TALTECH
Infotehnoloogia teaduskond

Kevin Koppel 179362IABB

MOBIILIRAKENDUS OSTLEMISEKS ANDROIDI PLATVORMIL

Bakalaureusetöö

Juhendaja: Roger Kerse

Tehnikateaduse magister

Tallinn 2020

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: Kevin Koppel

03.05.2020

Annotatsioon

Antud bakalaureusetöö eesmärgiks on luua Android platvormile mobiilirakendus, mis võimaldab rakenduse lõppkasutajal läbida kaupluses kogu ostlemisprotsess kasutades enda mobiiltelefoni. Mobiilirakenduse kulg jaguneb põhiliselt kolmeks sammuks: kaupluse valimine, ostukorvi koostamine skaneerides tooteid ning maksmine läbi rakenduse.

Mobiilirakendus aitab muuta ostlemisprotsessi lõpptarbija jaoks kiiremaks ja mugavamaks. Kaupluse seisukohalt aitab mobiilirakendus kokku hoida tööjõukuludelt ja/või potentsiaalsetelt iseteeninduskassa soetus- ja halduskuludelt.

Rakenduse põhiliseks arenduskeeleks oli Java, kuid vaadete kujundamisel kasutati ka Androidi versiooni XML-ist. Lisaks rakendusele loodi ka mitmeid MySQL ja JSON formaadis andmebaase ning ka Javas arendatud server, et toetada erinevaid rakenduse funktsioone.

Antud töös antakse ülevaade mobiilirakenduse üldisest arhitektuurist ning analüüsitakse detailsemalt ka rakenduse ülesehitust ja disaini. Lisaks analüüsitakse rakenduse kasutatavust ja kasutegurit ning tuuakse välja lisafunktsionaalsusi ja parandusi, mida annab rakendusega edasiarenduse käigus teha.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 41 leheküljel, 5 peatükki, 14 joonist, 0 tabelit.

Abstract

Shopping mobile application for Android

The purpose of this thesis is to develop a mobile application for Android platform, which allows user to go through the whole process of shopping by using his/hers mobile phone. Application's user flow is divided into three main parts: choosing the store, filling the shopping cart by scanning the products and paying for the products through the application.

Mobile application helps making the shopping process quicker and more convinient for the end user. For the stores, it can help to reduce the labor costs and/or reduce the potential expense of acquiring and maintaining self-checkout machines.

Main development language for application was Java, although Android's version of XML was also used when designing the views. In addition to application, multiple MySQL and JSON format databases were created and also Java-based server to support different functions of application.

The thesis gives an overview of general architecture of mobile applications and also the structure and design of the application is analyzed in detail. Overall usability and efficiency of the application is being analyzed. Also additional functionalities and fixes, that could added to the application, will be provided.

The thesis is in estonian and contains 41 pages of text, 5 chapters, 14 figures, 0 tables

Lühendite ja mõistete sõnastik

Android	Operatsioonisüsteem mobiilsetele seadmetele
Android Studio	Arenduskeskkond Android-i operatsioonisüsteemil olevatele seadmetele rakenduste arendamiseks
CVC	*Card Verification Code,* kaardi kinnituskood
IntelliJ	Java arenduskeelel põhinev arenduskeskkond
JAR	Pakendatud faili formaat, mis hoiab endas Java klasside faile, metaandmeid ja ressursse
Java	Programmeerimisekeel
JSON	*JavaScript Object Notation,* andmevahetusformaat, mis põhineb *JavaScript* programmeerimiskeelel
MySQL	Relatsioonilise andmebaasi haldamise süsteem
PaaS	*Platform as a Service,* Platvorm teenusena
POST päring	Päring, mis saadetakse serverisse ressursi loomiseks või uuendamiseks
QR-kood	*Quick Response*, kahemõõtmeline maatrikskood, mis võimaldab edastada infot skaneerivasse seadmesse
REST API	*Representational State Transfer Application Programming Interface,* Teenus, mis kasutab HTTP päringuid suhtlemiseks
SQL	*Structured Query Language,* Andmebaasi päringu keel relatsiooniliste andmebaaside jaoks
XML	*Extensive Markup Language,* Märgistuskeel, mille eesmärk on struktureeritud info jagamine infosüsteemide vahel
HTTP	*HyperText Transfer Protocol,* protokoll dokumentide vahetamiseks veebis

Sisukord

Jooniste loetelu

Tabelite loetelu

Ei leia illustratsiooniloendi kirjeid.

1. Sissejuhatus

Bakalaureusetöö eesmärk on luua Androidi platvormile mobiilirakendus, mis võimaldab lõppkasutajal teha ostlemisprotsessi kiiremaks ja mugavamaks. Rakenduse abil on kasutajal võimalik skaneerida kõik oma soovitud tooted kasutades enda nutitelefoni, millel on internetiühendus ja töötav kaamera. Lisaks toimub ka maksmine läbi rakenduse, kasutades selleks kasutaja poolt salvestatud maksekaarti, millel on internetiostude võimalus.

Rakenduse kood: https://github.com/kevinkoppel/Fred/tree/newBranch. Github-is *Readme.md* failis on ka link rakenduse alla laadimiseks ning triipkoodid rakenduse testimiseks.

1.1 Taust

Jaekaubanduse sektor on Eestis olnud pidevalt kasvamisstaadiumis. See tähendab, et kaupluste müügimahud on kasvavas trendis ja sellega seoses on kasvamas ka konkurents. Jaekaubanduse ettevõtete arv on aastatel 2014-2018 tõusnud 8,8% ning nende ettevõtete summaarne müügitulu on samas vahemikus tõusnud 17% [1]. Kauplused peavad pidama sammu tehnoloogia arenguga, et püsida konkurentsis. Toidupoodide näol on näha, et üha enam kauplusi integreerivad erinevaid tehnoloogilisi võimalusi, et võimaldada enda klientidele mugavam ja kiirem ostukogemus. Teiseks eesmärgiks tehnoloogiliste võimaluste integreerimisel on kulude kokkuhoid. Põhiliseks muutuseks tänapäeva jaekaubanduses on see, et üha rohkematel kauplustel on installeeritud iseteeninduskassad. [2]

Iseteeninduskassade installeerimine on muutunud väga populaarseks just sellepärast, et need suudavad täita mõlemat ülaltoodud eesmärki – iseteeninduskassad aitavad kauplustel kulusid kokku hoida tööjõukuludelt ning parandavad ka kliendi ostukogemust. Ostukogemuse parandamine seisneb kassajärjekordade vältimises. Iseteeninduskassade integreerimisel kauplustesse väheneb töötajate töömaht ja ettevõttel on võimalik tööjõukuludelt kokku hoida. Globaalselt 120 suurima jaemüügiketi puhul eeldatakse, et kokku suudetakse säästa ligi 1,4 miljardit dollarit tööjõukuludelt. [3]

Iseteeninduskassade kõrval on teiseks ostuprotsessi automatiseerimise meetodiks klientidele ostude skanneerimiseks spetsiaalsete pultide võimaldamine. Selle meetodi puhul toimib maksmine ikkagi iseteeninduskassas, kuid puldid aitavad klientidel ostuprotsessi kiirendada

võimaldades toodete skaneerimise kohe ostusaalis toodet võttes. Lisaks saab klient panna toote kohe enda kotti, tänu millele saab vältida toodete ümbertõstmist korvist/kärust kotti.

1.2 Probleem

Iseteeninduskassade integreerimisega seostuvateks probleemideks on kõrged soetamiskulud ja füüsilise ruumi vajadus. Tüüpilise nelja realise iseteeninduskassa integreerimise hind on 125000 dollarit [4]. Isegi mõningate suurtemate kauplusketide jaoks võivad iseteeninduskassade soetamiskulud olla liiga kõrged ning seetõttu lisatakse iseteeninduskassad vaid teatud kauplustesse [5]. Väiksema pindalaga kaupluste puhul on potentsiaalseks probleemiks ka füüsilise ruumi puudus iseteeninduskassa installeerimiseks.

