TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Martin Lillemets IAPB134915

NÕMME KABEKLUBI TURNIIRI HALDAMISE RAKENDUS

Bakalaureusetöö

Juhendaja: Meelis Antoi
MSc

Tallinn 2019

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: Martin Lillemets

19.05.2019

Annotatsioon

Lõputöö eesmärk on luua veebirakendus, mis pakub lihtsalt lahendust kabeturniiride loomiseks ja läbiviimiseks. Rakenduses on kasutatud programmeerimiskeeli PHP ja TypeScript.

Töö esimeses pooles annab autor ülevaate olemasolevatest rakendustest ning analüüsib loodava rakenduse nõudeid. Analüüsitakse kahte erinevat turniirisüsteemi ja võimalikke tehnoloogiaid veebirakenduse loomiseks. Seejärel kirjeldatakse rakenduse arenduskäiku ja tulevikuplaane ning rakenduse testimist.

Töö tulemuseks on veebirakendus, millega saab koostada individuaal- ja meeskonnaturniire. Peale selle pakub rakendus lihtsalt ülevaadet käimasolevatest turniiridest ja tulemustest.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 31 leheküljel, 6 peatükki, 16 joonist, 3 tabelit.

Abstract

Nõmme Checkers club tournament management application

The goal of the thesis is to create a web application that provides a simple solution for creating and running checkers tournaments. The application uses PHP and TypeScript programming languages.

In the first half of this thesis, the author gives an overview of existing application and analyzes the requirements of the application. Then the author describes two different tournament systems and possible technologies for creating the application. The author explains the development and testing process and further plans for development.

The result of this thesis is a web application which allows simple creation and management of future checkers tournaments. In addition, the application gives real time overview of ongoing tournaments.

The thesis is in Estonian and contains 31 pages of text, 6 chapters, 16 figures, 3 tables.

Lühendite ja mõistete sõnastik

AJAX	*Asyncronous JavaScript and XML*
Angular	TypeScripti raamistik
API	Application Program Interface
Cron	Ajapõhine tööde planeerija Linux'i operatsioonisüsteemides
CSS	*Cascading Style Sheets*
Git	Avatud lähtekoodiga versioonihaldustarkvara
HTTP	*Hypertext Transfer Protocol*
JavaScript	Dünaamiline prototüübipõhine keel, mida kasutatakse peamiselt veebilehtedes skriptimiskeelena.[1]
JSON	*JavaScript Object Notation*
JWT	JSON Web Token
Linux	Avatud lähtekoodiga operatsioonisüsteemide perekond mis põhinevad Linux'i tuumal
Linux (tuum)	Avatud lähtekoodiga Unix-i laadne operatsioonisüsteemi tuum
MPA	*Multi page application*
MVP	*Minimum Viable Product*
ORM	*Object-relational mapping*
PHP	*Hypertext Preprocessor* on skriptimiskeel, mida kasutatakse peamiselt serveripoolsetes lahendustes dünaamiliste veebilehtede loomisel
REST	*Representational State Transfer*. Tarkvara arhitektuuristiil
SPA	*Single page application*
SQL	*Structures Query Language.* 1970-ndal IBM poolt loodud andmete pärimise keel[2]
TypeScript	Avatud lähtekoodiga programmeerimiskeel mis põhineb JavaScript-il.
Yii	PHP raamistik

Sisukord

1 Sissejuhatus....10
2 Analüüs....12
2.1 Kabeklubi poolt kasutusel olevad meetmed turniiride läbiviimiseks....12
2.2 Loodava rakenduse nõuded....13
2.2.1 Rakenduse funktsionaalsed nõuded....13
2.2.2 Rakenduse mittefunktsionaalsed nõuded....14
2.3 Šveitsi süsteem....15
2.3.1 Põhikriteeriumid....15
2.3.2 Mängija vabaks jätmine....16
2.3.3 Mängijate paarimine....16
2.4 Ringsüsteem....18
2.4.1 Mängijate paarimine....18
2.4.2 Värvide määramine....19
2.5 MPA või SPA disainimuster....20
3 Rakenduse arendus....22
3.1 Arenduskäik....22
3.1.1 Kasutaja registreerimise ja sisselogimise vaated....23
3.1.2 Mängijate haldus....24
3.1.3 Meeskondade haldus....25
3.1.4 Turniiride haldus....26
3.1.5 Turniiride läbiviimine....28
3.2 Turvalisus....30
3.3 Andmete liikumine serveri ja kliendi vahel....30
3.4 Andmebaas....31
3.5 Kasutatud tehnoloogiad....32
3.5.1 PHP....33
3.5.2 TypeScript....33
3.6 Kasutatud tööriistad....33

3.6.1 PhpStorm....33
3.6.2 Git....34
3.6.3 Adminer....34
3.6.4 Gii....34
3.6.5 Postman....34
4 Veebirakenduse testimine....35
4.1 Testimise meetodid....35
4.2 Testimise tulemused....37
5 Tulevikuarendused....38
5.1 Suhtlus väliste keskkondadega....38
5.2 Mitmekeelsus....38
5.3 Tulemuste jagamine....39
5.4 Kasutajatebaasi laiendamine....39
5.5 Mobiilirakendus....39
5.6 Turniiride analüüs....39
6 Kokkuvõte....40
Kasutatud kirjandus....41
Lisa 1 – Github'i link....43
Lisa 2 – Ringsüsteemi paarimise kood....44
Lisa 3 – Turniiri lisamise ja muutmise vorm....47

Jooniste loetelu

Joonis 1: Esimese vooru mängijate paarimine.....16
Joonis 2: Mängijate paarimine peale ümbertõstmist.....17
Joonis 3: Ringsüsteemi I vooru mängijate paarimine.....19
Joonis 4: Kohtuniku registreerimise vaade.....23
Joonis 5: Mängija muutmisvaade.....24
Joonis 6: Mängijate nimekirja vaade.....25
Joonis 7: Meeskonnaliikmete haldamise vaade.....26
Joonis 8: Turniiri muutmise vaade.....27
Joonis 9: Turniiri osalejate sidumise vaade.....27
Joonis 10: Turniiri detailvaade.....28
Joonis 11: Vooru tulemuste sisestamise vaade.....29
Joonis 12: Turniiri tulemuste vaade.....29
Joonis 13: JWT võtme genereerimine [17] 30
Joonis 14: Andmebaasi skeem.....32
Joonis 15: Mängija lisamise vaate valideerimisvead.....36
Joonis 16: Vigase autentimisvõtmega meeskonna nimekirja päring.....37

Tabelite loetelu

Tabel 1: Ringsüsteemi voorude paarid.....19
Tabel 2: MPA eelised ja puudused.....20
Tabel 3: SPA eelised ja puudused.....21

1 Sissejuhatus

Antud bakalaureusetöö eesmärgiks on arendada rakendus, mis võimaldab korraldada ja läbi viia kabeturniire kasutades erinevaid turniirisüsteemi formaate. Rakendus arendatakse Nõmme kabeklubi jaoks ning selle ülesanne on lihtsustada kabeturniiride koostamist ja läbiviimist. See peab olema kasutaja jaoks lihtne ja mugav kasutada. Kasutajateks hakkavad olema kabekohtunikud, kes koostavad, haldavad ja viivad läbi turniire. Rakendus on arendatud veebiplatvormile, et tagada võimalikult lihtne ja mugav ligipääs kõigile kasutajatele.

2015. aastast on Nõmme Eesti kabepealinn. Nõmme kabeklubi korraldab aastas mitmeid turniire. Turniiride koostamiseks on vaja koostada turniiritabelid mängijatest, mis panevad paika, kes omavahel mängivad. Praegu kasutusel olevad programmid on iganenud ning kasutajale keerulised. Samuti puudub ühtne keskkond, kus saaks turniire hallata ning käimasolevaid turniire jälgida.

Praegu määratakse suurematel turniiridel osalejate paarid Šveitsi reeglite kohaselt kohtuniku poolt. Selline paaridesse määramine võib mõne osaleja jaoks olla ebaõiglane, kuna kohtunik ei pruugi tegutseda erapooletult. Autori loodav rakendus määrab paarid automaatselt sisse ehitatud reeglite kohaselt, muutes paaridesse määramise võimalikult erapooletuks ning lihtsaks.

2018. aastal lõi Kerdo Kullamäe IT Kolledžis bakalaureusetööna Eesti kabeliidu jaoks veebirakenduse, mis võimaldas korraldada individuaalturniire [3] . Antud rakendus üritas kõrvaldada puudusi, mis hetkel kabeliidu poolt kasutusel olevate turniiride läbiviimisel eksisteerivad, kuid see jäi kahjuks poolikuks ning reaalselt kasutamata. Lisaks puudus meeskonnaturniiride läbiviimise võimalus. Praeguse töö autori eesmärgiks on arendada sarnane veebirakendus Nõmme kabeklubile, mis lisaks individuaalturniiridele võimaldaks ka meeskonnaturniiride läbiviimist, ning jõuda arenguetappi, kus loodav veebirakendus võetakse reaalselt ka kabeklubi poolt kasutusele.

Bakalaureusetöö on jaotatud erinevateks osadeks. Esmalt tutvustatakse üldist tausta ning olemasolevaid lahendusi. Seejärel analüüsitakse võimalikke lahendusi ja kirjeldatakse rakenduse tegemiseks kasutatud tehnoloogiaid. Kolmandas peatükis arutatakse uue rakenduse arenduse ja implementeerimisega seotud teemasid. Peale seda kirjeldatakse rakenduse testimise meetodeid ja tulemusi. Viimases peatükis vaadeldakse tehtud töö tulemust ning võimalikke suundi rakenduse edaspidiseks arendamiseks.

2 Analüüs

Selles peatükis tutvustatakse kõigepealt kabeklubi poolt kasutusel olevaid turniiride läbiviimise meetmeid. Seejärel tuuakse välja eri tüüpi nõuded, mida loodav rakendus peab rahuldama, ning kirjeldatakse ka Šveitsi ja ringsüsteemi. Peatüki lõpus võrreldakse kahte võimalikku arhitektuuritüüpi, mida saab kasutada rakenduse realiseerimiseks.

2.1 Kabeklubi poolt kasutusel olevad meetmed turniiride läbiviimiseks

Hetkel kasutab Nõmme kabeklubi kabeturniiride koostamiseks ja läbiviimiseks enamjaolt 3 meedet. Võimalus on koostada turniiritabelid käsitsi või kasutada ühte kahest levinud programmist: Chessarbiter [4] ja Swiss Perfect [5] .

- Kõige pikemalt kasutatud viis on koostada turniiritabelid käsitsi. See on väga ajakulukas korraldajatele, kuid pakub paindlikkust voorude vahepeal turniiritabelite uuesti koostamisel kui mõni osalejatest on sunnitud katkestama. Antud viis on aga vastuvõtlik inimeksimustele ning avab võimaluse, et turniiris osalejaid ei ole rahul, kuidas korraldajad neid paaridesse jagavad.

- Chessarbiter on poolakate poolt 2004. aastal loodud programm. Sellega saab koostada nii individuaal- kui ka meeskonnaturniire. Chessarbiter pakub osalejate paarimiseks Šveitsi süsteemi või ringsüsteemi. Programm täidab küll kabeklubi nõuded, kuid vajaks hädasti uuendusi. Seda pole uuendatud juba 10 aastat ning vajab töötamiseks spetsiifilisi nõudeid. Programm töötab arvutis, kus on kasutusel Windowsi operatsioonisüsteem ning sellel puudub võimalus näha tulemusi interneti vahendusel.

- Kolmandaks turniiride läbiviimise meetmeks on kabeklubi kasutanud Swiss Perfect programmi. See on maailmas kõige levinum male- ja kabeturniiride korraldamise programm, mida on kasutatud üle kümne aasta. Programm pakub sarnaselt Chessarbiterile paarimiseks Šveitsi süsteemi ning ringsüsteemi. Selle

suureks miinuseks on aga meeskonnaturniiride korraldamise võimaluse puudumine.

Kõik väljatoodud viisid on ajale jalgu jäänud. Chessarbeiter ja Swiss Perfect katavad küll enamuse turniiri koostamise nõuetest, kuid on väga suure õppekõveraga. Samuti ei anna ükski antud meetmetest kasutajale reaalajas infot turniiritulemustest. Käsitsi korraldavate turniiride tabelite ja tulemuste jagamine on piiratud ning eespool mainitud programmide jaoks on vaja kindlat operatsioonisüsteemi nende jooksutamiseks. Tänapäeval peaks olema rakendus kasutaja riistvarast suhteliselt sõltumatu ning vabalt ligipääsetav.

