TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Artur Thomas Laherand 185245IADB

Sukeldumislogi rakenduse arendamine iOS platvormile

Bakalaureusetöö

Juhendaja: Einar Kivisalu
MSc (Informaatika)

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: Artur Thomas Laherand

17.05.2021

Annotatsioon

Käesoleva bakalaureusetöö eesmärgiks on luua mobiilirakendus iOS operatsioonisüsteemile, mille peamine eesmärk on pidada sukeldumislogi. Töö tulemus võib huvi pakkuda paljudele sukeldujatele, kes on huvitatud sukeldumislogi pidamisest.

Töös analüüsitakse olemasolevaid lahendusi ja kirjeldatakse hea sukeldumislogi rakenduse eeldused ja nõuded. Seejärel seletatakse lahti programmeeritava rakenduse arhitektuur, kasutatud tehnoloogiad ja arenduse protsess.

Töö tulemuseks on töötav ja intuitiivne sukeldumislogi rakendus, kus on olemas andmete pilve salvestus, gruppide tegemise, liitumise ja kustutamise süsteem ja kus oleks andmete sisestus lihtsustatud.

Lõputöö on kirjutatud eesti keeles ja sisaldab teksti 41 leheküljel, 7 peatükki, 21 joonist.

Abstract

Developing a diving log application on iOS platform

The aim of this bachelor's thesis is to create a mobile application for the iOS operating system, the main purpose of which is to keep a diving log. The result of the work may be of interest to many divers who are interested in keeping a diving log.

The thesis analyzes the existing solutions and describes the prerequisites and requirements for the application of a good diving log. The architecture of the programmable application, the technologies used, and the development process are then explained.

The result is a working and intuitive diving log application with cloud data storage, a system for creating, joining and deleting groups, and with implemented data entry simplification.

The thesis is written in Estonian and contains 41 pages of text, 7 chapters, 21 drawings.

Lühendite ja mõistete sõnastik

API	*Application programming Interface* ehk rakenduse programmeerimisliides.
Apple	Riist- ja tarkvara arendav ning tootev ettevõte.
App Store	Apple ettevõtte poolt toodetud seadetes kasutatav rakenduste pood.
Hobisukeldumine	Sukeldumine, kus osalevad hobisukeldujad või *scuba diver*'id, kes on vähemalt esimese sukeldumise kursuse läbinud sukeldujad ning kes käivad sukeldumas meelelahutuse ning puhkamise eesmärgil.
I18n	*Internationalization* ehk mitmes keeles rakenduse vaatamise võimalus
IDE	*Integrated development environment* ehk keskkond tarkvara arendamiseks.
iOS	Apple mobiilseadmete operatsioonisüsteem.
IPhone	Apple ettevõtte poolt toodetud mobiiliseade.
Logiraamat	Logiraamat on vihik kuhu kirjutatakse pärast iga sukeldumist kirja sukeldumise andmed, mis on tavaliselt: kuupäev ja kellaaeg, maksimaalne sügavus, vee- ja õhu temperatuur ja olud, vee all olemise aeg ning sukelduja kommentaarid.
MVC	*Model-View-Controller* on Apple ettevõtte eelistatuim rakenduse arhitektuuritüüp.
Objective-C	Programmeerimiskeel, mis oli peamiselt kasutusel Apple'i seadmetele rakenduste arendamisel enne Swifti olemasolu.
Pop-up	Hüpikmenüü

Professionaalne sukeldumine	Sukeldumine, kus osalevad välja õpetatud sukeldujad ning neile makstakse nende töö eest.
Promise	Programmi osa, mis vastutab andmebaasist andmete pärimise eest.
Prototüüp	Prototüüp on uue kavandatava toote mittetäielik esialgne teostus.
SensorTower	Veebirakendus, kus saab vaadata erinevate rakenduste populaarsuse statistikat.
Sukeldumiskell	Sukeldumiskell või sukeldumiskompuuter on sukelduja varustusse kuuluv kell, mis näitab kui kaua on vee all oldud, sügavust, maksimaalset sügavust ning tõusukiirust.
Swift	Apple ettevõtte poolt tehtud programmeerimiskeel, mis on mõeldud nende seadmetele rakenduste arendamiseks.
XCode	Apple ettevõtte poolt loodud IDE.

Sisukord

Jooniste loetelu

1 Sissejuhatus

Enamik sukeldujaid, olgu nad professionaalid või hobisukeldujad, peavad sukeldumise logiraamatut. Professionaalidel ka tihti nõutakse logiraamatu olemasolu erialase töökogemuse tõestamiseks. Logiraamatut täidetakse pärast iga sukeldumist ning sinna pannakse kirja tehtud sukeldumise andmed. Logiraamatut on hea pidada enda sukeldumiste jälgimiseks ning hiljem nende meenutamiseks. Kui on kahtlusi sukelduja kogemuses siis võidakse küsida logiraamatut tema kogemuse tõestamiseks, aga selle võib üldjuhul asendada ka asjakohaste küsimuste esitamisega või hinnates tema hakkama saamist varustuse kokkupanekul.

Bakalaureusetöö eesmärgiks seadis autor sukeldumislogi rakenduse loomise iOS platvormile, võttes arvesse sellise rakenduse jaoks vajalikke funktsionaalseid nõudeid ja vajadusi. Loodav rakendus on mõeldud kasutamiseks kõigile sukeldujatele, kes soovivad pidada oma sukeldumispäevikut enda nutitelefonis ning seda lihtsustatud kujul.

Bakalaureusetöö teoreetilises osas analüüsitakse nutitelefonis logiraamatu pidamise probleeme ning uuritakse olemasolevaid lahendusi.

Töö praktilises osas analüüsitakse iOS platvormile rakenduste loomise võimalusi ning tuuakse välja sukeldumislogi rakenduse nõuded. Selles osas uuritakse ka erinevaid tehnoloogiaid ja vahendeid, mida antud töö käigus valmiva rakenduse loomisel kasutatakse. Lisaks kirjeldatakse rakenduse testimise protsessi ja viise.

Töö viimases osas tuuakse välja rakenduse võimalikud edasiarendused ja puudused.

2 Logiraamatu pidamine

Antud peatükis kirjeldab autor logiraamatu pidamise võimalusi, analüüsib nende probleeme, püstitab ülesande ning kirjeldab lahendamiseks kasutatava metoodika. Lisaks teeb autor ülevaate tööst.

2.1 Taust

Tänapäeval peavad enamik maailma sukeldujaid logiraamatuid oma sukeldumiste kohta. Logiraamatusse tehakse iga sukeldumise kohta sissekanne, mis sisaldab tehtud sukeldumise andmeid, millest enamik loetakse sukeldumisel kaasas olevast sukeldumiskellalt. Sukeldumiskell on vajalik sukeldumise ajal sügavuse ning sukeldumise kestuse teadmiseks ning igal kellal peab olema funktsionaalsus, mis hoiatab ohtude eest, mis võivad tekkida sukeldumisel sügavust või muid tegureid mitte jälgides (näiteks kessoontõbi). Sukeldumiskellasid toodavad paljud erinevad ettevõtted ning ei ole olemas standardit, missuguseid andmeid sukeldumiskell näitama peab pärast sukeldumist. Standardi vajalikkus toodi välja ka 2013. aastal reaserchgate.net portaalis ilmunud väljaandes [1]. Siiski on teatud andmed, mida näitavad kõik professionaalsed sukeldumiskellad. See tagab klientidele valikuvõimaluse logiraamatute vahel.

Sukeldujad kasutavad enamasti ühte kahest logipidamise vahendist. Enamik sukeldujaid kasutavad sukeldumispäevikuna paberist vihikut. Mõnedes sukeldumisklubides on traditsiooniks kujunenud, et pärast esimese sukeldumiskursuse läbimist antakse igale läbinule isiklik vihik kuhu täidetakse koos oma esimene sukeldumine. Paberile saab pärast iga sukeldumist korraldaja käest ka templit küsida, mis ei ole kohustuslik, aga paljudele meeldib neid koguda, sest templid on igas kohas erineva disainiga. Vihiku alternatiiviks on mobiilirakendus. Mobiilirakenduse eeliseks on mobiiliseadmete väga suur kasutus üle maailma ja mobiilirakendust kasutades ei pea sukelduma minekuks ettevalmistudes logiraamatut eraldi meeles pidama.

Apple’i AppStore’s on väga palju erinevaid logipidamise rakendusi, millel on ka väga palju kasutajaid. Näiteks Strava rakenduse kasutajate arv oli 2021. aasta jaanuaris 76 miljonit [2]. Sukeldumiste logimiseks aga Strava ega teised populaarsed rakendused ei kõlba, sest

sukeldumise logimiseks on vaja kirja panna palju erinevaid andmeid, mida need rakendused ei võimalda.

Sukeldumist harrastab teiste sportidega võrreldes vähe inimesi ning seetõttu logi pidamiseks on eraldiseisva rakenduse loomine kõige parem lahendus.

2.2 Probleem

Mobiiliseadmes logiraamatut mugavalt pidamiseks on vaja tagada logiraamatu säilivus ka mobiilseadet vahetades. Praegused levinumad mobiilirakendused kasutavad selle jaoks varufaili süsteemi, kus algul tuleb vanas telefonis teha varufail, seejärel see uude seadmesse saada ning seal lahti pakkida. See süsteem on keeruline ning see on ka paberi kasutamise suur eelis, et ta on füüsilisel kujul olemas ning uut vihikut soetades ei pea muretsema vana kadumise pärast.