Iseteeninduskassade kasutamisega seotud probleemidena on välja toodud iseteeninduskassade tehnilist keerukust. Igal kauplusel on iseteeninduskassa erineva tarkvaraga ja selle tõttu on inimestel raske ära harjuda iseteeninduskassade kasutusega. [6]

1.3 Eesmärk

Eesmärgiks on luua iseteeninduskassa iga inimese taskusse. Mobiilirakendus peab olema piisavalt kvaliteetne, et kauplustel oleks võimalik iseteeninduskassade soetamiselt kokku hoida, kaotamata kvaliteedis, mida nad suudavad kliendile pakkuda.

Mobiilirakenduse arendamisel oli eesmärk arendada rakendus, mis on lõppkasutajale kiire ja mugav kasutada. Mobiilirakendus peab olema disainitult selliselt, et kasutaja oskab rakendust juba esmakordsel kasutusel iseseisvalt kasutada ning disain peab olema modernne ja minimalistlik. Lisaks peab rakenduse integreerimine uue kauplusega olema võimalikult kerge ja madala ajakuluga.

1.4 Ülevaade tööst

Mobiilirakendus on arendatud Android platvormi kasutatavatele nutitelefonidele. Arendamiseks kasutati Android Studio arenduskeskkonda ning põhiliseks arenduskeeleks oli Java.

Rakendus võimaldab kasutajal läbida ostlemisprotsess kiirelt ja mugavalt. Kasutaja koostab ostukorvi skaneerides tooted enda nutitelefoni kaamera abil. Ostukorvis olevate toodete eest maksab kasutaja läbi rakenduse kasutades eelnevalt lisatud maksekaarti, millel peab olema internetiostude võimalus.

Rakendusel on autentimissüsteem, mis arendati kasutades Firebase Authentication teenust. Kasutaja ja ostukorviga seotud andmeid hoitakse Firebase poolt pakutavas JSON andmebaasis ning näidistooteid hoitakse MySQL andmebaasis. Maksete teostamiseks kasutatakse Stripe makseplatvormi.

2. Metoodika

2.1 Ülevaade objektist

Mobiilirakendus on disainitud Androidi platvormil toodetele, kus Androidi versiooniks on Android 9 või Android 10. Lisaks on rakenduse kasutamiseks vajalik nõusolek seadme kaamera ja asukoha kasutamiseks. Nõusolekut küsitakse esilehel rakenduse esmakäivitusel.

Rakenduse kasutajalugu jaguneb põhiliselt kolmeks sammuks:

1. Kaupluse valimine – kasutaja valib rakendusega ühendatud kaupluste tabelist kaupluse, kus alustab ostlemist
2. Ostukorvi koostamine – kasutaja skaneerib tooted, mida soovib osta ja lisab need ostukorvi
3. Maksmine – kasutaja maksab ostukorvis olevate toodete eest kasutades eelnevalt sisestatud maksekaarti

Esmakordsel kasutusel tuleb kasutajal lisaks eelnevatele sammudele veel registreerida ennast kasutajaks, logida sisse ning ostlemise alustamiseks on vajalik ka maksekaardi lisamine.

2.2 Ülevaade tööriistadest

Rakenduse arendus toimus Android Studio arenduskeskkonnas, kuid makseteenuse integreerimiseks arendati IntelliJ arenduskeskkonnas makseteenust toetav *back-end* server.

Arendamise põhi arenduskeeleks oli Java, kuid lisaks sellele kasutati rakenduse eriosades ka teisi arenduskeeli. Vaadete disainimisel kasutati Androidi versiooni XML-ist (*Extensible Markup Language)* ning andmebaasipäringutel kasutati SQL-i (*Structured Query Language)*.

Projekti arendusel kasutati versioonihalduseks GitHub-i. See võimaldas salvestada kõik rakenduse versioonid ja vajadusel võtta kasutusele eelnev versioon.

Autentimissüsteemi arendamisel kasutati Firebase authentication teenust. Selle abil sai luua autentimissüsteemi, mis hoiab kasutajaandmed turvaliselt Firebase-i andmebaasides. Firebase-i andmebaasides hoitakse lisaks kasutajaandmetele veel ostukorvis olevaid tooteid, mis on seotud konkreetse ostukorvi omanikuga.

Lisaks Firebase-i andmebaasile kasutatakse veel ka remotemysql.com-i poolt pakutavat MySQL formaadis andmebaasi. See andmebaas on rakenduse testperioodis asendamas kaupluse toodete andmebaasi.

Eeldusel, et kauplused ei ole nõus andma enda andmebaasidele otsest ligipiääsu, kasutati toodete kättesaamiseks andmebaasist Appery.io platvormil arendatud REST API-t. Nii MySQL-i kui ka Firebase-i vaheline suhtlus rakendusega toimis JSON formaadis.

Makseteenuse integreerimisel kasutati Stripe platvormi. Stripe-i abil on kasutajal võimalik maksta läbi rakenduse ning selle abil saab rakendus kontrollida ka kliendi maksekaardi kehtivust ja sisestatud kaardi info korrektsust. Lisaks võimaldab Stripe näha infot maksete kohta, mis on läbi rakenduse sooritatud.

Esmase staatilise prototüübi koostamiseks kasutati proto.io veebitööriista. Proto.io tööriista abil sai lihtsasti luua prototüübi kasutades disainimisel eelnevalt valmis loodud disainielemente. Lisaks võimaldab proto.io saata testijatele rakenduse eelvaate lingi, mille abil testijad näevad enda nutitelefonis prototüübi kasutajavaadet, kus saavad prototüüpi kasutada nagu reaalset rakendust. Prototüüp loodi enne rakenduse arendamise alustamist, et saada tagasisidet rakenduse kohta ja selle põhjal välja selgitada võimalikult hea rakenduse ülesehitus.

2.3 Ülevaade protsessist

Arendusprotsessi esimeseks sammuks oli esmase prototüübi koostamine proto.io abil. Prototüüp oli täiesti staatiline ning selle eesmärgiks oli visualiseerida rakenduse kasutajalugu ning saada selle kohta tagasisidet.

Tagasiside kogunemise toimus kvalitatiivse uurimuse vormis. Küsitletavad said proto.io keskkonnas või mobiilis allalaetuna prototüüpi testida ja andsid seejärel tagasisidet selle kohta. Kvalitatiivne uurimus viidi läbi kahes voorus, kus kahe vooru vahel viidi juba mõned tagasisidest inspireeritud muudatused prototüüpi sisse.

Peale prototüübi viimase versiooni valmimist alustati mobiilirakenduse arendamist. Arendus toimis põhiliselt järgides osaliselt Agiilset Scrumi tarkvaraarenduse metoodikat. [7] Osaline Agiilse Scrumi metoodika järgimine seisnes selles, et arendus jagunes nädalasteks sprintideks ning iga sprindi jaoks olid perioodi alguses ettenähtud funktsioonid kasutajalugude näol, mille pidi valmis arendama.

Arenduse ajal uuendati GitHub-is rakenduse versiooni peale iga uue funktsionaalsuse lisamist. See aitas luua kindlustunde, et kui uue funktsionaalsuse lisamisel peaksid tulema koodi vead, mida ei saa parandada, on võimalik võtta rakenduse eelnev versioon. Lisaks on tagantjärgi lihtsam vaadata iga funktsionaalsusega seotud koodi.

Arendamise ajal oli eesmärgiks järgida puhta koodi printsiipe. [8] See tähendab, et kood on kirjutatud selliselt, et see oleks loetav ka arendajatele, kes loevad antud rakenduse koodi esmakordselt. Kood kordab ennast võimalikult vähe ning kõik klasside, meetodite ja atribuutide nimed vastavad nende eesmärkidele.