2.2 Loodava rakenduse nõuded

Töö eesmärgiks on luua töötav rakendus, mis lihtsustaks turniiride koostamise ja haldamise. Et antud eesmärk täita võimalikult hästi, koostas autor funktsionaalsed nõuded koos Nõmme kabeklubi liikmete Heinar Jahu ja Irma Nahkoriga, kes viivad läbi igal aastal mitmeid turniire ning teavad täpselt olemasolevate rakenduste puuduseid ning vajadusi. Järgnevad nõuded sarnanevad osalt Kerdo Kullamäe töös [3] välja toodud nõuetega, kuna need töötati välja koostöös samade inimestega.

2.2.1 Rakenduse funktsionaalsed nõuded

Funktsionaalsed nõuded määravad ära, kuidas rakendus peaks käituma ning milliseid võimalusi pakub rakendus kasutajale. Antud nõuded on individuaalsed tegevused, mida süsteemiga peab saama teha [6] .

- Turniiriga seotud nõuded
    - Peab saama sisestada nime, kirjeldust, asukohta, kohtunikku, laua suurust, turniiri tüüpi, turniiril kasutatavat paarimise süsteemi, turniiri algusaega, lisaaega ja käikude kirjutamise kohustuslikkust
    - Šveitsi turniiri puhul peab saama veel sisestada voorude arvu
    - Meeskonnaturniiri puhul peab saama sisestada punktisüsteemi

    - Meeskonnaturniiril peab saama sisestada meeskonna suurust
    - Meeskonnaturniiril peab saama lisada/eemaldada meeskondi
    - Individuaalturniiril peab saama lisada/eemaldada mängijaid
- Mängijaga seotud nõuded
    - Peab saama sisestada nime, tiitlit, 64 ruudu kabe reitingut, 100 ruudu kabe reitingut, sünniaega, sugu
    - Peab saama lisada/eemaldada meeskonda
    - Peab saama lisada turniirile
- Meeskonnaga seotud nõuded
    - Peab saama sisestada nime
    - Peab saama lisada/eemaldada mängijaid
    - Peab saama lisada turniirile
- Kohtunikuga seotud nõuded
    - Peab saama sisestada kasutajanime, eesnime, perenime, parooli
    - Peab saama logida sisse rakendusse
    - Peab saama hallata turniire, mängijaid ja meeskondi
    - Peab saama alustada ja lõpetada turniiri
    - Peab saama sisestada voorude tulemusid
    - Peab saama alustada ja lõpetada vooru.

2.2.2 Rakenduse mittefunktsionaalsed nõuded

Mittefunktsionaalsed nõuded kirjeldavad, kuidas süsteem on üles ehitatud tarkvara seisukohast ja mille alusel hinnatakse selle töökäiku, mitte kindlaid omadusi [6] .

- Kasutajaliides peab olema eesti keeles
- Kood on kirjutatud inglise keeles
- Rakendus peab töötama järgnevate veebilehitsejate viimases kahes versioonis
    - Google Chrome
    - Internet Explorer
    - Firefox
    - Safari
- Rakendus peab olema kasutatav ka väiksema resolutsiooniga seadmetes.

2.3 Šveitsi süsteem

Šveitsi süsteem on mitte elimineerimisega turniiri formaat, kus on kindel arv voore. Antud süsteemi kasutatakse kui osalejaid on liiga palju, et korraldada kõik-kõigiga stiilis turniiri. Igas voorus paaritakse osalejad omavahel ning vooru lõppedes võidab see, kes kogus kõige rohkem punkte kokku [7] .

2.3.1 Põhikriteeriumid

1. Voorude arv otsustakse enne turniiri algust ning ei tohiks olla väiksem kui 1/3 ega suurem kui pool osalejate arvust.
2. Samad mängijad tohivad üksteisega mängida ainult ühe korra.
3. Paarimine toimib kindlate reeglite alusel ning paarid peaks koostama võimalikult võrdsed.
4. Kui võimalik, antakse mängijale mustad ja valged kabendid ühesugune arv kordi.
5. Kui võimalik, peaks mängija saama iga mäng erineva värvi kui eelmises voorus.

6. Turniiri lõppjärjestus määratakse mängijate kogutud punktide alusel [7] .

2.3.2 Mängija vabaks jätmine

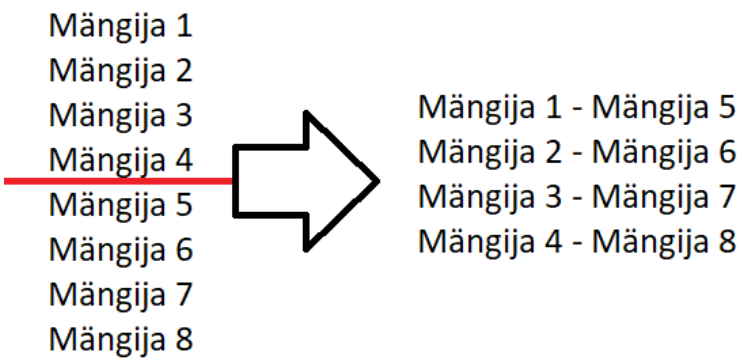
Juhul kui voorus on mängijaid paaritu arv, siis jääb vabaks viimases voorus madalaima reitinguga mängija. Mängija, kes on osalejate paaritu arvu tõttu üksi jäänud või kelle vastane ei ilmu välja, saab automaatselt kirja võidu. Sel juhul ei tohi teda jätta uuesti vabaks mängijate paaritu arvu tõttu. Selle valimisel arvestatakse, et grupi ülejäänud saaksid omavahel mängida ja nende värvide sobivust. Vaba vooru eest saab mängija sama palju punkte kui võidu eest.[7]

2.3.3 Mängijate paarimine

Mängijate paarimisel voorudes on samuti kindlad reeglid. Esimese vooru paarimine on eriline, kuna punktiarvestus puudub. Esimeses voorus paarimise jaoks on olemas mitmeid eri mooduseid. Enne esimest vooru reastatakse mängijad turniiritabelis arvestades:

1. Reitingut
2. Tiitlit
3. Mängijate arvatavat tugevust või tähestikulist järjekorda.

Seejärel paaritakse mängijad vastavalt Joonis 1-le.

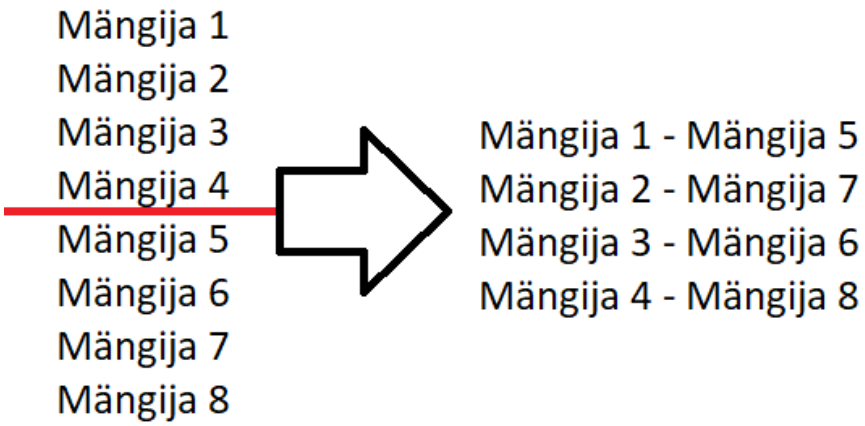


Joonis 1: Esimese vooru mängijate paarimine.

Peale esimest vooru käib paarimine kindlate reeglite alusel. Osalejad jaotatakse punktide alusel gruppidesse. Kokku võib paarida mängijaid, kes ei ole omavahel varem kohtunud, kusjuures kumbki mängija ei tohi mängida järjest ühe värviga rohkem kui 3 vooru. Paarimise hõlbustamiseks alustatakse paaride moodustamist alati kõrgemast grupist kuni keskmise grupini. Seejärel minnakse paarimisega madalaima punktigrupini ja paaritakse altpoolt üles. Keskmise punktigrupi paarimine toimub alati viimasena.

Enne punktigrupi paaride määramist kontrollitakse, kas grupis on mängijaid, kellel puudub sobiv vastane (st. on kõigiga juba kohtunud, punktigrupp on paaritu või ei saa vältida olukorda, kus tekib 3 korda järjest sama värv). Selliseid mängijaid nimetatakse liuglejateks ning nad tõstetakse vastavalt üle kõrgemasse/madalamasse punktigruppi.

Punktigrupp jagatakse vastavalt turniiritabelis olevale järjekorrale pooleks. Mängijaid hakatakse grupis ümber tõstma kui mõni paar on omavahel juba kohtunud. Allapoole paarimisel alustatakse vastase otsimisest kõrgemast (turniiritabelis eespool) mängijast. Sel juhul tõstetakse ümber madalamaid (turniiritabelis tagapool) mängijaid. Kui punktigrupis on 8 mängijat, siis paarimine käib vastavalt Joonis 1-le. Kui aga näiteks „Mängija 2” on juba mänginud mänginud „Mängija 6”-ga siis tõstetakse mängijad ümber vastavalt Joonis 2-le.



Joonis 2: Mängijate paarimine peale ümbertõstmist.

Kui keskmist punktigruppi ei suudeta paarida, siis samm-sammult laiendatakse seda järgmiselt: kui laskujaid on rohkem kui tõusjaid, siis keskmisest punktigrupist alumise grupi I paar tühistatakse ja selle paari mängijad tuuakse üle keskmisse punktigruppi.

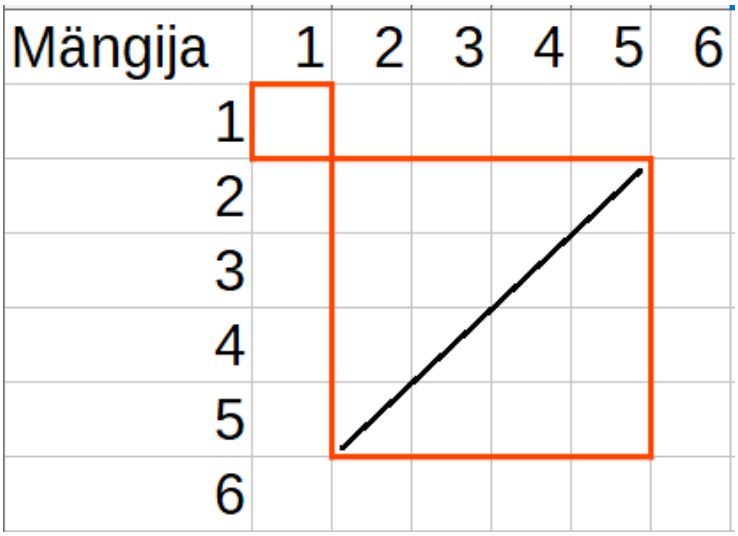
Seejärel proovitakse laiendatud keskmises punktigrupis uuesti paare moodustada. Kui laskujaid on vähem kui tõusjaid, siis tühistatakse keskmisest punktigrupist ülemise grupi viimane paar ja tuuakse vastavad mängijad keskmisse punktigruppi [7] .

2.4 Ringsüsteem

Kui turniiril osalejate arv on väike, eelistatakse enim ringsüsteemi kasutamist. Selle eelisteks on, et kõik osalejatest saavad mängida kõigiga läbi ning turniiri tulemused on võimalikult õiglased. Ringsüsteemi ei saa hästi rakendada suurte gruppide puhul, kuna mängude arv kasvab liiga suureks.

2.4.1 Mängijate paarimine

Kui turniiril on paarisarv osalejaid, loositakse mängijate järjekord (autori tehtavas rakenduses järjestatakse osalejad reitingu, pere- ja eesnime järgi) ning kantakse tabelisse. Mängijatest moodustakse ruudukujuline tabel, mille põhjal hakatakse mängijaid paarima. Tabelis moodustatakse vastavalt vooru numbrile mängijate kohta ruudud. Näiteks kui on turniiril kuus mängijat ning tegemist esimese vooruga, siis tehakse ruut üks ruut esimese mängija tabelisse ning teine mängijate kaks kuni viis ümber. Viimane mängija jäetakse nendest ruutudest välja (vt Joonis 3). Tehtud ruutudesse tõmmatakse alamdiagonaalid, mille kaudu määratakse paarid. Kuna mängija 1 ei saa iseendaga mängida, paaritakse tema mängijaga 6. Alamdiagonaalidega tuleb välja, et omavahel paaritakse ka mängijad 2 ja 5 ning 3 ja 4 [8] .



