TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Informaatikainstituut

IDU40LT

Allar Viinamäe 134302IAPB

TALLINNA TEHNIKAÜLIKOOLI TANTSUTÜDRUKUTE EESMÄRKIDE JA SAAVUTUSTE HALDAMISE SÜSTEEMI ARENDUS JA HÜBRIID MOBIILIRAKENDUS

Bakalaureusetöö

Juhendaja: Gunnar Piho
PhD
Dotsent

Tallinn 2016

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: Allar Viinamäe

23.05.2016

Annotatsioon

Lõputöö eesmärgiks on arendada Tallinna Tehnikaülikooli Tantsutüdrukutele ja Saltopoistele tulemuste ja eesmärkide haldamise süsteem. Süsteem on lahendatud hübriid mobiilirakendusena, mis võimaldab katta ühise koodibaasiga enamuse Android ja iOS operatsioonisüsteemidel töötavatest seadmetest. Hübriid mobiilirakenduse arendamiseks kasutati Ionic 2 raamistikku. Rakendus võimaldab kasutajal sisestada isiklikke eesmärke ja nende täitmisel salvestada sportlikke tulemusi ehk saavutusi fotojäädvustuse kujul. Süsteemi kasutajad jagunevad rühmadesse. Rühmaga liitunud kasutajale avanev rühmavaade annab ülevaate rühmaga liitunud kasutajate kogupunktidest, järjestades kasutajad edetabelisse. Rühmavaade näitab lisaks viimase poole aasta statistikat rühma saavutustest joongraafiku kujul.

Töö annab ülevaate hübriid mobiilirakenduse arendamiseks vajalikest tehnoloogiatest ja nende võrdlusest. Autor kirjeldab hübriid mobiilirakendusele valitud arhitektuuri ja valminud rakenduse kasutusjuhte.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 41 leheküljel, 3 peatükki, 13 joonist, 1 tabelit.

Abstract

Development of a management system for goals and results for Tallinn University of Technology Cheerleaders and a hybrid mobile application

The outcome of this thesis is a goals and results management system for Tallinn University of Technology cheerleaders. The system was developed as a hybrid mobile application, which allowed to cover the majority of Android and iOS based devices with a single code base. The hybrid mobile application development framework used was Ionic 2. The developed application allows the user to enter personal sports related goals and upon achieving those goals save them in the form of an image with the possibility of choosing an additional descriptive title. Users of the system are divided into teams. After joining a team, the team view gives an overview of the joined team members by ranking them in the scoreboard. The team view also displays team's statistics of the last six months by team's total achievements in the form of a line graph.

The thesis gives an overview of the developed hybrid mobile application and used technologies. Also, comparison of possible technologies to develop hybrid mobile applications is provided. The author gives an overview of the chosen architecture for the hybrid mobile application and describes the developed application's use cases.

The thesis is in Estonian and contains 41 pages of text, 3 chapters, 13 figures, 1 table.

Lühendite ja mõistete sõnastik

Angular 2	Dünaamiliste veebirakenduste loomiseks mõeldud eessüsteemi raamistik, mis võimaldab kirjeldada rakenduse komponente selgelt ja lühidalt, laiendades HTML-i ja JavaScripti süntaksit.
PostgreSQL	Vaba lähtekoodiga objekt-relatsiooniline andmebaasisüsteem.
MVC	*Model-view-controller*, tarkvara arhitektuuri muster, mida kasutatakse kasutajaliidese implementeerimisel. Jagab tarkvara arhitektuuri kolmeks eraldiseisvaks komponendiks (andmemudel, vaade ja kontroller).
Git	Hajutatud versioonihaldustarkvara.
Spring Boot	Infosüsteemide ja rakenduste loomiseks mõeldud raamistik, mis seob ja lihtsustab Javal põhinevate komponentide kasutamist. Võimaldab projekti ülesseadmisel mugavalt siduda projektiga erinevaid populaarsemaid teeke ja arendusvahendeid, eesmärgiga vältida üleliigset konfigureerimist ja jooksutada rakendust võimalikult lühikese ajaga.
API	*Application Programming Interface*, rakendusliides, programmiliides - protokollid ja vahendid, mille alusel rakendus kasutab teise rakenduse teenuseid.
XML	*Extensible Markup Language,* märgistuskeel, andmevahetusformaat, mille eesmärgiks on info jagamine erinevate infosüsteemide vahel.
JSON	*JavaScript Object Notation*, andmevahetusformaat, alternatiiv XML-ile.
HTML	*HyperText Markup Language*, hüpertekst-märgistuskeel, enimlevinud kodeerimissüsteem (tekstivorming) veebidokumentide loomiseks.
CSS	*Cascading Style Sheets*, kaskaadlaadistik, veebilehtede valmistajatele ja kasutajatele mõeldud laadistik.

Sass	*Syntactically Awesome Style Sheets*, CSS-i laiendav kaskaadlaadistik, mis lisab programmeerimiskeelele omaseid võimalusi tavapärasele CSS laadistikule (näiteks muutujad).
REST	*Representational State Transfer*, tarkvara arhitektuuri stiil, koosneb veebiteenusele määratud juhenditest ja reeglitest, eesmärgiga toimida veebis parimate jõudluse, mastaapsuse ja modifitseeritavate tulemustega.

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Käesoleva bakalaureusetöö eesmärk on arendada Tallinna Tehnikaülikooli Tantsutüdrukutele ja Saltopoistele tulemuste ja arengu jälgimise süsteem, luua hübriidne mobiilirakenduse prototüüp ja anda ülevaade selleks vajaminevatest vahenditest. Seni on treeneritel kasutusel olnud *Microsoft Exceliga* loodud tabelid, kuid nende täitmine on ajapikku muutunud tülikaks ja ebamugavaks. Mobiilirakenduse peamiseks eesmärgiks on luua mugav vahend treeningrühma tulemuste haldamiseks ja arengu jälgimiseks.

1.1 Taust ja probleem

Tallinna Tehnikaülikooli Tantsutüdrukute ja Saltopoiste organisatsiooni liikmete arv on viimaste aastatega jõudsalt kasvanud. See on tekitanud vajaduse moodustada erinevate tasemete ja eesmärkidega treeningrühmad. Treenerid on treeningrühmade arengu jälgimiseks ja tulemuste haldamiseks võtnud kasutusele *MS Exceliga* loodud tabelid. Tabelitesse teevad sissekandeid treenerid ja treeningrühmade liikmed kollektiivselt. Tabelite eesmärk on ühelt poolt motiveerida rühmaliikmeid oma varasemaid tulemusi ja treeningkaaslasi ületama, soodustades sellega treeningrühmade arengut. Teiselt poolt annavad sportlike saavutustega ja liikmete individuaalse infoga täidetud tabelid treeneritele väärtuslikku infot, mille alusel treeninguid läbi viia. Vastavalt rühma tasemele ja arengu tempole saavad treenerid seada rühmale uusi väljakutseid ja planeerida arengukava.

1.2 Ülesande püstitus ja eesmärk

Arenduse tulemusena valmib treeneritele ja liikmetele mõeldud mobiilirakenduse prototüüp, mille eesmärk on eelkõige vähendada aega, mis kulub treeneritel ja liikmetel arengu jälgimiseks ja tulemuste haldamiseks. Rakendus koondab ja lihtsustab info kättesaadavust ning loob ka uue võimaluse tulemuste sisestamiseks mobiilse seadmega fotojäädvustuste kujul. Autor annab lisaks ülevaate ka mõningatest hübriid mobiilirakenduste arendamiseks mõeldud vahenditest, tehnoloogiatest ja arhitektuurist ning kirjeldab tervikliku süsteemi arendamiseks vajaminevaid komponente.

2 Vajalikud alusteadmised

2.1 Hübriid mobiilirakendus

Tänapäeva maailmas, kus erinevad tootjad uuendavad pidevalt oma mobiilseid operatsioonisüsteeme ja seadmeid ning lisavad neile uusi unikaalseid võimalusi, on tekkinud mobiilsete rakenduste arendajatel omaette eesmärk pakkuda klientidele kõige värskemaid nutitelefonide tehnoloogilisi võimalusi. Selleks, et katta ühise koodibaasiga loodud rakendusega suurem osa mobiilsete operatsioonisüsteemide turust (Android, iOS, Windows Phone) ja samal ajal pääseda ligi seadmetel kasutatavatele tarkvarast sõltuvatele komponentidele, näiteks kaamera, graafika API-le ja multipuutele, on viimastel aastatel hakanud hoogsalt arenema hübriidsete mobiilirakenduste arendus.

Hübriidsete mobiilirakenduste arendamiseks mõeldud raamistikud ühendavad endas võimalust arendajatel kasutada neile juba tuntud tüüpiliste veebirakenduste arendamiseks vajalikke vahendeid (HTML, CSS, JavaScript [1]) ning ühtlasi pääseda ligi ka nutitelefonidele endile. Lisaks näevad hübriidrakendused välja ja tunduvad käes kui funktsionaalsed (ingl *native*) mobiilirakendused.

Hübriid mobiilirakendus on erinevalt veebirakendusest mobiilse seadme operatsioonisüsteemist ja sellele mõeldud arendusvahenditest sõltuv.

Rakenduse iOS versiooni arendamiseks sõltumata lisaarendusvahenditest (näiteks programmeerimiskeskkond) on vaja:

- Soovitavalt omada Mac OS X operatsioonisüsteemiga arvutit. Võimalik on kasutada ka Mac OS X operatsioonisüsteemiga virtuaalmasinat.
- Lähtekoodi kompileerimiseks Xcode [2] programmeerimiskeskkond.

Rakenduse publitseerimiseks iOS mobiilirakenduste veebipoes App Store on vajalik soetada ka Apple arendaja litsents [3].

Rakenduse Android versiooni arendamiseks on vaja vastava Android versiooni kompileerimiseks mõeldud Androidi arendustarkvara. Publitseerimiseks Androidi veebipoodi Play Store on vajalik soetada Google arendaja litsents [4].

2.2 Kasutajaliidese arendus

Lubades mistahes rakenduse kasutajal näha, muuta, luua ja kustutada andmebaasis olevaid andmeid võib luua rakendusele kasutajaliidese, kuid sellise mõttega loodud kasutajaliides ei ole tihtipeale see, mida kasutaja tegelikult vajab. Eelkõige vajab kasutaja hea kasutatavusega kasutajaliidest, mis ei ajaks kasutajat segadusse. *The Virtual Window* meetod on süstemaatilise printsiibiga lähenemine kasutajaliidese arendusele, mis soovitab esmalt välja mõelda, millised vaated võiks loodaval süsteemil olla ja kuidas nendes vajalikku infot kajastada. Selliseid vaateid nimetab kirjeldatud meetodi peamine looja Soren Lauesen virtuaalseteks akendeks (ingl *Virtual Windows* [5]). Käesolevas töös alustatakse rakenduse arendust kasutajaliidese väljamõtlemisest, lähtudes virtuaalsetest vaadetest, eesmärgiga luua võimalikult mugav ja arusaadav kasutajaliides.

Eelnimetatud meetodi järgi luuakse esmalt nimekiri ülesannetest mida kasutaja ja süsteem koos teevad. Seejärel luuakse ülesannete nimekirja põhjal nimekiri andmetest, mis toetavad kirjapandud ülesannete täitmist.