Sukeldumise logimisel peab välja kirjutama palju erinevaid andmeid, mida saab logiraamatu mobiilversioonis lihtsustada. Lihtsustatud kujul peaks iga sukelduja sisestama ainult need andmed, mis erinevad teiste sukeldujate andmetest. See on saavutatav gruppide süsteemiga, kus saab teha logi sisendi kavandi, kuhu teised saavad sisestada oma andmed ning lõpuks salvestada sukeldumine oma logisse.

Sukeldujad tahaksid huvi pärast vaadata teiste sukeldujate logiraamatuid aga pabervihiku kujul logiraamatute kasutus ei võimalda seda efektiivselt teha. Mobiilirakenduses saaks teiste logiraamatute vaatamise mugavamaks ning kiiremaks muuta.

2.3 Ülesande püstitus

Käesoleva lõputöö eesmärk on luua iOS operatsioonisüsteemil töötav mobiilirakendus, mille peamine eesmärk on sukeldumiste logimine. Valminud rakendus peab võimaldama sukeldumisi logida, vaadata, gruppidesse jagada, omavahel grupeerida ja kustutada. Samuti peab seal olema võimalik vaadata teiste sukeldujate logisid, kui need ei ole privaatseks määratud. Rakenduses peab olema ka andmete pilve salvestus.

Rakendus realiseeriti prototüübina Axure RP 9 keskkonnas ning hiljem rakendusena iOS platvormil Swifti keeles, kasutades XCode integreeritud arenduskeskkonda. Programmi testitakse IPhone X mobiiliseadmel, mis kasutab iOS 11.1.2 operatsioonisüsteemi.

2.4 Metoodika

Enne rakenduse arendust analüüsiti esmalt teisi turul leiduvaid lahendusi ja nende põhjal otsustatati, mis on hea sukeldumise logiraamatu puhul oluline. Selleks uuriti nii logiraamatu vihiku versiooni kui ka olemasolevaid logiraamatu rakendusi.

Analüüsist saadud andmete põhjal koostati rakenduse prototüüp, mis visualiseerib seda, milline rakendus peaks valmides välja nägema ja, mis lihtsustas tervet arendusprotsessi. Prototüübi koostamine võtab ka tunduvalt vähem aega kui rakenduse programmeerimine ja annab hea ettekujutuse võimalikest alternatiivlahendustest disaini küsimustes. [6]

Rakenduse valmimiseks kasutati mitut erinevat vahendit ja keskkonda, mille hulka kuuluvad XCode, Swift, Axure RP 9, iOS, IPhone X jm.

2.5 Ülevaade tööst

Käesolev töö koosneb neljast osast. Esimeses osas kirjeldatakse analüüsi osa. Seal analüüsitakse olemasolevaid lahendusi ning tuuakse välja hea logiraamatu nõuded, mis jagatakse kahte rühma: funktsionaalsed nõuded ja mittefunktsionaalsed nõuded. Teises osas kirjeldatakse arendusprotsessi. Seal tutvustatakse arenduse käigus kasutatud vahendeid, keskkondasid ning tehnoloogiaid. Kolmandas osas kirjeldatakse rakenduse testimise protsessi ja testimise tulemusi. Töö viimases osas tuuakse välja rakenduse võimalikud edasiarendused.

3 Analüüs

Enamikul maailma inimestest on tänapäeval nutitelefon. 2021. aasta jaanuari andmete järgi ~27% kõigist maailma nutitelefonidest kasutavad iOS operatsioonisüsteemi [3]. Androidi operatsioonisüsteemi kasutavad ~72% seadetest ning seega Android on turuliider. Peamine põhjus miks autor otsustas antud lõputöös arendatav rakendus programmeerida iOS platvormile, sest Androidi seadmed oluliselt varieeruvad üksteisest ning sinna rakendusi arendades tuleb silmas pidada erinevaid Android tarkvara versioone ning iga seadme omapärasusi [4]. Selle töö eesmärgi täitmine Android platvormil oleks seetõttu palju mahukam ettevõtmine. Autor kasutab ka ise Apple seadmeid ning plaanib tulevikus lõputöö praktilise osana valmivat rakendust ise ka kasutama hakata.

Lõputöö eesmärgiks on arendada välja mugav ja intuitiivne sukeldumise logiraamatu rakendus iOS platvormile. Praeguseks hetkeks on turul sarnaseid lahendusi juba olemas ja seetõttu otsustas töö autor selles peatükis neid lahendusi omavahel võrrelda, tuua välja nende plussid ja miinused ning teha järeldused – milline peaks hea sukeldumise logiraamat olema.

3.1 Olemasolevate lahenduste analüüs ja võrdlus

Olemasolevate lahenduste valik analüüsi toimus iOS rakenduste poe uurimise käigus ning logiraamatute näited paberi kujul on saadud veebist leitud näidetest. Välja toodud on vaid üks vihiku näide, sest vihikud erinevad üksteisest vaid disaini poolest ning funktsionaalset vahet neil ei ole. Mobiilirakendusi valiti populaarsuse, kasutatavuse ja lõputööga võimalikult sarnase funktsionaalsuse põhjal.

Olemasolevaid lahenduste allalaadimiste statistikat võrreldi SensorTower veebirakenduses. Kõikide olemasolevate lahenduste allalaadimiste numbrid jäid alla 5000 allalaadimise (veebirakendus ei näita täpsemalt kui <5000, kui allalaadimisi on vähe). Maailmas on kokku umbes üheksa miljonit sukeldujat [5]. SensorTower veebirakenduses läbi viidud uuringus selgus et turule pääsemise võimalus on suur ning on palju potentsiaalseid uusi kasutajaid hea lahenduse olemasolul.

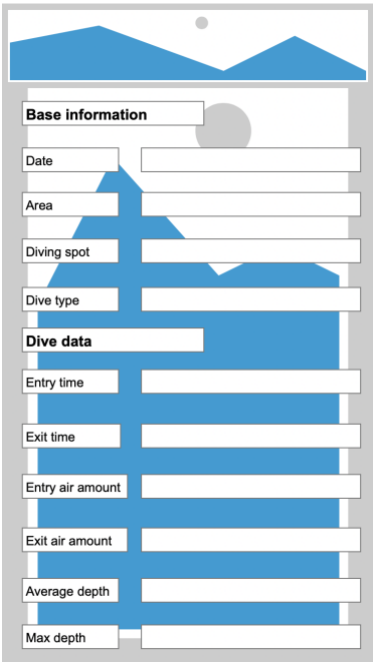
Turu-uuringu käigus tehti kindlaks, et turul ei eksisteeri veel ühtegi lahendust, mis hõlmaks endas logiraamatu pilve salvestamist, gruppide süsteemi ning sellest tingitud sukeldumise info sisestamise lihtsustatust ning teiste sukeldumislogide vaatamist. Selle uuringu põhjal tehti järeldus, et arendataval rakendusel võiks leiduda huvilisi sukeldujate seas.

3.1.1 My Diving Log

Rakenduse disain on halb, sest näiteks rakendusel puudub pealeht ning rakendust esimest korda avades tuleb lahti sinine leht seitsme nupuga, mille peal on sümbolid (vt. Joonis 1). Ei ole arusaadav mille jaoks mõned nupud on ja seitsmest nupust nelja vajutades ei juhtu midagi. Need neli nuppu on sukeldumiste sorteerimiseks, milleks tavaliselt piisab ühest nupust, mis avab sorteerimisvalikud. Sellel rakendusel on aga hästi disainitud sukeldumiste lisamise võimalus, mis on arusaadavate sümbolitega tähistatud (vt. Joonis 2). Arusaadavad sümbolid lisavad rakendusele intuitiivsust ning lisavad sellele erksust. Antud töös arendatavas rakenduses kasutatakse sarnase disainiga sukeldumise lehte (vt. Joonis 3).
Selles rakenduses ei ole võimalik näha teiste inimeste logisid ning telefoni vahetades ei saa tavakasutaja enda andmeid uude seadmesse üle viia.

Joonis 1. My Diving Log rakenduse pealeht.

Joonis 2. My Diving Log rakenduse sukeldumise lisamise leht.

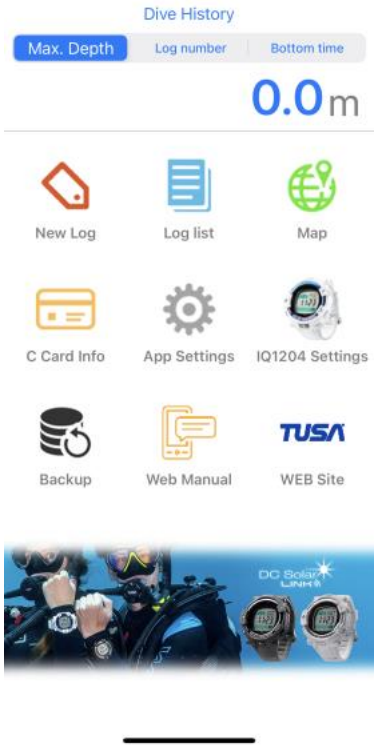


Joonis 3. Rakenduse prototüübi sukeldumise lisamise leht.

