TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Hendrik Argus 142688IAPB

TAASKASUTAMIST MÄNGUSTAVA MOBIILIRAKENDUSE ARENDUS

Bakalaureusetöö

Juhendaja: Martin Verrev
MSc

Tallinn 2021

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: Hendrik Argus

15.05.2021

Annotatsioon

Bakalaureusetöö raames on loodud Androidi mobiilirakendus, mis lubab vahetada esemeid mängulises võtmes. Rakendus on suunatud inimestele, kes soovivad vahetada vinüülplaate.

Rakendus lubab kasutajal vinüülplaate talletada ning asuda nendega otsima potentsiaalseid vahetusi teiste kasutajate vinüülplaatidega, lükates ekraanil kuvatavaid vinüülplaate ekraani paremasse või vasakusse äärde. Rakendus tuvastab, kui on tekkinud kahepoolne soov vahetada vinüülplaate, luues rakendusesisese vestlusakna, kus osapooled saavad arutada, kas ja kuidas seda vahetust võiks realiseerida.

Bakalaureusetöö esitamise hetkel on rakendus saadaval Google Play poes nime alt „Vahetaja“.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 48 leheküljel, 5 peatükki, 17 joonist, 0 tabelit.

Abstract

"Development of a mobile application that aims to gamify re-use"

An Android application, which enables the trading of items in a gamified manner, was developed, in the course of this Bachelor's thesis. Currently, the application is targeted towards people who wish to exchange vinyl records.

The application allows the user to upload his/her vinyl records and look for potential trades with other users' records. Finding potential trade matches occurs via swiping (swipe a picture of a vinyl record to the right or left part of the screen). The application detects when a mutual trading request exists between two users, at which point a chat window between the users will open inside the application. The chat window gives the users the opportunity to discuss if and how they wish to proceed with the exchange of vinyl records.

During the submission of the Bachelor's thesis, the application is available for download in the Google Play store and it is called "Vahetaja".

Lühendite ja mõistete sõnastik

Android	Peamiselt nutitelefonidele mõeldud operatsioonisüsteem.
Tagarakendus	Veebiteenus, mis serveerib andmeid kasutajaliidesele.
JavaScript	Objektorienteeritud programmeerimiskeel, mida kasutatakse peamiselt veebilehtede skriptimisel.
Java	Objektorienteeritud programmeerimiskeel.
React-Native	Kasutajaliidese arendamiseks loodud raamistik, mis toetab JavaScripti.
Spring Boot	Tagarakenduse arendamiseks loodud raamistik, mis toetab Javat.
REST	Tarkvaraarhitektuuri laad, mis määrab veebirakendusele kindlad reeglid.
SQL	Andmebaasi relatsiooniline päringukeel.
AWS	Amazon Web Service'i teenuse lühend.
MariaDB	Relatsiooniline andmebaasi haldamise süsteem.
IDE	Integreeritud programmeerimiskeskkond.
VSCode	Integreeritud programmeerimiskeskkond, mis toetab näiteks nii Java kui ka JavaScripti koodi.
APK	Rakenduse tüüp, mida on võimalik käivitada Androidi seadmel.
Debug	Termin, mida kasutatakse arendamisel tekkinud vigade jälitamiseks.
Node.js	JavaScripti käitamiskeskkond.
Gradle	Rakenduste ehitamise automatiseerimiseks loodud vahend.
JDK	Java Development Kit – tarkvarakomplekt, mis võimaldab kompileerida ja jooksutada Java rakendusi.
AVD manager	Android Studio sisene teek, mis toetab seadistatud riistvaraprofiile, millega on võimalik jooksutada virtuaalseid Android seadmeid.
JSON	Andmevahetusvorming, mis põhineb JavaScriptil.
JWT	JSON kujul veebitõend.
HTTP	Protokoll andmete edastamiseks arvutivõrkudes.
GET ja POST päring	HTTP päringumeetodid.

IOS	Mobiiliseadmete operatsioonisüsteem.
WebSocket	Täistupleks andmevahetusprotokoll.

Sisukord

Jooniste loetelu

1. Sissejuhatus

Viimaste aastate üheks suurimaks ja kiiremini kasvamaks trendiks on taaskasutus. Üha levinum on prügi sorteerimine, *second-hand*-poodidest kauba ostmine, uuskasutuskeskusesse esemete viimine või internetikeskkondades kasutatud kaupade ostmine ja müümine – need kõik on näited üha kasvavast taaskasutusest. Kohusetundliku inimesena on selle trendi järgimine ka minu jaoks väga oluline, sest taaskasutamine on jätkusuutliku majandamise üks alustaladest.

Lähtuvalt eelnevast otsustasin siinse bakalaureusetöö raames arendada välja Androidi rakenduse „Vahetaja“, mille eesmärk on kaasa aidata taaskasutuse levikule. Täpsemalt soovisin luua platvormi, mis võimaldab vahetada teiste rakenduse kasutajatega esemeid ja teha seda mängulises võtmes.

Kuigi rakendus „Vahetaja“ võiks teoreetiliselt sobida igasuguste esemete vahetamiseks, valisin esialgu testimiseks väga konkreetse sihtgrupi, kellele vahetamisekeskkonnast palju kasu võiks olla. Selleks osutusid vinüülplaatide kogujad ja huvilised ning rakendus ongi esialgu mõeldud vinüülplaatide vahetamiseks.

Projekti esimene samm oli kasutajaliidese arendus, kasutades staatilisi näidisandmeid. Seejärel oli fookuses tagarakenduse arendus. Esmalt töötasin välja andmebaasiga suhtlemiseks, autentimiseks ning päringute vastu võtmiseks vajaliku loogika. Kui kasutajaliidese ja tagarakenduse esmased versioonid olid valmis, algas testimine ning rakenduse pidev parandamine ning uuendamine.

Töös olid nii kasutajaliidese kui ka tagarakenduse arendamine olulised ülesanded, kuid põhiliseks fookuseks sai kasutajaliidese arendus. Selle põhjuseks on rakenduse ülesehitus ning sihtgrupp – kasutajate kogemus rakendust kasutades on põhiline faktor, mis taolise funktsiooniga rakendusi üksteisest eristab ning mis enamasti otsustab, see, kas kasutajad on valmis platvormil aega veetma ning seda korduvalt kasutama.

Siinses töös esitan rakenduse arendusega seotud taustainfo, s.h rakenduse kirjelduse, sihtgrupi, eesmärgid ja konkurendid. Samuti pakun detailse ülevaate arendusprotsessi eri etappidest ning rakenduse täpse arhitektuuri kirjelduse. Lisaks teen ka kokkuvõtte rakenduse testimise meetoditest ja tulemustest ning pakun välja, millised võiksid olla järgmise arendusetapi eesmärgid.

2. Taust

Töö eesmärk on luua mobiilirakendus „Vahetaja“, mis võimaldab vahetada teiste rakenduse kasutajatega esemeid mängulises võtmes. Rakenduses on kasutajal võimalik lisada ning vahetada esemeid. Ese on rakenduses kirjeldatud pildi ning lühikese tekstilise kirjeldusega. Vahetamine toimub nii, et peale enda eseme lisamist ning sellega vahetamisele asudes kuvatakse ekraanile korraga üks vastu pakutav ese ning sellega on võimalik kas nõustuda või mitte. Juhul kui esemest keeldutakse, kuvatakse ekraanile järgmine vastu pakutav ese. Kui aga nõustutakse ning ka teine osapool on varem nõustunud, tekitatakse teise osapoolega rakendusesisene vestlus, mis võimaldab arutada edasi, kas ja kuidas vahetus realiseerida. Rakenduse esialgu loodud versioon ei paku vahetamise hõlbustamiseks näiteks kullerteenust — see, kuidas vahetust realiseerida, on osapoolte endi otsustada. Vahetuse saab kasutaja rakendusesiseselt lõpetada, andes tagasisidet vahetuspartnerile või kustutades vahetusse pakutud eseme. Kasutajate tagasiside mõjutab iga konkreetse kasutaja esemete prioriteetsust edaspidistel vahetuste otsimisel.

2.1. Sihtgrupp

Rakenduse esialgseks vahetatavate esemete kategooriaks olen valinud vinüülplaadid. Sellise valiku põhjuseks oli põhimõte, et kui rakendus võimaldaks vahetada ükskõik mis sorti esemeid, tekiks hulgaliselt probleeme. Näiteks tekiks suur hulk esemeid, mis üksteisest erinevad nii hinna, otstarbe, suuruse ja teiste sarnaste esemeid iseloomustavate parameetrite poolest. See tekitaks olukorra, kus suur osa potentsiaalseid vahetusi oleks kasutajale sobimatud ning ihaldatava eseme leidmiseks kuluks suurem hulk aega.

Vinüülplaatide suurused jagunevad peamiselt kolmeks: *LP* ehk albumi diameeter on 300 mm, *EP*-plaadi diameeter on 250 mm ning singelplaadi diameeter 175 mm. Kaalu poolest jäävad vinüülplaadid enamasti 180 kuni 220 grammi vahele [1]. Vinüülplaadid on oma

suuruselt, kaalult ning otstarbelt üksteisega lihtsalt võrreldavad ning nende vahetamine on aktuaalne, seda on näha näiteks nii Facebooki gruppides kui ka laatadel.

2.2. Rakenduse otstarve ning eesmärgid

Loodava rakenduse eesmärk on soodustada taaskasutust — pakkuda võimalust vabaneda soovimatutest asjadest ning saada vastu midagi sellist, mida soovitakse. Rakendus aitab vähendada soovimatute asjade minemaviskamist ja selle kaudu kasutaja mõju keskkonnale.

Eeltöö käigus analüüsisin alternatiivseid lahendusi. Kasutajate arvu põhjal toon esile kolm suuremat, mis oma funktsionaalsuselt iseloomustavad ka teisi sarnaseid platvorme:

- SmartSwap [2] on esemete vahetuskeskkond, mis töötab punktipõhises süsteemis — iga eseme eest, mis antakse või saadetakse teisele kasutajale, saadakse üks punkt. Punkte kasutatakse esemete tellimiseks teistelt SmartSwapi kasutajatelt. Esemetele ei saa määrata rahalist väärtust. Kullerteenuse kasutamise korral katab saaja kohaletoimetamise kulud. Fikseeritud sihtgrupp puudub, keskkond proovib katta kõiki võimalikke kategooriaid.

- Hula Trading [3] — mobiilirakendus esemete vahetamiseks, võimalus kombineerida vahetatavaid esemeid, näiteks pakkuda mitut eset ühe eseme vastu. Fikseeritud sihtgrupp puudub, toetab kõikide võimalike esemete vahetust.