Käesoleva bakalaureusetöö raames valmiva prototüübi ülesannete nimekiri:

- Võimalus lisada isiklik saavutus
- Võimalus lisada isiklik eesmärk
- Võimalus luua treeningrühm
- Võimalus märkida isiklik eesmärk täidetuks
- Võimalus märkida treeningutel kohalolijaid päeva alusel
- Treeneril peab olema võimalus määrata temale kuuluva rühma liikmeid kaastreeneriteks

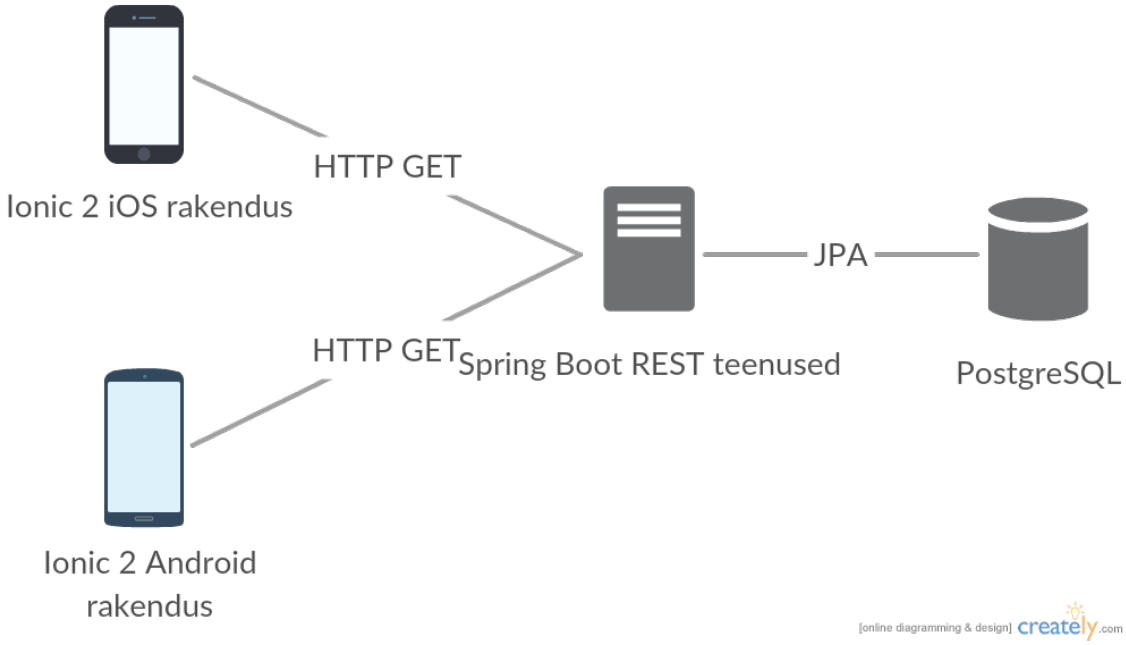
Ülesannete nimekirja toetavad vajalikud andmed:

- Saavutused
- Eesmärgid
- Kasutaja andmed
- Rühmad

Andmete ja ülesannete nimekirja põhjal kirjeldab autor vajaminevaid vaateid:

- Edetabeli vaade
- Isiklike eesmärkide vaade
- Isiklike saavutuste vaade
- Isikliku profiili vaade
- Rühma loomise vaade
- Rühmaga liitumise vaade
- Rühma liikmete vaade kohalolijate märkimiseks ja õiguste muutmiseks

2.3 Arhitektuur



Joonis 1: Valminud rakenduse loogiline arhitektuur

Rakenduse jaoks valitud loogiline arhitektuur (Joonis 6) võimaldab kasutajate andmeid säilitada väljaspool seadme lokaalset mälu ja neid vajadusel teiste kasutajatega jagada.

Autor valis Spring Boot raamistiku turvaliseks vahekihiks PostgreSQL-i ja Ionic 2 [6] rakenduse vahele, vältimaks otseühendust andmebaasi ja kliendipoolse rakenduse vahel. Suhtlus Spring Bootiga loodud tagakomponendi ja Ionic 2 rakenduse vahel toimub autori kirjutatud REST teenuste abil.

Relatsiooniliste andmete töötlemiseks ja haldamiseks on tagakomponendis kasutusel Java Persistence API (JPA) [7]. JPA võimaldab arendajal andmebaasi tabeleid

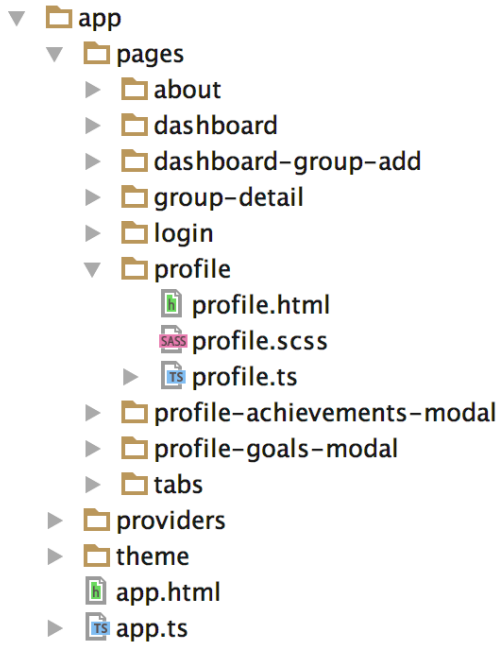
defineerida läbi Java objektide, kiirendades sellega andmebaasi loomist näiteks väiksemate mobiilirakenduste puhul.

2.3.1 Eeskomponendi arhitektuur ja kataloogide struktuur

Eeskomponendi (ingl *front end*) lähtekood vajab hallatavuse ja viimistletuse huvides kataloogide struktuuri, mis ka pikemas perspektiivis soodustaks võimalikult produktiivset jätkuarendust.

Angular 2 raamistikus kasutatakse vaadete kuvamiseks HTML malle ja kogu nendega seotud loogika kirjutatakse JavaScriptis [1]. Angulari arhitektuur on oma olemuselt moodulipõhine, mis tähendab, et rakendus üritatakse jaotada võimalikult mugavateks mooduliteks ja uued komponendid kirjutatakse kasutades vajaminevaid mooduleid. Komponendid omakorda kontrollivad vaateid tervikuna või osaliselt. Komponente luuakse ja uuendatakse, või hävitatakse vastavalt rakendust kasutava kasutaja otsustele. Lisaks eristatakse Angularis ka teenuskihi elemente, mis vastutavad näiteks väliste teenustega suhtlemise või rakenduse konfigureerimise eest [8].

Erinevalt tüüpilisest MVC põhisest kataloogide struktuurist, kus mudelid, vaated ja kontrollerid jaotatakse vastavatesse kataloogidesse, jaotatakse valminud rakenduse osad komponendipõhiselt kataloogidesse. Ühes komponendi kataloogis sisaldub üldiselt HTML vaade ehk mall, vaadet kontrolliv JavaScripti komponent ja vaate disaini sisaldav Sass [9] laadistik. Selline lähenemine hoiab komponente iseseisvamatena ja vähendab vajaliku komponendi otsimiseks vajaminevat aega.



Joonis 2: Eeskomponendi kataloogide struktuur valminud hübriidrakenduses

2.3.2 Tagakomponendi arhitektuur ja testjuhitud arendus

Käesoleva bakalaureusetöö raames valminud hübriidrakenduse tagakomponent (ingl *back end*) on andmebaasi ja kliendipoolse eeskomponendi vaheline kiht, mis vastutab turvalise andmeliikluse ja -halduse eest.

Spring Boot raamistikul põhineva tagakomponendi eraldatus eeskomponendist võimaldab arendajal vahetada või integreerida mistahes kliendipoolne eeskomponent tagakomponendiga. See teeb omakorda tulevikus ajaliselt odavamaks amortiseerunud eeskomponendi vahetuse ja võimaldab juba olemasolevat andmestikku kasutada uute rakenduste loomiseks.

Loodud tagakomponent kujutab endast REST teenustel põhinevat API-t. Teenuste ja eeskomponendi vaheliseks andmevahetusformaadiks on valitud kergekaaluline JSON, mis kujutab endast JavaScripti objekti ja liidestub seega mugavamalt eeskomponendis kasutatava Angulariga kui XML. Springi REST teenuste kontrollerid tagastavad Java objekte automaatselt JSON formaadis, kasutades selleks Jackson JSON [10] teeki. REST teenused on arhitektuuriliselt lihtsad ja inimloetavad ning annavad autorile võimaluse hoida tagakomponendi kihti võimalikult õhukesena ja seega hästi testitavana.

Tagakomponendi REST teenuste arendamisel lähtuti testjuhitud arenduse (ingl *Test-Driven Development*) põhimõtetest:

- Arendaja ei kirjuta ühtegi uut rida koodi enne kui on kirjutatud läbikukkunud automaattest [11].

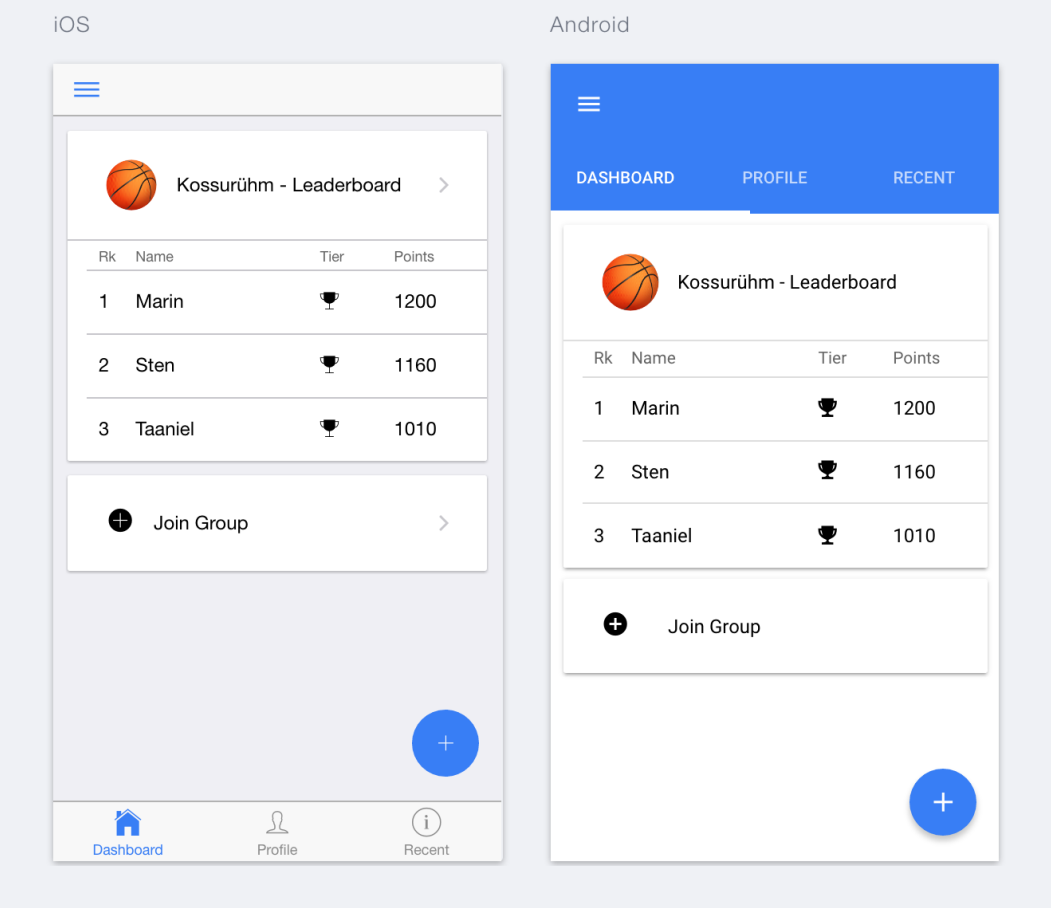
- Duplikaatkoodi elimineerimine [11].

Selline lähenemine annab programmeerimisel saadava tagasiside üle parema kontrolli ja väldib mittetöötava koodi kirjutamist.