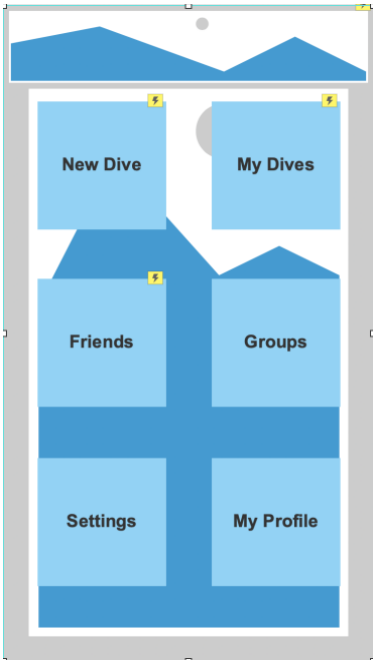
3.1.2 TUSA Diving Log

Antud rakendus on tehtud sukeldumisvarustust tootva ettevõtte poolt ning on mõeldud kasutamiseks nende toodetud sukeldumiskellaga DC Solar Link. See kell ühendub rakendusega *bluetooth*-i kaudu ning saadab sinna sukeldumise andmed automaatselt. Rakendust saab

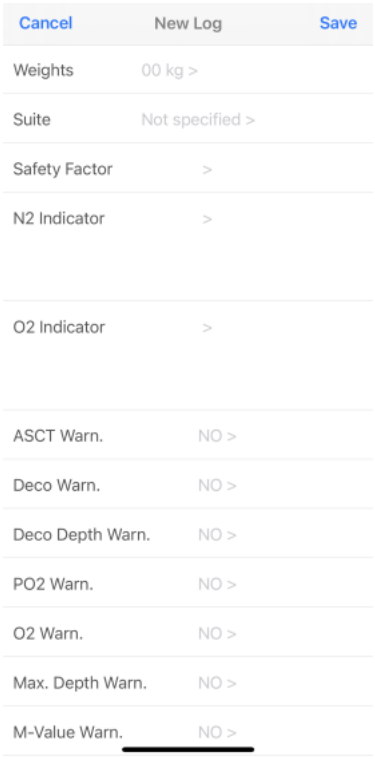
kasutada ka ilma DC Solar Link kellata. Rakenduse disain on üpris hea (vt. Joonis 4). Sellel on ilus pealeht, kus on nupud ilusate ja arusaadavate sümbolitega, mis selgitavad mida see nupp teeb. Miinuseks on pealehe kohal olevad kolm nuppu kust saab muuta statistika vaadet, mis tundub vale koht selle jaoks. Antud töös arendatava rakenduse pealehel kasutatakse sarnast disaini (vt. Joonis 5). Logi kirje lisamisel on võimalus sisestada ka andmeid mida muudes rakendustes sisestada ei saa (vt. Joonis 6). See võimalus on kasutu enamik sukeldujatele, sest enamik sukeldumiskellasid, mis ei ole selle ettevõtte poolt toodetud, neid parameetreid ei näita. Samuti on sellel rakendusel olemas andmete salvestus, aga see käib läbi iOS platvormi, mis on kasutaja jaoks keerulisem ning piiravam kui andmete pilve salvestamine.



Joonis 4. Tusa Diving Log rakenduse pealeht.



Joonis 5. Rakenduse prototüübi pealeht.

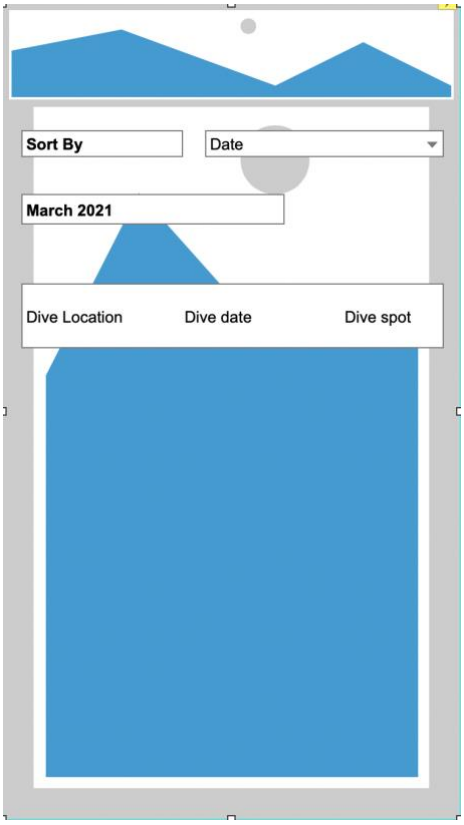


Joonis 6. Tusa Diving Log rakenduse sukeldumise lisamise leht. Pildil on näha andmeid, mida tavaline sukeldumiskell ei näita.

3.1.3 Deep Tools

Antud rakendus on mõeldud pigem professionaalidele kasutamiseks ning sisaldab funktsionaalsust, et teha arvutusi, mis on vajalikud hobisukeldumisest raskemate ning sügavamate sukeldumiste planeerimiseks. Selle rakenduse logiraamatu pidamise võimalus on kesine, sest uue sukeldumise lisamisel küsitakse liiga vähe informatsiooni selle kohta. Rakenduses puudub ka võimalus oma logi ja andmeid salvestada. Rakenduses on hästi disainitud kõigi sukeldumiste leht, kus saab sukeldumisi sorteerida ja otsida ning näidatakse veidi statistikat kõigi sukeldumiste kohta (vt. Joonis 7). Antud töös arendatava rakenduse kõikide sukeldumise lehel kasutatakse selle disaini aspekte (vt. Joonis 8).

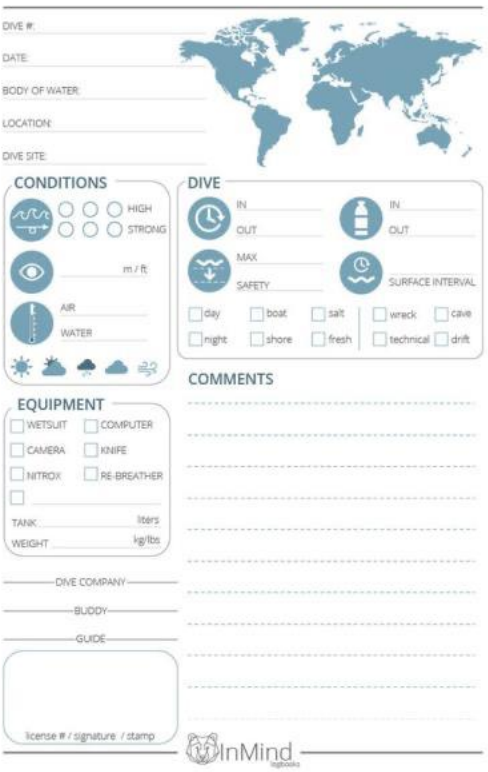
Joonis 7. Deep Tools rakenduse kõikide sukeldumiste vaatamise leht.



Joonis 8. Rakenduse prototüübi kõikide sukeldumiste vaatamise leht.

3.1.4 Füüsiline logiraamat

Füüsiline logiraamat näeb välja nagu tavaline vihik või päevik, milles üldjuhul iga leht on mõeldud uue sukeldumise sissekande jaoks (vt. Joonis 9). Sukeldumispäevikust saab hea ettekujutuse, millist infot sukeldujad logida tahavad.



Joonis 9. Füüsilise logiraamatu ühe sukeldumise sissekandmise leht.

3.2 Rakenduse nõuded

Antud peatükis kirjeldatakse arendatava rakenduse funktsionaalsed- ning mitte funktsionaalsed nõuded. Funktsionaalsed nõuded kirjeldavad rakenduse käitumist või tegevust, mille abil saab kasutaja oma soovitud tulemuse. Mittefunktsionaalsed nõuded käsitlevad rakenduse arhitektuuri osa, mis hõlmab rakenduse jõudlust, kasutatavust ning käideldavust.

3.2.1 Rakenduse funktsionaalsed nõuded

1. Andmete pilve salvestus.
2. Sukeldumise logimise võimalus
3. Kõikide sukeldumiste vaatamise võimalus
4. Logi kirje muutmise ja kustutamise võimalus
5. Sõprade logide vaatamise võimalus
6. Grupi loomise, muutmise ja kustutamise võimalus
7. Grupiga liitumise ning grupist välja astumise võimalus

8. Grupi logi kirje loomise võimalus
9. Enda profiili muutmise ja kustutamise võimalus
10. Rakendus peab toetama i18n funktsionaalsust

3.2.2 Rakenduse mittefunktsionaalsed nõuded

1. Rakenduse disain peab olema kasutajasõbralik ning intuitiivne
2. Rakenduse kasutajaliides peab olema inglise keelne
3. Rakendus peab töötama iOS operatsioonisüsteemil
4. Rakendusel peab olema avaleht kust saab erinevate tegevuste juurde navigeerida

4 Arendus

Lõputöö käigus on valmimas sukeldumislogi rakendus. Selles peatükis tutvustab autor rakenduse tehnilist poolt ning arenduses kasutatud tehnoloogiaid ja nende kasutust.

4.1 Rakenduse tehnoloogiline ülevaade

Arendatav rakendus on iOS operatsioonisüsteemipõhine ning on kirjutatud XCode keskkonnas Swift keeles. Autor valis Swift keele sellepärast, et see on ainuke programmeerimiskeel, mis on Apple ettevõtte poolt tehtud spetsiaalselt nende seadmetele rakenduste kirjutamiseks. Programmeerimiskeelte valikus oli ka Objective-C, aga seda kasutati pigem enne Swifti olemasolu ning ka mitmed artiklid soovitasid Swifti selle uudsuse, kiiruse ning lihtsuse tõttu [7][8][9].