3. Tulemused

Mobiilirakendusele sai kokku lisatud 9 kasutajavaadet:

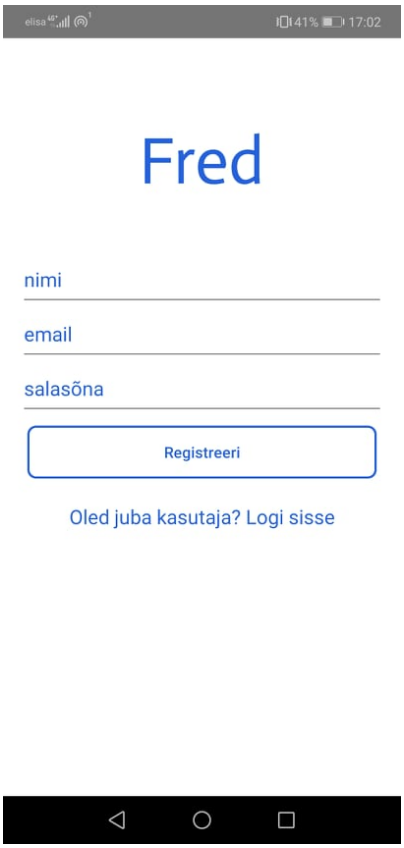
1. Registreerimisvaade
2. Sisselogimise vaade
3. Esileht
4. Profiilivaade
5. Maksekaardi lisamise vaade
6. Kaupluse valimise vaade
7. Toodete skanneerimise vaade (koos ostukorvi fragmenti ja toote kinnituse fragmendiga)
8. Makse kinnitamise vaade
9. Õnnestunud makse vaade

Iga vaade sisaldab dünaamilisi elemente, mille sisu sõltub kasutaja poolt tehtud operatsioonidest.

3.1 Kasutajasüsteem

Kasutajasüsteemi loomisel kasutati protsessi lihtsustamiseks Google poolt arendatud Firebase Authentication teenust. Firebase Authentication teenusesse on sisse ehitatud erinevad validaatorid ja alamteenused, mis võimaldavad turvalise autentimissüsteemi üsnagi lihtsa vaevaga rakendusesse integreerida.

Registreerimisvaates on kasutajalt küsitud tema nime, meiliaadressit ja salasõna. Nimele mingeid piiranguid ei ole seatud, kuid meiliaadressi korrektsust ning salasõna tingimusele vastavust kontrollib Firebase Authentication teenusesse sisse integreeritud validaatorsüsteem.



Joonis 1. Registreerimisvaade

Registreerimisnupule klikates saadetakse registreeritud kasutaja andmed Firebase-i poolt pakutavasse andmebaasi ning kasutajale kuvatakse sisselogimise vaade. Juhul kui kliendil on juba kasutaja registreeritud, võib ta vajutada „Oled juba kasutaja? Logi sisse“ väljale, kust ta suunatakse otse sisselogimise vaatesse.

Sisselogimise vaates küsitakse kasutaja meiliaadressit ning salasõna. Sisselogimise nupule vajutades käivitatakse Firebase Authenticationi sisene teenus, mis kontrollib kas sisestatud meiliaadress ja salasõna vastab mõnele andmebaasis olevale kasutajale.

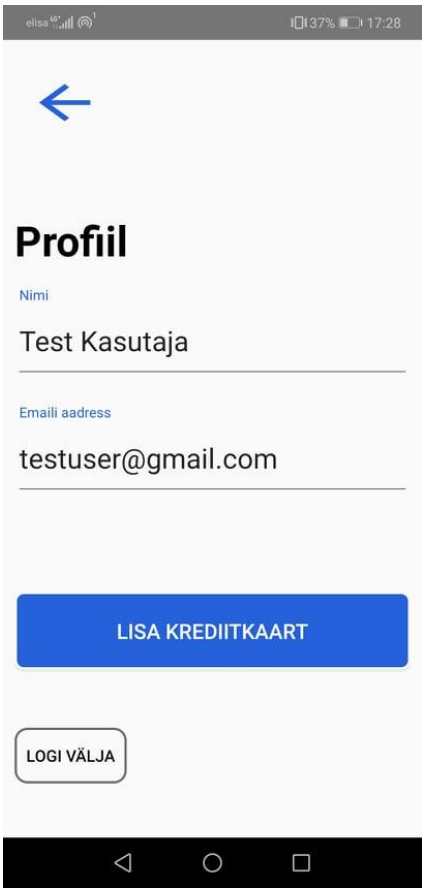
Joonis 2. Sisselogimise vaade.

Juhul kui rakendus kinnitab, et sisestatud meiliaadress ja salasõna vastavad andmebaasis olevatele andmetele, suunab rakendus kasutaja esilehele. Rakenduses ei toimu mingit automaatset välja logimist ehk kui kasutaja on juba ühe korra ära registreerinud, siis järgmisel korral rakendust avades suunatakse ta otse esilehele, jättes vahele registreerimise ja sisselogimise vaated.

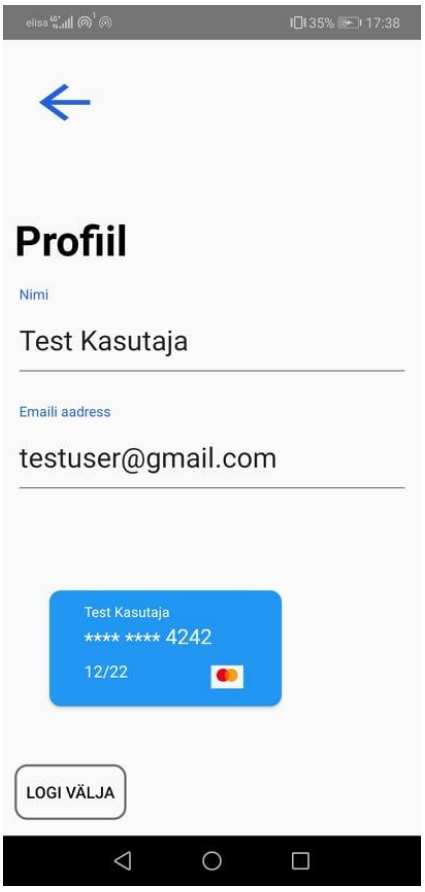
Esilehele kuvatakse kasutajale tervitus koos tema poolt registreerimisel sisestatud nimega. Nime kuvamise jaoks teeb rakendus andmebaasipäringu kasutajaandmete tabelisse koos unikaalse kasutaja identifikaatoriga. Esilehel on kasutajal võimalik edasi minna kas kaupluse valimis vaatesse või profiilivaatesse. Lisaks selle küsitakse esmasel rakenduse kasutamisel kasutajalt nõusolekut seadme kaamera ja asukoha kasutamiseks.

Joonis 3. Esilehe vaade.

Vajutades „Ostle“ nuppu, suunatakse kasutaja edasi kaupluse valimise vaatesse. Esilehe vaate paremal üleval asuvale profiili ikoonile vajutades suunatakse kasutaja profiilivaatesse. Profiilivaates kuvatakse kasutajale tema kasutajaga seotud andmed, milleks on nimi, meiliaadress ning juhul kui maksekaart on lisatud, siis ka maksekaardi info. Juhul kui maksekaarti lisatud ei ole, kuvatakse profiilivaates kasutajale „Lisa krediitkaart“ nupp, mille vajutamisel suunatakse maksekaardi lisamise vaatesse. Lisaks on profiilivaates nupp, millega saab tagasi esilehe vaatesse ning nupp, millega saab kasutajast välja logida (suunatakse sisselogimise vaatesse).



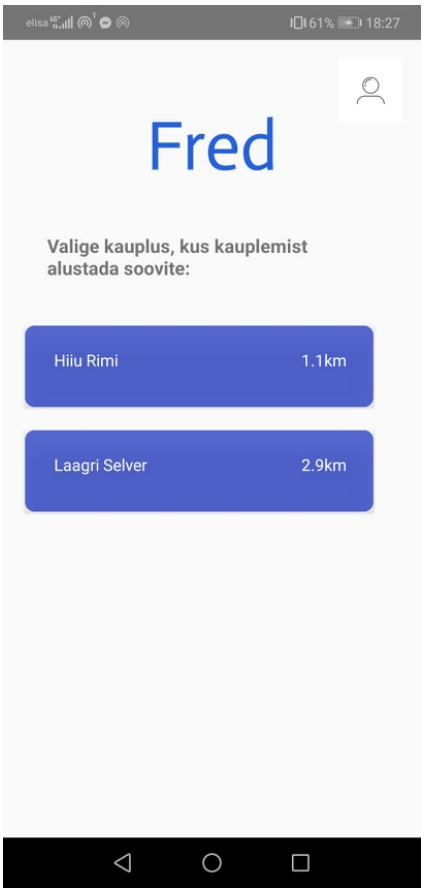
Joonis 4. Profiilivaade ilma maksekaardita..



