TALLINNA TEHNIKAÜLIKOOL

Infotehnoloogia teaduskond

Uku-Laur Jagomägi 179789IADB

Elektritõukeratta laadimiskapi eesrakenduse arendamine

Bakalaureusetöö

Juhendaja: Meelis Antoi

Magistrikraad

Tallinn 2022

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: Uku-Laur Jagomägi

21.11.2021

Annotatsioon

Käesoleva diplomitöö põhieesmärk oli valmis disainida ning programmeerida elektritõukeratta laadimisjaama eesrakendus. Samuti kuulus eesmärkide hulka laadimisjaama visuaalne disainimine.

Antud lõputöö analüüsi osas eritleti elektritõukeratta laadimisjaama potentsiaalsete klientide kasutajalugusid ning kasutajate funktsionaalseid ja mittefunktsionaalseid nõudeid. Käesoleva diplomitöö analüüsi osas on võrreldud erinevaid arenduskeskkondi ning arendustehnoloogiaid.

Käesoleva diplomitöö skoop koosneb eesrakenduse disainimisest Axure tarkvara kasutades, eesrakenduse arendamisest kasutades Xamarin platvormi ning toote enda disainimisest.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 46 leheküljel, 5 peatükki, 18 joonist, 0 tabelit.

Abstract

Electric Scooter Charging Station's Front-End Application Development

The purpose of this final thesis is to design a scooter charging station, a front-end application and to develop the front-end application.

The author analysed potential clients user stories and their functional and non-functional requirements. The author of this thesis has compared different development environments and development technologies.

The scope of this final thesis includes creating the application's prototype using Axure software, development of the front-end application using Xamarin platform and designing the scooter charging station itself.

The thesis is in Estonian and contains 46 pages of text, 5 chapters, 18 figures, 0 tables.

Lühendite ja mõistete sõnastik

Adobe PhoneGap	Ristplatvorm mobiilirakenduste arendamiseks
Android	Modifitseeritud Linux kernelil baseeruv mobiiltelefonide operatsioonisüsteem
API	*Application Programming Interface*, rakendusliides
Code-Behind	C# kood, mis töötab koos visualiseerimiseks mõeldud XAML failidega
Git	Vabavaraline versioonihaldus süsteem
iOS	Apple korporatsiooni poolt loodud mobiiltelefonide operatsioonisüsteem
Native	spetsiifilisele operatsioonisüsteemile kirjutatud kood
NDK	*Native Development Kit* – tööriistakomplekt, mis võimaldab manustada komponente Android rakenduste omakoodi kasutamiseks
NuGet package	Kokku pakitud fail .nupkg faililaiendiga
RGB	Liitvärvimudel
Stackoverflow	Avalik platvorm programmi koodi küsimuste ning vastuste kohta
UX	*User experience* ehk kasutaja kogemus
Xamarin	Ristplatvorm mobiilirakenduste arendamiseks

Sisukord

Jooniste loetelu

Koodinäidete loetelu

1 Sissejuhatus

Enne autode laialdast levikut liiklesid inimesed põhiliselt pikamaa distantsidel hobustega, kuid lühemate vahemaade läbimistel eelistati siiski jalutamist või jalgrattaga sõitmist. Elektrilised tõukerattad on uueks ning huvitavaks trendiks tänases linnalises mikrotranspordis. Mikro-liiklus on linnaline transpordiviis, mille raames ei liiguta enam kui 8 kilomeetrit. Mikro liiklemisvahendite alla kuulusid eelnevalt vaid isesõitvad autod, ühiskasutatavad jalgrattad, jalgsi käimine, kuniks murdsid 2018. aastal end sinna ringi elektritõukerattad [11]. Elektritõukerattad pole ainsad uued liikmed mikro liiklemisvahendite seas. Sinna hulka kuuluvad ka elektrirulad, elektrilised üksrattad ja tasakaaluliikurid.

Antud lõputöö autor on ka ise tihe elektritõukeratta kasutaja ning omab enda isiklikku tõukeliikurit. Autor on peaaegu igapäevase elektritõukeratta kasutajana tihti seisnud silmitsi küsimusega, et kuhu panna enda tõukeliikur laadima, kui on tahtmine minna näiteks randa päevitama või kuhu hoiustada enda elektriline tõukeratas ning portfell kui on vaja minna poodi sisseoste tegema.

Käesolevas lõputöös vaadeldakse ning analüüsitakse probleemi Eestis ning võrreldakse juba eksisteerivaid sarnaseid lahendusi, mida pakutakse elektriliikurite kasutajatele. Antud probleemi lahendamiseks pakub autor välja elektritõukeratastele mõeldud laadimiskapi, milles on võimalik laadida ilmastikukindlalt nii elektritõukerattaid, elektrirulasid, elektrilisi üksrattaid, tõukeliikureid kui ka hoiustada enda esemeid.

Lõputöö eesmärgiks on valmis disainida laadimiskapp, eesrakendus ning luua elektritõukeratta laadimiskapi mobiilirakendus Android operatsioonisüsteemiga nutitelefonidele.

Vastavalt lõputöö eesmärkidele on autor püstitanud endale järgnevad ülesanded:

- Disainida valmis elektritõukeratastele mõeldud laadimiskapp.
- Disainida valmis elektritõukerataste laadimiskapi eesrakendus.
- Valmis programmeerida elektritõukerataste laadimiskapi eesrakendus.

Antud lõputöö lähtetingimused hõlmavad endas algteadmist, et mobiiltelefoni eesrakendus luuakse Android mobiilidele.

1.1 Metoodika

Antud lõputöös antakse ülevaade lähemalt probleemi sisu ning selle olemasolu kohta, seejärel pakutakse välja lõputöö skoopi jääv probleemi paikav lahendus, mille raames disainitakse ning luuakse eesrakendus.

Lahenduse analüüsi osas leitakse disainimise alustaladele toetudes sobiv visuaalne tulemus. Samuti analüüsitakse ja põhjendatakse tehnoloogiate kui ka arendustarkvarade valikuid. Analüüsi hulka kuulub ka erinevate kasutajagruppide funktsionaalsete ning mittefunktsionaalsete nõuete kirjeldamine.

Antud lõputöö on üles ehitatud erinevates osades, milles vaadeldakse toote disaini, eesrakenduse disaini ning eesrakenduse valmimist. Samuti pakutakse välja järgmised verstapostid täies mahus funktsioneeriva toote välja töötamiseks, mis lõputöö skoopi ei mahtunud.

Lõputöö kasutatud kirjandus ning viitamine on tehtud Tallinna Tehnikaülikoolis IEEE stiilis.

2 Ülevaade probleemist

Elektritõukeratastel on viimastel aastatel olnud rahva hulgas suur menu. Iga aastaga näeme tänavapildis üha rohkem elektritõukerattaid. Näiteks 2018. aasta juulikuu lõpus võis leida Eesti pealinnas, Tallinnas liiklemas kõigest 200-250 Bolti ettevõtte koosseisu kuuluvat tõukeliikurit. Sama aasta augustis kahekordistas Bolt suure nõudluse tõttu Tallinna tänavatel laenutatavate elektritõukerataste arvu [2]. Mõne aastaga on muutunud elektritõukeratastega liikumine paljude inimeste igapäeva transpordi osaks. Nimelt 2021. aasta 5. mai seisuga võis leida Tallinna tänavatelt 1000 Bolti tõukeratast ning üle poole tuhande Tuule tõukeliikuri [8]. Paraku ei rahulda kõikide elektritõukeratta entusiastide vajadusi renditavad elektritõukerattad. Näiteks võib Bolti ja Tuule tõukeliikureid parkida ainult kindlaks määratud tsoonidesse, mille eiramine toob kliendile kaasa 5-40 eurose trahvi, sõltuvalt sellest, kui kaugele tsoonist tõukeratas pargitud sai [3]. Paljud kliendid elavad parkimistsoonidest eemal või sõidavad elektritõukerattaga sedavõrd tihti, et laenutamine muutub kulukaks. Bolt ettevõtte renditasud on järgnevad:

Tallinnas

- Tasuta sõidu alustamine;
- 0.15 € iga sõidetud minuti eest.

Tartus

- 0.50 € sõidu alustamine;
- 0.15 € iga sõidetud minuti eest.

Pärnus

- Tasuta sõidu alustamine;
- Esimesed 10 minutit maksavad 0.20 €/min ja iga järgnev sõidetud minut 0.15 € [3].

See tähendab seda, et Pärnus maksab 12 km distantsi läbimine Bolti tõukeratastega 5 eurot, eeldades et klient ei sõida aeglustatud tsoonides, kus on elektritõukeratta rentimise teenust pakkuv firma kiirust piiranud. Sellisel kliendil kuluks koju-tööle marssruudil päevas 10 € ning kuus 200 €. Seetõttu on muutunud populaarseks isiklike tõukeliikurite

soetamine, kui ühe korraliku elektritõukeratta hind on ~600 €. Laenutatavate elektritõukerataste parkimine on muidugi väga mugav – masina võib jätta lubatud tsooni järgmistele klientidele laenutamiseks. Kahjuks on aga isiklike liikurite puhul parkimine ebamugav. Näiteks poodi või tööle minnes tuleb liikur kindlasti kaasa vedada. Vahel soovib inimene lihtsalt linna peal ringi käia ning samal ajal jätta enda elektritõukeratta laadima ning seljakoti seljast heita. Tallinnas eksisteerivad küll mõned tõukeliikurite laadimiseks mõeldud parklad, kuid ükski nendest ei võimalda tõukeratast või mõnda muud elektriliikurit laadida ilmastikukindlalt ning samuti puudub võimalus enda tarbeesemete hoiustamiseks.

