Tallinna Ülikool
Digitehnoloogiate instituut

SPORDIENNUSTUSPORTAALI MOBIILIRAKENDUSE ARENDUS KASUTADES REACT NATIVE RAAMISTIKKU

Bakalaureusetöö

Autor: Mikk Sillaste
Juhendaja: Romil Rõbtšenkov

Autor: ..... „ ..... „2018
Juhendaja: ..... „ ..... „2018
Instituudi direktor: ..... „ ..... „2018

Tallinn 2018

Autorideklaratsioon

Deklareerin, et käesolev bakalaureusetöö on minu töö tulemus ja seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on viidatud.

....

(kuupäev) (autor)

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina Mikk Sillaste (sünnikuupäev: 01.03.1987)

1. annan Tallinna Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose “Spordiennustusportaal mobiilirakenduse arendus kasutades React Native raamistikku”, mille juhendaja on Romil Rõbtšenkov, säilitamiseks ja üldsusele kättesaadavaks tegemiseks Tallinna Ülikooli Akadeemilise Raamatukogu repositooriumis.

2. olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile.

3. kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest tulenevaid õigusi.

Tallinnas, _____

(digitaalne) allkiri ja kuupäev

Sisukord

Sissejuhatus

Mobiilsetel seadmetel internetti kasutavate inimeste hulk kasvab igapäevaselt. 2017. aasta augustis moodustas mobiilse interneti kasutus 52.64% kogu maailma veebiliiklusest (Statista, 2017). Mobiilse interneti kiire levik mõjutab otseselt ka spordiennustust vahendavate kihlveokontorite tegevust. Suur osa spordiennustustest tehakse online-kihlveokontorites, panustamise juures on seega väga suur roll info ajakohasuse faktoril. Prognoositakse, et aastal 2018 moodustab mobiilne panustamine 40% kogu veebis panustamise turust (Pushtechnology, 2015). Samuti on välja toodud, et 90% nutiseadmete kasutajatest eelistavad rakendusi mobiilsetele veebilehtedele (Khalaf, 2016).

Mobiilirakenduse arendamine võib olla väga kulukas. Eriti juhul, kui on soov luua platvormipõhine mobiilirakendus nii Android kui ka iOS operatsioonisüsteemile. See tähendab kahe täiesti eraldiseisva arendusmeeskonna ülevalpidamist. Suurematel kihlveokontoritel on selline võimalus mobiilirakenduse tarbeks olemas. Väiksematel ning alles alustavatel spordiennustusportaalidel võib mobiilirakenduse arendamine esialgu üle jõu käia ja nad piirduvad kohaneva veebilehega (ingl *responsive website*), mis on kasutatav mobiilis või tahvelarvutis läbi veebilehitseja. Paratamatult jääb selline kasutajakogemus alla mobiilirakendusele.

Käesoleva bakalaureusetöö autor on pikalt töötanud spordiennustusega seotud ettevõttes ja huvitub mobiilirakenduste arendusest. Seetõttu tuligi idee luua spordiennustusportaali mobiilirakendus nii Android kui ka iOS operatsioonisüsteemile, kasutades selleks platvormist sõltumatut (ingl *cross-platform*) React Native[1] raamistikku. Arendatav rakendus on mõeldud näidismalliks potensiaalsetele klientidele, kes on mobiilirakendust mitteomavad spordiennustust vahendavad portaalid. Mobiilirakendus muudaks nende teenuse erinevatel nutiseadmetel paremini kasutatavaks. Vastava rakenduse arenduse ettevalmistused tegi autor oma seminaritöös "Spordiennustusportaali mobiilirakenduse nõuete analüüs ja disain" (Sillaste, 2017).

Spordiennustusportaal on keerukas süsteem, mis koosneb mitmest osast:

- sisu (ingl *content*) ehk ennustuseks saada olevad spordisündmused, turud, koefitsendid ning spordisündmuste ja turgude staatused;

[1] https://facebook.github.io/react-native/

- kasutajad ehk kõik mis on seotud portaali kasutajate konto andmetega;
- panustamine ehk see osa, mis seob omavahel sisu ja kasutajad.

Antud bakalaureusetöö eesmärk on luua spordiennustusportaali mobiilirakenduse sisu osa Android ja iOS operatsioonisüsteemidele nii, et seda on võimalik integreerida potensiaalse kliendi juba toimiva kasutajate haldamise ja panustamise süsteemiga.

Bakalaureusetöö on jaotatud kolme peatükki. Esimene peatükk teeb ülevaate rakenduse funktsionaalsetest nõuetest ja disainist. Järgmine tutvustab arenduseks kasutatavat React Native raamistikku. Kolmas peatükk kirjeldab põhjalikult kõiki rakenduse arendusetappe tuues välja arenduse käigus tekkinud probleemid ja nende võimalikud lahendused.

1. Funktsionaalsus ja disain

Peatükk annab ülevaate spordiennustusportaali mobiilirakenduse põhifunktsionaalsustest ja disainist, mis olid välja töötatud autori seminaritöös “Spordiennustusportaali mobiilirakenduse nõuete analüüs ja disain” (Sillaste, 2017). Eesmärgi saavutamiseks tegi autor ülevaate kolmest, juba olemasolevast, samasse valdkonda kuuluvast rakendusest. Nende rakenduste võrdlusest saadud tulemuste ja autori enda teadmiste põhjal kirjeldati miinimumnõuded, millele arendatava rakenduse esimene versioon peaks vastama. Loodi ka disaini prototüübid kõigile esialgse rakenduse vaadetele.

1.1. Põhifunktsionaalsused

Rakenduse funktsionaalsed nõuded saab tinglikult jagada kolme gruppi:

- üldised nõuded;
- kasutaja andmetega seotud toimingud;
- panustamisega seotud tegevused.

Üldiste nõuete all saab välja tuua omadused, mis on vajalikud peaaegu iga mobiilirakenduse arendusel ja ei ole otseselt seotud spordiennustusportaali rakendusega. Nendest kõige tähtsam on rakenduse olemasolu nii Android kui ka iOS platvormil. Teiseks on oluline võimalus märguandeid (ingl *push notifications*) saata ja vastu võtta, seda teevad pea kõik mobiilirakendused.

Kasutaja andmetega seotud funktsionaalsustest on samuti enamus sellised, mis on iseloomulikud ka paljudele muudele veebi- ning mobiilirakendustele. Näiteks kasutaja autentimine ja uute kasutajate registreerimine. Kuna kasutajate autentimiseks, registreerimiseks ja andmete töötlemiseks on palju erinevaid meetodeid, siis antud töös autor konkreetselt sellele osale rakendusest suuremat tähelepanu ei pööra. Rakendus peab olema arendatud nii, et seda on võimalik integreerida erinevate kasutajate haldamise süsteemidega. Samas leidub kasutaja andmetega seotud nõuete hulgas ka sellised, mis on spetsiifilised just hasartmängudega tegelevatele portaalidele. Siia alla kuulub kindlasti rahakott (ingl *wallet*), mille kaudu saab kasutaja kõikidel oma rahalistel tehingutel silma peal hoida. Veel on nendeks näiteks kontojääk, erinevad võimalused raha sisse- ja väljakandmiseks, kogu rahaliste tehingute ajalugu. Sarnaselt

kasutaja andmete haldamisega on erinevaid võimalusi ka rahakoti haldamiseks. Tegutsevad mitmed ainult rahakoti teenuse pakkujad. Sellest tulenevalt peab ka valmiv mobiilirakendus olema ehitatud nii, et seda on võimalik siduda erinevate rahakottidega, vastavalt tellija soovile.

Käesolev bakalaureusetöö keskendub peamiselt panustamisega seotud funktsionaalsustele, mis on põhiline osa spordiennustusportaalist. Portaali kasutajad peavad saama panustada kahte erinevat tüüpi mängudele: *pre-match* ehk veel mitte alanud mängudele ja *live* ehk käimasolevatele mängudele. Panuse tegemise funktsioonalsuse koha pealt ei ole nende kahe vahel suuri erinevusi. Mõlema puhul peavad olema näha kõik sel hetkel antud mängule panustamiseks saada olevad turud koos reaalajale vastavate koefitsentide ja turu staatusega, mis määratakse *back-end* API poolt. Suurima erinevusena nõuete koha pealt võib välja tuua andmete värskendamise, mis *live* mängudel peab olema tunduvalt kiirem kui *pre-match* sündmustel. Käimasolevatel mängudel ei tohiks andmete värskendamine kauplemise (ingl *trading*) platvormi ja rakenduse *front-end* lehe vahel olla rohkem kui viis sekundit. Mitte alanud mängude puhul on andmete värskendamise kriteerium palju leebem ning see võib varieeruda 10 sekundist kuni mõne minutini. Loomulikult sõltuvad need nõuded konkreetsest kliendist ning rakendust peab saama vastavalt kliendi vajadustele konfigureerida.

Kõikide ennustuste kinnitamine toimub panustussedeli (ingl *betslip*) kaudu, millel kuvatakse kasutajale kogu informatsioon võimalike panuste kohta. Turgudel olevate valikute (ingl *selection*) peale vajutades saab kasutaja lisada endale sobivad valikud panustussedelile, kus talle näidatakse vastavalt nende arvule võimalikud panuste tüübid. Kui kasutaja on valinud panuse tüübi ja sisestanud panuse suuruse, siis näidatakse talle tema ennustuse võimalik võidusumma vastavalt panuse suurusele, valiku koefitsendile ja panuse tüübile. Miinimum ja maksimum panuse suurus sõltub kasutaja enda ja vastatava mängu limiitidest. Need on paika pandud kihlveo kontori poolt kasutaja eelnevat panustamise ajalugu ja panustamiseks valitud turule seatud limiite arvestades. Informatsiooni panuste limiitide kohta saab panustussedel *back-end* API'st. Sinna saadetakse päring panuse andmetega ning saadakse vastu panust lubav või keelav vastus koos vastava veateatega.

1.2. Disain

Lähtuvalt autori enda kogemustest ja seminaritöö käigus välja selgitatud nõuetest tuleb spordiennustusportaali mobiilirakenduse esimese versiooni puhul disaini poole pealt põhirõhk asetada kasutajamugavusele. Kõige olulisem on kiire panuste tegemise süsteem ning lihtne ja arusaadav navigeerimine rakenduse erinevate vaadete vahel. Kasutaja peab vähese vaevaga leidma üles just talle vajalikud mängud. Samuti on tähtis, et rakenduse väljanägemine erinevatel platvormidel oleks võimalikult ühesugune. Kuna arendatav rakendus on mõeldud näidismalliks klientidele, siis ei ole värvilahendusele väga suurt tähelepanu pööratud. Esialgses demorakenduses on kasutatud põhiliselt musta ja valget tooni. Spordiennustusportaali vaadete disaini saab jagada kolmeks osaks: navigatsioon, kasutaja isikuandmed ja panustamine.

Lihtsasti arusaadav menüü ja navigeerimine rakenduse erinevate osade vahel on kindlasti üheks heaks kasutajamugavuse näitajaks. Loodava rakenduse põhimenüü on nähtav rakenduse jaluses. Selle kaudu saab kasutaja minna avalehele, algavate mängude lehele, juba alanud mängude lehele ja minu panuste lehele. Samuti on kiire panuse tegemise võimalus väga tähtis osa spordiennustusportaali kasutajamugavusest. Panustussedel, mille kaudu tehakse panuseid, on kasutajatele kogu aeg nähtav ja ligipääsetav rakenduse päises. Erinevate vaadete lingid jaluses ning päises on kuvatud ikoonidena Jaluses on lisatud ikoonile ka vastava lehe nimetus. Nii ikoonid kui ka lingi nimi on seadistatavad vastavalt tellija soovile.

Kasutaja isikuandmete osa koosneb viiest erinevast vaatest, mille disaini prototüübid valmisid seminaritöö “Spordiennustusportaali mobiilirakenduse nõuete analüüs ja disain” (Sillaste, 2017) käigus ja on välja toodud töö lisades (vt Lisa 1):

- mitte sisse loginud kasutaja avaleht;
- sisse loginud kasutaja avaleht;
- kasutaja registreerimise leht;
- kasutaja isikuandmete leht;
- kasutaja rahakonto leht.

Nagu ka nimest võib aru saada, siis avaleht on vaade, mis kuvatakse kasutajale rakenduse avamisel. Antud lehel kuvatud informatsioon oleneb sellest, kas kasutaja on rakenduse avamise hetkel sisse logitud või mitte. Sisse logitud kasutaja näeb enda enda konto andmeid, linke

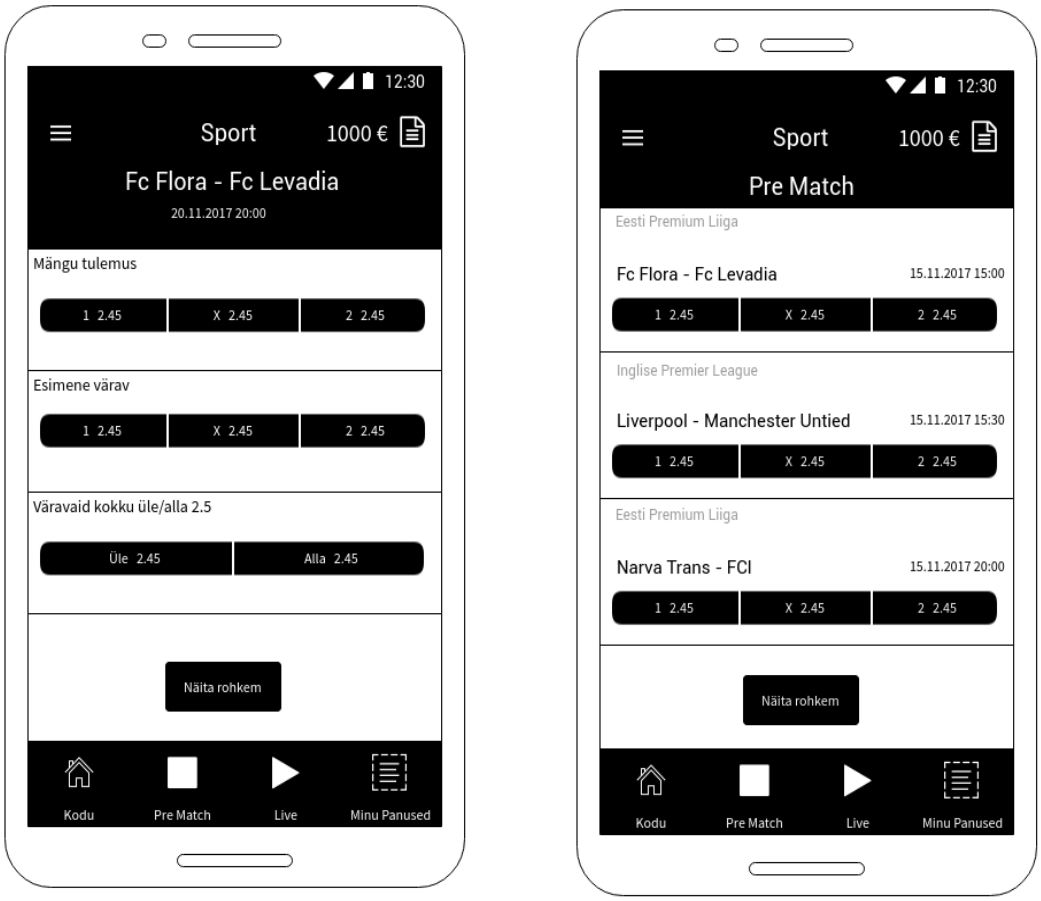
kasutaja andmetega seotud vaadetele ja nuppu välja logimiseks. Mitte sisse logitud kasutaja saab sisestada kasutajanime ja parooli ning nende abil rakendusse sisse logida. Seda loomulikult siis, kui sisestatud andmed on valiidsed. Seal samas on ka nupud, mille kaudu on võimalik minna uue kasutaja registreerimise lehele või uuendada oma parooli, juhul kui see on meelest läinud. Kasutaja registreerimise leht on vorm uue kasutaja loomiseks. Seal on kuvatud väljad, mis on vaja täita, et luua uus kasutajakonto ja nupp registreerimise avalduse esitamiseks. Hiljem saab neid andmeid vaadata ja muuta isikuandmete lehel, millele on ligipääs registreeritud sisse loginud kasutajal. Rahakonto leht, mis võimaldab kasutajatel hallata kõiki spordiennustusportaali kontoga seotud rahalisi tehinguid, peab näitama tehingute ajalugu, kus sisse- ja väljamaksed on üksteisest kindlalt eristatavad. Samuti on sellel lehel ka nupud sissemakse ja väljamakse tegemiseks.

Spordiennustuportaali põhifunktsionaalsus on panustamine. See tähendab, et klientidele tuleb pakkuda võimalikult head kasutajakogemust neid huvitavate mängude leidmisel ja ennustuste tegemisel. Väga olulisel kohal on panustamisega seotud lehtede disain. Esialgse rakenduse panustamise poole pealt saab välja tuua kuus põhivaadet, mille disaini prototüübid valmisid seminaritöö "Spordiennustusportaali mobiilirakenduse nõuete analüüs ja disain" (Sillaste, 2017) käigus ja on välja toodud töö lisades (vt Lisa 2):

- *pre-match* mängude leht;
- *live* mängude leht;
- *pre-match* konkreetse mängu leht;
- *live* konkreetse mängu leht;
- panustussedel;
- minu panuste leht.

Pre-match mängude lehe (vt Joonis 1) avamisel on sellel kuvatud kuni viis järgmisena algavat mängu, mis on järjestatud algusaja järgi varasemad eespool. Samal lehel on ka nupp kõikide *pre-match* mängude laadimiseks. Iga mängu kohta selles loendis on kuvatud mängu algusaeg, võistlejad ja mängu võitja turg koos vastavate valikute ja koefitsentidega, mille peale vajutades lisatakse see kasutaja panustussedelile. Kõikide mängude laadimise nupule vajutamisel avaneb kasutajale loend spordialadest, millele on hetkel võimalik *pre-match* panuseid teha. Spordiala nimele vajutades avaneb vastava spordiala leht. Antud lehel on kuvatud kõik regioonid ja võistlused, mis hetkel on *pre-match* panustamiseks saadaval. Võistluse peale vajutades näidatakse kasutajale selle võistluse veel mitte alanud mängud, kuhu on võimalik panuseid teha.