4.1.1 iOS

iOS on Apple ettevõtte poolt tehtud operatsioonisüsteem, mis töötab eksklusiivselt iPhone nutitelefonidel, iPad tahvelarvutitel ning iPod multimeediamängijatel. iOS on hetkel maailmas populaarsuselt teine nutitelefoni operatsioonisüsteem. 2021. aasta jaanuari andmetel olid kõikidest maailmas müüdud nutitelefonidest ~27% iPhone telefonid [3]. Ennustatakse, et iOS'il töötavaid rakendusi on aastaks 2022 tõuseb iPhone'ide kasutus veel 4% [10]. Rakenduste arendamine iOS operatsioonisüsteemile toimub XCode arenduskeskkonnas. Arendamiseks on vajalik ka iOS'i Standard Development Kit ehk SDK, mis võimaldab programmeerijal luua rakendusi iOS platvormile. XCode on samuti arendatud Apple'i poolt ning see võimaldab teha rakendusi kõikidele Apple'i seadmetele. Kasutades XCode'i ja iOS'i SDK-d on võimalik arendada rakendusi Swift ning Objective-C keeltes. Tasub ka ära märkida, et XCode töötab vaid macOS'i operatsioonisüsteemil.

4.1.2 QSEE SuperLite

QSEE SuperLite on arvuti rakendus, mis on mõeldud andmebaasi skeemide loomiseks. Rakendus on väga mugav ja intuitiivne kasutamiseks ning skeemidel on ilus ja arusaadav disain. Antud töö praktilise osa andmebaasi skeem on koostatud selles rakenduses.

4.1.3 UserDefaults

Andmete salvestamiseks on iOS operatsioonisüsteemil Defaults andmebaas, kuhu saab salvestada andmeid võti-väärtus paaridena. UserDefaults klass on programmaatiline liides mille kaudu saab suhelda Defaults andmebaasiga. Defaults andmebaas on mugav lahendus kasutaja poolt sisestatud sukeldumiste salvestamiseks ilma internetiühenduseta, et pärast rakenduse sulgemist jääksid kasutaja sisestatud sukeldumised alles kuni internetiühenduse tekkimiseni.

4.1.4 SwiftUI

SwiftUI on Apple ettevõtte poolt toodetud programmeerimiskeele Swift jaoks loodud arendusraamistik, mis on uue põlvkonna raamistik ning mis muudab rakenduse disainimise palju kergemaks kui see muudes keeltes on. SwiftUI kasutab deklaratiivset süntaksi, tänu millele saab arendaja kirjutada intuitiivset koodi, mis teeb täpselt seda mida ta ütleb. SwiftUI on ka sisse ehitatud kõikidesse iOS operatsioonisüsteemi kasutatavatesse seadetesse ning seetõttu see on kõige rohkem jõudlust ja võimalusi pakkuv Swifti raamistik. [11]

4.2 Rakenduse kirjeldus

Töö autor arendab lõputöö käigus iOS operatsioonipõhise rakenduse, milles saab pidada sukeldumislogi, gruppe luua, gruppidega liituda ning nendest lahkuda ning sõprade sukeldumislogisid vaadata. Rakenduses on ka sukeldumiste logimine tänu gruppidele lihtsustatud. Rakendusel on kokku 17 erinevat vaadet.

4.2.1 Sisse logimine, laadimisleht ning avaleht

Rakenduse avamisel saab soovi korral (kui seda pole juba tehtud) sisse logida või uue kasutaja luua. Pärast sisse logimist tuleb lahti laadimisleht andmete laadimise ajaks. Laadimislehest edasi tuleb avaleht, kus on 6 tegevuse juurde minemise nuppu (vt. Joonis 5).
Selline disain on väga intuitiivne ja kasutajasõbralik. Sellise disainiga ei teki kasutajal kunagi segadust ning tal on lihtne jälgida mis rakenduses toimub. Sisse logimise lehe programmeerimine oli keeruline, sest rakendust peab olema võimalik kasutada ka ilma interneti ühenduseta ehk siis ka ilma sisse logimiseta. Laadimislehe implementeerimise peamine

keerukus seisnes andmete laadimise ootamise järel ning seejärel avalehe programmeerimine pani aluse rakenduse struktuurile ning edasisele arendusprotsessile.

4.2.2 Profiili ning seadete lehed

Kasutajal on võimalus vaadata ning muuta enda profiilil olevat infot. Kasutaja saab ka vaadata ning muuta seadeid, mis määravad ära kasutaja poolt eelistatavaid ühikuid ning kas kasutaja soovib, et tema logi oleks avalik.

Kasutaja poolt valitud seaded hakkavad tööle kohe kui kasutaja neid muudab ning salvestab. Kõige raskem oli siin implementeerida kasutaja logi avalikuks ning privaatseks muutmise nuppu.

4.2.3 Uue sukeldumise logimise leht ning *pop-up*

Kui kasutaja soovib uut sukeldumist logida ning ta on mingis grupis, siis talle tuleb ette *pop-up*, mis küsib, kas kasutaja soovib logida sukeldumist endale või mingisse gruppi. See on logimise lihtsustamise protsessi jaoks vajalik, mis eeldab et üks inimene grupist täidab ära sukeldumise kõigi andmetega ning ülejäänud grupi liikmed saavad kasutada seda logikirjet šabloonina. Kui kasutaja ei ole üheski grupis või soovib lihtsalt endale sukeldumist logida, siis avaneb logimise leht, kuhu saab kasutaja oma andmed sisestada.

Pop-up'i tegemine oli keerukas, sest oli keeruline välja mõelda head disaini. Kui pop-up oli tehtud siis tegi autor kaks mudelit sukeldumiste logimise jaoks, üks *struct* ja teine klass. *Struct*'i kasutas autor grupi sukeldumise loomiseks, sest sellisel juhul saab teha objektist mitu koopiat, mis käituvad nagu iseseisev objekt. Tavalise sukeldumise kirje lisamise mudel on klass, sest seda kopeerida kuskile vaja ei ole ning tähtis on vaid selle üks eksemplar.

4.2.4 Minu sukeldumiste leht

Kasutajal on võimalus vaadata enda logitud sukeldumisi. Iga sukeldumise peal on ka kirjas kuupäev, koha nimetus, vee all oldud aeg ning maksimaalne sügavus. Sukeldumist vajutades

avaneb eraldi leht, kus on kirjas kõik sukeldumise andmed. Kui kasutaja sukeldumislogi on avalik, siis sama lehte saavad vaadata ka selle kasutaja jälgijad.

Peamine murekoht oli sellel lehel andmete näitamise disain. Igal sukeldumislogi kirjel on juba lehel mõned andmed peal ning need tuli ilusti paigutada ja nähtavaks teha.

4.2.5 Jälgijate leht ning jälgijate lisamise leht

Kasutajal on võimalus vaadata teiste kasutajate profiile, keda ta jälgib. Jälgijate lehelt näeb ta teiste kasutajate profiile, keda ta jälgib. Teiste kasutajate profiilidelt saab vaadata nende sukeldumislogisid ning iga sukeldumise infot.

Jälgijate lehelt saab ka edasi minna jälgijate lisamise lehe peale. Seal on otsimisriba, kus saab nime või unikaalse identifitseerija järgi teisi kasutajaid otsida.

Jälgijate funktsionaalsuse lisamisel oli kõige keerulisem andmebaasis tabelite koostamine ning ka jälgimise ning selle lõpetamise implementeerimine.

4.2.6 Gruppide lehed

Gruppide lehel saab kasutaja vaadata gruppe, kus ta parasjagu on ning ka ise uusi gruppe luua. Gruppide lehel on ka võimalus navigeerida gruppide otsimise lehele ning sealt astuda teistesse gruppidesse. Gruppide otsimise lehel on otsimisriba, kuhu saab sisestada grupi nime või unikaalse grupi identifitseerija.

Gruppide süsteemi arenduse keerukus seisnes grupi sukeldumislogi kirje lisamise võimaluse arendusega.