Probleem: Eesti linnades puuduvad elektritõukerataste hoiustamiseks ning laadimiseks mõeldud kapid.

Käesoleva lõputöö autor on ise mitu aastat puutunud kokku eelnevalt kirjeldatud probleemiga ning proovib leida sellele lahenduse. Antud lõputöö lähtetingimused hõlmavad endas algteadmist, et mobiiltelefoni eesrakendus luuakse Android mobiilide. Sellest tulenevalt on lõputöö **eesmärgiks** luua Android operatsioonisüsteemiga mobiiltelefonidele elektritõukeratta laadimiskapi eesrakendus, kasutades Xamarin platvormi.

2.1 Eksisteerivad lahendused

2.1.1 Bikeep

Bikeep OÜ on 2012. aastal loodud ettevõte, mis tegutseb peamiselt avalike jalgrataste parklate või parkimiseks mõeldud lukkude tootmisega. Ettevõte on loodud 2012. aastal Meelis H, Kustas K ja Ott R poolt Eestis. Bikeep OÜ ettevõtte peakontorid asuvad Tallinnas ning Ameerika Ühendriikides San Franciscos ning nad on esindatud kokku 12. erinevas riigis üle maailma. 2020. aasta augustis tuli Bikeep välja oma esimese elektritõukerataste laadimiseks mõeldud tootega, mis paigutati Park Inn by Radisson Meriton Conference & Spa Hotel'li juurde Tallinnasse. Bikeep laadimispunktidesse on võimalik parkida enim levinud tootjate elektritõukerattaid [4].

Bikeep elektritõukerataste laadimiseks mõeldud seadmetel on õige mitu puudust:

- Laadida saab ainult enim levinud elektritõukerattaid;
- Laadida ning lukustada saab ainult elektritõukerattaid;
- Laadida pole võimalik ilmastiku kindlalt;
- Laadimise ajaks pole võimalik enda kotti hoiustada;

Laadida saab ainult enim levinud elektritõukerattaid – Bikeep elektritõukerataste laadimisseadmetes saab laadida ainult kindla otsikuga elektritõukerattaid. Kui inimesel on mõni vähem levinud elektritõukeratas, siis selle masina laadimine pole võimalik.

Laadida ning lukustada saab ainult elektritõukerattaid – Bikeep elektritõukerataste laadimisseadmed on ehitatud sedasi, et lukustada on võimalik ainult elektritõukerattaid. Lukustamise mehhanism on järgmine: metallist plaati on tehtud sisselõige ning plaadi äärde on paigutatud riivina töötav metallist pulk. Metallist pulga liigutamisel saab elektritõukeratta liigutada avausse ning uuesti riivi ette tõmmates on tõukeliikur pargitud. Väiksemate elektriliikurite, näiteks elektrirula, elektrilise üksratta ning tasakaaluliikurite puhul pole võimalik tooteid sellesse kinnitamise mehhanismi lukustada. Samuti puuduvad nende toodete laadimiseks vajalikud otsikud [4].

Laadida pole võimalik ilmastiku kindlalt – Bikeep elektritõukerataste laadimisseadmed asuvad tänavatel ning tõukerattad jäävad lukustatuna lihtsalt õue. Vihmasema ilma või palava päikese korral pole võimalik enda tõukeliikurit ilmastikukindlalt laadida. Mõnda elektritõukeratast pole soovitatav vihmase ilmaga õues laadidagi.

Laadimise ajaks pole võimalik enda kotti kuhugi jätta – Bikeep elektritõukerataste laadimisseadmed on disainitud ainult elektritõukerataste laadimiseks, kuid mitte kliendi vara hoiustamiseks.

2.2 Uue lahenduse skoop

Suurimaks puuduseks on täna eksisteerivatel elektritõukerataste laadimisjaamadel võimalus laadida elektritõukerattaid ilmastikukindlalt ning samuti puudub võimalus hoiustada laadimise ajaks enda tarbeesemeid, näiteks kotti. Veel enam jääb vajaka võimalusest parkida teisi elektrilisi sõidukeid, näiteks elektrilist monojalgratast, tasakaaluliikureid ja elektrirulasid.

Tulenevalt eelnevalt nimetatud probleemidest juba eksisteerivatel toodetel, pakub lõputöö autor välja lahenduseks toote, kus saab hoiustada ja laadida ilmastikukindlalt nii elektritõukerattaid kui ka tasakaaluliikureid, elektrirulasid, elektrilisi üksrattaid. Samuti pakub lõputöö autor välja toote disaini. Antud lõputöö lahenduse skoopi jääb Xamarin arendusplatvormil valmis kirjutatud elektritõukerataste laadimiskapi eesrakendus, eesrakenduse disain ning toote enda disain.

Käesoleva lõputöö eesmärgid on järgnevad:

- Disainida laadimiskapp, mis rahuldab enamike elektriliikurite vajadusi;
- Disainida kasutajasõbralik eesrakendus;
- Kirjutada valmis eesrakendus.

3 Loodava eesrakenduse analüüs

3.1 Nõuete määramine

Nõuded põhinevad kasutajalugudel, kus põhirolliks on elektritõukeratta laadimisjaama kasutaja.

3.1.1 Funktsionaalsed nõuded

Kasutaja-põhised funktsionaalsed nõuded:

- Kasutajana soovin registreeruda;
- Kasutajana soovin, et minu e-mail ning parool jäetakse meelde;
- Kasutajana soovin sisse ja välja logida;
- Kasutajana soovin enda profiilil muudatusi teha;
- Kasutajana soovin vajadusel enda mobiiltelefoni numbrit, e-maili või parooli vahetada;
- Kasutajana soovin näha enda parkimiste ajalugu koos parkimise asukohaga, kellaajaga ning maksumusega;
- Kasutajana soovin maksta krediit- või deebetkaardiga;
- Kasutajana soovin uuendada või vahetada kasutatavat makse kaarti;
- Kasutajana soovin näha kui kaua on minu pooleli olev parkimine kestnud ning kui suur on selle maksumus;
- Kasutajana soovin näha kui palju on minu pooleli oleva parkimise jooksul elektrilist sõidukit laetud;
- Kasutajana soovin näha läbipaistvat informatsiooni teenuse maksumuse kohta;
- Kasutajana soovin näha kaardi peal lähimaid elektritõukeratta parkimisjaamu.

3.1.2 Mittefunktsionaalsed nõuded

Kasutaja-põhised mittefunktsionaalsed nõuded:

- Kasutajana soovin, et rakendus oleks lihtne ja loogiline;
- Kasutajana soovin, et rakendus küsiks luba minu seadme positsioneerimiseks;
- Kasutajana soovin, et rakendus ei kulutaks liiga palju mobiiltelefoni akut;
- Kasutajana soovin, et minu andmeid talletatakse turvaliselt.

3.1.3 Tulevikus loodavad funktsionaalsused

Antud lõputöö põhifunktsionaalsuste hulka kuuluvad ainult eesrakenduse arendamisega kaasnevad funktsionaalsused. Tulevikus arendatakse järgnevad funktsionaalsused:

- Andmete salvestamine andmebaasi;
- Maksete teostamine;
- Reaalse parkimise alustamine.

3.2 Tagarakenduse poolsed nõuded

Tagarakenduse poolsed nõuded eesrakendusele on järgnevad:

- Eesrakendus peab suutma suhelda PostgreSQL andmebaasiga;
- Eesrakendus peab aluseks võtma tagarakenduse poolt valmistatud andmebaasi skeemi;
- Eesrakendus peab suutma andmeid pärida API lõpp-punktidest;
- Eesrakendus peab suutma andmeid salvestada, kasutades API lõpp-punkte.

3.3 Tehnoloogia valik

Android operatsioonisüsteemiga mobiiltelefonidele on võimalik arendada väga paljude erinevate tehnoloogiate abil. Antud lõputöö raames vaadeldakse lähemalt enim kasutatavaid tehnoloogiaid [7].

Java oli kunagi kõige enamlevinud programmeerimiskeel Android mobiilide rakenduste arendamiseks. Kuna paljud Android mobiilirakendused on kirjutatud Java programmeerimiskeeles, on Java kasutajatel suur kommuun ning probleemide korral on palju abi olemas internetikeskkondades nagu näiteks Stackoverflow. Java keeles Android mobiilirakenduste kirjutamise kõige suuremaks puuduseks on võimalus hiljem kasutada koodi mugavalt iOS operatsioonisüsteemidega mobiiltelefonide rakenduse arendamisel [7].

Kõige enam on levinud Android platvormipõhiste mobiilirakenduste arendamise puhul programmeerimiskeel Kotlin. Kotlin on Google poolt kuulutatud ametlikuks Androidi aplikatsioonide arendamiseks mõeldud programmeerimiskeeleks 2017. aastal. Kotlin suudab töötada koos Javaga ning jookseb Java virtuaalmasina peal. Suurim erinevus Kotlini ning Java vahel seisneb selles, et Kotlin võtab ära Java keele ebamugavad ja üleliigsed tavapärad nagu näiteks *nullpointer* erindid. Samuti nagu Java keel on mõeldud põhiliselt Android mobiilidele rakenduste kirjutamiseks, täidab sama otstarvet ka Kotlin, kuid Kotlini puhul säilib võimalus Kotlin Multiplaform'i kasutades kirjutada rakendusi nii Android kui ka iOS mobiiltelefonidele [7].