- Freecycle [4] — veebikeskkond, mis võimaldab leida asukoha põhiselt esemeid, millest soovitakse loobuda. Geograafilised alamjaotused sõnalisel otsingul.

Järgnevalt toon välja loodava rakenduse eelised eelnevalt esitatud sarnaste teenustega võrreldes.

- Ükski kirjeldatud kolmest rakendusest ei ole mänguline, erinevalt loodavast rakendusest, mille omamoodi lähenemine esemete vahetamiseks on kasutaja jaoks värske ning paeluv. Mängulisust lisab rakendusele vahetuseks mõeldud esemete valimine *swipe*'ides, nagu populaarses tutvumiskeskkonnas Tinder [5].

- Lihtne ja lakooniline kasutajaliides ning mängulises võtmes esemete ühekaupa võrdlemine ning vahetamine ei nõua kasutajalt palju aega ja süvenemist, ehk et rakenduse kasutajal on suurem võimalus jääda rakendust kasutama peale esimest kasutuskorda [6].

- Paljud sarnased teenused (eelnevalt esitatutest näiteks Freecycle ning SmartSwap on veebipõhised ning neil puudub mobiilirakendus. Mobiilirakenduste kasutus võrreldes veebipõhiste teenustega on tõusutrendis [7].

Loodava rakenduse puudused eelnevalt välja toodud sarnaste teenustega võrreldes.

- Loodav rakendus ei võimalda isikliku esemevaramu nii täpset haldamist ja kategoriseerimist võrreldes konkureerivate teenustega. Samas on selle arvelt, juhul kui ära antavaid esemeid pole palju, rakenduse kasutamine oluliselt mugavam kui alternatiivsete lahenduste puhul.

- Ei ole võimalik eraldi otsida esemeid näiteks nime järgi, mida soovid millegi vastu vahetada.

3. Nõuded rakendusele

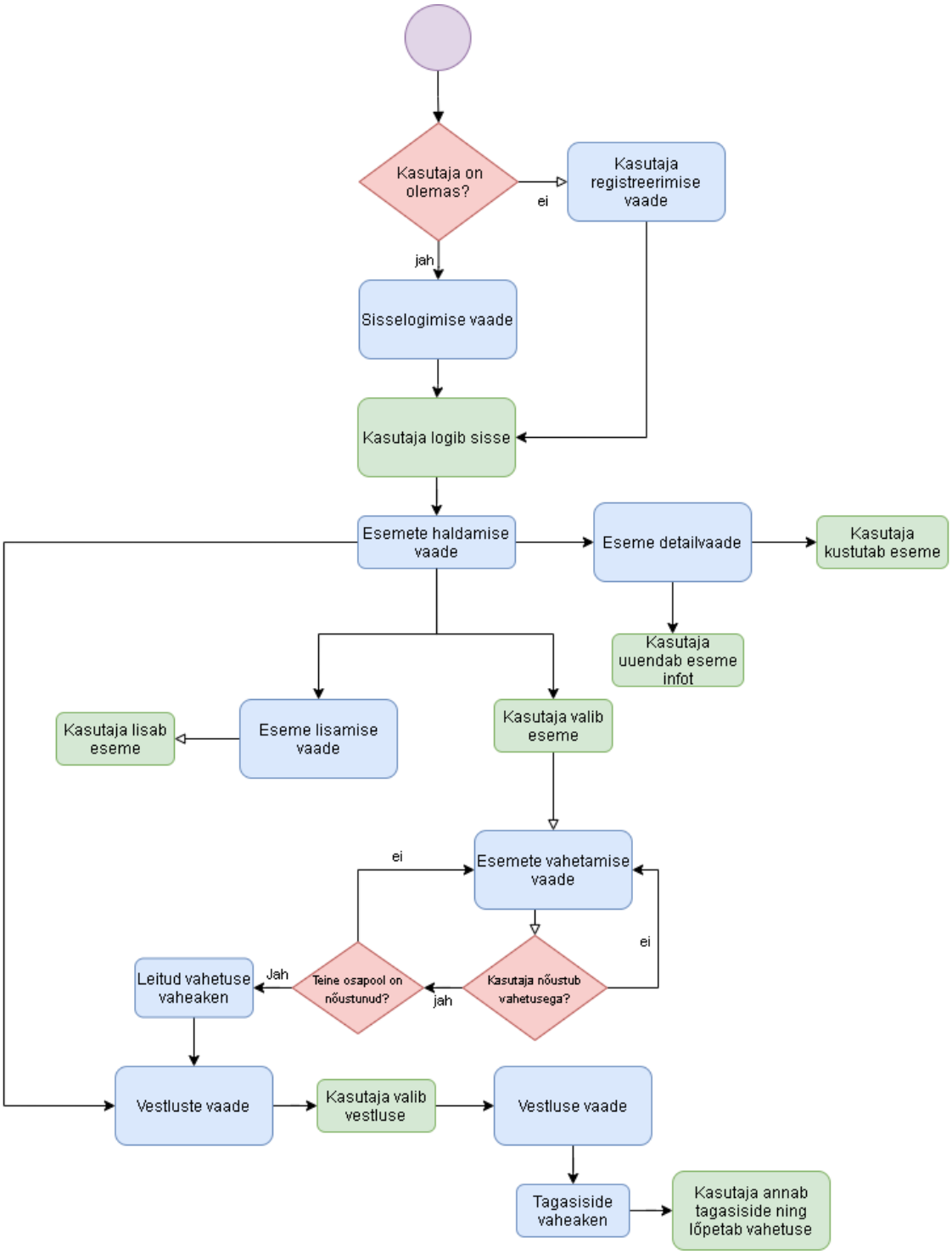
Rakendus peab katma kasutaja vaatest järgmiseid funktsionaalseid nõudeid:

1. Kasutajate haldamine
    1. Kasutaja loomine meilipõhiselt — kasutaja sisestab meiliaadressi, kasutajanime ning salasõna, mille alusel luuakse uus kasutaja. Ühe meiliaadressi kohta tohib rakenduses eksisteerida üks kasutaja ning salasõna peab olema üle viie tähemärgi pikkune.
    2. Sisselogimisel kasutajasessiooni tekitamine — sisselogimisel peab rakendus hoidma kasutajat sisselogituna seni, kuni kasutaja välja logib või kasutajale määratud veebitõend aegub.
    3. Kasutaja profiilipildi lisamine ning muutmine.
    4. Väljalogimine.
2. Esemete haldamine
    1. Eseme lisamine — kasutaja saab lisada rakendusse enda esemeid, kirjeldades neid artistinime, albumi või loo nime, pildi ning vajadusel lisainfo põhjal. Eseme pilti peab olema võimalik lisada kas seadme enda mälust või kasutades selleks otse seadme kaamerat.
    2. Eseme kustutamine — rakendus peab lubama kustutada lisatud esemeid ning seeläbi ka kustutama kõik esemega seotud aktiivsed vahetusega seotud vestlused.
    3. Eset kirjeldava info muutmine — rakendus peab lubama ka peale eseme lisamist eset kirjeldavat infot uuendada.
3. Esemete vahetamine
    1. Esemele vahetuseks sobivate esemete leidmine ja kuvamine — rakendus peab suutma teiste kasutajate esemete seast leida esemeid, mis ei ole veel rakenduse kontekstis ära vahetatud ning mille vahetamisest pole teine osapool juba keeldunud. Vahetuseks pakutavate esemete järjekorda mõjutab kasutajale varasemalt antud tagasiside.
    2. Pakutava võimaliku vahetusega nõustumine või keeldumine — kasutajal peab olema võimalik nõustuda või keelduda pakutava esemega tõmmates see ekraanil kas paremale või vasakule.

3. Mõlema osapoole poolt kinnitatud vahetuste tuvastamine — kui nõustutakse pakutava esemega ning teine osapool on varem samuti nõustunud, peab rakendus kasutajat sellest teavitama, kuvades vaheakna, mille kaudu on võimalik liikuda vestlusaknasse vahetuspartneriga vahetust arutama.

4. Vestlused
    1. Vestlusakna tekkimine peale mõlema osapoole nõustumist potentsiaalse vahetusega.
    2. Sõnumite saatmine ning lugemine.
    3. Vestluse lõpetamine ning vahetuspartneri tagasisidestamine.

Funktsionaalsed nõuded kasutaja vaatest on visualiseeritud Joonisel 1.



Joonis 1. Tegevusskeem kasutaja vaatest. Sinisena on välja toodud vaated ja vaheaknad, rohelisega on märgistatud kasutaja tegevused.

4. Realisatsioon

4.1. Tehnoloogilised valikud

Kasutajaliidese arendamiseks sai valitud raamistik React-Native [8]. React-Native kasuks otsustasin seetõttu, et sellel on väga palju aktiivseid kasutajaid [9] ning tänu sellele on suurem tõenäosus leida tekkivatele probleemidele hõlpsamini lahendusi. Lisaks kiirendas arendusprotsessi varasem arenduskogemus kõnealuse raamistikuga. Kasutajaliidese arendamisel lisandus ka hulgaliselt teeke, mis tegid väiksemate funktsionaalsuste arendamise kiiremaks ja efektiivsemaks.

Tagaarakenduse arendamiseks sai valitud Java raamistik Spring Boot [10]. Selle raamistiku valisin peamiselt isikliku varasema kogemuse tõttu, tänu millele on arendusprotsess kiirem ja efektiivsem. Lisaks sellele on selles raamistikus lihtne suhelda SQL andmebaasidega ning käsitseda REST-päringuid, mida kasutajaliides tagarakendusele saadab. Tagarakenduse *host*'imiseks olen valinud Amazon Web Service'i [11] Elastic Beanstalk teenuse, mis võimaldab lihtsasti jooksutada ja konfigureerida Spring Boot rakendust.

Andmebaasi haldamise süsteemiks on valitud MariaDB [12], mis on sarnaselt enam tuntud andmebaasisüsteemiga MySql [13] relatsiooniline andmebaas. Andmebaasi *host*'imiseks olen valinud Amazon Web Service'i RDS-teenuse, mis võimaldab samuti lihtsasti konfigureerida ja jooksutada MariaDB andmebaasi.

Kuna rakendus peab võimaldama ka pildifaile salvestada, olen nende talletamiseks valinud Amazon S3 Bucketi teenuse.

4.2. Arendusprotsess

Rakenduse edukaks arenduseks sai välja valitud sobivad töövahendid arenduseks, mugav lokaalne töökeskkond, milles on võimalik kiiresti proovida koodimuudatusi, ja samuti sai paika pandud arendusprotsessi erinevad etapid, et arendus kulgeks süstemaatiliselt ning oleks võimalik paremini jälgida tööprogressi.