Joonis 5. Profiilivaade maksekaardiga.

3.2 Kaupluse valimine

Rakendus on mõeldud universaalse rakendusena ehk rakendusega võiksid liituda erinevad kauplused ja kaupluste ketid. Kasutajal on võimalik vaadata kaupluste nimekirjast välja endale lähim kauplus, kus soovib ostelda. Kaupluste nimekiri asub kaupluse valimise vaates, kuhu pääseb ligi esilehelt (joonis 3) „Ostle“ nuppu vajutades.



Joonis 6. Kaupluse valimise vaade.

Kaupluste nimekirjas kuvatakse kasutajale kõik kauplused, mis on rakendusse integreeritud. Kaupluse nimekirjas on näha kaupluse nimi ning kaupluse distants kasutajast.

Vaate loomisel kontrollitakse esmalt kas kasutaja on andnud nõusoleku seadme asukohateenuste kasutamiseks. Juhul, kui nõusolekut antud ei ole, kuvatakse kasutajale nõusoleku andmise aken. Ilma asukohateenuste kasutamise õiguseta kaupluse kuvamise vaadet ei saa näidata.

Kõik rakendusse integreeritud kauplused on lisatud andmebaasi koos kaupluse koordinaatidega. Kaupluse valimise vaadet avades küsib rakendus seadmelt selle hetkeseid asukoha koordinaate. Poe ja seadme asukoha koordinaate kasutades viib rakendus läbi koordinaatarvutuse, mille põhjal saab teada seadme ja poe vahelise distantsi. Distants arvutatakse n.ö linnulennult.

Poe valimisel suunatakse kasutaja edasi toodete skanneerimise vaatesse.

3.3 Ostukorvi koostamine

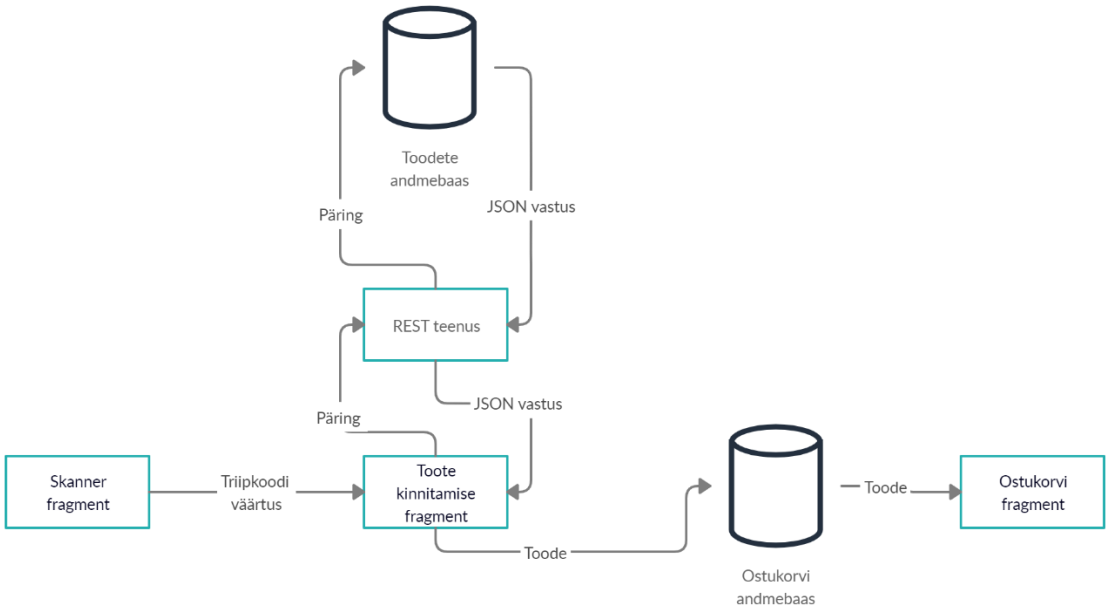
Kogu rakenduse põhifunktsiooniks võib lugeda toodete skaneerimise abil ostukorvi koostamise funktsiooni. See on ka funktsioon, mille arendamisele sai kõige rohkem rõhku pandud. Tähtis oli luua funktsioon, mis võimaldab kasutaja mugavalt ja kiirelt enda soovitud ostukorv kokku panna.

Ostukorvi koostamine toimub toodete skanneerimise vaates. Vaade koosneb kolmest erinevast fragmendist:

1. Skanneri fragment
2. Ostukorvi fragment
3. Toote kinnitamise fragment

Vaade kuvab korraga kahte fragmenti. Skanneri fragmenti kuvatakse koguaeg, kuid toote kinnitamise ja ostukorvi fragmenti kuvatakse kordamööda. Vaate avamisel kuvatakse skanneri ja ostukorvi fragmenti, kuid peale toote skanneerimist asendatakse ostukorvi fragment toote kinnitamise fragmentiga. Kui kasutaja kinnitab toote ära, lisatakse toode ostukorvi ning toote kinnitamise fragment asendatakse jällegi ostukorvi fragmentiga.

Kasutaja jaoks toimib ostukorvi koostamine selliselt, et kasutaja skaneerib enda soovitud toote triipkoodi, mille järel kuvatakse kasutajale toote kinnitamise fragment, kus kasutaja näeb toote nimetust ja hinda ning saab muuta toote kogust. Seejärel saab kasutaja vajutada toote kinnitamise nupule, mille järel kuvatakse kasutajale ostukorv, kus on antud toode olemas.

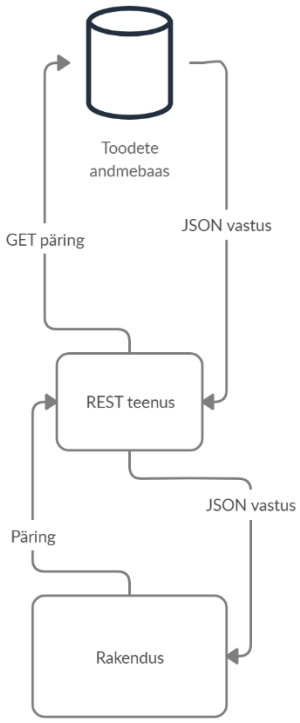


Joonis 7.Ostukorvi koostamine.

3.3.1 Toote lisamine

Toote skanneerimiseks peab kasutaja skaneerima tootepakendil oleva triipkoodi. Rakendusse on integreeritud teenus, mis suudab skaneeritud triipkoodilt välja lugeda triipkoodile vastava numbrilise koodi. Skaneerimiseks peab kasutaja ainult suunama oma kaamera triipkoodi suunas ning skanner tuvastab ise ära kui mõni triipkood on vaateväljas. Toote lisamiseks peab olema toodete andmebaasis toode, millel on skaneeritud tootega kattuv triipkoodi väärtus.

Peale toote skaneerimist saadab rakendus skaneeritud triipkoodile vastava numbrilise koodiga päringu andmebaasi, kust saadakse vastusena tagasi toode, millel on andmebaasi lisatud sama numbriline kood. Rakenduse ja andmebaasi vaheliseks lüliks on koostatud REST teenus, mille abil saadetakse rakendusele soovitav toode JSON formaadis. Kusjuures igale eri kauplusele vastab kindel REST teenus ehk toote skaneerimisel tehakse andmebaasi päring selle kaupluse andmebaasi, mille kasutaja kaupluse valimise vaates valis.



Joonis 8. Rakenduse ja toodete andmebaasi suhtlus.