Joonis 3: Ringsüsteemi I vooru mängijate paarimine

Järgmises voorus tehakse uus ruut mängijate 2 ja 1 ümber ning teine ruut mängijate 3 kuni 5 ümber ning nad jagatakse sarnaselt esimese vooruga paaridesse. Tabel 1 toob välja mängijate paarimise 6 mängijaga ringsüsteemis kõigi viie vooru kohta [9] .

Tabel 1: Ringsüsteemi voorude paarid.

Voor	**Vooru mängijate paarid**		
1. voor	1 ja 6	2 ja 5	3 ja 4
2. voor	6 ja 4	5 ja 3	1 ja 2
3. voor	2 ja 6	3 ja 1	4 ja 5
4. voor	6 ja 5	1 ja 4	2 ja 3
5. voor	3 ja 6	4 ja 2	5 ja 1

2.4.2 Värvide määramine

Kui turniiril on paarisarv osalejaid, siis peale mängijate järjestamist on kindlalt teada ainult viimase mängija partiide värvid. Kui ta on vastamisi tabeli esimese poole mängijatega, siis mängib ta mustade kabenditega, kui aga vastaseks on tabeli teise poole mängija, siis saab ta mängida valgete kabenditega. Turniiri lõppedes on viimane mängija kasutanud musti kabendeid ühe võrra rohkem kui valgeid. Näiteks kui turniiril on kuus mängijat ja järjekorras viimane mängija on „Mängija 6”, siis eelpool kirjeldatud kabendite värvide määramine käib just tema kohta. Ülejäänud mängijate kabendite värvi määramine eri voorude puhul sõltub paaris olevate mängijate järjekorranumbrite

summast. Kui summa on paarisarv, siis mängib väikseim number mustade kabenditega ja suurem valgetega, kui aga summa on paaritu arv, siis vastupidi. Näiteks 1.voorus on paarides mängijad 1-6, 2-5 ja 3-4 (vt Tabel 1). Kuna „Mängija 6” on järjekorras viimane ja tema kabendite värvid on määratud kindlate reeglite kohaselt, siis ei pea selle paari puhul summat kokku liitma. „Mängija 1” on tabeli esimese poole mängija, järelikult saab „Mängija 6” mustad kabendid ning „Mängija 1” valged kabendid. Mängupaaride 2-5 ja 3-4 puhul tuleb kokku liita nende järjekorra numbrid, mis mõlemal juhul annavad summaks seitsme. Seitse on paaritu arv, seega paaride 2-5 ja 3.-4 puhul saavad „Mängija 2” ja „Mängija 3” valged kabendid ning „Mängija 5” ja „Mängija 4” mustad kabendid [8] .

2.5 MPA või SPA disainimuster

Enne rakenduse loomist analüüsis autor kahte erinevat disainimustrit, mida kasutatakse veebirakenduste ehitamiseks, et määrata kindlaks, milline on selle töö jaoks sobivam variant. Nendeks mustriteks on MPA (multi page application) ja SPA (single page application).

MPA on klassikaline disainimuster, mida on kasutatud veebilehtede arendamiseks aastaid. Iga lehe navigeerimisega saadetakse päring serverisse ning uuendatakse kogu lehte. See tähendab, et serverist laetakse uuesti ka need andmed, mis ei muutunud. Tabel 2 toob välja MPA puudused ja eelised.

Tabel 2: MPA eelised ja puudused.

Eelised	**Puudused**
Lihtne otsingumootorite optimiseerimine	Äppide tegemine on väga suur investeering kuna taaskasutavat koodi ei ole.
Vajab väiksemat hulka eri tehnoloogiaid.	Serveri- ning kliendipoolne kood on tihedalt seotud ning raskesti eristatavad.
Olemas on suur valik erinevat lahendusi mis on kohe kasutamisvalmis ning ei nõua suurt investeeringut	

SPA on veebirakendus, mis simuleerib töölauarakenduste tööd. See tähendab, et kui kasutaja liigub lehel ringi, siis lehte uuendatakse dünaamiliselt ainult nende andmetega, mida kasutaja küsib. Tabel 3 annab ülevaate SPA eelistest ja puudustest [10] , [11] .

Tabel 3: SPA eelised ja puudused.

Eelised	**Puudused**
Lehel navigeerimine on sujuv ning kiire	Halb otsingumootorite optimeerimine kuna andmete laadimine käib läbi JavaScript-i.
Skaleeruvat disaini on lihtsam teha	Kui kasutaja on keelanud JavaScript-i brauseris siis veebileht ei tööta
Serveri- ja kliendipoolne kood on selgelt eraldatud.	
Äppide tegemine on lihtsam kuna serveripoolset koodi saab taaskasutada ka äpi andmete jaoks	

Antud rakenduse jaoks otsustas autor kasutada SPA disainimustrit. Kuna tulevikus on plaan teha äpp, mis võimaldaks pealtvaatajatel lihtsat infot saada turniiride kohta, siis selle tarbeks on SPA disainimuster parem. Kasutaja seisukohast on rakenduse sujuvus äärmiselt oluline ning seda võimaldab SPA paremini teha kui MPA. SPA puhul saab praegu tehtud koodi kasutada ka äpi andmete jaoks, kuna serveripoole lähtekood ei ole tihedalt seotud veebist nähtava kasutajaliidesega. Valitud disainimustri eeliseks on ka autori kogemustepagas. Nimelt on autoril mitmeaastane kogemus SPA rakenduste tegemisel.

3 Rakenduse arendus

Eelnevas peatükis toodi välja, et loodava rakenduse jaoks kasutatakse SPA disainimustrit. Sellest üksi aga rakenduse loomiseks ei piisa. Järgnevalt kirjeldatakse rakenduse loomiskäiku ning selle arendamisel kasutatavaid tööriistu.

3.1 Arenduskäik

Esmalt analüüsis autor rakenduse nõudeid ning enda oskuseid ja valis antud rakenduse jaoks sobivad tehnoloogiad. Peale seda otsustas ta millist arhitektuuri tuleks kasutada. Autor otsustas serveripoolse koodi ehitada Yii2 [24] raamistikul, kuna tal on 2-aastane kogemus selle kasutamisel ning vajadusel on võimalus pidada aru teiste pika kogemusepagasiga arendajatega. Front-end kirjutamisel otsustati kasutada Angular7 [13] raamistikku, kuna see on üks kõige paremaid lahendusi serveri- ja kliendipoolse osa eraldi tegemiseks. Lisaks on sel suur kasutajapõhi ning mitmed tuleviku veebilahendused kirjutatakse just Angulari raamistikku kasutades.

Rakenduse sujuvaks arendamiseks jagas autor vajalikud funktsionaalsused eraldi loogilisteks mooduliteks. Seejärel arendas autor iga mooduli eraldi arendustsüklis. Mooduliteks olid näiteks meeskondade- ja mängijate haldus.

Arendamist alustas autor sellest, et tegi rakenduse jaoks kaustadest põhja. Põhi koosnes api ja frontend kaustast, mis eraldavad serveri- ja kliendipoolset lähtekoodi. Seejärel ühendas autor ära projekti versioonihaldustarkvaraga GIT [14] . Peale seda võttis autor Yii2 arendajate poolt tehtud üldise projekti malli ning muutis seda vastavalt oma rakenduse vajadustele. Eemaldati üleliigsed koodiosad ning lisati baasfunktsionaalsus, mis on vajalik API lõpp-punktide tegemiseks. Selle hulka kuulus ka lõpp-punktide turvalisuse tagamine.

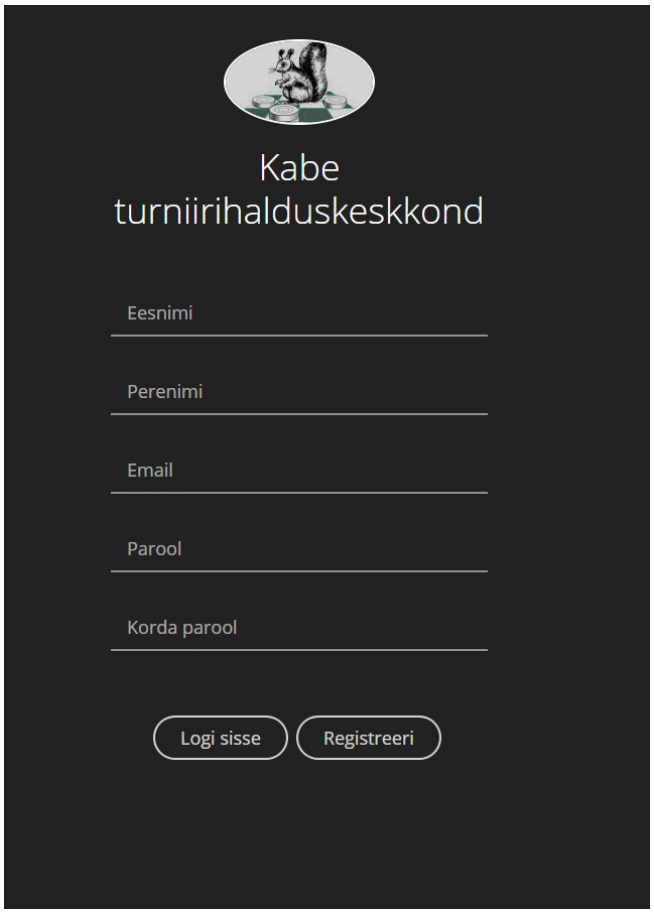
Kui serveripoolne rakenduse põhi oli seadistatud, suundus autor kliendipoolse koodi kaustade struktuuri seadistamiseni kasutades Angulari raamistiku malli. Ilusa kujunduse

saamiseks otsustas autor kasutada „SB Admin 2” [15] , mis on avatud lähtekoodiga kasutajaliidese kujundusemall.

Peale rakenduse arhitektuuri ja koodipõhjade implementeerimist analüüsis autor rakenduse funktsionaalseid nõudmisi ning disainis nende põhjal andmebaasi.

3.1.1 Kasutaja registreerimise ja sisselogimise vaated

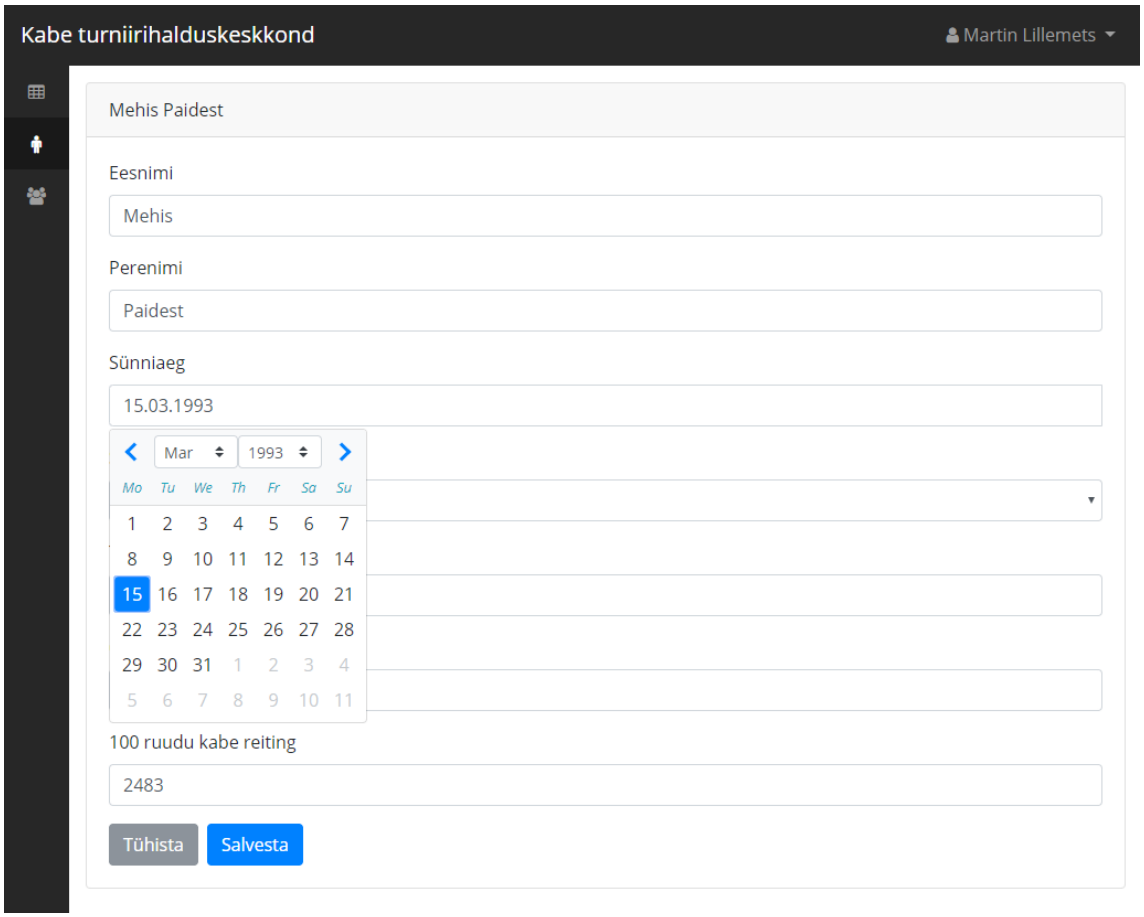
Rakenduse vaadete arendamist alustati kohtuniku registreerimise ja sisselogimise vaadetest (vt. Joonis 4). Registreerimiseks on vajalikud kasutaja eesnimi, perenimi, email ning parool.