Iga mängu nimetus, nii *pre-match* mängude avalehel, kui ka võistluse lehel, on link selle konkreetse mängu lehele (vt Joonis 1). Konkreetse *pre-match* mängu lehel on näidatud sündmuse algusaeg, võistlejad ja kõik selle mängu ennustamiseks saada olevad turud. Igal turul on mitu valikut, millele vajutades see lisatakse panustussedelile.

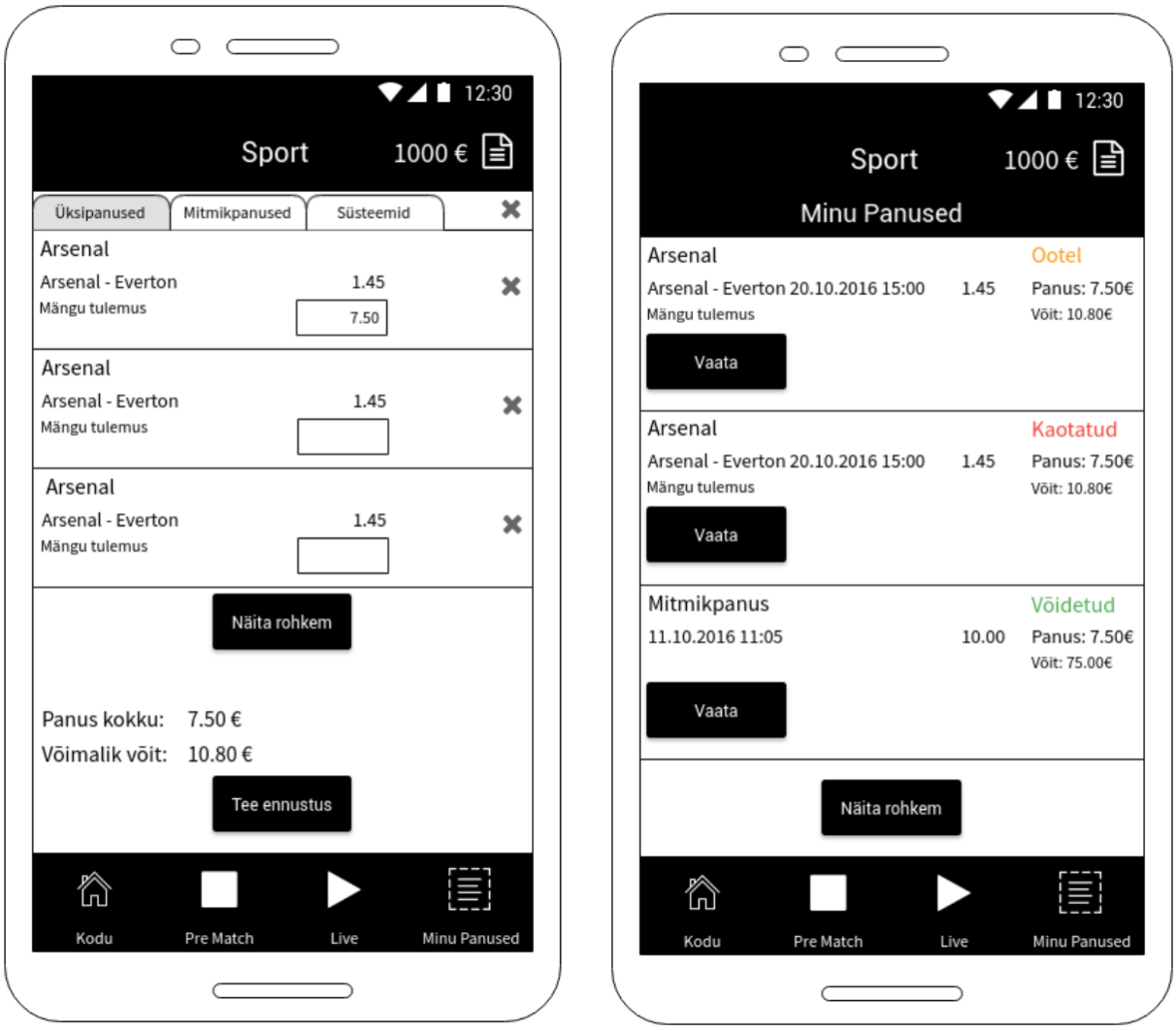


Joonis 1. Pre-match mängude (vasakul) ja konkreetse pre-match mängu vaate (paremal) disaini prototüübid (Sillaste, 2017)

Live mängude lehe avamisel näeb kasutaja spordialade kaupa kõiki hetkel käimasolevaid mänge, mis on panustamisele avatud. Nii nagu mitte alanud sündmuste puhul on siingi iga mäng loendis lingiks konkreetse mängu lehele. Iga mängu kohta on kuvatud mängu võitja turg koos vastavate valikute ja koefitsentidega. Valiku peale vajutades lisatakse see kasutaja panustussedelile. Erinevus mitte alanud mängudega on vaid see, et juba alanud sündmuste korral on ekraanil näidatud mängu hetke staatus, skoor ja võistlejad. Konkreetse *live* mängu lehel on kuvatud kõik selle mängu ennustamiseks saada olevad turud, kus iga valik on nupp, millele vajutades see lisatakse panustussedelile. Nii *pre-match* kui ka *live* sündmuste turgude valikud saavad olla panustamiseks avatud (ingl *open*) või peatatud (ingl *suspended*). Avatud valiku nupu taustavärv

on must ja peatatud valiku nupu taustavärv on hall. Seda selleks, et kasutajatel oleks koheselt nähtav, millised valikud on hetkel panustamiseks avatud ja millised mitte.

Panustussedel (vt Joonis 2) koosneb kolmest kaardist (ingl *tab*): üksikpanused (ingl *singles*), mitmikpanused (ingl *multis*) ja süsteemid (ingl *systems*). Kõikidel nendel kaartidel on näidatud kasutaja poolt panustamiseks valitud turud. Iga valiku kohta peab olema kuvatud valikunimi, mäng, turg ja koefitsent. Samuti on panustussedelil koht panuse sisestamiseks ja kasutajale on näidatud kogu panustatav summa koos võimaliku võiduga. Valikute eemaldamiseks panustussedelilt on iga valiku taga kuvatud "X"-kujuline ikoon. Panuste kinnitamiseks on panustussedeli alaosas "Kinnita" (ingl *submit*) nupp. Kõiki kinnitatud panuseid näeb kasutaja "Minu panused" lehel (vt Joonis 2).



Joonis 2. Panustussedeli (vasakul) ja "Minu Panused" vaate (paremal) disaini prototüübid (Sillaste, 2017)

Üksikpanuste juures on näidatud valik, mäng, turg, panuse kinnitamise aeg, panustatud summa, võimalik võit ja panuse staatus. Mitmikpanustel on kuvatud panuse kinnitamise aeg, panustatud summa, võimalik võit, panuse staatus ja nupp “Vaata”. Sellele nupule vajutades näidatakse kõik valikud, mängud, turud ja koefitsendid, mis on seotud vastava panusega. Panusel saab olla kolm staatust:

- ootel (kollane), kui panus on kinnitatud, kuid mängude tulemused ei ole veel teada;
- kaotatud (punane), kui kinnitatud panuse tulemus on teada ja see on kaotatud;
- võidetud (roheline), kui kinnitatud panuse tulemus on teada ja see on võidetud.

Välja selgitatud funktsionaalsed nõuded ja valminud disaini prototüübid on aluseks rakenduse edasistele arendusetappidele. Järgmine peatükk teeb ülevaate antud bakalaureusetöö raames valmiva mobiilirakenduse arenduseks valitud raamistikust.

2. React Native

Arendusvahendite valikul sai määravaks autori eesmärk luua mobiilirakendus Andorid ja iOS operatsioonisüsteemidele nii, et ei peaks arendama kahte rakendust eraldi. Tänapäeval on platvormist sõltumatuid (ingl *cross-platform*) mobiilirakenduse arendust võimaldavaid vahendeid mitmeid. Lõplik valik sõltub tavaliselt konkreetse rakenduse nõuetest ja arendaja eelnevast kogemusest. Kuna antud töö raames arendatava spordiennustusportaali mobiilirakenduse jõudlus peab olema võimalikult lähedane päris *native* rakendusele, siis välistas autor hübriidrakenduste arenduseks mõeldud raamistikud. Käesoleva töö autori valik langes React Native[2] raamistikule, mis keskendub hübriidrakenduse asemel *native* mobiilirakenduse arendamisele. Kõige tõsisemaks konkurendiks React Native raamistikule on samasugust funktsionaalsust pakkuv Xamarin[3], mis on Microsofti poolt arendatav raamistik iOS, Android ja Windows mobiilirakenduste arendamiseks. Kui React Native raamistikus on kasutusel JavaScript programmeerimiskeel, siis Xamarin'is on selleks C#. Jõudluse ja kasutajaliidese graafilise poole pealt on React Native ja Xamarin võrdsed (AltexSoft, 2018). Mõlemad raamistikud on avatud lähtekoodiga ja tasuta, kuid Xamarin'i kasutamiseks on vaja Microsoft Visual Studio[4] arenduskeskkonda. Visual Studio'st on küll olemas tasuta versioon, kuid kogu funktsionaalsuse kasutamiseks on vajalik *Professional* või *Enterprise* litsentsi ostmine (AltexSoft, 2018). See ja suurem arendajate kogukond olid peamisteks teguriteks, miks autor otsustas React Native raamistiku kasuks. Järgnevalt tutvustatakse React Native raamistikku lähemalt ja tehakse ülevaate selle kasutamisest juba olemasolevates rakendustes.

2.1. Ülevaade raamistikust

Tavalise *native* mobiilirakenduse loomisel tuleb Android rakenduse arenduseks kasutada Java programmeerimiskeelt ja iOS rakenduse jaoks Swift'i või Objective-C'd. See tähendab, et neid tuleb arendada üksteisest täiesti eraldi, kuigi funktsionaalsused on ühesugused. React Native on raamistik, mille abil saab arendada mobiilirakendusi Android ja iOS operatsioonisüsteemidele, kasutades selleks vaid JavaScript programmeerimiskeelt. See raamistik erineb sama võimalust pakkuvatest hübriidrakenduste raamistikest kuna JavaScriptis kirjutatud kood kompileeritakse *native* Java ja Swift koodi. Sellest tulenevalt on React Native rakendused oma jõudluse poolest

[2] https://facebook.github.io/react-native/
[3] https://www.xamarin.com/
[4] https://www.visualstudio.com/vs/

väga sarnased *native* mobiilirakendustele, mis on kirjutatud operatsioonisüsteemile omases keeles. Seda väidavad ka Blink[5] nutikella mobiilirakenduse arendajad, kelle testide põhjal ei ole märgata mingit jõudluse vahet React Native ja päris *native* rakenduse vahel (Sriraman, 2016). Samas tuleb meeles pidada, et React Native ei ole 100% *native*, kuid seda on võimalik arendada nii *native* rakenduse sarnaseks kui vaja (Kazula & Grajcar, 2018). See tähendab, et on ikkagi võimalik osa rakendusest kirjutada kasutades Java't või Swift'i. React Native raamistik on loodud Facebooki poolt ja põhineb kasutajaliideste arenduseks mõeldud React[6] teegil. React Native sai alguse Facebook'i häkatonil aastal 2013. 2015. aasta märtsis tehti see ametlikult avatud lähtekoodiga kättesaadavaks GitHub'is[7]. Tänaseks on sellest saanud üks enim kasutatavaid raamistikke mobiilirakenduste arendamisel. Lisandunud on moodul (ingl *plugin*) Windows platvormi jaoks, mis võimaldab React Native rakendusi luua ka Windows operatsioonisüsteemiga seadmetele. React Native'i suure populaarsuse põhjustena saab välja tuua järgmised tegurid (Novick, 2017):

- Tööriistad rakenduses vigade leidmiseks ja kõrvaldamiseks.
- *Live reload*, mis tähendab, et pärast koodi muudatuste salvestamist laetakse rakendus automaatselt uuesti koos tehtud muudatustega. See kiirendab oluliselt arendajate tööd.
- *Hot reload*, mis tähendab, et pärast koodi muudatuste salvestamist laetakse uuesti ainult see komponent, mida muudeti, mitte terve rakendus.
- Komponentide hierarhia kontrollimine.
- Veebiarendusest inspireeritud kujundustehnikad.
- 80% - 90% JavaScriptis kirjutatud koodist on kasutatav nii Android kui ka iOS platvormil.

React Native raamistik on pidevas arenduses nii Facebooki tarkvarainseneride kui ka üha kiiremini kasvava kasutajate kogukonna poolt.

[5] https://www.visualstudio.com/vs/
[6] https://reactjs.org/
[7] https://github.com/

2.2. Olemasolevad rakendused

Tänaseks on juba tuhandeid mobiilirakendusi, mis on täielikult või osaliselt kirjutatud React Native raamistikus. Siinkohal on välja toodud mõned tuntumad neist (Who's using React Native?, kuupäev puudub):

- sotsiaalvõrgustik Facebook;
- Facebook'i reklaamide haldamise tarkvara Facebook Ads Manager;
- piltide ja videode jagamise sotsiaalvõrgustik Instagram;
- lühiajalist majutust pakkuv Airbnb;
- internetis suhtlemist võimaldav Skype;
- elektriautosid valmistav Tesla;
- jaekaubandusettevõte Walmart.

Facebook Ads Manager[8] rakendus oli kõige esimene täielikult React Native raamistikus kirjutatud mobiilirakendus nii iOS kui ka Android operatsioonisüsteemile. Tol hetkel oli React Native üpris värske tehnoloogia, mis ei olnud ennast veel täielikult tõestanud. Facebook'i programmeerijad puutusid selle projekti käigus kokku nii mõnegi probleemiga, millest olulisematena võib välja tuua järgmised (Witte & von Weitershausen, 2015):

- Keeruline on töötada eraldi iOS ja Android koodirepositooriumiga. Isegi siis kui see on automatiseeritud ja kasutatakse parimaid tööriistu. Sellest projektist alates kasutab Facebook mõlema platvormi jaoks ühist JavaScripti koodirepositooriumit.
- Kõiki muudatusi peab põhjalikult testima mõlemal platvormil.

Tuntud sotsiaalmeedia rakenduse Instagram[9] tarkvarainseneride andmete kohaselt on nende React Native raamistikus kirjutatud komponentide jagatud JavaScript'i koodi osa iOS ja Android platvormidel järgnev (Instagram Engineering, 2017):

- postituse propageerimine – *post promote* 99%;
- SMS'i teel kasutaja kindlaks tegemine – *SMS captcha checkpoint* 97%;
- kommentaaride modereerimine – *comment moderation* 85%;

[8] iOS: https://itunes.apple.com/us/app/facebook-ads-manager/id964397083?mt=8, Android: https://play.google.com/store/apps/details?id=com.facebook.adsmanager
[9] iOS: https://itunes.apple.com/app/instagram/id389801252?mt=8, Android: https://play.google.com/store/apps/details?id=com.instagram.android

- reklaamide haldamine – *lead gen ads* 87%;
- tõukemärguannete seaded – *push notification settings* 92%.

Näiteks Walmart'i[10] veebipoe mobiilirakenduse arendajad võtsid React Native'i kasutusele hübriidrakenduse asemel. Nemad põhjendasid raamistiku valikut järgmiste kasuteguritega (Bresnan, Safi, Patel, & Keerti, 2016):

- 95% rakenduse koodist saab jagada iOS ja Android platvormi vahel.
- Kogu funktsionaalsuse rakendamise eest vastutab üks tiim.
- Suurepärane arendaja kogemus, sest koodi muudatusi näeb koheselt.
- Üks programmeerimiskeel mõlema platvormi jaoks.
- Sama kasutajaliidese koodi saab kasutada iOS ja Android platvormil.
- Sama äriloogika koodi saab kasutada nii iOS ja Android platvormil kui ka veebirakendustes.
- React Native rakenduse arenduseks kulub vähem aega kui *native* rakenduse arenduseks mõlemale platvomile.
- Võimalus avaldada iOS ja Android platvormi rakendus samal ajal.
- React Native rakenduse jõudlus on samaväärne *native* rakendusega.
- Võimalus kasutada platvormi spetsiifilist kasutajaliidese disaini.
- Samu automatiseeritud teste saab kasutada iOS ja Android platvormil.
- Võimalus rakendusi kiiresti uuendada ilma rakenduse poe (ingl *app store*) kinnitamise protsessita.

Samas toodi välja ka mõned kitsaskohad, millele tasub tähelepanu pöörata (Bresnan et al., 2016):

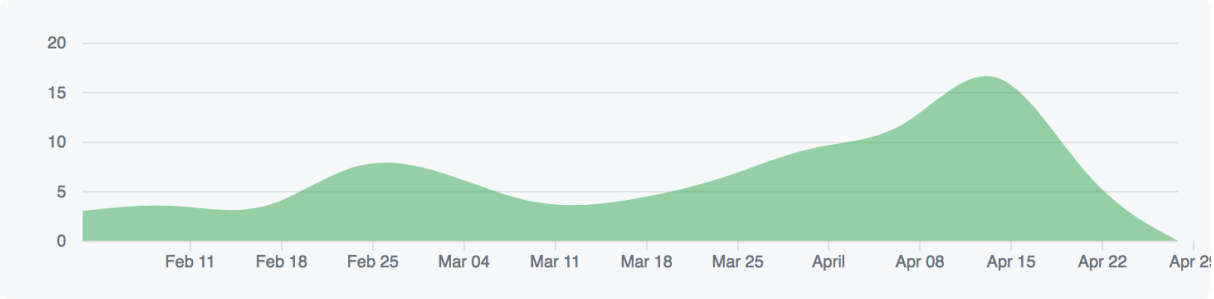
- Funktsionaalsuse erinevused iOS ja Android platvormide vahel. iOS platvorm toetab rohkem funktsionaalsusi kui Android.
- React Native kood käitub silumisrežiimis (ingl *debug mode*) ja tavalises režiimis erinevalt, mis raskendab vigade põhjuste leidmist.
- Koormus- ja jõudlustestide tegemiseks ei ole piisavalt häid vahendeid.
- React Native raamistiku versiooni uuendamisel võib tekkida rakenduse osade mitteühilduvus.