2.4 Rakenduse teostuse vahendid

2.4.1 Ionic 2

Valminud hübriidrakenduse loomisel kasutatud Ionic 2 [6] raamistik võimaldab rakenduse kirjutamisel jooksvalt jälgida rakenduse arengut. Selleks, et kirjutada rakendust ja samaaegselt jälgida veebilehitsejas (Joonis 2) või emulaatoris rakenduses toimunud muudatusi käivitab Ionic lokaalse arendusserveri. Ionic 2-e meeskond nimetab sellist funktsionaalsust *LiveReloadiks*, mis lubab failide salvestamisel automaatselt värskendada rakenduse vaadet veebilehitsejas kui ka mobiiliseadmete emulaatoris. Programmeerimisel tekkivate vigade avastamisele aitab hästi kaasa ka *LiveReloadi* võimalus suunata serveri ja rakenduse logid otse käsureale, kust rakendus käivitati.



Joonis 3: Ionic võimaldab jälgida jooksvalt iOS ja Android platvormidel rakenduse arengut

2.4.2 Programmeerimiskeskkond ja testimine

Rakendust arendati integreeritud programmeerimiskeskkonnas IntelliJ IDEA [12]. Rakenduse Android versiooni testimiseks kasutati *Android Virtual Device* emulaatorit Samsung Galaxy S6. Rakenduse iOS versiooni testimiseks kasutati iOS simulaatorit ja reaalset seadet iPhone 6s.

Reaalsel seadmel - iPhone 6s - testimisel on oluline, et tagakomponendi vastu HTTP päringute tegemisel oleks päringute aadressiks määratud masina IP aadress, kus tagakomponent jookseb, sest reaalne seade ja masin, kust rakendus kompileeritakse, omavad erinevat IP aadressi. Lisaks on oluline, et masina IP aadress oleks seadmelt kättesaadav, olles näiteks samasse võrku ühendatud. Muutuvast IP-st tingitud päringu baasaadressi pideva muutumise probleemi lahendab autor kasutades eeskomponendis konstantset baasaadressi *BASE_URL* väärtust.

3 Lahenduse kirjeldus

3.1 Rakenduse mittefunktsionaalsed nõuded

Tüüp	Nõude kirjeldus
Keel	Hübriidrakenduse kasutajaliides peab olema inglise keeles, kuid rakendus tuleks üles ehitada nii, et uute keelte lisamine ei nõuaks suuri arhitektuurilisi ümbermuudatusi (näiteks eesti keele lisamine).
Kasutajaliides	Töötavas rakenduses peab kasutajaliides olema mobiili/tahvelarvuti põhine.
Töökiirus	Rakenduse töökiirus on sõltuv kasutaja interneti kiirusest, kuid keskmise internetiühenduse juures ei tohi vaadete kuvamine ja andmetega täitmine aega võtta üle 5 sekundi.
Töökindlus	Valminud tagakomponendi tõrgeteta töö on vajalik kasutajate eesmärkide saavutamiseks. Tõrked tekitaksid ebamugavusi rakenduse kasutamises nii treeneritele kui ka rühmaliikmetele. Taasteaja siht (ingl *recovery time objective*) – (“maksimaalne talutav (lubatav) ärifunktsiooni, ressursi või süsteemi käideldamatuse kestus pärast intsidenti” (AKIT, 2016) [13]) - juhul kui tekib veaolukord ja andmebaas või tagakomponendi rakendus kahjustub, siis tuleb need taastada viimase tehtud varukoopia põhjal tunni jooksul peale rikke põhjuse kõrvaldamist ja serveri töökorda saamist. Taasteseisu siht (ingl *recovery point objective*) – (“üks intsidendijärgsele taastele seatud eesmärke: andmete taaste kestusena või säilinud andmete vanusena väljendatav lubatav andmete kadu; määrab varukopeerimise minimaalse lubatava sageduse; aeg, millele eelnevad andmed peavad olema täielikult taastatud (näiteks eelmine tund, eelmine tööpäev, eelmine nädal” (AKIT, 2016)) [14] - maksimaalselt võivad kaduma minna viimase tunni andmed, seega sellele eelnevad andmed peavad olema täielikult taastatud.
Turvalisus	Kasutaja profiiliandmete turvalisus sõltub suures osas Facebooki turvalisusest. Eeskomponendi ja tagakomponendi vaheline suhtlus peab toimuma üle turvalise HTTPS protokolli.
Testitavus	Testidega peavad olema kaetud järgmised tagasüsteemi komponendid: • *UserController* – kasutajaga seotud andmete pärimise teenus. • *DashboardController* - armatuurlaual rühmadega seotud päringute teenus.

	• *ProfileController* - kasutaja profiiliga seotud päringute teenus. Tegeleb peamiselt eesmärkide ja saavutustega seotud ressursi -ja salvestamispäringutega. • *TeamController* - põhiline rühma vaatega seotud teenus.

Tabel 1: Olulisemad mittefunktsionaalsed nõuded

3.2 Rakenduse funktsionaalsed nõuded

3.2.1 Kasutusjuhud

Kasutusjuht: Kasutaja autentimine

Tegutsejad: Rühmaliige, Treener
Lühikirjeldus: Süsteem autendib subjekti kasutades Facebooki konto andmeid. Kui subjekt on identifitseeritud, siis suunatakse subjekt süsteemi avavaatele (Joonis 2).

Kasutusjuht: Uue rühma loomine

Tegutsejad: Rühmaliige, Treener
Lühikirjeldus: Subjekt loob rühma, millega teised subjektid liituda saavad. Kui subjekt loob rühma, siis muudab süsteem subjekti oleku rühma siseselt "treeneriks".

Kasutusjuht: Rühmaga liitumine

Tegutsejad: Rühmaliige, Treener
Lühikirjeldus: Subjektil peab olema võimalik valida nimekirjast rühm, millega soovi korral liituda. Pärast rühma valimist seob süsteem rühma identifikaatori kasutajaga.

Kasutusjuht: Isiklike saavutuste lisamine

Tegutsejad: Rühmaliige, Treener
Lühikirjeldus: Subjektil peab olema võimalik pildistada ja salvestada sportlik saavutus. Saavutusele saab lisada ka kategooria ning nimetuse. Süsteem loob uue saavutuse või seob olemasoleva kasutajaga.

Kasutusjuht: Isiklike eesmärkide lisamine

Tegutsejad: Rühmaliige, Treener
Lühikirjeldus: Subjekt valib ja seejärel salvestab kategooria ja nimetuse alusel sportliku eesmärgi.

Kasutusjuht: Eesmärgi oleku muutmine

Tegutsejad: Rühmaliige, Treener
Lühikirjeldus: Eesmärgi täitnud subjekt muudab eesmärgi olekut. Täidetud eesmärgi olek on saavutus. Süsteem muudab eesmärgi oleku saavutuseks.

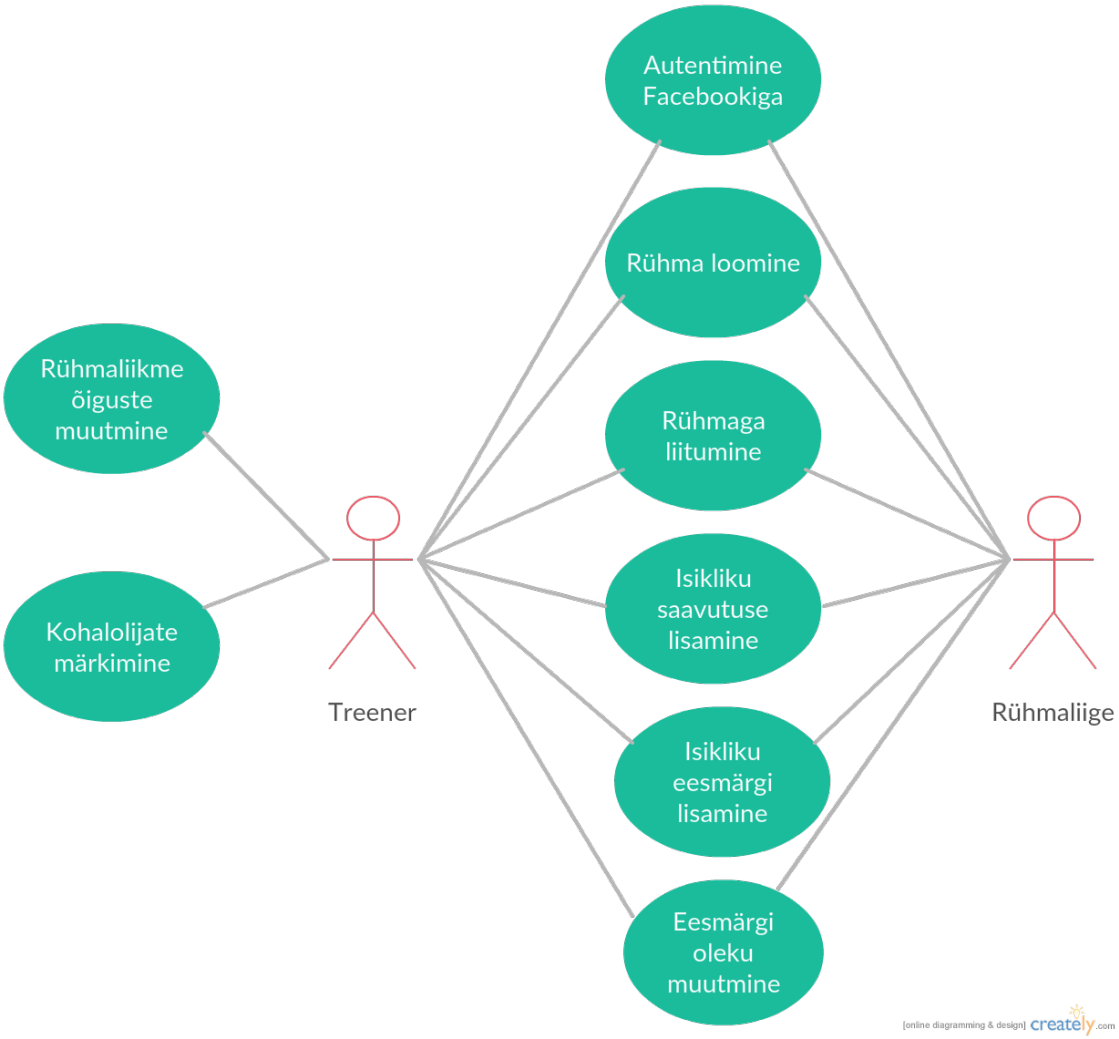
Kasutusjuht: Rühmaliikme õiguste muutmine

Tegutsejad: Treener
Lühikirjeldus: Rühmasiseselt süsteem eristab treeneri ja rühmaliikme rolli. Treener peab saama määrata rühmaliikme treeneriks.

Kasutusjuht: Kohalolijate märkimine

Tegutsejad: Treener

Lühikirjeldus: Subjekt peab saama rühma vaates avada nimekirja rühma liikmetest ja märkida kohalviibimist päeva alusel.



Joonis 4: Kasutusjuhtude diagramm

3.2.2 Kasutaja autentimine

Kasutaja autentimine on lahendatud kasutades sotsiaalvõrgustiku Facebook kasutajat ning vajalik informatsioon kasutaja kohta on salvestatud läbi tagakomponendi PostgreSQL andmebaasi.

