prototüüp, mis võimaldab tarbijatel koostada kombineeritud ostukorve ning seejuures määrata piiranguid ka poodide arvule, läbitavale teepikkusele ja transpordiks kuluvale ajale. Lisaks viidi läbi lahenduse põhjalikum testimine. Testimise tulemused kinnitasid, et lõputöö tulemused aitavad lahendada hästi ka algselt sõnastatud probleemi. Seega said lõputöös püstitatud eesmärgid edukalt täidetud.

Kasutatud kirjandus

[1] Statistikaamet, „Tarbijahinnaindeksi suurimaks mõjutajaks oli jaanuaris toit,“ 7. veebruar 2023. [Võrgumaterjal]. Loetud aadressil: https://www.stat.ee/et/uudised/tarbijahinnaindeksi-suurimaks-mojutajaks-oli-jaanuaris-toit. [Kasutatud 7. märts 2023].

[2] Postimees, „OSTUKETI UURING ⟩ Vaid iga kolmas saab endale kõik vajaliku lubada,“ 20. jaanuar 2023. [Võrgumaterjal]. Loetud aadressil: https://kodu.postimees.ee/7695035/ostuketi-uuring-vaid-iga-kolmas-saab-endale-koik-vajaliku-lubada. [Kasutatud 6. märts 2023].

[3] H.-M. Kullaste, P. Lepik ja K. Sildmets, „OSTUKORV | Kuus ketti, 30 toodet. Kalleima ja odavaima ostukorvi vahe on 30 eurot,“ Delfi Ärileht, 16. jaanuar 2023. [Võrgumaterjal]. Loetud aadressil: https://arileht.delfi.ee/artikkel/120128686/ostukorv-kuus-ketti-30-toodet-kalleima-ja-odavaima-ostukorvi-vahe-on-30-eurot. [Kasutatud 7. märts 2023].

[4] Statistikaamet, „Leibkonnad,“ [Võrgumaterjal]. Loetud aadressil: https://www.stat.ee/et/avasta-statistikat/valdkonnad/heaolu/leibkonnad. [Kasutatud 7. märts 2023].

[5] Statistikaamet, „Ostujõu kalkulaator,“ [Võrgumaterjal]. Loetud aadressil: https://www.stat.ee/et/ostujou-kalkulaator. [Kasutatud 7. märts 2023].

[6] Riigi Teataja, „Tarbijakaitseseadus,“ [Võrgumaterjal]. Loetud aadressil: https://www.riigiteataja.ee/akt/124112021004?leiaKehtiv. [Kasutatud 12. märts 2023].

[7] Eesti Konjunktuuriinstituut, „Toidukaupade keskmised jaehinnad Eesti kauplustes,“ [Võrgumaterjal]. Loetud aadressil: https://www.ki.ee/hinnad/kauplused.htm. [Kasutatud 15. märts 2023].

[8] M. Lees, „POSTIMEHE OSTUKORV ⟩ Suur ülevaade: kes on hinnaliider ning kes suutis üllatada,“ 2022 detsember 9. [Võrgumaterjal]. Loetud aadressil: https://tarbija.postimees.ee/7666689/postimehe-ostukorv-suur-ulevaade-kes-on-hinnaliider-ning-kes-suutis-ullatada. [Kasutatud 14. märts 2023].

[9] Eesti Konjunktuuriinstituut, „Eesti elanike toidukaupade ostueelistused ja hoiakud,“ detsember 2020. [Võrgumaterjal]. Loetud aadressil: https://www.pikk.ee/wp-content/uploads/2021/05/Eesti_elanike_toidukaupade_ostueelistused_ja_hoiakud_2020.pdf.

[10] K. Koppel, „Erandlik hinnatõus: aastaga kallinesid kõik toidukaubad,“ 9 detsember 2022. [Võrgumaterjal]. Loetud aadressil: https://www.err.ee/1608813943/erandlik-hinnatous-aastaga-kallinesid-koik-toidukaubad. [Kasutatud 15. märts 2023].

[11] K. Sildmets, „Toidupoed: üldine hinnatõus on küll aeglustumas, kuid see ei kehti toidukaupadele,“ 18. jaanuar 2023. [Võrgumaterjal]. Loetud aadressil:

https://arileht.delfi.ee/artikkel/120129788/toidupoed-uldine-hinnatous-on-kull-aeglustumas-kuid-see-ei-kehti-toidukaupadele. [Kasutatud 15. märts 2023].

