TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Erik Martin Murde 213278IADB

Veebirakendus rütmimänguturniiride organiseerimiseks

Bakalaureusetöö

Juhendaja: Meelis Antoi
Magistrikraad

Tallinn 2025

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: Erik Martin Murde

02.01.2025

Annotatsioon

Käesoleva lõputöö eesmärgiks on luua veebirakenduse prototüüp rütmimängu osu!mania turniiride organiseerimiseks, läbiviimiseks ja jälgimiseks. Loodav rakendus viib siiani mitme lahenduse kasutamist nõudnud funktsionaalsused läbi ühtse kasutajaliidese kokku üheks tervikuks, lihtsustades seeläbi osu!mania turniiride organiseerimise ja läbiviimise protsessi.

Arendusprotsessi käigus luuakse veebirakenduse prototüüp, mis võimaldab organiseerida ja läbi viia lihtsama ülesehitusega osu!mania turniire ning jälgida nii hetkel käivate kui ka varem toimunud turniiride toimumise kulgu. Rakendus on eelkõige mõeldud vähemate ressursside ja väiksema kogemustega organiseerijatele, mistõttu on suur rõhk pandud kasutajasõbralikkusele ning turniiri protsesside automatiseerimisele.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 55 leheküljel, 7 peatükki, 9 joonist, 4 tabelit.

Abstract

Web Application for Organizing Rhythm Game Tournaments

The aim of this thesis is to create a web application prototype for organizing, conducting and monitoring tournaments of the rhythm game osu!mania. The application brings together functionalities that have so far required the use of several different solutions through a single user interface into one whole, thereby simplifying the process of organizing and conducting osu!mania tournaments.

During the development process, a web application prototype is created, which allows organizing and conducting simple osu!mania tournaments as well as monitoring both current and previously held tournaments. The application is primarily intended for organizers with fewer resources and less experience, which is why great emphasis is placed on user-friendliness and the automation of tournament processes.

The thesis is in Estonian and contains 55 pages of text, 7 chapters, 9 figures, 4 tables.

Lühendite ja mõistete sõnastik

API	*Application Programming Interface*, rakendustarkvara liides
Authorization Code Grant	OAuth volituse tüüp, mis nõuab kasutaja autentimist
Client Credentials Grant	OAuth volituse tüüp, mis nõuab klientrakenduse autentimist
Client ID	OAuth rakenduse registreerimisel loodav identifiakaator
Client Secret	OAuth rakenduse registreerimisel loodav räsi
CSS	*Cascading Style Sheets*, veebilehtede kujunduskeel
DTO	*Data Transfer Object*, andmeedastusobjekt
HTML	*Hypertext Markup Language*, veebilehtede märgenduskeel
JPA	*Jakarta Persistence API*, standard Java objektide ja andmebaasi vaheliste suhete määramiseks
JSON	*JavaScript Object Notation*, JavaScript programmeerimiskeele objektidest inspireeritud andmevorming
JSX	*JavaScript XML*, React raamistikus kasutatav HTML-laadne süntaks komponentide kirjeldamiseks
Lobby	Turniiri kvalifikatsioonietappi loodud ajapilu, kuhu saab registreeruda kindel arv mängijaid või meeskondi
LocalStorage	Veebibrauseris sisalduv andmehoidla võti-väärtus paaride salvestamiseks
NoSQL	*Not only SQL*, termin mitterleatsiooniliste andmebaaside kategoriseerimiseks
Online	Võrgukeskkonnas olev
SQL	*Structured Query Language*, relatsiooniliste andmebaaside päringute loomise keel
VSRG	*Vertical Scrolling Rhythm Game*, vertikaalselt keriva mänguväljaga rütmimäng

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Osu! on populaarne tasuta 2D ehk kahedimensiooniline rütmimäng, mille lõi 2007. aastal Dean „Peppy“ Herbert ning millel on miljoneid registreeritud kasutajaid [1]. Osu! koosneb neljast mängurežiimist, millest populaarseim on osu!standard ning antud lõputöös käsitletav on osu!mania [2]. Tegu on VSRG tüüpi rütmimänguga, mis lisandus osu! mängule 2012. aastal. Osu!maniat mängitakse tavaliselt klaviatuuriga ning mängu eesmärgiks on saavutada valitud tasemel võimalikult suur skoor, milleks on vaja tabada mänguväljale langevaid noote nii täpselt kui võimalik.

Tänu oma olemusele sobib osu!mania hästi võistluste korraldamiseks. Enamik antud mängu võistlustest on kogukonna poolt läbiviidavad turniirid, milles võib osaleda paarkümmend kuni paarsada mängijat. Turniirid algavad perioodiga, kus registreeritakse mängijaid ning kogutakse personaliliikmeid. Turniiride läbiviimine toimub etappide kaupa. Igal etapil on kindel mängitavate tasemete grupp ning eraldi matšide ajakava.

Hetkel on turniiride organiseerimine ja läbiviimine aeganõudev, järsu õppimiskõveraga ning ebamugav tegevus, kuna puudub lihtne ja mugav viis turniiri andmete töötlemiseks ning matšide läbiviimiseks. Enamus andmeid töödeldakse kas käsitsi või kasutatakse selleks tabelarvutusprogramme. Lisaks eelmainitule on probleemideks ka mängijate poolt turniiride leidmine ja jälgimine ning andmete arhiveerimine. Kõik see võib eemale peletada potensiaalseid uusi organiseerijaid, mängijaid ja vaatajaid.

Käesoleva lõputöö eesmärgiks on luua veebirakendus, mis aitaks lihtsustada ja automatiseerida turniiride organiseerimist, vajalike andmete töötlemist, matšide läbiviimist ning statistika näitamist. Valmiv lahendus vähendaks turniiride haldamisega seotud erinevatele tegevustele kuluvat aega ning muudaks organiseerimise kättesaadavamaks suuremale hulgale inimestele. Eriti kasulik oleks lahendus autori hinnangul vähemate ressursside ja väiksema kogemustega organiseerijatele.

2 Ülesandepüstitus

Käesolevas peatükis tutvustatakse põhjalikumalt käsitletava teema tausta, kirjeldatakse autori poolt lahendatavat probleemi, püstitatakse lõputöö raames loodava lahenduse eesmärk ning peatüki lõpus selgitatakse, millise metoodika alusel plaanitakse eelnevalt püstitatud eesmärgini jõuda.

2.1 Taust

Rütmimängud on muusikapõhised mängud, mille mänguviis on tihedalt seotud rütmi ja muusikaga ning kus kasutaja sisendid vastavad mingil moel taustal mängiva muusika rütmile. Populaarseimad rütmimängud on ajalooliselt olnud inspireeritud kitarrist, klaverist või mõnest muust muusikainstrumendist, kuid loodud on ka mitmeid muid mänguvariante [3].

Mitmed allikad on täheldanud rütmimängude kasulikkust nii vaimse kui füüsilise tervise edendamisel. Teadusajakirjas „Frontiers in Public Health“ 2022. aastal läbiviidud uuringus leiti, et muusikal põhinevate videomängude harjutamine võib leevendada depressiooni, ärevuse ja stressi negatiivseid mõjusid [4]. Lisaks on inimesed märganud rütmimängudest tulenevat positiivset mõju nende käte-silmade koordinatsioonile, reageerimiskiirusele ja rütmitundlikkusele [5]. Füüsilist tegevust nõudvate rütmimängude puhul, näiteks virtuaalreaalsuses mängitav Beat Saber, on ka tähele pandud positiivset mõju mängija füüsilisele tervisele [6].

Käesolevas lõputöös käsitletakse rütmimängu nimega osu!mania, mis on VSRG tüüpi rütmimäng ning üks osu! neljast mängurežiimist. Mänguväli on jaotatud veergudeks ning igale veerule vastab klaviatuuriklahv. Mängu jooksul langevad igale veerule noodid. Eesmärk on lüüa õiget klahvi nii täpselt kui võimalik ja saavutada seeläbi suurim skoor. Osu!maniat mängitakse erinevate veergude arvuga, millest populaarseimad on 4, 6 ja 7 veeruga variandid. Suurema veergude arvude puhul on suurem fookus sõrmede koordinatsioonil ja nootide mustrite lugemisel ning väiksema arvu puhul füüsilisel kiirusel ja vastupidavusel.

Osu!mania turniire on korraldatud mängurežiimi algusaegadest saadik. Suurimad neist on maailmakarikad, mida korraldab osu! meeskond ning kus mängivad tihti maailma parimad mängijad. Esimene osu!mania maailmakarikas toimus 2014. aastal [7] ning seda on korraldatud igal järgneval aastal. Maailmakarikatele on looud osu! kodulehel eraldi turniiride alamleht, kust on võimalik neid lihtsasti jälgida [8]. Enamik osu!mania turniire on aga korraldatud mängu kogukonna poolt, on väiksemad ning ei ole turniiride alamlehel esile toodud.

Turniiride organiseerimise käigus tekib suurel hulgal töötlust nõudvaid andmeid. Vaja on töödelda registreerunud mängijaid ja meeskondi, personali liikmeid, mängitavate tasemete gruppe ja nende organiseerimisega seotud personaliliikmete poolt soovitatud tasemeid ning palju muud. Lisaks on vaja hallata andmeid matšide läbiviimise käigus, kus on sisestada ja uuendada mängijate skoore, valitud tasemeid ja muid matši jooksul vajaminevaid andmeid.

2.2 Probleemi kirjeldus

Hetkel töödeldakse enamik turniiride andmetest, näiteks mängitavate tasemete nimekirjad, läbi erinevate tabelarvutusprogrammide [9]. Tegu on aeganõudva tegevusega, millele kulutatakse palju väärtuslikku aega. Mängijate registreerimine ja personalitaotluste saatmine toimub läbi käsitsi loodud vormide. Andmete liigutamine vormidest tabelitesse on jällegi keeruline. Palju on loodud erinevaid skripte, mis tegevust automatiseeriksid, kuid nende leidmine ja kasutamine on problemaatiline inimestele, kellel puuduvad teadmised programmeerimisest ja skriptide kasutamisest.

Kasutatavate meetodite õppimis- ja kasutamiskeerukusest tulenevalt on kogukonnas suures nõudluses inimesed, kellel on olemas vajalikud teadmised ja kogemused. Neid on aga väga piiratud hulgal ning tihti pole neil piisavalt aega organiseerijate aitamiseks, mistõttu peavad paljud turniiride korraldajad õppima ise selgeks vajalikud oskused tabelarvutusprogrammide kasutamiseks.