Pisut vähem levinud programmeerimiskeel Android mobiilirakenduste kirjutamiseks on C++. Antud programmeerimiskeelt saab kasutada Androidi rakenduste programmeerimiseks ainult koos NDK'ga ehk Android *Native Development Kit*'iga. C++ programmeerimiskeele boonuseks on see, et kirjutatud programmiosi saab kasutada hiljem ka iOS rakenduste valmistamiseks [7].

C# on süntaksilt Java sarnane programmeerimiskeel, millega polnud pikka aega võimalik väljaspool Windowsi süsteeme midagi programmeerida. See probleem leidis aga lahenduse kui avaldati Xamarin.Android(eelnevalt tuntud kui Mono nime all). Xamarin on platvorm, mille abil on võimalik kirjutada rakendusi nii Android kui ka iOS mobiiltelefonidele [7].

Programmeerimiskeeles Püüton on samuti võimalik kasutada Androidi aplikatsioonide välja arendamiseks kuigi Android ei toeta *native* Püütoni arendust. Sellest murekohast saab eemale põigelda kasutades tööriistu, mis konverteerivad Püütoni aplikatsioonid ümber Androidi pakkideks, mida on juba võimalik jooksutada Android mobiiltelefonides [7].

HTML, CSS ja JavaScript on ka võimalik kasutada Androidi rakenduste arendamiseks. Selle tarbeks tuleb võtta appi Adobe PhoneGap raamistik mis on toetatud Apache Cordova poolt. PhoneGap raamistik lubab kasutada veebipõhiseid arendusoskuseid, et ehitada hübriidrakendusi, mida näidatakse „veebivaatena“ kuid pakendatakse kui aplikatsiooni [7].

Dart on viimaseks laiemalt levinud programmeerimiskeeleks, millega on võimalik arendada Androidi operatsioonisüsteemiga mobiiltelefonidele rakendusi. Dart on vabavaraline programmeerimiskeel, mis toidab Flutter raamistikku. Flutter raamistik on viimasel ajal saanud palju kõlapinda, kuna selle abil on võimalik suure jõudlusega ja hea välimusega rakendusi kirjutada nii veebi, töölauale kui ka mobiiltelefonidele lühikese aja jooksul [7].

Antud lõputöö raames loodav eesrakendus on küll suunatud Android operatsioonisüsteemiga mobiiltelefonidele, kuid lõputöö skoopist välja jäänud edasiste plaanide hulgas on eesmärk teha rakendus kättesaadavaks ka iOS mobiiltelefonidele. Käesoleva lõputöö autoril on endal kõige pikaajalisem kokkupuude Java ning C# programmeerimiskeeltega, kuid puudub suurem kogemus mobiilirakenduste arendamise vallas. Kuna elektritõukerataste laadimiskapi tagarakendus luuakse C# keeles, autoril on soov kasutada juba endale tuttavat programmeerimiskeelt ning loodav eesrakendus võiks olla hiljem kasutatav ka iOS aplikatsiooni loomise juures, on autoril mõistlik valida kasutatavaks tehnoloogiaks C# ja Xamarin.

3.4 Arenduskeskkonna valik

Arenduskeskkonna valikud on jagatud koodihaldus keskkonnaks ning koodi arenduskeskkonnaks.

Järgnevalt võrdlen enimlevinud koodihaldus ning versioonihaldus keskkondi.

Azure DevOps Services on hiljaaegu populaarsust kogunud pilvepõhine keskkond, kus on võimalik kuni 5 liikmelistel tiimidel hoiustada vabavaralisi koode tasuta. Azure Devops koosneb paljudest erinevatest komponentidest. Näiteks on veel lisaks Azure Devops Repositories koodihoidla moodulile ka:

- Azure Pipelines;
- Azure Release Pipelines;
- Azure Boards;
- Azure Test Plans;
- Azure Artifacts.

Azure Boards kujutab endast planeerimise, jälgimise ning tiimisiseste vestluste pidamise moodulit. Azure Pipelines on mõeldud CI/CD *pipeline*'ide jooksutamiseks, kui on soov kasutusele võtta automatiseeritult rakendusi. Azure Release Pipelines on Azure Pipelines'i lisamoodul mis on mõeldud toodangu klastrisse automatiseeritud *release*'ide tegemiseks. Azure Test Plans on mõeldud koodibukettidel uurimuslike testide teostamiseks. Azure Artifacts on moodul, mida kasutatakse pakettide salvestamiseks, näiteks Dockeri konteinerite versioonihalduseks ning ehitus *pipeline*'ide artefaktide hoidmiseks [10].

Bibucket on ettevõtte Atlassian Git'i repositoorium, mis on mõeldud profesionaalsetele tiimidele. Bitbucket annab tsentraalse koha, kust manageerida Git'i repositooriume, koostööd teha lähtekoodiga ning läbi juhatada arenduse voolu. Bitbucketil on järgmised head funktsioonid:

- Ligipääsu haldamise keskkond;
- Töövoo kontrollimise moodul;
- Tõmbamise taotluste süsteem;
- Jira integratsioon;
- Terviklik Rest API [1].

GitHub on koodi võõrustamise platvorm, mis hõlmab endas koodihalduse funktsionaalsusi. GitHub annab võimaluse töötada sama koodi kallal inimestel üle terve maailma. GitHub on kõigest koodihoidla, see tähendab seda, et ta ei oma endas teisi funktsionaalsusi nagu Bitbucketil ning Azure Devopsil. Tasuta versiooni juures tuleb hoida kood vabavarana ning avalikuna. GitHub omab väga head integreeritust teiste platvormidega [12].

GitLab on sarnaselt GitHubile väga levinud koodihaldus ning talletamise platvorm. GitLab on avatud lähtekoodiga projekt, millel on üle 3000 toetaja. Samuti nagu GitHubi puhul, tuleb GitLabis hoida koodi avatuna ning avalikuna, et osa saada tasuta versiooni võimalustest [13].

Antud lõputöö autor on oma erialase karjääri jooksul kõige enam kokku puutunud Bitbucketi ning Azure Devopsi võimalustega. Kuna esialgu pole autoril plaanis hakata arendatavat rakendust pilves talletama, sest rakendus on alles oma ehitamise järgus ning esialgsete plaanide juurde ei kuulu rakenduse kasutusele võtmine *pipeline*'dega. Seetõttu on autoril kõige mõistlikum kasutada Bitbucketi versioonihaldust ning koodihoidlat.

Koodiarenduskeskkonna valimisel tuleb lähtuda sellest, et arenduskeskkond toetaks nii C# arendust kui ka Xamarin platvormi. Tänasel päeval on turul kaks suuremat sellist koodiarenduskeskkonda – JetBrains Rider ning Visual Studio. JetBrains Rideri puhul tuleb välja tuua, et tasuta versioon omab umbes sama palju funktsionaalsusi kui Visual Studio, kuid toote litsents maksab vähem. Käesoleva lõputöö autor on rohkem kokku puutunud C# programmeerimiskeele arendamise käigus Visual Studioga. Sellest tulenevalt otsustas käesoleva lõputöö autor võtta arenduskeskkonnaks Visual Studio [6].

3.5 Arhitektuur

Eesrakenduse arhitektuur koosneb eesrakendusest endast, mis tuleb laadida igal kliendil enda Android mobiiltelefoni. Esialgne lahendus on tehtud nii, et andmebaas on lokaalne, kuna projekt on veel prototüübi faasis ning suurt kasutajate voogu rakendust kasutamas pole.

3.6 Disain

Disaini peatüki võib jagada kaheks erinevaks peatükiks:

- Toote disain;
- Eesrakenduse disain.

3.6.1 Toote disain

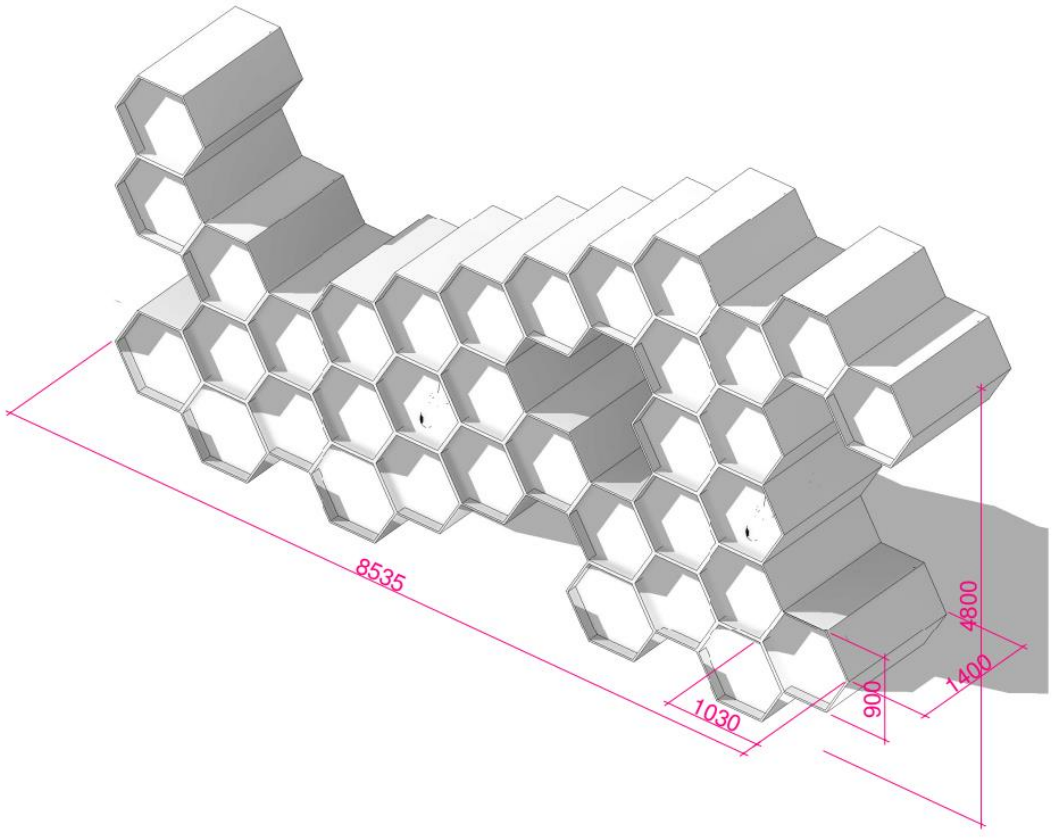
Elektritõukeratastele mõeldud laadimisjaama disaini alustaladeks on võetud järgmised printsiibid:

- Elektritõukerattaid peab olema võimalik laadida ilmastikukindlalt;
- Elektritõukeratta laadimisjaama peaks olema võimalik jätta hoiule klientide tarbeesemeid, näiteks kotte;
- Elektritõukerataste laadimiskeskus peab võimaldama laadida erinevate tootjate elektriliikureid;
- Elektritõukerataste laadimiskeskus peab võimaldama laadida lisaks elektritõukeratastele ka teisi elektriliikureid;
- Elektritõukeratta laadimisjaam peaks olema skaleeritav ehk võiks koosneda juurde lisatavatest moodulitest;
- Elektritõukeratta laadimisjaam peab nägema hea välja ning võiks anda linnaruumile midagi juurde.

Esimese tingimuse rahuldamiseks tuleb teha toote disainis mööndus, et elektriliikurit ilmastikukindlalt laadida saaks. Mööndus kujutab endast seda, et iga elektriliikur peab asuma seintega ääristatud kapslis ning olema täielikult isoleeritud teistest.

Joonis 1. Elektritõukeratta laadimisjaama disain, vaade 1

Teise tingimuse rahuldamiseks tuleb elektritõukeratta laadimiskeskuse disaini lisada iga kapsli juurde lisaruum, et mahuks ära ka keskmine kott.



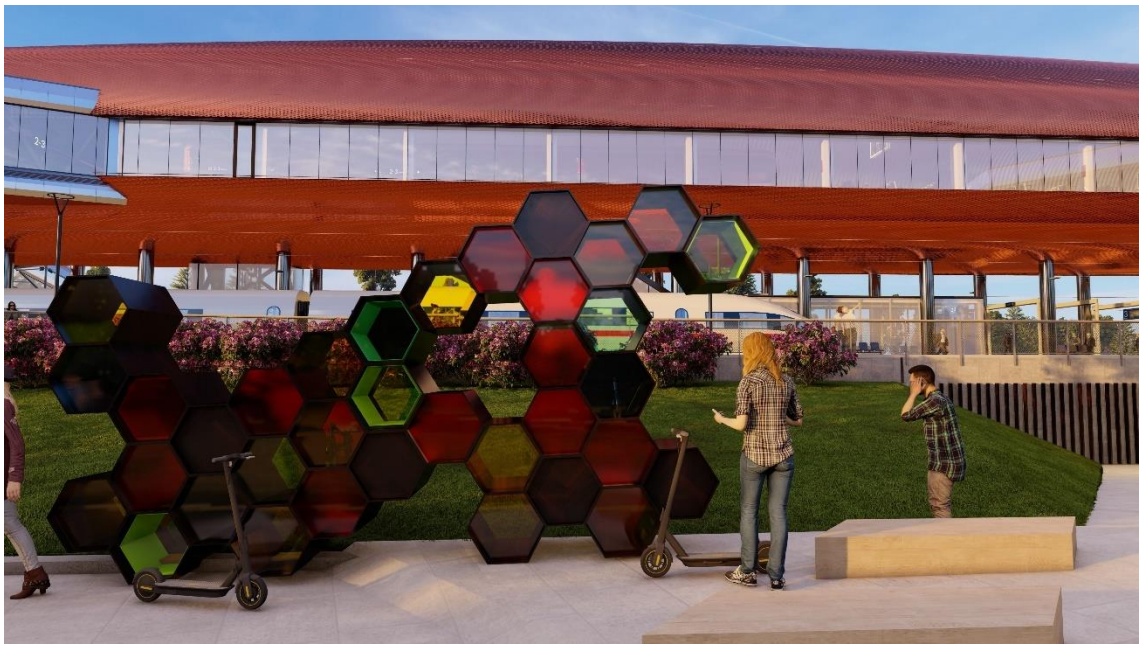
Joonis 2. Elektritõukeratta laadimisjaama disain mõõtudega

Kolmanda tingimuse rahuldamiseks tuleb iga elektritõukeratta laadimiskeskuse kapsel varustada erinevate populaarseimate kui ka vähem populaarsemate elektritõukerataste otsikutega. Variant B oleks igasse kapslisse jätta 20V otsik, kuhu saab iga inimene panna vajadusel isikliku laadija.

Joonis 3. Elektritõukeratta laadimisjaam koos laaditavate seadmetega

Joonis 4. Elektritõukeratta laadimisjaam koos laaditavate seadmetega lähivõte

Neljanda tingimuse rahuldamiseks tuleb elektritõukeratta laadimiskeskus disainida nii, et sinna mahuvad parkima ning laadima elektrirulad, elektrilised üksrattad, tasakaaluliikurid ning elektritõukerattad.



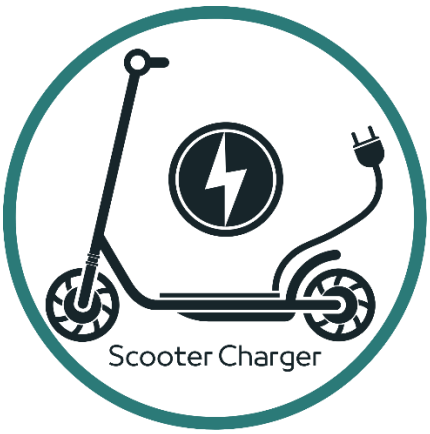
Joonis 5. Elektritõukeratta laadimisjaama disain, vaade 2

Viienda ning kuuenda tingimuse täitmine on üksteisega tihedalt seotud. Autor mõtles ning katsetas pikalt toote idee kaasautoriga, milline võiks olla antud produkti kontseptsioon ning lõpptulemus on järgmine. Saadud toote disain on skaleeritav, lisades uusi kärje mooduleid juurde.

Joonis 6. Elektritõukeratta laadimisjaama disain, vaade 3

Loodava lõputöö eesrakenduse tarbeks disainis autor mobiilirakendusele logo, mille loomisel lähtus autor *less is more* printsiibist, ehk parem on viia fookus ainult hädavajalikule kui kasutada liiga palju üleliigset. Logo on disainitud võimalikult

minimalistlikult ning eesmärk on logo kaudu edasi anda arusaadav mõte, et tegu on elektriliste tõukerataste laadimiseks mõeldud tootega.



Joonis 7. Mobiilirakenduse logo

Täiendavaid elektritõukeratta laadimisjaama visuaalse kontseptsiooni näidiseid võib leida lisast (Lisa 2).

3.6.2 Eesrakenduse prototüübi disain Axure tarkvara baasil

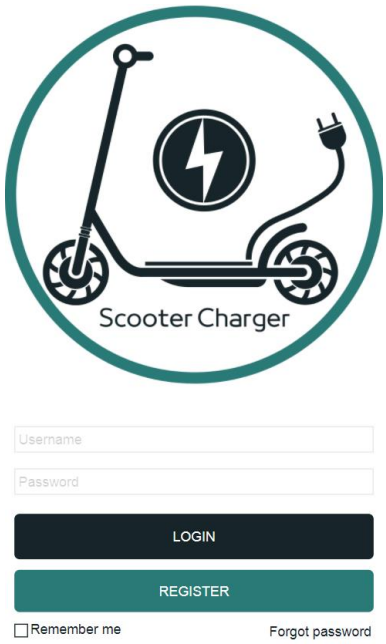
Eesrakenduse disaini loomisel on võetud eelkõige arvesse kasutajalugudest saadud informatsioon. Esimene loogiline samm mobiilirakenduse loomisel on kasutajakogemuse disaini programme kasutades esmase prototüübi loomine, kuna näiteks Xamarin platvormi kasutades on sellise prototüübi valmistamine keerukas ning aeganõudev.

Axure on UX Prototüüpide, spetsifikatsioonide ning diagrammide ehitamise tööriist. Axure RP on üks vähestest UX disainimise tööriistadest, mis annab võimaluse realistlike ning funktsioneerivate prototüüpide ehitamiseks.

Prototüübi loomise eesmärgiks on katsetada kasutajalugudest välja joonistuva eesrakenduse funktsionaalsust. Prototüübi valmistamine kasutajalugude programmides aitab kokku hoida vale arendusega kaasnevaid kulusid. Vale arenduse all peab autor silmas seda, et Xamarin platvormil tehtud UX disaini muudatusi on palju ajamahukam teostada kui Axure RP tarkvaras tehtud muudatusi.