[12] Delfi Ärileht, „Kas ostukorvi hinnavõrdlustes võrreldakse võrreldavaid tooteid?," 29 september 2022. [Võrgumaterjal]. Loetud aadressil: https://arileht.delfi.ee/artikkel/120074670/kas-ostukorvi-hinnavordlustes-vorreldakse-vorreldavaid-tooteid. [Kasutatud 15. märts 2023].

[13] Coop, [Võrgumaterjal]. Loetud aadressil: https://www.coop.ee/. [Kasutatud 22. märts 2023].

[14] A. Rogatšova, „Veebirakenduse loomine kauplusketide sooduskampaaniate vahendamiseks," Tartu Ülikool, Narva, 2021.

[15] A. Abdullajev, „Rakendus hindade jälgimiseks toidupoodides," Tallinna Tehnikaülikool, Tallinn, 2022.

[16] M. Lees, „Postimehe e-ostukorv: juust ja või on läinud soodsamaks," Postimees, 1. märts 2023. [Võrgumaterjal]. Loetud aadressil: https://tarbija.postimees.ee/7722813/postimehe-e-ostukorv-juust-ja-voi-on-lainud-soodsamaks. [Kasutatud 21. märts 2023].

[17] Teatmik.ee, „BeboTech OÜ," [Võrgumaterjal]. Loetud aadressil: https://www.teatmik.ee/et/personlegal/16646135-BeboTech-O%C3%9C. [Kasutatud 22. märts 2023].

[18] BeboTech OÜ, „Toidukaupade hinnavõrdlus – Bebo - Sinu Virtuaalne Assistent," [Võrgumaterjal]. Loetud aadressil: https://bebo.ee/. [Kasutatud 22. märts 2023].

[19] B. Maidra, „Toidukaupluste kodulehekülgede kasutamine ja võimalused," Tartu Ülikool, Tartu, 2010.

[20] Eesti Konjunktuuriinstituut, „Elanike toitumisharjumused ja toidukaupade ostueelistused," Tallinn, 2007.

[21] MDN contributors, „Geolocation API," [Võrgumaterjal]. Loetud aadressil: https://developer.mozilla.org/en-US/docs/Web/API/Geolocation_API. [Kasutatud 25. märts 2023].

[22] Apple, „Core Location," [Võrgumaterjal]. Loetud aadressil: https://developer.apple.com/documentation/corelocation. [Kasutatud 25. märts 2023].

[23] Android, „Get the last known location," [Võrgumaterjal]. Loetud aadressil: https://developer.android.com/training/location/retrieve-current. [Kasutatud 25. märts 2023].

[24] GeeksforGeeks, „Travelling Salesman Problem using Dynamic Programming," [Võrgumaterjal]. Loetud aadressil: https://www.geeksforgeeks.org/travelling-salesman-problem-using-dynamic-programming/. [Kasutatud 26. märts 2023].

[25] V. Krõšin, „Asukohapõhiste veebisündmuste algoritmid (Twitteri andmete näitel)," Tartu Ülikool, Tartu, 2014.

[26] Google, „Routes API," [Võrgumaterjal]. Loetud aadressil: https://developers.google.com/maps/documentation/routes. [Kasutatud 26. märts 2023].

[27] Google, „Traveling Salesperson Problem," [Võrgumaterjal]. Loetud aadressil: https://developers.google.com/optimization/routing/tsp. [Kasutatud 26. märts 2023].

[28] Google, „Install OR-Tools,“ [Võrgumaterjal]. Loetud aadressil: https://developers.google.com/optimization/install. [Kasutatud 26. märts 2023].

[29] Google, „Routes Preferred API,“ [Võrgumaterjal]. Loetud aadressil: https://developers.google.com/maps/documentation/routes_preferred. [Kasutatud 26. märts 2023].

[30] Mapbox, „Optimization API v1,“ [Võrgumaterjal]. Loetud aadressil: https://docs.mapbox.com/api/navigation/optimization-v1/. [Kasutatud 26. märts 2023].

[31] C.-R. Puhum, „Kantar Emor: Lidl haukab Eestis turgu, Coop nõrgeneb,“ 3. november 2022. [Võrgumaterjal]. Loetud aadressil: https://majandus.postimees.ee/7640213/kantar-emor-lidl-haukab-eestis-turgu-coop-norgeneb. [Kasutatud 26. märts 2023].

