TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Argo Linnaste 179041

HÜBRIID-MOBIILIRAKENDUSE LOOMINE ANDMETE KÄTTESAADAVUSE LIHTSUSTAMISEKS EESTI JALGPALLI LIIDU NÄITEL

Bakalaureusetöö

Juhendaja: Meelis Antoi
MSc

Tallinn 2020

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Argo Linnaste

16.05.2020

Annotatsioon

Antud bakalaureusetöö eesmärk on luua hübriid-mobiilirakendus jalgpallitulemuste ja muude spordiandmete jälgimiseks, mida saab kasutada nii Android kui IOS platvormidel. Rakendusega on kasutajal võimalik reaalajas jälgida tulemusi, lisada lemmikuks võistkondi ning saada teavitusi kasutaja lemmikute tulemuste kohta. Töö tulemusena valmib rakenduse beetaversioon, mis läbib ka esialgse väikesemahulise testimisfaasi.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 47 leheküljel, 7 peatükki, 31 joonist.

Abstract

Creation of a Hybrid Mobile Application for Simplifying Data Access on the Example of the Estonian Football Association

The aim of this bachelor's thesis is to create a hybrid mobile application for tracking football results and sports data, which can be used on both Android and iOS platforms. With the application, the user can monitor the results in real time, add teams as favourites and receive notifications about the user's favourite’s results. As a result of the thesis, a beta version of the application will be completed, which also goes through an initial small-scale testing phase.

The thesis is in Estonian and contains 47 pages of text, 7 chapters, 31 figures.

Lühendite ja mõistete sõnastik

Vaade	Vahend rakenduses objektide kuvamiseks. Ühe ekraani kuva.
HTTP	*HyperText Transfer Protocol* - hüperteksti edastusprotokoll.
BLoC muster	*Business Logic Component* - *Flutteri* arenduskeskkonnas kasutusel olev oleku juhtimise muster.
JSON	*JavaScript Object Notation* – andmevorming/süntaks andmete salvestamiseks ning vahetamiseks.
Skoop	Programmi osa, milles nimetatud deklaratsioon kehtib.
Vaikebrauser	Arvuti või mobiilse seadme veebibrauser, mis on vaikimisi määratud veebilehti avama.
Kasutajaliides	Ühenduslüli kasutaja ja rakenduse vahel. Kasutajaliides teeb rakenduse funktsionaalsuse kasutajale kättesaadavaks.
Dependency injection	Sõltuvuse süstimine, mille eesmärk on süstida süsteemi sisse funktsionaalsust tagavad komponendid, millest süsteem töö sõltub.
API	*Application Programming Interface* – rakenduse programmeerimisliides
Tarkvara raamistik	Juba varem loodud funktsionaalsuse komplekt, mida on arendajal võimalik ümber kirjutada vastavalt soovile ja äriloogikale.
Android	Avatud lähtekoodiga Linuxil põhinev operatsioonisüsteem mobiilsetele seadmetele.
iOS	*Apple*'i poolt loodud operatsioonisüsteem mobiilsetele seadmetele.
Tõuketeavitus	Mobiilseadmetel kasutuses olev kasutajale informatsiooni saatmiseks mõeldud sõnum.
Background worker	Taustatöötaja – aeganõudvate operatsioonide jaoks loodud taustal töötav süsteem.
SQLite	Andmebaasihaldussüsteemi tarkvarakogu, mida on lihtne rakendada ja administreerida.

Sisukord

Jooniste loetelu

1 Sissejuhatus

Tänapäeval on inimesed muutunud suuresti oma telefonidest sõltuvateks. Kogu avalikule informatsioonile on oluline lihtsamini ligi pääseda. Isegi vähesel määral info leidmise raskendused pärsivad inimeste elu ja muudavad suhtumist kogu teabega seotud süsteemi negatiivseks.

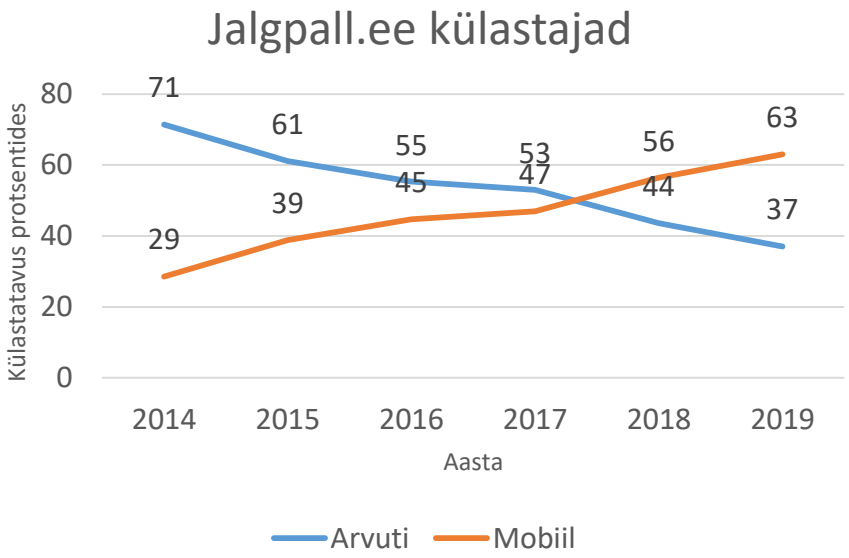
Eesti jalgpalliga, eelkõige laste ja meeste madalamate liigade jalgpalliga seotud võistlustega ning seonduvate andmete kättesaadavus on tülikas ja nendeni jõudmine ebamugav. Infoni jõudmiste raskuste tõttu on inimesed vähem huvitatud jalgpalli tulemustest ja tabelite jälgimisest, mis omakorda vähendab huvi spordisse ja seotust nimetatud spordialaga.

Diplomitöö autori huvi probleemi lahendamise vastu tekkis eelnevalt kirjeldatud probleemide tõttu, kuna autor tegeleb ka ise jalgpalliga. Võistluskaaslaste tulemuste, seisude ja tabelite pidev otsimine on olnud ajakulukas ning tülikas. Käesolevas töös esitatav rakendus aitaks jalgpalliga seotud kogukonda, hoides inimesi sündmustega pidevalt kursis nii rakenduse kasutusmugavuse kui ka mobiilsete teavituste kaudu.

Käesoleva töö eesmärk on lihtsustada Eesti jalgpalliga seotud andmetele kättesaadavust ja muuta nii tegevsportlastele kui ka jalgpallihuvilistele inimestele mängude ning tabelite jälgimine lihtsamaks. Antud eesmärki plaanitakse saavutada hübriid-mobiilirakenduse kaudu.

2 Varasem süsteem ning planeeritav rakendus

Eesti Jalgpalli Liidu ametlik kodulehekülg [1] avati uuendatud kujul 2017 aasta algul. Senini käigus olnud veebilehe lahendus on olnud piisav, et ära toita kõikide külastajate jalgpalliandmete vahendamise vajadused, kuid mida aasta edasi, seda enam on lehe kasutajad liikunud üle mobiilsetele seadmetele (Joonis 1).



Joonis 1 Jalgpall.ee külastatavus

Senise kodulehe kasutajatelt kogutud tagasisidest on selgunud, et käigusoleva veebilehe ülesehitus on liialt keeruline ning andmete kättesaadavus raskendatud. Pidevalt on vajalik läbida ühesuguseid samme ja iga uus päring tähendas, et kogu otsingu teekond algas otsast peale. Puudus võimalus andmeid sorteerida ning personaliseerida, mis muutiski lehel navigeerimise tülikaks. Planeeriti luua hübriid-mobiilirakendus, mis pidi aitama kõiki seni mainitud probleeme lahendada.

Organisatsioon seadis rakendusele ka mõningad lähtetingimused, milleks olid:

- rakendus peab töötama nii Android kui IOS seadmete peal;
- kasutajatel peab olema lemmikute salvestamise ehk personaliseerimise võimalus;
- rakendusel peab olema tõuketeadete saatmise võimekus;
- minimaalne elujõuline toode peab valmima 6 kuuga.