Töö kirjutamise ajal polnud Ionic 2-e arendusmeeskond väljastanud ametlikku Facebookiga autentimise juhendit ega dokumentatsiooni. Sellest tulenevalt on rakenduses kasutaja autentimist võimaldav mehhanism sarnane vanema soovitusliku implementatsiooniga [15], kuid ümberkirjutatud uuema - Ionic 2 - tehnoloogia abil. Suhtlemist funktsionaalse Facebooki rakendusega ja autentimiseks vajalike teenuste kasutamist lihtsustab *Apache Cordova Facebook Plugin* [16]. *Apache Cordova Facebook Plugin* pistikprogrammi kasutamisel võrreldes JavaScripti baasil toimivate

teekidega (*CordovaOauth, OpenFB, Facebook SDK for JavaScript with AngularJS* [15]) ei tule nähtavale mobiilsele veebilehitsejale omased komponendid, näiteks veebiaadressi sisestamise jaoks mõeldud riba. Kasutajal on võimalus autentida mobiilirakendus ilma Facebooki kasutajanime ja parooli sisestamata, kui kasutajal on seadmesse funktsionaalne Facebooki rakendus juba installeeritud ja sinna ka sisse logitud. See on omakorda oluline kasutajakogemuse poolelt, kuna vähendab rakenduse kasutamiseks vajaminevaid samme.

Autentimise turvalisuse määrab suures osas Facebook. Kasutaja profiili kohta võib saada varjatud infot Facebooki poolt tagastatud turvamärgise äraarvamisel või Facebooki konto kasutajanime ja parooli varastamisel. Viimase korral seatakse ohtu kõik süsteemid, mis kasutavad Facebooki autentimise teenust.

3.2.3 Kasutaja staatuse tuvastamine

Rakenduse avamisel pöördub eeskomponent esmalt Facebooki serveri poole päringuga, mille vastus annab informatsiooni selle kohta, kas rakendust kasutav klient on Facebooki sisse loginud ja autoriseerinud parajasti kasutatava rakenduse. Päringu teostab *Apache Cordova Facebook Plugin* [16] meetodiga *.getLoginStatus(Function success, Function failure)*, mis kutsutakse välja autori kirjutatud komponendis *FbProvider*. Meetod tagastab JSON formaadis objekti *authResponse* juhul, kui kasutaja on sisse logitud. Parameetreid meetodile kaasa ei anta.

Võimalikud kasutaja staatused on:

1. *connected* - kasutaja on Facebooki sisse logitud ja rakendus on autoriseeritud. Kasutaja suunatakse rakenduse peavaatele [17].

2. *not_authorized* - kasutaja on Facebooki sisse logitud, kuid rakendus pole autoriseeritud [17].

3. *unknown* - kasutaja pole Facebooki või rakendusse sisse logitud ja pole teada kas rakendus on autoriseeritud. Kasutaja suunatakse sisse logimiseks mõeldud vaatele [17].

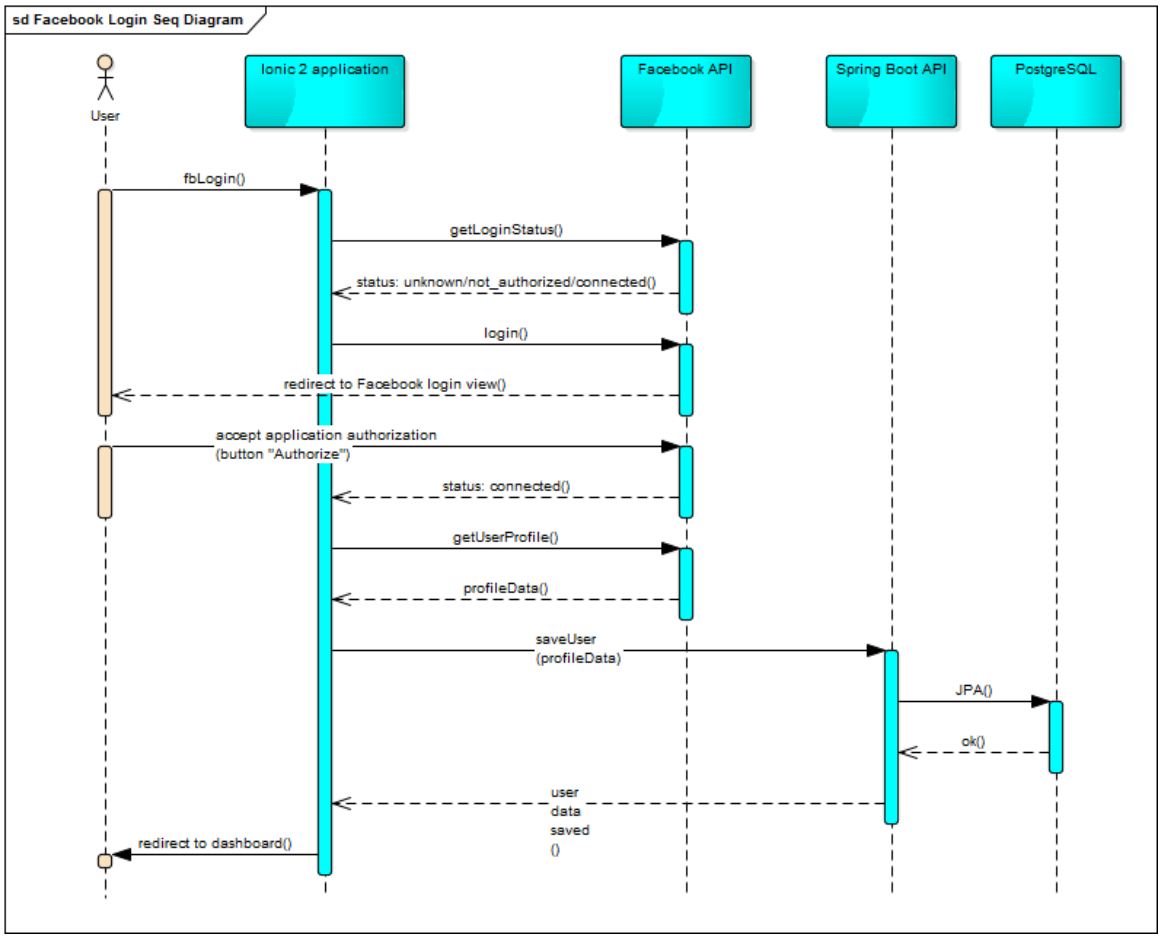
Näide juba sisse loginud kasutaja puhul objektist *authResponse* JSON formaadis:

```
{
  "status":"connected",
  "authResponse":{
    "session_key":true,
    "accessToken":"EAA...",
    "expiresIn":"5183853",
    "sig":"...",
    "userID":"127...",
    "secret":"..."
  }
}
```


Rakenduse edasine käitumine sõltub päringu vastusest.

3.2.4 Kasutaja sisse -ja väljalogimine

Kasutaja suunatakse sisselogimiseks mõeldud vaatele juhul, kui Facebooki API poolt tagastatud kasutaja staatus on *unknown*. Sealt avaneb kasutajal võimalus autentida ja siseneda Facebooki kasutajaga rakendusse vajutades vastavale nupule - “Log In with Facebook”. Nupule vajutades päritakse veelkord kasutaja olek, tegemaks kindlaks ega kasutaja pole vahepeal läbi Facebooki seadete kasutaja staatust muutnud või kasutatavat kontot vahetanud. Seejärel suunab Facebooki API kasutaja koos määratud õigustega rakendust autoriseerima ning viimaks päritakse Facebookilt kasutaja profiili andmeid, mis salvestatakse hilisemaks kasutuseks autori andmebaasi. Kasutaja andmete varundamine autori andmebaasi muudab hiljem rühma andmete sorteerimise mugavamaks, näiteks väljastades vajadusel treenerile kõikide rühma liikmete e-posti aadressid. Viimaks suunatakse kasutaja üldisele rühmade tulemuste lehele (Joonis 7).



Joonis 5: Edukas Facebookiga sisselogimine jadadiagrammina

Väljalogimiseks vajutab kasutaja nuppu “Logout”, misjärel küsitakse kasutaja nõusolekut ja kutsutakse välja meetod *.logout(Function success, Function failure)*. Meetod käitub vastavalt järgnevates olukordades:

1. Juhul, kui kasutaja on funktsionaalsesse Facebooki rakendusse sisse logitud, siis logitakse kasutaja välja ainult käesoleva töö raames arendatud rakendusest [18].

2. Juhul, kui kasutaja pole Facebooki rakendusse sisse logitud ja sisselogimisel valminud rakendusse logis kasutaja sisse ka Facebooki rakendusse, siis logitakse kasutaja välja nii Facebooki rakendusest kui ka valminud rakendusest [18].

3. Kasutaja logis mõnda teise rakendusse ja selle käigus Facebooki rakendusse. Seejärel logis kasutaja valminud rakendusse. Väljalogimisel kummastki rakendusest logitakse kasutaja välja ka Facebooki rakendusest [18].

Eduka väljalogimise tulemusena suunatakse kasutaja sisselogimisvaatele.

3.2.5 Tagakomponendist andmete pärimine

Andmete pärimine tagakomponendist toimub autori programmeeritud REST teenuste kaudu. Valminud REST teenuste eeskomponendipoolsed päringud peavad sisaldama Facebookiga autentimisel saadud turvamärgist (ingl *access token*). Turvamärgise kehtivust kontrollib tagakomponent luues ühenduse Facebookiga ja kontrollides turvamärgise kehtivust. Kui turvamärgis kehtib, siis sisalduvad eeskomponendi poolt tehtud ressursside päringu vastuses soovitud andmed.

Spring Social Facebook [19] pakub tagakomponendis Facebookiga ühenduse loomiseks abstraktset teenust *FacebookConnectionFactory*, mille eesmärk on lihtsustada Facebookiga ühenduse loomist nii, et arendaja ei pea kirjutama reaalset ühenduse konfiguratsiooni, avamist ja sulgemist. *FacebookConnectionFactory* võtab parameetritena kaasa Facebooki arendaja konsoolis defineeritud unikaalse raskenduse identifikaatori (ingl *App ID*) ja rakenduse salajase räsiväärtuse (ingl *App Secret*). Ühendus luuakse meetodiga *.createConnection()*, mis võtab parameetrina kaasa turvamärgise. Kui turvamärgis kehtib ja ühendus Facebookiga õnnestub, siis tagastatakse päritud andmed klientrakendusele.

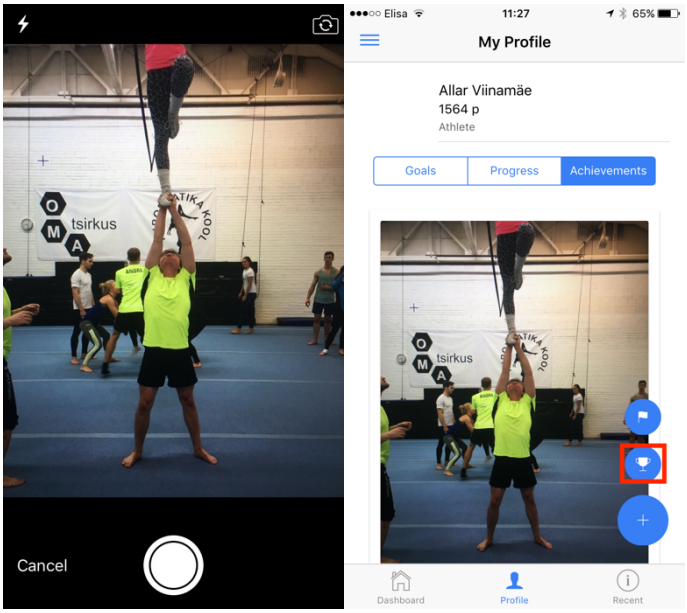
3.2.6 Isikliku saavutuse lisamine ja kuvamine

Autor defineerib "saavutust" kui sportlikku tulemust, mida kasutaja soovib infona talletada. Saavutused salvestatakse fotojäädvustuse kujul kasutaja profiili vaates rubriigi "Achievements" alla. Kasutaja punktiarvestuseks peab saavutusele olema lisatud ka nimetus ja kategooria.