[32] Mapbox, „Mapbox pricing,“ [Võrgumaterjal]. Loetud aadressil: https://www.mapbox.com/pricing/#optimizedtrips. [Kasutatud 26. märts 2023].

[33] Google, „Pricing,“ [Võrgumaterjal]. Loetud aadressil: https://mapsplatform.google.com/pricing/. [Kasutatud 26. märts 2023].

[34] S. Wu, „Web Scraping Basics,“ Towards Data Science, 15. juuli 2020. [Võrgumaterjal]. Loetud aadressil: https://towardsdatascience.com/web-scraping-basics-82f8b5acd45c. [Kasutatud 29. märts 2023].

[35] A. Walters, „CSS Selector vs XPath: Your Pocket Cheat Sheet,“ 14 mai 2022. [Võrgumaterjal]. Loetud aadressil: https://testrigor.com/blog/css-selector-vs-xpath-your-pocket-cheat-sheet/. [Kasutatud 29. märts 2023].

[36] V. Draxl, „Web Scraping Data Extraction from Websites,“ University of Applied Sciences Technikum Wien, Viin, 2018.

[37] EUR-Lex , „Directive 96/9/EC of the European Parliament and of the Council of 11 March 1996 on the legal protection of databases,“ [Võrgumaterjal]. Loetud aadressil: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=celex:31996L0009. [Kasutatud 30. märts 2022].

[38] Your Europe, „Database protection,“ [Võrgumaterjal]. Loetud aadressil: https://europa.eu/youreurope/business/running-business/intellectual-property/database-protection/index_en.htm. [Kasutatud 30. märts 2023].

[39] CURIA, „Case C-762/19,“ [Võrgumaterjal]. Loetud aadressil: https://curia.europa.eu/juris/document/document.jsf?text=&docid=242039&pageIndex=0&doclang=en&mode=req&dir=&occ=first&part=1&cid=3436680. [Kasutatud 31. märts 2023].

[40] E. Kun, „Is the EU's Sui Generis Database a Success or a Hurdle for the Free Flow of Information?,“ 20. veebruar 2022. [Võrgumaterjal]. Loetud aadressil: https://www.linkedin.com/pulse/eus-sui-generis-database-success-hurdle-free-flow-information-kun/?trk=articles_directory. [Kasutatud 31. märts 2023].

[41] Ironclad, „Are Your Terms and Conditions Legally Binding?,“ [Võrgumaterjal]. Loetud aadressil: https://ironcladapp.com/journal/contracts/terms-and-conditions-legally-binding/. [Kasutatud 31. märts 2023].

[42] M. Koster, G. Illyes, H. Zeller ja L. Sassman, „RFC 9309 Robots Exclusion Protocol,“ september 2022. [Võrgumaterjal]. Loetud aadressil: https://www.rfc-editor.org/rfc/rfc9309.html. [Kasutatud 31. märts 2023].

[43] The Web Robot Pages, „Can a /robots.txt be used in a court of law?," [Võrgumaterjal]. Loetud aadressil: https://www.robotstxt.org/faq/legal.html. [Kasutatud 31. märts 2023].

[44] Boardfy, „What is the EAN of a product and why is it important for your e-commerce?," [Võrgumaterjal]. Loetud aadressil: https://www.boardfy.com/what-is-the-ean-of-a-product-and-why-is-it-important-for-your-e-commerce/. [Kasutatud 5. aprill 2023].

[45] Your Europe, „Nutrition declaration," [Võrgumaterjal]. Loetud aadressil: https://europa.eu/youreurope/business/product-requirements/food-labelling/nutrition-declaration/index_en.htm. [Kasutatud 4. aprill 2023].

[46] ERR, „Emor: Lidl on hõivanud seitse protsenti turust, enim on nõrgenenud Coop," 3 november 2022. [Võrgumaterjal]. Loetud aadressil: https://www.err.ee/1608776224/emor-lidl-on-hoivanud-seitse-protsenti-turust-enim-on-norgenenud-coop. [Kasutatud 2. aprill 2023].

[47] TIOBE, „TIOBE Index," [Võrgumaterjal]. Loetud aadressil: https://www.tiobe.com/tiobe-index/. [Kasutatud 4. aprill 2023].