3.3.2 Toote kinnitamine

Kui REST teenuselt on saadud vastus, kus on JSON formaadis kõik antud toote atribuudid (toote nimetus, hind, triipkoodi väärtus), kuvatakse kasutajale skanneri fragmendi all toote kinnitamise fragment. Toote kinnitamise fragmendis kuvatakse kasutajale toote nimetus ja hind ning lisaks saab kasutaja muuta toote kogust (koguse vaikeväärtuseks on 1).

Joonis 9. Toote kinnitamise fragment.

Toote kinnitamise fragmendis olev hinna väli on dünaamiline. Hind muutub vastavalt sellele, kas kasutaja suurendab või vähendab toote kogust.

Ostukorvi lisamise ikoonile vajutades asendatakse kasutajavaates toote kinnitamise fragment ostukorvi fragmendiga. Ikoonile vajutades käivitub ka protsess, kus kinnitatud toode lisatakse ajutisse ostukorvi andmetabelisse. Ostukorvi andmetabel võimaldab kasutajale kuvada ostukorvi fragmendis kõiki ostukorvis olevaid tooteid ja välja arvutada ka ostukorvi kogusumma, mida saab hiljem kasutada ka makse protsessis. Ostukorvi andmetabel kustutatakse peale makse sooritamist.

3.3.3 Ostukorv

Ostukorvi fragmendi kuvamisel näidatakse kasutajale kõiki tooteid, mis asuvad ostukorvide andmebaasis, konkreetse kasutaja id-ga andmetabelis. Peale toote kinnitamist kuvatakse kasutajale ostukorv, kus on kohe näha ka sama toode, mis just kinnitati.

Ostukorvi fragmendis on võimalik kasutajal konkreetse toote järel oleval „x“ märgile vajutades antud toode ostukorvist eemaldada. Toodet ostukorvist eemaldades, eemaldatakse see kohe ka ostukorvi andmetabelist ning lahutatakse selle toote hind kogu ostukorvi kogusummast. Toote eemaldamisel kaob toode kasutajavaatest koheselt, ning ka uus ostukorvi kogusumma muutub koheselt. Kogusumma arvutamisel on ka arvesse võetud olukorda, kus eemaldatakse toode, mille kogus on suurem kui 1. Sellisel juhul lahutatakse eemaldatava toote hinna ja koguse korrutis.

Joonis 10. Ostukorvi fragment.

Nii ostukorvi kui ka toote kinnitamise fragmendi ajal kuvatakse paralleelselt ka skanneri fragmenti. Skanneri fragmenti ei ole kasutajal võimalik välja lülitada ega peatada ehk skanner on koguaeg valmis skaneerima. Sellega on oht, et võib tekkida olukordi, kus kasutaja skaneerib mõne toote tahtmatult, kuid toote kinnitamise vorm peaks sellises olukorras tahtmatute toodete ostukorvi lisamise vältima.

Kui kasutaja on lisanud oma ostukorvi kõik tooted, mida ta soovib osta, on tal järgmisena võimalus minna maksma. Makse alustamiseks tuleb kasutajal vajutada ostukorvi fragmendis „Maksa“ nupule. Tahtmatute maksete vältimiseks see nupp veel makset kohe ei soorita. „Maksa“ nupule vajutades suunatakse kasutaja makse kinnitus vaatesse.

Joonis 11. Makse kinnitamise vaade.

3.4 Maksesüsteem

Maksesüsteemi integreerimisel tuli integreerida kaks erinevat teenust:

1. Maksekaardi lisamine
2. Makse teostamine

Nende teenuste integreerimiseks kasutati Stripe platvormi. Stripe on platvorm, mis võimaldab maksesüsteemi integreerimiseks erinevaid alamteenuseid ning pakub rakenduse omanikule ka ülevaate lehte, kust näeb erinevaid andmeid platvormi maksetega seoses. [9]

Maksesüsteemi integreerimiseks oli vaja luua ka seda toetav mikroteenus. Tänu sellele ei pea rakendusest läbi käima maksekaartidega seotud andmeid, vähendades andmete lekkega seotud turvariske.

3.4.1 Makse mikroteenus

Mikroteenuse eesmärk on vältida olukorda, kus mobiilirakendusele tehtud rünnakute puhul on ründajal ligipääs rakenduse kasutajate maksekaardi andmetele. Mikroteenuse ja rakenduse vahelise suhtlusel kasutatakse Stripe poolt genereeritud kasutajate ja maksekaartide identifikaatoreid, mis ei sisalda infot maksekaardi kohta.

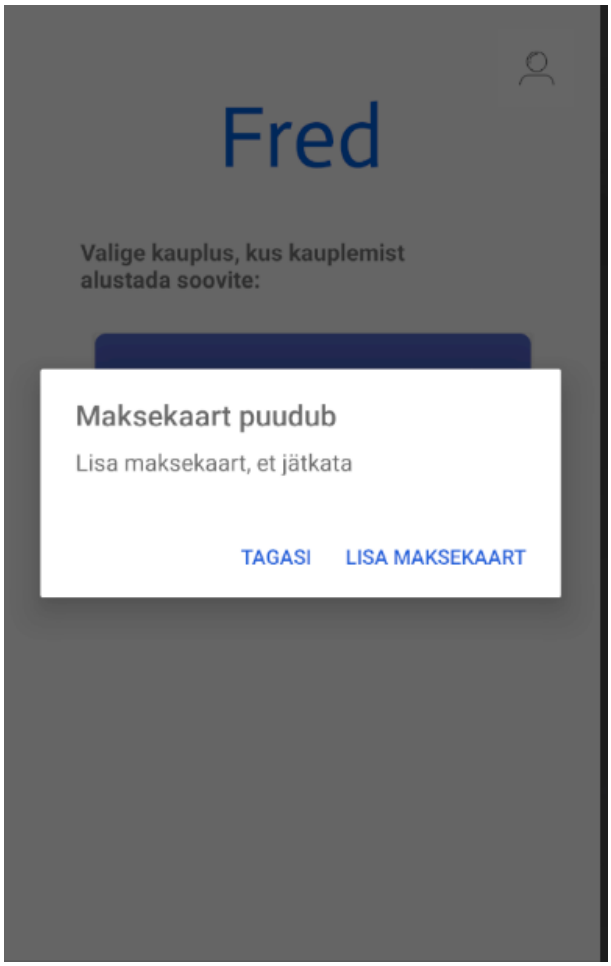
Teenuse loomisel kasutati IntelliJ platvormi. Teenuse arendamiseks kasutati Spark Javat, mis on Java arenduskeelel põhinev raamistik. Spark on raamistik, mis ongi mõeldud mikroteenuste arendamiseks ehk selle abil on võimalik lihtsa vaevaga valmis arendada REST teenus. [10]

Teenuse testimisel käivitati see lokaalses seadmes, kuid selle valmimisel pandi teenus veebimajutusse ülesse, et see oleks globaalselt kasutatav. Veebimajutuseks kasutati Heroku platvormi, mis pakub *PaaS* (*Platform as a Service)* teenust. Enne Heroku-sse üles panemist oli kogu mikroteenus vaja pakendada ühte JAR (*Java Archive)* faili.

Mikroteenus koosneb kahest kahest põhimeetodist ning neid toetavatest abimeetoditest. Üks põhimeetod on mõeldud kaardi lisamise funktsiooni jaoks ning teine põhimeetod on makse teostamise jaoks.