[10] iOS: https://itunes.apple.com/us/app/walmart-app-shopping-savings/id338137227?mt=8, Android: https://play.google.com/store/apps/details?id=com.walmart.android

React Native raamistik on lisanud mobiilirakenduste arendusse palju uusi funktsionaalsusi, mis on tuttavad veebiarendusest, kuid mida ei ole enne kasutatud mobiiliarenduses. Olemasolevate React Native rakenduse arendajate poolt välja toodud pidepunkte ja React Native raamistiku uuenduslikke omadusi saab ära kasutada spordiennustusportaali mobiilirakenduse arendusel. Järgnev peatükk kirjeldab detailselt kogu rakenduse arendusprotsessi.

3. Rakenduse arendus

Rakenduse koodi versioonihaldusel kasutab autor GitHub veebikeskkonda. Antud rakenduse lähtekood on kättesaadav aadressil https://github.com/mikksillaste/DemoBetMobileApp ja aktiivne arendustöö on kestnud üle kahe kuu (vt Joonis 3).

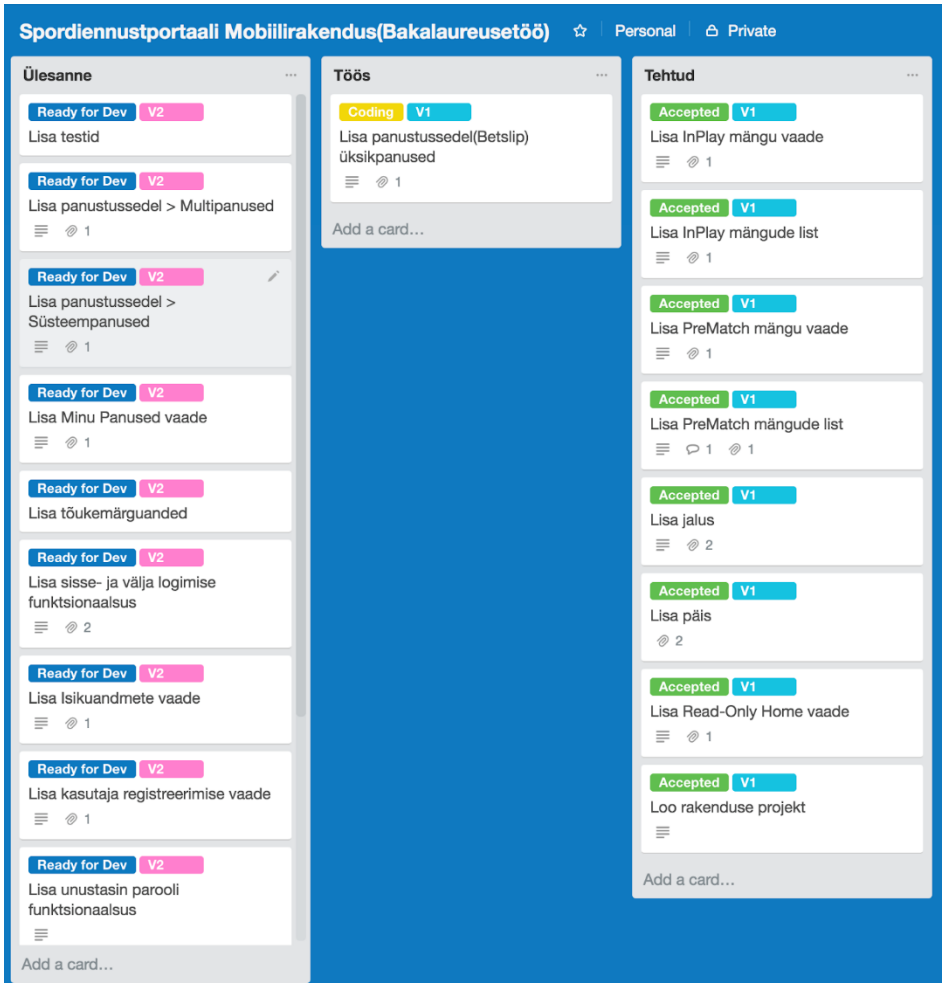


Joonis 3. Koodi uuenduste graafik GitHub'is

Töös kasutatavad koodinäited on tähistatud `eristiiliga`. Käesoleva bakalaureusetöö raames valmib spordiennustusportaali mobiilirakenduse esimene versioon, mis keskendub peamiselt sellele, et kogu kauplemise (ingl *trading)* platvormil panustamiseks saada olev sisu on kasutajale nähtav. Rakendus peab olema saadaval nii Android kui ka iOS platvormile ja seda peab olema võimalik siduda kliendipoolse kasutajate haldamise süsteemiga. Esimese versiooni käigus luuakse uus React Native rakenduse projekt, navigatsioonisüsteem, menüü erinevate vaadete vahel liikumiseks, täieliku funktsionaalsusega *pre-match* ja *live* mängude vaated ning kohahoidjad (ingl *placeholders*) rakenduse teises versioonis arendatavatele või kliendipoolsest süsteemist integreeritavatele vaadetele. Esialgne arendatav rakendus on ingliskeelne kuna demorakenduse arendusel kasutatava serveri poolt tulevad andmed on inglise keeles. Rakenduse disaini prototüübid on eestikeelsed.

Rakenduse funktsionaalsete nõuete ja disaini prototüüpide põhjal jagas autor, kes on ühtlasi ka antud rakenduse ainus arendaja, kogu vajaliku arendustöö ülesanneteks. Iga ülesanne kirjeldab üht rakenduse vaadet või funktsionaalsust. Ülesanded jagunesid kahte gruppi (vt Joonis 4):

- V1 ehk versioon 1 – vaated ja funktsionaalsused, mis peavad olema rakenduse esimeses prototüübis.
- V2 ehk versioon 2 – vaated ja funktsionaalsused, mis lisatakse järk-järgult peale rakenduse esimese prototüübi avaldamist.



Joonis 4. Spordiennustusportaali mobiilirakenduse tööde nimekiri ja hetke staatust näitav Trello tahvel esimese versiooni arenduse lõpufaasis

Arenduse hetkeseisu tõhusama jälgimise võimaldamiseks kasutas arendaja Trello[11] projektijuhtimise tarkvara, tekitades sinna iga ülesande jaoks eraldi kaardi (vt Joonis 4). Iga kaardi peal on kirjas ülesande raames tehtava töö kirjeldus ja hetke staatus .

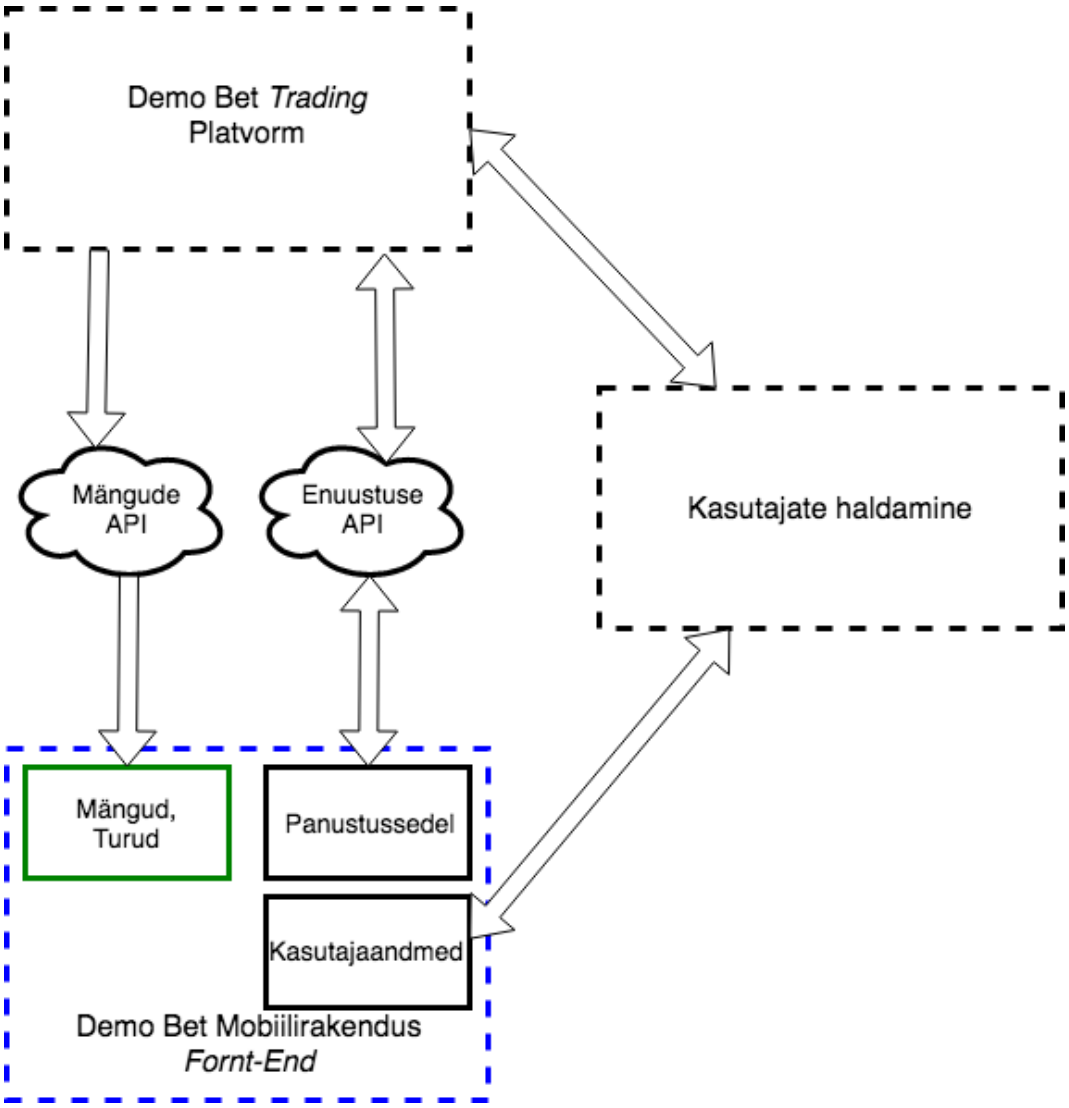
3.1. Rakenduse arhitektuur

Arendatav mobiilirakendus on kliendipoolne rakendus hõlmates endas kolme komponenti:

- mängud, turud – ennustuseks saada olevad spordisündmused, turud, koefitsendid ning spordisündmuste ja turgude staatused;
- kasutajaandmed – kõik mis on seotud portaali kasutajate konto andmetega;
- panustussedel – osa, mis seob omavahel kaks eelmist ja kinnitab panuse reeglitepärasust.

[11] https://trello.com

Kõik panuste tegemiseks vajaminevad andmed tulevad *trading* platvormilt, millega mobiilirakendus suhtleb vastavate API'de kaudu. Serveri poolel on mängude, ennustuse ja kasutajate haldamise API'd (vt Joonis 5). Mängude API'st saab kliendipoolne rakendus kogu informatsiooni *trading* platvormist tulevate mängude, turgude, koefitsentide ja mängude staatuse kohta. Ennustuse API'sse saadab kliendipoolne rakendus päringu panuse andmetega ja saab vastuseks panust keelava või lubava sõnumi.



Joonis 5. Rakenduse arhitektuur; kogu spordiennustusportaali mobiilirakendus on välja toodud sinise katkendjoonega ja käesoleva töö raames valmiv komponent rohelise joonega

Kasutajate andmetega seotud toimingud käivad läbi kasutajate haldamise API, mis võib olla igal kliendil eraldiseisev teenus. Kasutajate haldamise süsteem peab suhtlema ka *trading* platvormiga kuna panuste kinnitamisel on vaja teada kasutaja andmeid.

3.2. Arenduskeskkonna ülesseadmine ja projekti loomine

Enne React Native mobiilirakenduse arendusega alustamist tuleb üles seada arenduseks vajalik keskkond. Olenevalt arendamiseks kasutatava arvuti operatsioonisüsteemist on selleks mitu võimalust. Käesoleva rakenduse arendusel kasutab autor macOS High Sierra operatsioonisüsteemiga MacBook Pro sülearvutit. Esmalt tuleb paigaldada vajalikud integreeritud programmeerimiskeskkonnad (Napier, 2017):

- Xcode[12] iOS rakenduse "ehitamiseks" (ingl *build*) ja käitamiseks (ingl *run*) simulaatoril.
- Android Studio[13] Android rakenduse "ehitamiseks" ja käitamiseks emulaatoril.
- WebStorm[14] React Native rakenduse projekti loomiseks ja koodi kirjutamiseks.

Lisaks on veel vaja paigaldada Java arenduskeskkond ehk JDE[15] ja mõned käsurea tööriistad. Kõige olulisemad neist on Node.js[16] ja NPM[17]. Node.js on JavaScripti käitussüsteem (ingl *runtime*), mille peale on ehitatud React ja React Native raamistikud (Napier, 2017). NPM on Node'i paketihaldur (ingl *Node Package Manager*), mis võimaldab erinevate Node'i pakettide, näiteks React Native'i, lisamist arendusprojekti (Napier, 2017). Nüüd saab macOS X operatsioonisüsteemi *Terminal'is* käsuga `sudo npm install react-native-cli -g` paigaldada globaalse React Native käsurea liidese (ingl *React Native Command Line Interface*). Sellega on kogu React Native projekti alustamiseks vajalikud tööriistad arendaja masinasse paigaldatud. Kindlasti on veel palju tööriistu, mida React Native rakenduste arendamisel kasutatakse, kuid mis antud projekti puhul ei tundunud autorile vajalikud.

React Native projekti loomiseks on erinevaid võimalusi. Antud rakenduse puhul kasutab autor WebStorm programmeerimiskeskkonda, kus saab väga lihtsalt luua uue React Native projekti koos vastava failistruktuuri ja konfiguratsiooniga (React Native, 2018).

DemoBetMobileApp projekti juurkaustas on kolm kausta:

- `android`, mis sisaldab Android *native* Java koodi.

[12] https://developer.apple.com/xcode/
[13] https://developer.android.com/studio/index.html
[14] https://www.jetbrains.com/webstorm/
[15] http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html
[16] https://nodejs.org/en/
[17] https://www.npmjs.com/

- `ios`, mis sisaldab iOS *native* Objective-C koode.
- `node_modules`, mis sisaldab installeeritud Node.js pakette.

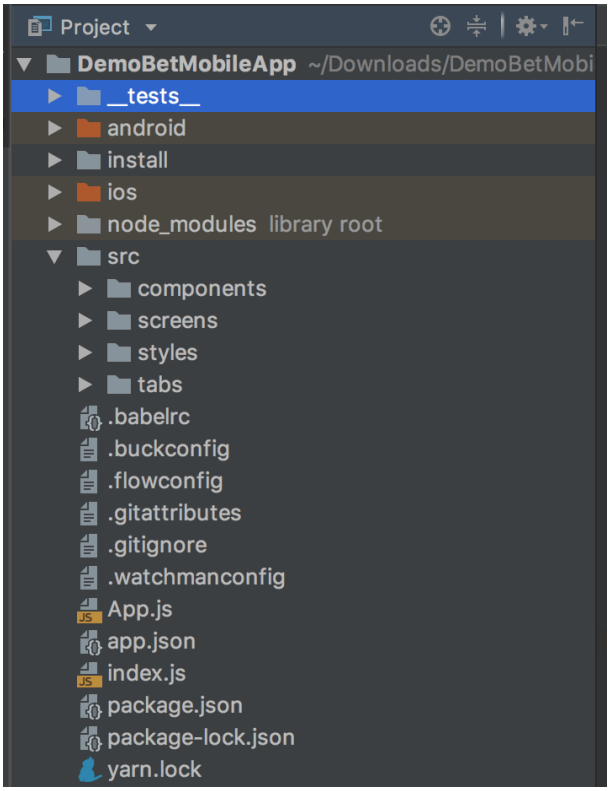
Samuti on projekti juurkausta tekitatud mõningad failid. Enamus neist on erinevad konfiguratsioonifiailid, kuid seal on veel kaks tähtsat JavaScripti faili: `App.js` ja `index.js`. `App.js` failis on kood, mis kuvatakse rakenduse käivitamisel ja `index.js` fail registreerib rakenduse alguspunkti komponendi (Abernathy, 2018). Nendele lisaks loob autor projekti juurkausta uue kausta nimega `src` ja sinna sisse omakorda neli kausta:

- `components`, kus asuvad kohandatud komponendid, mida tuleb mitmel pool rakenduses kasutada.
- `screens`, kus asuvad rakenduses olevad vaated, mis ei avane jaluses olevast vahelehtede ribamenüüst(ingl *tab bar)*.
- `styles`, kus asuvad rakenduses kasutatavad kujunduse failid.
- `tabs`, kus asuvad rakenduse vaated, mis avanevad jaluses olevast vahelehtede ribamenüüst.

Peale testide kirjutamiseks vajaliku Jest[18] platvormi lisamist projektile lisandub eelpool mainitutele veel `__tests__` kaust, kus on rakenduse komponenditestid[19]. Sellega on rakenduse lõplik failistruktuur paigas (vt Joonis 6).