[48] Tutorialspoint, „Python - Overview," [Võrgumaterjal]. Loetud aadressil: https://www.tutorialspoint.com/python/python_overview.htm. [Kasutatud 4. aprill 2023].

[49] E. Boyle, „Static Types vs Dynamic Types. Stop fighting and make my life easier already," [Võrgumaterjal]. Loetud aadressil: https://instil.co/blog/static-vs-dynamic-types/. [Kasutatud 4. aprill 2023].

[50] DoNotApply, „C# vs. Java: which is the best programming language for performance?," 13 september 2022. [Võrgumaterjal]. Loetud aadressil: https://medium.com/@donotapply/c-vs-java-which-is-the-best-programming-language-for-performance-61f9bb23656d. [Kasutatud 5. aprill 2023].

[51] Spring, „Scheduling Tasks," [Võrgumaterjal]. Loetud aadressil: https://spring.io/guides/gs/scheduling-tasks/. [Kasutatud 4. aprill 2023].

[52] S. Poudel, „Setup A Scheduled Repeating Task With .NET Core," [Võrgumaterjal]. Loetud aadressil: https://stacksecrets.com/dot-net-core/scheduled-repeating-task-with-net-core. [Kasutatud 4. aprill 2023].

[53] Mapbox, „Java SDK," [Võrgumaterjal]. Loetud aadressil: https://docs.mapbox.com/android/java/guides/. [Kasutatud 4. aprill 2023].

[54] A. Ivanovs, „The Most Popular Front-end Frameworks in 2023," 25. veebruar 2023. [Võrgumaterjal]. Loetud aadressil: https://stackdiary.com/front-end-frameworks/. [Kasutatud 4. aprill 2023].

[55] Angular, „TypeScript configuration," [Võrgumaterjal]. Loetud aadressil: https://angular.io/guide/typescript-configuration. [Kasutatud 4. aprill 2023].

[56] Create React App, „Adding TypeScript," [Võrgumaterjal]. Loetud aadressil: https://create-react-app.dev/docs/adding-typescript/. [Kasutatud 4. aprill 2023].

[57] Vue.js, „Using Vue with TypeScript," [Võrgumaterjal]. Loetud aadressil: https://vuejs.org/guide/typescript/overview.html. [Kasutatud 4. aprill 2023].

[58] R. Sheldon, „How to choose between SQL and NoSQL databases," 13. aprill 2021. [Võrgumaterjal]. Loetud aadressil: https://www.red-gate.com/simple-talk/databases/nosql/how-to-choose-between-sql-and-nosql-databases/. [Kasutatud 5. aprill 2023].

[59] DB-Engines, „DB-Engines Ranking,“ [Võrgumaterjal]. Loetud aadressil: https://db-engines.com/en/ranking. [Kasutatud 5. aprill 2023].

[60] K. Hristozov, „MySQL vs PostgreSQL -- Choose the Right Database for Your Project,“ Okta Developer, [Võrgumaterjal]. Loetud aadressil: https://developer.okta.com/blog/2019/07/19/mysql-vs-postgres. [Kasutatud 5. aprill 2023].

[61] IBM, „What is three-tier architecture?,“ [Võrgumaterjal]. Loetud aadressil: https://www.ibm.com/topics/three-tier-architecture. [Kasutatud 13. aprill 2023].

[62] Spring Framework, „Class RestTemplate,“ [Võrgumaterjal]. Loetud aadressil: https://docs.spring.io/spring-framework/docs/5.2.1.RELEASE/javadoc-api/index.html?org/springframework/web/client/RestTemplate.html. [Kasutatud 15. aprill 2023].

[63] Spring Framework, „Interface WebClient,“ [Võrgumaterjal]. Loetud aadressil: https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/web/reactive/function/client/WebClient.html. [Kasutatud 15. aprill 2023].

[64] GitHub, „Unable to load io.netty.resolver.dns.macos.MacOSDnsServerAddressStreamProvider,“ [Võrgumaterjal]. Loetud aadressil: https://github.com/netty/netty/issues/11020. [Kasutatud 15. aprill 2023].

[65] PostgreSQL, „F.15. earthdistance,“ [Võrgumaterjal]. Loetud aadressil: https://www.postgresql.org/docs/current/earthdistance.html. [Kasutatud 16. aprill 2023].