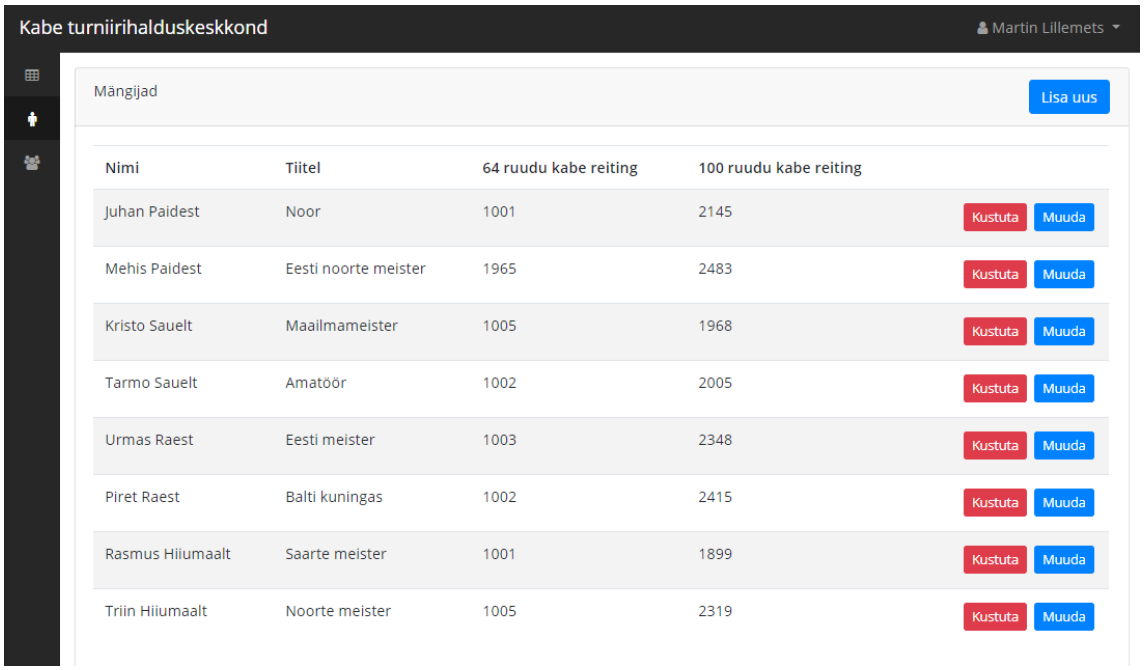
Joonis 4: Kohtuniku registreerimise vaade.

3.1.2 Mängijate haldus

Et kohtunikud saaksid mängijaid turniiridele lisada, peab olema neil võimalus mängijaid hallata. Selleks tegi autor mängijate jaoks kolm vaadet: mängijate lisamine, mängijate muutmine (vt Joonis 5) ja mängijate nimekirja vaade (vt Joonis 6), mis pakuks ülevaadet olemasolevatest mängijatest. Nende vaadete kõige suuremaks ajakuluks oli mängija sünnikuupäeva väli. Antud välja implementeerimine oli keeruline, kuna kasutajale kuvatav väärtus, serverisse saadetav väärtus ning kuupäeva valimise komponendi kuupäeva väärtus olid kõik erineval kujul.



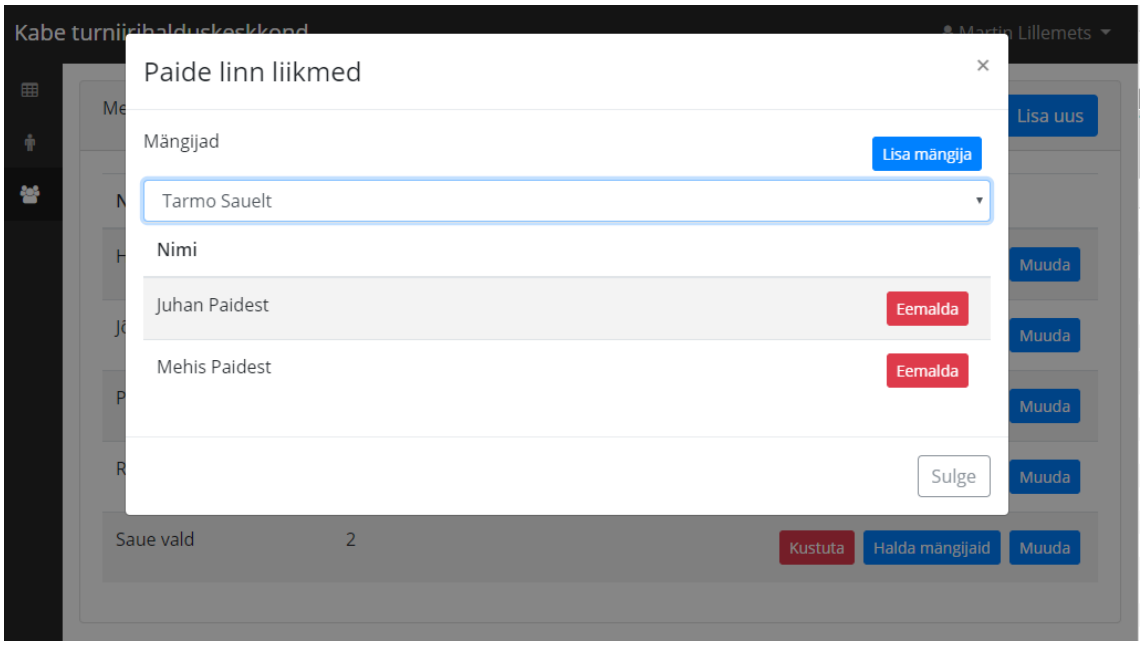
Joonis 5: Mängija muutmisvaade.



Joonis 6: Mängijate nimekirja vaade.

3.1.3 Meeskondade haldus

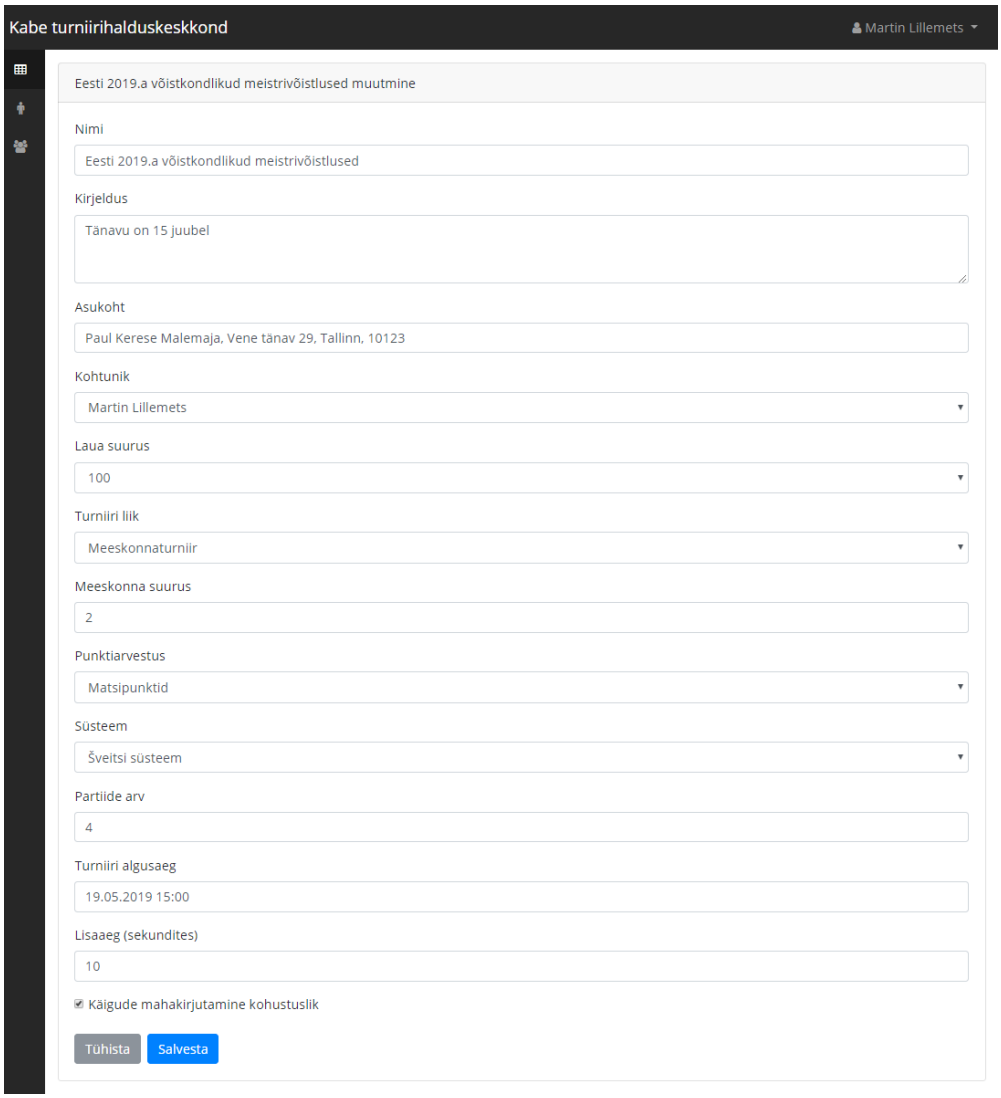
Meeskonnaturniiride läbiviimiseks oli vaja autoril teha meeskondade haldus. Meeskond ise on grupp mängijad, kes võistlevad turniiril üheskoos. Meeskondadeks võivad turniiridel olla näiteks koolid, kes saadavad enda esindajad, või erinevad kabeklubid. Kõigepealt tegi autor meeskondade nimekirja vaate, kus kuvatakse meeskondade nimesid ning liikmete arvu. Seejärel läks ta edasi meeskonna lisamise ja muutmise vaatele. Lisamise ja muutmise vaadete tegemine sujus autoril kiiresti, kuna meeskonnal on ainult üks kasutaja poolt sisestatav väli. Selles arendamise tsüklis läks põhiaeg mängijate ja meeskondade sidumise vaate tegemiseks (vt Joonis 7).



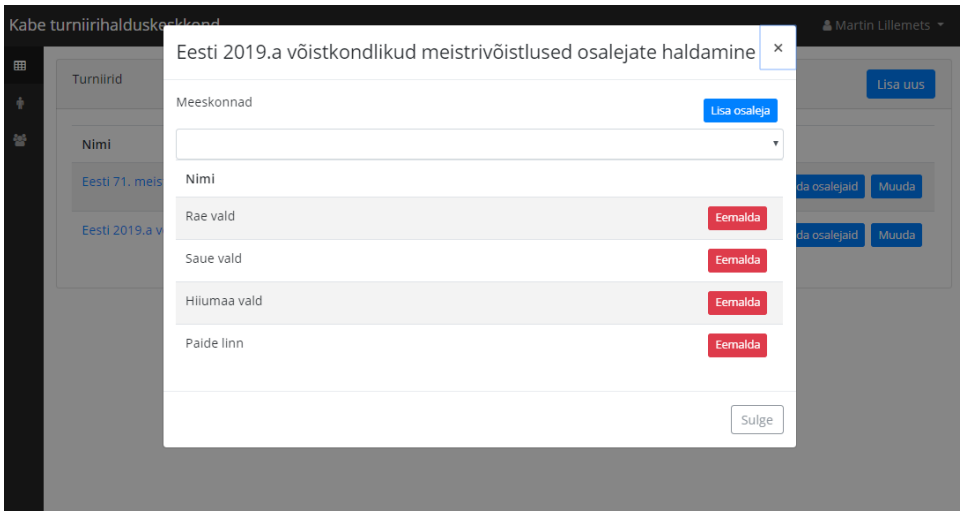
Joonis 7: Meeskonnaliikmete haldamise vaade.