3.4.2 Maksekaardi lisamine

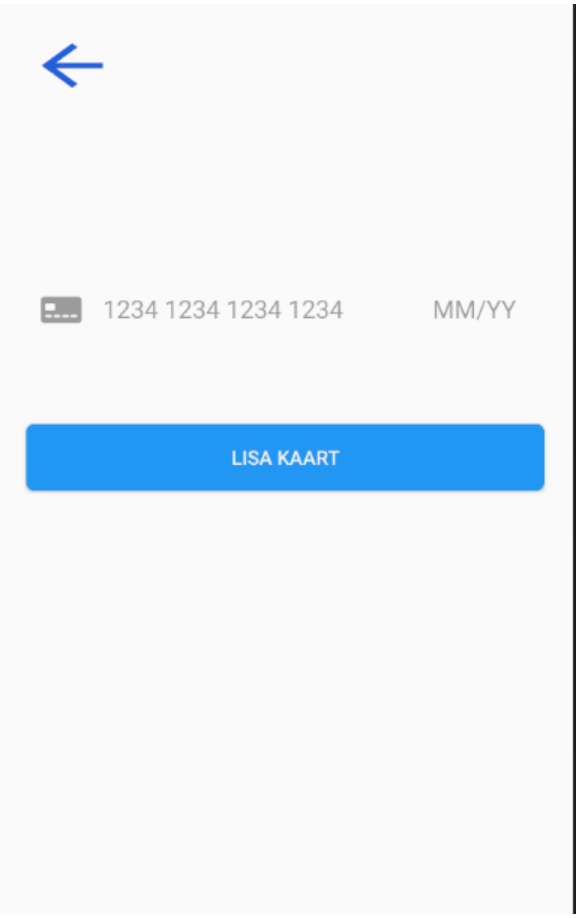
Maksekaart peab olema kasutajal lisatud enne, kui kasutaja alustab ostlemist. Maksekaardi olemasolu valideerimine toimub kaupluse valimise vaates (Joonis 6) kauplusele vajutades. Juhul, kui maksekaarti lisatud ei ole kuvatakse kaupluse valimise vaates kasutajale hüpikaken (Joonis 12) teadaandega maksekaardi puudumise kohta. Hüpikaknas on ka nupp „Lisa maksekaart“, millele vajutades suunatakse kasutaja maksekaardi lisamise vaatesse (Joonis 13). Teine variant maksekaardi lisamise vaatesse jõudmiseks on profiilivaates (Joonis 4) „Lisa maksekaart“ nupule vajutades.



Joonis 12. Hüpikaken maksekaardi puudumise korral.

Maksekaardi lisamise vaates (Joonis 13) on kasutajalt küsitud kaardi number, aegumiskuu, CVC (*Card Verification Code)* kood ja kaardi omaniku postiindeks. Stripe teenuste abil valideeritakse kasutaja poolt sisestatud andmete õigele formaadile vastavus koheselt. Tänu sellele kasutaja näeb kohe kui ta on midagi valesti kirjutanud, mitte ei pea vajutama „Lisa kaart“ nupule selle valideerimiseks.

Kaardi andmete sisestus väli on ülesehitatud dünaamiliselt. Esialgu paistab ainult kaardi numbri ja aegumiskuu väli, kuid kaardi numbri sisestamise liigub väli animatsiooniga paremale ning paljastab ka CVC koodi ja postiindeksi sisestamise väljad.



Joonis 13. Kaardi lisamise vaade.

„Lisa kaart“ nupule vajutades saadab rakendus maksesüsteemi mikroteenusele päringu, mis sisaldab endas sisestatud kaardi andmeid. Päringu kätte saamisel mikroteenus kasutab omakorda Stripe platvormile sisse ehitatud mikroteenuseid, mille abil luuakse maksekaardi instants. Stripe lisab enda andmebaasi maksekaardi objekti ja ka kliendi objekti, millele on külge pandud talle kuuluv maksekaardi identifikaator.

Kui maksekaart ja klient on Stripe andmebaasi lisatud, saadab mikroteenus rakendusele maksekaardi ja kliendi identifikaatorid. Seejärel lisab rakendus antud identifikaatorid kasutaja andmebaasi, selle kasutaja dokumendi alla, kes lisas maksekaardi. Maksekaardi ja kliendi identifikaatorite olemasolu aitab hiljem kasutajal makset sooritada.

3.4.3 Makse sooritamine

Makse sooritamiseks tuleb kasutajal makse kinnitamise vaates (Joonis 11) vajutada nupule „Maksa“. Makse sooritamiseks saadab rakendus makse mikroteenusele POST päringu,

millega antakse parameetritena kaasa maksekaardi identifikaator, makse kliendi identifikaator (Stripe sisene interpretatsioon kasutajast) ja ostukorvi kogusumma. Maksekaardi ja kliendi identifikaatorid saadakse rakenduse kasutaja andmebaasist ning kogusumma arvutatakse liites kokku kõik ostukorvi andmetabelis olevate toodete maksumused.

Makse mikroteenus kasutab Stripe poolt arendatud teenuseid, et koostada saadud parameetrite abil Stripe poolt struktureeritud makse atribuut. Antud makse atribuut edastatakse Stripe-le ning makse õnnestumise puhul saadab mikroteenus rakendusele vastuse, et rakendus saaks makse õnnestumist kinnitada. Vastuse saamisel kuvab rakendus kasutajale vaate (Joonis 14), kus on sõnum makse õnnestumise kohta ning ka QR-kood (*Quick Response*) poest väljumiseks.

Joonis 14. "Makse sooritatud" vaade.

QR-koodi eesmärk on valideerida, et kasutajal on toodete eest makstud enne kui ta poest lahkub. Selline lahendus eeldab, et kauplusel on installeeritud mingit sorti füüsilised väravad, mis avanevad QR-koodi skaneerides. Hetkeseisuga genereerib rakendus juhusliku QR-koodi ning see ei ole seotud ühegi andmebaasiga, mis võimaldaks kauplusel valideerida makse õnnestumist.

Stripe võimaldab enda platvormil saada põhjalik ülevaade maksetest, mis on sooritatud läbi rakenduse. Administraatorkasutajaga sisse logides on võimalik näha kõiki makseid, mis on sooritatud. Lisaks näitab Stripe erinevaid maksetega seotud diagramme, mis aitavad saada parema ülevaate rakenduse edukuseks ning mille abil saab kohandada ärilist strateegiat, et parandada tulemusi.

4. Analüüs

Lõppkasutajale on mobiilirakendus ostlemisprotsessi mugavamaks ja kiiremaks muutmiseks. See tähendab, et rakendus peab olema väga arusaadav, kiire ja veatu, et klient alustaks rakenduse kasutamist ja seejärel ka järgmisel kaupluse külastusel otsustaks uuesti rakendust kasutada.

Mobiilirakenduse arenduse ajal läks kõige suurem rõhk osukorvi koostamise vaate korrektsele toimimisele. Eesmärk oli toote skaneerimise ja seejärel kinnitamise protsess teha võimalikult kiireks ja mugavaks kasutajale, sest need on funktsioonid, mida kasutatakse kõige rohkem.

4.1 Rakenduse üldine kasutatavus

Rakenduse üldise kasutatavusega võib rahule jääda. Kõik funktsioonid sai rakenduses toimima, kuid kindlasti annab neid paremaks ja kiiremaks teha. Kõige suuremaks valukohaks sai toote info kuvamine toote kinnitamise vaates (Joonis 9). Peale toote skaneerimist tekib sisse väike viide enne kui toote info kuvatakse. Esimese toote skaneerimisel tekib sisse kõige suurem viide (ligi 2 sekundit), kuid järgmiste toodete skaneerimisel kuvab toote info juba rahuldava kiirusega.

Teiseks annaks parandada kaupluse valimise vaates (Joonis 6) kaupluste kuvamise kiirust. Hetkel laeb vaate kiirelt ära, kuid kauplused ilmuvad umbes 1-2 sekundit hiljem. See on arvatavasti tingitud sellest, et distantsi arvutamiseks peab rakendus esmalt saama nii rakenduse kui ka poe koordinaadid ja alles siis saab teha koordinaatarvutuse, et kuvada kaupluse distantsi kasutajast.

Muus osas toimib rakendus kiirelt ja on vähemalt testkasutajatele olnud ka lihtsasti aru saadav. Kõik vaated avanevad kiirelt ja dünaamilised väljad täituvad koheselt, kui välja arvata kaks varasemalt välja toodud erandit.

4.2 Kasutajavaadete disain

Kasutajavaadete disainimisel oli põhieesmärgiks luua minimalistlik disain, kus ei ole mingeid üleliigseid disainielemente ega värvilisi taustasi jms. Tänapäevaste rakenduste puhul võib panna minimalistlikkuse ja modernsuse vahele võrdusmärgi ja selle pärast valiti ka selline lähenemine disaini puhul.