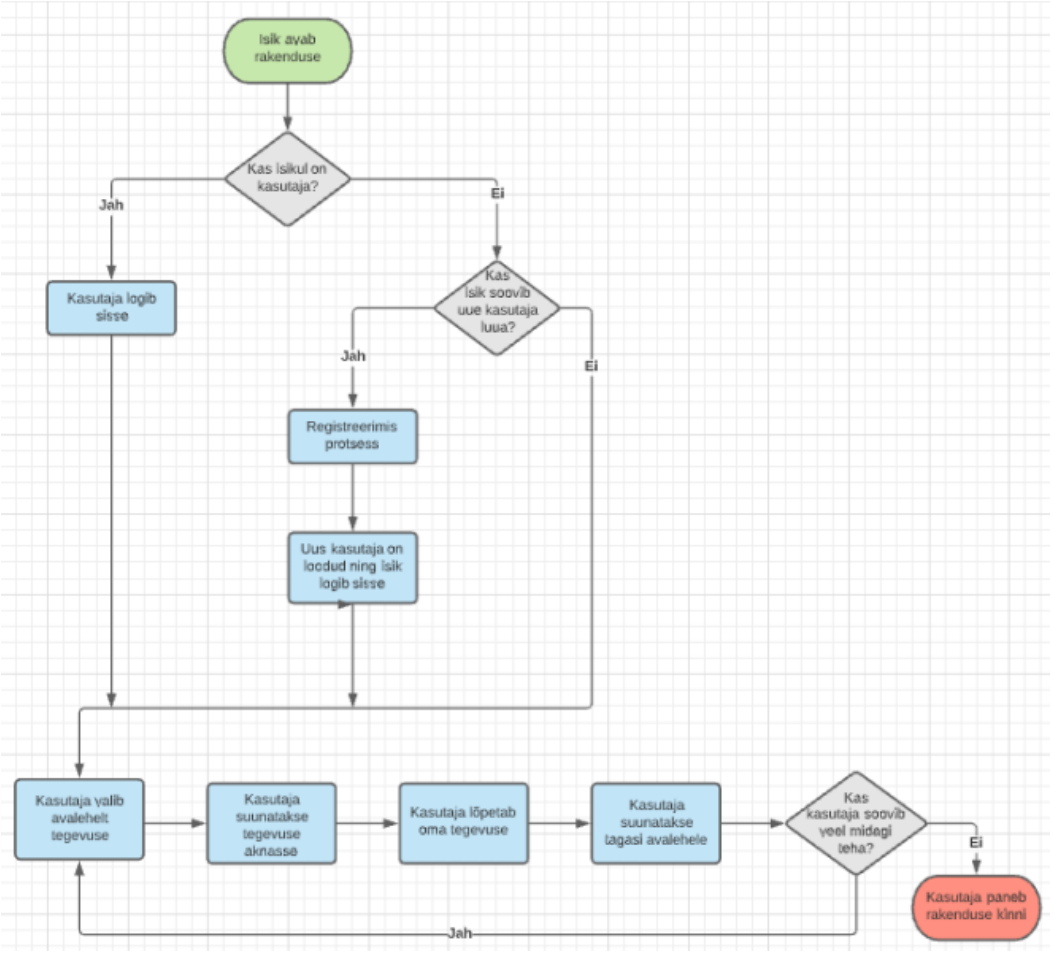
4.3 Rakenduse arendamiseks koostatud skeemid

Enne rakenduse arendusprotsessi koostas autor 8 skeemi ning rakenduse prototüübi.

4.3.1 Kasutaja kasutusloo diagrammid

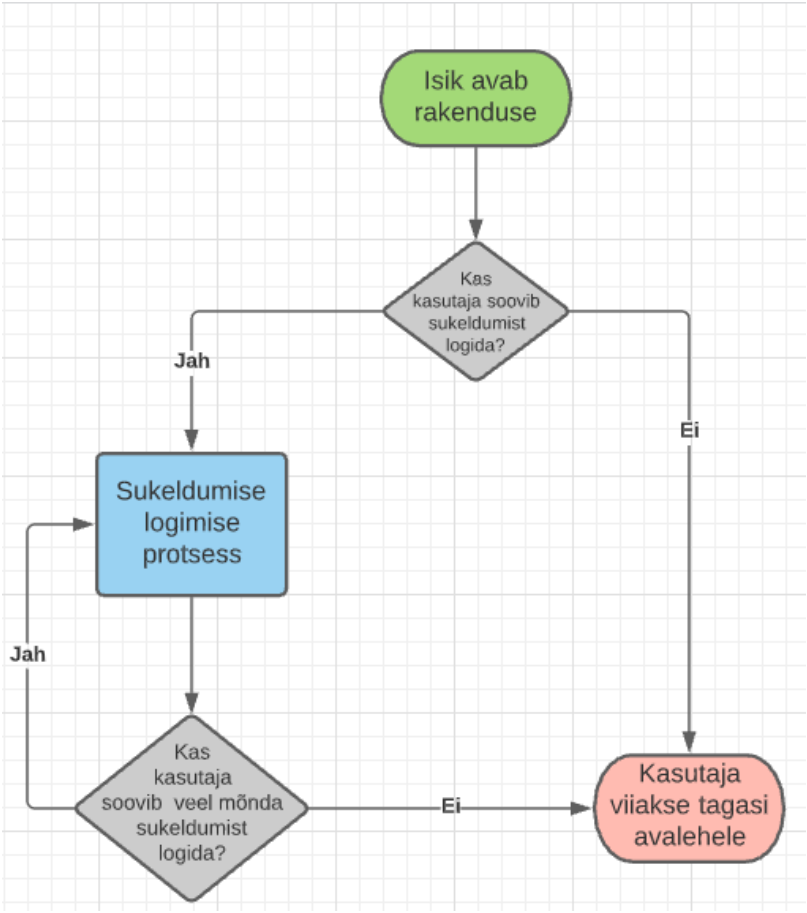
Kasutuslugude diagrammid on head alused rakenduse arenduseks, sest see annab hea ülevaate vajalikust funktsionaalsusest ning seab alustalad, milline rakenduse struktuur olema peaks.

Esiteks koostas autor kasutaja sisselogimise kasutusloo diagrammi (vt. Joonis 10), kus ta pani paika, kuidas peab kasutaja jaoks sisse logimise protsess välja nägema.



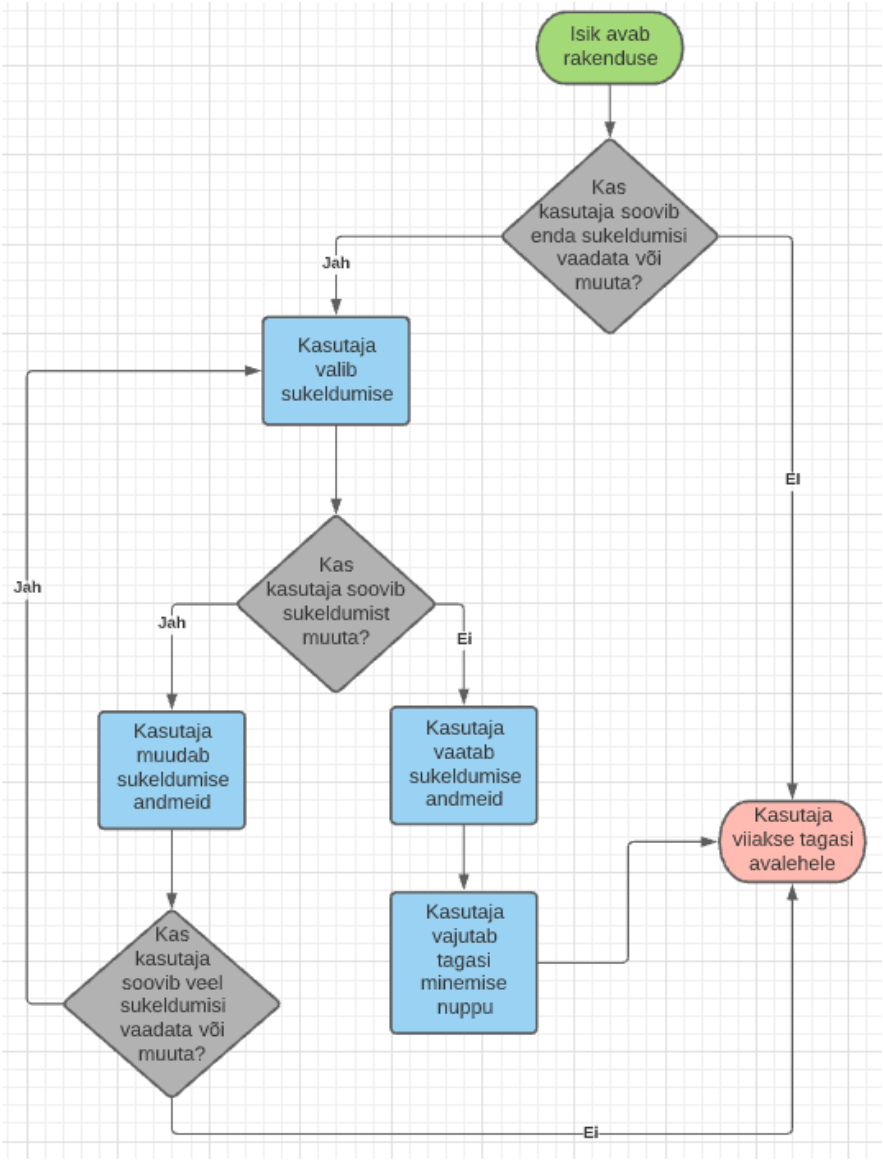
Joonis 10. Kasutaja sisselogimise kasutusloo diagramm.

Järgmiseks koostas autor kasutaja sukeldumise logimise protsessi kasutajaloo, mis kirjeldab kasutaja tegevuskava sukeldumise logimisel (vt. Joonis 11).