[18] https://facebook.github.io/jest/

[19] Üksiku programmi või mooduli test, mille otstarve on tagada analüüsi- ja programmeerimisvigade puudumine. (http://www.eki.ee/dict/its/index.cgi?Q=unit+test&F=M&C06=et&C10=1)



Joonis 6. Spordiennustusportaali mobiilirakenduse Demo Bet projekti failistruktuur

Kohalikus masinas rakenduse jooksutamiseks iOS simulaatoril tuleb käsureal navigeerida rakenduse juurkausta ja sisestada käsk `react-native run-ios`. Vaikimisi avatakse rakendus iPhone 6'l, kuid seadet on võimalik muuta lisades käsu lõppu konkreetse seadme muutuja: `--simulator="iPhone X"` (React Native, kuupäev puudub). Viimasel juhul avaneb rakendus iPhone X simulaatoril. Rakenduse Android versiooni nägemiseks on esmalt vaja käima panna Android emulaator ja seejärel käsureal korraldusega `react-native run-android` käivitada React Native rakendus.

3.3. Kohandatud komponendid ja kujundus

Projekti loomise alapeatükis välja toodud failistruktuuris oli mainitud kaust `components`, kus asuvad kohandatud komponendid, mida saab mitmel pool rakenduses kasutada. Rakenduse lähtekoodi parema haldamise ja hilisemate muudatuste lihtsustamise seisukohalt on tähtis hoida mitmes kohas kasutatavate komponentide põhikoodi ühes kohas. React Native võimaldab luua kohandatud (ingl *custom*) komponente, mida saab teistesse komponentidesse hiljem sisse importida ja seal kasutada. Samuti saab iga komponendi juures ära määrata parameetrid, mida on vaja kaasa anda selle kasutamisel. Käesolevas töös arendatava rakenduse puhul on nendeks

erinevad nupud, mida mitmetes vaadetes kasutatakse. Turu (ingl *market*) valiku nupp on üks sellistest. Iga turu võimalik valik on lehel kuvatud nupuna ja selle peale vajutades saab valiku lisada panustussedelile. Turu valiku nupp on kasutusel kõikides mängude kuvamise vaadetes. Turu valiku nupu lähtekood asub `components` kaustas `MarketButton.js` failis. Koodis on kirjeldatud nupu kasutamiseks vajalikud parameetrid, funktsionaalsus ja kujundus (vt Koodinäide 1).

```
static propTypes = {
       selectionName: PropTypes.string.isRequired,
       price: PropTypes.string.isRequired,
       style: PropTypes.number.isRequired,
       fixtureId: PropTypes.string.isRequired,
       fixtureName: PropTypes.string.isRequired,
       selectionId: PropTypes.string.isRequired,
};
<TouchableOpacity
       activeOpacity={0.5}
       style={this.props.style}
       onPress={() => {alert(
       "Selection Name: " + this.props.selectionName + "\n" +
       "Selection Id: " + this.props.selectionId + "\n" +
       "Fixture Name: " + this.props.fixtureName + "\n" +
       "Fixture Id: " + this.props.fixtureId + "\n" +
       "Selection Price: " + this.props.price );
 }
}>
<Text numberOfLines={1} style={ styles.marketButtonText}>
       {this.props.selectionName}
</Text>
       <Text style={styles.marketButtonText}>{ this.props.price }</Text>
</TouchableOpacity>
```

Koodinäide 1. Kohandatud React Native nupu komponent koos nõutud parameetrite ja lisatud kujundusega

React Native raamistikus ei ole spetsiaalset keelt või süntaksit komponentide kujunduse määramiseks. Kõik komponendid aksepteerivad atribuuti `style` ja rakenduse kujundus lisatakse JavaScriptis. (React Native, kuupäev puudub). Siiski, selle asemel, et igale komponendile eraldi kujundus lisada, on parem tava hoida kogu rakenduse stiil ühes kohas. Selleks on `StyleSheet.create` käsk, millega saab luua stiililehe, kuhu panna kogu rakenduses kasutatavad kujunduse klassid. React Native stiiliatribuudid sarnanevad CSS atribuutidele ja käituvad sarnaselt, välja arvatud nimed, mis kirjutatakse *CamelCase*'is (vt Koodinäide 2).

```
import { StyleSheet } from 'react-native';
export default StyleSheet.create({

container: {
      flex: 1,
      backgroundColor: '#FFF',
 },
 fixturePageMatchState: {
      fontSize: 10,
      color: '#FFF',
 },
 fixtureListBody: {
      flex: 20,
      backgroundColor: '#FFF',
      justifyContent: 'center',
 },

});
```

Koodinäide 2. Lõik Demo Bet React Native rakenduse kujunduse koodist

React Native's saab komponendile kujunduse lisada ka tingimuslikult. Näiteks antud töö raames loodud rakenduse disaini nõuete kohaselt pidi vastavalt hetke olekule turu valiku nupule rakendama erineva kujunduse (vt Koodinäide 3).

```
import styles from '../styles/Styles';

style={item.Suspended === true ? ( styles.marketSuspendedButton ) :
styles.marketOpenedButton}
```

Koodinäide 3. Tingimusliku kujunduse lisamine

Antud rakenduse arendusel otsustas autor kogu kujunduse panna ühte `Styles.js` faili, mis asub `styles` kaustas. Selle kasutamiseks rakenduse vaadetes tuleb `Styles.js` fail enne vaate komponenti importida.

3.4. Navigatsioon

Rakenduse üks tähtsaim osa on erinevate lehtede vahel navigeerimine. Mobiilirakenduste kolm kõige populaarsemat navigatsioonimustrit on vahekaardi navigatsioon (ingl *tab navigation*), kus vaated vahetuvad kasutades vahekaardi menüüd (ingl *tab bar), stack* navigatsioon, kus vaated asetatakse teineteise peale ja modaalne navigatsioon (ingl *modal navigation*), kus vaade ilmub teiste vaadete peale (Shkut, kuupäev puudub). Demo Bet rakenduses on kasutatud kõiki kolme. React Native raamistikus on palju valmis komponente, mis töötavad nii Android kui ka iOS rakenduses. Komponente saab importida vastavatest teekidest, mis on võimalik projektile lisada käsurealt NPM'i kaudu. Navigatsiooni komponentide loomiseks on lisatud `react-navigation` teek. Samuti on vaja `react-native-vector-icons` teeki, mille abil saab

menüü linkidele lisada vajalikud ikoonid. Seejärel said imporditud vajalikud komponendid `App.js` faili, kus asub kogu rakenduse navigatsiooni kood (vt Koodinäide 4).

```
import {
StackNavigator,
TabNavigator,
} from 'react-navigation';
```

Koodinäide 4. Navigatisooni komponentide importimine react-navigation teegist

Rakenduse peamine menüü on jaluses asuv vahekaardi navigatsioon, mille kaudu on ligipääs rakenduse põhivaadetele: "*Home*", "*Pre-Match*", "*Live*" ja "*My Bets*". Lisaks põhimenüüle saavad kasutajad soovitud lehtedele liikuda põhivaadetel olevatelt nuppudelt, loendi üksustelt ja päises olevalt "Tagasi" (ingl *Back*) või "X" nupult. "X" nupp on kuvatud *modal* vaadetel, mis avades kuvatakse eelnevalt avatud olnud vaate peale (Shkut, kuupäev puudub). Antud rakenduses on sellisteks lehtedeks "Minu Andmed" (ingl *My Account*) ja "Tehingud" (ingl *Transactions*), mille täielik funktsionaalsus lisatakse rakenduse teises versioonis. Lehele navigeerides saadab `react-navigation` kaasa `navigation` objekti, millele on ligipääs vaate komponendilt. `Navigation` objekti kolm peamist omadust on (Shkut, kuupäev puudub):

- `navigate(screen)` funktsioon, mille abil saab navigeerida igale teisele vaatele.
- `goBack()` funktsioon, millega saab minna tagasi eelmisele vaatele.
- `state` objekt, kus on `routeName`, mis sisaldab praeguse vaate nime ja `params` objekti parameetritega, mis saadeti praegusele vaatele eelmiselt lehelt.

Näiteks *Pre-Match* võistluste loendi lehelt ühe kindla võistluse lehele liikudes antakse `navigation.state` objekti parameetritena kaasa võistluse nimi ja võistluse identifikaator (ingl *competition id*), et näidata õige võistluse mänge (vt Koodinäide 5).

```
onPress={() => navigate('PreMatchCompetitionFixtureList',{type: (item.Value),
      competitionId: (item.RouteValue)}
)}
```

Koodinäide 5. Navigation objekt lisatud parameetritega

Võistluse vaate komponendilt pääseb nendele parameetritele ligi `this.props.navigation.state` kaudu. Võistluse nime saab kätte `type` parameetri (vt Koodinäide 6) ja võistluse identifikaatori `competitionId` parameetri kaudu.

```
PreMatchCompetitionFixtureList: {
      screen: PreMatchCompetitionFixtureList,
      navigationOptions: ({ navigation }) => ({
      title: `${navigation.state.params.type}`,
      }),
}
```

Koodinäide 6. Võistluse nime parameetri type kasutamine pealkirjana võistluse lehe päises

Kui võrrelda Android ja iOS rakendust, siis navigatsiooni ja menüü lisamisel funktsionaalsuse poole pealt probleeme ei esinenud. Mõned väiksemad erinevused ilmnesid kasutajaliidese disainis. Esialgu oli probleemiks, et `StackNavigator` komponendi päise pealkirja ei olnud näha Android rakenduses. Kui see probleem sai lahendatud kujunduse omaduste muutmisega, siis tekkis uus probleem. Android rakenduses ei ole `StackNavigator` komponendi pealkiri kuvatud päise keskel, kuigi lisatud stiili atribuutide järgi peaks see nii olema. Samuti on iOS rakenduses pealkiri keskel. Kuna seda probleemi ei õnnestunud täielikult parandada, siis kõige parem lahendus hetkel on `StackNavigator` komponendi päise pealkirja joondamine iOS rakenduses keskele ja Android rakenduses päise vasakusse äärde (vt Joonis 7). Kusjuures kood on mõlemal platvormil täpselt sama.

Joonis 7. Pre-Match komponendi päis iOS (vasakul) ja Android (paremal) rakenduses

Väikesed erinevused on ka jaluses asuval ikoonidega peamenüül, kuid need on tingitud platvormide eripärast (vt Joonis 8).

Joonis 8. Peamenüü rakenduse jaluses iOS (vasakul) ja Android (paremal)

Siinkohal võib React Native'i hea omadusena välja tuua, et see ei tähenda kummalegi platvormile eraldi koodi kirjutamist, vaid piisab ühest ja samast JavaScript koodist.

3.5. *Pre-match* mängude vaated

Pre-match vaadete hulka kuulub kokku 5 vaadet (vt Lisa 3):

- *pre-match* mängude loend;
- *pre-match* sportide loend;
- *pre-match* võistluste loend;
- *pre-match* valitud võistluse mängude loend;
- *pre-match* konkreetse mängu leht.

Pre-match lehtedele on kasutajal ligipääs läbi rakenduse jaluses asuva menüü "*Pre-Match*" vahekaardi (ingl *tab*). Seal on vaikimisi kuvatud järgmised viis algavat mängu ajalises järjekorras koos mängu võitja turu valikutega. Mängude all on nupp "*All Pre Match*", mille peale vajutades avaneb loend sportidest. Iga sport on lingiks selle spordi võistluste lehele, kus võistluse peale vajutades näidatakse kasutajale vastava võistluse kõik mitte alanud mängud. Iga mängu nimi, nii võistluse lehel, kui ka *pre-match* mängu lehel, on link konkreetse mängu lehele. Vaadetel olevate andmete vahendamiseks rakenduse ja serveripoolse mängude API vahel kasutatakse React Native raamistiku poolt pakutavat *Fetch* API't[20], mis oma olemuselt on sarnane *XMLHttpRequest*[21] rakendusliidesele. Kui leht on rakenduses avatud, siis päritakse *pre-match* mängude andmeid vastavast API lõppsõlmest (ingl *endpoint*) iga viie sekundi järel ja uuendatakse komponendi konstruktoris algväärtustatud olekus (ingl *state*) olevat andmemassiivi (vt Koodinäide 7). Kui komponendi olek muutub, siis sellele vastavalt uueneb ka kasutajaliideses kuvatav informatsioon.

[20] https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API
[21] https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest

```
async getPreMatchUpNextFixtures(){

fetch('https://betstream2.betgenius.com/betstream-
view/customer/betgenius/'+'product/geniusbet/component/prematchupnexttree/json',
{method: "GET"})

.then((result) => result.json())
 .then((res) => {

this.setState({
             fixturesData:res.Leaves,
 loading: true
 })
})
.catch((error) => {
       console.error(error);

});

}
```

Koodinäide 7. Fetch päringu näide koos oleku muutmisega

Spordid, võistlused, mängud ja turud on vaadetel näidatud loenditena. Selleks on React Native raamistikus olemas mitmed erinevad valmis komponente, mida saab lihtsalt importida ja kasutada. Antud rakenduses kasutab autor selleks `FlatList` komponenti (vt Koodinäide 8). `FlatList` komponent on React Native's saadaval alates versioonist 0.43. Omades palju kasulikke funktsioone on see parim valik nii lihtsamate kui ka keerulisemate kerimisloendite loomisel (Novick, 2017). Kaks põhilist atribuuti, mida peab `FlatList`'i kasutamisel teadma on `data` ja `renderItem` (Carli, 2017). Esimene on andmemassiiv, millest loend tehakse ja teine on funktsioon, mis võtab massiivist üksiku elemendi ja loob selle jaoks komponendi (Carli, 2017) (vt Koodinäide 8).

```
<FlatList
data = {this.state.sportsList}
 renderItem = {({ item }) =>
             <View style={styles.sportsListItem}>
             <Text
       style={styles.sportsListItemSportName}
 onPress={() =>
navigate('PreMatchRegionCompetitionTree',
{
type: (item.Value),
sportName: (item.RouteValue)
}
)}
>
{item.Value}
</Text>
 </View>
}
keyExtractor={item => item.Id}
/>
```

Koodinäide 8. FlatList kasutamine sportide loendi kuvamisel

`FlatList` komponent töötab identselt nii iOS kui ka Android rakenduses ja see sobib suurepäraselt andmete näitamiseks *pre-match* vaadetel vastavalt disaini prototüüpidele.

3.6. *Live* mängude vaated

Live mängude vaateid on rakenduse esimeses versioonis kokku kaks: *live* mängude loend ja konkreetse *live* mängu leht (vt Lisa 4). Mõlemad on suuremas osas sarnased vastavate *pre-match* vaadetega. Oluline erinevus on andmete uuendamise sageduses, mis peab *live* mängude puhul olema kiirem. Sellest tulenevalt päritakse API'st andmeid iga ühe sekundi järel. Lehtedel näidatava informatsiooni koha pealt on suurim erinevus selles, et mängu algusaja asemel on *live* mängude puhul kuvatud kasutajatele mängude hetke staatus ja skoor. Siinkohal tuli arvestada sportide eripäradega, nii skoori märkimise, kui ka aja arvestamise osas. Sellepärast tuligi *live* mängude loendi ja konkreetse mängu lehe kood kirjutada nii, et erinevate sportide puhul kuvatakse erinev tabloo (vt Joonis 9).

Joonis 9. Korvpalli (vasakul) ja tennise (paremal) tabloo

See tähendab ühtlasi, et navigeerides mängude loendist konkreetse mängu lehele tuleb navigate funktsioonis kaasa anda spordi nimi. Siis teab konkreetse mängu vaate komponent, missuguse spordi tablood on parajasti vaja näidata.

3.7. Testimine ja edasine arendus

React Native raamistiku *live reload* ja *hot reload* funktsionaalsused on väga suureks abiks arendajapoolsel testimisel. Iga koodis tehtav muudatus on koheselt näha töötavas iOS simulaatoris või Android emulaatoris. Autori arvates kiirendab see olulisel määral arendustööd ja aitab varem avastada vigu rakenduse koodis.

Peale lokaalse arendajapoolse testimise iOS simulaatoril ja Andorid emulaatoril kasutati mobiilsel seadmel rakenduse esialgse versiooni testimiseks HockeyApp[22] platvormi. See on platvorm, kuhu registreeritud kasutaja saab üles laadida rakenduse beta-versiooni ja selle sealt mobiilsesse seadmesse alla laadida. Käesoleva töö raames valminud Demo Bet rakenduse esimest versiooni testiti kahel erineva ekraani suurusega Android seadmel ja kahel iOS seadmel: iPhone 6 ja iPhone X. Rakenduse funktsionaalse poole pealt suuremaid probleeme ei täheldatud, kuid tõstatati mõned probleemid disaini poolelt. Android seadmetel *modal screen* vaate avamisel ja sulgemisel on rakenduse päises näha häirivat virvendust. Väiksema ekraaniga Android telefonil on jalusel oleval menüül *pre-match* vahekaardi nimi kuvatud kahel real, mis põhjusab ikoonide ebaühtlase joonduse.

Käesoleva bakalaureusetöö raames valminud rakendus ei ole veel spordiennustusportaali mobiilirakenduse täislahendus. Kokku valmis seitse vaadet, mis näitavad reaalajas informatsiooni kõikide panustamiseks saada olevate turgude, *pre-match* ja *live* mängude kohta.

Rakendust tutvustavad videod on vastavalt platvormile saadaval järgmiselt:

- https://drive.google.com/file/d/1qZoT80cQkVTI8-uux2HzAubrsKiOIP0V/view?usp=sharing aadressil Androidi rakendust tutvustav video;
- https://drive.google.com/file/d/1X8OmevpaEmTr7bO1XjGWecDbaGaQQtfO/view?usp=sharing aadressil iOS rakendust tutvustav video.