Kasutaja saab oma saavutusi salvestada mistahes vaates, kasutades selleks materiaalsest disainist (ingl *Material Design*) tuntud ringikujulist kasutajaliidese komponenti *Floating Action Button* [20]. *Floating Action Button* (Joonis 8) koondab endas enimkasutatavaid kasutusjuhte ning on peaaegu alati kasutajale nähtaval kohal vaate alumises paremas nurgas.

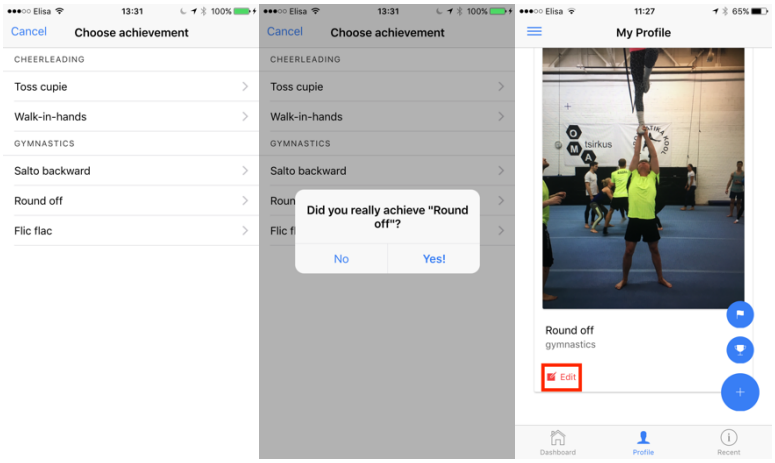
Seadme kaamera kasutamiseks on Ionic 2 rakendusse paigaldatud pistikprogramm *cordova-plugin-camera* [21], mis võimaldab kasutada JavaScripti baasil toimivat API-t seadmega piltide tegemiseks ja hiljem nendele ligipääsemiseks. Pildid saadetakse POST päringuga tagakomponendile ja salvestatakse kohaliku serveri failisüsteemi. Viide

üleslaetud faili asukohale failisüsteemis salvestatakse andmebaasi, vältimaks andmebaasi ummistumist suuremahuliste failidega.



Joonis 6: Pärast saavutuse pildistamist ja salvestamist kuvatakse see rubriigi "Achievements" alla

Pärast saavutuse pildistamist on kasutajal võimalik saavutusele juurde lisada ka saavutuse nimetus ja kategooria. Selleks vajutab kasutaja saavutuse all olevat nuppu "Edit", mille peale tehakse GET päring ja tagastatakse võimalikud saavutuste kategooriad ja nimetused. Avanevast moodulist saab kasutaja valida sobiva saavutuse nimetuse (Joonis 12). Nimetuse valimine on oluline punktiarvestuseks, sest punkte annavad ainult saavutused, millel on valitud nimetus. Igale saavutuse nimetusele on andmebaasis määratud vastavad punktid.



Joonis 7: Saavutuse pildistamise ja salvestamise järel on võimalik valida saavutusele vastav nimetus

Koodinäide Angular 2-s GET päringuga ressursside pärimiseks eeskomponendi teenuskihi komponendis *user-data.js*:

```
getAllAchievements() {
    return new Promise((resolve, reject) => {
        this.url = 'http://' + this.BASE_URL + ':8080/api/profile/goal'
            + '?facebookId=' + this.userId
            + '&token=' + this.userToken;
        this.http.get(this.url).subscribe(achievements => {
            resolve(achievements.json());
        }, error => {
            reject(error);
        });
    });
}
```

- `new Promise()` – Abstraktne objekt asünkroonsete teenuste kasutamiseks. Kui HTTP päring õnnestub, siis tagastatakse objekt koos väärtustega alles pärast `resolve(goals.json())` meetodi käivitamist ja tagastatavaks objektiks on `achievements`. Päringu ebaõnnestumisel tagastatakse objekti väärtus pärast `reject(error)` meetodi käivitamist ja tagastatava objektiks on `error`.

Kuna tagastatav objekt `achievements` on võti-väärtus JSON objekt (Lisa 3), siis tuleb Angular 2-s võtmed ja väärtused salvestada eraldi muutujatesse (`arrayOfKeys`, `arrayOfValues`):

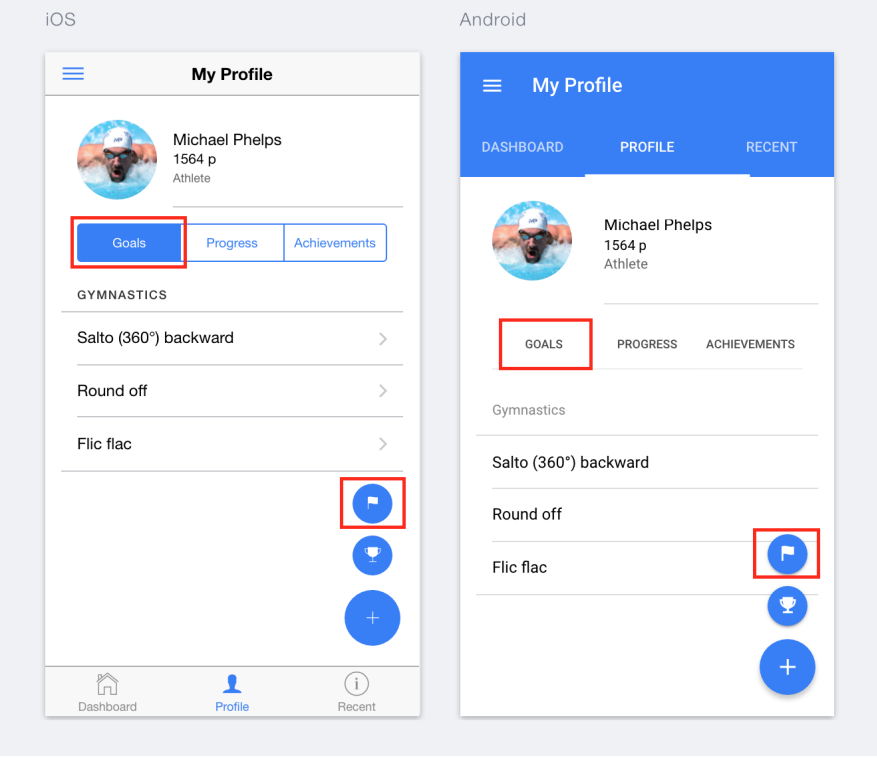
```
initAchievements() {
    this.userData.getAllAchievements().then((achievements) => {
        this.arrayOfKeys = Object.keys(achievements);
        this.arrayOfValues = Object.keys(achievements).map(key =>
achievements[key]);
    });
}
```

Seejärel on võimalik HTML vaates võtmeid ehk kategooriaid ja väärtusi ehk saavutusi itereerida ja kasutajale kuvada:

```
<ion-list>
        <div *ngFor='#category of arrayOfKeys; #i = index'>
            <ion-list-header>
                {{category}}
            </ion-list-header>
            <a ion-item *ngFor='#achievement of arrayOfValues[i]'
(click)="doConfirm(achievement.title)">
                {{achievement.title}}
            </a>
        </div>
</ion-list>
```

3.2.7 Isikliku eesmärgi lisamine ja oleku muutmine

Kasutaja saab eesmärke lisada mistahes vaates, kasutades selleks eelnevalt mainitud komponenti *Floating Action Button*. Punktiarvestuseks peab kasutaja määrama eesmärgi nimetuse ja kategooria. Eesmärke on võimalik salvestada ka eesmärkide loendist nimetust või kategooriat valimata, kuid sellisel juhul ei arvestata hiljem täidetud eesmärki punktiarvestuses. Kasutajal on võimalik märkida eesmärk sooritatuks kui eesmärk on täidetud. Eesmärk, mis on märgitud sooritatuks, muudab oleku "saavutuseks", millele kasutaja saab soovi korral lisada fotojäädvustuse täidetud eesmärgist.



Joonis 8: Peale eesmärgi lisamist kuvatakse need rubriigi "Goals" alla

3.2.8 Rühmade vaade ja liitumine

Rakendusse sisse logides avaneb esmalt kasutaja armatuurlaud (ingl *Dashboard*), mis kuvab kasutajale kõik rühmad, kuhu kasutaja end lisanud on (Joonis 2). Armatuurlaua vaatega seotud päringute eest vastutab tagakomponendis *DashboardController*.

Eeskomponendi GET näitepäring kõigi kasutajaga seotud rühmade pärimiseks:

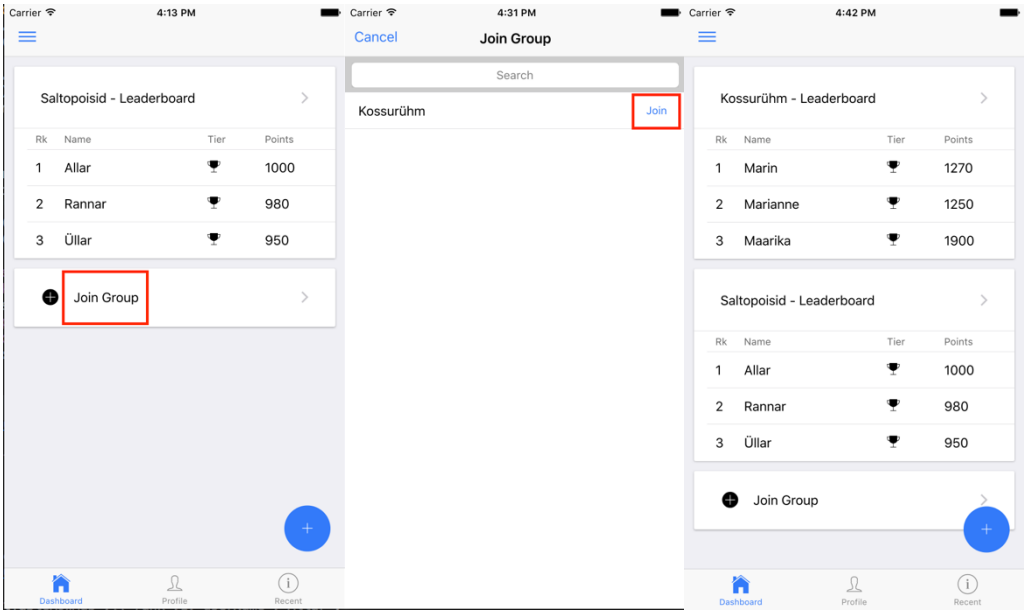
http://localhost:8080/api/dashboard?facebookId=127...&teamView=short&token=A...

- *facebookId* – kasutaja unikaalne identifikaator, mille järgi otsitakse kasutajale vastavavad andmed.

- *teamView – api/dashboard* teenus lubab valida, millisel kujul kasutaja rühmade andmed päritakse. Parameetri väärtus *short* tagastab lühendatud andmed rühmade kohta, kolme kõige rohkem punkte omavate rühmaliikme nimedega (Lisa 2).

- *token* – turvamärgis päringu autoriseerimiseks. Kehtetu turvamärgise korral tagastab tagakomponent tühja nimekirja ja HTTP staatuse 401 (mitteautoriseeritud päring).

Treeningtulemuste võrdlemiseks teiste kasutajatega on kasutajal võimalus liituda treeningrühmadega. Treeningrühmaga liitumiseks vajutab kasutaja nuppu “Join Group” üldises rühmade vaates (Joonis 11). Avanevas moodulis saab kasutaja kasutada otsingut endale sobiva treeningrühma leidmiseks, kusjuures kasutajale kuvatakse ainult need rühmad, millega kasutaja veel liitunud pole. Võimalikud rühmad tagastatakse näiteks GET päringuga */api/dashboard/join_group?facebookId=1...&token=E...&groups=all*, kus parameetri *groups* väärtus *all* tähistab kõiki rühmi, millega kasutaja pole veel liitunud. Mooduli vaates nupule “Join” (Joonis 11) vajutades tehakse POST päring */api/dashboard/join_group?facebookId=1...&token=E...* ning otsitakse vastava identifikaatoriga andmebaasist rühm ja lisatakse selle viide kasutaja rühmade alla. Tulemusena kuvatakse kasutaja armatuurlaual rühm, millega kasutaja liitus.