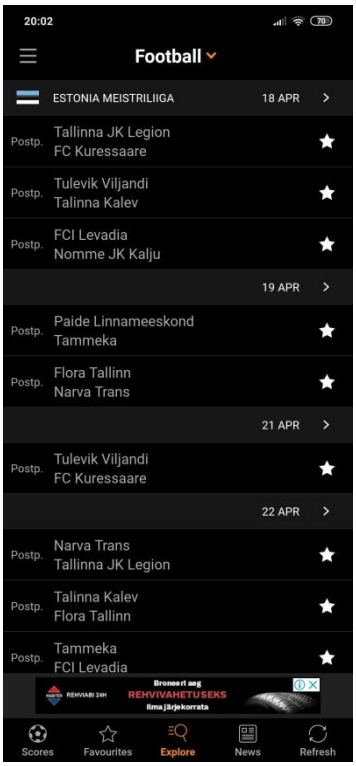
3 Olemasolevad rakendused

Käesolevas peatükis tutvustab ja analüüsib autor hetkel turul olemasolevaid rakendusi, mis on loodud sarnaste probleemide lahendamiseks väljaspool Eestit. Kõik analüüsitud rakendused on mõeldud kasutamiseks eelkõige väljaspool Eestit. Esitatud rakenduste analüüsimisel leidis autor mitmeid funktsionaalseid ja disainilahendusi, mida sobib kasutada ka uue rakenduse loomisel.

Analüüsitavateks rakendusteks on rahvusvaheline sporditulemuste edastamise rakendus *LiveScore*, Rootsis loodud rakendus *Forza Football* ning Saksamaa jalgpallirakendus *Bundesliga*.

3.1 LiveScore

LiveScore [2] on mobiilirakendus, mida käesoleva töö kirjutamise hetkel peetakse sporditulemuste reaalajas edastamise kindlaks turuliidriks. 1998. aastal loodud süsteem toimib suurepäraselt maailma suurimate spordivõistluste edastamiseks, kuid puudused ilmnesid just Eesti jalgpalli võistluste tulemuste ning andmete suhtes.



Joonis 2 *LiveScore*'i rakendus

Rakendust kasutades on võimalus otsida ja jälgida kasutajat huvitavaid spordialasid. Lisaks otsimise võimekusele saab jälgida nii spordialasid, võistkondi kui erinevate spordialade üksikuid võistlusi.

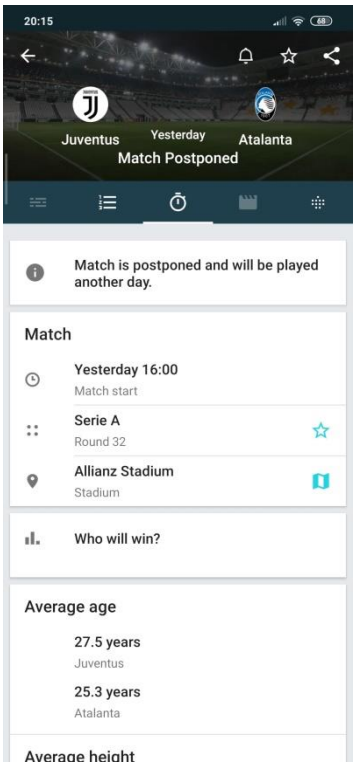
Käesolevat rakendust uurides ja analüüsides leidis autor mitmeid huvitavaid ning mugavust lisavaid elemente, mida võeti hiljem kasutusele ka antud töö raames loodavas rakenduses.

Ühe funktsionaalsuse elemendina leidis autor, et *LiveScore*'i rakenduse teavituste süsteem on lahendatud väga loogiliselt järgitavalt ning rakenduse kasutaja jaoks kergesti mõistetavalt. Teavitusi on kerge sisse ja välja lülitada ning neid personaliseerida oma vajaduste ja soovide põhjal. Kasutaja saab valida, milliste võistluste kohta ta teavitusi soovib ning eelistuse korral on võimalus ka lemmikuks valitud võistkondade kindlate võistluste teavitusi keelata.

Teine huvipakkuv element oli autori arvates mängude nimekirjadena näitamise lahendus. Nimekirjad olid väga kompaktselt ja silmale meeldivalt esitatud.

3.2 Forza Football

Järgmisena analüüsis autor Rootsis loodud rakendust *Forza Football* [3]. Rakenduse lähemal uurimisel leidis autor mitmeid visuaalseid disainielemente, mille vastu tal huvi tekkis.

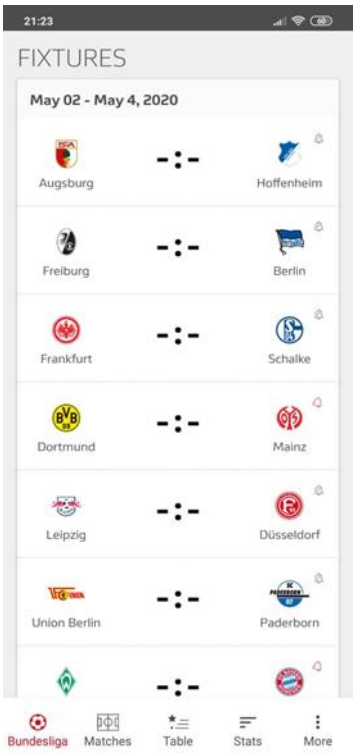


Joonis 3 Forza Football rakendus

Üheks huvipakkuvatest elementidest oli meetod, kuidas luua käimasoleva mänguvaate kasutajaliidest. Autorile meeldis lahendus näidata ekraani ülaosas käimasolevas mängus osalevaid meeskondi ning selle alla luua vahekaartide abil mänguelementide navigeerimise võimalus. Kasutajamugavuse mõistes pakkus antud lahendus piisavalt lihtsat ja üheselt mõistetavat andmete edastamise loogikat.

3.3 Bundesliga

Viimaseks analüüsitud rakenduseks oli Saksamaa jalgpallitulemuste nägemiseks loodud rakendus *Bundesliga* [4]. Rakenduse lähemal uurimisel leidis autor mitmeid visuaalselt meeldivaid kasutajaliidese elemente.



Joonis 4 Bundesliga rakendus

Üheks nimetatud elementidest oli moodus, kuidas analüüsitud rakenduses oli lahendatud nimekirjas olevate mängude näitamine. Kogu nimekirja ruumi ja värvide kasutus oli esitatud väga puhtalt ning visuaalselt rahuldust pakkuvalt.

3.4 Olemasolevate rakenduste analüüsi tulemus

Analüüsist selgus, et nii *Forza Football*'il kui ka *LiveScore*'il on olemas vajalik funktsionaalsus, et kajastada suuremaid Eesti jalgpallivõistluseid, milleks on meeste meistriliiga ehk Premium liiga, esiliiga, superkarikas ning Tipneri karikavõistlus. Rakenduses *Forza Football* on olemas ka Eesti koondiste jälgimise võimekus. Kolmas valitud rakendus, *Bundesliga* kajastab vaid Saksamaa jalgpalli, kuid rakendus pakkus autori oma kasutajaliidese tõttu sellegipoolest huvi.

Lisaks sellele sai kinnitust, et meeste ja naiste madalamaid liigasid ning noorte jalgpallivõistluseid ei ole, antud bakalaureusetöö loomise hetkel, reaalajas võimalik mugavalt ning mobiilselt jälgida.

Rakenduste analüüs andis autorile mitmeid uusi mõtteid, kuidas uues arendatavas rakenduses lahendada nii funktsionaalsuse kui ka kasutajaliidesega seotud ülesandeid.

4 Kasutatavad tehnoloogiad

Käesolevas peatükis annab autor ülevaate kõikidest antud projektis kasutatud tehnoloogiatest ning nende valimise põhjustest.

4.1 Flutter

Mobiilse rakenduse arendust alustas autor esmalt sobiva arendusplatvormi otsimisega. Põhjusel, et rakendus pidi lähtetingimuste järgi töötama nii Android kui ka IOS operatsioonisüsteemide peal, siis otsustas autor, et kõige sobivam on antud rakendus on luua hübriid-mobiilirakendusena.

Antud otsus tähendas aga seda, et oli vajalik teha uurimustööd, mis aitaks otsustada, milline võimalikest hübriidrakenduste raamistikest oleks antud projekti jaoks kõige sobivam. Töö autoril oli kooli kaudu juba olemas eelnev kogemus hübriid-mobiilirakendustega ning nende arenduskeskkondade omapäradega, mistõttu oli lihtsam valida endale sobiv raamistik, mis täidaks kõik uue süsteemi vajalikud tingimused [5].

Autor valis võrdluseks välja 3 raamistikku, mida projekti alguse hetkel erinevates internetikeskkondades kõige populaarsemateks peeti [6]. Nendeks raamistikeks olid *Ionic* [7], *React Native* [8] ning *Flutter* [9].