[22] https://hockeyapp.net/

Videotel on näha rakenduse erinevate vaadete vahel liikumine, täisfunktsionaalsusega pre-match ja live mängude vaated ning kohahoidjad (ingl *placeholders*) vaadetele, millele lisatakse funktsionaalsus rakenduse teises versioonis. Hoolimata sellest on võimalik juba tehtut kasutada ühe osana täislahendusest. Selleks tuleb olemasolev rakendus integreerida süsteemidega, mis pakuvad hetkel Demo Bet'i mobiilirakenduses puuduvaid funktsionaalsusi. Edasise arenduse lõplik eesmärk on praegusele rakendusele lisada puuduv funktsionaalsus, et tulevikus pakkuda klientidele täisväärtuslikku spordiennustusportaali mobiilirakenduse lahendust. Esmane eesmärk on parandada testimisel avastatud vead, muuta rakenduse koodistruktuur paremini hallatavaks ja testida olemasolevat funktsionaalsust potensiaalsete klientidega.

Kokkuvõte

Käesoleva bakalaureusetöö teema tulenes sellest, et mobiilse interneti kiire levik mõjutab otseselt spordiennustust vahendavate kihlveokontorite tegevust. 90% nutiseadmete kasutajatest eelistavad mobiilsete veebilehtede asemel rakendusi (Khalaf, 2016). Seega vajab iga veebis spordiennustusi vahendav kihlveokontor tihedas konkurentsis püsimiseks nutiseadmel töötavat lahendust. Suurtel kihlveokontoritel ei ole probleem eraldi mobiilirakenduse arendus erinevatele platvormidele, kuid väiksematele või alustavatele spodiennutuse vahendajatele võib see kulukuse tõttu üle jõu käia.

Käesoleva töö autor on pikalt olnud seotud spordiennustustööstuses tegutseva ettevõttega. Sealt tuligi idee selle probleemile lahenduse leidmiseks – arendada spordiennustusportaali mobiilirakendus, mis toimib nii iOS kui ka Android platvormil. Kuna spordiennustusportaali mobiilirakenduse täislahendus on keerukas süsteem, siis käesoleva bakalaureusetöö eesmärk oli valmis teha rakenduse esimene osa, mida saab vastavalt kliendi soovidele kohandada ja edasi arendada. See osa hõlmab endas iOS ja Android operatsioonil töötavat rakendust, kus on näha reaalajas ennustuseks saada olevad spordisündmused, turud, koefitsendid ning spordisündmuste ja turgude staatused.

Eesmärgi saavutamiseks andis töö autor esmalt ülevaate seminaritöö käigus välja selgitatud rakenduse funktsionaalsetest nõuetest ja disainist. Täiendavalt lisati ka mõningad eelnevalt tähelepanuta jäänud detailid, nii funktsionaalsuse kui ka disaini osas. Seejärel anti ülevaade mobiilirakenduse arenduseks kasutatud React Native raamistikust ja toodi välja põhjused, miks just see osutus valituks. Lisaks analüüsiti juba olemasolevaid sama raamistikuga arendatud rakendusi ja toodi välja millest õppust võtta ning mida vältida.

Lõpetuseks kirjeldati põhjalikult rakenduse arenduse etappe. Alustades ülevaatega arendusprotsessist ja arenduskeskonna ülesseadmisest ning lõpetades üksikasjaliku kirjeldusega rakenduse erinevate osade arenduses. Lisaks toodi välja mõned arenduse käigus ilmnenud probleemid, leitud lahendused ja plaanid rakenduse edasiseks arenduseks. Käesoleva töö raames valmis spordiennustusportaali mobiilirakenduse Demo Bet esimene versioon nii iOS kui ka Android platvormile.

Bakalaureusetöö käigus omandas autor React Native raamistikus mobiilirakenduse arendamise algteadmised, mis on vajalikud antud rakenduse edasiseks arenduseks.

Kasutatud kirjandus

Abernathy, C. (2018). *React Native Tutorial: Building Android Apps with JavaScript.* Loetud aadressil https://www.raywenderlich.com/178012/react-native-tutorial-building-android-apps-javascript

AltexSoft. (2018, 19. veebruar). *Xamarin vs React Native vs Ionic: Cross-platform Mobile Frameworks Comparison* [ajaveebipostitus]. Loetud aadressil https://www.altexsoft.com/blog/engineering/xamarin-vs-react-native-vs-ionic-cross-platform-mobile-frameworks-comparison/

Bresnan, M., Safi, M.K., Patel, S., & Keerti. (2016, 19. detsember). *React Native at WalmartLabs* [ajaveebipostitus]. Loetud aadressil https://medium.com/walmartlabs/react-native-at-walmartlabs-cdd140589560

Carli, S. (2017, 19. aprill). *How to use the FlatList Component—React Native Basics* [ajaveebipostitus]. Loetud aadressil https://medium.com/react-native-development/how-to-use-the-flatlist-component-react-native-basics-92c482816fe6

Instagram Engineering. (2017, 7. veebruar). *React Native at Instagram* [ajaveebipostitus]. Loetud aadressil https://instagram-engineering.com/react-native-at-instagram-dd828a9a90c7

Kazula, B., & Grajcar, J. (2018, 24. jaanuar). *Why Use React Native for Your Mobile App?* [ajaveebipostitus]. Loetud aadressil https://stxnext.com/blog/2018/01/24/why-use-react-native-your-mobile-app/

Khalaf, S. (2016). *Seven Years Into The Mobile Revolution: Content is King... Again.* Loetud aadressil http://flurrymobile.tumblr.com/post/127638842745/seven-years-into-the-mobile-revolution-content-is

Napier, P. (2017, 6. veebruar). *Code: React Native: Setting Up the Environment* [ajaveebipostitus]. Loetud aadressil https://medium.com/@MadApper/react-native-setting-up-the-environment-5c310ef814d5

Novick, V. (2017). *React Native - Building Mobile Apps with JavaScript.* Birmingham: Packt Publishing Ltd.

Pushtechnology. (2015, 10. september). *Mobile on the Rise for Sports Betting Industry* [ajaveebipostitus]. Loetud aadressil https://www.pushtechnology.com/blog/mobile-rise-sports-betting-industry/

React Native. (2018). Loetud aadressil https://www.jetbrains.com/help/webstorm/react-native.html

React Native. (kuupäev puudub). Loetud aadressil https://facebook.github.io/react-native/

Shkut, K. (kuupäev puudub). *Cross-Platform Navigation in React Native.* Loetud aadressil https://rationalappdev.com/cross-platform-navigation-in-react-native/

Sillaste, M. (2017). *Spordiennustusportaali mobiilirakenduse nõuete analüüs ja disain* (seminaritöö). Loetud aadressil http://www.cs.tlu.ee/teemaderegister/

Sriraman. (2016, 11. november). *What we learned after using React Native for a year* [ajaveebipostitus]. Loetud aadressil https://hashnode.com/post/what-we-learned-after-using-react-native-for-a-year-civdr8zv6058l3853wqud7hqp

Statista. (2017). *Mobile internet traffic as percentage of total web traffic in August 2017, by region*. Loetud aadressil https://www.statista.com/statistics/306528/share-of-mobile-internet-traffic-in-global-regions/

Who's using React Native? (kuupäev puudub). Loetud aadressil http://facebook.github.io/react-native/showcase.html

Witte, D., & von Weitershausen, P. (2015). *React Native for Android: How we built the first cross-platform React Native app.* Loetud aadressil https://code.facebook.com/posts/1189117404435352/react-native-for-android-how-we-built-the-first-cross-platform-react-native-app/

Summary

Sportsbook Mobile Application Development with React Native Framework

Bachelor's Thesis

People have started to use more and more mobile phones or tablets to use different Internet services. This is also affecting sports betting. As the timing of placing a bet makes big difference for punters then also the number of bets made from mobile devices has increased. It means that to remain competitive in the betting industry every online bookmaker needs decent mobile service. As people like to use mobile applications more than mobile websites then probably it would be useful to have a application. It is not a problem for the big bookmakers to have a mobile application development teams, but as the mobile development could be quite expensive then smaller or starting bookmakers can't afford it.

Author of this Bachelor's thesis have worked in betting industry for a while and that's where he got the idea for this problem – develop the sportsbook mobile application using cross-platform React Native framework. As the full solution of the sportsbook mobile application is really complicated system then the goal of this Bachelor's thesis is to complete the first part of the application. This includes creating a sportsbook mobile application for iOS and Android platforms, which displays available pre match and live betting content: fixtures, markets, prices and fixture statuses.

The goal of this Bachelor's thesis was achieved. First version of the sportsbook mobile application for iOS and Android was created and the full betting content including both pre match and live fixtures and markets are available in the created application.

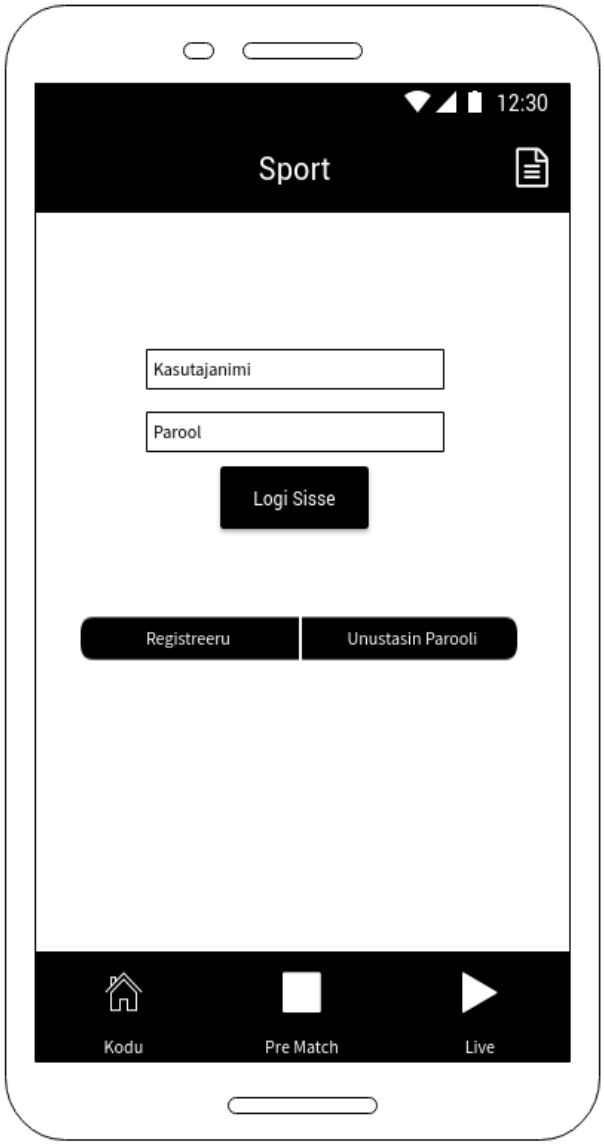
The first chapter of this thesis describes functional requirements and application design. The next chapter introduces React Native framework and gives overview of some already created React Native mobile application. The last chapter describes thoroughly the development process, setting up the development environment and development work. This chapter also briefly describes testing and the future development plans of the sportsbook mobile application.

LISAD

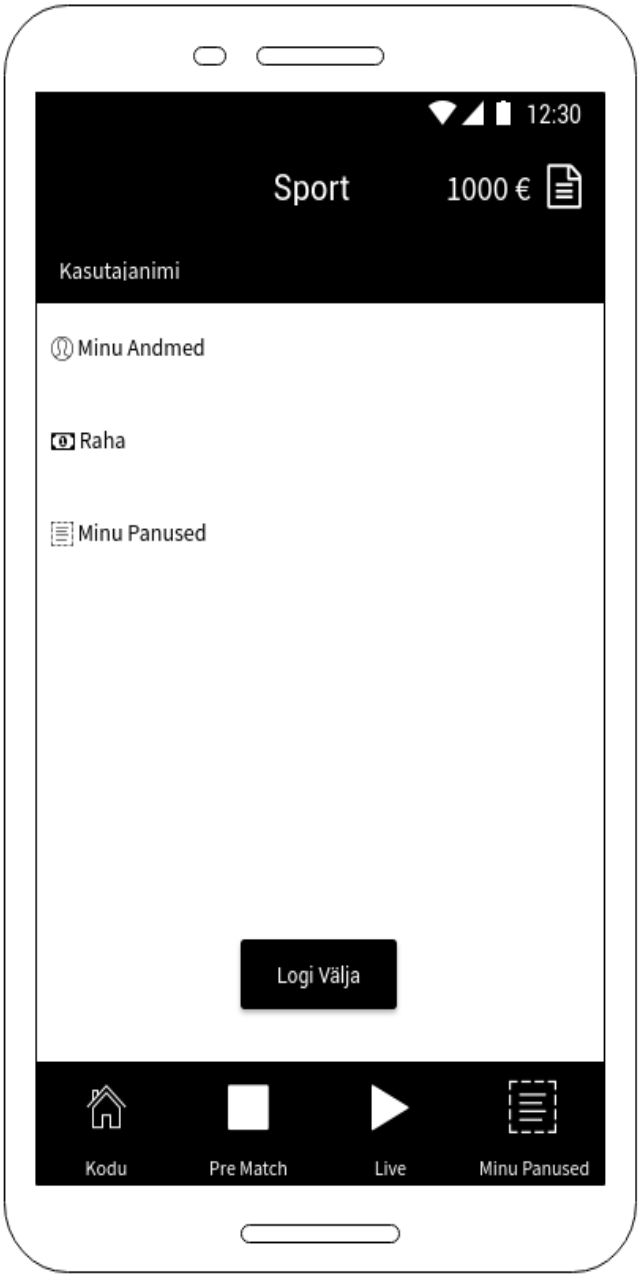
Lisa 1 Kasutaja isikuandmete vaadete disaini prototüübid

Käesolevas lisas välja toodud prototüübid on autori seminaritööst “Spordiennustusportaali mobiilirakenduse nõuete analüüs ja disain” (Sillaste, 2017).

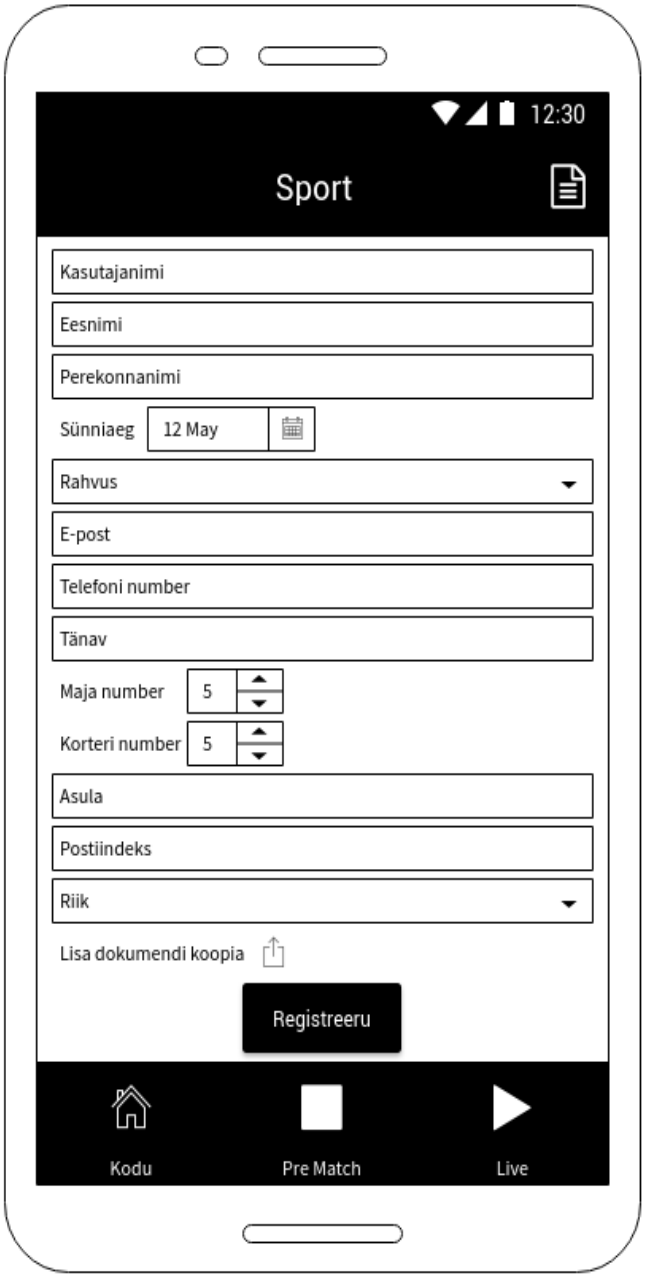
Mitte sisse loginud kasutaja vaate disaini prototüüp



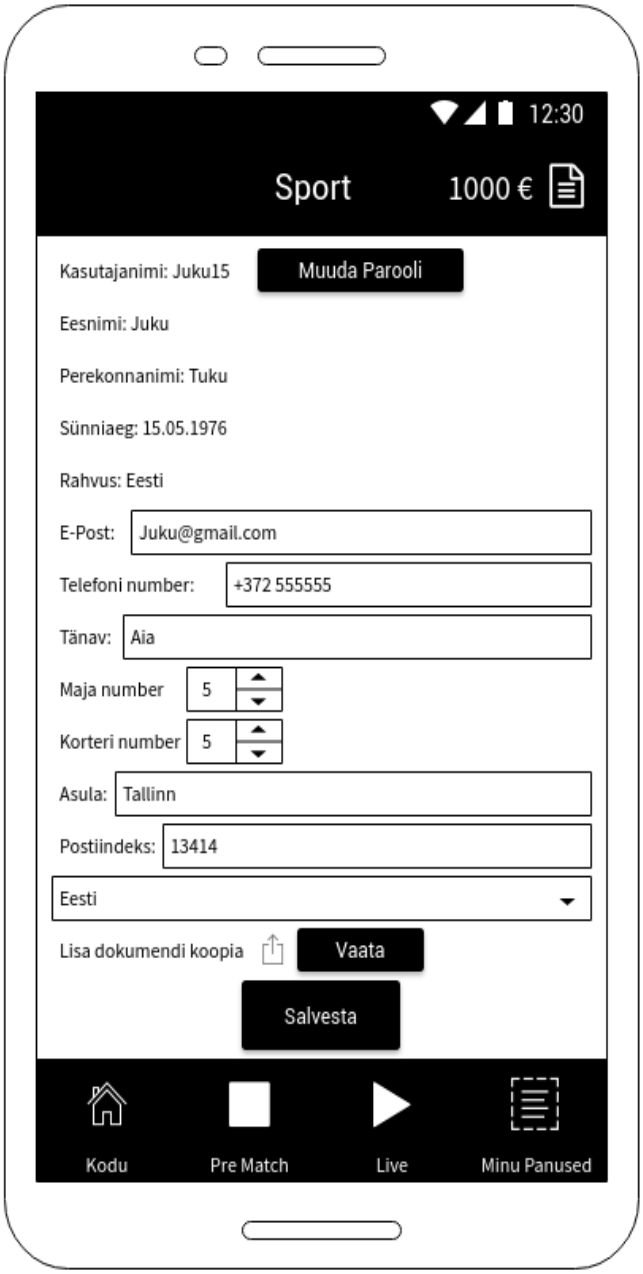
Sisse loginud kasutaja vaate disaini prototüüp