4.2.1. Arendusvahendid

Kuna rakenduse terviklahendus hõlmab endas nii tagarakendust kui ka kasutajaliidest, mis erinevad oma programmeerimiskeeltelt, siis võtsin koodi kirjutamiseks kasutusele VSCode'i [14], kuna antud tarkvara toetab väga hästi nii Java kui ka Javascripti arendust. See teeb arendusprotsessi mugavamaks, kuna ei pea kasutama nende arenduseks erinevaid IDE-sid.

Rakenduse lokaalses keskkonnas jooksutamiseks ja arendamiseks kasutasin Android Emulatorit ning rakenduse APK-faili ehitamiseks kasutasin Android Studiot [15]. Andmebaasi haldamiseks võtsin kasutusele programmi HeidiSQL [16]. Päringute *debug*'imiseks ning arendamiseks kasutasin Postmani [17].

4.2.2. Arenduskeskkonna seadistamine

Et arenduskeskkonda tööle seada (nii tagarakendus kui ka emulaatoris jooksev kasutajaliides), peavad olema eelnevalt installeeritud järgmised vahendid:

- Java SE Development Kit (JDK) 8 [18]
- Android Studio [15]
- Node.js [19]
- Gradle 4.0+ [20]
- MariaDB server [12]

Tagarakenduse lokaalses keskkonnas tööleseadmiseks tuleb esmalt luua kohalik MariaDB andmebaas, kuhu tuleb esmalt tekitada vajalikud tabelid (vt Lisa 1) ning

„backend/src/main/resources“ kaustas olevas „application.properties“ faili lisada vajalik info loodud andmebaasi kohta (vt Joonis 2).

```
spring.datasource.url=jdbc:mariadb://localhost:3306/exchang
erdb
spring.datasource.username=root
spring.datasource.password=parool
```

Joonis 2. Tagarakenduse andmebaasi konfiguratsiooni näidis.

Tagarakenduse jooksutamiseks on vajalik käsureal navigeeruda „/backend“ kausta ning jookustada käsklust „gradlew bootrun“.

Kasutajaliidese emulaatoris ülesseadmiseks on vajalik luua Android Studio AVD Manageri kaudu uus seade ning see tööle panna.

Kasutajaliidese emulaatoris jooksutamiseks on vajalik installeerida globaalselt *react-native-cli*.

Käsurealt installeerimise näide:

```
npm install -g react-native-cli
```

Seejärel tuleb käsureal navigeerida kausta „/app/ItemExchanger/“ ning jooksutada käsk „react-native run-android“, mille järel hakkab rakendus emulaatoris tööle.

4.2.3. Arenduse etapid

Rakenduse arenduse jaotasin eri etappidesse, et töö progress oleks lihtsamini jälgitav.

Esimese etapina arendasin rakenduse kasutajaliidese, milles olid kasutusel staatilised näidisandmed. Kui kasutajaliidese põhiline funktsionaalsus oli kaetud, oli võimalik hõlpsasti kaardistada kõik andmed, mida tagarakendus peab kasutajaliidesele saatma hakkama. See võimaldas määrata kõik päringud, mida tagarakendus peab toetama.

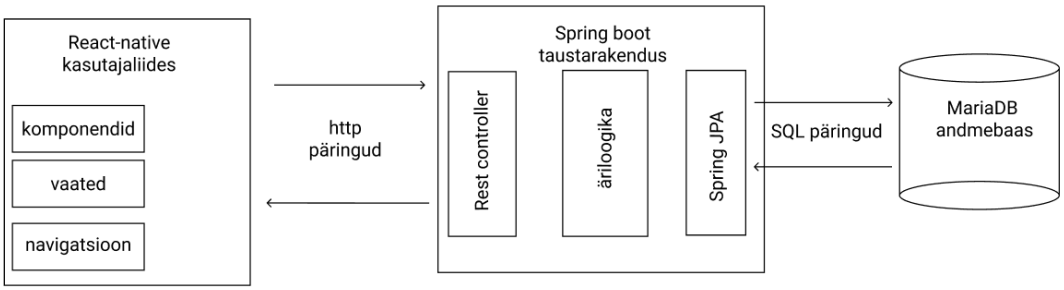
Teise etapina hakkasin arendama tagarakendust. Esmalt sai arendatud andmebaasiga suhtlemine, autentimine ning päringute vastuvõtmiseks vajalik loogika. Seejärel sai

lihtsasti hakata ükshaaval implementeerima rakenduse tööks vajalikke päringuid, mis kasutajaliidese arendamisel said ära määratud.

Kui nii kasutajaliides kui ka tagarakendus olid arendatud sellisse seisu, kus oli kaetud piisavalt nõutud funktsionaalsust, et saaks rakendust testida, hakkasin kaardistama puudujääke ning vigu, mida samm-sammult parandama hakata.

4.3. Rakenduse arhitektuur

Kasutajaliides suhtleb tagarakendusega, kasutades selleks http-päringuid. Kasutajaliidesest sõltumatult töötav tagarakendus töötleb kasutajaliidesese kaudu saadetud päringuid — peale päringu valideerimist rakendatakse vajalik äriloogika, mille käigus vajadusel suheldakse andmebaasiga. Tagarakenduse äriloogika kihis konstrueeritakse vastus saadud päringule, mis sisaldab vajadusel infot, mida kasutajaliides vastust kätte saades kasutajale kuvada saab. Kasutajaliidese ning tagarakenduse vaheliseks suhtluseks on kasutusel JSON andmestruktuuri (vt Joonis 3).



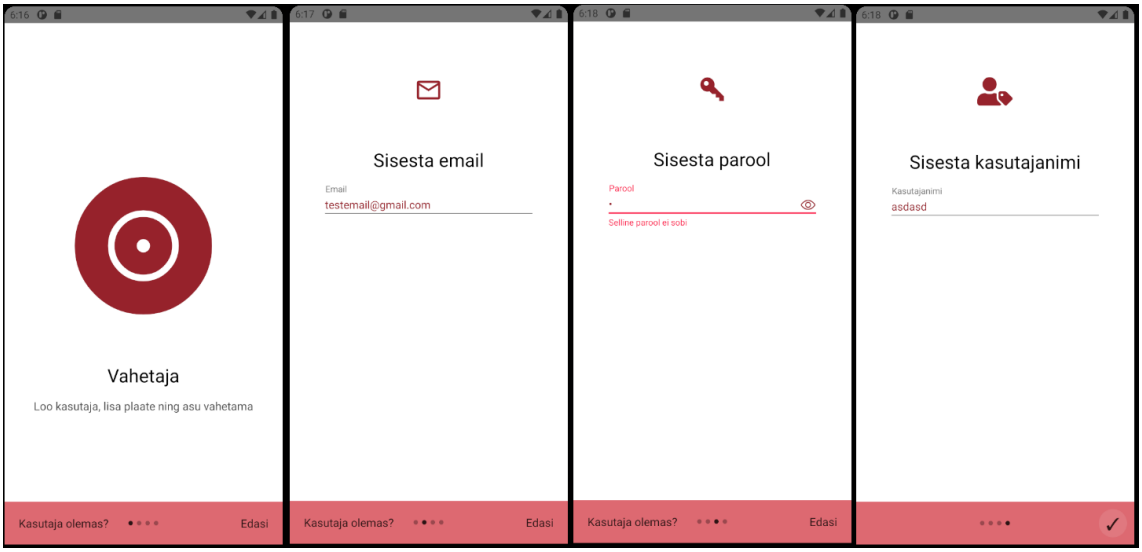
Joonis 3. Visuaalne kujutlus rakenduse kasutajaliidese, tagarakenduse ja andmebaasi vahelisest suhtlusest.

4.3.1. Kasutajaliidese arhitektuur

Kasutajaliidese arhitektuur jaguneb peamiselt kolmeks osaks — vaated, navigatsioon ning komponendid. Vaadete vahel on võimalik navigeerida ning vaated koosnevad väiksematest eraldiseisvatest või sõltuvatest komponentidest. Kasutajaliideses kasutatavad ikoonid on pärit *react-native-vector-icons* [21] raamistikust.

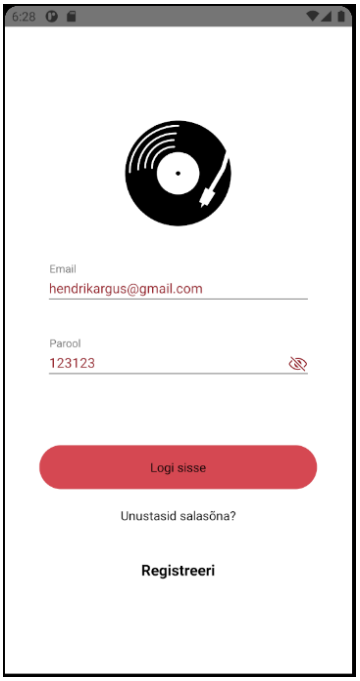
Järgnevalt annan ülevaate kasutajaliidese vaadetest, mis on kirjeldatud joonisel 1.

- Registreerimise vaade — kasutaja registreerimise vaatel võtsin kasutusele raamistiku *react-native-onboarding-swiper* [22], mis lubab hõlpsasti ehitada samm-sammulist registreerimist. Selline registreerimise ülesehitus jaotab registreerimisprotsessi mitmeks väiksemaks sammuks, mis on kasutajale mugavam, kuna kasutajalt küsitakse vähem informatsiooni korraga. Lisasin ka väljade kohese valideerimise, et kasutajale oleks võimalik kohe märku anda, kui mõni sisend ei sobi. Lisaks on veel parooli määramisel võimalik vaadata ka varjatud parooli, vajutades silmaikoonile (vt Joonis 4, kolmas ekraanitõmmis). Kui registreerimise blankett on täidetud ning kõik sisestatud väärtused sobivad, tekib nupp kasutaja loomiseks. Sellele vajutades logitakse rakendusse kasutajaga kohe sisse.