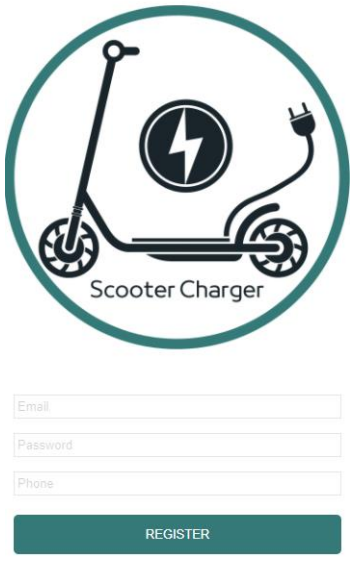
Elektritõukeratta laadimisjaama mobiilirakenduse avavaade on kujutatud Joonis 8. Avavaate juurde kuulub kasutaja võimalus sisse logida või registreeruda. Samuti on lisatud võimalus jätta kasutaja parool ning e-mail meelde. Kasutajatel, kes on unustanud parooli, on võimalus parool ära vahetada, vajutades nupule „*Forgot Password*“. Antud disaini puhul lähtus autor mõttest, et lihtsuses peitub võlu ning tähtis on kuvada klientide

jaoks olulisi kasutajalugusid, mis on sisselogimine, registreerimine, kasutajatunnuste meelde jätmine ning vajadusel parooli meelde tuletamine või vahetamine.



Joonis 8. Mobiilirakenduse sisse logimise leht

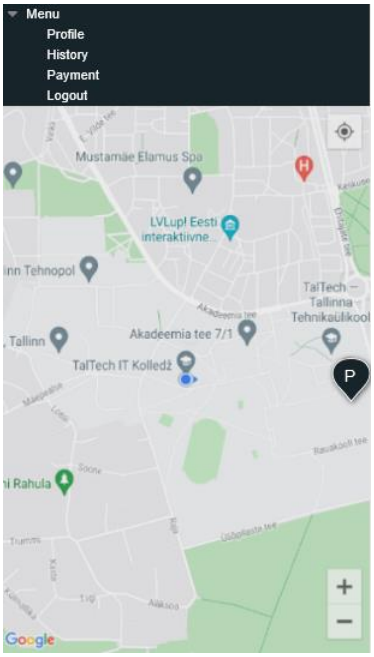
Kasutaja registreerimise vaade on kujutatud Joonis 9. Kasutajal palutakse sisestada enda e-maili aadress, parool ning mobiiltelefoni number. Vajutades nuppu Register, registreeritakse kasutaja ning lubatakse ta kodulehele, kus on kasutaja juba sisse logitud.



Joonis 9. Mobiilirakenduse registreerimise leht

Kasutaja sisse logimise järgne põhivaade on kujutatud Joonis 10. Põhivaatel kuvatakse Google Mapsi kaarti, millel on ära toodud kasutaja asukoht ning lähimad elektritõukeratta

laadimisjaamad kasutajale. Kaardi täisfunktsionaalseks töötamiseks peab kasutaja andma loa rakendusel tema asukohta jälgimiseks. Kui kasutaja vajutab mõne elektritõukeratta laadimisjaama nööpnõela peale, avaneb hüpikaken ning kasutaja saab soovi korral alustada parkimise ning laadimisega. Kui kasutaja tahab navigeerida mõnele muule lehele, tuleb vajutada kolme joonega nupu peale vasemal üleval servas, kust on võimalik kasutajal valida järgnevate lehekülgede vahel – profiil, ajalugu, maksmise meetodid.

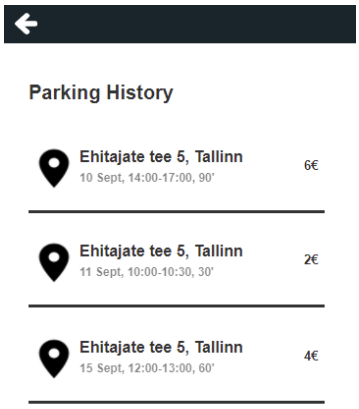


Joonis 10. Mobiilirakenduse koduleheküg

Kasutaja profiili vaade on kujutatud Joonis 11. Kasutajal on võimalik soovi korral vahetada või uuendada enda parooli, e-maili aadressi või mobiiltelefoni numbrit. Kasutajal on ka võimalus vahetada enda profiilipilti, kui ta vajutab enda nime kõrval oleva halli ikooni peale. Antud kasutaja on juba oma profiilipilti vahetanud.

Joonis 11. Mobiilirakenduse kasutaja profiili leht

Kasutaja ajaloo vaade on kujutatud Joonis 12. Kasutajal on võimalik vaadata eelnevalt sooritatud parkimisi ning laadimisi. Iga laadimine tekitab uue rea, kuhu on üles märgitud kasutaja poolt valitud elektritõukeratta laadimisjaama nimi, asukoht ning parkimise maksumus.



Joonis 12. Mobiilirakenduse parkimiste ajaloo leht

Kasutaja pangakaartide lehekülje vaade on kujutatud Joonis 13. Pangakaartide lehel saab kasutaja valida milline maksevahend on tal parasjagu aktiivne ning kasutajal on võimalik omada mitut erinevat maksevahendit. Aktiivset maksevahendit on kasutajal võimalik valida tehes linnukese soovitud kaardi taha. Samuti on kasutajal võimalik lisada uusi maksevahendeid, kui peaks selleks vajadus tekkima ning maksevahendi peale klikkides on võimalik kaart ära kustutada.

Joonis 13. Mobiilirakenduse maksmise meetodite leht

3.7 Analüüsi kokkuvõte

Antud lõputöö analüüsi osas eritles autor erinevaid tehnoloogiaid, arendustarkvarasid, elektritõukeratta laadimisjaama potentsiaalsete klientide kasutajalugusid ning kasutajate funktsionaalseid ning mittefunktsionaalseid nõudeid.

Tehnoloogia valiku osas otsustas autor kasutada C# ja Xamarin platvormi. Valikut põhjendavad kolm järgmist aspekti. Esiteks on lõputöö raames loodav eesrakendus mõeldud küll lõputöö skoopis ainult Android mobiiltelefonidele, kuid väljaspool lõputöö skoopi on soov rakendus teha kättesaadavaks ka iOS mobiiltelefonidele – C# ja Xamarin pakuvad seda võimekust. Teine põhjus seisneb selles, et käesoleva lõputöö autoril on varasem kokkupuude C# programmeerimiskeelega. Kolmas põhjus peitub selles, et antud eesrakenduse tagarakendus kirjutatakse C# keeles ning nii on lihtsam kahte rakendust omavahel suhtlema panna.

Arenduskeskkonna osas langes otsus Visual Studio kasuks, sest lõputöö autoril on varasem suurem kogemus antud arenduskeskkonnaga ning Xamarini integratsioon on antud keskkonnas väga mugav. Antud lõputöö koodihoidlaks valis autor Bitbucketi enda eelneva erialase kogemuse pärast.

4 Programmi arendus

Programmi arenduse peatükk on jagatud olulisemate vaadete vahel peatükkideks, kus annab autor ülevaate, milline näeb välja kindla vaate jaoks kirjutatud rakenduse kood ning milline on visuaalne tulemus. Xamarini failides on vaadete jaoks mõeldud .xaml faililaiendiga failid ning iga Xamarin-Formsi tüüpi .xaml faililaiendiga failile on juurde genereeritud automaatselt samanimeline .xaml.cs tüüpi fail, kuhu on võimalik kirjutada antud lehe spetsiifilist C# koodi. Näiteks, mis juhtub siis kui vajutada kindlat nuppu.

4.1 Sissejuhatus peatükki

Käesoleva lõputöö autoril puudus enne projekti alustamist suurem kokkupuude mobiiltelefonide rakenduste kirjutamisega ning kokkupuude Xamariniga puudus täielikult. Parema arusaama saavutamiseks, läbis lõputöö autor kaks erinevat veebipõhist kursust Xamarini kohta 60 tunni mahus:

- The Complete Xamarin Developer Course: iOS And Android! (Udemy);
- Xamarin Forms: Build Native Mobile Apps with C# (Mosh Hamedani).

Kursuste käigus omandas autor põhitõed Xamarini kohta ning lahendas ära juhendajate poolt püstitatud ülesanded.

Eesrakenduse arendamisel viis autor visuaalses pooles rõhu lihtsusele ning kasutaja mugavusele. Samuti proovis autor hoida platvormipõhise koodi kirjutamist pigem väiksena. See tähendab seda, et eesmärk on võimalikult palju koodi kirjutada *cross-platform* projektis, kust kompileeritakse kood nii Android kui ka iOS operatsioonisüsteemiga mobiiltelefonidele, samadest projekti failidest.

4.2 Eesrakenduse visuaalne disain

Eesrakenduse visuaalse disaini osas võeti eeskujuks eesrakenduse prototüüp, milles tehti mõningased muudatused värvide ning nuppude paigutuste osas (vaata peatükis 3.6.2).