Sarnasel viisil toimub ka matšide käigus andmete töötlemine. Matšide läbiviimisel kasutatakse spetsiaalseid kohtunikele loodud lehti [10]. Olenevalt nende formaadist võib olla vajalik kohtunikel kõik mängijate saavutatud skoorid ja muu info käsitsi sisestada, mis võib osutuda suure mängijate arvu korral väga mahukaks. Ebamugav on ka matšide

läbiviimiseks vajalike andmete, nagu näiteks mängitavate tasemete nimekirja, importimine. Tihti valmistatakse vajalikud lehed kohtunikele ette, kuid seda ei tehta alati ning kogemusteta kohtunikele võib nende efektiivne kasutamine endiselt keeruliseks ülesandeks osutuda.

Lõpetuseks saab välja tuua ka asjaolu, et turniiride reklaamimine toimub peamiselt läbi osu! turniiride foorumi [11] või sotsiaalmeedia, kus võivad nad kiiresti teiste postituste alla mattuda. Lisaks puudub foorumi lehel otsing ja võimalus turniire mängurežiimi või muude parameetrite alusel filtreerida, mis teeb turniiride leidmise mängijatele ja vaatajatele veelgi keerulisemaks.

2.3 Lahenduse metoodika

Käesoleva lõputöö käigus analüüsitakse esmalt käsitletavat probleemi ning uuritakse erinevaid olemasolevaid ja hetkel rakendatud lahendusi, sealhulgas nende tugevaid ja nõrku külgi. Probleemi analüüsi osa lõpus pannakse lähtuvalt lahenduse eesmärgist ja analüüsi tulemustest paika loodava rakenduse skoop, mis võimaldaks eesmärgi realiseerida. Sealjuures peab arvestama ka lõputöö kirjutamiseks antud aja piiratusega.

Lahenduse analüüsi osa alguses määratakse skoobist lähtudes rakendusele funktsionaalsed ja mittefunktsionaalsed nõuded, mis kirjeldavad skoobi realiseerimiseks vajalikud funktsionaalsused. Nõudeid kirjeldatakse kasutajalugude kujul, mis on jaotatud erinevate persoonade vahel. Siinkohal toimub ka osu! API kasutuse analüüs, mille käigus uuritakse, mis tüüpi andmeid on API kaudu võimalik pärida ning milliste piirangutega peab API kasutamisel arvestama.

Lahenduse analüüsi osa jätkub tehnoloogiate valikuga, mille käigus uuritakse erinevaid tehnoloogiaid ja arendusvahendeid ning valitakse koos põhjendusega välja, millised neist sobivad eesmärgi realiseerimiseks kõige paremini. Analüüsi osa lõppeb rakenduse arhitektuuri ja kasutajaliidese kujunduse kirjeldusega, kus tuuakse välja lahendused, mis on autori hinnangul rakenduse realiseerimiseks kõige sobivamad.

Järgneb rakenduse teostuse osa, mis käsitleb nii andmebaasi andmemudeli kirjeldust kui ka esi- ja tagarakenduse arendust. Siinkohal tuuakse välja kasutatud struktuur, lahendusvõtted ning kirjeldatakse rakenduse osu! API-ga integreerimise protsessi. Teostuse osa lõppeb rakenduse testimisprotsessi kirjeldusega.

Lõputöö viimases osas antakse ülevaade valminud rakenduse kõige olulisematest funktsionaalsustest ning hinnatakse nende vastavust analüüsi käigus püstitatud nõuetele. Viimasena pakutakse välja erinevaid võimalikke viise rakenduse edasiarenduseks väljaspool käesoleva lõputöö skoopi.

3 Probleemi analüüs ja rakenduse skoop

Käesolevas peatükis analüüsitakse osu!mania turniiride ülesehitust nende toimimisest selgema ülevaate saamiseks, mis on aluseks rakenduse skoobi kirjeldamisele. Lisaks uuritakse ja hinnatakse erinevaid eksisteerivaid ning hetkel kasutuses olevaid lahendusi. Peatükk lõppeb lõputöö raames loodava rakenduse skoobi määramisega.

3.1 Osu!mania turniiride ülesehitus

Järgnevalt analüüsitakse osu!mania turniiride ülesehitust, läbiviimise protsessi ning turniiridega seotud osapoolte rolle ja ülesandeid. Tegu on autori hinnangul olulise sammuga, kuna turniiridest detailse ülevaate saamine võimaldab selgemini kirjeldada loodava rakenduse nõudeid ning aitab kaasa rakenduse skoobi määramisele. Informatsioon turniiride kohta tuleneb autori teadmistest ja osu! turniiride kogukonna poolt loodud materjalidest.

3.1.1 Turniiri formaat ja osalised

Turniiri loomise algfaasis peab organiseerija otsustama, mis tüüpi turniiriga on tegemist. Selleks tuleb turniiri eesmärgid läbi mõelda ja valida mitmete erinevate parameetrite vahel [12, pp. 30-32]. Turniiri keerukusele ei ole piiranguid, kuid enamik piirduvad lihtsa ja arusaadava ülesehitusega, millega lõputöö autor skoobi määramisel arvestab.

Oluline personal kogutakse tavaliselt enne turniiri avalikustamist, kuid juhul kui mõne personalirolli liikmetest jääb puudu, võetakse vastu ka personalitaotlusi [12, p. 29]. Turniiri avalikustamisele järgneb tavaliselt periood, kus mängijad ja meeskonnad saavad ennast registreerida. Enne turniiri algust toimub ka mängijate sõelumine, kus mittesobivad registreerunud mängijad turniirilt eemaldatakse [13, p. 4].

3.1.2 Turniiri läbiviimine

Turniirid koosnevad enamjaolt etappidest. Igale etapile määratakse kindel arv mängitavaid tasemeid läbi protsessi, mille nimi on *mappooling*. Tasemed on tavaliselt jaotatud kategooriatesse. Osu!mania puhul jagatakse tasemed sõltuvalt nendes leiduvatest

noodimustritest. Igasse etappi (v.a kvalifikatsioon) kuulub ka viigimurdja. *Mappooling* on üks olulisemaid protsesse turniiris, kuna „halbade“ tasemete etappi paigutamine võib turniiri mainet ja mängijate kogemust negatiivselt mõjutada. Tegu on üsnagi keerulise ülesandega, millele tuleks suurt tähelepanu pöörata [14].

Enamikel turniiridel eelneb muudele etappidele kvalifikatsioonietapp, mille eesmärk on piirata turniiril mängivate mängijate arvu. Tavaliselt luuakse selle tarvis *lobby*-de nimekiri, millele mängijad end registeerida saavad. Kvalifikatsioonietapi käigus mängitakse läbi kõik selles olevad tasemed ning kvalifikatsiooni tulemuste alusel moodustatakse mängijate pingerida ehk *seeding*. Igale mängijale määratakse seejärel sõltuvalt nende kohast pingereal *seed* ehk seeme, mida kasutatakse kvalifikatsioonile järgneva etapi matšide osaliste valimisel [13, pp. 4-5].

Igale järgnevale etapile moodustatakse vastav matšide ajakava. Tavaliselt toimub nädalas üks etapp, misjärel langeb kindel arv mängijaid turniirilt välja. Protsessi jätkatakse kuni järele jääb vaid kaks mängijat, misjärel toimub finaal. Matše viivad läbi kohtunikud ning tihti kasutatakse andmete haldamiseks spetsiaalseid kohtunikele loodud lehti, mis lihtsustavad matšide läbiviimist [10].

3.1.3 Personaliliikmete rollid ja ülesanded

Igal turniiril võivad personalirollid ja ülesanded olla mõnevõrra erinevad ning tihti esineb ka juhte, kus ühel personaliliikmel on mitu rolli. Üldises pildis on aga välja kujunenud järgnevad rollid:

- Organiseerija – Turniiri looja. Otsustab turniiri ülesehituse, hangib personali ning vastutab reklaamimise, auhindade ja muude küsimuste eest. Organiseerijaid võib olla ka mitu.
- Admin – Tehniline roll. Ülesandeks on turniiri andmete haldamine ja erinevate osapooltega suhtlemine. Tihti täidavad rolli organiseerijad [12, pp. 23-25].
- *Mappooler* – Rolli ülesandeks on hinnata loodud ja soovitatud tasemete sobivust ning viia igas tasemes läbi *mappooling* protsessi.
- *Mapper* – Rolli ülesandeks on luua turniiril mängitavaid tasemeid. Tavaliselt teevad tihedat koostööd *playtester* ja *mappooler* rollidega.

- *Playtester* – Rolli ülesandeks on testida *mapper*-ite loodud tasemeid ja anda vajalikku tagasisidet tasemete erinevate aspektide osas [13, pp. 45-46].

- Kohtunik – Rolli ülesandeks on läbi viia matše ja kvalifikatsioonietapi *lobby*-sid ning panna kirja nende tulemusi [12, pp. 4-7].

- Striimer – Rolli ülesandeks on turniiriga seotud otseülekannete haldamine. Tihti kasutatakse sellest spetsiaalset tarkvara, mille funktsionaalsus jääb väljapoole lõputöö skoopi [13, pp. 9-32].

- Kommentaator – Rolli ülesandeks on otseülekandes jooksvate matšide ja muude turniiriga seotud otseülekannete kommenteerimine ja seeläbi vaatajate kaasahaaramine [13, pp. 32-34].

- *Spreadsheeter* – Rolli ülesandeks on hallata turniiriga seotud andmeid. Tavaliselt tehakse seda läbi tabelarvutuskeskkonna. Väiksemate turniiride puhul täidavad tihti antud rolli organiseerijad või adminid, kuna kogenud *spreadsheeter*-eid leidub osu!mania kogukonnas piiratud arvul [12, p. 14].

- Graafiline disainer – Rolli ülesandeks on disainida turniiriga seotud graafilisi elemente. Siia kuuluvad näiteks bänner ja tabelite disain tabelarvutuskeskkonnas. Oskuste puudumise korral on soovitatav palgata professionaal [12, p. 22].

3.2 Olemasolevad lahendused

Autori poolt välja pakutavale rakendusele ei ole lõputöö kirjutamise hetkel loodud analoogseid lahendusi. Sellest tulenevalt on antud alampeatükis analüüsitud lahendusi, mida kasutatakse kas turniiride tutvustamiseks või nende organiseerimiseks ning millest on autoril võimalus võtta loodava rakenduse tarvis inspiratsiooni.

3.2.1 Osu! turniiride kodulehed

Suurematele turniiridele arendatakse mõnikord koduleht, et aidata turniiri reklaamida ning selle kohta informatsiooni jagada. Allpool analüüsitakse kahte säärast kodulehte:

- Global Taiko Showdown – Tegu on veebikeskkonnaga, mis tutvustab samanimelise seeria alla kuuluvaid osu!taiko ning nüüd ka osu!catch mängurežiimide turniire [15]. Veebikeskkonnas on võimalik turniiridele

registreerida ja näha nendel mängitavaid tasemeid, mängijate, personali ja meeskondade nimekirju, matšide ajakavasid ning statistikat. Kuna autoril on vaid tavakasutaja õigused, ei ole teada kas veebirakendus sisaldab lisaks eelmainitule veel funktsionaalsusi. Rakendust kasutatakse vaid kindla turniiriseeria raames ning see ei võimalda mistahes uute turniiride loomist ega haldamist. Lisaks ei võta lahendus arvesse osu!mania mängurežiimi eripärasid.