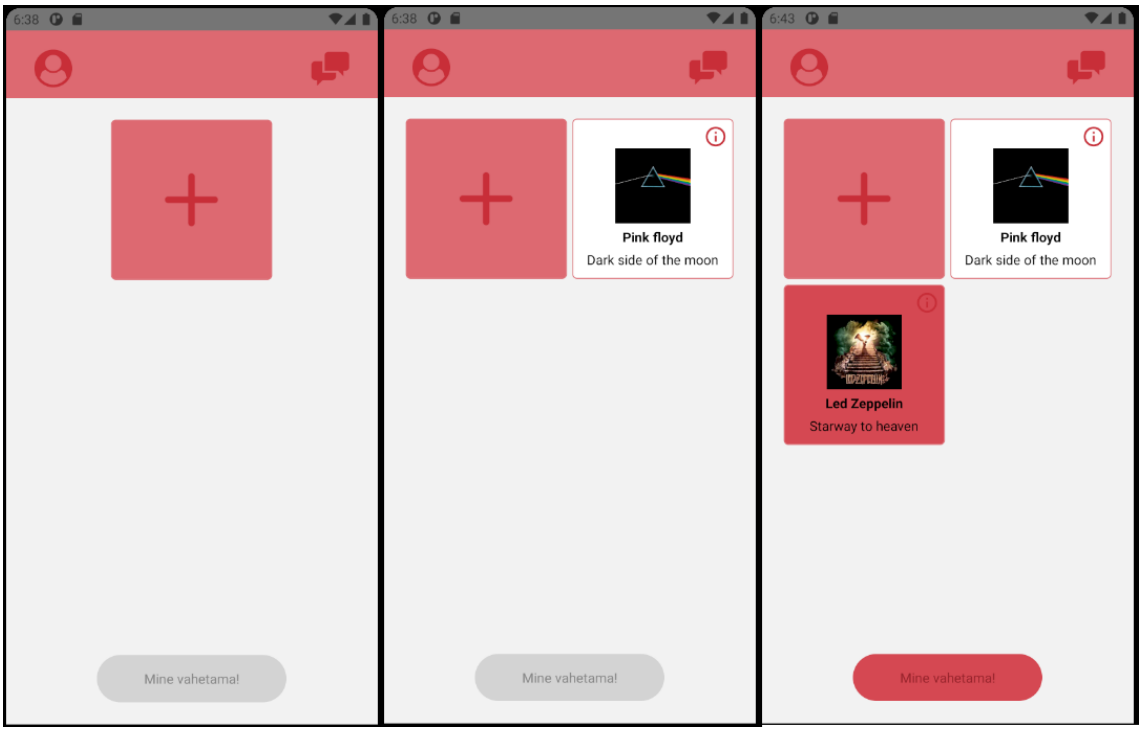
Joonis 4. Registreerimise vaate näide.

- Sisselogimise vaade — vaade hõlmab endas meiliaadressi ja parooli välja ning sisselogimise nuppu. Lisaks on ka võimalik minna tagasi registreerumise vaatesse, vajutades nupule “Registreeri”. Nagu ka registreerimise vaate lahenduses, on ka selles vaates võimalik sisestatud parooli nähtavaks teha, vajutades silmaikoonile (vt Joonis 5).



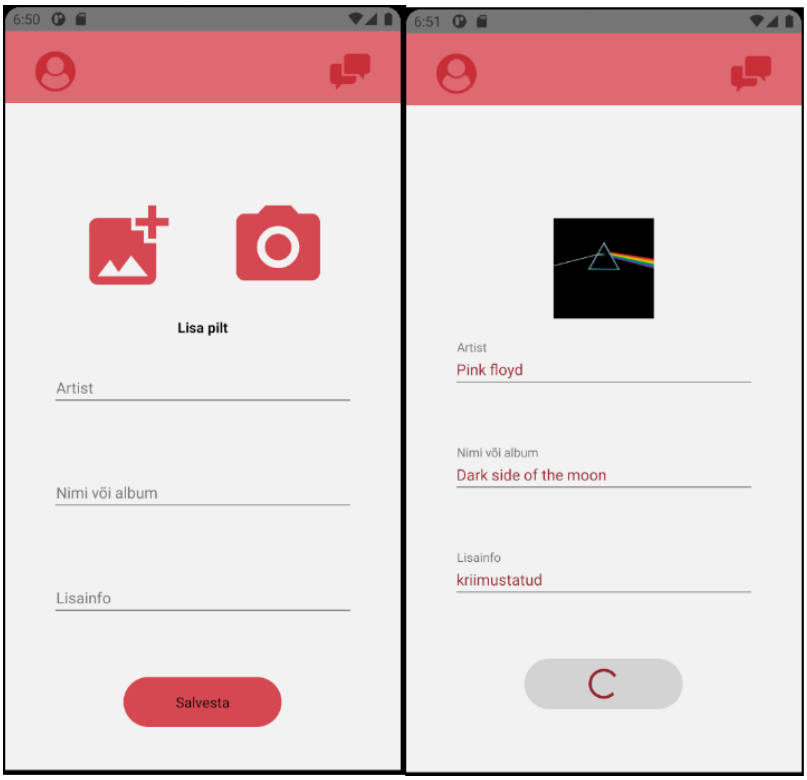
Joonis 5. Sisselogimise vaate näide.

- Esemete haldamise vaade — sellesse vaatesse suunatakse kasutaja kohe pärast sisselogimist. Vaate üleval asuvalt ribalt on võimalik liikuda edasi profiili vaatesse ning vestluste vaatesse. Eseme lisamise vaatesse on võimalik liikuda, vajutades plussmärgiga nupule (vt Joonis 6). Kui ühtegi eset ei ole lisatud, on plussmärgiga nupp joondatud ekraani keskele ning kui tekib juurde esemeid, joondatakse need kahte veergu ning eseme lisamise nupp jääb alati esimese rea esimeseks elemendiks (vt Joonis 6). Kui ekraanile enam esemed ära ei mahu, on võimalik ekraanil kerida allapoole, et neid näha. Eset kujutavate elementide küljes on ka *i*-tähega ikoon, millele vajutades navigeeritakse kasutaja eseme detailvaatesse. Vaadet on võimalik ka värskendada, tõmmates ekraanil ülevalt alla. Eset kujutava elemendi peale vajutades on võimalik see aktiveerida ning see muudab aktiivseks “Mine vahetama!” nupu, millele vajutades liigutakse valitud esemega esemete vahetamise vaatesse (vt Joonis 6, kolmas ekraanitõmmis).



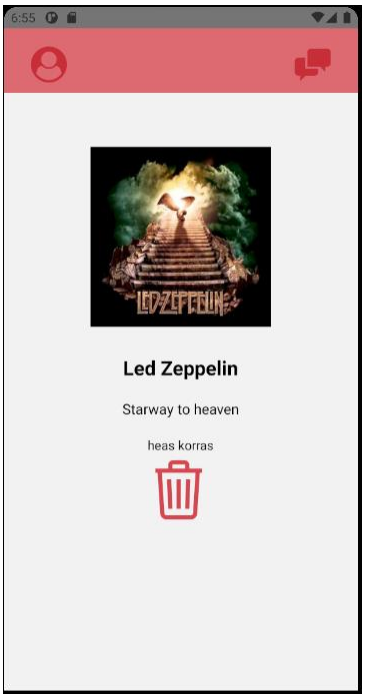
Joonis 6. Esemete haldamise vaate ekraanitõmmised.

- Eseme lisamise vaade — see vaade võimaldab lisada eseme, küsides kasutajalt selleks eset iseloomustavat pilti, artisti nime, loo või albumi nime ning lisainfot. Loo või albumi nimi ning lisainfo ei ole kohustuslikult täidetavad väljad. Pilti on võimalik lisada kahel moel — kas seadme galleriist või mälust, või siis otse kaamerat avades pilti tehes (vt joonis 7). Kuna eseme lisamise blankett ulatub väiksemate ekraanidega seadmetel ekraani alumise osasse, siis oli vajalik kasutada lahendust, mis võimaldaks väljadele fokuseerides liigutada neid ülespoole, et seadme klaviatuur neile ette ei tuleks. Eseme lisamiseks tuleb vajutada “Salvesta” nupule, mille kohale tekib animeeritud ketramist kujutav ikoon, kuniks ese on taustal salvestunud. Seejärel suunatakse kasutaja tagasi esemete haldamise vaatesse, kus on juba lisatud eset võimalik näha.



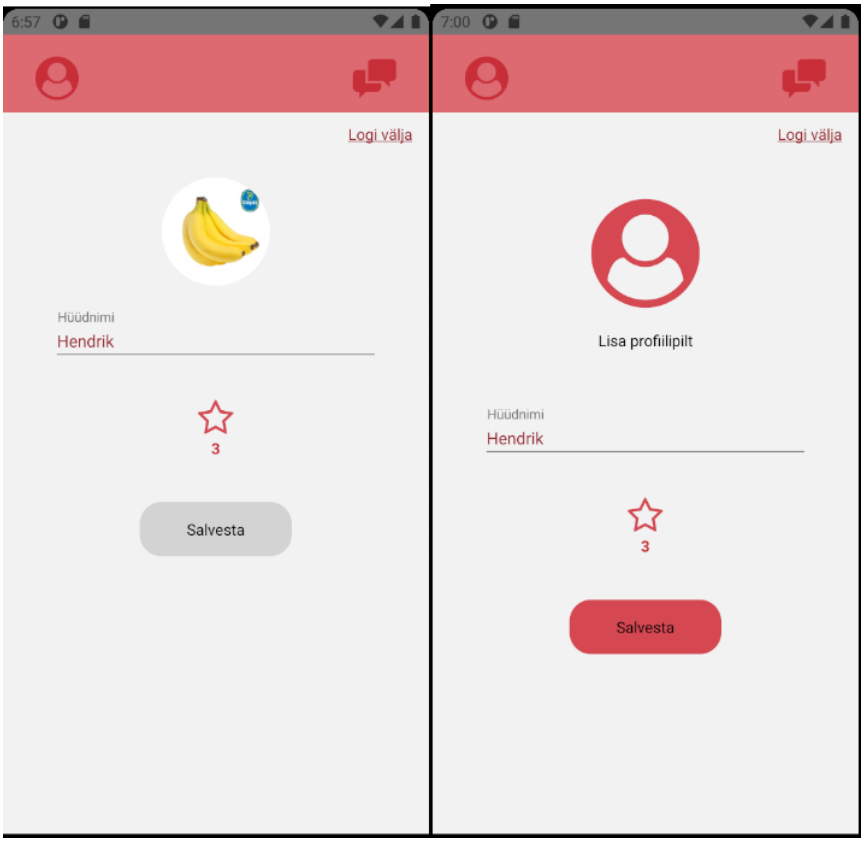
Joonis 7. Eseme lisamise vaate ekraanitõmmised.

- Eseme detailvaade — selles vaates on võimalik näha kogu informatsiooni eseme kohta ja ka kustutada ese, vajutades prügikasti ikoonile (vt joonis 8).