4.3 Eesrakenduse avaleht

Eesrakenduse avalehe disaini loomist tuleb alustada *App.xaml* failist, kus tuleb *App* meetodis defineerida rakenduse käivitamisel avatav lehekülg. Järgnevas koodikirjelduses on näha, kuidas defineerida ilma andmebaasita avaleheks sisse logimise leht.

```
public App()
{
    InitializeComponent();
    MainPage = new LoginPage();
}
```

Koodinäide 1. Mobiilirakenduse põhilehe defineerimine andmebaasita

Järgnevas näites on näha, kuidas defineerida avaleht koos andmebaasiga. Andmebaas on vajalik kasutajate registreerimiseks.

```
public static string DatabaseLocation = string.Empty;

public App(string databaseLocation)
{
    InitializeComponent();
    MainPage = new LoginPage();
    DatabaseLocation = databaseLocation;
}
```

Koodinäide 2. Mobiilirakenduse põhilehe defineerimine andmebaasiga

Koodirida `MainPage = new LoginPage();` omistab rakenduse avaleheks LoginPage'i.

Avalehe tüübiks valis autor *ContentPage* tüüpi lehekülje, mis annab võimaluse erinevate teiste moodulite lisamiseks.

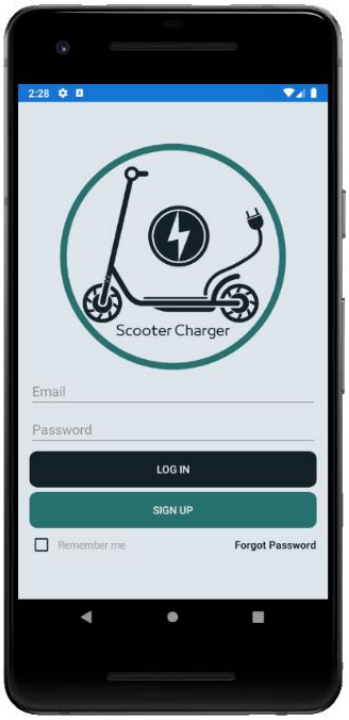
Järgnev koodinäide on avalehe *.xaml* failist, kus defineeritakse erinevaid objekte.

```
<StackLayout BackgroundColor="AliceBlue"
             VerticalOptions="Center"
             Margin="15">
        <Image x:Name="iconImage"
               Aspect="AspectFit"
               HorizontalOptions="FillAndExpand"
               HeightRequest="300"
               Source="logo3.png"/>
        <Entry x:Name="emailEntry"
               Placeholder="Email"
               Keyboard="Email"/>
        <Entry x:Name="passwordEntry"
               Placeholder="Password"
               IsPassword="True"/>
        <Button x:Name="loginButton"
                Text="Log In"
                BackgroundColor="#17252A"
                TextColor="{StaticResource buttonTextColor}"
```

```
                Style="{StaticResource normalButton}"
                Clicked="loginButton_Clicked"
                CornerRadius="10"/>
        <Button x:Name="signupButton"
                Text="Sign Up"
                BackgroundColor="#2B7A78"
                TextColor="{StaticResource buttonTextColor}"
                Style="{StaticResource normalButton}"
                Clicked="signupButton_Clicked"
                CornerRadius="10"/>
</StackLayout>
```

Koodinäide 3. Mobiilirakenduse põhilehe defineerimine .xaml failis

Eelnev koodinäide annab järgneva visuaalse tulemuse:



Joonis 14. Mobiilirakenduse sisselogimise leht

Iga *ContentPage* tüüpi leheküljel on autoril mõistlik defineerida veerutüübiks *StackLayout*, millele omistatakse osa ekraani pinnast, millele saab anda soovitud asukoha, suuruse, värvi või marginaali. Oluline on tähele panna seda, et igal *ContentPage* tüüpi lehel on võimalik defineerida ainult üks põhiline paigutusmuster. *StackLayout* on kõige mõistlikum valik, kuna sinna on võimalik defineerida teisi *StackLayout*'e või muid kuvamisviise. *StackLayout*'iga on mugav objekte telefoni ekraanile paigutada. Esimesena on defineeritud eesrakenduse sisselogimislehe logo3.png. Antud pildi parameetriteks on kaasa antud sellised väärtused, et pilt hõivaks horisontaalselt terve ekraani laiuse ning vertikaalselt on pildi maksimum suuruseks 300 pikslit. Järgmisena on autor defineeritud kaks sisend välja sisselogimiseks – e-maili ja parooli välja. Mõlemale sisend väljale on

antud *x:Name* parameeter, mida kasutatakse hiljem andmete sidumiseks andmebaasiga. Oluline on täheldada, et sisend väljadele on ka autor märkinud, mis tüüpi informatsiooni sisendina oodatakse. Näiteks on määratud e-maili väljale klaviatuuri tüübiks „e-mail“, mis avab kasutajal puudutuse järel klaviatuurivaate, mille abil kuvatakse kasutajale klaviatuuri vasakusse alumisse nurka „@“ sümbol. Sisselogimise väljale on autor lisanud parameetri *IsPassword* väärtuseks tõene, mis tähendab seda, et kui kasutaja vajutab väljale ning alustab kirjutamist, siis kuvatakse kirjutatud teksti tärnidena, kuna tegemist on parooliga. Kahe sisend välja alla on autor lisanud kaks nuppu – Sisselogimine ning kasutaja registreerimine. Mõlemale väljale on autor defineerinud ka taustavärvi ning teksti värvi. Xamarin platvorm võimaldab värve defineerida mitmel moel. Esimene variant, mida võib näha tasutavärvi välja juures (*BackgroundColor* parameeter), on defineerida värv RGB koodina. Teine variant värvide defineerimiseks, teostatakse *App.xaml* faili abil, kuhu on võimalik defineerida parameetrina kindel värv, mida on hiljem võimalik läbivalt projekti *.xaml* failides kasutada. Sellist lähenemist on kasutatud *TextColor* parameetri juures. Koodinäide:

```
<Application.Resources>
    <ResourceDictionary>
        <Color x:Key="buttonColor">#2B7A78</Color>
        <Color x:Key="buttonTextColor">#FEFFFF</Color>
        <Style x:Key="normalButton" TargetType="Button">
            <Setter Property="BackgroundColor"
                    Value="{StaticResource buttonColor}"/>
            <Setter Property="TextColor"
                    Value="{StaticResource buttonTextColor}"/>
            </Style>
    <ResourceDictionary>
<Application.Resources>
```

Koodinäide 4. Mobiilirakenduse värvide defineerimise viisid

Autor on lisanud nupule *CornerRadius* parameetriga raadiuse, mis muudab nupu servad kumeraks. Viimase olulise parameetrina on autor defineerinud Koodinäide 3 *Clicked* parameetri. *Clicked* parameetriga määratakse, mida peab rakendus tegema nupu vajutamise korral. Väärtus, mis omistatakse *Clicked* parameetrile, tekitab samanimelise tühja meetodi lehekülje *.xaml.cs* laiendiga faili. Koodinäide *LoginPage.xaml.cs* failist ning parameetriga `Clicked="loginButton_Clicked“` seotud funktsioonist:

```
async void loginButton_Clicked(object sender, EventArgs e)
{
    bool isEmailEmpty = string.IsNullOrEmpty(emailEntry.Text);
    bool isPasswordEmpty = string.IsNullOrEmpty(passwordEntry.Text);

    if (isEmailEmpty || isPasswordEmpty)
```

```
    {
        await DisplayAlert("Failed", "Username or Password are incorrect", "Ok");
    }
    else
    {
        bool result = await Auth.LoginUser(emailEntry.Text, passwordEntry.Text);
        if (result)
            App.Current.MainPage = new HomePageMasterDetail();
    }
}
```

Koodinäide 5. Mobiilirakenduse sisselogimise nupule vajutamise näide

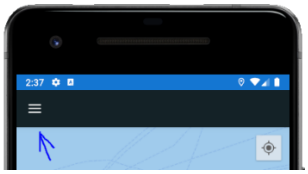
Antud funktsioon võtab sisse kaks argumenti: *object sender* ning *EventArgs e*. Kui kasutaja e-maili sisendi väli või parooli väli on tühjad, kuvatakse kasutajale hoiatus, et e-maili väli või parooli väli on täitmata. Juhul kui väljad on täidetud, proovitakse autentida kasutaja *Auth* klassi funktsiooniga *LoginUser*. Koodinäide:

```
public static async Task<bool> LoginUser(string email, string password)
{
    try
    {
        return await auth.LoginUser(email, password);
    }
    catch(Exception ex)
    {
        await App.Current.MainPage.DisplayAlert("Error", ex.Message, "Ok");
        string registerMessage = "There is no user record corresponding to this
identifier";
        if (ex.Message.Contains(registerMessage))
            return await RegisterUser(email, password);
        return false;
    }
}
```

Koodinäide 6. Mobiilirakenduse sisselogimise meetod

4.4 Eesrakenduse koduleht

Kõige keerulisem osa eesrakenduse koodist peitus autori jaoks just kodulehe kirjutamises. Kodulehe paremas ülevas nurgas asub kolme triibuga navigatsiooniriba või menüü paneel.