[66] JGraphT, „HeldKarpTSP,“ [Võrgumaterjal]. Loetud aadressil: https://jgrapht.org/javadoc/org.jgrapht.core/org/jgrapht/alg/tour/HeldKarpTSP.html. [Kasutatud 17. aprill 2023].

[67] MDN Web Docs, „Using the Geolocation API,“ [Võrgumaterjal]. Loetud aadressil: https://developer.mozilla.org/en-US/docs/Web/API/Geolocation_API/Using_the_Geolocation_API. [Kasutatud 19. aprill 2023].

[68] Tintef, „React Google Places Autocomplete,“ [Võrgumaterjal]. Loetud aadressil: https://github.com/Tintef/react-google-places-autocomplete. [Kasutatud 19. aprill 2023].

[69] Mapbox, „Search box,“ [Võrgumaterjal]. Loetud aadressil: https://docs.mapbox.com/mapbox-search-js/api/react/search/. [Kasutatud 19. aprill 2023].

[70] MDN Web Docs, „Client-side storage,“ [Võrgumaterjal]. Loetud aadressil: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Client-side_web_APIs/Client-side_storage. [Kasutatud 19. aprill 2023].

[71] Rimi, „Kauplused,“ [Võrgumaterjal]. Loetud aadressil: https://www.rimi.ee/kauplused. [Kasutatud 13. mai 2023].

[72] Maxima, „Kauplused,“ [Võrgumaterjal]. Loetud aadressil: https://www.maxima.ee/kaupluseketid. [Kasutatud 13. mai 2023].

[73] Selver, „Kauplused,“ [Võrgumaterjal]. Loetud aadressil: https://www.selver.ee/kauplused. [Kasutatud 13. mai 2023].

[74] Prisma, „Kauplused,“ [Võrgumaterjal]. Loetud aadressil: https://www.prismamarket.ee/store/list. [Kasutatud 13. mai 2023].

[75] Coop, „Kauplused,“ [Võrgumaterjal]. Loetud aadressil: https://www.coop.ee/kauplused. [Kasutatud 13. mai 2023].

[76] J. Nielsen, „Response Times: The 3 Important Limits,“ 1. jaanuar 1993. [Võrgumaterjal]. Loetud aadressil: https://www.nngroup.com/articles/response-times-3-important-limits/. [Kasutatud 14. mai 2023].

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

Mina, Rasmus Vahelaan

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Rakendus targa hinnavõrdluse tegemiseks toidukaupadele“, mille juhendaja on Märt Kalmo
   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 – Ülevaade RFC 9309 standardi kasutusest Eesti e-toidupoodides

E-pood	robots.txt faili sisu	Selgitus
Rimi	rimi.ee/epood/robots.txt User-agent: * Disallow:	Kõigil robotitel on lubatud täielik juurdepääs lehele.
Prisma	prismamarket.ee/robots.txt # See http://www.robotstxt.org/wc/norobots.html for documentation on how to use the robots.txt file # # To ban all spiders from the entire site uncomment the next two lines: User-Agent: * User-agent: AdsBot-Google Sitemap: https://prismamarket.ee/sitemap.xml Sitemap: https://prismamarket.ee/entry/sitemap.xml Sitemap: https://prismamarket.ee/products/sitemap.xml Sitemap: https://prismamarket.ee/recipes/sitemap.xml Disallow: /shoppinglist Disallow: /order Disallow: /orders Disallow: /profile Disallow: /fb Disallow: /intent # to stop a waster of bandwidth User-agent: MJ12bot Disallow: /	Kõigil robotitel on keelatud juurdepääs järgnevatele lehtedele: 4. /shoppinglist; 5. /order; 6. /orders; 7. /profile; 8. /fb; 9. /intent. MJ12bot robotil on keelatud juurdepääs kogu lehele.
Barbora	barbora.ee/robots.txt User-agent: * Disallow: /nauji-mano-duomenys/ Disallow: /mano-prekes/ Disallow: /krepselis Disallow: /pristatymas Disallow: /atsiskaitymas Disallow: /mani-dati/ Disallow: /grozs Disallow: /piegade Disallow: /apmaksa Disallow: /minu-andmed/ Disallow: /ostukorv Disallow: /tarneaknad Disallow: /maksmine Disallow: /moje-towary Disallow: /koszyk Disallow: /dostawa Disallow: /platnosc	Kõigil robotitel on keelatud juurdepääs järgnevatele lehtedele: 1. /nauji-mano-duomenys/; 2. /krepselis; 3. /pristatymas; 4. /atsiskaitymas; 5. /mani-dati/; 6. /grozs; 7. /piegade; 8. /apmaksa; 9. /minu-andmed/; 10. /ostukorv; 11. /tarneaknad; 12. /maksmine; 13. /moje-towary; 14. /koszyk; 15. /dostawa; 16. /platnosc.