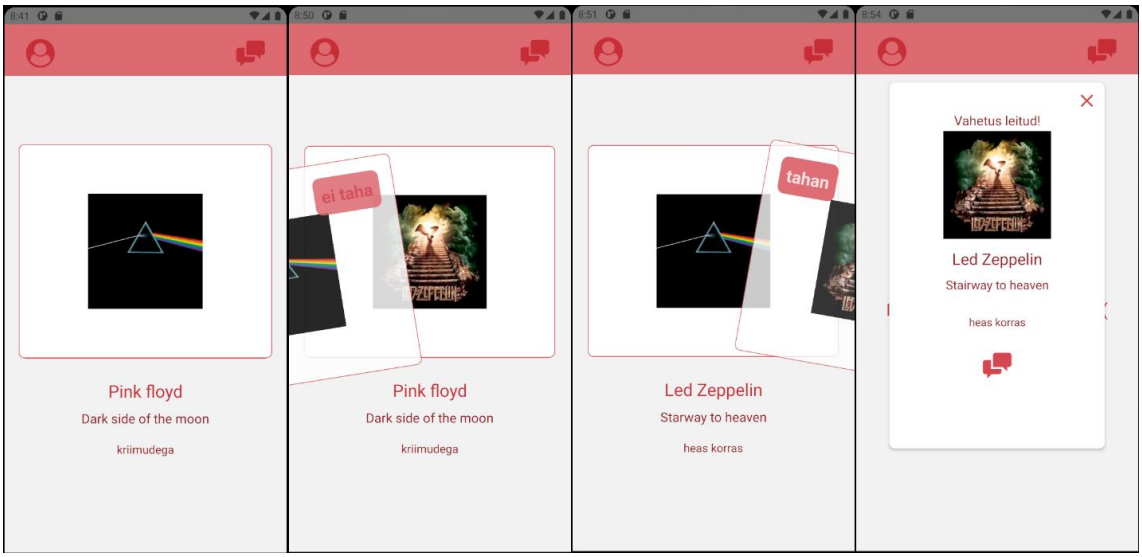
Joonis 8. Eseme detailvaate ekraanitõmmis.

- Profiili vaates kuvatakse infot kasutaja kohta koos tagasisidehinnanguga. Vaates on võimalik muuta oma hüüdnime, mida kuvatakse teistele kasutajatele, ning ka lisada profiilipilt. Samuti on selles vaates võimalik ka välja logida. “Salvesta” nupp aktiveerub, kui on lisatud või muudetud profiilipilti või hüüdnime (vt joonis 9).



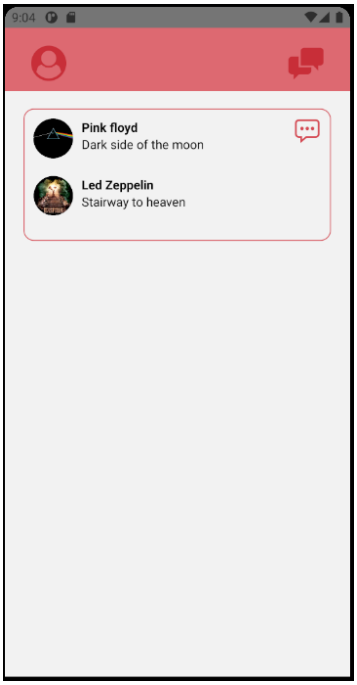
Joonis 9. Profiili vaate ekraanitõmmised.

- Esemete vahetamise vaade — selles vaates toimub potentsiaalsete vahetustega nõustumine või keeldumine, kui kasutaja tõmbab ekraanil eseme kaarti vastavalt paremale või vasakule (vt joonis 10). Potentsiaalse vahetusega nõustumisel või keeldumisel ehk kaardi ekraani serva lükkamisel kuvatakse ekraanile järgmine ese. Kui peale potentsiaalse vahetusega nõustumist rakendus tuvastab, et ka pakutava eseme omanik on antud vahetusega nõustunud, kuvatakse ekraanile vaheaknas teade, et vahetus on leitud. Vaheaknal on ka nupp, millele vajutades saab liikuda otse vestluste vaatesse. Kui rohkem vahetusse pakutavaid esemeid ei leidu, kuvatakse vastav teade ekraanile.



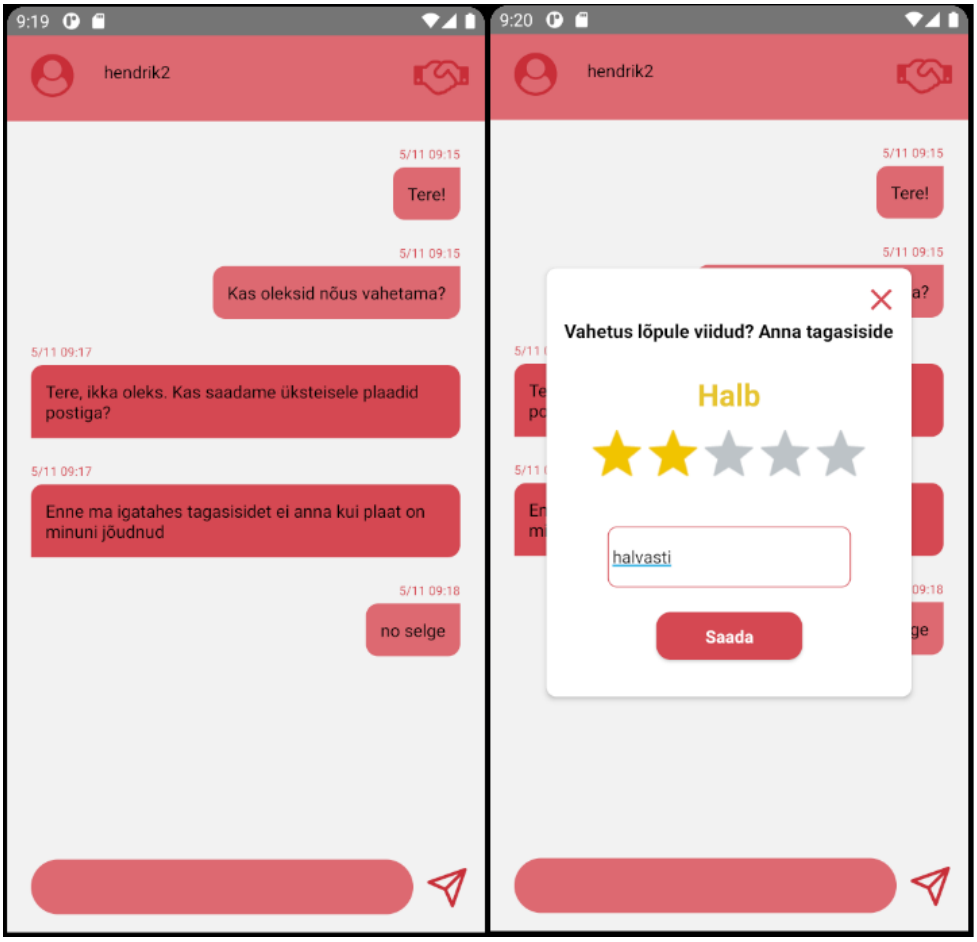
Joonis 10. Esemete vahetamise vaate ekraanitõmmised.

- Vestluste vaade — selles vaates kuvatakse kõiki leitud potentsiaalseid vahetusi, mille peal olevale vestluse ikoonile vajutades suunatakse kasutaja edasi vestluse vaatesse. Samuti on võimalik avada eseme detailvaade, kust on näiteks näha lisainfot eseme kohta, mida vahetuse komponendi peal näha ei ole (vt joonis 11).



Joonis 11. Vestluste vaate ekraanitõmmis.

- Vestluse vaade — selles vaates on võimalik sõnumeid vahetada potentsiaalse vahetuse teise osapoolega ning vahetust lõpetada, andes tagasisidet vahetuspartneri kohta tagasiside vaheaknas, mis avaneb, kui vajutada üleval paremal asuvale käesurumist kujutavale ikoonile (vt joonis 12).



Joonis 12. Vestlusakna ning tagasiside vaheakna ekraanitõmmised.

4.3.2. Tagarakenduse arhitektuur

Spring-boot tagarakendus jaguneb oma ülesehituselt peamiselt erinevateks komponentideks, nendest olulisemateks on *controller, service, repository, entity* tüüpi komponendid.

- *Controller* — kaardistab vajalikke päringuid, määratledes sisendandmete ning väljundandmete nõutud struktuuri ning päringu päised. Kutsub välja vastava Service-klassi funktsiooni, millele antakse kaasa päringu küljes olevad andmed.
- *Service* — selles kihis rakendatakse *controller*'i kaudu saadud andmetega vajalik äriloogika, vajadusel suheldakse andmebaasiga läbi *repository* ning tagastatakse vajalikud andmed.
- *Repository* — tegeleb andmebaasiga suhtlemisega, kasutades selleks *entity*-tüüpi klasse.
- *Entity* — hoiab endas vastava andmebaasi tabeli rea struktuuri (1 instants *entity*-tüüpi klassist kajastab ühte andmebaasi tabeli rida), lubab *repository*-klassil suhelda andmebaasiga hõlpsamini.

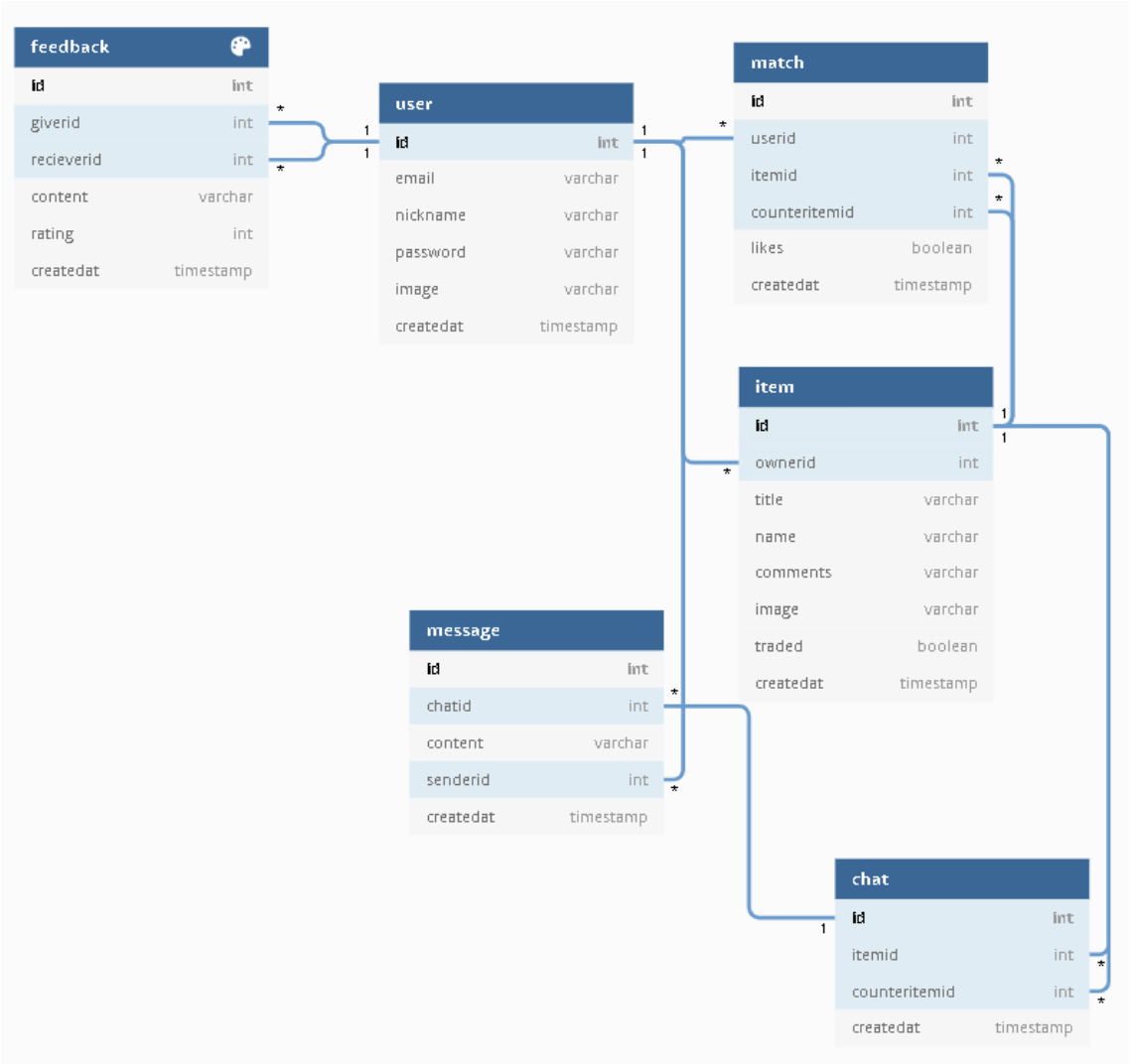
Kõik päringute JSON struktuuris olevad sisend- ja väljundandmed on kaardistatud tagarakenduse Java klassidega (vt Joonis 14), mis võimaldavad lihtsasti päringust tulevatele andmetele äriloogikat rakendada ja neid tagastada. Lisaks on kasutusel Java raamistik Lombok [23], mis võimaldab lisada klassidele lihtsasti vajalikku funktsionaalsust, nagu näiteks konstruktorid ning *get* ja *set* meetodid (vt Joonis 13)