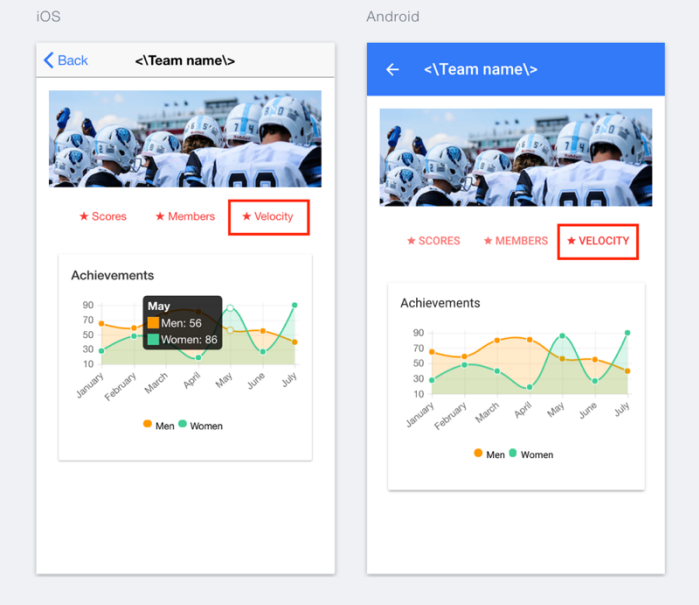


Joonis 9: Rakenduses rühmaga liitumine iOS platvormil

3.2.9 Rühma saavutuste statistika

Rühma vaates, rubriigi "Velocity" all, kuvatakse rühmaga liitunud kasutajate saavutuste kogus ja esitatakse need parema ülevaate saamiseks joongraafiku kujul. Saavutused summeeritakse igakuiselt ja kasutaja näeb korraga viimase poole aasta statistikat. Joongraafikul vastavat punkti puudutades kuvatakse väike moodul täpsema infoga meeste ja naiste saavutuste hulga võrdlusest (Joonis 10).

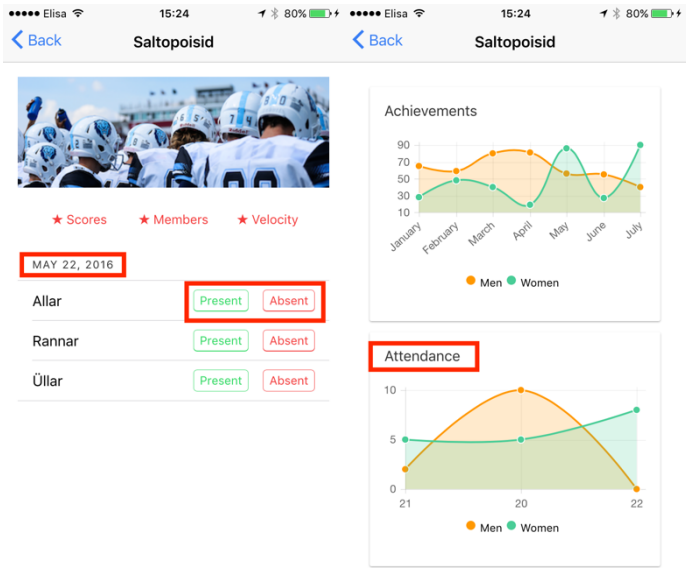
Joongraafiku visualiseerimiseks on taaskasutatud vaba lähtekoodiga Angular 2 abil graafikute visualiseerimiseks mõeldud komponenti *ng2-charts* [22], mis põhineb JavaScripti teegil *Chart.js* [23].



Joonis 10: Joongraafik rühma saavutustest kuvatakse rühma vaates rubriigi “Velocity” all

3.2.10 Kohalolijate märkimine

Rühma treener saab märkida kohalolijaid käimasoleva päeva alusel rühma vaates rubriigi “Members” all. Juhul, kui rühmaliige võtab trennist osa, võib treener märkida rühmaliikme kohalolijaks vajutades nupule “Present”. Juhul, kui rühmaliige trennist puudub, võib treener märkida rühmaliikme puudujaks vajutades nupule “Absent”. Statistika kohalolijate kohta kuvatakse kasutajatele rühma vaates rubriigi “Velocity” all (Joonis 13).



Joonis 11: Kohalolijaid märgitakse rühma vaates käimasoleva päeva alusel

4 Lahenduse analüüs

4.1 Ionic 2 võrdlus konkureerivate vahenditega

Järgnevalt on toodud võrdlus mõningate autori arvates suurimate hübriid mobiilirakenduste arendamiseks mõeldud raamistike vahel. Autor hindab ja arutleb mõningate raamistike omaduste üle, mis võiks mõjutada valikut erinevate raamistike vahel ning olla abiks infosüsteemide arendusalaste eesmärkide täitmisel. Võrdlusesse on valitud Ionic 2 [6], React Native [24] ja Xamarin [25]. Nii React Native, Xamarin ja Ionic 2 toetavad iOS, Android kui ka Windows mobiilseid platvorme.

4.1.1 Ionic 2

Ionic 2 [6] on vaba lähtekoodiga eeskomponendi arenduseks vajalik raamistik eesmärgiga võimaldada arendada hübriidseid mobiilirakendusi. Raamistikus kasutatavad põhilised arendusvahendid on HTML5 ja Angular 2 ning toetatud operatsioonisüsteemid on iOS, Android ja Windows. Ionic baseerub *Apache Cordoval*, mis on funktsionaalsete mobiilirakenduste arendamiseks mõeldud vaba lähtekoodiga arendusplatvorm. Lisaks kasutab Ionic 2 JavaScriptis serverirakenduste kirjutamiseks mõeldud süsteemi Node.js ja versioonihaldustarkvara Git. Kuigi iOS rakenduste arendamiseks on vajalikud Mac OS X ja Xcode ning Android rakenduste arendamiseks on vajalik Android SDK, võimaldab Ionic siiski vaadata ja parandada rakenduse arenduse käigus tekkivaid vigu ja disaini ka tavalises veebilehitsejas.

Kuna Ionic interpreteerib HTML-i ja JavaScriptiga kirjutatud rakenduse sisu, siis tulevad koodi kirjutamisel tekkinud vead nähtavale alles rakenduse jooksutamise ajal. Xamarin käitub selles osas teisiti, kuvades vigu kohe pärast koodi kompileerimist, lühendades sellega võimalike vigade avastamiseks ja parandamiseks kuluvat aega. Teisest küljest, on koodi interpreteerimine arendustsükli lühendamise mõttes Ionicu puhul eelis ja tänu Ionicu võimalusele jooksvalt jälgida rakenduses toimuvaid muutusi kiirendab see oluliselt arendust. React kompileerib ja interpreteerib koodi jooksvalt,

lühendades sellega vigade avastamiseks kuluvat aega ja kiirendades arendustsükleid samaaegselt.

Angular 2, raamistik Ionic 2 raamistiku sees, sisaldab oma süntaksis mõningaid JavaScriptile tundmatuid elemente, näiteks *(click)=”onSelect(hero)”* [26]. See tähendab, et tegemist pole puhta JavaScriptiga ja seega pikeneb raamistiku tundmaõppimiseks kuluv aeg.

4.1.2 React Native

React Native [24] on oma loojate poolt kirjeldatud kui JavaScripti teeki, mis lubab arendada ühise koodibaasiga mobiilirakendusi erinevatele platvormidele ja seejuures programmeerida funktsionaalseid platvormi komponente võimalikult otse läbi JavaScripti. See tähendab, et kui Ionic paigutab enamus rakendusest *WebView* sisse, mis on funktsionaalse rakenduse mõttes veebilehitseja otse rakenduse sees, siis Reacti komponendid suhtlevad otse funktsionaalsete komponentidega. Näiteks pöördub Reactis kuvamiseks mõeldud komponent *<View>* iOS-i puhul *UIView* ja Androidi puhul *android.view* poole. See annab Reactile jõudluse mõttes eelise Ionic 2 ees.

Erinevalt Ionicust ja Xamarinist, mis kujutavad endast raamistikke, on React Native-i puhul tegemist üksnes teegiga ja rakenduse arenduseks võib vaja minna veel teisigi teeke. Nende otsimine ja lisamine võib osutuda ajamahukaks. Raamistike eelis on see, et raamistikud sisaldavad kohe palju olemasolevat funktsionaalsust. Arendusmeeskonnad eelistavad tihtipeale ühtset tuntud raamistikku selle asemel, et iga arendaja eraldi komponente juurde otsiks.

Arendajad, kes varem on JavaScripti kirjutanud, ei pea palju oma teadmisi täiendama, kuna React on oma olemuselt JavaScript, kuid lisanduvad mõned Reactile omased keelelised täiendused.

4.1.3 Xamarin

Xamarin [25] on oma lähtekoodi avaldanud mitmeplatvormiline .NET-il ja C# põhinev raamistik. Võrreldes eelpool nimetatud raamistikega on Xamarin vanem ja seega mõnes mõttes küpsem. See võib pakkuda huvi juba vilunud .NET arendajatele, kes soovivad vältida ümber õppimist ja jääks pigem .NET raamistiku piiridesse. Xamarin ja React konkureerivad pigem programmeerimiskeele eelistamise küsimustes.

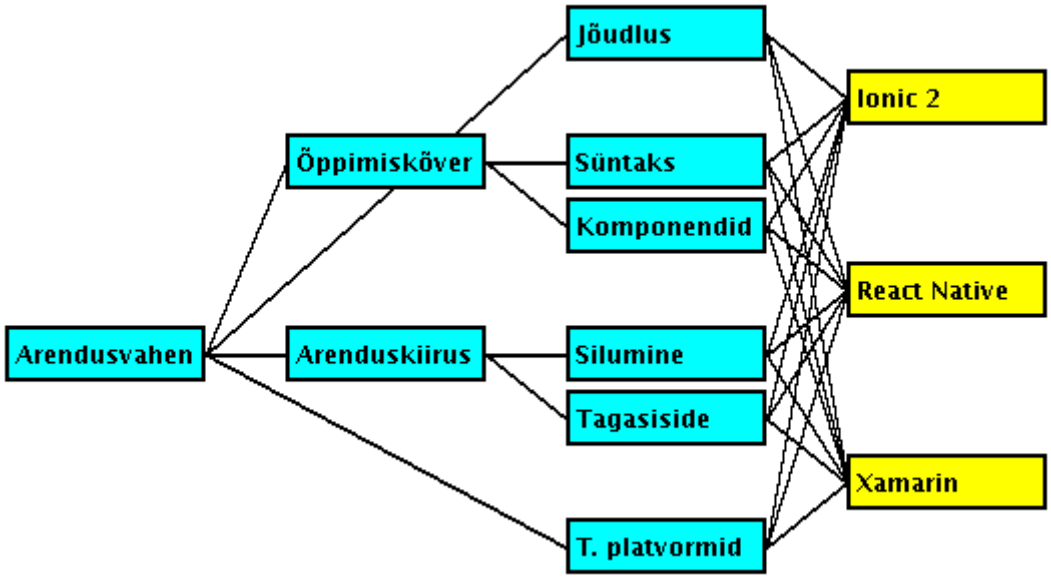
Sarnaselt Reactile, võimaldab Xamarin arendada funktsionaalseid platvormi komponente otse läbi C#-i. Sellist omadust on oluline arvestada jõudluse hindamisel.