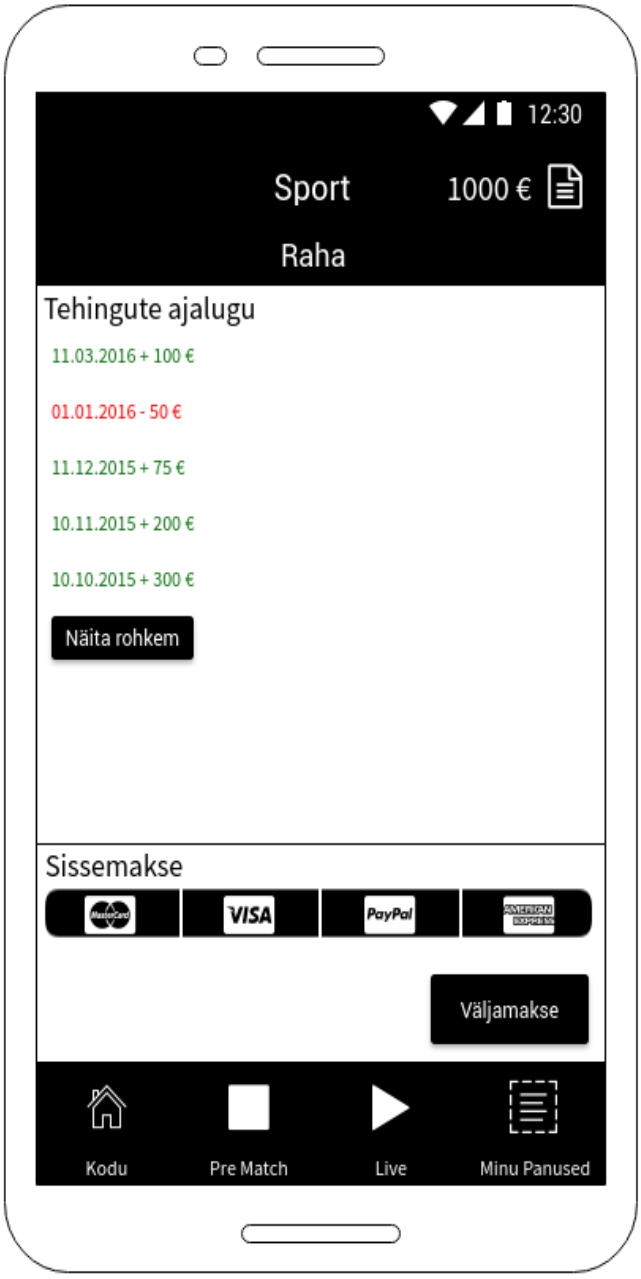
Kasutaja registreerimise vaate disaini prototüüp



Kasutaja isikuandmete vaate disaini prototüüp



Kasutaja rahakonto vaate disaini prototüüp

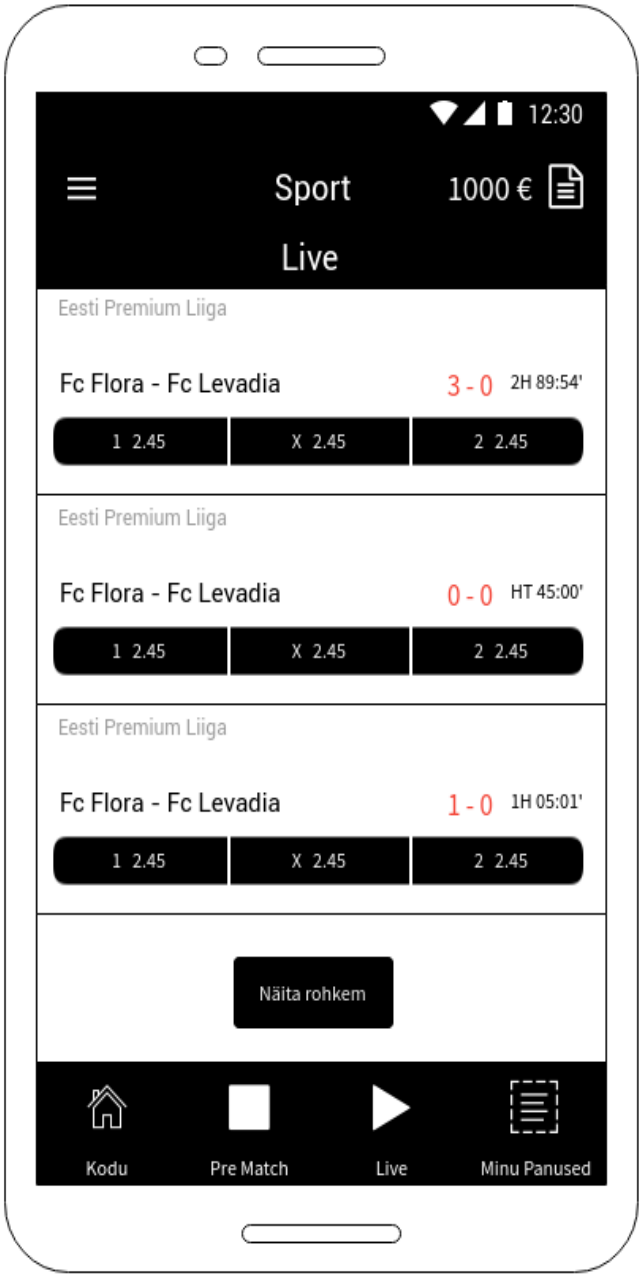


Lisa 2 Panustamisega seotud vaadete disaini prototüübid

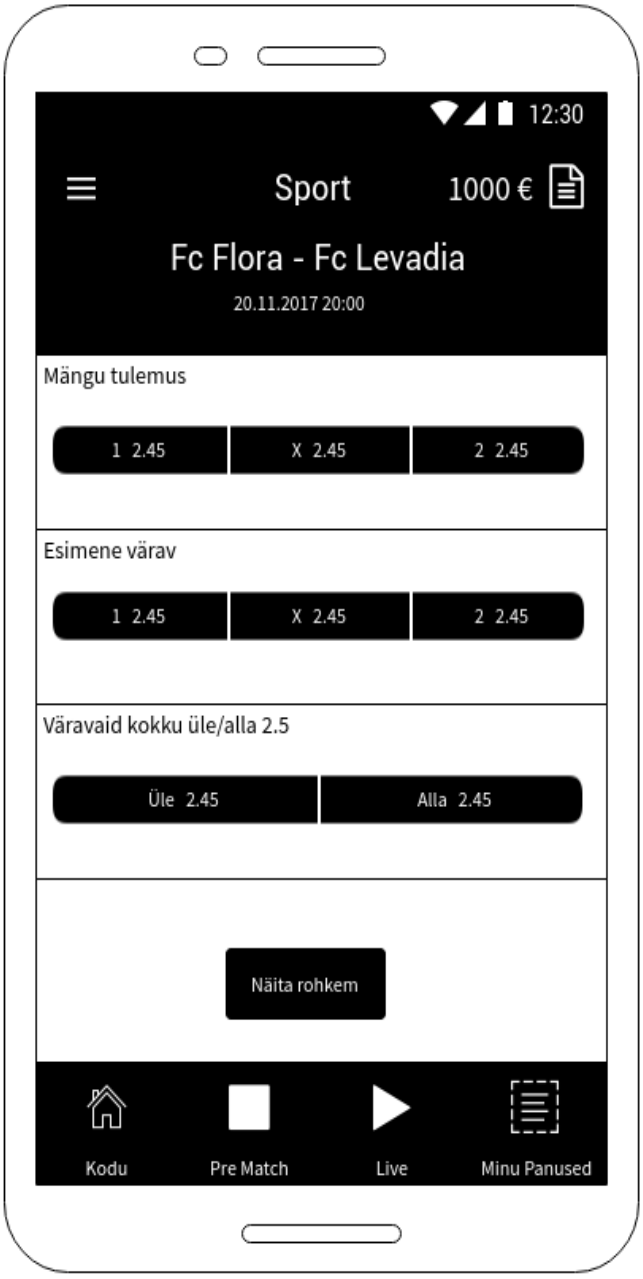
Käesolevas lisas välja toodud prototüübid on autori seminaritööst “Spordiennustusportaali mobiilirakenduse nõuete analüüs ja disain” (Sillaste, 2017).

Pre-match mängude vaate disaini prototüüp

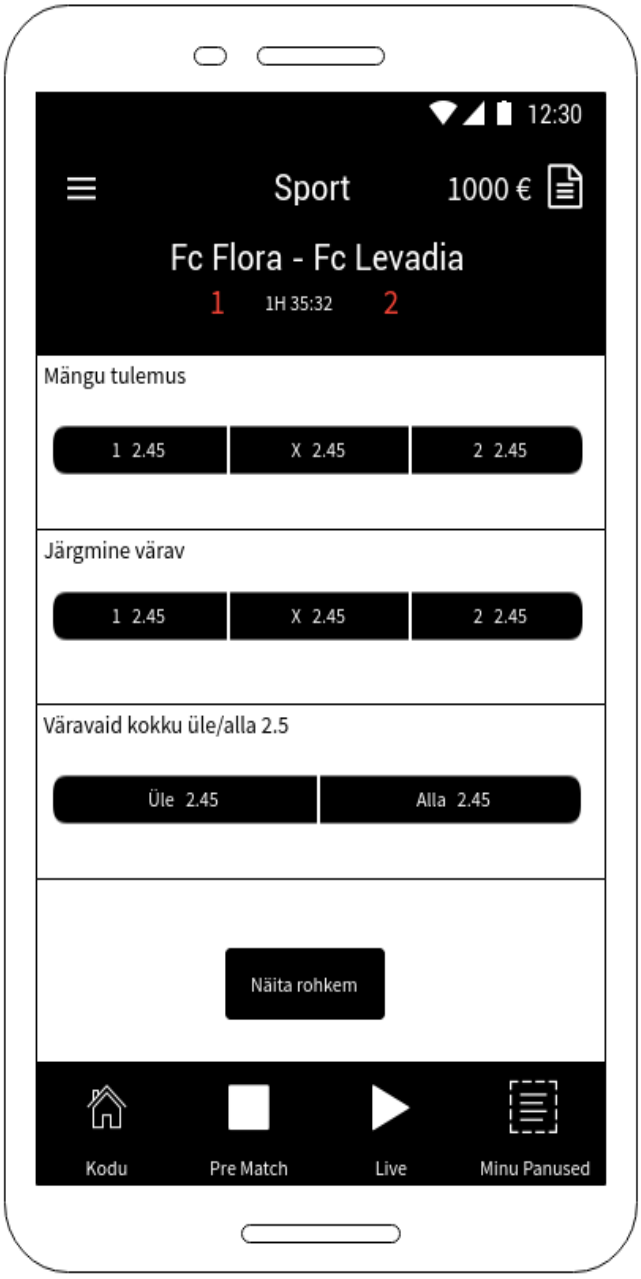
Live mängude vaate disaini prototüüp



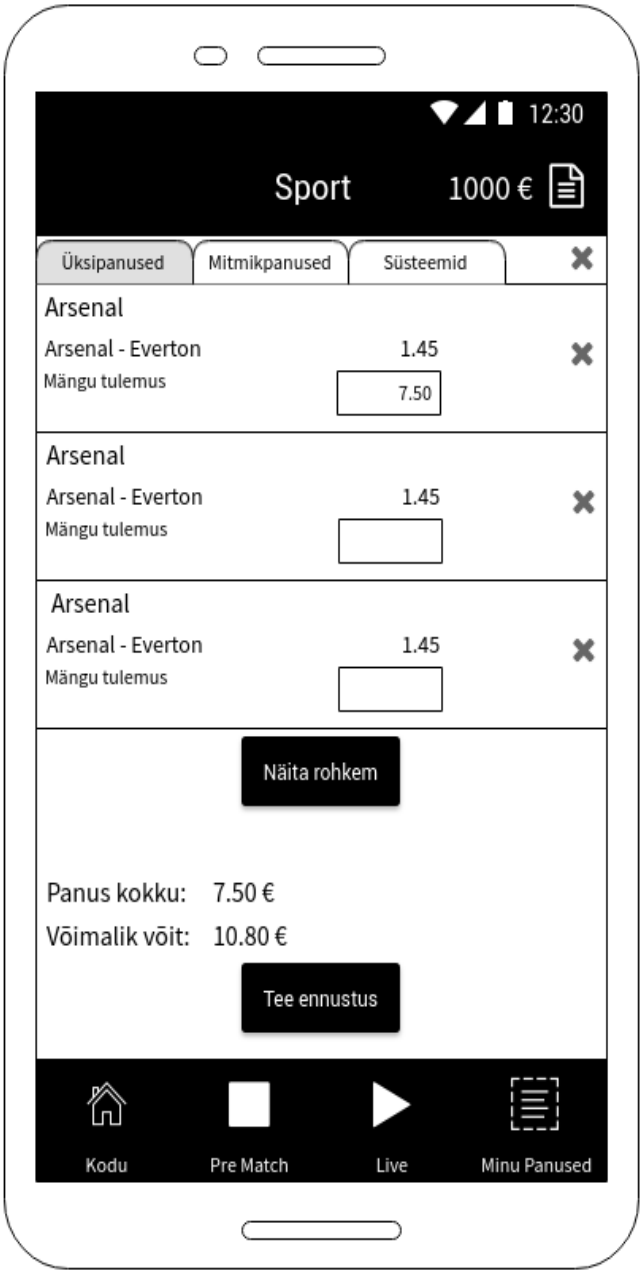
Pre-match konkreetse mängu vaate disaini prototüüp