Kõik 3 eeltoodud erinevad üksteisest nii ülesehituse kui jõudluse poolest. *React Native* ning *Ionic* kasutavad programmeerimiskeelena *JavaScripti*, mis on programmeerimiskeel, mida kasutatakse eelkõige veebiarenduses sisu loomiseks ning manipuleerimiseks. Neist kahest erineb aga *Flutter*, mille programmeerimiskeeleks on *Dart*, mis on 2011. aastal *Google'i* poolt loodud C-stiilis süntaksiga, objektorienteeritud, klassipõhine keel (Lisa 1). *Flutteri* suurimaks eeliseks teiste raamistike ees on kasutajaliidese valmimise kiirus. [10]

Esialgse planeeringu kohaselt pidi rakendus test- ehk beetaversiooni jõudma kahe jalgpallihooaja vahelise perioodi lõpuks, mis tähendas, et kogu arendustegevusele seati ajaline piirang. Pärast mõningat analüüsi, võttes arvesse autori varasemaid kogemusi ning ajalisi piiranguid, osutus sobivaks raamistikuks *Flutter*. Suurimaks faktoriks raamistiku valikul oli eelkõige autori suur kogemus *Flutteriga* nii koolis kui väljaspool kooli, mis tähendas, et vähem aega kulus valitud raamistiku õppimisele.

4.2 Firebase

Firebase on *Google'i* arendatud platvormiülene pilveteenus. *Firebase* pakub kasutajatele erinevaid taustateenuseid, mille hulka kuuluvad näiteks pilveteenused, andmebaasi ning serveri teenused ja paljud muud.

Antud teenus ja teenusepakkuja said rakenduse loomiseks valitud eelkõige oma rakendamise lihtsuse ning kiiruse tõttu. Lisaks sellele on *Firebase*'i teenuse kasutamine tasuta, mis lisaks teenuse kiirusele, oli üks põhilistest valiku põhjendustest.

Antud töös kasutab autor *Firebase'i* pilveteavituste süsteemi [11] hübriidrakenduse kasutajatele teavituste saatmiseks, mis toimub *Firebase legacy* HTTP protokolli [12] kaudu. Teavitused saadetakse HTTP päringutega .Net taustasüsteemist koos vajalike andmetega *Firebase'i* süsteemi, kus toimub nende andmete töötlemine ning tõuketeavituste klientidele ehk rakenduse kasutajatele edastamine.

4.3 ASP.NET Core

Kõiki rakendusega seotud andmeid kliendi ja andmebaasi vahel vahendab *ASP.NET Core* raamistik [13]. See on Microsofti loodud tasuta ja avatud lähtekoodiga hallatav tarkvararaamistik, mis laiendab sama ettevõtte loodud .Net raamistiku, Windowsi, Linuxi ja macOS'i operatsioonisüsteemidele.

Projekti alguse hetkeks oli *ASP.NET Core* raamistik juba organisatsioonis käigus. Raamistiku oli senini kasutatud API ehk programmeerimisliidese arendamiseks. Eelnevalt loodud funktsionaalsust autor antud töös ei kirjelda.

Autori eesmärk oli luua uus taustal perioodiliselt töötav lahendus, mis tegeleks reaalajas käimasolevate jalgpallimängude andmete uuendamisega ning teavituse süsteemi haldamisega. Koolis õpitu põhjal olid autoril .Net ja ASP.NET raamistikega töötamiseks juba vajalikud teadmised olemas, mistõttu sujus taustal käigus oleva töötaja (inglise keeles *background worker*) arendusprotsess kiiresti.

Süsteemi loomiseks võttis autor aluseks Microsofti dokumentatsioonis [14] oleva lahenduse, mille alusel luuakse funktsionaalsuse saavutamiseks vajalik skoop taustal töötava teenuse sees ning kasutatakse sõltuvuse süstimise (inglise keeles *dependency injection*) meetodit [15].

Andmebaasi kontekst, mis on vahelüliks andmebaasi ja raamistiku mudelite vahel, süstitakse süsteemi ning rakendatakse äriloogika, mille kaudu saadakse andmebaasist parameetri alusel küsitud andmed. Pärast andmete lugemist salvestatakse need omakorda kindla võtme põhjal vahemällu, kust hiljem väljastatakse päringute vastustena JSON vormingus.

Eelnevalt kirjeldatud taustateenuse meetodil loodi ka mobiilirakenduse automaatteavituste süsteem. Teenus teeb perioodiliselt päringuid andmebaasi, küsib seadistatud parameetrite põhjal välja nii teavitused kui ka teavitustega seotud kasutajad ning saadab andmed HTTP protokolli kaudu *Firebase'i* pilvesõnumite süsteemi.

5 Valminud rakenduse kirjeldus

Käesolevas peatükis kirjeldatakse valminud rakenduse erinevate vaadete funktsionaalsust koos kuvatõmmistega.

Rakendus on kasutamiseks mõeldud eelkõige inimestele, kellel on huvi jalgpalli kui spordiala vastu, kuid ära ei saa unustada ka potentsiaalseid uusi kasutajaid. Rakenduse ülesehituse puhul on mõeldud, et kasutamine oleks väga mugav ning käepärane, seda ka juhul kui kasutajatel puudub eelnev kokkupuude jalgpalliga.

Kogu rakenduse andmete pärimine ning suhtlus andmebaasiga toimub serveris töötava *ASP.NET Core* raamistiku kaudu.

Rakendus kasutab päringuteks ning andmete oleku hoidmiseks BLoC arhitektuurimustrit [16]. Antud muster haldab vajalikku äriloogikat ja kasutab andmevooge (inglise keeles *data stream*), et võtta vastu sündmusi, mis saadetakse välja rakenduse erinevates vaadetes. Peale äriloogika rakendamist edastatakse vastuvõetud andmed uue olekuna tagasi rakenduse vaadetesse. Rakendus kasutab äriloogika teostamiseks HTTP protokolli, mille abil suheldakse serveris oleva taustasüsteemiga päringute kaudu (Lisa 2). Kui rakendus on saanud kätte päringu vastuse, kutsutakse nende andmetega esile oleku muudatus, mis omakorda toob kaasa selle olekuga seotud kasutajaliidese elementide ülesehituse. Vajalike BLoC funktsionaalsust tagavate klasside seadistamine leiab aset „main.dart“ failis, kust algab ka kogu rakenduse käivitamine (Lisa 3).

5.1 Kasutaks registreerimine

Rakenduse esmakordsel kasutamisel avaneb kasutajal võimalus end registreerida, siseneda juba olemasoleva kasutaja või külalisena.

Kui rakendust kasutav persoon ennast registreerida ei soovi, on tal võimalus siseneda külalisena, mis tähendab, et kogu rakenduse funktsionaalsus säilib, kuid hiljem rakendust uuesti paigaldades on vaja kogu registreerimise protsess uuesti läbi käia.

5.1.1 Kasutaja loomine

Kasutaja loomiseks on rakendust kasutaval inimesel vaja tekstikastidesse sisestada oma ees- ja perekonnanimi, e-posti aadress ning parool. Pärast vajalike andmete sisestamist on kohustuslik veel nõustuda rakenduse privaatsus- ja kasutustingimustega.

Kasutaja sisendeid kontrollitakse nii lokaalselt mobiilse seadme poolel kui ka serveris oleva ASP.NET raamistiku kaudu. Kontrollitakse ka kasutaja sisendite pikkuste sobivust antud süsteemi nõuetele ning e-posti aadressi puhul, kas sisestatud aadress ei ole juba kasutuses. Vigaste sisendite korral näidatakse kasutajale vastava välja juures veateadet, mis selgitab antud viga. Kui sisendid vastavad kõikidele seatud nõuetele ning kasutaja on nõustunud privaatsus- ja kasutustingimustega, suunatakse kasutaja järgmisesse vaatesse.