Rakenduse põhivärvideks on valge ja sinine. Sinine värv sai suurel osal valitud täna psühholoogilistele uuringutele, mis toovad välja eri värvide psühholoogilised efektid kasutajatele. Sinise värvi puhul on välja toodud, et sinine värv süstib kasutajatesse rohkem usaldusväärsust toote vastu. [11]

Kuna disain ei olnud rakenduse arendamisel põhiprioriteediks, jäid mõningates vaadetes sisse n.ö disainilised vead. Põhiliseks veaks jäi erinevat tooni sinise värvi kasutamine. Lisaks jäi mobiilirakenduse logo eri vaadetes erinevatele kõrgustele.

Üks rakenduse osa, mille disaini võib kritiseerida on toote kinnitamise fragment (Joonis 9). Ajapuuduse tõttu ei jõudnud seda osa ilusamaks disainida.

4.3 Funktsioonide analüüs

Järgnevalt analüüsin erinevate rakenduse juurde kuuluvate funktsioonide erinevaid osasid nende toimimist.

4.3.1 Registreerimine ja sisselogimine

Registreerimine ja sisselogimine on mõlemad funktsioonid, mis kuuluvad autentimissüsteemi alla. Google pakub autentimissüsteemi integreerimiseks väga head teenust, nimega Firebase Authentication. Firebase Authenticationi abil oli registreerimise ja sisselogimise funktsiooni ning selle juurde kuuluvate osade integreerimine üsnagi kiire protsess. Integreerimisel oli suureks abiks ka põhjalik dokumentatsioon, mis selle teenuse juurde kuulus.

Firebase Authenticationisse on sisse ehitatud erinevad validaatorid, mis kontrollivad kas sisestatud meiliaadress vastab meiliaadressi formaadile. Lisaks pakub Firebase Authentication ka andmebaasi tuge ehk kõik kasutaja andmed on võimalik salvestada otse Firebase

andmebaasi, mis on JSON formaadis. Registreerimisel lisab rakendus Firebase andmebaasi kasutaja identifikaatoriga dokumendi, mis sisaldab endas järgnevaid atribuute:

1. Nimi
2. Meiliaadress
3. Maksekaardi identifikaator
4. Makse kliendi identifikaator

Maksekaardi ja makse kliendi identifikaatorite väljad jäävad esialgu tühjaks, kuid maksekaardi lisamisel täidetakse need ära.

Kokkuvõttes sai korralikult toimiva rakendusele korralikult toimiva autentimissüsteemi, mis on ka turvaline. Rakendus laseb registreerida ainult siis, kui lisad õiges formaadis andmed ehk ei saa tekkida olukordi, kus tekib vigu mittestandardsete kasutajaandmete tõttu. Juurde võiks lisada kõrgemad standardid salasõna lisamisel, et suurendada rakenduse turvalisust.

4.3.2 Kaupluse valimine ja kaupluse integreerimine

Kaupluse valimise vaates sai kõige suuremaks takistuseks kaupluse distantsi arvutamine seadmest. Korrektse distantsi leidmiseks sai testitud erinevaid koordinaatarvutus valemeid ja võrreldud nende tulemusi veebis pakutavate koordinaatkalkulaatori tulemustega. Parimaks tulemuseks saadi 8%-lise eksimusega distantsi tulemus. See tähendab, et see on kindlasti üks funktsioon, mida peab edasiarenduse faasis korrigeerima.
Kaupluste integreerimisel rakendusse eeldati, et kauplused ei ole nõus andma otseühendusi oma andmebaasidele. Kaupluse integreerimisel eeldatakse, et kauplus võimaldab kasutada REST API-t, et teha andmebaasipäringuid toodete kätte saamiseks. Näidis andmebaasiks kasutati MySQL formaadis andmebaasi ning API koostati appery.io platvormil.

4.3.3 Ostukorvi koostamine

Kasutajale mugava ja kiire ostlemisprotsessi võimaldamiseks on ostukorvi koostamise funktsioon kõige suurema mõjuga. Ostukorvi koostamisel peab toimima kõik korrektselt ja kiirelt.

Skaneerimisel on tähtis, et triipkoodi saab skaneerida erinevate nurkade alt ning, et skaneerimine toimub kiirelt. Skaneerimise funktsiooni sai rakenduses tööle rahuldavas korras. Testimisperioodil kasutati Huawei P20 Lite seadet, mis on odavama klassi telefon, kuid ka

sellega toimis skaneerimine kiirelt. Skaneerimisel tekkis probleeme, kui skaneerida pimedas keskkonnas.

Ostukorvi koostamise puhul on ainukeseks mitterahuldavaks osaks toote info laadimise kiirus. Skaneerimis tulemuse saamisel tekib sisse viide, mille jooksul rakendus saab toodete andmebaasist kätte toote info. Kõige suurem probleem on esimese toote skaneerimisel, kus toote info laadimine võtab aega umbes 2 sekundit. Järgmiste toodete skaneerimisel kuvab rakendus toote info umbes 0,5 sekundiga. Probleemi põhjustajaks on appery.io poolt pakutav API teenus, mille päringute vastuse saamisele kulub enamus viiteajast.

Ostukorvi osa töötab rakenduses korrektselt. Kõik kinnitatud tooted kuvatakse ostukorvis ning ostukorvi kogumaksumuse arvutustulemused on korrektsed.

4.3.4 Maksesüsteem

Maksesüsteemi integratsioon toimus läbi Stripe poolt pakutavate teenuste. Maksesüsteem koosneb kahest osast: maksekaardi lisamine ja makse sooritamine. Hetkel on maksesüsteem testperioodis ehk lisada on võimalik ainult Stripe poolt ette antud näidiskaarte ja makse sooritamisel ei krediteerita ega debiteerita ühtegi reaalset kontot.

Maksesüsteemi mõlemad osad sai korrektselt toimima. Kaardi lisamisel on näha, et maksekaart lisatakse Stripe andmebaasi ning rakenduse kasutaja andmetabelisse lisatakse lisatud maksekaardi identifikaatorid. Makse sooritamisel on näha, et Stripe platvormil kuvatavad maksedetailid vastavad reaalsusele.

4.4 Rakenduse kasutegurid

4.4.1 Kasutegurid liitunud kauplustele

Rakenduse kasutegurid kauplusele on tööjõukulude kokkuhoid, tööjõu puuduse leevendamine ja parema teeninduskvaliteedi pakkumine kliendile.

Rakenduse sihtgrupp kaupluste näol jaguneb kaheks:

1. Kauplused, millel on iseteeninduskassad installeeritud
2. Kauplused, millel ei ole iseteeninduskassasid installeeritud

Esimese sihtgrupi kaupluste puhul on sihtrgrupi realiseerimine raske ülesanne. Kauplustel on juba installeeritud iseteeninduskassad, mille kasutegurid kattuvad rakenduse omadega. Selle sihtgrupini jõudmiseks tuleks teha rakenduses edasiarendusi, mis suudavad suurendada rakenduse kasutegurit võrreldes iseteeninduskassade omadega. Teiseks võimaluseks selle sihtgrupini jõudmiseks on läbi n.ö lumepalliefekti. Kui rakendusega on liitunud juba mitmed teise sihtrühma kauplused ning saavutatud piisav kasutajabaas, on kasulik ka esimese sihtrühma kauplustel liituda, kuna suurema kasutajabaasiga on kasutegurite jõud suurenenud.

Teise sihtgrupi puhul on rakenduse kasutegur koheselt märgatavalt suurem. Selle sihtgrupi kauplustel on võimalik astuda samm lähemale klienditeeninduse automatiseerimisele ilma tegemata investeeringuid iseteeninduskassade soetamise näol. Kuigi sellesse sihtrühma kuuluvate kaupluste käibed ja külastatavuse arvud jäävad alla esimese sihtrühma omadele, võimaldavad selle sihtrühma kauplused suurendada kasutajabaasi, et saada müügiargument esimese sihtrühma kauplustele liitumiseks.