Live konkreetse mängu vaate disaini prototüüp



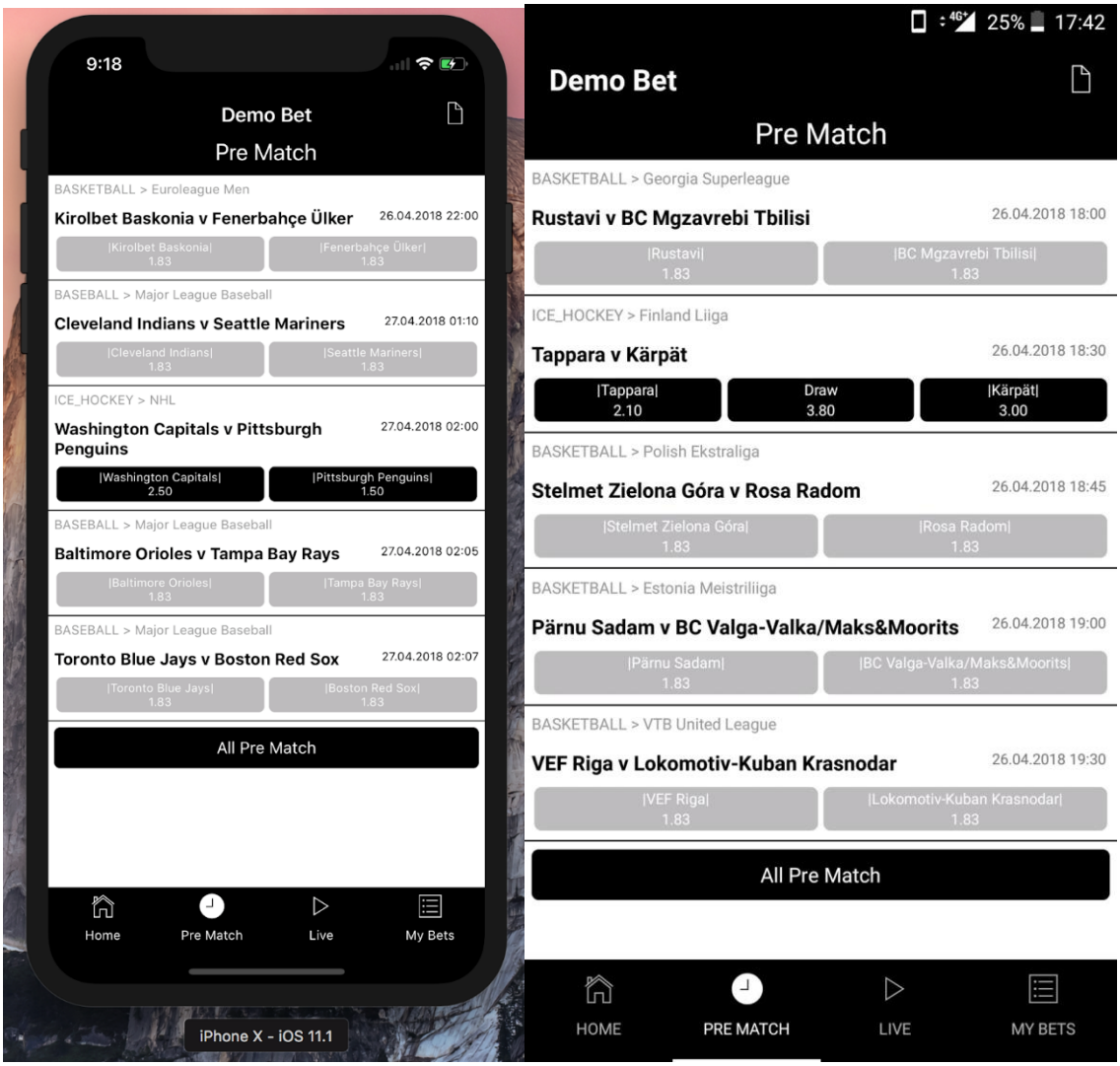
Panustussedeli vaate disaini prototüüp



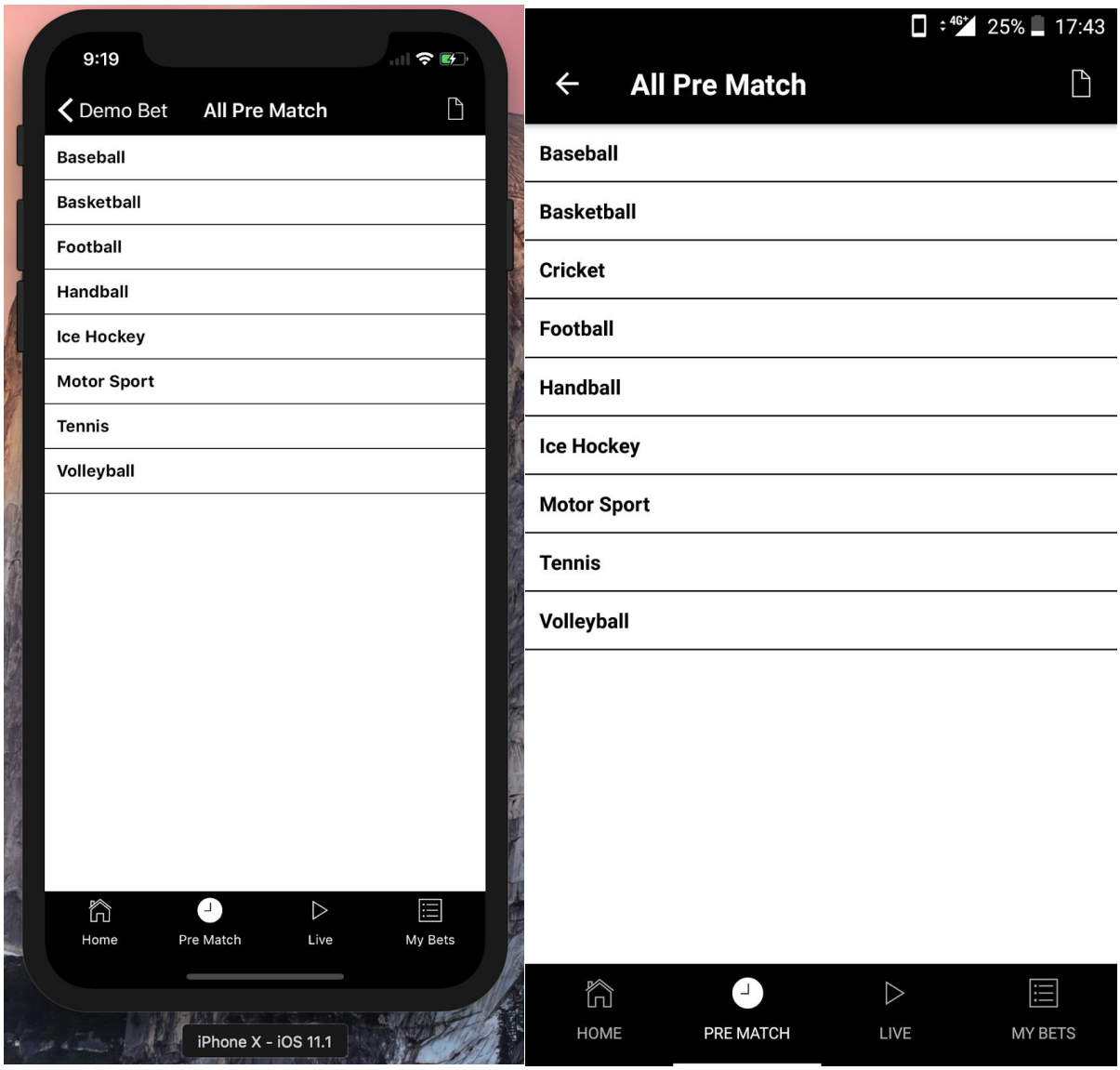
Minu panused vaate disaini prototüüp

Lisa 3 *Pre-Match* mängude vaated rakenduses

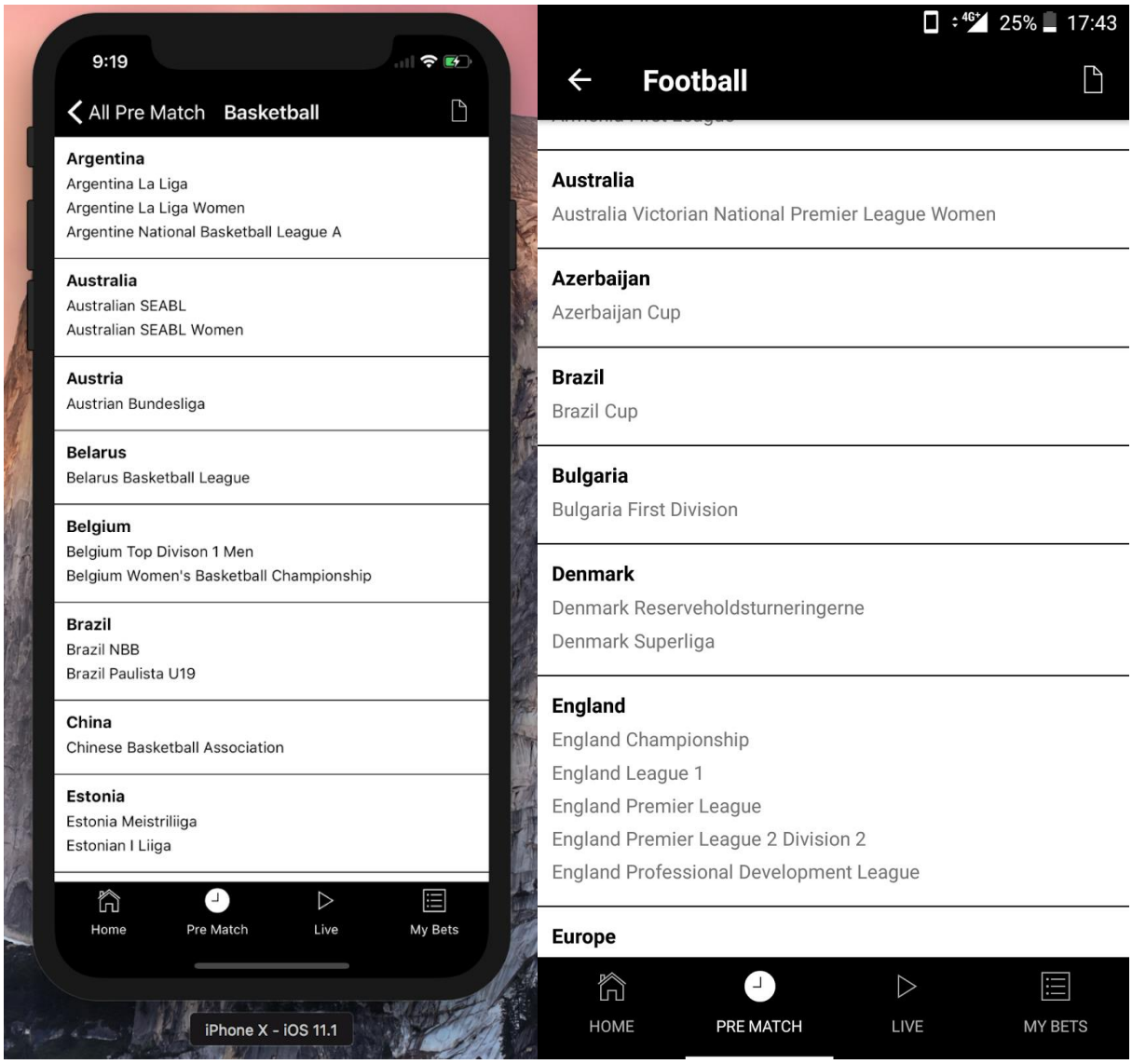
Pre-match mängude loend iOS (vasakul) ja Android (paremal)



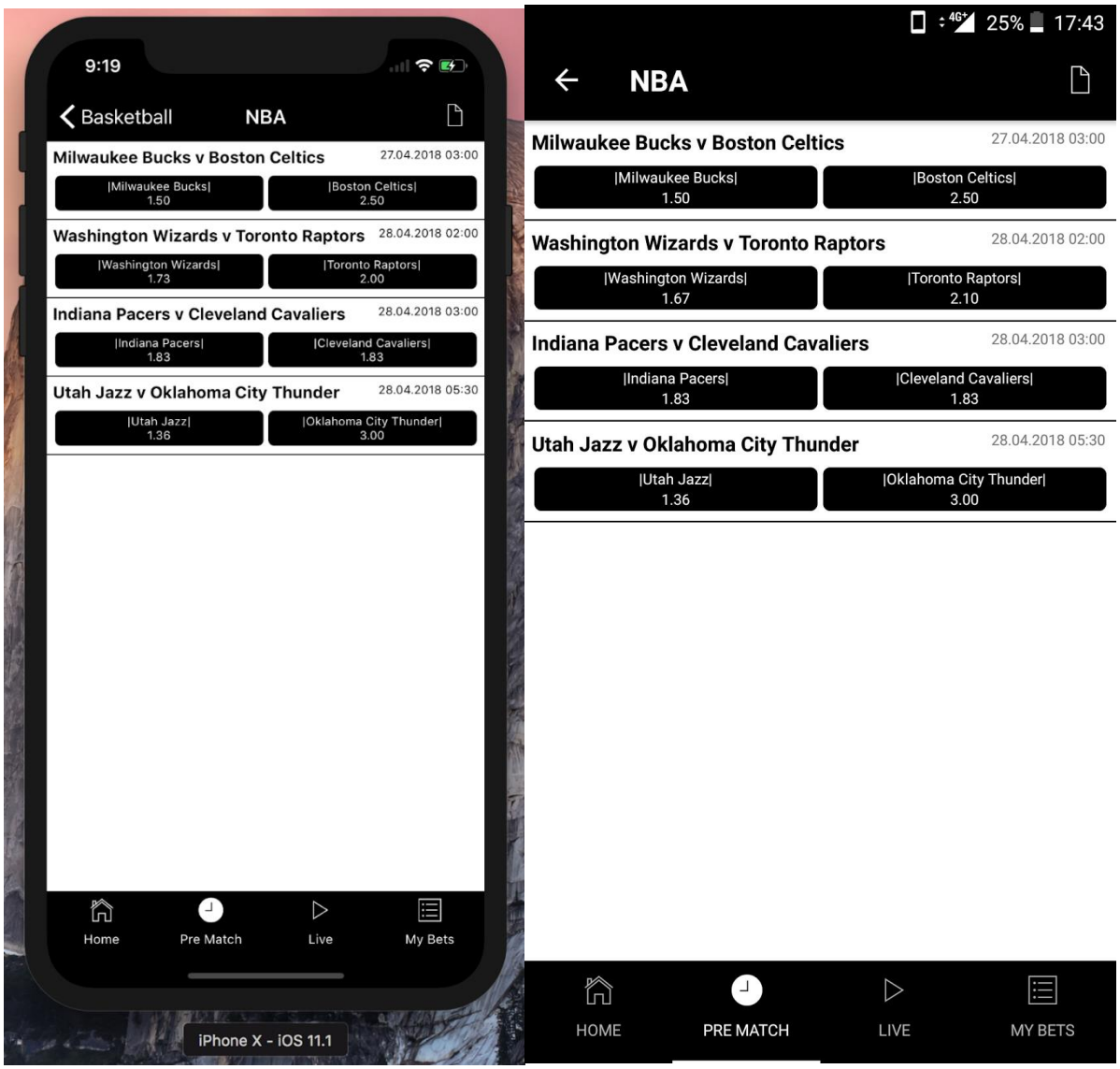
Pre-match sportide loend iOS (vasakul) ja Android (paremal)



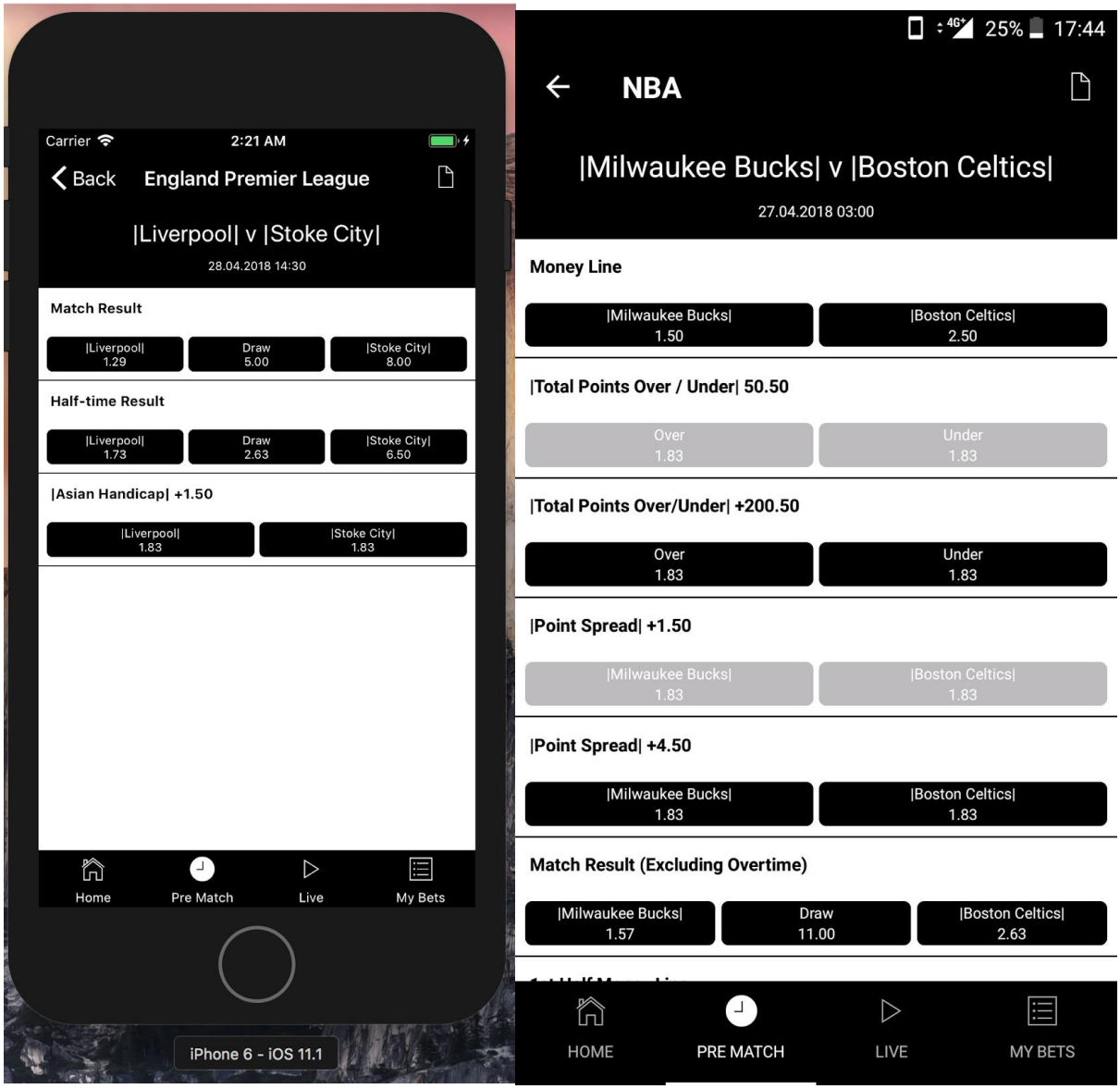
Pre-match võistluste loend iOS (vasakul) ja Android (paremal)



Pre-match valitud võistluste mängude loend iOS (vasakul) ja Android (paremal)