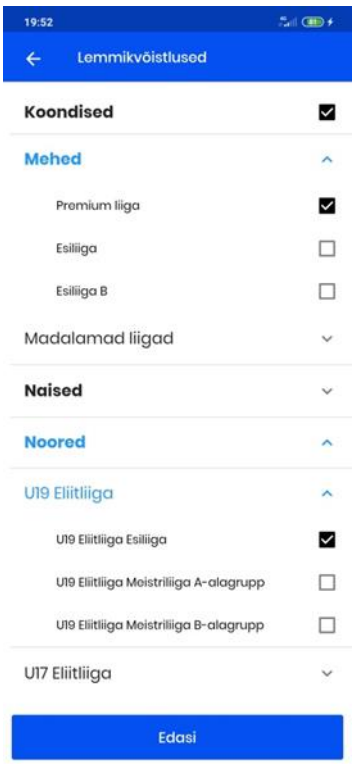
Joonis 5 Kasutaja loomine

5.1.2 Lemmikute valimine

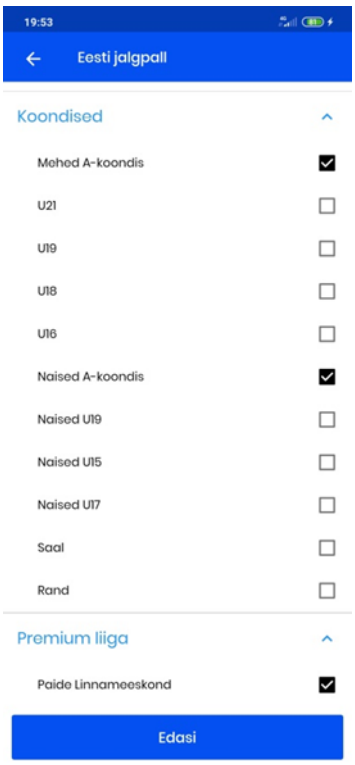
Pärast registreerumist või külalisena sisenemist on kasutajal võimalus valida sel aastal käimasolevate võistluste ja nendel võistlustel osalevate võistkondade hulgast oma lemmikud.

Lemmikvõistluste vaates esitatakse kasutajale ripploendis esmalt nimekiri, mille alamelementideks on sel hooajal toimuvad jalgpallivõistlused. Iga võistluse nimest paremal pool paiknevad märkeruudud, millel klikkides muutuvad valikud aktiivseks. Valitud võistluste põhjal tehakse taustsüsteemi HTTP päring, mille vastuse järgi genereeritakse kasutajale järgmises vaates eelmisele sarnane vaade. Uues vaates näidatakse kasutajale ripploendites uusi nimekirju, mis on seekord täidetud kõikide valitud võistlustel antud hooajal mängivate võistkondade nimedega. Antud võistkondade hulgast saab kasutaja välja valida need, kelle vastu tal huvi on ja kellele ta soovib hiljem menüüst ning lemmikute vaatest kiiret juurdepääsu.

Joonis 6 Lemmikute valimine - I



Joonis 7 Lemmikute valimine - II



Joonis 8 Lemmikute valimine - III

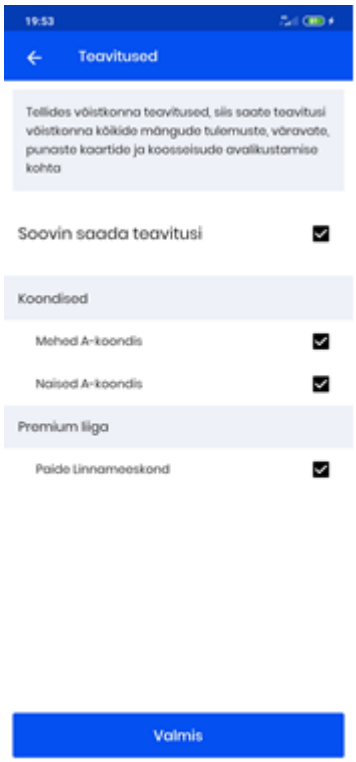
5.1.3 Teavituste valimine

Pärast lemmikute valimise protsessi on kasutajal võimalik seadistada valitu põhjal mobiilsete teavituste saamise seaded. Teavitusi saadetakse kasutajale *Firebase'i* pilvesõnumite süsteemi kaudu.

Kasutajal on võimalik seadistada üldised teavituste saamise sätted. Selle valikuga on kasutajal lihtsasti võimalik teavituste süsteemi sisse ja välja lülitada. Lisaks sellel on võimalus valida ka iga lemmikvõistkonna kohta teavituste saamine.

Teavitusi saadetakse kasutajale lemmikuks valitud võistkondade kõikide mängude tulemuste, väravate, punaste kaartide ning koosseisude avalikustamiste kohta.

Pärast kasutajale sobivate seadete valimist saab kasutaja vajutada ekraani allosas olevale nupule „Valmis“, mis töötleb registreerimise kasutajavoos valitud andmed korrektsesse vormingusse ning saadab need HTTP päringuga taustasüsteemile, kus kasutajaandmed andmebaasi tabelitesse salvestatakse.



Joonis 9 Teavituste seadistamine

5.2 Rakenduse avakuva

Rakenduse avakuva on vaade, mida näidatakse kasutajale äpi avamisel. Andmed päritakse taustsüsteemilt välja iga kord kui kasutaja rakenduse avab.

Avakuval näidatakse sel päeval otse-eetris olevate mängude nimekirja. Kasutajal on võimalus nende mängude peale klikkides avada mängu staatusele vastava vaate, olgu selleks siis tulevane, käimasolev või juba lõppenud mäng.

Teisena näidatakse avakuval nimekirja uudistest nende avaldamise kuupäeva järjestuses. Uudisele klikkides avaneb vaade, milles saab uudist lähemalt uurida.

Avanevas vaates on näha sellele konkreetsele uudisele määratud pilt koos pealkirja ning sisuga.

Kolmandaks visualiseeritakse kasutajale nimekiri Eesti Jalgpalli Liidu *Youtube'i* lehele laetud videotest, millele klikkides avaneb kasutajale kas seadmesse installitud *Youtube'i* rakendus või selle puudumisel seadmele määratud vaikebrauser.

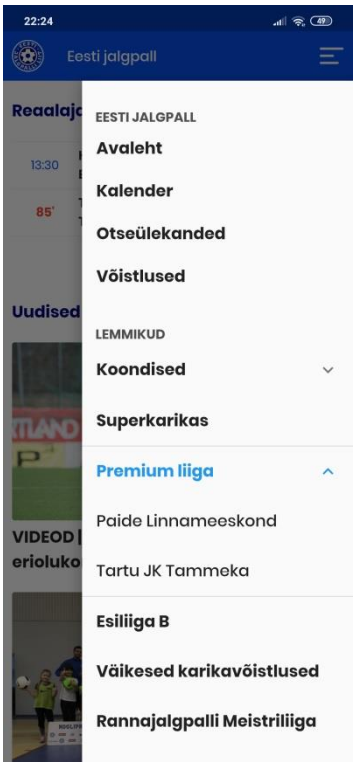
Lisaks on vaate paremas alumises nurgas staatiliselt ujuv ikoon, mille peale klikkides avaneb kasutajale nimekiri tema valitud lemmikutest.

Vaatesse ilmuvad andmed küsitakse, BLoC mustrit kasutades, välja kolme nimekirjana objektidest, mis seejärel vaatesse visualiseeritakse.

Joonis 10 Rakenduse avakuva

5.3 Rakenduse menüü

Mobiilirakenduse menüü koosneb kuuest staatilisest kirjest, millele on lisaks nimekiri kasutaja valitud lemmikutest. Menüü eesmärk on muuta ligipääs soovitud võistlustele ning võistkondadele mugavaks ning aega säästvaks.



Joonis 11 Rakenduse menüü

5.4 Mängu loetelu elemendid

Rakenduse erinevates vaadetes esitatud mängude nimekirjad koosnevad elementidest, mis muudavad oma sisemisi koostisosi sõltuvalt sellest, mis staatuses antud mäng on.

Mängude staatuste võrdlemine toimub kasutaja valitud lemmikute põhjal. Serveripoolse andmebaasi koormuse vähendamiseks salvestatakse kasutaja lemmikud rakenduse sisesesse lokaalsesse SQLite andmebaasi. Antud lahenduse kaudu on vaadete ehitamisel võimalik mänge kasutaja lemmikutega kiirelt võrrelda, mis vähendab nii mobiilse seadme kui serveri ressursi kasutust, muutes rakenduse kasutust kiiremaks.

Mängud jaotuvad tulevasteks, käimasolevateks ja lõppenud mängudeks. Käimasolevad jagunevad omakorda erinevateks mängu faasideks nagu esimene poolaeg, vaheaeg, teine poolaeg ja nii edasi.

Käimasolevate ja tulevaste mängude puhul lisatakse iga mänguelemendi juurde lisaks veel ka teavituste kelluke, millele klikkides tehakse valitud mängu ning kasutaja andmetega päring taustsüsteemile. Päringu tulemusena lisatakse või, juba lisatud mängu

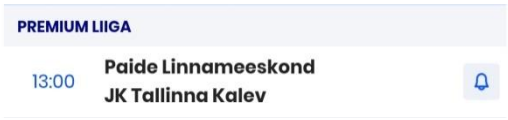
puhul, eemaldatakse antud mäng kasutaja selekteeritud mängude hulgast, millele ta teavitusi soovis saada.