3.1.4 Turniiride haldus

Veebirakenduses olev funktsionaalsus on tehtud just turniiride haldamiseks. Peale tugivaadete arendamist alustas autor kõige keerulisema osa arendust. Taaskord lõi ta kõigepealt turniiri nimekirja vaate, mis kuvab kasutajale turniiri nime, tüüpi, staatust ja osalejate arvu. Peale seda arendas autor turniiri lisamise ja muutmise vaated (vt Joonis 8), kus kohtunik saab seadistada kõik turniiri reeglid. Turniiri osalejate vaates (vt Joonis 9) saab kohtunik lisada turniirile osalejaid. Vastavalt turniiritüübile pakutakse kohtunikule süsteemis olevaid mängijaid või meeskondasid.



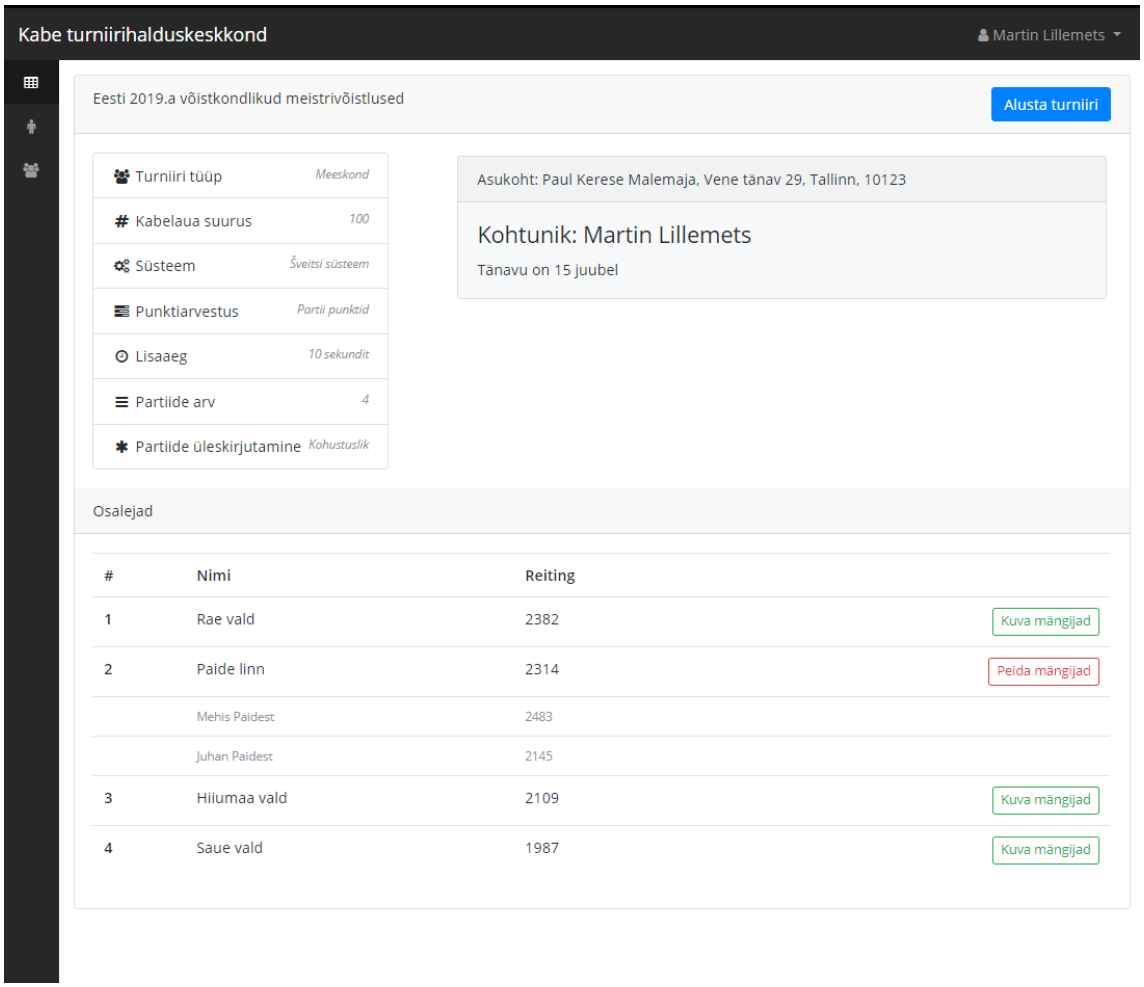
Joonis 8: Turniiri muutmise vaade.



Joonis 9: Turniiri osalejate sidumise vaade.

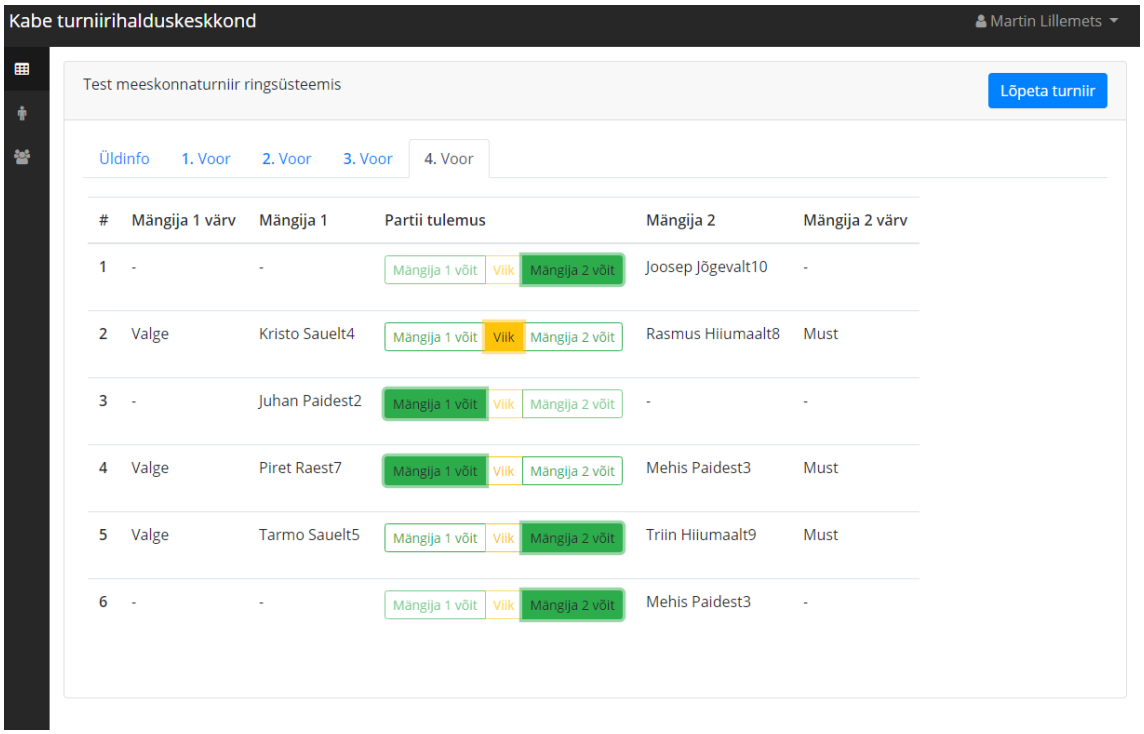
3.1.5 Turniiride läbiviimine

Turniiride läbiviimiseks tegi autor turniiri kõigepealt turniiri detailvaate (vt Joonis 10), mis annab kohtunikule ülevaate turniiri seadetest ja osalejatest. Tehtud vaates on ka turniiri alustamise nupp, mida vajutades küsitakse kohtunikult veel viimast kinnitust enne turniiri alustamist.

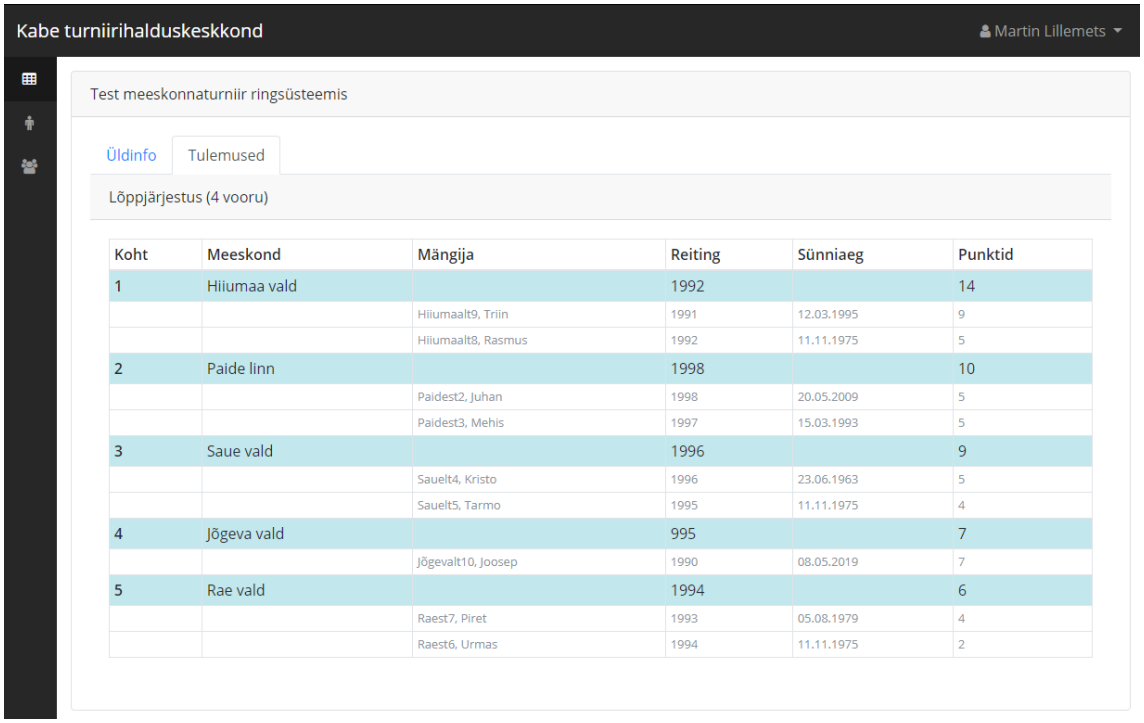


Joonis 10: Turniiri detailvaade.

Turniiri alustamine tähendab rakenduse jaoks esimese vooru loosimist. Rakendus paarib kõik osalejad vastavalt turniiri reeglitele (vt. Lisa 2 – Ringsüsteemi paarimise kood) ning suunab kohtuniku esimese vooru vaatesse. Antud vaatest saab kohtunik sisestada iga selle vooru paari tulemused. Kui voorus on kõikide paaride tulemused ära sisestatud, saab kohtunik minna turniiriga edasi järgmise vooruga (vt. Joonis 11). Kui viimane voor on mängitud, saab kohtunik turniiri lõpetatuks märkida, mille tagajärjel suunatakse kohtunik turniiri tulemuste vaatesse (vt. Joonis 12). Turniiri tulemuste vaates kuvatakse turniiris osalejate paremusjärjestus koos punktidega.