4.1.4 Võrdlemine Saaty meetodiga

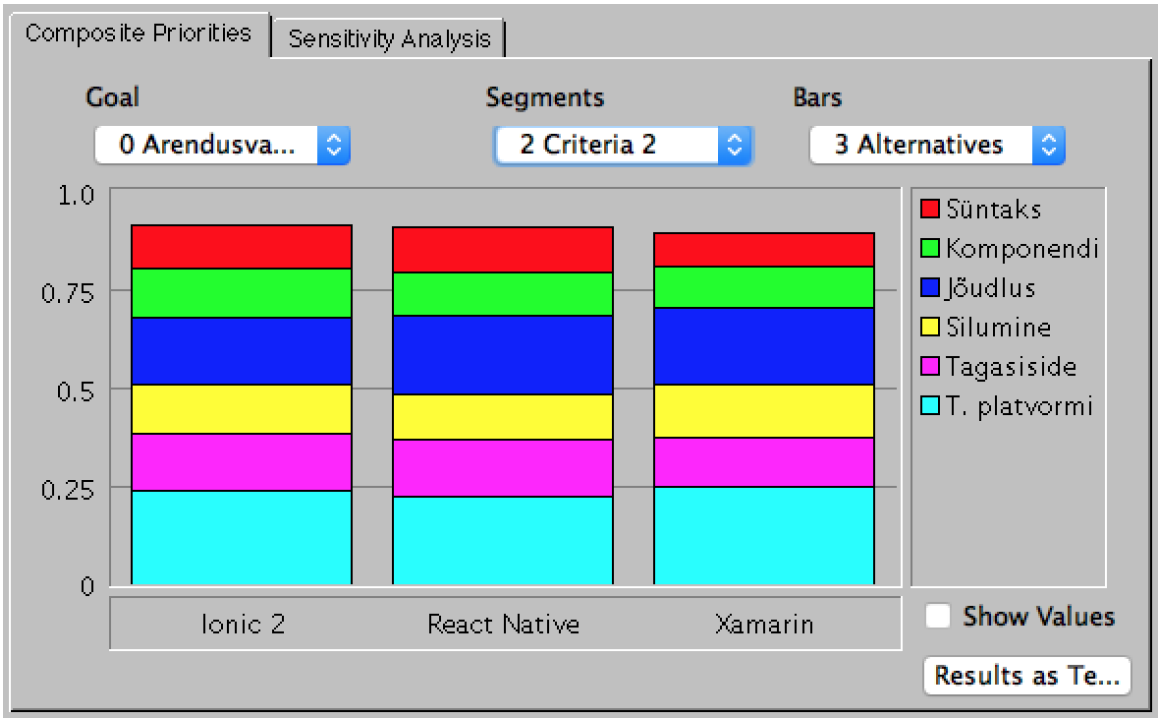
Erinevate arendusvahendite hindamisel on kasutatud Saaty meetodit [27]. Mitme hindamiskriteeriumiga võrdlemiseks ja arendusvahendite prioritiseerimiseks on kasutatud veebipõhist HIPRE 3+ tarkvara [28].

Kriteeriumid, mille põhjal võrdlus sooritati:

- Jõudlus
- Süntaks ja selle õppimine
- Arhitektuuri komponendid ja nende õppimine
- Silumise mõju arenduskiirusele
- Programmeerimisel saadava tagasiside mõju arenduskiirusele
- Toetatud platvormid



Joonis 12: Hindamiskriteeriumid, mille põhjal võrdlus sooritati



Joonis 13: Võrdluste tulemus

4.1.5 Võrdluste kokkuvõte

Arendusvahendite võrdlemisel selgus, et kuna lõpptulemustes suuri erinevusi ei ole, siis valik sõltub eelkõige arendaja või meeskonna eelistustest.

Võib öelda, et kui tegemist on funktsionaalsete mobiilirakenduste arendajatega, siis nende jaoks ei pruugi erinevad hübriid rakenduste arendamiseks mõeldud raamistikud suurt midagi tähendada. Sellegi poolest, on eelpool kirjeldatud raamistikud heaks vahendiks arendajatele, kes soovivad kasutada juba omandatud veebirakenduste loomise oskusi ja ühise koodibaasiga kiirelt katta suurimad mobiilsed platvormid.

Kuna React Native ja Xamarin kuuluvad sarnasesse leeri, siis toob autor välja fundamentaalse erinevuse Reacti ja Xamarini ning Ionic 2 vahel. Ionic 2 on pigem HTML-i keskne ja eristab HTML-i ning JavaScripti komponente, samal ajal kui React Native seob neid ühtseks.

4.2 Hinnang kasutatud tehnoloogiale

Käesoleva bakalaureuse lõputöö raames oli autoril võimalus tutvuda uue ning pidevalt areneva veebi –ja mobiilirakenduste loomiseks mõeldud tehnoloogiaga – Angular 2.

Angularil põhinevat hübriidsete mobiilirakenduste loomiseks mõeldud vahendi Ionic 2 kasutamisega toob autor välja mõningad tähelepanekud:

- õppimiskõver – võrreldes vanema AngularJS [29] versiooniga, puuduvad Angular 2-s AngularJS-i spetsiifilised komponendid (näiteks *controller* ja *$scope*), seejuures sarnaneb Angular 2-e süntaks rohkem puhtale JavaScriptile. Lisaks on Angular 2-e dokumentatsioon enamjaolt kirjutatud JavaScripti laiendavale keelele TypeScript [30], seega rõhub Angular 2 arendusmeeskond tugevalt TypeScripti kasutamisele.

- vähe levinud – arendajal on võimalus kasutada valmisolevaid Ionic 2-e kasutajaliidese komponente ning neid oma vajaduste järgi modifitseerida, kuid komponentidel puudub veel täielik dokumentatsioon. Autor peab vajalikuks dokumentatsiooni täiendamist ja uute komponentide lisamist (näiteks akordionina avanev nimekiri). Populaarses arendajate küsimuste ja vastuste portaalis Stack Overflow [31] on veel vähe Ionic 2-e ja Angular 2-e arendamisel tekkivate probleemide lahendusi, mis sunnib arendajat rohkem dokumentatsioonist lahendust otsima ja aega kulutama.

- programmeerimiskeskkonna toetus – autori valitud integreeritud programmeerimiskeskkonnas IntelliJ pole JavaScripti refaktoreerimise toetus samal tasemel kui näiteks Java koodi puhul, s.t suur osa refaktoreerimisest tuli teha käsitsi muutujate ja funktsioonide nimetusi ümber kirjutades.

- pidevalt uuenev ja ebastabiilne – Ionic 2 rakenduse arendamiseks vajaminevaid teeke haldab *Node Package Manager* [32]. Autori eesmärk oli arendada rakendus võimalikult ajakohaste teekidega, seega tuli leida tasakaal kasutatavate versioonide vahel, kuna pidevalt uuenevad teegid ei olnud alati üksteisega ühilduvad.

- erinevused Android ja iOS rakenduste vahel – Ionic 2 meeskonna poolt valmistatud kasutajaliidese komponentide puhul tuleb arvestada, et neid modifitseerides peab testima ja muutma nii Android kui ka iOS versiooni eraldi, vastavalt iOS kasutajakogemuse juhtnööridele [33] ja Androidi puhul *Material Designi* juhtnööridele [34], et tagada soovitud kasutajakogemus. Käesoleva töö autor ei täheldanud muid erinevusi valminud rakenduste ja lisatud

pistikprogrammide funktsionaalsuses, seega täidab Ionic 2 oma eesmärki – võimaldada kirjutada ühise koodibaasiga mitmele erinevale mobiilsele platvormile rakendusi.

- veebirakenduste komponentide taaskasutamine – keerulisi veebirakenduste loomiseks kasutatavaid komponente on võimalik edukalt ära kasutada ka mobiilirakendustes. Käesoleva töö autor kasutas dünaamilise joongraafiku loomiseks vaba lähtekoodiga komponenti *ng2-charts* [22].

4.3 Lahenduse rakendamine

Loodud rakenduse arhitektuur kujutab endast kolme põhilist eraldiseisvat komponenti – kliendipoolne Ionic 2 raamistikul põhinev rakendus, REST teenuseid jagav Spring Boot raamistikul põhinev tagakomponent ja PostgreSQL andmebaas. Selleks, et valminud rakendus kasutajateni jõuaks, tuleks soetada Apple [3] ja Google [4] arendaja litsents, kompileerida vastavatele platvormidele rakendused ning need üles laadida Google Play-sse ning App Store-i. Rakendus on suunatud TTÜ Tantsutüdrukute ja Saltopoiste tarbeks, kuid kasutatav ka muude treeningrühmade poolt. Spring Booti portatiivsust arvestades, võib tagakomponendi kompileerida *.jar* tüüpi failiks ning tarnida sobivale serverile, mis suudab Java rakendusi tööle panna. Kompileeritud *.jar* tüüpi rakendus sisaldab endas veebiserverit ning kõiki rakendusele vajalikke sõltuvusi. Rakenduse käivitamine ei vaja eraldi ehitamise tööriistu (ingl *build tools*), seadistamist ega konfigureerimist, seega piisab rakenduse käivitamiseks käsust: *java –jar rakenduse_nimi.jar*. Serverile peab lisaks olema installeeritud PostgreSQL andmebaasisüsteem ja loodud Springi rakenduses defineeritud nimetusega andmebaas. Andmebaasi tabelid ja väljad luuakse automaatselt tänu Java Persistence API-le (JPA). Andmebaasi andmete varundamiseks ja vajadusel hiljem taastamiseks on soovitatav serveril käivitada automaatsed andmebaasi varundamise skriptid [35].

4.4 Infosüsteemi analüüs

Hübriidsel platvormil arendatud prototüüp võimaldab kompileerida nii Android kui ka iOS versiooni rakendusest ühise koodibaasi abil. Sellega kaetakse suurem osa TTÜ Tantsutüdrukutel ja Saltopoistel omatavaid seadmeid. Arendatud rakenduse kasutuselevõtmise lihtsustamiseks on kasutaja autentimine võimaldatud Facebooki

kasutajaga, mis elimineerib vajaduse registreerida eraldi kasutajakonto süsteemi jaoks – kasutaja ei pea meeles pidama üleliigseid kasutajanimesid ega paroole.

Ressursside pärimine toimub autori kirjutatud REST teenuste kaudu. REST teenustele antakse kaasa Facebookiga autentimisel saadud kehtiv turvamärgis. Tagakomponent küsib infot turvamärgise kehtivuse kohta Facebookilt. Turvalisuse aspekti osas määratakse sellisel juhul enamuse vastutusest Facebookile, kui välja arvata krüpteeritud HTTPS suhtlus eeskomponendi ja tagakomponendi vahel ning tagakomponendi ja Facebooki vahel. Turvamärgise kontroll eeldab lisapäringut Facebooki Graph API vastu. Ühe kasutaja kohta kehtib 200 päringut tunnis, kuid limiit kehtib rakenduse põhiselt, s.t kui rakendusel on 100 kasutajat, siis rakenduse alt võib teha kuni 20 000 päringut tunnis, kusjuures kõik 20 000 päringut võib teha üks kasutaja, pärast mida ei saa üksi teine kasutaja enam päringuid teha [36]. Kirjeldatud piirang ei takista rakenduse edukat toimimist, kuid rakenduse edasiarenduse skoop võiks sisaldada Facebookiga autentimist läbi tagakomponendi ning tagakomponent võiks seejuures turvamärgiseid ise väljastada. Alternatiivse lahendusena võib juba edukalt valideeritud Facebooki poolt tagastatud turvamärgise salvestada tagakomponendi vahemällu (ingl *cache*), mis väldiks üleliigseid päringuid vastu Facebook Graph API-d.