Juhul kui kasutajal on üks või teine mänguga seotud võistkondadest juba lemmikute hulgas, on tal võimalik konkreetse mängu kellukesele klikkides keelata ära selle mängu kohta teavituste saamine.

Joonis 12 Tulevane mäng - 1

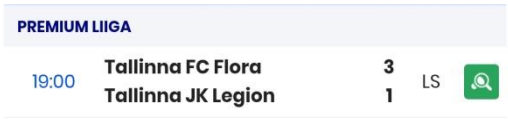
Joonis 13 Käimasolev mäng - 1

Joonis 14 Toimunud mäng - 1



Joonis 15 Tulevane mäng – 2

Joonis 16 Käimasolev mäng – 2



Joonis 17 Toimunud mäng - 2

5.5 Võistluse vaade

Võistluse vaade avaneb kasutajale, kui ta valib menüüs või lemmikute hulgas olevale võistluse nimele klikkides. Võistluse vaade on lahendatud vahekaartide kaudu ning nende muutmist reguleerib ekraani alaosas paiknev vahekaartide menüü.

5.5.1 Võistluse avakuva

Võistluse avakuva (Lisa 4) sarnaneb oma ülesehituselt rakenduse avakuva vaatega. Vaates (Joonis 10) on kasutajal võimalik näha päringu tegemise ajatempliga võrreldes

viite järgnevat mängu. Erinevus seisneb ekraani allosas olevas sakkide menüüs. Antud menüü juhib vaates olevate vahelehtede vahetumise protsessi.

Lisaks on avakuval nähtaval ka hetkel valitud võistlusega seotud uudised, mille peale klikkides avaneb valitud uudis eraldi vaates.

Joonis 18 Võistluse vaate avakuva

5.5.2 Võistluse tabelivaade

Võistluse vaatele järgneval vahelehel (Joonis 11) on kasutajal võimalik näha reaalajas kehtivat tabeliseisu. Kasutaja saab tabelit vasakule ja paremale nihutades nähtavale tuua lisaks mängitud mängudele ja punktide arvule ka võistkonna kohta käivate muude võistluste tulemuste statistikat. Klikkides võistkonna nimele avaneb kasutajale vaade valitud võistkonna andmetega.

Võistluse tabelivaade on üles ehitatud virna (inglise keeles *stack*) meetodil, mis lubab kasutajaliidese elemendid asetada üksteise kohale. Virna alumise osana on vaates võistkonna positsioon, nimi ning logo. Ülemise liigutatava elemendina on võistkonnaga seotud andmed, mis näitavad mängitud mängude tulemuste statistikat.

Joonis 19 Võistluse tabelivaade

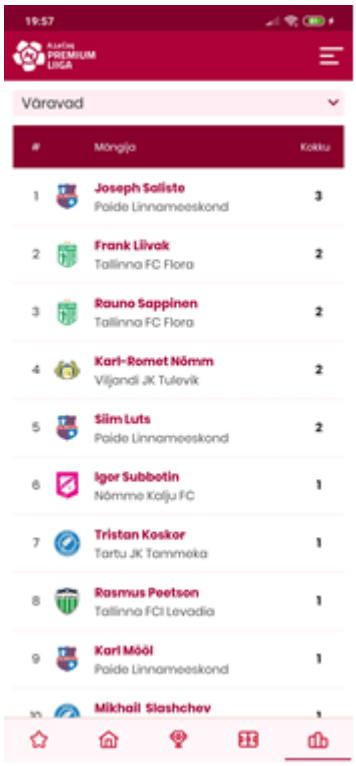
5.5.3 Võistluse mängude vaade

Võistluse kolmandas vaates (Joonis 12) avanevad kasutajale nimekiri kõikidest sellel võistlusel mängitavatest mängudest. Antud nimekiri on jaotatud tulevasteks ning juba toimunud mängudeks. Nimekirja elementidele klikkides avaneb uus vaade vastavalt mängu staatusele.

Joonis 20 Võistluse mängude vaade

5.5.4 Võistluse statistika vaade

Viimasena on võistluse vahekaartide menüü alt leitav statistika vaade (Joonis 13), mis kuvab nimekirja käimasoleva hooaja löödud väravate ja saadud karistuste kohta. Kasutajal on võimalik rippmenüü kaudu valida, milliseid andmeid ta näha soovib. Vastavalt rippmenüü valikule teeb rakendus päringu taustsüsteemile, mis omakorda esitab soovitud andmed. Lisaks on rakenduse kasutajal mängija nimele vajutades võimalik avada vaade, mis sisaldab valitud mängija andmeid koos konkreetse mängija tulemuste statistikaga.



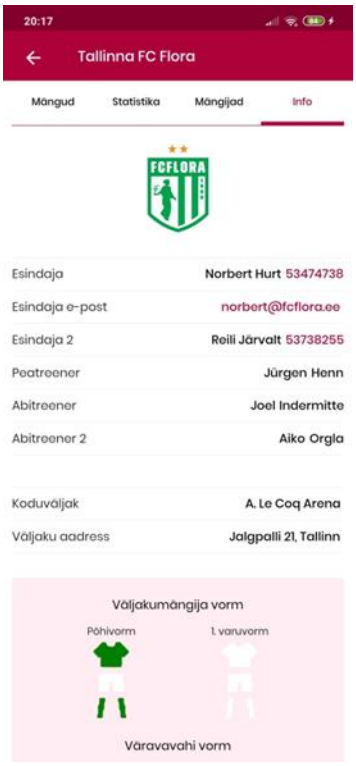
Joonis 21 Võistluse statistika vaade

5.6 Võistkonna vaade

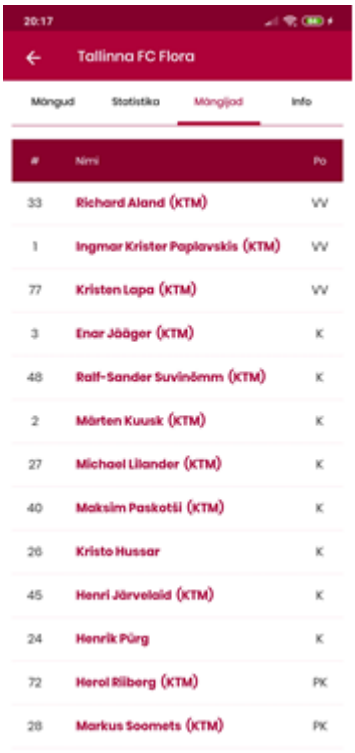
Võistkonna vaates on kasutajal võimalik näha vahelehti, mis on täidetud andmetega valitud tiimi kohta. Esitatavateks andmeteks on võistkonna mängud, registreeritud mängijad (Joonis 16) ning võistkonna kohta käiv üldinfo (Joonis 15).

Registreeritud mängijate vaates kuvatakse kasutajale nimekirja kõikidest käimasoleval hooajal võistkonna nimekirja kantud mängijatest.

Info vaates on nähtav võistkonna üldinfo koos logo, kontaktandmete ning võistkonna mängijate võistlusvormidega.



Joonis 22 Võistkonna info vaade



Joonis 23 Võistkonna mängijate vaade

5.7 Mängija vaade

Mängija vaates (Joonis 16) on nähtavad andmed ja statistika käimasoleva hooaja kohta. Lisaks üldinfole on kasutajal võimalik avatud lehelt vaadata mängija kohta tulemuste statistikat ning filtreerida seda mängija poolt osaletud võistluste suhtes.

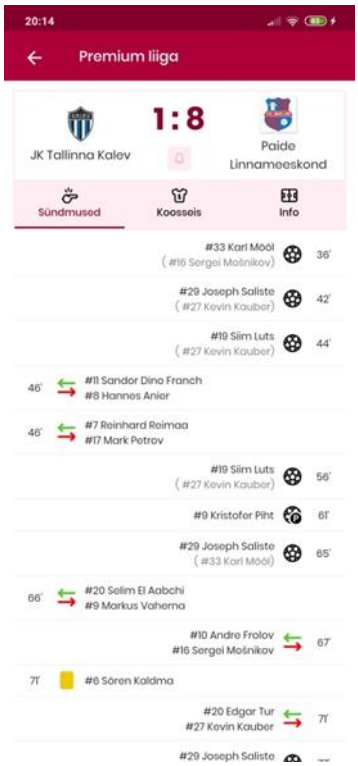
Joonis 24 Mängija profiilivaade

5.8 Mängu vaade

Mängu vaate esimesel alamlehel (Joonis 17) on kasutajale nimekirjana nähtavad kõik ekraanil lahti oleva mängu sündmused.