Selver	selver.ee/robots.txt User-agent: Googlebot-Image Disallow: User-agent: * Disallow: /error Disallow: /api/ Disallow: /*?dir= Disallow: /*?order= Disallow: /*?product_brand= Disallow: /*?product_segment= Disallow: /*?product_manufacturer= Disallow: /*?product_country_of_origin= Disallow: /*?product_dietary_info= Disallow: /*?product_main_ecategory= Disallow: /*?price= Disallow: /*?limit= Disallow: /*&product_brand= Disallow: /*&product_segment= Disallow: /*&product_manufacturer= Disallow: /*&product_country_of_origin= Disallow: /*&product_dietary_info= Disallow: /*&product_main_ecategory= Disallow: /*&price= Disallow: /*&limit= Disallow: /search?q=* Disallow: /kulleri-valik Disallow: /kulleri-saadavus Disallow: /minu-tooted Disallow: /my-account/ Disallow: /ru/my-account/ Disallow: /b2b-login Disallow: /ru/b2b-login Disallow: /timeslot/ Disallow: /ru/selver/ Disallow: /ru/timeslot/ Sitemap: https://www.selver.ee/sitemap.xml	Googlebot-Image robotil on lubatud täielik juurdepääs lehele. Kõikidel teistel robotitel on keelatud juurdepääs järgnevatele lehtedele: 1. /error 2. /api/ 3. /*?dir= 4. /*?order= 5. /*?product_brand= 6. /*?product_segment= 7. /*?product_manufacturer= 8. /*?product_country_of_origin= 9. /*?product_dietary_info= 10. /*?product_main_ecategory= 11. /*?price= 12. /*?limit= 13. /*&product_brand= 14. /*&product_segment= 15. /*&product_manufacturer= 16. /*&product_country_of_origin= 17. /*&product_dietary_info= 18. /*&product_main_ecategory= 19. /*&price= 20. /*&limit= 21. /search?q=* 22. /kulleri-valik 23. /kulleri-saadavus 24. /minu-tooted 25. /my-account/ 26. /ru/my-account/ 27. /b2b-login 28. /ru/b2b-login 29. /timeslot/ 30. /ru/selver/ 31. /ru/timeslot/

Coop		Kõigil robotitel on lubatud juurdepääs järgnevatele lehtedele: 1. /et/tooted?filters=price_sale 2. /ru/produkty?filters=price_sale Kõigil robotitel on keelatud juurdepääs järgnevatele lehtedele: 1. /*query= 2. /*orderby= 3. /*filters= 4. /et/kulaline 5. /et/registreerimine 6. /et/sisselogimine 7. /ru/gost 8. /ru/registracija 9. /ru/avtorizovatsya