- The Perennial – Tegu on veebikeskkonnaga, mis on loodud samanimelise osu!standard mängurežiimi turniiri tutvustamiseks [16]. Veebikeskkond võimaldab registreerimist ning turniiri reeglite, meeskondade ja personali, mängitavate tasemete ning ajakava vaatamist, kuid ei sisalda endas statistikat. Keskkond on loodud vaid ühe kindla turniiri kohta informatsiooni näitamiseks ja mängijate registreerimiseks ning ei võimalda uute turniiride loomist ega nende haldamist.

Mõlema lahenduse puhul on tegemist eelkõige turniiride informatsiooni väljanäitava veebirakendusega, mille ainsaks autorile nähtavaks lisafunktsionaalsuseks on mängijate registreerimine. Keskkonnad ei lahenda autori hinnangul lõputöös käsitletavat probleemi, kuna turniiri või turniiriseeria jaoks veebilehe loomine nõuab arendaja palkamist ning seda saavad endale lubada vaid vähesed rohkemate ressurssidega organiseerijad. Lisaks puudub lahendustel avalikkusele kättesaadav turniiride loomise ja haldamise funktsionaalsus, mis on probleemi lahendamiseks vajalik.

3.2.2 Google Sheets

Google Sheets on osu!mania turniiride organiseerimisel laialt kasutuses olev tabelarvutuse veebirakendus, mille peamisteks funktsionaalsusteks on arvutustabelite loomine ja vormindamine, andmete analüüsimine ning nende visualiseerimine [17]. Üheks rakenduse suureks tugevuseks on kasutajate võime redigeerida tabeleid reaalajas, mis on ka Google Sheets-i üks peamisi eeliseid võrreldes Microsoft Excel-iga [18]. Turniiridel kasutatakse veebikeskkonda just tänu selle võimsusele ja reaalajas redigeerimise võimalusele.

Veebirakenduse peamiseks puuduseks on kasutamise keerukus, mis tuleneb selle suurest võimalike operatsioonide hulgast. Google Sheets sisaldab endas sadu funktsioone [19], kuid paljudel organiseerijatel puuduvad rakenduse tugevate külgede ära kasutamiseks

piisavad kogemused. Turniiride andmete haldamise lihtsustamiseks on loodud mitmeid abistavaid skripte, mis aitaksid kaasa tegevustele nagu osu! API-ga suhtlemine [20]. Paljudel organiseerijatel aga puudub skriptide kasutamise kogemus ning neile võib see keeruliseks osutuda. Puudusena saab ka välja tuua, et Google Sheets ei võimalda mängijate ega personali registreerimist vaid selleks tuleb kasutada eraldi lahendust.

3.2.3 Google Forms

Paljud osu!mania turniirid kasutavad mängijate registreerimiseks ja personalitaotluste haldamiseks Google Forms veebikeskkonda. Tegu on tasuta rakendusega, mis võimaldab mugavalt luua *online* vorme ja küsitlusi ning jälgida ja analüüsida nende vastuseid. Veebikeskkonna peamisteks tugevusteks võib lugeda kasutajaliidese lihtsust ning paindlikkust nii vormi sisu kui ka kujunduse osas [21]. Lisaks on võimalik integreerida see Google Sheets keskkonnaga, misjärel liigutatakse vormide vastused automaatselt kasutaja valitud tabelisse [22]. Viimane on ka üks peamisi põhjuseid, miks organiseerijad kipuvad registreerimiste halduseks just Google Forms keskkonda kasutama.

Kuigi veebikeskkonnal on olemas Google Sheets rakendusega integreerimise võimalus, ei lahenda see täielikult lõputöös käsitletavat probleemi ega automatiseeri lõpuni registreerimise protsessi. Vormid tuleb endiselt luua käsitsi ning Google Sheets keskkonda laekunud andmeid tuleb endiselt analüüsida ja vormindada. Lisaks ei ole integreerimine ühegi teise firma tabelarvutusrakendusega Google Forms keskkonda sisse ehitatud [21]. Antud keskkond käsitleb üldisest probleemist vaid väikest osa. Kogu probleemi lahendamiseks oleks aga vaja keskset keskkonda, mis sisaldab endas kogu vajalikku funktsionaalsust.

3.3 Loodava rakenduse skoop

Analüüsitud eksisteerivad lahendused kas ei sisalda avalikkusele kättesaadavat uute osu!mania turniiride organiseerimise ja haldamise funktsionaalsust või on nende kasutamine liigselt keerukas ja ebamugav. Sellest tulenevalt pakub autor välja uue lahenduse, mis viiks turniiride kodulehtede kasutajasõbralikkuse ja disaini koos tabelarvutus- ja vormide loomise keskkondade funktsionaalsusega kokku üheks tervikuks. Uus lahendus võtaks arvesse turniiri erinevate personalirollide ülesandeid ja vajadusi ning keskenduks tegevuste automatiseerimisele ning kasutajasõbralikkusele.

Lahenduse realiseerimiseks on mitmeid viise. Üks võimalus oleks luua eraldi arvuti -või mobiilirakendus, kuid see nõuaks allalaadimist ning piiraks inimeste hulka, kellele rakendus kättesaadav on. Seetõttu sobib autori hinnangul lahenduseks kõige paremini brauseripõhine rakendus, kuna see võimaldab rakendust kasutada paljude erinevate seadmete peal ega nõua eraldi allalaadimist [23].

Funktsionaalsuste määramisel lähtub autor turniiride ülesehituse analüüsist (vt peatükki 3.1). Rakenduse peamiseks funktsionaalsuseks on uute turniiride loomine ja seadistamine. Lisaks on oluline, et turniiridega saaksid liituda mängijad ja personal ning et neid oleks lihtne hallata. Lahendus peab võimaldama ka etappide loomist ning kvalifikatsioonile ja igale etapile matšide ajakavade määramist. Rakendus peab hõlbustama *mappooling*-u läbiviimist, võimaldades nii tasemete soovitamist kui nende lisamist etappidesse. Matšide ja kvalifikatsiooni läbiviimisel peab saama kasutada kõiki funktsionaalsusi, mida sisaldavad kohtunikele loodud lehed [10]. Kvalifikatsiooni puhul on oluline ka *seeding*-u genereerimine. Viimaks peab rakendus võimaldama turniiri statistika vaatamist. Kõikide eelmainitud funktsionaalsuste puhul on oluline silmas pidada, et need oleks kasutaja jaoks võimalikult automatiseeritud ja lihtsasti kasutatavad.

Automatiseerituse ja kasutajasõbralikkuse hõlbustamiseks kasutatakse ära osu! API poolt pakutavaid võimalusi, mille näitena saab tuua osu! kontoga sisselogimise või tasemete ja skooride informatsiooni pärimise. API kasutamine aitab kaasa paljude protsesside automatiseerimisele ja vähendab vajadust käsitsi informatsiooni sisestamise järele. Täpsem info osu! API kohta on välja toodud lahenduse analüüsi osas (vt peatükki 4.2).

Skoobi määramisel tuleb arvestada lõputöö kirjutamiseks antud aja piiratusega. Sellest tulenevalt käsitleb loodav rakendus vaid osu!mania mängurežiimi, kuna autor on sellega kõige tuttavam. Lisaks võimaldab rakendus organiseerida vaid lihtsamaid turniiriformaate, kuna lõputöö kontekstis on oluline põhifunktsionaalsuse saavutamine ning lihtsama ülesehitusega turniirid moodustavad autori hinnangul umbkaudu 80% kõikidest osu!mania turniiridest. Rakenduse suure mahu tõttu on lõputöö skoobist välja jäetud ka kohanduv kasutajaliides.

4 Lahenduse analüüs

Käesolevas peatükis määratakse rakendusele funktsionaalsed ja mittefunktsionaalsed nõuded ning valitakse kasutatavad tehnoloogiad. Lisaks analüüsitakse kuidas toimub olemasolevate andmete pärimine läbi osu! API ning kirjeldatakse loodava rakenduse arhitektuuri ja kasutajaliidese disaini põhimõtteid.

4.1 Nõuete määramine

Rakendusele funktsionaalsete ja mittefunktsionaalsete nõuete määramisel lähtus autor osu!mania turniiride ülesehituse analüüsist, olemasolevate lahenduste analüüsist ning rakenduse skoobist (vt peatükki 3). Funktsionaalsete nõuete kirjeldamiseks on kasutatud kasutajaloo formaati ning need on jaotatud kahe persoona vahel, milleks on tavakasutaja ja turniiri personal. Viimase alla kuuluvad kõik osu!mania turniiride ülesehituse analüüsil välja toodud rollid (vt peatükki 3.1.3).

4.1.1 Funktsionaalsed nõuded

Tavakasutaja persoonaga seotud funktsionaalsed nõuded:

- Tavakasutajana soovin sisse logida osu! kontoga, et ma saaksin enda osu! kasutaja andmed rakendusse üle kanda.
- Tavakasutajana soovin seadistada enda profiili, et saaksin seda personaliseerida.
- Tavakasutajana soovin näha hetkel toimuvate ja varem toimunud turniiride nimekirju, et leida endale meeldivaid turniire.
- Tavakasutajana soovin näha turniiri poodiumi, et teada saada turniiri võitja.
- Tavakasutajana soovin näha turniiri mängijate, meeskondade ja personali nimekirju, et saada ülevaade turniiril osalejatest.
- Tavakasutajana soovin näha turniiri etapi tasemete nimekirja ja matšide ajakava, et teaksin mis tasemed etapis on ja millal ma neid mängima pean.

- Tavakasutajana soovin näha turniiri statistikat nii etappide kui ka tasemete kaupa, et saada parem ülevaade turniiri tulemustest.
- Tavakasutajana soovin ennast või enda meeskonda turniirile registreerida, et saaksin turniiril osaleda.
- Tavakasutajana soovin ennast või enda meeskonda kvalifikatsiooni *lobby*-sse registreerida, et saaksin kvalifikatsioonil osaleda.
- Tavakasutajana soovin saata personalitaotlusi ning jälgida nende staatust, et saaksin turniiri läbiviimisel kaasa aidata.
- Tavakasutajana soovin näha mulle saadetud personalikutseid ning neid kas vastu võtta või tagasi lükata, et saaksin enne turniiri avalikustamist kaasa aidata.