Vajalik oleks nii tagakomponendile kui ka Ionic 2 rakendusele mõningased täiendused, et parandada mõlema komponendi funktsionaalsust ja kasutusmugavust. Edasiarenduste esimeses plaanis on lisada treeningrühmadest lahkumise võimalus, video kujul saavutuste salvestamise võimalus. Lisaks on serveri, kus tagakomponent töötab, piltide ja videotega ummistamise vältimiseks plaan salvestada meediafailid eraldi failide hoiustamiseks mõeldud serverile.

5 Kokkuvõte

Käesolevas töös loodi Tallinna Tehnikaülikooli Tantsutüdrukutele ja Saltopoistele eesmärkide ja tulemuste haldamise süsteem. Töös on antud ülevaade hübriid mobiilirakenduse arendamiseks vajalikust tehnoloogiast ja nende võrdlusest. Lisaks kirjeldatakse valminud hübriid mobiilirakendusele valitud ning hiljem edasiarendust soodustavat arhitektuuri. Rakenduse kasutajad saavad rakenduse siseselt liituda treeningrühmadega ja võrrelda enda isiklikku tulemuste punktisummat teiste rühmaliikmetega. Lisaks näevad rühmaliikmed rühma üldist arengut joongraafiku kujul, mis esitab viimase poole aasta jooksul täidetud eesmärkide ja treeningutest osavõtmise hulka.

Loodud rakenduses turvamärgise põhjal toimivat Facebookiga autentimise süsteemi näidet on võimalik taaskasutada muude Ionic 2 ja Angular 2 rakenduste loomisel. Autori arendatud REST arhitektuurist lähtuvaid veebiteenuseid võetakse aluseks rakenduse edasiarendusel ühtse arhitektuuri säilitamiseks.

Töö käigus eeskomponendi arendamisel kasutatud Ionic 2 raamistik täidab oma eesmärki, milleks on kiired ja paindlikud arendustsüklid hübriidsete mobiilirakenduste loomiseks. Baaskoodis toimunud muudatusi on võimalik jooksvalt jälgida Ionic 2-e arendusvahendite poolt käivitatud lokaalse arendusserveri abil. Ionicu taaskasutatavad ja paindlikud kasutajaliidese komponendid vähendavad arendaja ajakulu vaadete loomisel mitmele mobiilsele platvormile korraga. Kiire kasutajaliidese loomine koos Soren Laueseni väljatöötatud printsiibiga *The Virtual Method*, mis soovitab lähtuda süsteemi arendamisel vaadete ja nendes kuvatavast infost, aitavad pöörata tähelepanu kasutusmugavusele ja minimaalsele vajalikule funktsionaalsusele võimalikult varajases arenduse algstaadiumis.

Kasutatud kirjandus

[1] „JavaScript,“ [Võrgumaterjal]. Available: https://www.javascript.com/. [Kasutatud 13 05 2016].

[2] Apple Inc., „Xcode - What's New - Apple Developer,“ [Võrgumaterjal]. Available: https://developer.apple.com/xcode/. [Kasutatud 13 05 2016].

[3] Apple Inc., „Apple Developer Program - Apple Developer,“ [Võrgumaterjal]. Available: https://developer.apple.com/programs/ . [Kasutatud 13 05 2016].

[4] „Google Play Developer Console,“ [Võrgumaterjal]. Available: https://play.google.com/apps/publish/signup/ . [Kasutatud 13 05 2016].

[5] S. Lauesen, „User Interface, Soren Lauesen,“ [Võrgumaterjal]. Available: http://www.itu.dk/~slauesen/SorenUID.html. [Kasutatud 13 05 2016].

[6] Drifty, „Ionic Documentation - Ionic Framework,“ [Võrgumaterjal]. Available: http://ionicframework.com/docs/v2/. [Kasutatud 13 05 2016].

[7] „Java Persistence API - Wikipedia, the free encyclopedia,“ [Võrgumaterjal]. Available: https://en.wikipedia.org/wiki/Java_Persistence_API. [Kasutatud 13 05 2016].

[8] Angular, „Architecture Overview - ts,“ [Võrgumaterjal]. Available: https://angular.io/docs/ts/latest/guide/architecture.html. [Kasutatud 13 05 2016].

[9] „Sass Documentation,“ [Võrgumaterjal]. Available: http://sass-lang.com/documentation/file.SASS_REFERENCE.html. [Kasutatud 20 05 2016].

[10] FasterXML, LLC, „JacksonHome - FasterXML Wiki,“ [Võrgumaterjal]. Available: http://wiki.fasterxml.com/JacksonHome. [Kasutatud 13 05 2016].

[11] K. Beck, „Test-Driven Development By Example,“ [Võrgumaterjal]. Available: http://www.eecs.yorku.ca/course_archive/2003-04/W/3311/sectionM/case_studies/money/KentBeck_TDD_byexample.pdf. [Kasutatud 13 05 2016].

[12] JetBrains, „IntelliJ IDEA the Java IDE,“ [Võrgumaterjal]. Available: https://www.jetbrains.com/idea/. [Kasutatud 13 05 2016].

[13] „AKIT - Andmekaitse ja infoturbe leksikon,“ [Võrgumaterjal]. Available: http://akit.cyber.ee/term/315-recovery-time-objective. [Kasutatud 13 05 2016].

[14] „AKIT - Andmekaitse ja infoturbe leksikon,“ [Võrgumaterjal]. Available: http://akit.cyber.ee/term/316-recovery-point-objective. [Kasutatud 13 05 2016].

[15] D. Jabif, „Ionic Tutorial - How to add Facebook Native login to your Ionic app,“ [Võrgumaterjal]. Available: https://ionicthemes.com/tutorials/about/native-facebook-login-with-ionic-framework. [Kasutatud 13 05 2016].

[16] Wizcorp, „Apache Cordova Facebook Plugin,“ [Võrgumaterjal]. Available: https://github.com/Wizcorp/phonegap-facebook-plugin. [Kasutatud 13 05 2016].

[17] „FB.getLoginStatus() - Web SDKs - Dokumentatsioon - Facebook for Developers,“ [Võrgumaterjal]. Available: https://developers.facebook.com/docs/reference/javascript/FB.getLoginStatus.

[Kasutatud 13 05 2016].

[18] „Logout - Web SDKs - Dokumentatsioon - Facebook for Developers,“ 13 05 2016. [Võrgumaterjal]. Available: https://developers.facebook.com/docs/reference/javascript/FB.logout.

[19] K. D. R. C. Craig Walls, „Spring Social Facebook Reference,“ [Võrgumaterjal]. Available: http://docs.spring.io/spring-social-facebook/docs/2.0.3.RELEASE/reference/htmlsingle/. [Kasutatud 15 05 2016].

[20] „Buttons: Floating Action Button - Components - Google design guidelines,“ [Võrgumaterjal]. Available: https://www.google.com/design/spec/components/buttons-floating-action-button.html#. [Kasutatud 13 05 2016].

[21] The Apache Software Foundation, „apache/cordova-plugin-camera: Mirror of Apache Cordova Plugin camera,“ [Võrgumaterjal]. Available: https://github.com/apache/cordova-plugin-camera. [Kasutatud 17 05 2016].

[22] Valor Software, „valor-software/ng2-charts: Beautiful charts for Angular2 based on Chart.js,“ [Võrgumaterjal]. Available: https://github.com/valor-software/ng2-charts. [Kasutatud 13 05 2016].

[23] „Chart.js | Open source HTML5 Charts for your website,“ [Võrgumaterjal]. Available: http://www.chartjs.org/. [Kasutatud 13 05 2016].

[24] Facebook, „React Native | A framework for building native apps using React,“ [Võrgumaterjal]. Available: https://facebook.github.io/react-native/. [Kasutatud 13 05 2016].

[25] Xamarin Inc., „Mobile App Development & App Creation Software - Xamarin,“ [Võrgumaterjal]. Available: https://www.xamarin.com/. [Kasutatud 13 05 2016].

[26] Angular, „User Input - ts,“ [Võrgumaterjal]. Available: https://angular.io/docs/ts/latest/guide/user-input.html. [Kasutatud 13 05 2016].

[27] T. L. Saaty, „Analytic hierarchy process - Wikipedia, the free encyclopedia,“ [Võrgumaterjal]. Available: https://en.wikipedia.org/wiki/Analytic_hierarchy_process . [Kasutatud 13 05 2016].

[28] J. M. Raimo P. Hämäläinen, „Web-HIPRE,“ [Võrgumaterjal]. Available: http://hipre.aalto.fi/. [Kasutatud 13 05 2016].

[29] Angular, „AngularJS — Superheroic JavaScript MVW Framework,“ [Võrgumaterjal]. Available: https://angularjs.org/. [Kasutatud 13 05 2016].

[30] „TypeScript - JavaScript that scales.,“ [Võrgumaterjal]. Available: https://www.typescriptlang.org/. [Kasutatud 13 05 2016].

[31] „Stack Overflow,“ [Võrgumaterjal]. Available: http://stackoverflow.com/ . [Kasutatud 13 05 2016].

[32] „npm,“ [Võrgumaterjal]. Available: https://www.npmjs.com/. [Kasutatud 13 05 2016].

[33] „iOS Human Interface Guidelines: Designing for iOS,“ [Võrgumaterjal]. Available: https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/. [Kasutatud 13 05 2016].

[34] „Introduction - Material design - Google design guidelines,“ [Võrgumaterjal].

Available: https://www.google.com/design/spec/material-design/introduction.html. [Kasutatud 13 05 2016].

[35] „Automated Backup on Linux - PostgreSQL wiki,“ [Võrgumaterjal]. Available: https://wiki.postgresql.org/wiki/Automated_Backup_on_Linux. [Kasutatud 13 05 2016].

[36] „Rate Limiting - Graph API - Dokumentatsioon - Facebook for Developers,“ [Võrgumaterjal]. Available: https://developers.facebook.com/docs/graph-api/advanced/rate-limiting. [Kasutatud 13 05 2016].

[37] H. Vallaste, „e-Teatmik: IT ja sidetehnika seletav sõnaseletusraamat,“ 13 05 2016. [Võrgumaterjal]. Available: http://www.vallaste.ee/index.asp.

Lisa 1.

- Viide valminud rakenduse eeskomponendi koodi repositooriumile - https://github.com/aglo35/Pace
- Viide valminud rakenduse tagakomponendi koodi repositooriumile - https://github.com/aglo35/BackPaceV2

Lisa 2. Näide tagakomponendi lühendatud rühmade vaate tagastatavast vastusest

```
{
  teamName: "Kossurühm",
  shortTableRowList:
  {
    rank: 1,
    userName: "Marin",
    tier: "...",
    points: 1270
  },
  {
    rank: 2,
    userName: "Marianne",
    tier: "...",
    points: 1250
  },
  {
    rank: 3,
    userName: "Maarika",
    tier: "...",
    points: 1100
  }
},
{
  teamName: "Saltopoisid",
  shortTableRowList:
  {
    rank: 1,
    userName: "Allar",
    tier: "...",
    points: 1000
  },
  {
    rank: 2,
    userName: "Rannar",
    tier: "...",
    points: 980
  }
}
```

Lisa 3. Näide tagakomponendi võimalike saavutuste tagastatavast vastusest


```
{
  "cheerleading":[
    {
      "id":1,
      "imagePath":...,
      "title":"Toss cupie",
      "category":"cheerleading",
      "points":80,
      "achieved":false
    },
    {
      "id":2,
      "imagePath":...,
      "title":"Liberty",
      "category":"cheerleading",
      "points":50,
      "achieved":false
    }
  ],
  "gymnastics":[
    {
      "id":3,
      "imagePath":...,
      "title":"Salto backward",
      "category":"gymnastics",
      "points":50,
      "achieved":false
    },
    {
      "id":4,
      "imagePath":...,
      "title":"Flic flac",
      "category":"gymnastics",
      "points":50,
      "achieved":false
    }
  ]
}
```