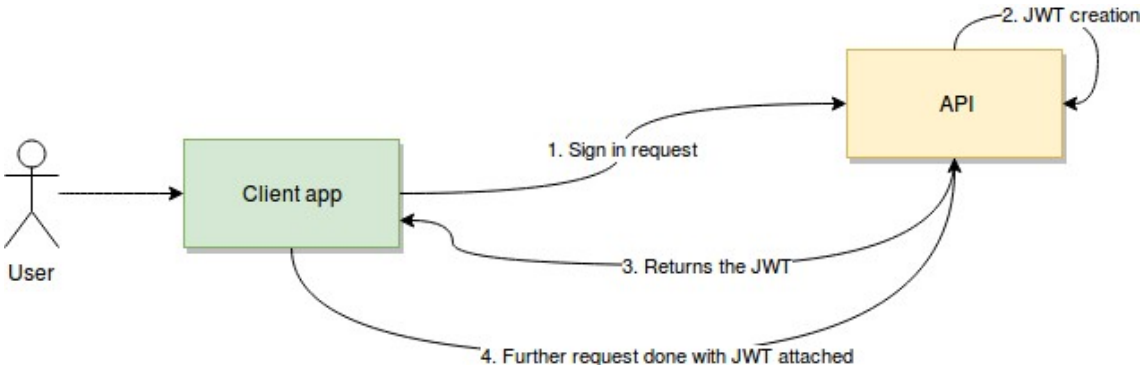
Joonis 11: Vooru tulemuste sisestamise vaade.



Joonis 12: Turniiri tulemuste vaade.

3.2 Turvalisus

Rest API töötab põhimõttel, et iga päring on individuaalne ning puudub ühtne session. Seepärast peab iga päringuga olema kaasas autoriseerimise võti, mis ütleks rakendusele, kas päringu tegija tohib andmetele ligi pääseda. Selle tarbeks kasutab autor JWT (JSON web token) võtit, mida kontrollitakse enne igat päringut. JWT on standardiseeritud JSON-i kuju, mille abil saab turvaliselt andmeid erinevate rakenduse osade vahel saata [16] .Võtme genereerimine ja kasutamine on selgitatud Joonis 13-l.



Joonis 13: JWT võtme genereerimine [17] .

3.3 Andmete liikumine serveri ja kliendi vahel

Disainitaval rakendusel on selgelt eraldatud serverikood (edasipidi yii) ja kliendipoolne kood (edasipidi angular). Eespool nimetatud rakenduse osade vahel käib aga suhtlus, kus kasutaja tegevuse tagajärel saadab või küsib angular andmeid yii käest.

Andmete transportimine käib läbi REST API. See tähendab, et kogu andmesuhtlus käib JSON formaadis ja läbi HTTP päringute.

Meeskonna loomise päringu näide:

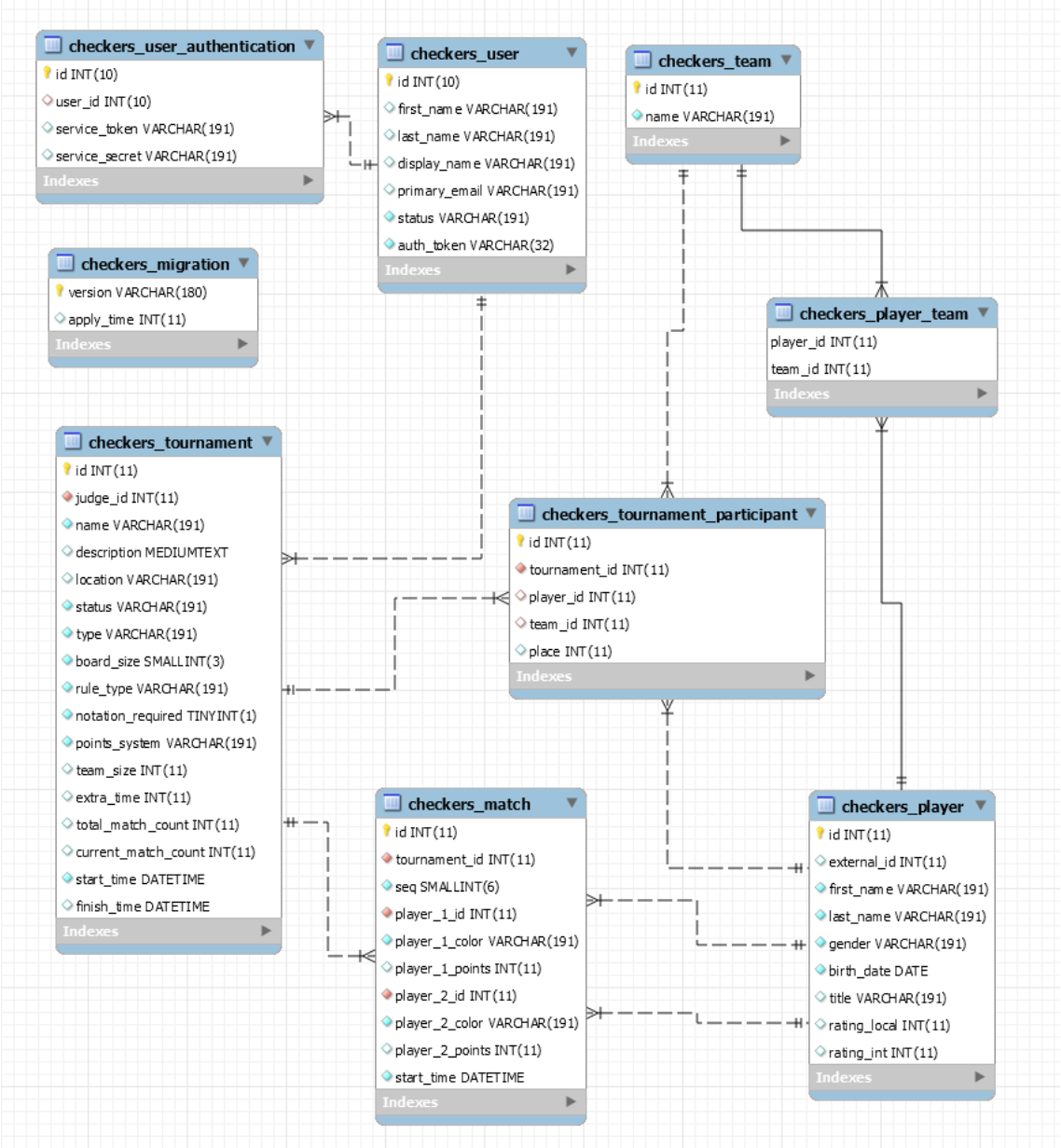
curl -X POST https://checkers.martin.brainart.ee/team/team -H 'Accept: application/json' -H 'Content-Type: application/json' -H 'X-Authorization: Bearer eyJ0eXAiOiJKV1...' -d '{"name":"Nõmme Kabeklubi"}'

Antud päringust on näha, et päring tehakse https://checkers.martin.ee/team/team aadressile POST tüüpi päringuna. Päringu päises Content-Type defineerib ära andmete

formaadi. Kuna meeskonna loomine on lubatud ainult kohtunikule, siis on päringuga kaasas ka autentimise võti X-Authorization päise all. Näites on võtit lühendatud ... (kolme punkti) abil. Kõige lõpuks on päringuga kaasas meeskonna andmed, mis kasutaja oli sisestanud.

3.4 Andmebaas

Rakenduse andmebaas koosneb üheksast tabelist (vt Joonis 14). Tabelites checkers_user ja checkers_user_authentication hoitakse andmeid kohtunike ja nende kasutajate kohta. Tabelis checkers_team hoitakse meeskondade infot ning tabelis checkers_player mängijate infot. Andmebaasi suurimaks tabeliks on checkers_tournament tabel, mis hoiab endas turniiridega seotud andmeid. Andmebaasis on kaks seoste tabelit checkers_player_team ja checkers_tournament_participant, millest esimene seob mängijad meeskondadega ja teine seob meeskonnad ja mängijad turniiridega. Vooru infot hoitakse checkers_match tabelis, mis seob ära, kes kellega mängis, mis värvi kabenditega keegi mängis ning kui palju punkte sai. Viimaseks tabeliks on süsteemitabel, mis hoiab ülevaadet rakenduse andmebaasi struktuurist ning võimaldab vajadusel lihtsalt modifitseerida tabelite struktuuri ning väljasid.



Joonis 14: Andmebaasi skeem.

3.5 Kasutatud tehnoloogiad

Rakendus koosneb kahest erinevast koodigrupist. Serveripoolne osa rakendusest on kirjutatud PHP keeles Yii2 raamistiku peale. Raamistik kasutab andmebaasiga suhtlemiseks Yii sisse pakendatud ORM-i. Rakenduse kliendipoolne osa on kirjutad TypeScript [18] keeles ning kasutatud on Angular 7 raamistikku.

3.5.1 PHP

Autor kasutas serveripoolse keelena PHP-d. PHP on kõige laialt levinum skriptimiskeel mida kasutatakse veebilehtede tegemiseks. 79% maailma veebilehtedest kasutab PHP'd serveripoolse keelena [19] . PHP fokuseerib peamiselt serveripoolsele skriptimisele ning sellega on võimalik teha andmebaasi päringuid, genereerida dünaamilisi veebilehtede sisu ja manipuleerida küpsiseid. PHP-d saab kasutada kõikides suuremates operatsioonisüsteemides nagu Linux, Windows, macOS. PHP skripte saab jooksutada ilma serveri või brauserita ning nende jooksutamiseks on vaja ainult PHP parserit. See teeb PHP ideaalseks, et jooksutada skripte regulaarselt cron-i(Linux) või sündmuste planeerijaga (Windows-is) [20] .

3.5.2 TypeScript

TypeScript on Microsofti poolt JavaScript-i peale ehitatud vabavaraline programmeerimiskeel. See on disainitud JavaScript arendajatele ning suurtele JavaScript-i rakendustele. TypeScript aitab organiseerida koodi dünaamiliselt laetavateks mooduliteks. TypeScript toetab tüübipõhist programmeerimist ja võimaldab arendajatel kasutada produktiivseid tööriistu ja arendamisvõtteid. TypeScript on sünteetiline suhkur JavaScriptile, mis tähendab, et kõik JavaScript-i rakendused töötavad ka TypeScript-is [18] .

3.6 Kasutatud tööriistad

Rakenduse loomisel kasutati erinevaid tööriistu, mis aitasid kaasa selle lihtsamaks ja kiiremaks arenduseks.

3.6.1 PhpStorm

PhpStorm on IntelliJ IDEA loodud PHP keele IDE (Integrated development environment). PhpStorm teeb koodi kirjutamise kiiremaks, kuna pakub automaatset koodi täitmist ja koodi korrektsuse kontrolli. Sellega on võimalik jooksvalt koodi süntaktiliselt kontrollida laialt levinud koodikirjutamiste standarditega, mis kindlustavad, et kogu kood on ühtse stiiliga ning arusaadavalt kirjutatud. PhpStormis on sisse ehitatud ka CSS ja TypeScripti tekstiredaktorid ning see toetab samuti

andmebaasidega ühendumist. Antud rakenduse jaoks oli IDE eriti hea, sest peale vajalike keelte toe on PhpStormil ka Yii2 ja Angulari raamistiku tugi [21] .

3.6.2 Git

Git on avatud lähtekoodiga versioonihaldustarkvara. Selle looja Linus Torvalds lõi git-i, et lihtsustada Linuxi tuuma arendamist. Seda kasutatakse koodi terviklikkuse tagamiseks, jagamiseks ning programmeerijate koodide ühendamiseks. Git hoiab koodist ülevaadet ning iga lokaalne koodihoidla hoiab ülevaadet koodi mineviku kohta. Git pakub võimalusi koodi erinevate arendusstaadiumite versiooneerimiseks ning vajadusel isegi tagasi võtmiseks, kui mingi osa koodist on vigane [14] , [22] .

3.6.3 Adminer

Adminer on PHP keeles kirjutatud tööriist, mis on mõeldud hõlpsustamaks andmebaaside haldamist. Adminer pakub lihtsat ja võimsat kasutajaliidest, et teha andmebaasioperatsioone nagu tabelite loomine, välisvõtmete lisamine jpm. Autor kasutas antud tööriista, et luua vajalikud andmebaasitabelid ning genereerida andmebaasitabelite loomise SQL laused [23] .

3.6.4 Gii

Gii on veebipõhine koodi genereerimismoodul, mis on tehtud Yii raamistiku jaoks. Gii võimaldab arendajatel teha kiiresti ja mugavalt andmebaasitabelitele vastavaid olemi andmemudelid. Gii ühendub andmebaasiga ning genereerib olemasolevate tabelitele nende väljade põhjal vastavad PHP failid, mis peegeldavad ja hoiavad andmebaasist tulevaid andmeid [24] .