Turniiri personali persoonaga seotud funktsionaalsed nõuded:

- Turniiri personalina soovin luua uusi turniire ja seadistada nende parameetreid, et saaksin luua enda vajadustele vastavaid turniire.
- Turniiri personalina soovin hallata personalitaotlusi ning saata kutseid kasutajatele personaliga liitumiseks, et saaksin hankida personaliliikmeid nii enne kui pärast turniiri avalikustamist.
- Turniiri personalina soovin hallata mängijaid, meeskondi ja personali, et mul oleks hea ülevaade kõikidest turniiril osalejatest.
- Turniiri personalina soovin luua uusi etappe, nende parameetreid seadistada ning lisada ja eemaldada etappidest tasemeid, et saaksin luua ja konfigureerida etappe vastavalt turniiri vajadustele.
- Turniiri personalina soovin etapis mängitavaid tasemeid soovitada ja näha teiste soovitatud tasemeid, et lihtsustada *mappooling*-u protsessi.
- Turniiri personalina soovin luua, ümberajastada ja läbi viia matše ning salvestada nende tulemusi, et oleks tagatud etappide efektiivne toimimine.
- Turniiri personalina soovin luua ja läbi viia kvalifikatsiooni *lobby*-sid ning salvestada nende tulemusi, et oleks tagatud kvalifikatsiooni efektiivne toimimine.

- Turniiri personalina soovin määrata nähtavuse taset turniirile ning selle etappidele ja statistikale, et saaksin kontrollida informatsiooni avalikustamist.

4.1.2 Mittefunktsionaalsed nõuded

Rakenduse mittefunktsionaalsed nõuded ei ole seotud ühegi kindla persoonaga:

- Rakendus peab saatma võimalikult vähe osu! API päringuid, et mitte koormata liialt osu! andmebaasi.
- Rakendus peab vigaste sisendite korral väljastama veateateid, et kasutajad saaksid selge ülevaate vigade põhjustest.
- Rakenduse kasutajaliisese kujundus peab olema sidus, et selle kasutamine oleks lihtsamini õpitav.
- Rakenduse kasutajaliidese kujundus peab olema intuitiivne, et selle kasutamine oleks võimalikult mugav.

4.2 Olemasolevate andmete pärimine läbi osu! API

Osu! andmebaasis hoitakse palju andmeid, mida oleks kasulik loodava rakenduse tarvis ära kasutada. Nende pärimiseks kasutab autor ära osu! API poolt pakutavaid võimalusi. Osu! API võimaldab osu! andmebaasiga suhtlemist veebipäringute kaudu. Antud API loodi selleks, et lihtsustada osu! teemaliste veebirakenduste arendamist ning see on tasuta avalikult kasutatav. Osu! API-l on kaks versiooni, millest ajakohasem ja loodava rakenduse poolt kasutatav on versioon 2 [24]. Osu! API on avatud lähtekoodiga ning inimestel on võimalus panustada selle arendusse [25]. Osu! API kasutamise lihtsustamiseks on loodud ka vastav dokumentatsioon [26].

Ligipääsuõiguste andmiseks kasutab osu! API *Open Authorization 2.0* ehk OAuth2 autoriseerimisprotokolli, mis võimaldab anda klientrakendusele ligipääsu kaitstud ressurssidele ilma kasutaja tundlikke andmeid väljastamata. Selle saavutamiseks autenditakse kasutaja läbi autoriseerimisserveri ning tagastatakse klientrakendusele OAuth2 *access token* ehk tõend, milles on kirjas rakendusele antud õigused [27]. Kuna kasutaja autentimine ei toimu läbi klientrakenduse enda serveri, ei jagata sellega kasutaja

salasõna ega muud tundlikku informatsiooni ning seega võib OAuth2 kasutamine tõsta rakenduse turvalisust [28].

Enamik osu! API lõpp-punktidest nõuab kehtiva OAuth2 tõendi olemasolu, mille saamiseks tuleb esmalt registreerida osu! kasutajaga seotud klientrakendus. Seda on võimalik teha olemasoleva osu! kasutaja seadistuste lehel. Kui rakendus on registreeritud, väljastatakse kasutajale *Client ID* ehk rakenduse identifikaator, ja *Client Secret* ehk räsi, mida kasutatakse rakenduse autentimiseks [26].

Enne tõendi väljastamist on vaja ka vastavat volitust. Osu! API kasutab kahte tüüpi OAuth2 volitust, milleks on *Authorization Code* volitus [29, pp. 24-31] ja *Client Credentials* volitus [29, pp. 40-42]. Nendest esimese saamine nõuab kasutajapoolset autentimist, mis toimub antud juhul osu! serveris. Edukal autentimisel tagastatakse autoriseerimiskood, mida saab kasutada OAuth2 tõendi pärimiseks. Teine volituse tüüp nõuab vaid klientrakenduse autentimist, milleks kasutatakse rakenduse registreerimisel väljastatud identifikaatorit ja räsi. Käesolevas rakenduses on kasutusel vaid *Authorization Code* volitus, kuna kõik osu! API kasutamist nõudvad funktsionaalsused eeldavad sisselogitud kasutaja olemasolu ning saavad seega ära kasutada sisselogimise käigus saadud OAuth2 tõendit.

Osu! API-le on seatud ka päringute piirmäär, milleks on 1200 päringut minutis. Osu! loojat on soovitatav aga teavitada juba siis, kui päringute arv minutis ületab 60 [26]. Autori arvamusel see vähemalt esialgu nii suureks ei kasva, kuid ettevaatusmeetmena tuleks osu! API päringute arvu siiski minimeerida. Selle saavutamiseks kasutatakse andmete pärimiseks eelisjärjekorras rakenduse enda andmebaasi ning osu! API-t kasutatakse vaid sisselogimiseks ning osu! andmebaasist vajalike andmete pärimiseks.

Osu! API võimaldab pärida kõikvõimalikke osu! mänguga seotud andmeid, kuid loodava rakenduse raames kasutatakse nendest vaid väikest osa. Esiteks läheb vaja mängus läbiviidud matšide andmeid, kust on võimalik kätte saada mängijate poolt saavutatud skoorid iga mängitud taseme kohta. Matšide andmed on hädavajalikud kohtunike töö automatiseerimiseks. Lisaks on tarvis andmeid mängitavate tasemete kohta, et saaks neid turniirietappidesse lisada ja rakenduses välja näidata. Viimasena on vaja osu! kasutajate andmeid, et oleks võimalik hallata kasutajaid ja nendega seotud rolle.

Kokkuvõtteks on autori hinnangul osu! API kasutamine rakenduses olulise tähtsusega, kuna see hõlbustab paljude protsesside automatiseerimist, mis on üks loodava rakenduse peamistest eesmärkidest. Esiteks ei ole autoril tarvis nõuda klientidelt uue kasutaja loomist, kuna osu! API võimaldab osu! kasutajaga sisselogimist. Teiseks ei pea rakendus hoidma andmebaasis kasutajate paroole ega muud tundlikku informatsiooni, kuna sisselogimisel kasutatakse OAuth2 protokolli. Viimaks aitab osu! API kasutamine märkimisväärselt vähendada kasutaja poolt käsitsi sisestatava informatsiooni hulka uute objektide loomisel. Näitena saab tuua, et uue taseme loomisel on kasutajal vaja sisestada vaid selle identifikaator. Kõik muu vajalik informatsioon päritakse läbi osu! API.

4.3 Kasutatavate tehnoloogiate valik

Enne arendusprotsessi algust on oluline valida rakenduses kasutatavad tehnoloogiad, mille hulka kuuluvad andmebaasisüsteem ning esi- ja tagarakenduse arendamiseks kasutatavad programmeerimiskeeled ja raamistikud. Selleks määratakse tehnoloogiate võrdluse aluseks olevad mõõdikud, misjärel antakse valikutest tabeli kujul ülevaade ning tehakse nende põhjal otsus.

4.3.1 Andmebaasisüsteem

Andmebaasid saab laialt võttes jagada relatsioonilisteks (SQL) ja mitterelatsioonilisteks (NoSQL). Relatsioonilised andmebaasid võimaldavad hoida andmeid struktuuritud kujul ja teha keerulisi päringuid läbi SQL päringukeele, kuid on vähem paindlikud ega pole nii skaleeritavad. Mitterelatsioonilised andmebaasid võimaldavad andmete hoidmisel suuremat paindlikkust, on paremini skaleeritavad ning sobivad hästi siis, kui andmete kuju ei ole selgelt määratud või hoitakse väga suure mahuga andmeid [30].

Loodava rakenduse jaoks on autor otsustanud kasutada relatsioonilist andmebaasi, kuna rakenduse andmed on oma olemuselt struktureeritud ning relatsiooniline andmebaas väljendab hästi nii andmete struktuuri kui andmetevahelisi seoseid. Lisaks tagab see paremini andmete terviklikkuse ning võimaldab kasutada andmete pärimiseks SQL päringukeelt, millega autor tuttav on. Viimaks ei ole plaanis hoida andmebaasis väga suure mahuga andmeid, mis vähendab mitterelatsiooniliste andmebaaside kasulikkust.

Autorile on oluline, et andmebaasisüsteem oleks vabavaraline. Loodav rakendus on oma olemuselt suhteliselt lihtne ning seega ei vaja see paljusid tasuliste süsteemide poolt

pakutavaid lisafunktsionaalsusi. Kuigi tasulistel andmebaasisüsteemidel võivad olla ka tasuta versioonid, on neile tihti seatud piiranguid andmete mahu, andmebaasi jõudluse ning süsteemi funktsionaalsuse osas, mis võivad tulevikus probleeme põhjustada. DB-Engines edetabelist selgub, et populaarseimad vabavaralised relatsioonilised andmebaasisüsteemid on MySQL, PostgreSQL ja SQLite [31]. Valiku tegemisel lähtus autor andmebaasi paindlikkusest, keerukusest ja skaleeritavusest ning enda kogemusest:

- PostgreSQL – Tegu on tasuta ja avatud lähtekoodiga objekt-relatsioonilise andmebaasisüsteemiga. Võrreldes teiste vaadeldud süsteemidega on PostgreSQL paindlikum ja toetab väga suurt osa kõikidest SQL funktsioonidest. Lisaks võimaldab see kasutada keerurkamaid andmetüüpe, näiteks JSON-it. PostgreSQL tugevuseks on ka kõrge jõudlus, seda eriti keerulisemate päringute puhul. Suuremast paindlikkusest tulenevalt on PostgreSQL aga ka keerulisem süsteem ning selle õppimine on aeganõudvam [32].

- MySQL – Tegu on tasuta ja avatud lähtekoodiga andmebaasisüsteemiga. Võrreldes PostgreSQL-iga on MySQL vähem paindlik ega sisalda nii palju SQL funktsionaalsusi, kuid selle kasutamine on lihtsam ja mugavam. MySQL tugevusteks on skaleeritavus ning kõrge jõudlus, seda eriti lihtsamate päringute puhul [32]. See on sobivam projektide loomiseks, kus võimsamate süsteemide poolt pakutavate funktsionaalsuste olemasolu väga oluline ei ole.