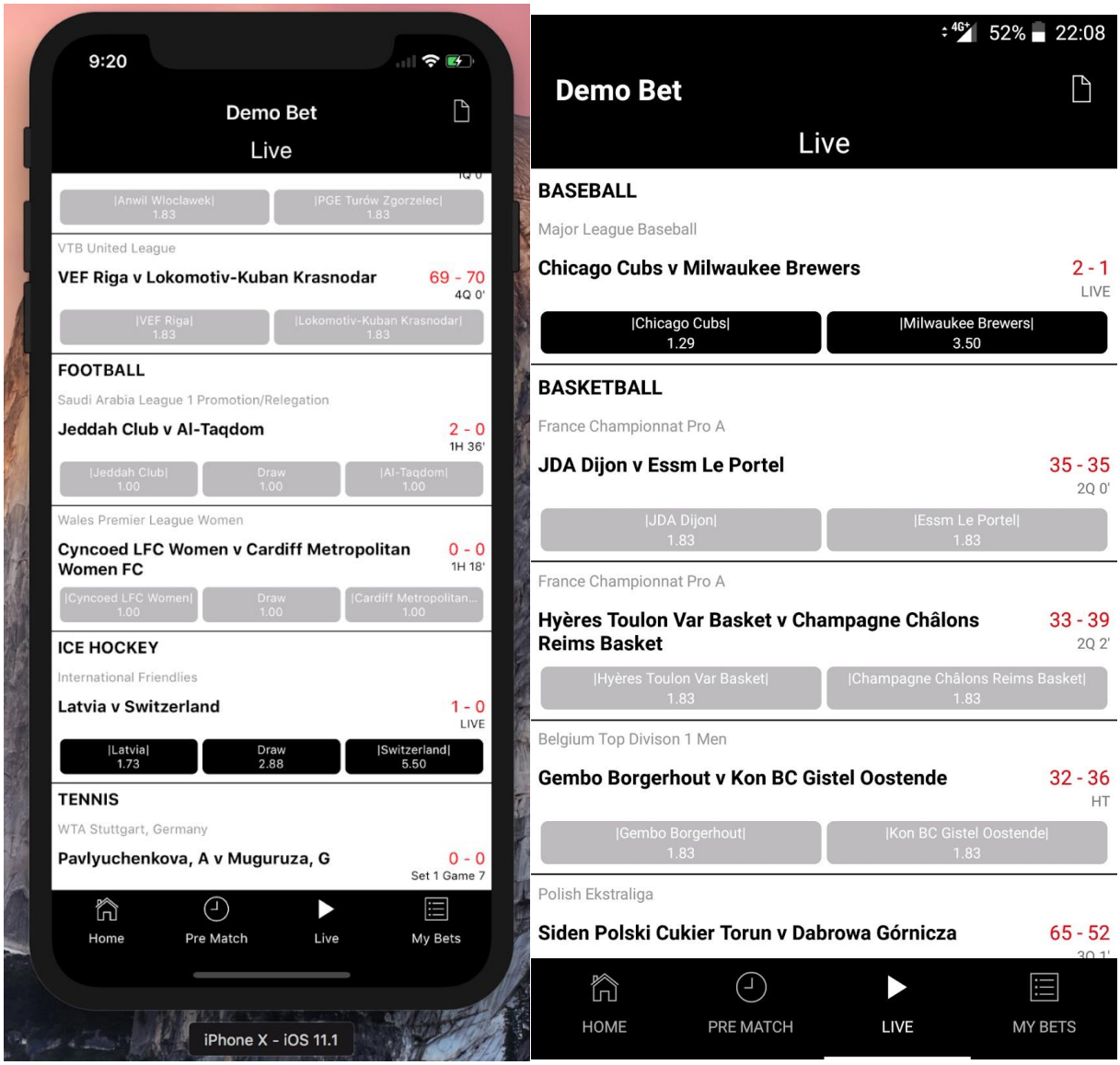


Pre-match mängu leht iOS (vasakul) ja Android (paremal)

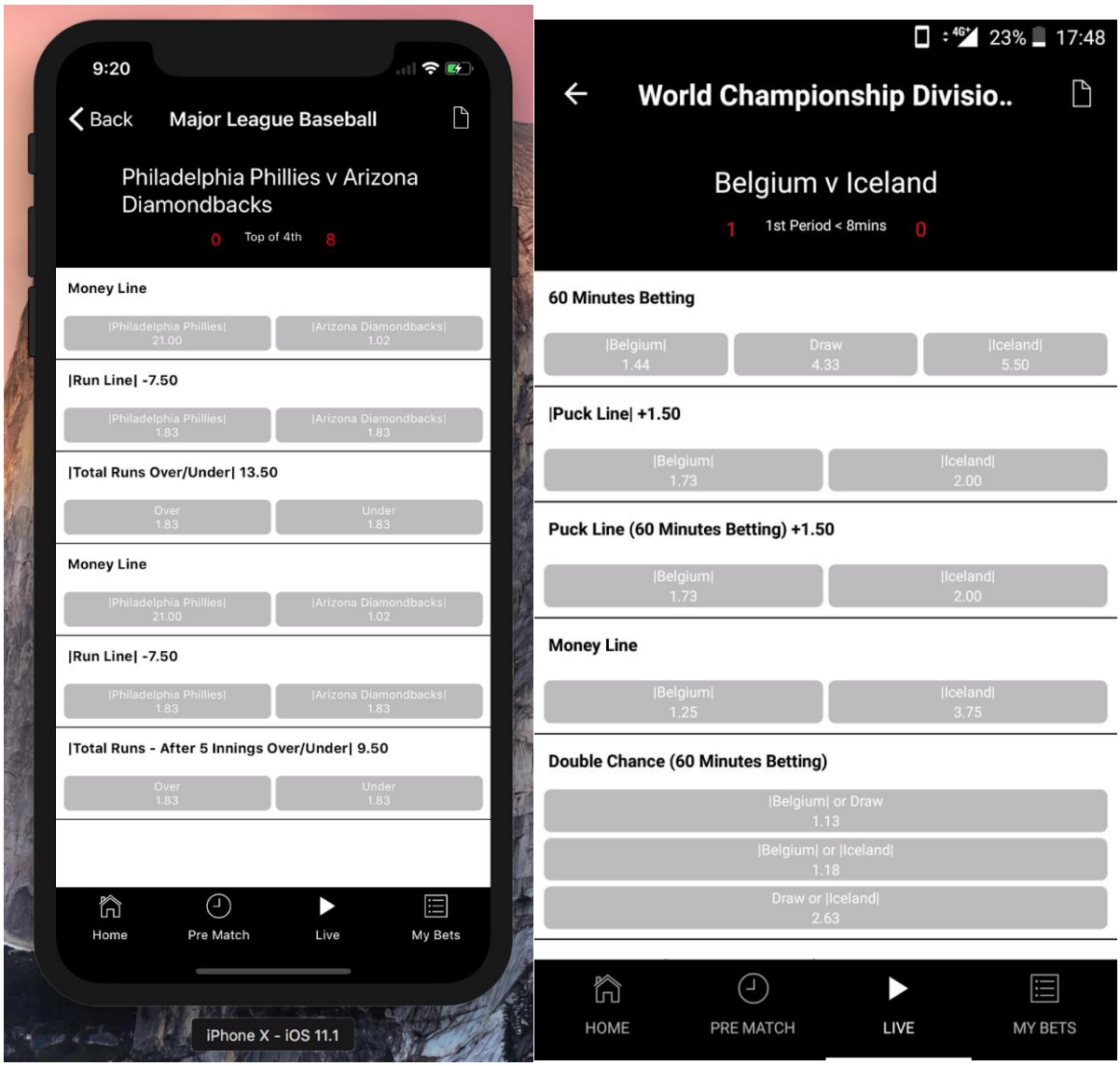


Lisa 4 *Live* mängude vaated rakenduses

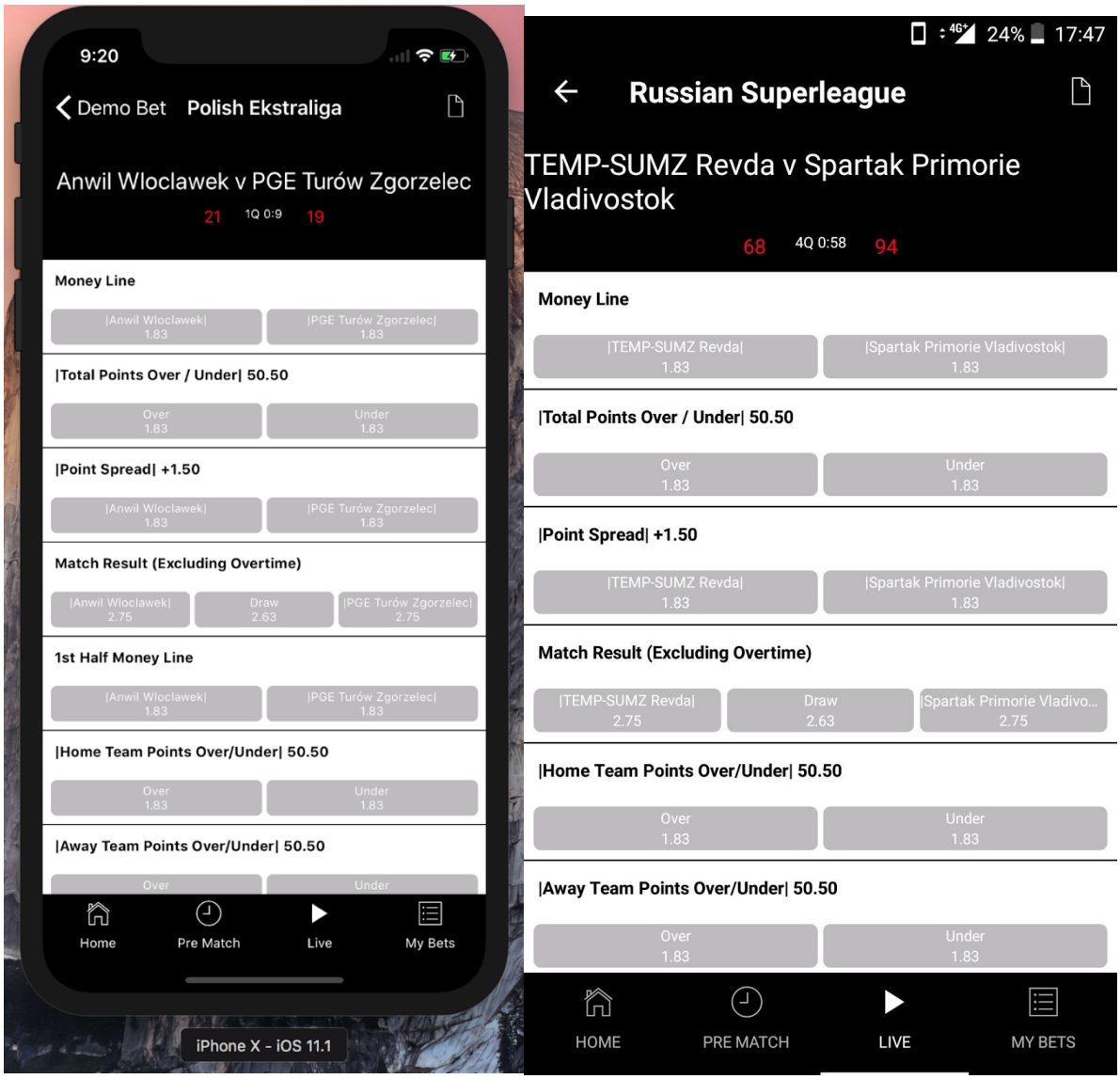
Live mängude loend iOS (vasakul) ja Android (paremal)



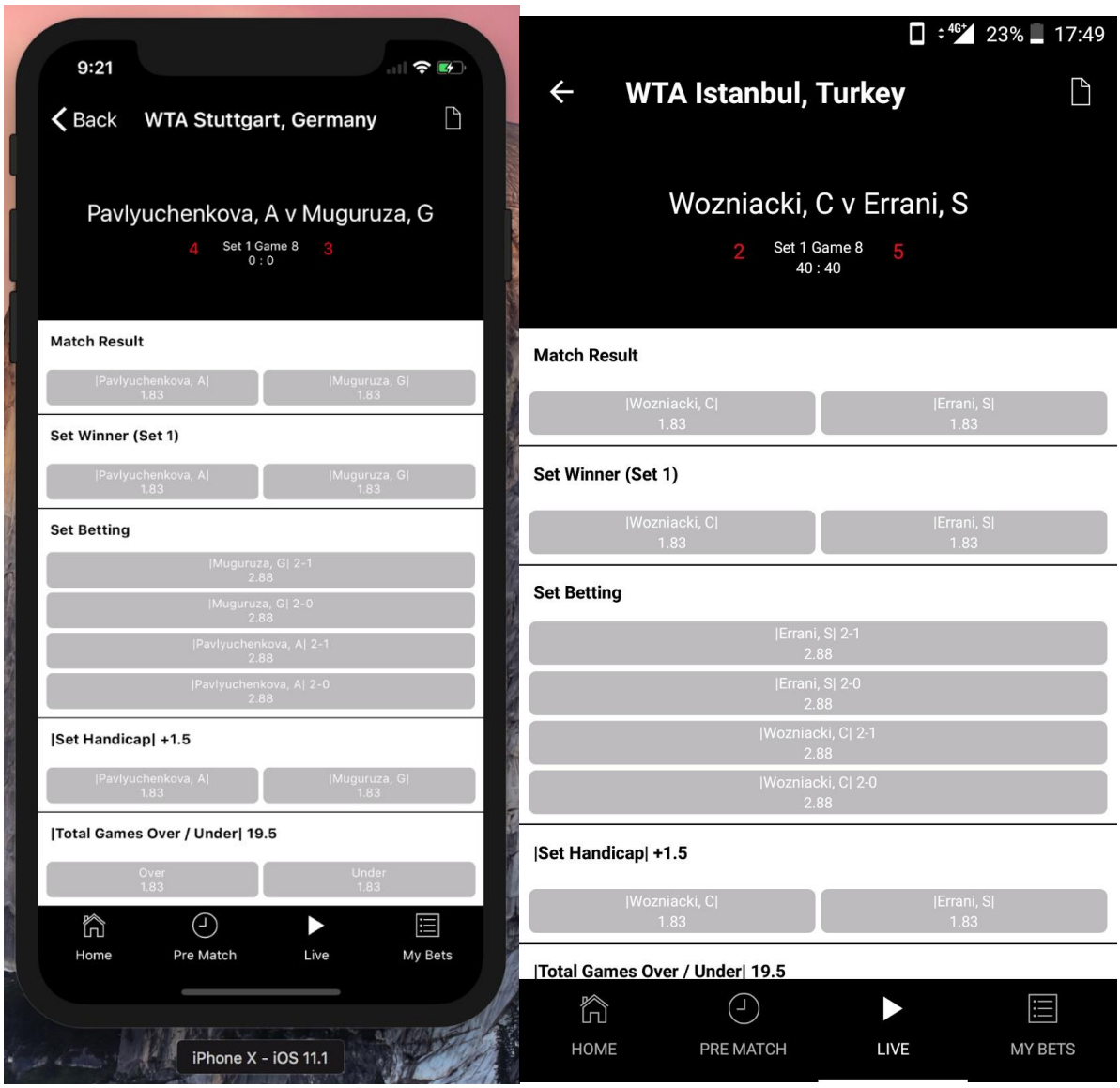
Live vaikimisi mängu vaade iOS (vasakul) ja Android (paremal)



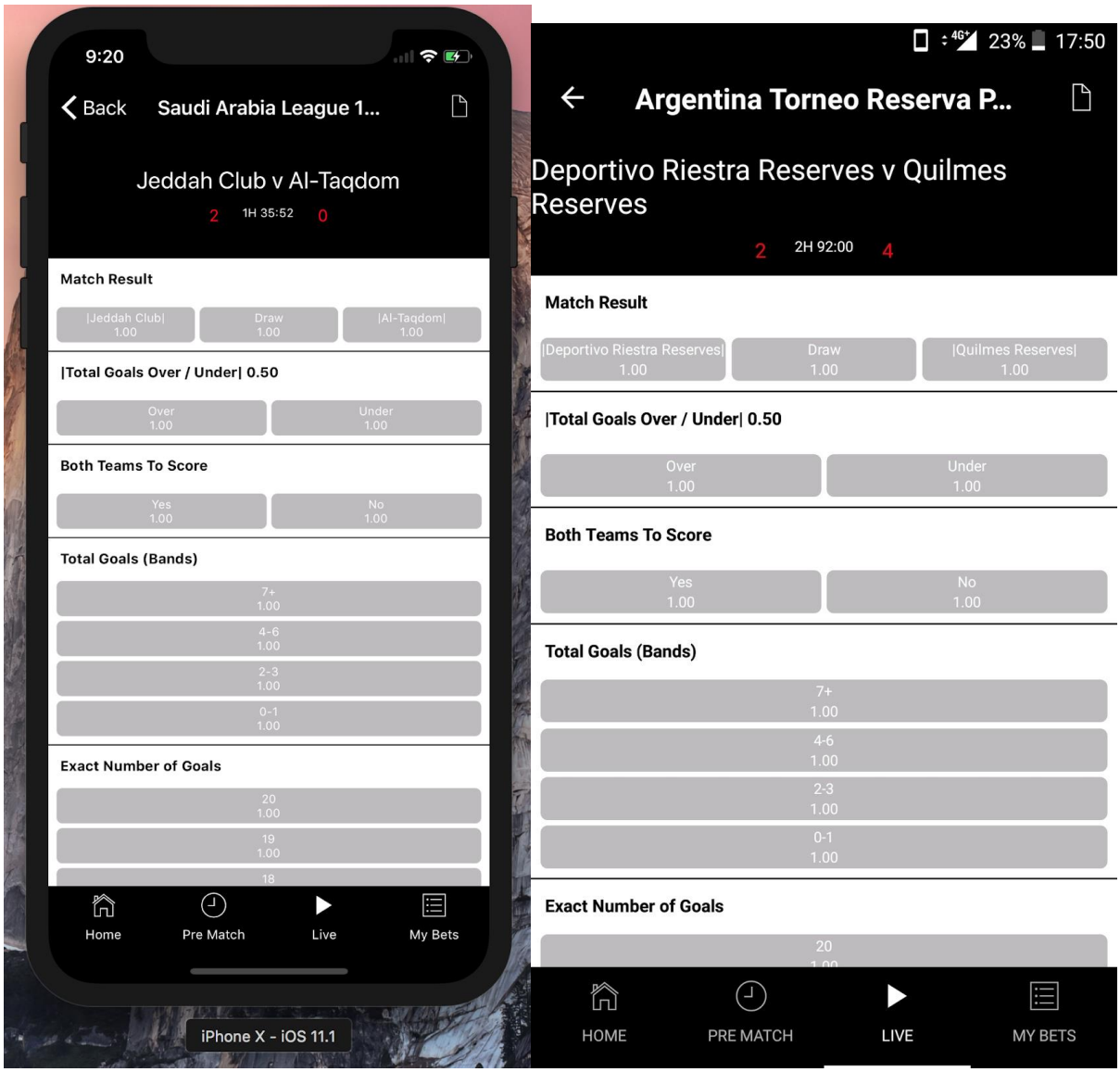
Live korvpalli mängu vaade iOS (vasakul) ja Android (paremal)



Live tennise mängu vaade iOS (vasakul) ja Android (paremal)



Live jalgpalli mängu vaade iOS (vasakul) ja Android (paremal)



Lisa 5 Teised rakenduse vaated

Sisse loginud kasutaja avalehe vaade iOS (vasakul) ja Android (paremal)