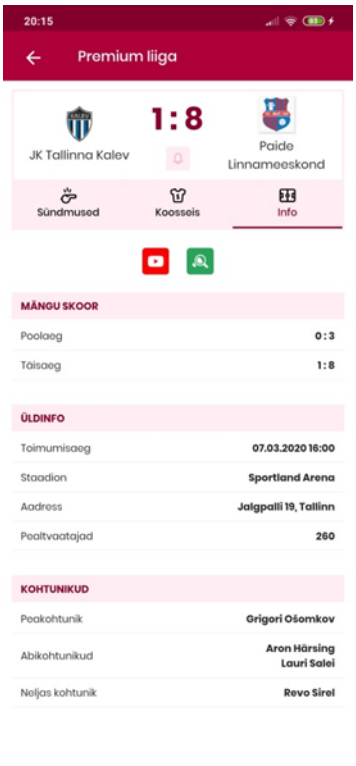
Teisel alamlehel (Joonis 18) visualiseeritakse kasutajale graafiliselt mängus osalenud võistkondade alustavast koosseisu ning selle järel nimekirju vahetusmängijatest ning treenerist. Alustava võistkonna graafiline komponent näidatakse ekraanile 5x7 ruudustikuna, mille igale ruudule vastab mängija positsiooninumber. Mängijate andmed päritakse andmebaasist koos positsioonidega, mille põhjal nad ruudustikku lisatakse. Lisaks graafilisele ruudustikule näidatakse kasutajale nimekirju mõlema võistkonna vahetusmängijatest.

Kolmandalt alamlehelt (Joonis 19) leiab kasutaja andmed ekraanil avatud mängu kohta koos mängu skoori ning mängu kajastava ajakirjanduse ja otseülekande ikoonidega. Ikoonidele klikkides avanevad kasutajale vastavad keskkonnad, *Youtube'i* video puhul *Youtube'i* pleier ning ajakirjanduse artikli puhul telefoni vaikebrauser.



Joonis 25 Mängu sündmuste vaade

Joonis 26 Mängu koosseisu vaade



Joonis 27 Mängu info vaade

5.9 Kalendri vaade

Kalendrivaates on kasutajal võimalik kuupäevavalija (Joonis 20) kaudu otsida mänge. Sisestatud kuupäeva kohta tehakse vastav päring andmebaasi, mis tagastab kasutajale kõik valitud kuupäeval toimuvad mängud (Joonis 21).

Lisaks kuupäevavalijale on võimalus muuta valitud kuupäeva nihutades horisontaalselt paiknevat kuupäevade nimekirja ning klikkides soovitud kuupäevale animeeritakse uus valitud kuupäev kasutaja vaates oleva horisontaalse nimekirja vasakpoolseimaks elemendiks.

Joonis 28 Kalendri vaade -1

Joonis 29 Kalendri vaade - 2

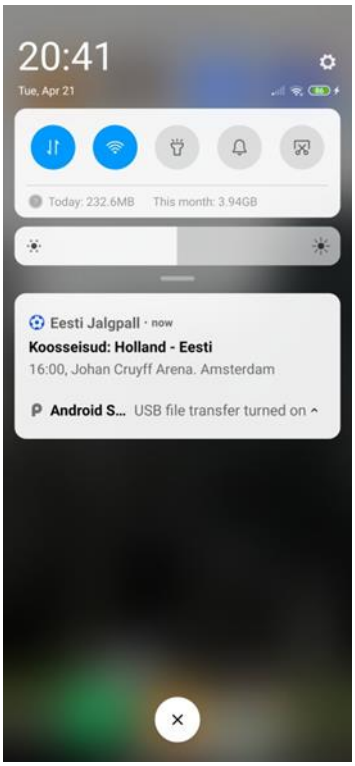
5.10 Teavitused

Teavituste süsteemi konfiguratsioon toimub rakenduse „*Helper*“ klassis olevas „*configureFireBaseMessaging*“ meetodis. Antud meetodis võetakse vastu *Firebase* süsteemist saadetud teavitused ning parsitakse saadud andmed ning edasi antud parameetrite põhjal, rakendatakse vajalik äriloogika.

Teavitusi vastu võttes ilmuvad need kasutajale kahel erineval viisil.

Esimene neist ilmub siis, kui kasutajal on rakendus suletud. Seadme teavituste paneelile ilmub teavitus (Joonis 22), mille peale klikkides avaneb rakendus, loetakse sisse kaasa antud andmed ja kasutaja suunatakse automaatselt teavitusega seotud mängu vaatesse.

Teine viis teavituse ilmumiseks on, kui kasutajal on teavituse saabumise hetkel rakendus avatud. Ekraani ülaosas ilmub nähtavale konteiner (Joonis 23) mängu andmetega, mille peale klikkides suunatakse kasutaja samuti mänguga seotud vaatesse.



Joonis 30 Teavituse vaade - 1

Joonis 31 Teavituse vaade - 2

6 Valminud mobiilse rakenduse analüüs

Hübriidrakenduse beetaversiooni valmimisel viis autor läbi erineva otstarbega teste, mis mõõtsid rakenduse kasutusmugavust ning kasutaja jaoks vajalike andmete leitavust. Testid koostas autor eesmärgiga mõõta rakendusest leitavate andmete leidmise kiirust. Lisaks testimisele kogus autor ka testgrupi käest tagasisidet äpi funktsionaalsuse kohta.

Autor valis välja 10 inimese suuruse testgrupi, kellest

- 3 olid rakenduse arendusega seotud või selle arendusprotsessidest kuulnud;
- 4 olid jalgpalliga seotud ning varasema süsteemi ehk Eesti Jalgpalli Liidu kodulehega tuttavad;
- 3 testgrupi liikmetest ei olnud jalgpalliga seotud.

6.1 Kasutajate tagasiside

6.1.1 Rakenduse võrdlus eelneva lahendusega

Autor viis läbi testid, et analüüsida valminud rakenduse kasutuskiirust võrreldes eelnevalt kasutuses olnud veebilehe lahendusega. Testidest ilmnes, et rakendust kasutades suutsid kasutajad etteantud ülesandeid täita märgatavalt kiiremini. Ülesannete lahendamine mobiilirakenduses oli keskmiselt 3-5 sekundit kiirem kui veebilehe kaudu. Kiirus ilmnes eelkõige testidega, mida läbides oli kasutajal võimalik saada edu mobiilirakenduses rakendatud lemmikute süsteemist.

Suurema osa kasutajate tagasisidest ilmnes, et teste läbisid nad meelsamini mobiilirakenduse kui veebilehe kaudu. Testide läbiviimisel veebilehe kaudu ilmnes kasutajatel frustratsiooni, kuna vajalike andmeteni jõudmiseks oli vajalik teha suurel hulgal erinevaid lisa-klikke.

6.1.2 Rakenduse töökindlus

Uue mobiilirakenduse testimise käigus suuri vigu ei ilmnenud. Andmete edastamine oli stabiilne ning toimiv. Positiivsena tõid rakenduse kasutajad välja võrguühendusega seotud vigade puhul võimaluse lihtsasti andmeid serverist uuesti küsida. Antud funktsionaalsus jättis kasutajatele väga töökindla mulje.

6.1.3 Rakenduse kasutusmugavus

Kasutusmugavus oli rakenduse testijate arvates rahuldav. Positiivsete joontena toodi välja rakenduse disainielementide lihtsuse ning vaadete puhtuse. Kasutajate hinnangul oli rakendust hea vaadata.

Mõningatele testijatele, kes olid varasemat süsteemi juba kasutanud, tekkisid probleemid varasema süsteemi kasutamise harjumuste tõttu. Toodi välja, et kui varasemal süsteemil olid võistluse lehel olles kõik ühejäänud seotud võistlused ühel lehel nähtaval, siis uue mobiilse süsteemi kohaselt tuleb järgmisesse võistluse vaatesse minna soovides avada menüü ning sealt valida võistluste nimekiri.

6.1.4 Rakenduse funktsionaalsus

Tagasisidest selgus, et kasutajad jäid testimise käigus rakenduse funktsionaalsusega rahule. Suur osa kasutajatest tõid eriti positiivselt välja tõuketeavituste saamise võimaluse. Samuti meeldis kasutajatele võimalus lisada nii võistlusi kui võistkondi lemmikute hulka, mis andis kasutajatele võimaluse omada kiiret ligipääsu valitud elementidele.