- SQLite – Tegu on tasuta ja avatud lähtekoodiga andmebaasisüsteemiga. SQLite tugevusteks on selle väike suurus ning seadistamise ja kasutamise lihtsus. See on aga võrreldes eelnevate süsteemidega kõvasti vähem paindlik. Näiteks ei ole SQLite-i sisse ehitatud andmetüüpi kuupäevade hoidmiseks. Lisaks ei ole SQLite hästi skaleeritav ega sobi suuremate projektide arendamiseks [33].

Tabel 1. Andmebaasisüsteemide võrdlus.

Andmebaas	**Paindlikkus**	**Keerukus**	**Skaleeritavus**	**Kogemus**
PostgreSQL	Suur [32]	Keskmine [32]	Hea	Keskmiselt
MySQL	Keskmine [32]	Madal [32]	Hea	Väike
SQLite	Madal [33]	Väga madal [33]	Piiratud	Väike

SQLite pole autori hinnangul loodava rakenduse jaoks piisavalt paindlik ega skaleeritav. Alles jäänud valikud on mõlemad loodava rakenduse jaoks sobivad, kuid PostgreSQL on paindlikum, sisaldab rohkem funktsioone ning autoril on sellega rohkem kogemust. Eelnevast tulenevalt otsustas autor PostgreSQL-i kasuks.

4.3.2 Tagarakenduse tehnoloogia

Tagarakenduse tehnoloogia valik koosneb programmeerimiskeeltest (edaspidi keeltest) ja raamistikest, millest esimesena vaadeldakse keeli. Lõputöö kirjutamiseks antud aja piiratust arvestades on autori hinnangul mõistlik valida keelte vahel, mille kasutamisega on tal kogemusi, millel on suur kogukond ning mis on sobivad veebirakenduste loomiseks. Keelte populaarsuse hindamiseks kasutas autor TIOBE indeksit [34]. Eelnevatest kriteeriumitest lähtuvalt otsustas autor teha valiku PHP, Java, C#-i ja Pythoni vahel. Kindla keele valiku puhul mängib rolli autori kogemus ning keele keerukus, jõudlus ja tüübitus. Autor eelistab staatilist tüübitust, kuna see võimaldab vigu enne rakenduse käivitamist avastada. Lisaks on staatiliselt tüübitud kood autorile loetavam ja arusaadavam, mis aitab vähendada vigade tegemise tõenäosust:

- PHP – Tegu on objektorienteeritud, dünaamiliselt tüübitud ning interpteteeritud keelega. Võrreldes teiste analüüsitud keeltega seisneb PHP eripära selles, et HTML märgistuskeel kirjutatakse teenusepoolel ning tihti lisatakse PHP kood otse selle sisse, kasutades selleks spetsiaalseid märgendeid. PHP on hinnatud tänu selle õppimise ja kasutamise lihtsusele, kõrgele jõudlusele ning väga heale dokumentatsioonile [35].

- Java – Tegu on objektorienteeritud, staatiliselt tüübitud ning kompileeritud keelega. Java on hinnatud tänu lihtsusele, turvalisusele, robustsusele ning ka sellele, et Java koodi saab käivitada erinevatel platvormidel. Üheks Java puuduseks on suurem mälukasutus võrreldes teiste keeltega [36]. Javal on olemas suur kogukond ning põhjalik dokumentatsioon, mis lihtsustab abi leidmist. Lisaks on Javale loodud ehitustööriist nimega Gradle, mis võimaldab teekide lihtsat haldamist, automaatset testimist ning rakenduse ehitamist [37].

- C# – Tegu on objektorienteeritud, staatiliselt tüübitud ning kompileeritud keelega, mida kasutatakse laialdaselt veebirakenduste arendamiseks ning mida hinnatakse tänu selle suurele paindlikkusele. Keelel on tänu oma populaarsusele olemas suur

kogukond, mis teeb abi leidmise kergeks. Lisaks on sellele loodud väga hea dokumentatsioon. Võrreldes teiste keeltega on C#-i õppimine keerulisem [38].

- Python – Tegu on objektorienteeritud, dünaamiliselt tüübitud ning interpreteeritud keelega. Pythonit hinnatakse peamiselt tänu selle lihtsale süntaksile ja algajasõbralikkusele, suurele kogukonnale ning nii keele kui ka selle teekide paindlikkusele. Pythoni peamiseks puuduseks on selle madalam jõudlus ja puudulikum turvalisus võrreldes teiste vaadeldud keeltega [39].

Tabel 2. Tagarakenduse programmeerimiskeelte võrdlus.

Keel	**Kogemus**	**Keerukus**	**Jõudlus**	**Tüübitus**
PHP	Väike	Väga madal [35]	Üle keskmise [35]	Dünaamiline
Java	Palju	Madal [36]	Kõrge [36]	Staatiline
C#	Keskmiselt	Keskmine [38]	Kõrge [38]	Staatiline
Python	Väike	Väga madal [39]	Keskmine [39]	Dünaamiline

Tagarakenduse loomisel on lisaks keeltele olulised ka raamistikud, kuna need lihtsustavad oluliselt arendusprotsessi ja pakuvad võimsaid teeke ja tööriistu, mis on abiks erinevate veebirakenduste arendamisega seotud aspektide osas. Raamistiku valikul on oluline autori kogemus ning mingil määral ka raamistiku jõudlus. Piiratud ajast tulenevalt tuleb silmas pidada ka raamistike õppimis- ja kasutamiskeerukust. Loodav rakendus sisaldab palju kasutajarolle ning sellest tulenevalt peaks raamistik omama head lahendust rollipõhise autoriseerimise ja turvalisuse saavutamiseks:

- Laravel – Tegu on raamistikuga, mida kasutatakse PHP veebirakenduste arendamiseks. Laravel on keskendunud arendusprotsessi kiirendamisele ja arendaja mugavusele. Laravelil on olemas sisseehitatud lahendus autoriseerimise konfigureerimiseks. Laraveli puudusena on välja toodud keeruline dokumentatsioon, mis võib raamistiku õppimist aeglustada [40].

- Spring Boot – Tegu on Javal põhineva Spring raamistiku mooduliga, mis loodi viimase kasutamise lihtsustamiseks ning mis kiirendab Springil põhinevate

projektide loomist läbi automaatse konfiguratsiooni kasutamise [41]. Kuna Spring Boot kasutab Spring raamistikku, on sellele võimalik lisada erinevaid mooduleid, näiteks Spring Security, mis võimaldab mugavat autoriseerimise haldust ning kaitseb rakendust levinumate rünnakute eest [42]. Lisaks on raamistikul olemas suur kogukond, mis aitab kaasa probleemidele korral abi leidmisele.

- ASP.NET Core – Tegu on levinud mitmeplatvormilise ning avatud lähtekoodiga veebirakenduste loomise raamistikuga, mille puhul kasutatakse enamasti C# keelt. Raamistik on hinnatud tänu oma väga kõrgele jõudlusele, modulaarsusele ja paindlikkusele. ASP.NET Core sisaldab häid sisseehitatud lahendusi turvalisuse ja rollipõhise autoriseerimise saavutamiseks [43] ning sellel on olemas ka suur kogukond ja põhjalik dokumentatsioon.

- Django – Tegu on raamistikuga, mida kasutatakse Pythonil põhinevate veebirakenduste arendamiseks. Djangot hinnatakse tänu selle paindlikkusele, turvalisusele ja heale skaleeritavusele. Raamistik pakub olemasolevaid lahendusi kasutajate autoriseerimiseks ja rünnakute eest kaitsmiseks. Selle tugevuseks on ka suur kogukond ja põhjaliku dokumentatsiooni olemasolu. Django puudustena on välja toodud selle õppimiskeerukus ja keskpärane jõudlus [44].

Tabel 3. Tagarakenduse raamistike võrdlus.

Raamistik	**Kogemus**	**Jõudlus**	**Keerukus**	**Turvalisus**
Laravel	Puudub	Kõrge [40]	Madal [40]	Sobiv
Spring Boot	Keskmiselt	Kõrge	Alla keskmise [41]	Sobiv
ASP.NET Core	Keskmiselt	Väga kõrge	Keskmine	Sobiv
Django	Puudub	Keskmine [44]	Üle keskmise [44]	Sobiv

Vaadeldud keeltest on autori eelistusteks C# ja Java, kuna tegu on staatiliselt tüübitud keeltega ning autoril on nende kasutamisega rohkem kogemusi. Autorile ei sobi PHP, kuna serveri poolne HTML teeb tema hinnangul koodi struktureerimise keerulisemaks ega võimalda esi- ja tagarakendust nii lihtsalt iseseisvalt arendada ning lisaks puudub

autoril kogemus Laravel raamistikuga. Pythoni puhul on küll tegu lihtsa keelega ja autoril on sellega veidi kogemust, kuid autoril puudub kogemus Django raamistikuga ning selle keerukuse tõttu võib selle õppimine võib osutuda liiga aeganõudvaks. Kõigest sellest lähtuvalt jäävad valikusse Java ning C# koos vastavate raamistikega.

Mõlemad tehnoloogiad on loodava rakenduse jaoks hästi sobivad, seega sai otsustavaks autori kogemus, raamistike kasutamise keerukus ning olemasolevate tööriistade valik. Võrreldes C# keelega on autoril Java keelega rohkem kogemust ning on seda ka töö kontekstis kasutanud. Javal on olemas väga hea ehitustööriist Gradle näol, mis on paindlikum analoogsetest C# tööriistadest. Lisaks on Spring raamistiku kasutamine tänu Spring Boot moodulile lihtsam kui ASP.NET Core-i kasutamine. Eelnevast lähtudes otsustas autor valida tagarakenduse arenduseks Java programmeerimiskeele koos Spring raamistikuga, kasutades Spring Boot moodulit.

4.3.3 Esirakenduse tehnoloogia

Veebilehe struktuuri loomiseks ja välimuse kujundamiseks kasutatakse valdavalt HTML märgistuskeelt ja CSS kujunduskeelt, kuid veebilehele interaktiivsuse lisamiseks on kõige populaarsem JavaScript programmeerimiskeel. See võimaldab lisada veebilehele animatsioone, graafikat ja muid interaktiivseid ning visuaalseid elemente. Lisaks võimaldab JavaScript veebilehe sisu dünaamiliselt uuendada [45].

JavaScript on dünaamiliselt tüübitud keel. Kuna autor eelistab staatilist tüüpimist, otsustas ta valida esirakenduse keeleks TypeScripti. Tegu on staatiliselt tüübitud objektorienteeritud keelega, mis põhineb JavaScriptil, kasutab sama süntaksit ning on loodud selle täiendamiseks [46]. Sarnaselt JavaScriptile on TypeScript väga laidaldaselt kasutatav ning omab head dokumentatsiooni.