Lisa 3 – Kombineeritud ostukorvi testi 1. katse

Toode	**Prisma**	**Coop**	**Prisma + Coop**
Alma piim 2.5% 1.5L	1,39 €	1,55 €	1,39 €
Valio Gefilus Keefir 2.5% 1kg	1,49 €	1,49 €	1,49 €
Alma hapukoor 20% 500g	1,55 €	1,89 €	1,55 €
Alma kohvikoor 10% 380ml	0,95 €	0,99 €	0,95 €
Farmi kohupiim 5% 200g	0,99 €	1,15 €	0,99 €
Alma joogijogurt 900g metsamarjamaitseline	1,39 €	1,49 €	1,39 €
Valio Eesti juust 350g	3,89 €	4,29 €	3,89 €
Kodukoha Kanamunad L 10tk	2,19 €	2,49 €	2,19 €
Banaan 1kg	1,29 €	1,49 €	1,29 €
Talukartul kartul kollane 2.5kg	2,99 €	2,79 €	2,79 €
Kadarbiku porgand 500g	0,99 €	0,99 €	0,99 €
Punane kapsas 1kg	1,49 €	1,49 €	1,49 €
Kirsstomat punane 250g	1,49 €	1,59 €	1,49 €
Mugulsibul 1kg	1,09 €	0,99 €	0,99 €
Maks&Moorits Seahakkliha 500g	2,79 €	3,49 €	2,79 €
Tallegg Eestimaine broileririnnafilee 400g	4,19 €	3,99 €	3,99 €
Maks&Moorits Seaprae lõigud 400g	2,89 €	2,79 €	2,79 €
Rakvere viiner 500g	2,29 €	2,75 €	2,29 €
Eesti Pagar Rehe rukkileib 390g	1,05 €	0,76 €	0,76 €
Hiiumaa Pagar Hiiu Talusai 480g	1,25 €	1,05 €	1,05 €
Paulig Classic kannukohv 500g	4,49 €	5,99 €	4,49 €
Baltix Tatar 1kg	3,19 €	2,65 €	2,65 €
Tartu Mill Täistera jämedad kaerahelbed 1kg	1,99 €	1,79 €	1,79 €
Tartu Mill Täistera neljaviljahelbed 1kg	1,99 €	1,79 €	1,79 €
Tartu Mill Täistera Penne Rigate pasta 500g	1,15 €	1,39 €	1,15 €
Kalew nisujahu T-550 1kg	1,19 €	1,39 €	1,19 €
Veski Mati Sushi riis 500g	1,89 €	2,49 €	1,89 €
Kokku	**53,53 €**	**57,00 €**	**51,50 €**

Lisa 4 – Kombineeritud ostukorvi testi 2. katse

Toode	**Prisma**	**Coop**	**Prisma + Coop**
MO Saaremaa täispiim 3.8%-4.2% 1L	0,95 €	1,29 €	0,95 €
Rakvere pelmeenid 900g külmutatud	3,29 €	4,49 €	3,29 €
Alma puding šokolaadi-metsapähkli 230g	0,79 €	1,09 €	0,79 €
Alma kodujuust murakamoosiga 200g	1,69 €	1,45 €	1,45 €
Actimel jogurtijook 4*100g maasika	2,29 €	1,49 €	1,49 €
Onu Eskimo Vanilliplombiir 1L	3,99 €	4,79 €	3,99 €
Kalev piimašokolaad Mesikäpp 300g vahvliga	3,89 €	4,99 €	3,89 €
Pure punase greibi mahl 1L	2,89 €	3,29 €	2,89 €
Jacobs Crema kohviuba 1kg	19,49 €	9,99 €	9,99 €
Saaremaa juustuampsud koduaia ürtide 200g	2,29 €	2,79 €	2,29 €
Nõo Hommikupeekon 135g viilutatud	1,89 €	2,35 €	1,89 €
Maks&Moorits Seahakkliha 500g	2,79 €	3,49 €	2,79 €
Kalew nisujahu T-405 2kg	2,69 €	2,09 €	2,09 €
Pealinna Mini-frikadellid 350g külmutatud	1,79 €	2,29 €	1,79 €
Hiiumaa Pagar Hiiu Talusai 480g	1,25 €	1,05 €	1,05 €
Leibur Ruks seemnepala 260g	1,29 €	0,91 €	0,91 €
Kokku	**53,26 €**	**47,84 €**	**41,54 €**

Lisa 5 – Optimaalsete poekülastuste leidmise test

NB! Poekülastuste leidmisel arvestatakse, et läbitakse kõik etteantud asukohad ning jõutakse tagasi alguspunkti (vt peatükki 4.3), näiteks: tarbija algne asukoht – pood – tarbija algne asukoht.