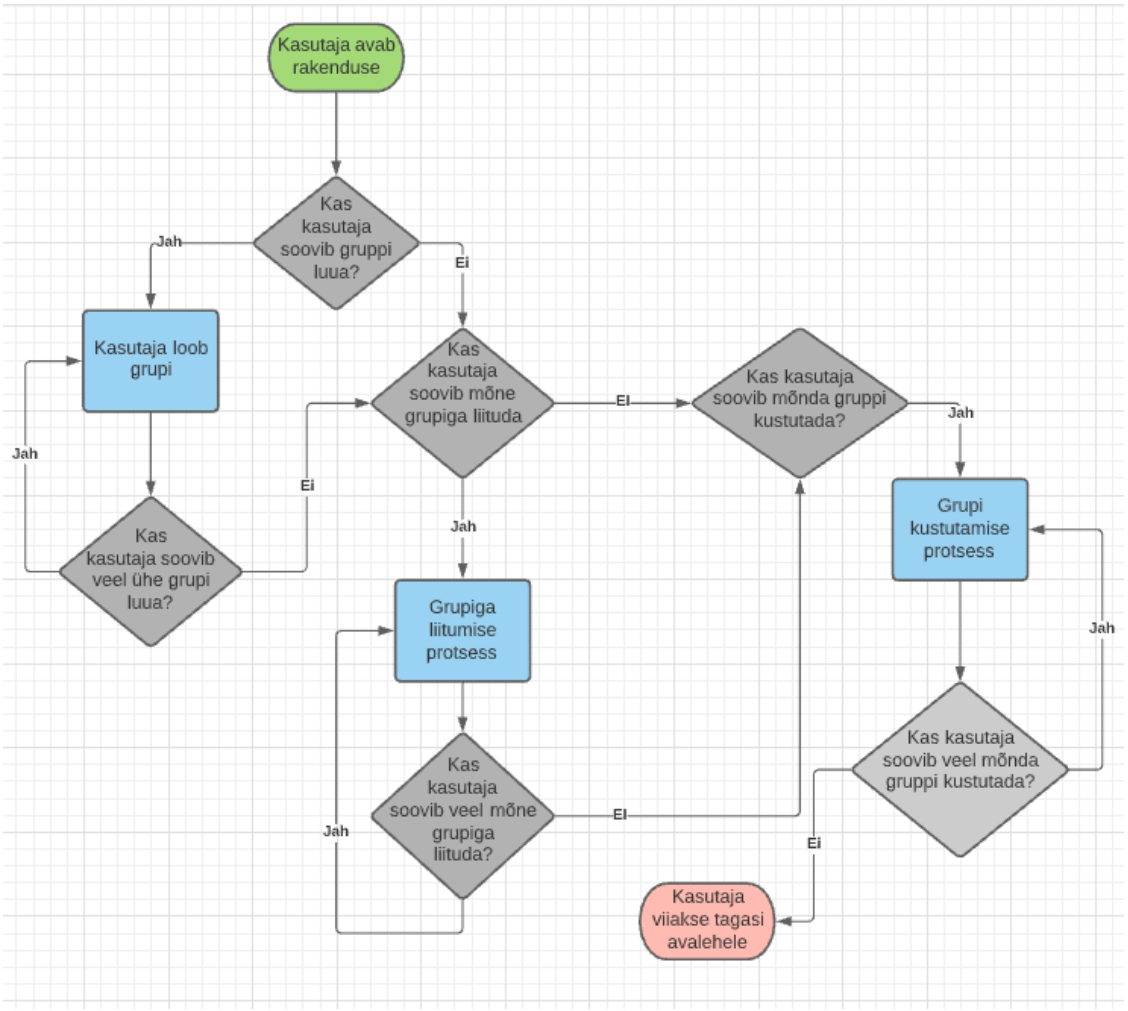
Joonis 11. Kasutaja sukeldumise logimise kasutusloo diagramm.

Seejärel koostas autor sukeldumise vaatamise ja muutmise kasutusloo diagrammi, mis kirjeldab kasutaja tegevuskäiku sukeldumiste vaatamisel või kui kasutaja soovib mõne sukeldumise infot muuta (vt. Joonis 12).



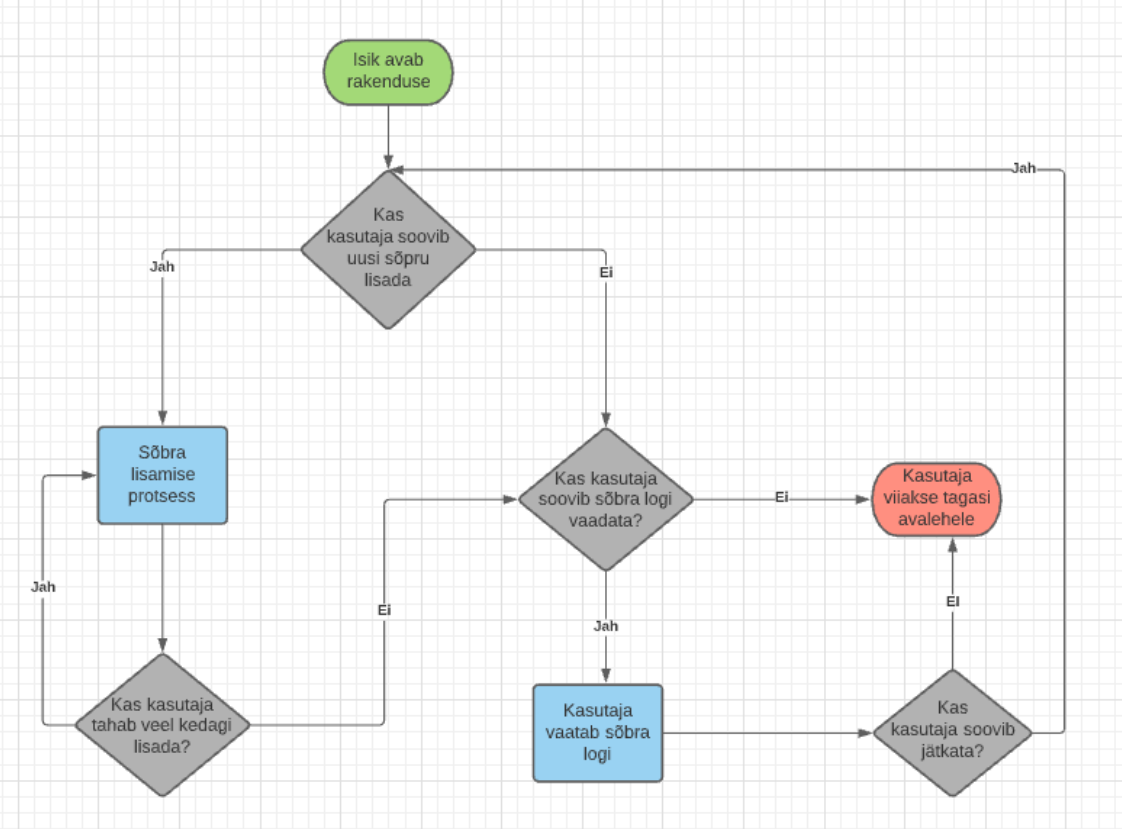
Joonis 12. Kasutaja sukeldumiste vaatamise ja muutmise kasutajaloo diagramm.

Järgnes kasutaja grupi loomise, kustutamise ja grupiga liitumise kasutusloo diagramm. See diagramm kirjeldab kasutaja samme grupi loomiseks, kustutamiseks või sellega liitumiseks (vt. Joonis 13).



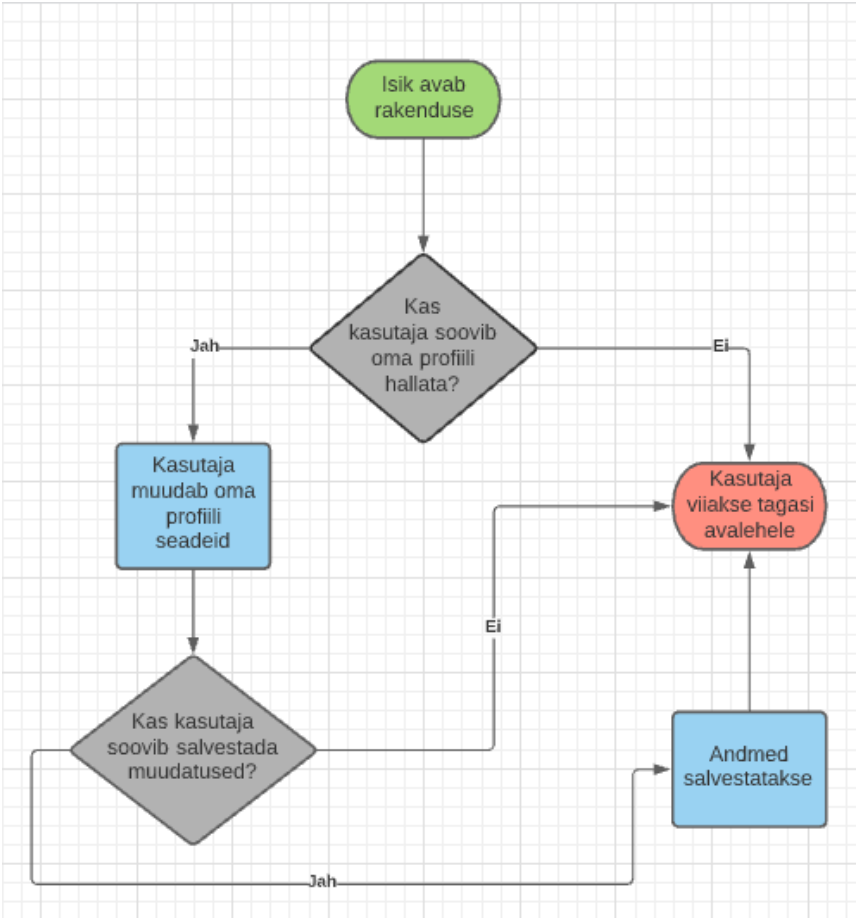
Joonis 13. Kasutaja grupi loomise, kustutamise ja grupiga liitumise kasutuslugude diagramm.

Järgnes kasutaja kasutuslugude diagramm sõbra lisamise ning sõprade logide vaatamise kohta. See diagramm kirjeldab kasutaja tegutsemisviisi sõprade jälgima hakkamiseks ning nende logide vaatamiseks (vt. Joonis 14).