Negatiivse tagasisidena tõid mõningad kasutajad välja selle, et lemmikute valimisel ei ole võimalust võistkondi nime järgi otsida. Lisaks pakuti välja, et teavituste süsteemi võiks saada veelgi rohkem personaliseerida. Kasutajate arvates võiks rakenduse teavitusi saada seadistada veelgi spetsiifilisemalt. Näitena pakuti välja võimalus valida mängude kohta ainult punaste kaartide või ainult löödud väravate teavituste saamise.

6.2 Tagasiside analüüs

Kasutajate tagasisidest selgus, et loodud rakendus täidab talle seatud lähtetingimused. Kasutajad saavad rakenduse abil sirvida andmeid äpis olevate vaadete kaudu ning reaalajas jälgida mängude tulemusi. Rakendusel on andmete personaliseerimise võimalused, mis aitavad kasutajatel jõuda soovitud andmeteni kiiremini, kui neid oleks saanud veebilahendust kasutades. Kasutajatel on võimalus valida lemmikuid, mille tulemusena ilmuvad need kõrvalmenüüsse kiireks ligipääsuks. Lemmikuid valides on kasutajatel võimalus seadistada ka tõuketeavituste saamist.

Rakendusel on funktsionaalsuse poolest eelis veebilehe lahenduse ees juba oma personaliseerimise ja tõuketeavituste tõttu. Teavituste süsteem aitab kasutajatel pidevalt kursis olla oma valitud lemmikute sportlike saavutustega. Kasutajamugavuse poolestki on rakendusel eelis veebilehe ees. Rakendusest eemaldatud ebavajalik funktsionaalsus ning puhtad disainilahendused muudavad kasutajakogemuse väga meeldivaks. Loodud rakendus on väga töökindel, mida kinnitab nii arenduse käigus tehtud pidev manuaalne testimine kui ka beetaversioonis oleva rakenduse väikesemahuline testimine valitud testgrupiga.

Testimise käigus pakuti autorile välja mitmeid rakenduse edasiarendamise võimalusi. Nendeks olid äpi sisene otsinguvõimalus, mis muudaks rakenduse kasutamise veelgi mugavamaks. Lisaks oli kasutajatel soov teavituste süsteemile veelgi suuremat personaliseerimist võimaldada. Veel anti tagasisidet mõningate vaadete kasutajaliidese optimeerimise võimalikest moodustest, milleks olid värvilahenduste ja ruumi kasutuse vähene muutmine.

7 Kokkuvõte

Eesti jalgpalliga, eelkõige laste ja meeste madalamate liigade jalgpalliga, seotud tulemuste ning muude sportlike andmete kättesaadavus oli tülikas ja nendeni jõudmine ebamugav. Antud diplomitöö eesmärk oli luua mobiilne rakendus, mis lihtsustaks spordialaga seotud andmete kättesaadavust. Püstitatud eesmärgi saavutamiseks uuris ja analüüsis autor juba olemasolevaid spordiinfo jagamise lahendusi ning valis välja parimad moodused rakenduse funktsionaalsuse tagamiseks.

Töö käigus valminud rakendusega saab kasutaja valida soovi korral välja oma lemmikvõistlused ja –võistkonnad ning jälgida mugavalt nende tulemusi. Soovi korral teavitab loodud süsteem kasutajat automaatselt, kui tema lemmikute sportlikes saavutustes on toimunud muutusi ning võimaldab kasutajal jälgida jalgpallimänge reaalajas.

Töö kirjutamise hetkel on antud rakendus beetaversioonis, kuid lähiajal on plaanis tutvustada seda huvilistele avalikuks kasutamiseks.

Kasutatud kirjandus

[1] Eesti Jalgpalli Liit. Eesti Jalgpalli Liidu kodulehekülg. [Online]. http://jalgpall.ee/

[2] LiveScore Ltd. LiveScore: Live Sport Updates. [Online]. https://play.google.com/store/apps/details?id=com.livescore&hl=en

[3] Forza Football. Forza Football - Live soccer scores. [Online]. https://play.google.com/store/apps/details?id=se.footballaddicts.livescore&hl=en

[4] DFL Deutsche Fußball Liga GmbH. BUNDESLIGA - Official App. [Online]. https://play.google.com/store/apps/details?id=com.bundesliga&hl=en

[5] Vipin Jain. (2019, July) The 17 Best Hybrid App Development Frameworks for 2020. [Online]. https://www.konstantinfo.com/blog/the-best-hybrid-app-development-frameworks/

[6] SPEC INDIA. (2019, July) React Native vs. Ionic vs. Flutter: Comparison of Top Cross-Platform App Development Tools. [Online]. https://codeburst.io/react-native-vs-ionic-vs-flutter-comparison-of-top-cross-platform-app-development-tools-71c8011309ac

[7] Drifty Co. Ionic Framework kodulehekülg. [Online]. https://ionicframework.com/

[8] Facebook Inc. React Native · A framework for building native apps using React. [Online]. https://reactnative.dev/

[9] Google LLC. Flutter - Beautiful native apps in record time. [Online]. https://flutter.dev/

[10] MindTech. (2019, May) Flutter vs. React Native vs. Ionic Which is better your App. [Online]. https://yourstory.com/mystory/flutter-vs-react-native-vs-ionic

[11] Google LLC. Firebase Cloud Messaging. [Online]. https://firebase.google.com/docs/cloud-messaging

[12] Google LLC. Firebase Cloud Messaging HTTP protocol. [Online]. https://firebase.google.com/docs/cloud-messaging/http-server-ref

[13] Microsoft Corporation. ASP.NET dokumentatsioon. [Online]. https://docs.microsoft.com/en-us/aspnet/core/?view=aspnetcore-3.1

[14] Microsoft Corporation. Background tasks with hosted services in ASP.NET Core. [Online]. https://docs.microsoft.com/en-us/aspnet/core/fundamentals/host/hosted-services?view=aspnetcore-3.1&tabs=visual-studio

[15] Microsoft Corporation. (2020, Mar.) Dependency injection in ASP.NET Core. [Online]. https://docs.microsoft.com/en-us/aspnet/core/fundamentals/dependency-injection?view=aspnetcore-3.1

[16] Chuks Opia. (2019, Jan.) How to handle state in Flutter using the BLoC pattern. [Online]. https://www.freecodecamp.org/news/how-to-handle-state-in-flutter-using-the-bloc-pattern-8ed2f1e49a13/

Lisa 1 – Võistluse tabeli mudel

```
import 'package:equatable/equatable.dart';

class LeagueTable extends Equatable {
  final int teamId;
  final String teamLogo;
  final String teamName;
  final String gamesPlayed;
  final String gamesWon;
  final String gamesDrawn;
  final String gamesLost;
  final String goalDifference;
  final String totalPoints;

  LeagueTable(
      {this.teamId,
      this.teamLogo,
      this.teamName,
      this.gamesPlayed,
      this.gamesWon,
      this.gamesDrawn,
      this.gamesLost,
      this.goalDifference,
      this.totalPoints});

  factory LeagueTable.fromJson(Map<String, dynamic> json) {
    int goalDifference =
        json['leagueTableScoredGoals'] - json['leagueTableConcededGoals'];

    return LeagueTable(
        teamId: json['leagueTableTeamId'],
        teamLogo: json['leagueTableTeamLogo'].toString(),
        teamName: json['leagueTableTeamName'].toString(),
        gamesPlayed: json['leagueTableGames'].toString(),
        gamesWon: json['leagueTableWins'].toString(),
        gamesDrawn: json['leagueTableDraws'].toString(),
        gamesLost: json['leagueTableLost'].toString(),
        goalDifference: goalDifference.toString(),
        totalPoints: json['leagueTablePoints'].toString());
  }

  @override
  List<Object> get props => [
        teamId,
        teamLogo,
        teamName,
        gamesPlayed,
        gamesWon,
        gamesDrawn,
        gamesLost,
        goalDifference,
        totalPoints
      ];
}
```