Joonis 15. Mobiilirakenduse navigeerimise/menüü nupp

Sellist tüüpi navigatsiooni saamiseks tuleb defineerida lehekülg *Shell* tüübina. Seda tuleb teostada lehe *.xaml.cs* failis ehk *Code Behind*-is. Koodinäide:

```
public partial class HomePageMasterDetail : Shell
{

}
```

Koodinäide 7. Mobiilirakenduse lehekülje tüübi defineerimine

Seejärel tuleb defineerida lehekülje *.xaml* failis lehekülg *Shell* tüübina. Koodinäide:

```
 <Shell xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="ScooterChargerMax.HomePageMasterDetail"
             xmlns:local="clr-namespace:ScooterChargerMax">
</Shell>
```

Koodinäide 8. Mobiilirakenduse lehekülje tüübi defineermine .xaml failis

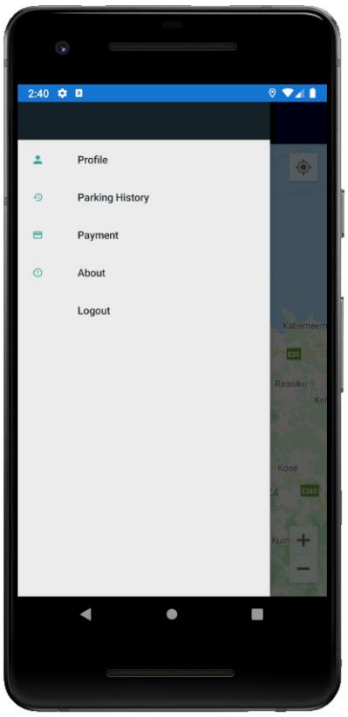
Selleks, et navigatsiooni nupule vajutades menüü kirjed ilmuks, tuli autoril lisada koduleheküljе .xaml faili järgmised read:

```
<MenuItem Text="Profile"
          StyleClass="MenuItemLayoutStyle"
          Icon="ic_person.png"
          Clicked="ProfileMenuItemClicked"
          FlyoutItem.IsVisible="False"/>
<MenuItem Text="Parking History"
          StyleClass="MenuItemLayoutStyle"
          Icon="ic_history.png"
          Clicked="HistoryMenuItemClicked"
          FlyoutItem.IsVisible="False"/>
<MenuItem Text="Payment"
          StyleClass="MenuItemLayoutStyle"
          Icon="ic_credit_card.png"
          Clicked="PaymentMenuItemClicked"
          FlyoutItem.IsVisible="False"/>
<MenuItem Text="About"
          StyleClass="MenuItemLayoutStyle"
          Icon="ic_error_outline.png"
          Clicked="AboutMenuItemClicked"
          FlyoutItem.IsVisible="False"/>
<MenuItem Text="Logout"
          StyleClass="MenuItemLayoutStyle"
          Clicked="LogoutMenuItemClicked"
          FlyoutItem.IsVisible="False"/>
```

Koodinäide 9. Mobiilirakenduse menüü väljade defineerimine .xaml failis

Autor märkis iga menüü parameetrite hulka *FlyoutItem.IsVisible* väärtuseks väär, sest nii avaneb menüü ribalt valitud lehekülg ilma navigatsiooni nuputa. Vastasel korral jääb suur väljalennu riba vasakule küljele avatuna.

Saadud tulemus näeb välja järgmine:



Joonis 16. Mobiilirakenduse avatud menüüriba

Samuti on lisatud iga menüü kirje juurde *Clicked* parameeter, mis *Code-Behind*'is kutsub esile näiteks „*Profile*" menüükirjele vajutades, sellise programmilõigu:

```
private void ProfileMenuItemClicked(object sender, EventArgs e)
{
    Shell.Current.GoToAsync($"{nameof(ProfilePage)}");
    Shell.Current.FlyoutIsPresented = false;
}
```

Koodinäide 10. Mobiilirakenduse „*Profile*" nupule vajutamisel täidetud funktsioon

Vastav programmilõik suunab kasutaja valitud lehele, antud näite kontekstis suunatakse kasutaja profiili leheküljele. Selleks, et kasutajaid saaks üldse teistele lehtedele marsruutida tuleb defineerida kodulehe *.xaml.cs* klassi failis marsruutimine. Koodinäide:

```
public HomePageMasterDetail()
{
InitializeComponent();

Routing.RegisterRoute(nameof(ProfilePage), typeof(ProfilePage));
Routing.RegisterRoute(nameof(HistoryPage), typeof(HistoryPage));
Routing.RegisterRoute(nameof(PaymentPage), typeof(PaymentPage));
Routing.RegisterRoute(nameof(AboutPage), typeof(AboutPage));
}
```

Koodinäide 11. Mobiilirakenduse navigeerimise/menüü kirjete marsruutimine

Ainus lehekülg, mida pole autor marsruutimisse lisanud, kuid mida võib leida kodu leheküljelt, on välja logimise leht. Selle menüü kirje peale vajutades tuleb kasutaja välja logida ning autor ei taha, et kasutaja sisselogimise lehele satuks sisselogitult. Seepärast pole marsruutimist tehtud. Koodinäide:

```
private void LogoutMenuItemClicked(object sender, EventArgs e)
{
    App.Current.MainPage = new LoginPage();
}
```

Koodinäide 12. Mobiilirakenduse marsruutimine väljalogimise nupule vajutades

Koduleheküljelt võib leida Google Maps'i kaardi, millel on kuvatud kasutaja ning kasutaja lähimad parkimisjaamad. Tagataustal kuvab koduleht väljalennu kirjena teist lehekülge, milleks on kaardi leht. Koodinäide:

```
<FlyoutItem Title="Map" Icon="ic_action_map.png"
            FlyoutDisplayOptions="AsMultipleItems">
    <ShellContent Route="Home"
                  ContentTemplate="{DataTemplate local:MapPage}"/>
</FlyoutItem>
```

Koodinäide 13. Mobiilirakenduse lehekülgede kuvamine pesastatud kujul

4.5 Eesrakenduse kaardi leht

Kaardi lehekülg lisatakse pealehe *.xaml* failile *ContentTemplate* tüübina, *ShellContent* veeru tüübi sees. Selleks tuleb *ContentTemplate* tüübi sees kasutada *DataTemplate* andmete kuvamise funktsionaalsust, mille abil öeldakse programmile, et põhilehe taustal tuleb kuvada kaardi lehekülge. Kaardi lehekülje lisamine valmistas autorile kõige enam probleeme terve projekti arendamise jooksul. Kaardi rakenduse tööle saamiseks tuleb esmalt lisada projekti *Xamarin.Forms.Maps* ja *Xam.Plugin.Geolocator* paketid. Seejärel tuleb teha Google Cloud Platformi APIs & Services uus projekt ning lisada uus API võti. Uue võtme turvaliseks tegemiseks tuleb jooksutada järgnev käsklus: `keytool -list -v -keystore /Users/user_name/.local/share/Xamarin/Mono\ for\ Android/debug.keystore -alias androiddebugkey -storepass Your_Password -keypass Your_Password` ning lisada saadud näpujälg volikirja. Järgmise sammuna pidi autor tegema muudatused *AndroidManifest.xml* faili, et kasutajate asukohta positsioneerida oleks võimalik ning selleks, et üldse kasutajatele kaarti kuvada saaks. Iga kasutaja peab rakenduse esimesel käivitamisel andma loa enda positsioneerimiseks, kui nad soovivad rakendust täisväärtuslikult kasutada. Näide *AndroidManifest.xml* failist. Tähelepanu väärib, et

API_KEY parameeter on muudetud ümber pseudoväärtuseks, sest autor plaanib vastavat võtit kasutada ka tulevikus:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
android:versionCode="1" android:versionName="1.0"
package="com.companyname.scooterchargermax" android:installLocation="auto">
	<uses-sdk android:minSdkVersion="21" android:targetSdkVersion="30" />
	<application android:label="ScooterChargerMax.Android"
android:theme="@style/MainTheme">
		<meta-data android:name="com.google.android.geo.API_KEY"
android:value="API_KEY_VALUE"/>
		<meta-data android:name="com.google.android.gms.version"
android:value="@integer/google_play_services_version"/>
		<uses-library android:name="org.apache.http.legacy"
android:required="false"/>
	</application>
	<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
	<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"
/>
	<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
	<uses-permission android:name="android.permission.INTERNET" />
	<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"
/>
</manifest>
```

Koodinäide 14. Mobiilirakenduse Android.Manifest.xml fail

Järgnevas koodinäites näitab autor milline näeb välja kaardi lehekülg. Kaardi kuvamiseks vajalik *.xaml* fail on iseenesest üsna triviaalne. Tuleb ainult kasutada Google Mapsi paketi moodulit, mis on NuGet paketina lisatud. Koodinäide:

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             xmlns:maps="clr-
namespace:Xamarin.Forms.Maps;assembly=Xamarin.Forms.Maps"
             x:Class="ScooterChargerMax.MapPage">

    <maps:Map x:Name="locationsMap"
              IsShowingUser="False"
              HorizontalOptions="FillAndExpand"
              VerticalOptions="FillAndExpand"
              MapType="Street"/>
</ContentPage>
```

Koodinäide 15. Mobiilirakenduse kaardilehekülje .xaml fail

Kaardi lehekülje *Code-Behind* ehk *.xaml.cs* fail osutus autori jaoks kõige keerukamaks ning mahukamaks ülesandeks – kaardi kuvamise loogika kirjutamine. Lehekülje avanemise korral kutsutakse *OnAppearing* meetodis välja kasutaja asukoha leidmise meetod, mille abil kuvatakse kasutaja asukoht kaardile. Seejärel kutsutakse välja *DisplayOnMap* meetod, mille abil kuvatakse kaardil kõik parkimisjaamad. Antud hetkel on tagarakenduse puudumise tõttu *DisplayOnMap* funktsiooni lisatud käsitsi asukoht, kus võiks potentsiaalselt asuda tulevikus üks parkimisjaam. Viimasena kutsutakse välja