.

```
package com.itemexchanger.backend.dto;

import java.util.List;

import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

@AllArgsConstructor
@NoArgsConstructor
@Getter
@Setter
public class ItemsResponseDto {
     private List<Item> items;
}
```

Joonis 13. Java klassi näide.

Selleks et talletada ning pärida andmebaasis kõiki vajalikke andmeid rakenduse tööks, töötasin välja andmebaasi mudeli, mis seda võimaldaks. Andmebaasi relatsiooniline mudel on kirjeldatud joonisel 14.



Joonis 14. Relatsiooniline andmebaasimudel.

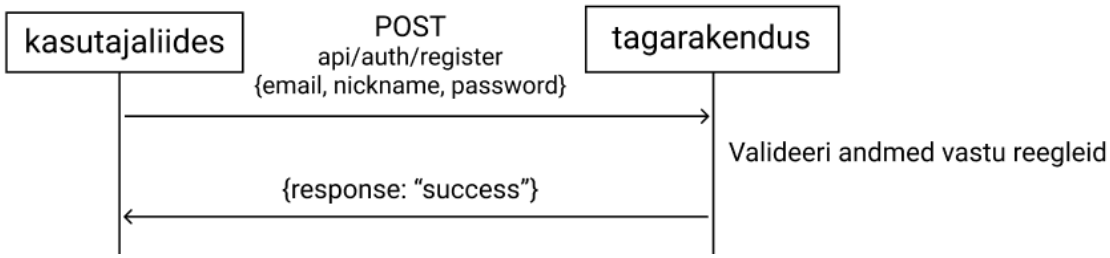
4.3.2.1. Kasutaja autentimine

Kasutaja registreerimiseks on vajalik meiliaadress, ühe meiliaadressi kohta saab eksisteerida üks kasutaja. Kasutaja loomisel edastatakse tagarakendusele POST päringuga meil, kasutaja valitud hüüdnimi ning parool. Tagarakendus kontrollib päringu sisendandmete vastavaust järgnevalt kirjeldatud reeglitele.

- Sisendsõned (meiliaadress, hüüdnimi, parool) peavad oma pikkuselt jääma fikseeritud suuruste vahele.
- Sama meiliaadressiga kasutajat ei tohi juba eksisteerida.
- Meiliaadress peab vastama kindlale mustrile, mis on kirjeldatud regulaaravaldisena.

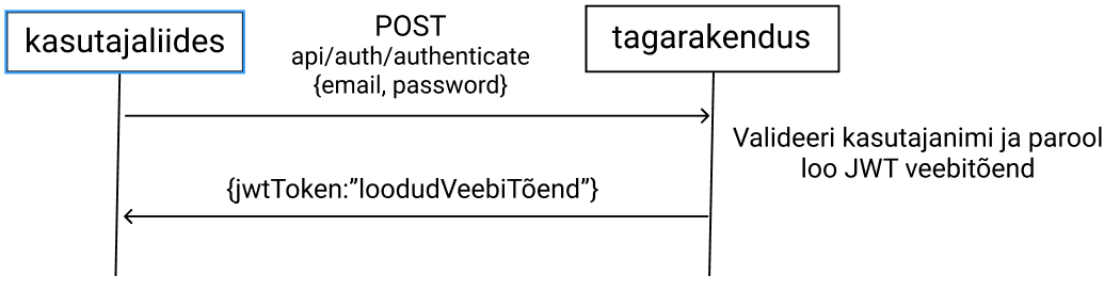
Kui sisendandmed on valideeritud, salvestatakse kasutaja andmed andmebaasi ning kasutajal on võimalik kohe sisse logida. Parool salvestatakse andmebaasi, kasutades Bcrypt [24] krüpteerimisalgoritmi.

Registreerimise päringu võimalikud vastused (vt Joonis 15) on *success*, *invalid_input*, *user_exists*.



Joonis 15. Registreerimise päringu kasutajaliidese ja tagarakenduse vaheline käsitlemine.

Kasutaja sisselogimisel tagastatakse kasutajale identsustõend veebitõendi (JWT web token [25]) vormingus. Kasutaja edasistele päringutele antakse kaasa antud veebitõend räsi kujul, milles sisaldub kasutajat puudutav info ning mida on tagarakendus suuteline lahti krüpteerima ning aru saama, millisele kasutajale antud veebitõend kuulub (vt Joonis 16).



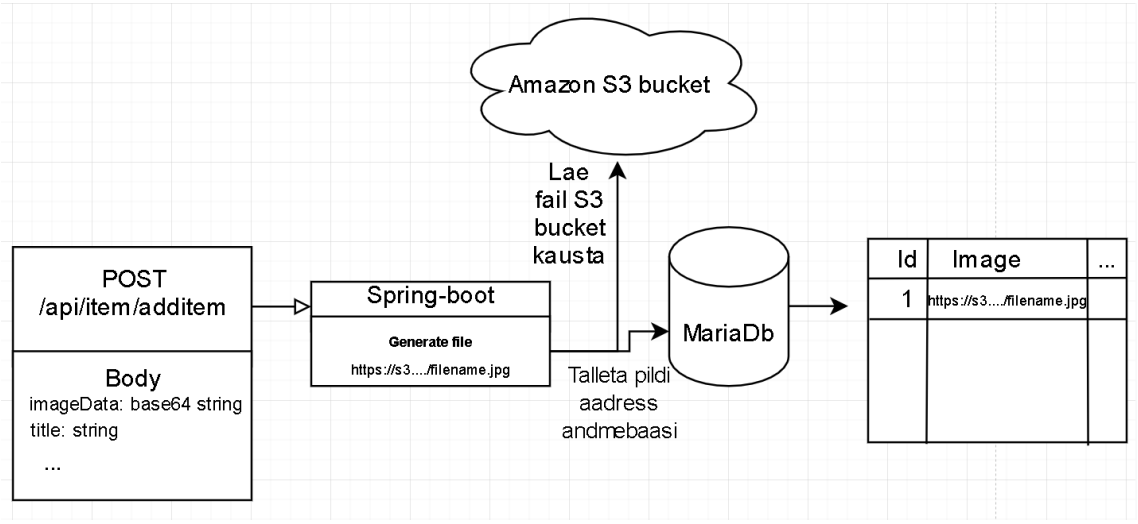
Joonis 16. Sisselogimise päringu kasutajaliidese ja tagarakenduse vaheline käsitlemine.

Kasutaja edasiste päringute päisesse annab kasutajaliides veebitõendi kaasa. Veebitõendis sisalduva info põhjal valideerib tagarakendus kasutaja ning äriloogika kihis kontrollitakse, et päringuga küsitav info oleks kasutajale ligipääsetav. Veebitõend kestab 100 päeva, mille möödudes peab kasutaja ennast uuesti rakenduses autentima. Veebitõendi väga pikk kestvusaeg sai valitud kuna rakendus ei hoia endas kasutaja jaoks tundlikku informatsiooni nagu näiteks krediitkaardi andmed.

4.3.2.2. Kasutaja esemete haldamine

Kasutajal on võimalik süsteemi esemeid lisada, enda esemete kohta infot pärida, muuta, kustutada ning ka pärida enda esemele vahetuseks sobilikke teiste kasutajate esemeid.

Eseme lisamine toimub läbi päringu POST *api/item/additem*. Päringu kehas antakse kaasa eset kirjeldav informatsioon, sealhulgas ka eset iseloomustava pildi metaandmed. Tagarakendus valideerib sisendandmed ning muudab pildi metaandmed pildifailiks, genereerides sellele unikaalse nime. Piltide kokkusurumise ning väiksemaks tegemisega tagarakendus ei tegele, selle teeb kasutajaliides ära enne päringu saatmist. Seejärel salvestab tagarakpendus pildifaili võrgus olevasse avalikku kausta, milleks on kasutusel Amazon S3 Bucket teenus. Pildifaili asukoht koos nimega salvestatakse andmebaasi koos muu eset kirjeldava infoga. Selline lahendus hoiab kokku andmebaasi mahtu, hoides andmebaasis pildifaili asukohta, mitte pildi metaandmeid (vt Joonis 17).



Joonis 17. Eseme lisamisel pildifaili talletamist illustreeriv skeem.

Kasutaja esemete pärimiseks on päring GET *api/item/items*. Päringus sisalduva veebitõendi põhjal määratakse päringu saatja kasutaja ning tagastatakse loend kõikidest kasutajale kuuluvate esemetest.

4.3.2.3. Vahetuste tuvastamine

Potentsiaalsete vahetuste kaardistamiseks olen loonud päringu POST *api/item/makeOffer*. Päringule antakse kaasa kasutaja id, kes tegevuse algatas, kasutaja eseme id, vastu pakutava eseme id, boolean väärtus, kas kasutaja soovib vahetust või mitte, vastavalt 1 või 0.