3.6.5 Postman

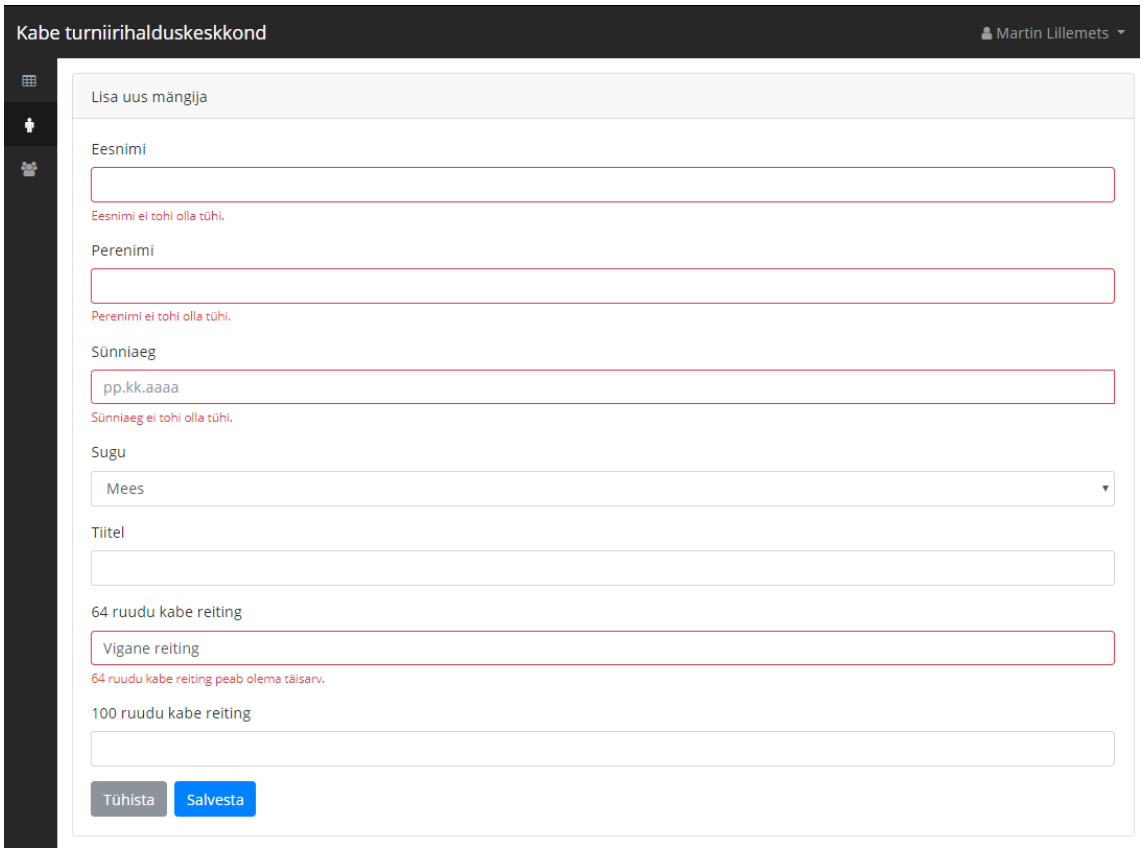
Postman on programmeerijatele suunatud abiprogramm, millega on võimalik testida API lõpp-punkte . Postman pakub mugavat kasutajaliidest, mille abil on võimalik teha HTTP päringuid ning anda neile kaasa vajalikke andmeid. Antud rakenduse arendamisel kasutas autor seda, et testida rakenduses olevaid api punkte ja nende valideerimisreegleid [25] .

4 Veebirakenduse testimine

Testimine leidis aset peale igat arendustsüklit, et leida jooksvalt rakenduse kitsaskohti ning need likvideerida. Järgnevalt toob autor välja rakenduse testimisel kasutatud meetodid ning testimise tulemused.

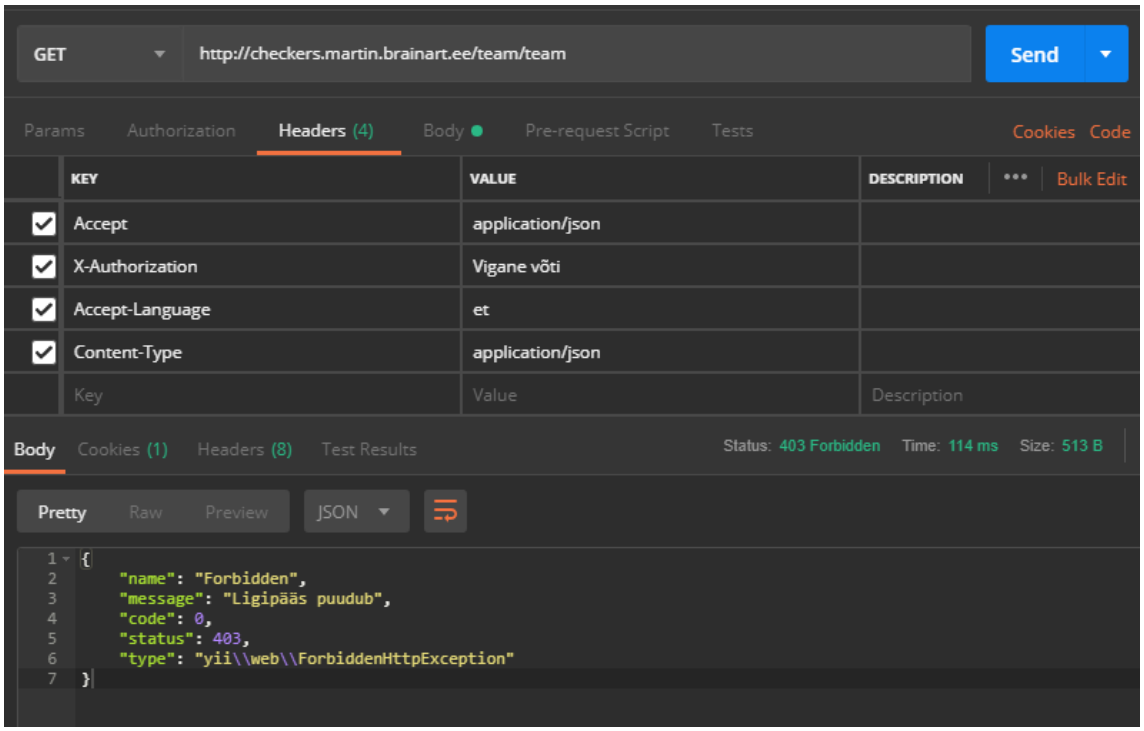
4.1 Testimise meetodid

Rakendust testiti vastavalt analüüsis väljatoodud funktsionaalsetele ja mittefunktsionaalsetele nõuetele. Testimise käigus navigeeriti rakenduses ringi ning testiti lehel olevaid nuppe, linke ja võimalusi. Testimisel üritati salvestada invaliidseid andmeid ning kontrolliti, kas rakendus käitub korrektselt ja näitab kasutajale veateateid. Joonis 15 näitab mängija lisamise vaates võimalikke vormiväljade valideerimisvigu. Näiteks ei tohi tühjaks jääda ees- ja perenime lahtrid. Lisaks ei saa nendes lahtrites olla ka arvulisi parameetreid. Reitingu lahtrisse ei saa sisestada muid sümboleid kui numbreid. Selle võib ka reitingu puudumisel tühjaks jätta. Antud juhul genereerib programm reitinguks automaatselt nulli. Lisaks võib jääda tühjaks ka lahter „Tiitel". Sünnikuupäeva lahtri puhul otsustas autor avaneva kalendervaate kasuks, et lihtsustada mängija loomist ning tagada kasutaja sünnikuupäeva sisestamise korrektses formaadis. Sarnane valideerimine käib kõikides rakenduses olevates vormides (vt. Lisa 3 – Turniiri lisamise ja muutmise vorm koodinäide valideerimisreeglite kohta).



Joonis 15: Mängija lisamise vaate valideerimisvead.

Testimise käigus testiti ka rakenduse turvalisust. Kuna rakendusse on ligipääs ainult autenditud kasutajatel, siis testiti rakenduses olevaid API-otsi Postman-iga. See võimaldas autoril testida ka rakenduse selliseid osi, mille testimine läbi kasutajaliidese ei ole võimalik. Joonis 16-l on näha Postmani päringu vastust, kui pärida meeskondade nimekirja API-lt andmeid vigase või puuduva autentimisvõtmega. Server tagastab veakoodi 403 ja teate, et ligipääs puudub.



Joonis 16: Vigase autentimisvõtmega meeskonna nimekirja päring.

4.2 Testimise tulemused

Kogu põhifunktsionaalsus, mis oli analüüsitud, töötas, kuid testimise käigus avastati puuduolevaid valideerimisreegleid ja rakenduse kitsaskohti. Üheks avastatud veaks oli mängijate reitingu väli. Algselt oli tehtud iga mängija jaoks üks reitingu väli, kuid testimise käigus tuli välja, et mängijatel peavad olema eraldi reitingud 64 ruuduse kabelaua ja 100 ruuduse kabelaua jaoks. Teine probleemne koht, mis ilmnes, oli mängijate paarimine paaritu arvu osalejate korral, mil rakendus ei paarinud osalejaid korrektselt ning võis ühe osaleja paarida vabana mitu vooru järjest.

Testimise käigus tekkinud probleemid olid autori hinnangul kergesti lahendatavad ning antud kitsaskohad eemaldati.

5 Tulevikuarendused

Selle lõputöö ulatuseks oli MVP rakenduse tegemine. Lõputöö raames tehtud rakendus pakub kasutajatele lihtsat kasutajaliidest turniiride läbiviimiseks, kuid rakendus ei ole autori arvates kindlasti valmis. Autor plaanib antud rakendust tulevikus edasi arendada. Selle tarbeks on autor analüüsinud tuleviku arengusuundi ning funktsionaalsuseid, mida on plaanis hiljem rakendusele lisada.

5.1 Suhtlus väliste keskkondadega

Esimeseks tulevikuarenduseks plaanib autor teha mängijate impordi Eesti kaberegistrist, mis impordiks süsteemi kaberegistris olevad mängijad ja nende andmed. Import on plaanitud üles ehitada serveris konsoolist käivitatavana, mis käiks iga kindla aja tagant cron'i abil. Kirjeldatud funktsionaalsuse implementeerimiseks on vaja kokkulepet Eesti Kabeliiduga, et nende poolt pakutaks väljundit vajalike andmetega. Samuti oleks vaja turniiride tulemusi sünkroniseerida Nõmme kabeklubi ja suuremate võistluste puhul ka Eesti kabeliidu lehega.

5.2 Mitmekeelsus

Rakendus on hetkel avalik ainult Eesti keeles. See piirab rakenduse kasutamist kohtunikel ning ka pealtvaatajatel, kes soovivad saada infot. Rakenduse mitmekeelsuse jaoks on põhjad tehtud nii serveri kui ka kliendipoolses koodis. Vaja on rakenduses olevad tekstid ümber teha, et kasutataks tõlkemeetodeid. Kui tekstid käivad läbi tõlkefunktsioonide, siis on võimalik tõlkida igat teksti soovitud keelde. Lõpuks oleks vaja arendada rakendusele keelevahetuse funktsionaalsus.

5.3 Tulemuste jagamine

Peale turniiride läbiviimist oleks vaja jagada turniiri tulemusi. Hetkel selline funktsioon puudub, kuid tulevikus võiks rakendus võimaldada nii pealtvaatajatel kui ka kohtunikel alla laadida lõpetatud turniiride tulemused PDF (tekstidokumendi tüüp) või Exceli tabeli kujul. Kindlasti võiks olla ka võimalus tulemused saata endale emailile.

5.4 Kasutajatebaasi laiendamine

Hetkel on rakendus mõeldud kasutamiseks ainult Nõmme kabeklubile. Selleks, et rakendust saaks kasutada ka teised Eesti (ja tulevikus ka välisriikide) kabeklubid, on vaja välja mõelda ja arendada rakendusele privileegidehaldus. Privileegid on vajalikud, et iga kabeklubi saaks hallata ainult enda klubiga seotud andmeid.

5.5 Mobiilirakendus

Üheks suuremaks tulevikuarenduseks on plaan luua mobiilirakendus, mis oleks suunatud just pealtvaatajatele ning mängijatele. Pealtvaatajad saaksid turniiridest parema ülevaate ning reaalajas teavitusi, kui käimasolevate turniiride tulemused uuenevad. Rakenduse põhirõhk oleks mängijatel, kes saaksid sisse logida ning näha endaga seotud andmeid. Antud rakendus pakuks mängijale ülevaadet tema tulemustest ja võimaldaks hallata mängijaga seotud andmeid.