Katse kirjeldus	**Võimalikud tulemused** (Leitud käsitsi Google Mapsi kasutades)	**Rakenduse poolt leitud tulemused** (Iga kombinatsiooni korral limiteeritud 3 parimat tulemust)
Asukoht: Raja 4C, Tallinn (TalTech IT Kolledž) **Määratud limiidid:** kuni 2 poodi, 10 km, 20 min	**Coop:** 1. Akadeemia Konsum (~8 min, 2,5 km) 2. Sõpruse Konsum (~8 min, 3,1 km) **Prisma:** 1. Prisma Mustamäe (~17 min, 6,1 km) **Coop + Prisma:** 1. Sõpruse Konsum + Prisma Mustamäe (~18 min, 6,9 km) 2. Akadeemia Konsum + Prisma Mustamäe (~19 min, 6,7 km)	**Coop:** 1. Akadeemia Konsum (~10 min, 3,4 km) 2. Sõpruse Konsum (~10 min, 3,5 km) **Prisma:** 1. Prisma Mustamäe (~17 min, 6,1 km) **Coop + Prisma:** 1. Sõpruse Konsum + Prisma Mustamäe (~19 min, 6,8 km) 2. Akadeemia Konsum + Prisma Mustamäe (~19 min, 7,0 km)
Asukoht: Ado Vabbe 6, Tartu **Määratud limiidid:** kuni 2 poodi, 15 km, 20 min	**Coop:** 1. Ihaste Konsum (~7 min, 2,4 km) 2. Mõisavahe Konsum (~9 min, 4,9 km) 3. Eedeni Maksimarket (~12 min, 5,6 km) 4. Kivilinna Konsum (~12 min, 7,6 km) 5. Karete Konsum (~16 min, 12,3 km) 6. Kivi Konsum (~19 min, 10,2 km) 7. Kvartali Maksimarket (~20 min, 10,2 km)	**Coop:** 1. Ihaste Konsum (~7 min, 2,3 km) 2. Mõisavahe Konsum (~10 min, 5,0 km) 3. Eedeni Maksimarket (~12 min, 5,5 km)

	Prisma: 1. Prisma Annelinn (~10 min, 6,3 km) 2. Prisma Sõbra (~19 min, 11,6 km) **Coop + Prisma:** 1. Mõisavahe Konsum + Prisma Annelinn (~12 min, 6,0 km) 2. Kivilinna Konsum + Prisma Annelinn (~14 min, 8,0 km) 3. Ihaste Konsum + Prisma Annelinn (~15 min, 8,5 km) 4. Eedeni Maksimarket + Prisma Annelinn (~18 min, 8,5 km) 5. Eedeni Maksimarket + Prisma Sõbra (~20 min, 11,6 km)	**Prisma:** 1. Prisma Annelinn (~10 min, 5,6 km) 2. Prisma Sõbra (~18 min, 13,71 km) **Coop + Prisma:** 1. Mõisavahe Konsum + Prisma Annelinn (~12 min, 5,7 km) 2. Kivilinna Konsum + Prisma Annelinn (~13 min, 7,6 km) 3. Ihaste Konsum + Prisma Annelinn (~15 min, 8,1 km)
Asukoht: Võru mnt 2, Ülenurme, Tartu maakond **Määratud limiidid:** kuni 2 poodi, 15 km, 30 min	**Coop:** 1. Ülenurme Konsum (0 min, 0 km) 2. Tõrvandi Konsum (~5 min, 3,2 km) 3. Karete Konsum (~13 min, 9,0 km) 4. Lõunakeskuse Maksimarket (~19 min, 14,5 km) 5. Kvartali Maksimarket (~25 min, 14,8 km) **Prisma:** 1. Prisma Sõbra (~23 min, 14,7 km) **Coop + Prisma:** 1. Ülenurme Konsum + Prisma Sõbra (~23 min, 14,7 km) 2. Karete Konsum + Prisma Sõbra (~23 min, 14,8 km)	**Coop:** 1. Ülenurme Konsum (~2 min, 0,1 km) 2. Tõrvandi Konsum (~9 min, 3,5 km) 3. Karete Konsum (~17 min, 9,2 km) **Prisma:** – **Coop + Prisma:** 1. Karete Konsum + Prisma Sõbra (~25 min, 14,7 km)

Lisa 6 – Rakenduse töökiiruse test

Katse kirjeldus	1. katse	2. katse	3. katse	4. katse	5. katse	Keskmine
Tooteid ostukorvis: 26 Toidupoode, mis jäävad limiidi sisse: 11	1,70 s	1,66 s	1,73 s	1,71 s	1,64 s	1,68 s
Tooteid ostukorvis: 26 Toidupoode, mis jäävad limiidi sisse: 102	1,73 s	1,60 s	1,71 s	1,57 s	1,78 s	1,68 s
Tooteid ostukorvis: 200 Toidupoode, mis jäävad limiidi sisse: 200	2,22 s	2,07 s	2,15 s	2,02 s	2,02 s	2,09 s