Potentsiaalsete vahetuste talletamiseks tagarakenduses lõin andmebaasi tabeli “match”. Iga potentsiaalse vahetusega keeldumise ja nõustumise korral liigub andmebaasi eelnevalt mainitud päringu küljes olev info ning ka kellaaeg ning kuupäev, millal toiming aset leidis.

Selleks, et kasutajale oleks võimalik kohe teada anda, kas teine osapool on juba varem antud potentsiaalse vahetusega nõustunud, tagastatakse vajadusel ka päringu vastuses vastav informatsioon.

Võimalikke vahetuseks sobivate esemete pärimiseks lõin päringu GET */api/item/getExchangableItems/{itemId},* millega on võimalik pärida listi võimalikest vahetuseks sobilikeks esemetest päringule kaasa antud esemele. Tagarakenduses

filtreeritakse välja kõik esemed, mida on juba varem kasutaja selle eseme vastu soovinud või mitte, ning ka kõik esemed, mille puhul on omanik juba märkinud, et ta neid ei soovi.

Et ka edaspidi oleks võimalik kasutajal pärida potentsiaalseid vahetusi, millega mõlemad osapooled on nõustunud, olen loonud päringu GET */api/item/getMatchedItems*. Päring tagastab listi esemete objektide paaridest.

4.3.2.4. Vestluste haldamine

Potentsiaalse vahetuse tekkimisel tekitatakse andmebaasi tabelisse *chat*-kirje, millel on märgitud vahetuse mõlema eseme id-d. Vahetuses osalevate kasutajate id-sid ei ole vajalik andmebaasi talletada, kuna esemed on juba andmebaasi tasemel kasutajatega seotud.

Sõnumite talletamiseks olen loonud andmebaasi tabeli "Message", mis on läbi välja *chat*'id välise võtmega seotud *chat*-tabeli kirjega. Andmebaasi talletatakse info selle kohta, kes sõnumi saatis, sõnumi sisu ning saatmise hetke kellaaeg ja kuupäev.

Sõnumite saatmiseks olen loonud päringu POST *api/chat/sendMessage*. Päringule antakse kaasa vestluse id (eelnevalt mainitud *chat*-tabeli primaarne võti) ning sõnumi sisu. Tagarakendus valideerib, kas vestlus, millesse sõnumit tahetakse lisada, kuulub kasutajale ning seejärel lisatakse info andmebaasi tabelisse *chat*.

Vestluse sõnumite pärimiseks olen loonud päringu GET *api/chat/messages/{chatid}*, millega on võimalik pärida kõiki sõnumeid päritavale vestlusele. Enne sõnumite tagastamist kontrollib tagarakendus seda, kas kasutaja on antud vestluse osapooleks. Lisaks lisatakse sõnumitele juurde märge, kas sõnum päringu esitaja saadetud või mitte.

4.3.2.5. Kasutaja tagasiside ning selle mõju

Kasutajale tagasiside andmiseks olen loonud päringu POST *api/feedback/sendfeedback*, millele antakse kaasa tagasiside arvuline hinnang ühest viieni, tagasisidet andva kasutaja antud vahetuses oleva eseme id, tagasisidestatava kasutaja vahetuses oleva eseme id ning sõnaline kommentaar. Tagarakenduses valideeritakse, et tagasiside antakse vahetuse

kohta, mis kuulub tagasisidestajale, ning ka see, et antav hinne oleks lubatud piirides (1 kuni 5). Seejärel muudetakse vahetuses olevate esemete *isTraded* omadus andmebaasis üheks, viidates, et vahetus on lõpetatud. Tagasiside hinne koos kommentaariga, kellaaja ning kuupäeva, tagasiside andja ja saaja id salvestatakse andmebaasi tabelisse “Feedback”.

Kasutaja tagasiside hinde pärimiseks olen loonud päringu GET *api/feedback/getFeedbackRating*. Tagarakendus summeerib kasutajale antud tagasiside hinde andmebaasi tabeli “Feedback” kirjete põhjal, millele liidetakse juurde 5. Tagasiside hinnete summa jagatakse tagasisidete arvuga, millele liidetakse ka 1. Selline lahendus hoiab lihtsasti ära probleemi, kui kasutajale ei ole veel tagasisidet antud ehk kasutaja tagasiside hinne oleks sellisel juhul 0.

5. Tulemused

5.1. Kasutajatestimine

Kui rakendus oli jõudnud seisu, kus suurem osa funktsionaalsetest nõuetest oli täidetud, andsin rakenduse esmalt testimiseks oma juhendajale. Esimesel rakenduse testimisel koos juhendajaga tuli välja ohtralt vigu, puudujääke ja ebamugavusi. Peamisteks probleemideks olid sisselogimine ja registreerumine, piltide aeglane laadimine, ebaselged kasutajaliidese vood, nagu näiteks see, et eseme lisamisel salvestamise nupule vajutades kasutajaliides ei indikeerinud kohe kuidagi, kas ese sai lisatud või mitte. Lisaks ilmnes palju varem märkamata jäänud väiksemaid vigu.

Peale avastatud vigade ja puudujääkide parandamist andsin rakenduse testimiseks viieliikmelisele tuttavatest koosnevale rühmale. Peale mõnepäevast rakenduse kasutust testrühma sees ilmus välja veel palju varem märkamata jäänud vigu ning ebamugavusi. Järgnevalt annan ülevaate esmasest kasutajatestimise tagasisidest rakenduse funktsionaalsete nõuete vaatest, funktsionaalsetest nõutest olen järgnevas nimekirjas eemaldanud punktid, millega seotud tagasisidet testrühmast ei saanud.

1. Kasutajate haldamine
    1. Sisselogimisel kasutajasessiooni tekitamine — ühel testrühma kasutajal tekkis olukord, et peale registreerimist ei õnnestunud sisselogida, mille põhjuseks oli andmebaasi tekkinud topeltkirje kasutajast.
    2. Kasutaja profiilipildi lisamine ning muutmine – kaks testrühma kasutajat teatasid probleemist, et ekraani üleval asuval käsklusteribal ei kuvatud profiilipilti peale profiilipildi lisamist.
2. Esemete haldamine
    1. Eseme lisamine — testrühma kõigi kasutujate seas tekitas segadust see, et peale eseme info sisestamist salvestamise nupule vajutades kasutajaliides ei indikeerinud, kas ese sai lisatud või mitte.
3. Esemete vahetamine
    1. Mõlema osapoole poolt kinnitatud vahetuste tuvastamine — testrühma tagasisidest selgus, et leitud vahetuste vaates kuvatakse teadet “Ühtegi aktiivset vahetust ei leitud” kuni taustal on andmed laetud, see tekitas kasutajates segadust.

4. Vestlused
    1. Vestluse lõpetamine ning vahetuspartneri tagasisidestamine – peale vahetuspartneri tagasisidestamist kustub kasutajaliidesest vestlus, see tekitas segadust testrühma kasutajate seas kuna ei olnud võimalik peale tagasisidestamist vestlusest järgi vaadata näiteks vahetatud kontaktinfot.

Järgnevalt loetlen kasutajatestimise tagasisidest välja tulnud puuduseid, mis funktsionaalsete nõuete vaatest välja jäid.

- Osal vaadetest on puudu indikaator, et andmeid laaditakse.
- Puudub võimalus peale seadme enda *back* nupuga liikuda tagasi esemete haldamise vaatesse.

Rakenduse testimisel välja tulnud vigade parandamine sai järgneva arenduse peamiseks eesmärgiks.

5.2. Edasise arenduse eesmärgid

Rakenduse arendusega on plaanis jätkata ning jõuda rakenduse valmidusega niikaugele, et see saadavaks teha ka laiemale publikule. Hetkel veel puudub rakendusest nii mõnigi oluline funktsionaalsus, mida lõpuutöö projekti raames ei jõudnud realiseerida. Järgnevalt loetlen mõned edasised arenduse eesmärgid.

- Kasutusele tuleks võtta mõni autentimise raamistik, nagu näiteks OAuth [26], mis võimaldaks integreerida peale meiliaadressiga kontoga kasutaja loomise ka sotsiaalmeedia kontosid nagu näiteks Facebook.
- Kasutaja asukoha määramine — teades kasutaja asukohta, oleks võimalik pakutavate vahetuste järjekorda sellest mõjutada ehk üksteisele lähemal olevad kasutajad näeksid üksteise esemeid esimeses järjekorras.
- Võimalus vaadata potentsiaalse vahetuspartneri teisi esemeid — see võimaldaks lihtsamini leida veel teisi vahetusi, mida kasutajad omavahel saaksid teha.
- Rakendus üle viia ka IOS platvormile — kui rakendus töötab ainult Androidiga, jääb paljudest potentsiaalsetest kasutajatest ilma.
- Luua kasutajatingimused ning privaatsustingimused, et hoida ära võimalike probleemide tekkimist.
- Vestlusakna lahendus viia WebSocket [27] ühendusele, et saavutada täielikult reaalajas toimiv vestlemine. Hetkel on sõnumi saatmisest nägemiseni maksimaalselt kuni umbes viiesekundiline viide.

Kokkuvõte

Lõputöö eesmärgiks oli luua mobiilirakendus, mis võimaldaks vahetada esemeid mängulises võtmes ning looks seeläbi uue võimaluse toetada taaskasutust.

Lõputöö teostamisel sõnastasin rakenduse funktsionaalsed nõuded, arendasin neid, jälgides väljatöötatava kasutajaliidese kui ka tagarakenduse ehk kogu rakenduse terviklahenduse.

Lõputöö projekti arendades õppisin, kuidas arendada mobiilirakendust alustades nii-öelda puhtalt lehelt, omandasin uusi teadmisi ja kogemusi nii kasutajaliidese kui ka tagarakenduse ja andmebaasi disainimisel ja arendamisel. Samuti omandasin kogemusi Amazon Web Service'i teenustega, kuidas neid teenuseid kasutada ja üles seada.

Kindlasti ei saa öelda, et rakendus oleks lõputöö esitamise ajal täiesti valmis ja veavaba, kuid sellest hoolimata katab rakendus kõiki rakenduse põhilise funktsionaalsuse toimimiseks vajalikke nõudeid. Rakenduse arendust jätkan ka tulevikus, lisades funktsionaalsust ja parandades üldisemalt kasutajakogemust, mida lõputöö projekti raames teha ei jõudnud. Sellegi poolest on hetkel rakendus saadaval ka Google Play poes nime all „Vahetaja".

Kasutatud kirjandus

[1] A. Mastering, „Bulletin E 3: Standards for Stereophonic Disc Records,“ 15.11.2014. [Võrgumaterjal]. Available: http://www.aardvarkmastering.com/riaa.htm. [Kasutatud 10.11.2014].