JavaScriptile on loodud väga võimsaid raamistikke, mis teevad esirakenduse arendamise palju kiiremaks ja mugavamaks ning võimaldavad lihtsamini luua paindliku kasutajaliidese. State of JavaScript 2022 küsitlusel osutusid nendest populaarseimaks React, Vue ning Angular [47]. Kuna loodavale esirakendusele ei ole seatud väga erilisi nõudeid, lähtub autor raamistiku valikul peamiselt enda kogemusest ning raamistiku õppimis- ja kasutamiskeerukusest. Kuna autor on otsustanud kasutada TypeScripti, on raamistiku valikul ka oluline, et sellel oleks olemas hea TypeScripti tugi. Lisaks on oluline teekide ja pluginate valik, kuna nende kasutamine lihtsustab arendusprotsessi:

- React – Tegu on kolmest vaadeldust raamistikust kõige populaarsemaga. Reacti peamine tugevus on selle komponentide põhine lähenemine, mis võimaldab efektiivset koodi taaskasutamist. Lisaks hinnatakse teeki tänu selle kasutamise lihtsusele, kõrgele jõudlusele ja testimise mugavusele. Tänu komponentide modulaarsusele ning jõudlusele on React ka väga hästi skaleeritav. Puudustena on välja toodud, et Reacti õppimine võib olla keerukas inimestele, kellel on vähe JavaScripti kasutamise kogemust [48] ning harjumatu võib olla ka komponentide kirjeldamiseks kasutatava JSX-i süntaks.

- Vue – Tegu on populaarse JavaScripti raamistikuga, mida kasutatakse erinevatel platvormidel kasutajaliideste arenduse kiirendamiseks. Vue suureks tugevuseks on selle kasutamise ja õppimise lihtsus ning hea dokumentatsioon. Võrreldes teiste raamistikega on Vue mahult väike ega sisalda liigseid faile. Raamistiku puhul hinnatakse ka selle kõrget jõudlust ning sarnaselt Reactiga ka komponentidel põhinevat arhitektuuri. Samas on Vue võrreldes React või Angular raamistikuga vähem küps ning seetõttu on sellele saadavam vähem teeke ja pluginaid [49].

- Angular – Tegu on avatud lähtekoodiga JavaScripti raamistikuga, mis on loodud kasutajaliideste arendamise lihtsustamiseks. Aastal 2016 tuli välja Angulari versioon 2, mis viis raamistiku üle TypeScript keele peale ja tegi võimalikuks komponentide kasutamise koodi taaskasutamiseks, mis on ka Angulari üks tugevusi. Raamistikku hinnatakse ka tänu selle kõrgele jõudlusele, heale dokumentatsioonile ja suurele väliste teekide ja pluginate valikule. Angulari suureks puuduseks on aga selle suur keerukus ja järsk õppimiskõver [50].

Tabel 4. Esirakenduse raamistike võrdlus.

Raamistik	**Kogemus**	**Keerukus**	**TypeScripti tugi**	**Teekide valik**
React	Keskmiselt	Keskmine [48]	Väga hea [51]	Lai
Vue	Väike	Madal [49]	Väga hea [52]	Keskmine
Angular	Puudub	Kõrge [50]	Vaikimisi [50]	Lai

Kõigil vaadeldud raamistikel on olemas piisavalt hea TypeScripti tugi, seega sai otsustavaks peamiselt autori kogemus, raamistiku keerukus ja teekide valik. Koheselt jääb valikust välja Angular, sest autoril puudub selle kasutamise kogemus ning kuna tegu on mahuka ja keeruka raamistikuga, ei ole autori hinnangul mõistlik üritada seda väga piiratud aja jooksul selgeks õppida. React ja Vue on väga sarnased ning on mõlemad esirakenduse arenduseks sobivad, kuid autor on tuttavam React raamistikuga ning selle teekidega. Lisaks on Reacti JSX süntaks autorile meelepärasem kui Vue mallide süntaks. Kõigest eelnevast lähtuvalt otsustas autor valida esirakenduse tehnoloogiaks TypeScript programmeerimiskeele koos React raamistikuga.

4.4 Versioonihalduskeskkonna valik

Git on laialdaselt kasutatav tehnoloogia, mis jälgib koodis tehtavaid muudatusi ning versioneerib koodi. See aitab vältida ja lahendada erinevate koodijuppide ühildumisest tulenevaid probleeme ning võimaldab vajadusel taastada koodi varasema versiooni, lihtsustades seeläbi arendusprotsessi ja arendajate vahelist koostööd [53]. Git hoiab informatsiooni hoidlates ehk repositooriumites, mille veebipõhiseks hostimiseks on loodud mitmeid keskkondi. Nendest populaarsemad on GitHub, BitBucket ja GitLab [54]. Kõigil kolmel keskkonnal on olemas nii tasulised kui ka tasuta versioonid [55]:

- GitHub – Tegu on vaadeldavatest keskkondadest populaarseimaga. GitHub on hinnatud tänu selle lihtsale kasutajaliidesele ja suurele jõudlusele ning see on väga populaarne avatud lähtekoodiga tarkvara loojate kogukondade seas. Võrreldes teiste keskkondadega sisaldab GitHubi tasuta versioon rohkem piiranguid [55].

- BitBucket – Tegu on Atlassani poolt loodud versioonihalduskeskkonnaga, mis integreerub hästi Jira ja muude Atlassiani ökosüsteemi kuuluvate toodetega. Tänu sellele on BitBucket populaarne Atlassani tooteid kasutavate ettevõtete seas. Puudusteks on aga keeruline kasutajaliides ja madalam jõudlus [55].

- GitLab – Võrreldes teiste keskkondadega on GitLab uuem ja vähem populaarne. GitLabi suurimaks tugevuseks on selle väga hea DevOps arendustsükli toetus, tänu millele on see populaarne DevOpsi kasutatvate ettevõtete seas. Lisaks on GitLabil lihtne kasutajaliides. Puudusteks on aga vähene kasutajate arv võrreldes teiste keskkondadega [55].

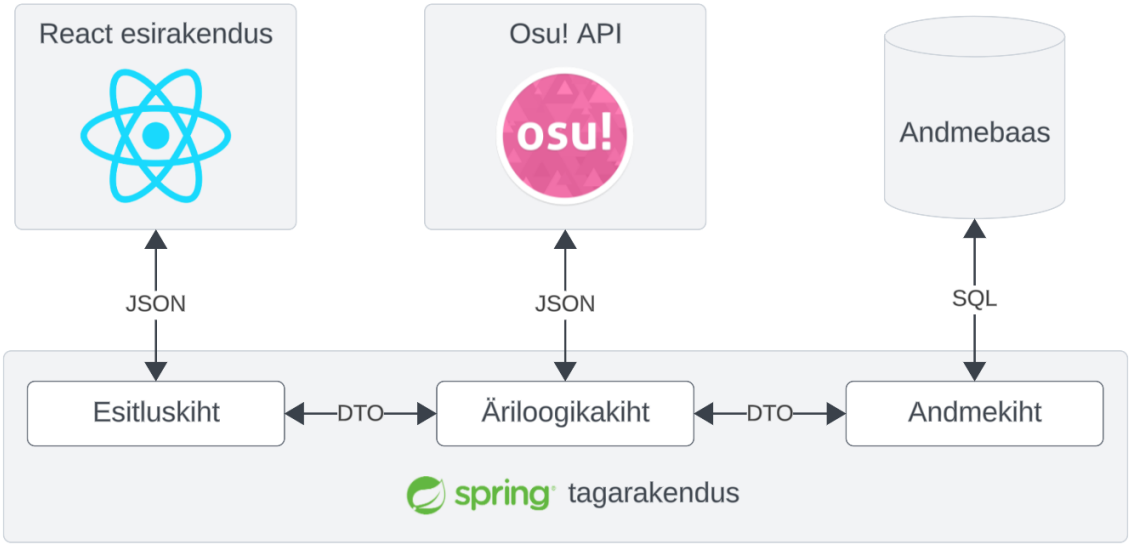
Vaadeldud keskkondadest hindas autor sobivaimaks GitHubi, kuna see on nendest kõige populaarsem, lihtsa kasutajaliidesega, on sobiv väiksemate projektide arendamiseks ning teeb vajadusel tulevikus teiste arendajate abi kaasamise lihtsaks. Lisaks ei ole GitLabi ega BitBucketi suurimad tugevused loodava rakenduse kontekstis väga kasulikud.

4.5 Rakenduse arhitektuur

Rakenduse arhitektuuri määramine on arendusprotsessi oluline osa, kuna hästi läbimõeldud arhitektuuriga tarkvara on pikemas perspektiivis lihtsam hallata ning sellesse on lihtsam ilma vigu tekitamata muudatusi sisse viia [56]. Mõistlik on arhitektuur paika panna juba arendusprotsessi alguses, kuna hiljem on seda palju keerulisem teha.

Puhtama arhitektuuri saavutamiseks on loodav rakendus jagatud erinevatesse kihtidesse, milleks on esirakendus ehk kliendipoolne osa, tagarakendus ehk serveripoolne osa, ning andmebaas. Rakenduse kihtidesse jaotamine hoiab nende funktsionaalsused ja vastutused üksteisest lahus ning võimaldab iga kihti eraldi käsitleda. Kihilise arhitektuuri peamisteks eelisteks on rakenduse mugavam arendamine ja testimine ning lihtsam muudatuste läbiviimine, kuna kihte saab arendada ja testida üksteisest sõltumatult. Lisaks võimaldab selline lähenemine rakenduse kihtide lihtsat taaskasutust. Näiteks on võimalik vahetada välja rakenduse kasutajaliides või andmebaas ilma, et peaks tagarakendust muutma [57]. Loodava rakenduse arhitektuur on kujutatud allpool (vt Joonis 1).

Loodava lahenduse rakendusliides järgib *Representational State Transfer* ehk REST arhitektuurilaadi. REST määrab ära tavad ja piirangud API-de vahelise andmevahetuse formaadile. API-sid, mis REST-i piirangutele vastavad, nimetatakse ka REST API-deks. Tavaliselt kasutatakse selliste API-de puhul andmete edastamiseks HTTP protokolli ning andmevahetus toimub REST API päringute ja vastuste kaudu. Päringus identifitseeritakse kliendi poolt nõutav ressurss ning määratakse sellele rakendatav tegevus. Vastuses on kirjas staatuskood ning vastuse sisu, mis sisaldab andmeid kas JSON-i või mõne muu andmevormingu kujul. REST API päringud ei hoia endas olekut, tänu millele on nad iseseisvad ega sõltu üksteisest [58]. Autor valis REST arhitektuurilaadi, kuna see sobib hästi kokku rakenduse kihilise arhitektuuriga, on laialdaselt kasutatav ning autoril on olemas selle kasutamise kogemus.



Joonis 1. Loodava rakenduse arhitektuur.