5.6 Turniiride analüüs

Autori viimaseks tulevikuplaaniks on hetkel veel ideetasandil olev turniiride analüüs, mis oleks suunatud mängijatele ja treeneritele. Analüüsi üks osa võiks sisaldada endas ülevaadet, kuidas mängija on ajalooliselt mänginud samade vastaste vastu. Samuti võiks rakendus sisaldada kõiki mängus tehtud käike, mis annab mängijale võimaluse analüüsida rahulikult tehtud otsuseid ning arendada ennast kabemängijana.

6 Kokkuvõte

Lõputöö eesmärgiks oli realiseerida veebirakendus, mis võimaldaks kabekohtunikel korraldada ja läbi viia kabeturniire. Töös kirjeldati rakenduse funktsionaalseid ja mittefunktsionaalseid nõudeid, mis kõik said rakenduses ka implementeeritud. Kasutaja saab valida kahe turniiritüübi vahel: Šveitsi süsteem ning ringsüsteem. Rakendus võimaldab süsteemi lisada mängijaid ja meeskondi ning registreerida kohtunikke. Kohtunikud saavad luua ja läbi viia turniire ning jälgida jooksvalt seisu.

Rakendus põhineb SPA disainimustril, millest serveripoolne kood on kirjutatud PHP-s ja kliendipoolne kood TypeScriptis. Lisaks kasutati Yii ja Angulari raamistikke. Rakenduses käib suhtlus läbi REST API ning selle turvalisus on tagatud läbi JWT võtmete.

Lõputöö raames arendatud rakendus on esmane prototüüp ning selle arendamist kavatseb autor tulevikus jätkata. Kirjutatud kood on kättesaadav avalikult Git-i salvest (vt. Lisa 1 – Github'i link). Selle praegune ülesehitus võimaldab lihtsaid edasiarendusi ning potentsiaalselt ka mobiilirakenduse loomist. Kuna antud rakendusel on reaalselt huvitatud osapooled, kes sooviksid selle tegelikku implementeerimist, on autor motiveeritud seda projekti jätkama.

Antud bakalaureusetöö teema oli autori jaoks üsna tundmatu. Kui enne töö kirjutamist arvas autor, et kabe on võrreldes malega imelihtne mäng, siis kirjutamise käigus selgus, et tegelikult see nii ei ole. Tänu rakenduse arendamisele tutvus autor kabemaailma ning selle reeglitega.

Kasutatud kirjandus

[1] MDN web docs. JavaScript. https://developer.mozilla.org/en-US/docs/Web/JavaScript [18.05.2019]

[2] Structured Query Language (SQL). Saadaval: https://docs.microsoft.com/en-us/sql/odbc/reference/structured-query-language-sql?view=sql-server-2017 [18.05.2019]

[3] Kullamäe, K. Kabekohtunike süsteemi veebirakendus. Diplomitöö. Tallinna Tehnikaülikool. 2018. [18.05.2019]

[4] http://www.chessarbiter.com [18.05.2019]

[5] http://www.swissperfect.com/ [18.05.2019]

[6] Eriksson, U. Functional vs Non Functional Requirements. 2012. Saadaval: https://reqtest.com/requirements-blog/functional-vs-non-functional-requirements/?fbclid=IwAR06aycjCEnhE1GysORboBHD4HKMJFdaDEu1pCU-meXyIZ0yexXHz7rO0mU [18.05.2019]

[7] FIDE Šveitsi süsteemi reeglid. Saadaval: http://www.maleliit.ee/varia/reeglid/FIDE_shveits.pdf [18.05.2019]

[8] Ringsüsteemis tabelite lugemine. Saadaval: http://www.maleliit.ee/varia/reeglid/ringsysteem.pdf [18.05.2019]

[9] Kabekoodeks. Saadaval: http://www.kabeliit.ee/index.php/Kabekoodeks/Lisa3 [18.05.2019]

[10] Melnik, I. Single page application (SPA) vs Multi page Application (MPA): Pros and Cons. 2018. Saadaval: https://merehead.com/blog/single-page-application-vs-multi-page-application/ [18.05.2019]

[11] Skólski, P. Single-page application vs. multiple-page application. 2016. Saadaval: https://neoteric.eu/blog/single-page-application-vs-multiple-page-application/? [18.05.2019]

[12] Yii framework. Saadaval: https://www.yiiframework.com/ [18.05.2019]

[13] Angular. Saadaval: https://angular.io/ [18.05.2019]

[14] Git. Saadaval: https://git-scm.com/ [18.05.2019]

[15] Start Bootstrap Admin 2. Saadaval: https://startbootstrap.com/themes/sb-admin-2/ [18.05.2019]

[16] Introduction to JSON Web Tokens. https://jwt.io/introduction/ [18.05.2019]

[17] Doglio, F. How to secure a REST API using JWT. 2019. https://blog.logrocket.com/how-to-secure-a-rest-api-using-jwt-7efd83e71432 [18.05.2019]

[18] TypeScript Language Specification. GitHub repository. 2016. https://github.com/Microsoft/TypeScript/blob/master/doc/spec.md [18.05.2019]

[19] W3Techs. Web Technology Surveys. Usage of server.side programming languages for websites. https://w3techs.com/technologies/overview/programming_language/all [18.05.2019]

[20] What can PHP do? Saadaval: https://www.php.net/manual/en/intro-whatcando.php [18.05.2019]

[21] PHP IDE PhpStorm. Saadaval: https://www.jetbrains.com/phpstorm/ [18.05.2019]

[22] Git. https://et.wikipedia.org/wiki/Git [18.05.2019]

[23] Adminer andmebaasihaldur. Saadaval: https://www.adminer.org/ [18.05.2019]

[24] Yii2 Gii Extension. GitHub repository. Saadaval.: https://github.com/yiisoft/yii2-gii [18.05.2019]

[25] Postman. Saadaval: https://www.getpostman.com/ [18.05.2019]

Lisa 1 – Github'i link

https://github.com/martinlillemets/checkers

Lisa 2 – Ringsüsteemi paarimise kood

```
    /**
     * @param Player[] $players
     * @return array|Match[]
     */
    public function generateMatches(array $players)
    {
        // shortcut
        $currentMatch = $this->currentMatch;
        // if odd number of players, add placeholder player
        if ($players % 2 !== 0) {
            $players[] = new Player();
        }
        $playerCount = count($players);

        // find index that plays with last player during this match

        if ($currentMatch % 2 !== 0) {
            $lastPlayerOpponentSeq = (int) round($currentMatch / 2);
        } else {
            // we get index by subtracting second square middle index from
total player count
            $lastPlayerOpponentSeq = (int) ($playerCount -
(round(($playerCount - $currentMatch) / 2)));
        }
        $matches = [];
        // this holds correct array index
        $opponentIndex = $currentMatch - 1;
        $firstSquare = true;
        // holds player seq that are used
        $matchedIndexes = [];
        foreach ($players as $index => $player) {
            // all matched, break out
            if (count($matchedIndexes) === $playerCount) {
                break;
            }

            // key holds current player seq
```

```
            $key = $index + 1;
            $match = $this->generateMatch();
            // we left the first square logic
            if ($firstSquare && $key > $currentMatch) {
                $firstSquare = false;
                // we find next opponent index when match matching second
square players
                // ex: 8 - 1 = 7 meaning we get here when $key is 4 and 4
should be matched with 7
                // -1 in the end because of last player + array indexes
starting from 0
                $opponentIndex = $playerCount - 2;
            }

            // if both players already matched, continue
            if(in_array($index, $matchedIndexes) && in_array($opponentIndex,
$matchedIndexes)) {
                $opponentIndex += $firstSquare ? -1 : 1;
                continue;
            }

            if ($key === $lastPlayerOpponentSeq) {
                $match->player_1_id = $player->id;
                $match->player_1_color = ColorEnum::WHITE;
                $match->player_2_id = end($players)->id;
                $match->player_2_color = ColorEnum::BLACK;
                $matchedIndexes[] = key($players);
            } else {
                $match->player_1_id = $player->id;
                $match->player_2_id = $players[$opponentIndex]->id;
                // get player seq sum
                $sum = $key + $opponentIndex + 1;
                $match->player_1_color = $this->getLowerSeqColor($sum);
                $match->player_2_color = $this->getHigherSeqColor($sum);
                $matchedIndexes[] = $opponentIndex;
            }
            $matchedIndexes[] = $index;

            $opponentIndex--;
            $matches[] = $match;
        }
        return $matches;
    }
```

```
/**
 * @param int $sum
 * @return string
 */
protected function getLowerSeqColor(int $sum)
{
    return $sum % 2 === 0 ? ColorEnum::BLACK : ColorEnum::WHITE;
}

/**
 * @param int $sum
 * @return string
 */
protected function getHigherSeqColor(int $sum)
{
    return $sum % 2 === 0 ? ColorEnum::WHITE : ColorEnum::BLACK;
}
```

Lisa 3 – Turniiri lisamise ja muutmise vorm

```
<?php

namespace checkers\tournament\forms;

use checkers\api\components\LoadableModelTrait;
use checkers\tournament\enums\BoardSizeEnum;
use checkers\tournament\enums\PointsSystemEnum;
use checkers\tournament\enums\RuleTypeEnum;
use checkers\tournament\enums\TournamentStatusEnum;
use checkers\tournament\enums\TournamentTypeEnum;
use checkers\tournament\models\Tournament;
use checkers\user\models\User;
use yii\base\Model;

/**
 * Class TournamentForm
 * @package checkers\tournament\forms
 *
 * @author Martin Lillemets <martinlillemets@gmail.com>
 */
class TournamentForm extends Model
{
    use LoadableModelTrait;

    /**
     * @var string
     */
    public $name;

    /**
     * @var string
     */
    public $location;

    /**
     * @var string
     */
```

```
public $description;

/**
 * @var string
 */
public $judgeId;

/**
 * @var string
 */
public $type;

/**
 * @var string
 */
public $status;

/**
 * @var int
 */
public $boardSize;

/**
 * @var string
 */
public $ruleType;

/**
 * @var string
 */
public $pointsSystem;

/**
 * @var int
 */
public $teamSize;

/**
 * @var int
 */
public $totalMatchCount;

/**
```

```
     * @var int
     */
    public $currentMatchCount;

    /**
     * @var int
     */
    public $extraTime;

    /**
     * @var string
     */
    public $startTime;

    /**
     * @var bool
     */
    public $notationRequired;

    /**
     * @var Tournament Set if we are in an update scenario
     */
    protected $existingTournament;

    /**
     * TournamentForm constructor.
     * @param Tournament|null $tournament
     */
    public function __construct(Tournament $tournament = null)
    {
        $this->existingTournament = $tournament;
        parent::__construct();
    }

    /**
     * @return array
     */
    public function rules()
    {
        $example = (new \DateTime())->format('d.m.Y 12:00');
        return [
            [['name', 'location', 'judgeId', 'type', 'boardSize', 'ruleType',
'startTime'], 'required'],
```

```
            [['boardSize', 'totalMatchCount', 'currentMatchCount',
'extraTime', 'judgeId', 'teamSize'], 'integer'],
            ['totalMatchCount', 'required', 'when' => function (self $model)
{
                return $model->ruleType === RuleTypeEnum::SWISS;
            }],
            ['teamSize', 'required', 'when' => function (self $model) {
                return $model->type === TournamentTypeEnum::TEAM;
            }],
            [['name', 'location', 'status', 'type', 'ruleType',
'pointsSystem'], 'string', 'max' => 191],
            ['startTime', 'date', 'format' => 'php:d.m.Y H:i', 'message' =>
"{attribute} peab olema {$example} formaadis"],
            ['startTime', 'validateStartTime'],
            ['notationRequired', 'boolean'],
            ['description', 'string'],
            ['boardSize', 'in', 'range' => BoardSizeEnum::getList()],
            ['pointsSystem', 'in', 'range' => PointsSystemEnum::getList()],
            ['ruleType', 'in', 'range' => RuleTypeEnum::getList()],
            ['status', 'in', 'range' => TournamentStatusEnum::getList()],
            ['type', 'in', 'range' => TournamentTypeEnum::getList()],
            ['judgeId', 'exist', 'targetClass' => User::class,
'targetAttribute' => 'id']
        ];
    }

    public function validateStartTime(string $attribute)
    {
        if (strtotime($this->{$attribute}) < time()) {
            $this->addError($attribute, 'Algusaeg ei tohi olla minevikus');
        }
    }
}
```