[2] „SmartSwap,“ [Võrgumaterjal]. Available: https://www.smartswap.com/. [Kasutatud 15.5.2021].

[3] „Hula Trading,“ [Võrgumaterjal]. Available: https://hula.trading/. [Kasutatud 16.5.2021].

[4] „Freecycle,“ [Võrgumaterjal]. Available: https://www.freecycle.org/. [Kasutatud 10.4.2021].

[5] Tinder Inc., „Tinder,“ [Võrgumaterjal]. Available: https://tinder.com/. [Kasutatud 17.5.2021].

[6] „Upland,“ [Võrgumaterjal]. Available: https://uplandsoftware.com/localytics/resources/blog/21-percent-of-users-abandon-apps-after-one-use/. [Kasutatud 10.4.2021].

[7] „Broadband Search,“ [Võrgumaterjal]. Available: https://www.broadbandsearch.net/blog/mobile-desktop-internet-usage-statistics. [Kasutatud 10.4.2021].

[8] Facebook, Inc, „React Native,“ [Võrgumaterjal]. Available: https://github.com/facebook/react-native.

[9] Enlyft, „Enlyft,“ [Võrgumaterjal]. Available: https://enlyft.com/tech/products/react-native. [Kasutatud 16.5.2021].

[10] „Spring Boot,“ Pivotal Software, [Võrgumaterjal]. Available: https://spring.io/projects/spring-boot. [Kasutatud 16.5.2021].

[11] Amazon.com, Inc, [Võrgumaterjal]. Available: https://aws.amazon.com/. [Kasutatud 16.5.2021].

[12] „MariaDB,“ [Võrgumaterjal]. Available: https://mariadb.org/. [Kasutatud 16.5.2021].

[13] Oracle Corporation, „MySQL,“ [Võrgumaterjal]. Available: https://www.mysql.com/. [Kasutatud 16.5.2021].

[14] Microsoft, „Visual Studio Code,“ [Võrgumaterjal]. Available: https://code.visualstudio.com/. [Kasutatud 16.5.2021].

[15] Google LLC, JetBrains s.r.o, „Android Studio,“ [Võrgumaterjal]. Available: https://developer.android.com/. [Kasutatud 16.5.2021].

[16] A. Becker, „HeidiSQL,“ [Võrgumaterjal]. Available: https://www.heidisql.com/. [Kasutatud 16.5.2021].

[17] Postman, Inc, „Postman,“ [Võrgumaterjal]. Available: https://www.postman.com/. [Kasutatud 16.5.2021].

[18] Oracle Corporation, „Oracle,“ [Võrgumaterjal]. Available: https://www.oracle.com/java/technologies/javase/javase-jdk8-downloads.html. [Kasutatud 16.5.2021].

[19] OpenJS Foundation, „NodeJS,“ [Võrgumaterjal]. Available: https://nodejs.org/en/. [Kasutatud 16.5.2021].

[20] Gradle Inc., „Gradle,“ [Võrgumaterjal]. Available: https://gradle.org/releases/. [Kasutatud 16.5.2021].

[21] J. Arvidsson, „React Native vector icons,“ [Võrgumaterjal]. Available: https://github.com/oblador/react-native-vector-icons. [Kasutatud 16.5.2021].

[22] „NPM,“ [Võrgumaterjal]. Available: https://www.npmjs.com/package/react-native-onboarding-swiper. [Kasutatud 16.5.2021].

[23] R. S. Reinier Zwitserloot, „Lombok,“ [Võrgumaterjal]. Available: https://projectlombok.org/. [Kasutatud 16.5.2021].

[24] D. M. Niels Provos, „Usenix,“ [Võrgumaterjal]. Available: https://www.usenix.org/legacy/publications/library/proceedings/usenix99/full_papers/provos/provos_html/node5.html. [Kasutatud 16.5.2021].

[25] Auth0® Inc, „JWT,“ [Võrgumaterjal]. Available: https://jwt.io/. [Kasutatud 16.5.2021].

[26] „OAuth,“ [Võrgumaterjal]. Available: https://oauth.net/. [Kasutatud 16.5.2021].

[27] Ian Fette, Google Inc, „Datatracker,“ [Võrgumaterjal]. Available: https://datatracker.ietf.org/doc/html/rfc6455. [Kasutatud 16.5.2021].

Lisa 1 – Andmebaasi versiooni info ning tabelite loomise päringud

```
-- --------------------------------------------------------

-- Host:                         127.0.0.1

-- Server version:               10.5.9-MariaDB - mariadb.org
binary distribution

-- Server OS:                    Win64

-- HeidiSQL Version:             11.0.0.5919

-- --------------------------------------------------------


/*!40101                                                    SET
@OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;

/*!40101 SET NAMES utf8 */;

/*!50503 SET NAMES utf8mb4 */;

/*!40014  SET  @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0 */;

/*!40101           SET           @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;



-- Dumping database structure for exchangerdb

CREATE DATABASE IF NOT EXISTS `exchangerdb` /*!40100 DEFAULT
CHARACTER SET latin1 */;

USE `exchangerdb`;


-- Dumping structure for table exchangerdb.chat
```

```
CREATE TABLE IF NOT EXISTS `chat` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `itemid` int(11) DEFAULT NULL,
  `counteritemid` int(11) DEFAULT NULL,
  `createdat` datetime DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `FK_CHAT_ITEMID` (`itemid`),
  KEY `FK_CHAT_COUNTERITEMID` (`counteritemid`),
  CONSTRAINT `FK_CHAT_COUNTERITEMID` FOREIGN KEY (`counteritemid`) REFERENCES `item` (`id`),
  CONSTRAINT `FK_CHAT_ITEMID` FOREIGN KEY (`itemid`) REFERENCES `item` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=15 DEFAULT CHARSET=latin1;

-- Data exporting was unselected.

-- Dumping structure for table exchangerdb.feedback
CREATE TABLE IF NOT EXISTS `feedback` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `giverid` int(11) DEFAULT NULL,
  `receiverid` int(11) DEFAULT NULL,
  `content` varchar(300) DEFAULT '',
  `rating` tinyint(4) DEFAULT NULL,
  `createdat` datetime DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `FK_FEEDBACK_USERID` (`giverid`) USING BTREE,
  KEY `FK_FEEDBACK_RECEIVERID` (`receiverid`),
  CONSTRAINT `FK_FEEDBACK_GIVERID` FOREIGN KEY (`giverid`) REFERENCES `user` (`id`),
```

```
  CONSTRAINT    `FK_FEEDBACK_RECEIVERID`    FOREIGN    KEY
(`receiverid`) REFERENCES `user` (`id`)

) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=latin1;


-- Data exporting was unselected.


-- Dumping structure for table exchangerdb.item

CREATE TABLE IF NOT EXISTS `item` (

  `id` int(11) NOT NULL AUTO_INCREMENT,

  `ownerid` int(11) NOT NULL,

  `title` varchar(100) NOT NULL,

  `name` varchar(100) NOT NULL,

  `image` varchar(255) NOT NULL,

  `comment` varchar(255) DEFAULT NULL,

  `createdat` datetime DEFAULT NULL,

  `traded` tinyint(1) DEFAULT 0,

  PRIMARY KEY (`id`),

  KEY `FK_ITEM_OWNERID` (`ownerid`),

  CONSTRAINT  `FK_ITEM_OWNERID`  FOREIGN  KEY  (`ownerid`)
REFERENCES `user` (`id`)

) ENGINE=InnoDB AUTO_INCREMENT=26 DEFAULT CHARSET=latin1;


-- Data exporting was unselected.


-- Dumping structure for table exchangerdb.matches

CREATE TABLE IF NOT EXISTS `matches` (

  `id` int(11) NOT NULL AUTO_INCREMENT,

  `itemid` int(11) DEFAULT NULL,
```

```
  `userid` int(11) DEFAULT NULL,
  `counteritemid` int(11) DEFAULT NULL,
  `isliked` tinyint(1) NOT NULL DEFAULT 0,
  `createdat` datetime DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `userid` (`userid`),
  KEY `itemid` (`itemid`),
  KEY `counteritemid` (`counteritemid`),
  CONSTRAINT `FK_MATCH_COUNTERITEMID` FOREIGN KEY
(`counteritemid`) REFERENCES `item` (`id`),
  CONSTRAINT `FK_MATCH_ITEMID` FOREIGN KEY (`itemid`)
REFERENCES `item` (`id`),
  CONSTRAINT `FK_MATCH_USERID` FOREIGN KEY (`userid`)
REFERENCES `user` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=76 DEFAULT CHARSET=latin1;

-- Data exporting was unselected.

-- Dumping structure for table exchangerdb.message
CREATE TABLE IF NOT EXISTS `message` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `chatid` int(11) DEFAULT NULL,
  `content` varchar(500) DEFAULT NULL,
  `senderid` int(11) DEFAULT NULL,
  `createdat` datetime DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `FK_MESSAGE_CHATID` (`chatid`),
  KEY `FK_MESSAGE_SENDERID` (`senderid`),
```

```
  CONSTRAINT  `FK_MESSAGE_CHATID`  FOREIGN  KEY  (`chatid`)
REFERENCES `chat` (`id`),

  CONSTRAINT `FK_MESSAGE_SENDERID` FOREIGN KEY (`senderid`)
REFERENCES `user` (`id`)

) ENGINE=InnoDB AUTO_INCREMENT=56 DEFAULT CHARSET=latin1;


-- Data exporting was unselected.


-- Dumping structure for table exchangerdb.user

CREATE TABLE IF NOT EXISTS `user` (

  `id` int(11) NOT NULL AUTO_INCREMENT,

  `password` varchar(255) NOT NULL DEFAULT '',

  `email` varchar(50) NOT NULL,

  `nickname` varchar(50) DEFAULT NULL,

  `image` varchar(255) DEFAULT NULL,

  `createdat` datetime DEFAULT NULL,

  `rating` int(11) NOT NULL DEFAULT 5,

  PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO_INCREMENT=18 DEFAULT CHARSET=latin1;


-- Data exporting was unselected.


/*!40101 SET SQL_MODE=IFNULL(@OLD_SQL_MODE, '') */;

/*!40014  SET  FOREIGN_KEY_CHECKS=IF(@OLD_FOREIGN_KEY_CHECKS
IS NULL, 1, @OLD_FOREIGN_KEY_CHECKS) */;

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT
*/;
```

Lisa 2 – Java klassi näide

```
package com.itemexchanger.backend.dto;

import java.util.List;

import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

@AllArgsConstructor
@NoArgsConstructor
@Getter
@Setter
public class ItemsResponseDto {
     private List<Item> items;
}
```