Rakenduse arhitektuur koosneb järgmistest kihtidest:

- React esirakendus – Tegemist on ainsa lõppkasutajale nähtava kihiga. Esirakenduse eesmärk on kasutajaliidese ja selles näidatavate vajalike andmete kuvamine kasutajatele. Andmete vahetamiseks suhtleb see tagarakenduse esitluskihiga REST API päringute teel ning vahetab andmeid JSON andmevormingu kujul.

- Java Spring tagarakendus – Sarnaselt rakenduse üldisele ülesehitusele on ka tagarakenduses kasutatud mitmekihilist arhitektuuri, kus iga kiht täidab kindlat rolli. Sellise lähenemise eelisteks on võimalus kihte üksteisest sõltumatult arendada ja testida, mis lihtsustsab arendusprotsessi [59]. Lisaks teeb selline lähenemine rakenduse struktuuri loogilisemaks ja paremini jälgitavamaks. Tagarakenduse pealmiseks kihiks on esitluskiht (REST API kontrollerid), mille ülesandeks on esirakenduse poolt tulevate päringute vastuvõtmine ning neile vastamine. Teiseks tagarakenduse kihiks on äriloogikakiht (teenused), mille ülesandeks on andmete töötlemine ja nendega äriloogika poolt sätestatud toimingute läbiviimine. Äriloogikakihis toimub ka integratsioon osu! API-ga, mille andmevahetusviis on analoogne esirakendusega. Viimaseks kihiks on andmekiht (repositooriumid ja domeeniobjektid), mille ülesandeks on andmebaasiga suhtlemine. Kihtide vahel toimub andmete liikumine DTO-de kujul, millega tagatakse järgnevatele kihtidele ainult vajalike andmete

edastamine. DTO-de kasutamine lihtsustab objektide struktuuri ning suurendab jõudlust, kuna väheneb kihtide vahel liikuvate andmete maht [60].

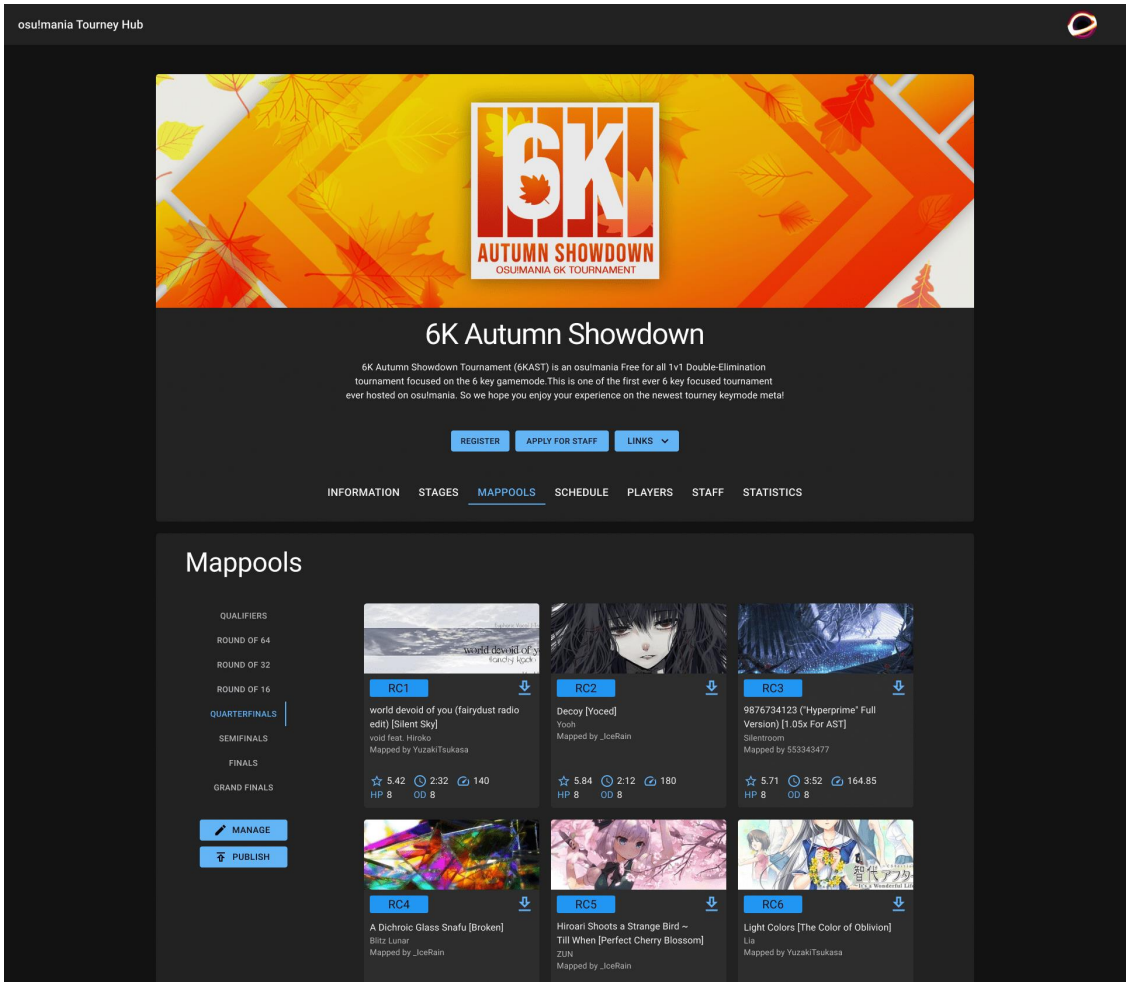
- PostgreSQL andmebaas – Kõiki rakendusega seotud andmeid hoitakse andmebaasis neile vastavates tabelites. Andmete vahetamiseks suhtleb andmebaas tagarakenduse andmekihiga, kasutades selleks SQL päringukeelt.

4.6 Kasutajaliidese disain

Vastavalt rakenduse mittefunktsionaalsetele nõuetele peab kasutajaliidese disain olema sidus ning selle kasutamine peab olema võimalikult lihtne ja intuitiivne. Turniiride organiseerimise ja haldamise seisukohalt on ka oluline, et nende aspektidega seotud tegevuste läbiviimine oleks kasutajale võimalikult mugav ega nõuaks liigset õppimist. Kuna rakenduse oluliseks osaks on ka matšide läbiviimise võimaldamine, peab tähelepanu pöörama ka antud tegevusega seotud kasutajaliidese vaadete kasutatavusele.

Esirakenduse visuaalseks stiiliks on autor valinud tumeda kasutajaliidese, kuna see langeb hästi kokku osu! kodulehe ja paljude muude osu! teemaliste veebirakendustega ning on autori isiklik eelistus. Antud stiili kasutades on kasulik silmas pidada mitmeid disainiprintsiipe. Esiteks ei ole tumeda kasutajaliidese korral tegemist ainult musta ja valge värviga, vaid pigem domineerivad erinevad hallitoonid. Teiseks tuleb värvide puhul vältida liigset küllastunust, kuna see tekitab silmadel väsimust. Kasutada tuleks heledaid, väheküllastunud värve. Kolmandaks peaks kasutajaliidese elementide eristamiseks kasutama varjude asemel hallitoone, kuna varjud ei paista tumedal taustal hästi välja. Viimaks peaks tekst olema puhta valge asemel veidi hallikam, kuna see muudab tumedal taustal teksti lugemise silmadele vähem väsitavaks [61].

Enne esirakenduse arendamise alustamist otsustas autor valmis teha kasutajaliidese prototüübi. Prototüüpimine on kiire ja mugav viis kasutajaliidese disainimiseks ja visualiseerimiseks ilma, et peaks tegelikku esirakendust valmis arendama. Prototüüp võimaldab kasutajaliidese elementide paigutuse, värvid ning detailid ilma suurema vaevata paika panna ning hoida tänu sellele märkimisväärselt kokku esirakenduse arendamisele kuluvat aega. Rakenduse tumeda kasutajaliidese ja vastavate disainiprintsiipide illustreerimiseks on siinkohal välja toodud loodud prototüübi turniirietapi tasemete nimekirja vaade (vt Joonis 2).



Joonis 2. Prototüübi turniirietapi tasemete nimekirja vaade.

Kasutajaliidese prototüübi loomiseks kasutas autor Figma tarkvara [62]. Tegu on võimsa ja paindliku veebirakendusega, millega on võimalik luua kasutajaliidese visuaalne pool ning osaliselt simuleerida ka selle funktsionaalsust, nagu näiteks nuppude ja linkidega erinevate rakenduse vaadete vahel liikumist. Figma valik tulenes asjaolust, et autor on seda ka varem prototüüpide loomiseks kasutanud ning selle kasutajaliides on autorile meelepärane.

Kasutajaliidese prototüübi arendusprotsessi kiirendamiseks kasutas autor Figma jaoks loodud Material UI komponentide kogumit [63]. Antud lahendus võimaldab kasutada suurt hulka valmis tehtud elemente, mille stiil vastab populaarsele Material UI React komponentide kogumile. Kuna autor plaanib viimast ka esirakenduse arendamisel kasutada, aitab Figma komponentide kogumi kasutamine soovitud välimust kergemini saavutada.