Joonis 14. Sõprade lisamise ja nende logide vaatamise kasutuslugude diagramm.

Viimase kasutuslugude diagrammi koostas autor kasutaja profiili seadete muutmise kasutuslugude kohta. See diagramm kirjeldab kasutaja tegutsemisviisi enda profiili seadete muutmiseks. (vt. Joonis 15).



Joonis 15. Kasutaja profiili muutmise kasutuslugude diagramm.

4.3.2 Kasutuslood

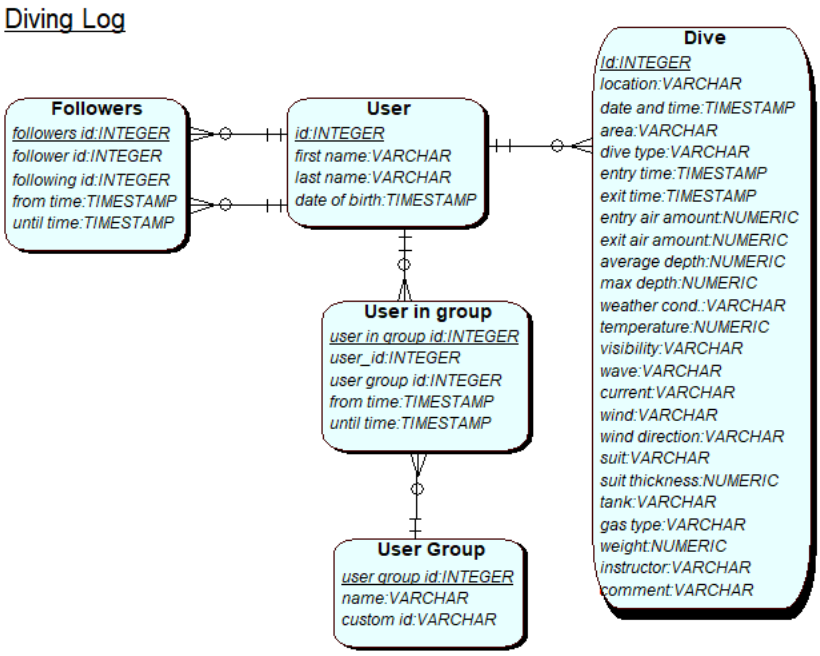
Kasutuslugude abil saab analüüsida, mida kasutaja sooviks teha ning milleks. Sellega saab hea ettekujutuse, milline funktsionaalsus rakendusel peaks olemas olema ning see seab ka vajaliku funktsionaalsuse hierarhiasse vajalikkuse järgi (vt. Joonis 16).

Kasutajaloo prioriteet	**Kasutajana ma sooviksin** *<midagi teha>*	**et ma saaksin** *<midagi saavutada>*
Kõrge	sukeldumist logida	sukeldumislogi pidada
Kõrge	sisse logida	andmeid salvestada
Kõrge	enda sukeldumisi vaadata ja muuta ja kustutada	sukeldumislogi hallata
Keskmine	uusi sõpru lisada	nende logisid vaadata
Kõrge	grupiga liituda	sukeldumise logimist lihtsustatult teha
Kõrge	gruppi luua	sukeldujaid gruppi lisada
Kõrge	gruppi kustutada	ainult aktiivseid gruppe näha
Kõrge	enda profiili näha ja muuta	enda profiili hallata

Joonis 16. Kasutuslood.

4.3.3 Andmebaasi ERD skeem

Andmebaasi ERD skeemi on vaja, et saada visuaalne ettekujutus rakenduse andmete hoiustamise struktuurist ning selle järgi on ka kergem andmebaasi luua. ERD skeemil on välja toodud olemid ning nende seosed teiste olemitega (vt. Joonis 17).



Joonis 17. Arendatava rakenduse ERD skeem.

4.4 Arenduse protsess

Arenduse algusjärgus oli peamiseks raskuseks autori vähene kokkupuude Swiftiga. Peamise õppematerjalina programmeerimiskeele õppimiseks kasutas autor Swifti dokumentatsiooni ja raamatut „Learning Swift: Building Apps for MacOS, iOS and Beyond“ [12] [13].

Programmeerimist alustas autor mudelite koostamisest koodis. Mudelite koostamisel tuli valida kas teha mudel *struct*'iks või klassiks. Nende vahe on see, et mudelit kopeerides *struct* loob kaks unikaalset koopiat mudelist, samal ajal kui klass viitab mudelile. *Struct*'i kasutamine oli väga oluline grupi sukeldumise lisamisel, sest seda kasutades saab üks kasutaja luua malli ning teised grupi kasutajad seda kasutada ning sinna ära täita oma sukeldumise info.

Pärast mudelite koostamist tegi autor valmis andmebaasi ning jätkas rakenduse arhitektuuri paika panemisega. Rakenduse peamiseks arhitektuuriks valis autor MVC, sest see on Swifti arendajate eelistatuim rakenduse arhitektuur ning enamasti kasutatakse seda vaikimisi [14].

Kui rakenduse arhitektuur sai paika, hakkas autor tegelema rakenduse vaadete ja funktsionaalsuse lisamisega. Selleks kasutas autor SwiftUI-d, mis tegi protsessi väga mugavaks, arusaadavaks ja kiireks. Sellega koos tegi autor valmis ka *promise*'d, mille eesmärgiks on õigete andmete pärimine andmebaasist.

Arendusprotsessi käigus rakenduse testimiseks kasutas autor iPhone SE teise generatsiooni simulaatorit ning füüsiliselt olemasolevat iPhone X nutitelefoni.

5 Testimine

Rakenduse testimine käib järk järgult ning tulemuste õigeks hindamiseks kasutab autor muudele tehnoloogiatele ja vahenditele lisaks ka viite testkasutajat, kes hindavad arendusprotsessi erinevatel etappidel rakenduse disaini, funktsionaalsust ning intuitiivsust. Testkasutajate kasutamise eesmärgiks on saavutada võimalikult kasutajasõbraliku ning intuitiivse disainiga rakendus ning rakenduse sisese funktsionaalsuse probleemide võimalikult varajane tuvastus. Testkasutajate tagasiside tulemusel rakenduses midagi suurt ei muudetud, sest oli positiivne ning probleeme disainiga ei olnud kellelgi. Tagasiside käigus olid aga avastatud ning likvideeritud mitmed funktsionaalsed probleemid.

Rakenduse testimiseks kirjutas autor ka automaattestid, mis lähtusid ühiktestimise põhimõtetest. Ühiktestimise käigus testitakse programmi alamprogramme või funktsioone [15]. Testid olid kirjutatud funktsionaalse testimise alusel ehk nende koostamisel oli aluseks spetsifikatsioon (ülesande kirjeldus), mitte kood ning testide koostamiseks kasutati ekvivalentsusklasside ning piirjuhtude meetodeid. Ekvivalentsiklasside idee on see, et sarnaste andmete puhul peaks programm ühte moodi käituma (kui ühe andmega tekib viga, tekib ka teisega). Piirjuhud seonduvad ekvivalentsiklassidega. Piiridel esineb rohkem vigu seega tuleb nendele eraldi testid kirjutada. [16]

Programmi testimiseks kasutas autor ka valge kasti ja musta kasti meetodeid. Valge kasti meetod lähtub sellest, et testija tunneb testitava tarkvara sisemist ülesehitust ja tööloogikat. Nii testis autor näiteks uue sukeldumise lisamist, kus ta proovis läbi erinevad sisendid nii õigete kui ka puudulike ja vigaste andmetega. Musta kasti meetod lähtub sellest, et testija ei tea, mis programmis sees on. Testjuhtumid koostatakse musta kasti meetodi puhul peamiselt kasutuslugude põhjal. [17]

Programmi teksti testimiseks ja puhastamiseks kasutas autor erinevaid tehnikaid:

1. Lause adekvaatsuse kriteerium – iga koodi lause peab vähemalt ühe korra töötama
2. Andmetepõhine testimine – testid tehakse andmestruktuuride baasil
3. Tsüklite testimine – tugineb tsükli korduste arvule, testitakse minimaalse ja maksimaalse arvu ümber.

6 Rakenduse võimalikud edasiarendused

Kui piisavalt paljud sukeldujad rakendust kasutama hakkavad saaks lisada rakendusse sukeldumisklubide loomise võimaluse. See võimaldaks klubidel infot enda kohta jagada ning sukeldumisreisidele sukeldumismallid ette teha. Samuti oleks nii kasutajal lihtsam enda tuttavaid jälgima hakata, sest klubi kaudu on arvatavasti kergem teist kasutajat leida.

Saaks lisada ka professionaalse instruktori märke profiilile vastava astme tõendava dokumendi esitamisel, kes saaks sukeldumisetele isikliku allkirja või templi panna. See tõstaks rakenduse usaldusväärsust sukeldujate maailmas.

Veel üks võimalik edasiarendus oleks luua rakendusele veebi versioon või koduleht ning laieneda ka Android operatsioonisüsteemi kasutatavate seadmete turule. See võimaldaks laieneda palju suuremale turule, sest 71% telefonidest kasutavad Android operatsioonisüsteemi [3].

Saaks ka lisada sotsiaalmeedia ühenduvuse ning automaatse sõprade leidmise võimaluse. See võimaldaks kasutajatel leida oma sõpru palju lihtsamalt. Samuti saaks sellega koos lisada sotsiaalmeediasse otse postitamise võimaluse. Nii saaksid inimesed enda sukeldumislogi kirjeid jagada sotsiaalmeedias ja see tõstaks rakenduse populaarsust.