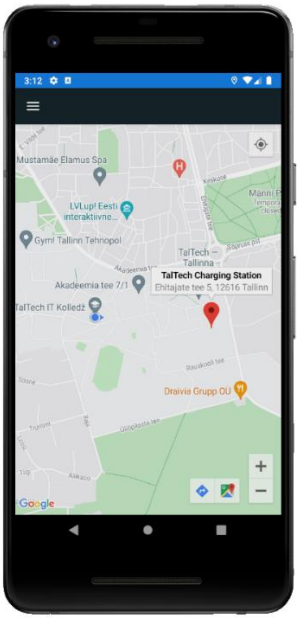
GetLocation funktsioon, mille abil saadakse teada kasutaja asukoht, ainult juhul kui kasutaja on andnud programmile õigused enda positsioneerimiseks.

```
[XamlCompilation(XamlCompilationOptions.Compile)]
public partial class MapPage : ContentPage
{
    IGeolocator locator = CrossGeolocator.Current;

public MapPage()
{
    InitializeComponent();
}

protected override void OnAppearing()
{
    base.OnAppearing();

    GetLocation();

    DisplayOnMap();
}

protected override void OnDisappearing()
{
    base.OnDisappearing();

    locator.StopListeningAsync();
}

private async void GetLocation()
{
    var status = await CheckAndRequestLocationPermission();

    if(status == PermissionStatus.Granted)
    {
        var location = await Geolocation.GetLocationAsync();

        locator.PositionChanged += Locator_PositionChanged;
        await locator.StartListeningAsync(new TimeSpan(0, 1, 0), 100);

        locationsMap.IsShowingUser = true;

        CenterMap(location.Latitude, location.Longitude);
    }
}


private void Locator_PositionChanged(object sender,
Plugin.Geolocator.Abstractions.PositionEventArgs e)
{
    CenterMap(e.Position.Latitude, e.Position.Longitude);
}

private void CenterMap(double latitude, double longitude)
{
    Xamarin.Forms.Maps.Position center = new
Xamarin.Forms.Maps.Position(latitude, longitude);
            MapSpan span = new MapSpan(center, 1, 1);

            locationsMap.MoveToRegion(span);
}
```

```
private async Task<PermissionStatus> CheckAndRequestLocationPermission()
{
    var status = await
Permissions.CheckStatusAsync<Permissions.LocationWhenInUse>();

    if (status == PermissionStatus.Granted)
        return status;
    status = await Permissions.RequestAsync<Permissions.LocationWhenInUse>();
    return status;
}
private void DisplayOnMap()
{
    var position = new Xamarin.Forms.Maps.Position(59.395, 24.6719);
    var pin = new Pin()
    {
        Type = PinType.Place,
        Position = position,
        Label = "Tallinn University of Technology",
        Address = "Ehitajate tee 5, 12616 Tallinn"
    };
}
```

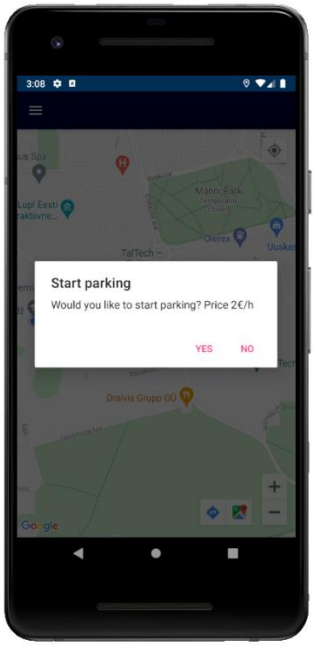
Koodinäide 16. Mobiilirakenduse kaardilehekülje *Code-Behind*

Saadud rakenduse kodulehe visuaalne tulemus on järgnev:



Joonis 17. Mobiilirakenduse kaardirakenduse vaade

Kui kasutaja vajutab laadimisjaama nööpnõela nupu peale, avaneb hüpikaken, kus küsitakse kasutaja käest kas ta soovib alustada laadimisega. Tulemus on järgnev:



Joonis 18. Mobiilirakenduse parkimise alustamise vaade

4.6 Edasised arendused

Käesoleva lõputöö raames sai valmis disainitud ning programmeeritud laadimisjaama eesrakendus. Samuti on käesoleva lõputöö raames autor valmis disaininud toote. Edasiste arenduste hulka kuuluvad järgmised arendused:

- Tagarakenduse arendamine;
- Andmebaasi koostamine ning ehitamine;
- Reaalse prototüübi ehitamine.

5 Kokkuvõte

Käesoleva diplomitöö põhieesmärk oli valmis disainida ning programmeerida elektritõukeratta laadimisjaama eesrakendus. Samuti kuulus eesmärkide hulka laadimisjaama visuaalne disainimine.

Antud lõputöö analüüsi osas eritleti elektritõukeratta jaama potentsiaalsete klientide kasutajalugusid ning kasutajate funktsionaalseid ning mittefunktsionaalseid nõudeid. Käesoleva diplomitöö analüüsi osas on võrreldud erinevaid arenduskeskkondi ning arendustehnoloogiaid.

Käesoleva lõputöö skoop koosnes eesrakenduse disainimisest Axure tarkvara kasutades, eesrakenduse arendusest kasutades Xamarin platvormi ning toote enda disainimisest. Antud lõputöös on välja toodud olulisemad koodilõigud eesrakendusest ning pildid toote disainist.

Antud lõputöö tulemusena valmis elektriliikurite laadimiskapi eesrakendus koos kaardirakendusega ning tulevase toote disain.

Diplomitöö „Elektritõukeratta laadimiskapi eesrakenduse arendamine“ võib lugeda õnnestunuks, kuna suudeti täita eesrakenduse arenduse, disainimise ning toote disainimise eesmärgid. Samuti on lõputöös ära märgitud tulevased arendused, mis võetakse töösse esimesel võimalusel.

Kasutatud kirjandus

[1] "Bitbucket: What is Bitbucket?" *Atlassian.* [Online]. Saadaval: https://confluence.atlassian.com/confeval/development-tools-evaluator-resources/bitbucket/bitbucket-what-is-bitbucket. Külastatud: 10.11.2021.

[2] "Bolt kahekordistab tõukerataste arvu Tallinnas kuni 500 rattani," *Digigeenius.* (30.07.2019). [Online]. Saadaval: https://digi.geenius.ee/rubriik/uudis/bolt-kahekordistab-toukerataste-arvu-tallinnas-kuni-500-rattani/ . Külastatud: 10.11.2021.

[3] "Bolti elektritõukerattad: 10 korduma kippuvat küsimust," *Bolt*. (25.05.2020). [Online]. Saadaval: https://blog.bolt.eu/et/bolt-elektritoukerattad/. Külastatud: 11.11.2021.

[4] "First Ever Bikeep Scooter Rental Station In Radisson Meriton Hotel," *Bikeep*. [Online]. Saadaval: https://bikeep.com/first-ever-bikeep-scooter-rental-station. Külastatud: 01.11.2021.

[5] "Hello World," *GitHub Docs.* [Online]. Saadaval: https://docs.github.com/en/get-started/quickstart/hello-world. Külastatud: 05.11.2021.

[6] "JetBrains Rider vs Visual Studio (with and without ReSharper)," *JetBrains*. [Online]. Saadaval: https://www.jetbrains.com/rider/compare/rider-vs-visual-studio. Külastatud: 20.11.2021.

[7] "Top Programming Languages for Android App Development," *Geeks for Geeks.* (22.11.2021). [Online]. Saadaval: https://www.geeksforgeeks.org/top-programming-languages-for-android-app-development/. Külastatud: 14.11.2021.

[8] "Tõukerataste uputus Tallinnas: operaatorid võitlevad turuosa pärast, linn vaatab plaksutades pealt," *Postimees*. (25.05.2021). [Online]. Saadaval: https://majandus.postimees.ee/7253695/toukerataste-uputus-tallinnas-operaatorid-voitlevad-turuosa-parast-linn-vaatab-plaksutades-pealt. Külastatud: Nov. 10, 2021.

[9] *Axure.* [Online]. Saadaval: https://www.axure.com/. Külastatud: 20.11.2021.

[10] *Azure Devops* [Online]. Saadaval: https://azure.microsoft.com/en-us/services/devops/#overview. Külastatud: 25.11.2021.

[11] C. Hardt, K. Bogenberger, „Usage of e-Scooters in Urban Environments," *Transportation Research Procedia*, vol. 37, pp. 155 – 162, 2019.

[12] *GitHub* [Online]. Saadaval: https://github.com/. Külastatud: 02.11.2021.

[13] *GitLab.* [Online]. Saadaval: https://about.gitlab.com/what-is-gitlab/. Külastatud: 20.11.2021.

[14] *Inforegister*. [Online]. Saadaval: https://www.inforegister.ee/12351012-BIKEEP-OU. Külastatud: 10.11.2021.

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

Mina, Uku-Laur Jagomägi

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Elektritõukeratta laadimiskapi eesrakenduse arendamine“, mille juhendaja on Meelis Antoi.
   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

21.11.2021

1 Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või enam isikut oma ühise loomingulise tegevusega ning lõputöö kaas- või ühisautor(id) ei ole andnud lõputööd kaitsvale üliõpilasele kindlaksmääratud tähtajaks nõusolekut lõputöö reprodutseerimiseks ja avalikustamiseks vastavalt lihtlitsentsi punktidele 1.1. ja 1.2, siis lihtlitsents nimetatud tähtaja jooksul ei kehti.

Lisa 2 - Elektritõukeratta laadimisjaama visuaalse kontseptsiooni näidised