Lisa 2 – Võistluse tabeli infotarnija

```
import 'dart:convert';
import 'dart:io';
import 'package:estonian_football_app/constants/app_constants.dart'
    as Constants;
import 'package:estonian_football_app/models/beach_league_table.dart';
import 'package:estonian_football_app/models/league_table_model.dart';
import 'package:estonian_football_app/resources/league_table_repository.dart';
import 'package:http/io_client.dart';

class LeagueTableProvider implements LeagueTableRepository {
  var client = new HttpClient();

  Future<List<LeagueTable>> fetchLeagueTable(int leagueId) async {
    client.badCertificateCallback =
        ((X509Certificate cert, String host, int port) => true);

    IOClient ioClient = new IOClient(client);

    String url = Constants.leagueTableApiUrl + leagueId.toString();
    print(url);

    final response = await ioClient.get(url);
    if (response.statusCode == 200) {
      var jsonData = await json.decode(response.body) as List;
      print(response.body);
      return jsonData.map((i) => LeagueTable.fromJson(i)).toList();
    } else {
      throw Exception('Failed to load league table data');
    }
  }

  @override
  Future<List<BeachLeagueTable>> fetchBeachLeagueTable(int leagueId) async {
    client.badCertificateCallback =
        ((X509Certificate cert, String host, int port) => true);

    IOClient ioClient = new IOClient(client);

    String url = Constants.beachleagueTableApiUrl + leagueId.toString();
    print(url);

    final response = await ioClient.get(url);
    if (response.statusCode == 200) {
      // If the call to the server was successful, parse the JSON
      var jsonData = await json.decode(response.body) as List;

      return jsonData.map((i) => BeachLeagueTable.fromJson(i)).toList();
    } else {
      // If that call was not successful, throw an error.
      throw Exception('Failed to load beach table data');
    }
  }
}
```

Lisa 3 – Rakenduse sisendpunkt

```
void main() {
  WidgetsFlutterBinding.ensureInitialized();
  SystemChrome.setPreferredOrientations([DeviceOrientation.portraitUp])
      .then((_) {
    runApp(
      MyApp(),
    );
  });
}

class MyApp extends StatefulWidget {
  @override
  State<StatefulWidget> createState() {
    return _MyAppState();
  }
}

class _MyAppState extends State<MyApp> {
  @override
  void initState() {
    super.initState();
  }

  @override
  Widget build(BuildContext context) {
    return ChangeNotifierProvider<AppProvider>(
      child: MultiBlocProvider(
        providers: [
          BlocProvider<LeagueNewsDetailsBloc>(
            create: (context) => LeagueNewsDetailsBloc(
              LeagueNewsProvider(),
            ),
          ),
          BlocProvider<LeaguePregameInfoBloc>(
            create: (context) => LeaguePregameInfoBloc(
              LeaguePreGameInfoProvider(),
            ),
          ),
          BlocProvider<TeamPlayersBloc>(
            create: (context) => TeamPlayersBloc(
              TeamPlayersProvider(),
            ),
          ),
          BlocProvider<TeamStatisticsBloc>(
            create: (context) => TeamStatisticsBloc(
              TeamStatisticsProvider(),
            ),
          ),
          BlocProvider<TeamPastGamesBloc>(
            create: (context) => TeamPastGamesBloc(
              LeagueGamesProvider(),
            ),
          ),
          BlocProvider<TeamUpcomingGamesBloc>(
            create: (context) => TeamUpcomingGamesBloc(
              LeagueGamesProvider(),
            ),
          ),
```

```
        BlocProvider<LeagueandteamBloc>(
          create: (context) => LeagueandteamBloc(
            ActiveLeaguesProvider(),
          ),
        ),
      ],
      child: RefreshConfiguration(
        /// Configure the default header indicator.
        /// If you have the same header indicator for each page, you need to set this
        headerBuilder: () => WaterDropHeader(),

        /// Configure default bottom indicator
        footerBuilder: () => ClassicFooter(),

        /// header trigger refresh trigger distance
        headerTriggerDistance: 80.0,

        /// custom spring back animate,the props meaning see the flutter api
        springDescription:
            SpringDescription(stiffness: 170, damping: 16, mass: 1.9),

        /// The maximum dragging range of the head. Set this property if a rush out of the view area occurs
        maxOverScrollExtent: 100,

        /// Maximum dragging range at the bottom
        maxUnderScrollExtent: 0,

        /// This property is incompatible with PageView and TabBarView.
        /// If you need TabBarView to slide left and right, you need to set it to true.
        enableScrollWhenRefreshCompleted: true,

        /// In the case of load failure, users can still trigger more loads by gesture pull-up.
        enableLoadingWhenFailed: true,

        /// Disable pull-up to load more functionality when Viewport is less than one screen
        hideFooterWhenNotFull: false,
        enableBallisticLoad: true,
        child: MaterialApp(
          debugShowCheckedModeBanner: false,
          theme: ThemeData(
              fontFamily: 'Poppins',
              textTheme: TextTheme(
                bodyText2: TextStyle(fontSize: 12),
              ),
              scaffoldBackgroundColor: Colors.white,
              appBarTheme: AppBarTheme(color: AppColors.blueColor2Primary)),
          title: 'Eesti Jalgpall',
          initialRoute: '/startup',
          onGenerateRoute: RouteGenerator.generateRoute,
          localizationsDelegates: [
            RefreshLocalizations.delegate,
            GlobalMaterialLocalizations.delegate,
            GlobalWidgetsLocalizations.delegate,
          ],
          supportedLocales: [
            const Locale('en', 'US'), // English
            const Locale('et', 'EST'), // Estonian
          ],
        ),
      ),
    ),
    create: (BuildContext context) {
      return AppProvider();
    },
  );
 }
}
```

Lisa 4 – Võistluse avakuva kasutajaliidese ülesehitus

```
class LeagueHomeFragment extends StatefulWidget {
  final LeagueColor leagueColor;
  final TabController tabController;

  const LeagueHomeFragment({Key key, this.leagueColor, this.tabController})
      : super(key: key);

  @override
  State<StatefulWidget> createState() {
    return _LeagueHomeFragmentState();
  }
}

class _LeagueHomeFragmentState extends State<LeagueHomeFragment>
    with AutomaticKeepAliveClientMixin {
  ActiveLeagueWithoutTeams league;
  LeagueColor leagueColor;
  TabController tabController;

  @override
  void initState() {
    super.initState();
    league = Provider.of<AppProvider>(context, listen: false).league;
    leagueColor = widget.leagueColor;
    tabController = widget.tabController;

    BlocProvider.of<LeagueHomeBloc>(context)..add(GetLeagueHome(league.id));
  }

  @override
  void didChangeDependencies() {
    super.didChangeDependencies();
    league = Provider.of<AppProvider>(context, listen: false).league;
  }

  @override
  Widget build(BuildContext context) {
    super.build(context);
    return new Scaffold(
      body: SingleChildScrollView(
        child: Container(
          margin: EdgeInsets.only(left: 10, right: 10),
          child: Column(
            children: <Widget>[
              BlocListener<LeagueHomeBloc, LeagueHomeState>(
                listener: (context, state) {
                  if (state is LeagueHomeError) {
                    Scaffold.of(context).showSnackBar(
                      SnackBar(
                        backgroundColor: AppColors.snackbarBackground,
                        elevation: 0,
                        duration: Duration(days: 1),
                        behavior: SnackBarBehavior.floating,
                        content: Text(Message.networkError),
                        action: SnackBarAction(
                          label: Message.loadAgain,
                          textColor: AppColors.snackbarText,
                          onPressed: () =>
                              BlocProvider.of<LeagueHomeBloc>(context)
                                ..add(GetLeagueHome(league.id)),
                        ),
                        shape: RoundedRectangleBorder(
                          borderRadius: BorderRadius.all(
                            Radius.circular(TextSize.snackBarRadius),
                          ),
                        ),
                      ),
                    );
                  }
                },
```

```
                child: BlocBuilder<LeagueHomeBloc, LeagueHomeState>(
                    builder: (widget, state) {
                  if (state is LeagueHomeLoading) {
                    return BuildLoading();
                  } else if (state is LeagueHomeLoaded) {
                    return Column(
                      children: <Widget>[
                        if (state.leagueHome.leagueGames.length != 0)
                          Column(
                            children: <Widget>[
                              LandingpageHeader(Message.nextGames,
                                  Message.allGames, leagueColor.color1, () {
                                tabController.animateTo(3);
                              }, leagueColor.color2Primary),
                              HomeScreenGamesList(
                                viewLocation: 1,
                                leagueGames: state.leagueHome.leagueGames,
                                leagueColor: leagueColor,
                              ),
                            ],
                          ),
                        SingleHeader(Message.news, leagueColor.color1, 20, 20),
                        News(
                          news: state.leagueHome.leagueNews,
                          leagueColor: leagueColor,
                        ),
                      ],
                    );
                  } else if (state is LeagueHomeGamesInitial) {
                    return Container();
                  } else if (state is LeagueHomeError) {
                    return Center(child: Text(state.message));
                  } else {
                    return BuildLoading();
                  }
                }),
              ),
            ],
          ),
        ),
      ),
    );
  }

  @override
  bool get wantKeepAlive => true;
}
```