Rakenduse kasutajaliidest saaks parandada nii kasutajamugavuse, intuitiivsuse kui ka kujunduse poole pealt.

7 Kokkuvõte

Käesoleva lõputöö eesmärgiks oli luua sukeldumislogi rakendus iOS platvormile, mis kasutab andmete pilve salvestamist, milles on implementeeritud gruppide süsteem, läbi mille on lihtsustatud sukeldumiste logimine ning milles on teiste kasutajate logide vaatamise võimalus.

Antud töö käigus valmis iOS rakendus, mis vastab eelmises lõigus seatud kriteeriumitele. Rakenduses on peale nende võimaluste ka seadete leht rakenduse seadistamiseks ning võimalus mitme keele vahel valida. Sellega on töö eesmärk täidetud.

Töö esimeses osas viis autor läbi analüüsi ning pani paika rakenduse tehnilised ning funktsionaalsed nõuded. Tänu sellele töötati välja 3 erinevat skeemi ja koostati prototüüp. Nende abil arendati välja rakendus.

Töö koostamisel sai autor palju teadmisi iOS platvormile rakenduste arendamisest ning üldiselt nutitelefonidele rakenduste programmeerimisest. Konkreetsemaid teadmisi omandas autor XCode keskkonna kasutamise ning Swifti programmeerimiskeele kohta.

8 Kasutatud kirjandus

[1] Wagner, M., Stieber, A, Jöbsti, E., Azzopardi, E., Stoianova, M. ja Sayer, M. (2013). Dive computers: the need for validation and standards. Kasutatud: 10.04.2021, https://bit.ly/2SfsJpq

[2] Curry, D. Strava Revenue and Usage Statistics (2021). Kasutatud: 13.04.2021, https://www.businessofapps.com/data/strava-statistics/

[3] O'Dea, S. Mobile operating systems' market share worldwide from January 2012 to January 2021 (2021). Kasutatud: 14.04.2021, https://bit.ly/3vwQEz8

[4] The Manifest. Android vs iOS Which Platform to Build Your App for First (2018). Kasutatud: 14.04.2021, https://bit.ly/3t48yHM

[5] Kieran, D. The Size of The Scuba Diving Industry (2019). Kasutatud: 14.04.2021, https://bit.ly/3vxuf4n

[6] TLU õppematerjalid. Prototüüpimine multimeediumitoodete loomisel. Kasutatud: 17.04.2021, https://bit.ly/3uVwoaf

[7] BuildFire. Objective C vs. Swift: Which is Better? (A Definitive Guide). Kasutatud: 20.04.2021, https://buildfire.com/objective-c-or-swift/

[8] Karczewski, D. Swift vs Objective-C: Which Should You Pick for Your Next iOS Mobile App (2020). Kasutatud: 20.04.2021, https://bit.ly/3t20Z47

[9] Solt, P. Swift vs Objective-C: 10 reasons the future favors Swift (2015). Kasutatud 20.04.2021, https://bit.ly/3t5LEQ2

[10] O'Dea, S. Number of iPhone users in the United States from 2012 to 2022 (2021). Kasutatud 05.05.2021, https://bit.ly/3eDP9s4

[11] „SwiftUI. Better apps. Less code.“ Kasutatud: 05.04.2021, https://developer.apple.com/xcode/swiftui/

[12] Swifti dokumentatsioon. Kasutatud: 05.04.2021 https://swift.org/documentation/

[13] Manning, J., Buttfield-Addison, P. ja Nugent, T. „Learning Swift: Building Apps for MacOS, iOS and Beyond“ (2017)

[14] Hudson, P. What is MVC? Kasutatud: 14.05.2021 https://bit.ly/3frtzY7

[15] Ühiktestimine. Kasutatud: 25.04.2021, https://www.wikiwand.com/et/%C3%9Chiktestimine

[16] TLU õppematerjalid. Tarkvara testimine. Kasutatud: 15.05.2021, https://bit.ly/33MYf0i

[17] E-Õppe arhiiv. Testimise tüübid. Kasutatud: 16.05.2021, https://bit.ly/3eRWvcz

Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

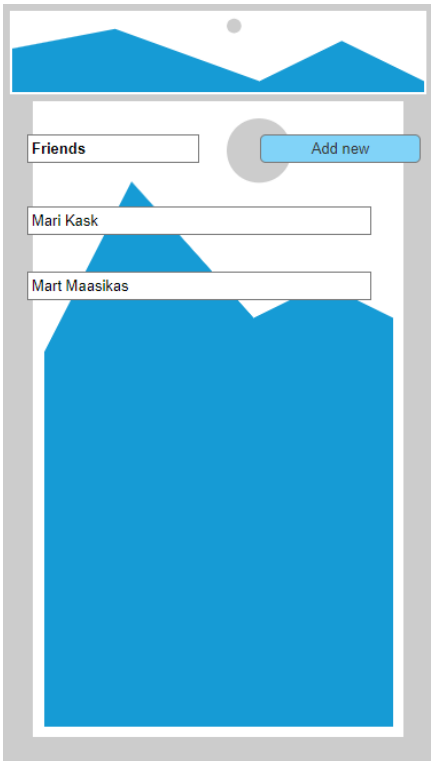
Mina, Artur Thomas Laherand

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "Sukeldumislogi rakenduse arendamine iOS platvormile" , mille juhendaja on Einar Kivisalu
   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.

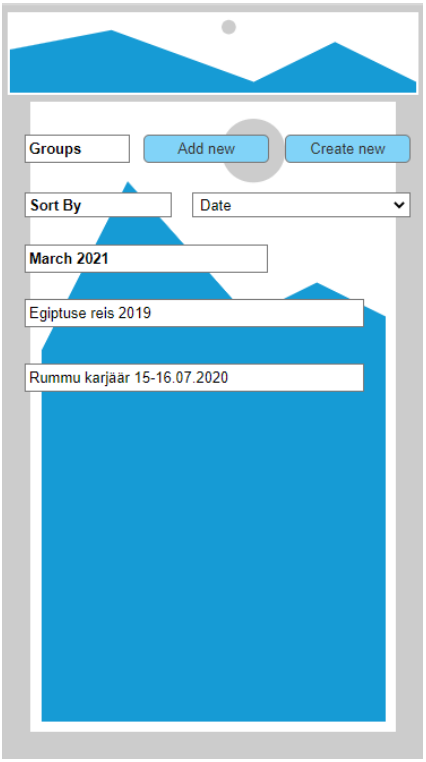
17.05.2021

Lisad

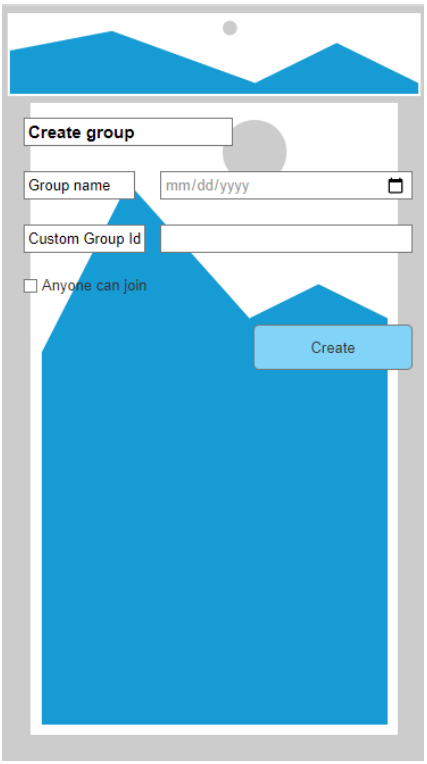
Rakenduse prototüüp ja mõned näited selle vaadetest:



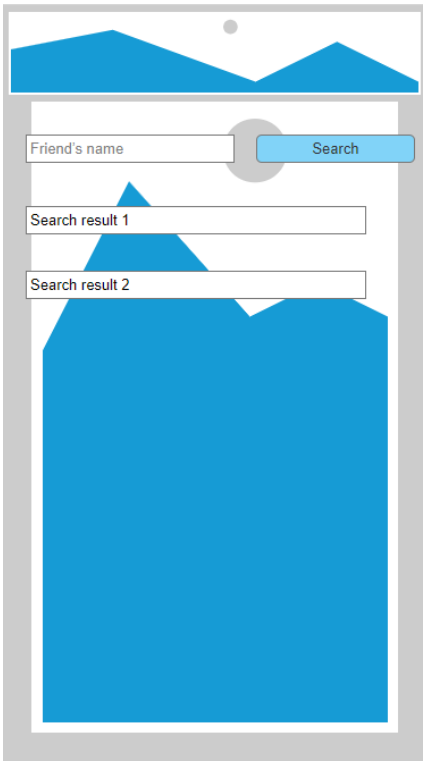
Joonis 18. Rakenduse prototüübi kasutaja sõprade vaade



Joonis 19. Rakenduse prototüübi kasutaja gruppide vaade



Joonis 20. Rakenduse prototüübi grupi loomise vaade.



Joonis 21. Rakenduse prototüübi sõprade otsimise vaade.

Rakenduse täielik prototüüp on leitav aadressil https://yfvskt.axshare.com.

Rakenduse kood asub aadressil https://www.github.com/arturlaherand/iDive.