4.4.2 Kasutegurid lõppkasutajale

Lõppkasutajale pakub rakendus ostlemisprotsessi läbimiseks kiiret ja mugavat lahendust. Kasutajad saavad tooted skanneerida juba ostusaalis toodet võttes ning soovi korral tooted asetada kohe kotti. Kui kõik tooted on skaneeritud saab kasutaja mugavalt maksta läbi rakenduse ning seejärel poest lahkuda.

Ostlemisprotsessi kiiruse ja mugavuse tõus on eriti märgatav kauplustes, kus puudub alternatiivvõimalus iseteeninduskassade kasutamise näol. Iseteeninduskassadega võrreldes on kasutegur väiksem, kuid siiski on see olemas. Rakendus võimaldab vältida protsessi, kus pead ostukorvist kõik tooted välja tõstma, skaneerima ja seejärel kotti pakkima. Puldisüsteemi puhul aitab rakendus kokku hoida aega maksmise protsessilt, selle asemel, et tagastada pult ja seejärel asuda iseteeninduskassasse maksma, on võimalik maksta kohe läbi rakenduse.

4.5 Rakenduse edasiarenduse võimalused

Rakenduse edasiarenduse puhul esimesteks sammudeks peaks olema rakenduse nõrkkohtade likvideerimine. Rakenduse nõrkkohtadeks on:

1. toote skaneerimise järel tekkiv viiteaeg enne kui toote info kuvatakse
2. Kaupluse distantsi arvutamisel tekkiv eksimus

3. Disainilised vead

Edasiarendusel on võimalik lisada erinevaid lisafunktsioone, mis muudavad rakenduse kaupluse ja lõppkasutaja jaoks atraktiivsemaks. Lisafunktsioonid, mis muudaksid kaupluste jaoks rakenduse atraktiivsemaks:

1. Kasutaja käitumise analüüs – näiteks võimaldada näha soojuskaarti kasutajate asetsemisest poes.
2. Turunduskanali loomine – võimaldada kauplustel kasutata rakendust turunduskanalina (näiteks kuvada esilehel teatud kaupluste sooduspakkumisi)

Lisafunktsioonid, mis muudaksid lõppkasutaja jaoks rakenduse atraktiivsemaks:

1. Ostunimekirja koostamise võimalus – enne kauplusesse minekut on võimalik koostada ostunimekiri
2. Kaupluse hindade võrdlus – Rakendus võrdleb ostunimekirja põhjal ostukorvi summa iga kaupluse kohta ning kuvab neid kasutajale
3. Poe sisese liikumise trajektoori optimisatsioon, mis põhineb ostulistil – rakendus kuvab ostu nimekirjas olevate toodete põhjal trajektoori kaupluses liikumiseks
4. Kokkuvõte kuludest – võimalus näha teatud perioodi kulutusi

4.6 Rakenduse äriline realisatsioon

Rakendust saab äriliselt realiseerida mitmeil viisil kasutades erinevaid hinnastamissüsteeme. Hinnastamise variandid:

1. Vahendustasu
2. Tellimuspõhine ehk *subscription based*
3. Ühekordne liitumistasu

Töö autori poolt on eelistatud vahendustasu varianti ehk igalt ostukorvilt saab rakenduse omanik teatud protsendi vahendustasuna. Sellise süsteemi puhul muutub rakendus ka kauplusele atraktiivsemaks, kuna kauplus maksab selles mahus, kui palju on rakendus talle kasu toonud. Arvestama peab ka sellega, et iga makse korral läheb Stripe-le vahendustasuna 1,4% kogusummast + 0,25 eurot.

Tellimuspõhise variandi puhul maksab kauplus rakenduse omanikul teatud perioodi eest eelnevalt kindlaks määratud summat. Sellise variandi puhul on nõrkuseks see, et kui

rakendusel ei ole algusperioodil suurt kasutajabaasi, võib kauplus lepingu tühistada, kuna ei soovita tühja maksta.

Ühekordne liitumistasu puhul kaob ära klienditeeninduse automatiseerimise investeeringute vältimise kasutegur. Rakenduse käivitamisfaasis on sellise hinnastamissüsteemi puhul keeruline leida kaupluseid, mis oleksid nõus rakendusega liituma.

5. Kokkuvõte

Tehnoloogiaga sammu pidamine mängib kaupluste puhul suurt rolli konkurentsis püsimiseks. Tehnoloogia abil on kauplustel võimalik hoida kokku kulusid ning võimaldada klientidele paremat ostlemiskogemust. Hetkel seisneb jaekaubanduses tehnoloogiaga sammu pidamine põhiliselt erinevate klienditeenindust automatiseerivate lahenduste integreerimise näol. Populaarseimateks lahendusteks on iseteeninduskassade ning puldisüsteemide installeerimine.

Antud töö eesmärgiks oli luua mobiilirakendus, mis võimaldab klienditeenindust automatiseerida madalamate investeeringutega. Mobiilirakendus peab võimaldama kasutajal läbida kogu ostlemisprotsess kasutamata kassasid või iseteeninduskassasid. Rakenduse arenduslikuks eesmärgiks oli luua rakendus, mis on lõppkasutajale kiire, mugav ja lihtne. Kaupluste jaoks pidi rakendusega liitumise protsess olema võimalikult väikse ajalise ja rahalise kuluga.

Töö käigus kirjeldas autor, millised on antud rakendusega seotud kasutegurid nii rakendusega liitunud kauplustele kui ka kasutajatele. Kaupluste puhul on esmaselt põhiliseks sihtrühmaks kauplused, millel ei ole veel installeeritud iseteeninduskassasid ega puldisüsteemi. Kaupluste, kus on iseteeninduskassad juba olemas, liitumiseks rakendusega oleks tugevaks müügiargumendiks eelnevalt kogutud kasutajabaas, mida saab eelnevalt saavutada läbi kaupluste, kus puuduvad iseteeninduskassad.

Rakendusel said valmis kõik funktsionaalsused, mis on vajalikud rakenduse üldiseks toimimiseks. Enne toote väljalaskmist tuleks parandada osade funktsionaalsuste toimimist ning kasutajaliidese disaini. Võimalusel võiks enne toote väljalaskmist olla ka lisatud punktis 4.6 välja toodud edasiarendused, et lihtsustada rakenduse ärilist realisatsiooni.

Kasutatud kirjandus

[1] Statistikaamet, „https://www.stat.ee/58225“.

[2] M. Pärli, „Jaeketid rajavad populaarsust koguvaid selvekassasid üha juurde,“ *https://www.err.ee/694181/jaeketid-rajavad-populaarsust-koguvaid-selvekassasid-uha-juurde.*

[3] P. D. N. N. S. P. J. Z. P. S. Bhaskar Nallapureddy, „Future of Self Checkout,“ Berkeley, University Of California.

[4] *http://web.mit.edu/2.744/www/Project/Assignments/humanUse/lynette/2-About%20the%20machine.html.*

[5] K. Ruus, „https://arileht.delfi.ee/news/uudised/enamus-poekette-investeerib-iseteeninduskassadesse-ka-valjaspool-tallinnat-vaid-prisma-peab-seda-liiga-kulukaks-sammuks?id=84790051,“ 2018.

[6] A. Rooväli, „"Tarbijate motiivid iseteeninduslike kassade eelistamiseks supermarketites",“ https://digikogu.taltech.ee/et/Item/0f70c6af-bd6d-404a-957f-601235cf7157, 2017.

[7] „Agile software development,“ *https://en.wikipedia.org/wiki/Agile_software_development.*

[8] T. D. Moor, „A few principles of clean code,“ *https://x-team.com/blog/principles-clean-code/,* 2019.

[9] https://stripe.com/.

[10] „Spark (software),“ *https://en.wikipedia.org/wiki/Spark_(software).*

[11] A. J. Elliot, „Color and psychological functioning: a review of theoretical and empirical work,“ *https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4383146/,* 2015.

[12] M. Tänava, „Iseteeninduskassadele ülemineku võimalused,“ 2011.