5 Lahenduse teostus

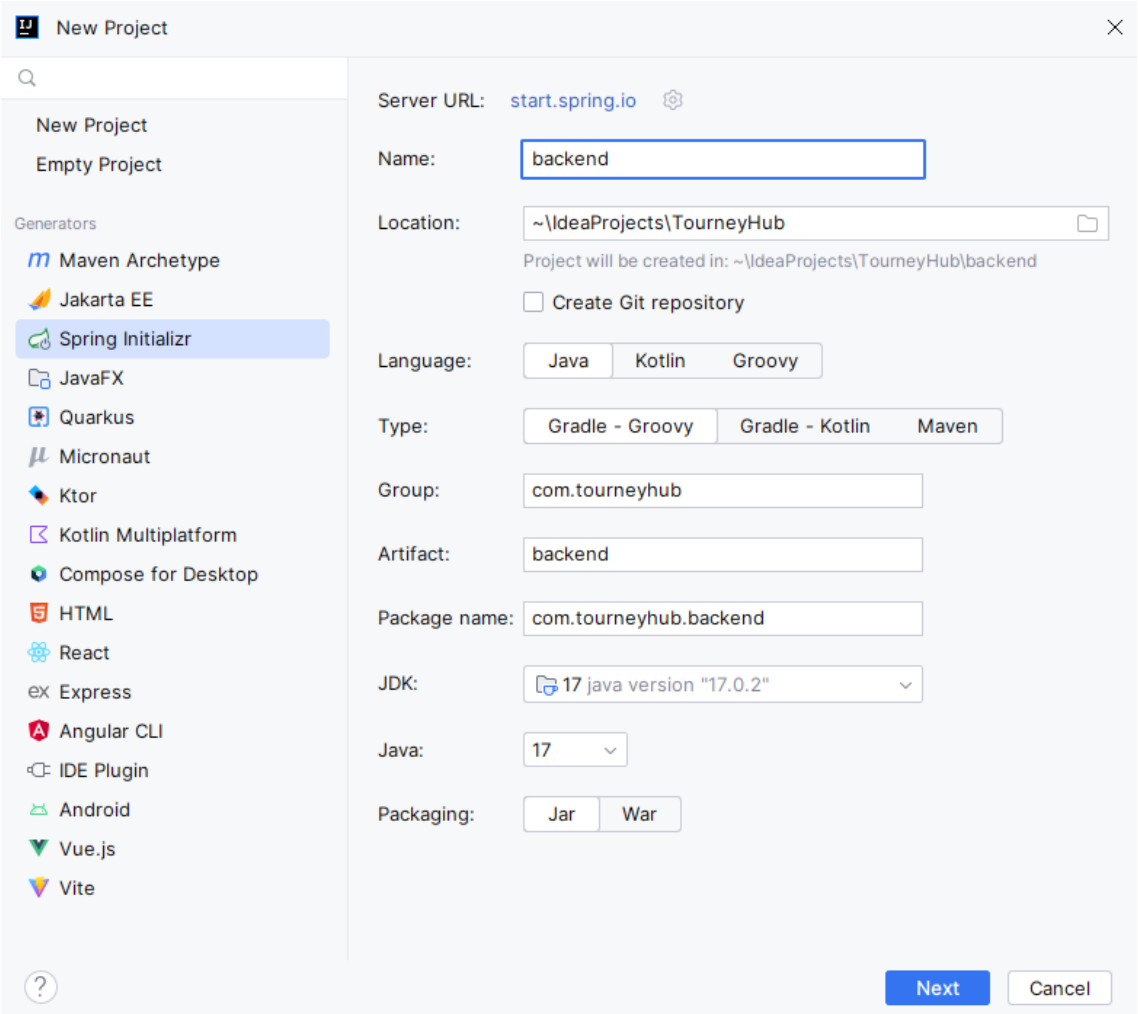
Käesolevas peatükis antakse ülevaade rakenduses kasutatava andmemudeli struktuurist ning kirjeldatakse valminud rakenduse esi- ja tagarakenduse arendusprotsessi ning arenduse käigus kasutatud raamistikke, teeke ja lahendusi.

5.1 Tagarakenduse arendus

Jägnevalt kirjeldatakse rakenduse serveripoolse osa ehk tagarakenduse arendusprotsessi ning kasutatud lahendusi. Vastavalt eelnevalt läbi viidud analüüsile kasutati arenduseks Java programmeerimiskeelt ning Spring Boot raamistikku. Andmebaasina võeti kasutusele PostgreSQL. Kood kirjutati arenduskeskkonnas IntelliJ IDEA.

5.1.1 Projekti loomine

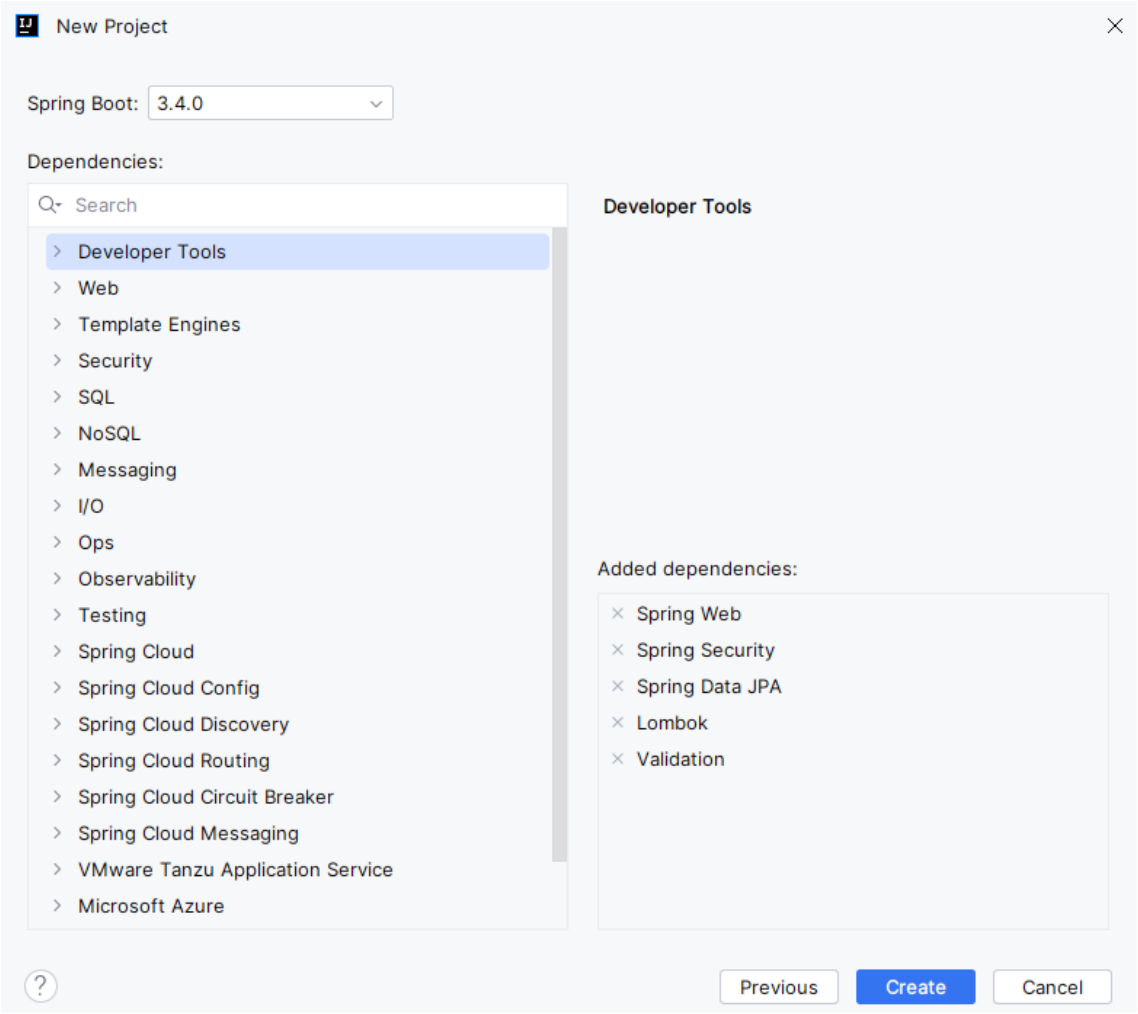
Projekti loomiseks on kasutatud Spring Initializr abitööriista, mis võimaldab kiiresti genereerida Spring Boot baasrakenduse. Genereerimise käigus luuakse projekti kataloogistruktuur ning seatakse üles sõltuvuste haldamiseks vajalik konfiguratsioon, kasutades seadistuse käigus valitud ehitustööriista (käesolvas projektis Gradle). Spring Initializr on saadaval veebipõhiselt, kuid see on integreeritud ka arenduseks kasutatud IntelliJ IDEA keskkonnaga ning seega saab seda antud keskkonnas uue projekti loomisel otse kasutada [64]. Allpool on välja toodud loodava projekti seadistus (vt Joonis 3).



Joonis 3. Projekti üldine seadistus.

Projekti üldisele seadistusele järgneb sõltuvuste valik. Siinkohal otsustas autor valida esialgu vaid tema hinnangul kõige olulisemad sõltuvused, kuna projekti loomise hetkel ei olnud autor veel kõikides kasutatavates sõltuvustes kindel ning neid on võimalik vajadusel lisada ka arendusprotsessi käigus (vt Joonis 4).

Raamistikest valiti Spring Web REST kontrollerite loomiseks, Spring Security kasutajate autentimiseks ja autoriseerimiseks ning Spring Data JPA andmebaasiga suhtlemiseks. Teekidest osutusid valituks Lombok ja Hibernate Validator. Lombok võimaldab annotatsioonide abil genereerida klassidele konstruktorid ning *getter*, *setter*, *equals*, *hashCode* ja *toString* meetodid, vähendades seeläbi korduva koodi kirjutamise vajadust [65]. Hibernate Validator on kasulik domeeniobjektide valideerimiseks, kasutades selleks JSR 380 spetsifikatsioonis defineeritud annotatsioone [66].



Joonis 4. Projekti sõltuvuste valik.

5.1.2 Andmebaasi seadistamine

Rakenduse PostgreSQL andmebaasi ülesseadmise lihtsustamiseks on see seadistatud töötama Dockeri konteineris. Selle saavutamiseks on loodud *docker-compose.yml* fail (vt Joonis 5), mis sisaldab kogu andmebaasi ülesseadmiseks vajalikku informatsiooni. Siia kuuluvad näiteks andmebaasi nimi, ligipääsuandmed ning pordid, mida andmebaas kuulab. Konfigureeritud konteineri saab luua ja käivitada käsuga *docker-compose up*. Et ühendada IntelliJ IDEA arenduskeskkond andmebaasiga, on vaja lisada selleks vajalik informatsioon rakenduse *application.properties* konfiguratsioonifaili.

```
services:
  db:
    container_name: tourney-hub-db
    image: 'postgres:16.4-bullseye'
    restart: unless-stopped
    environment:
      - POSTGRES_USER=postgres
      - POSTGRES_PASSWORD=postgres
      - POSTGRES_DB=tourney-hub-db
    ports:
      - "5432:5432"
    volumes:
      - tourney-hub-db-volume:/var/lib/postgresql/data

volumes:
  tourney-hub-db-volume:
```

Joonis 5. Rakenduse *docker-compose.yml* fail.

Andmebaasiga töötamise lihtsustamiseks on Javale loodud JPA standard, mis määrab ära erinevad meetodid ja annotatsioonid andmebaasi operatsioonide läbiviimiseks ning domeeniklasside põhjal tabelite loomiseks. Rakenduses kasutatakse andmebaasiga suhtlemiseks Hibernate raamistikku, mis on JPA standardi üks implementatsioone. Hibernate võimaldab teostada andmebaasioperatsioone ilma SQL lausete käsitsi kirjutamiseta ning genereerida andmebaasitabeleid, kasutades selleks JPA annotatsioone. Klassi põhjal tabeli loomiseks tuleb sellele lisada annotatsioon *@Entity*. JPA annotatsioonide abil on võimalik määrata ka tabelitevahelisi suhteid, muuta tabeli ja selle veergude nimesid ning palju muud [67].

Domeeniobjektide põhjal andmebaasi genereerimine on arenduse käigus mugav, kuid hiljem võib see tekitada probleeme andmebaasi struktuursete muudatuste tegemisel. Lisaks ei anna see arendajale täielikku kontrolli andmebaasi struktuuri üle. Sellest tulenevalt on rakenduses kasutusel Flyway teek, mis võimaldab kirjeldada andmebaasi struktuuri, kasutades versioneeritud SQL migratsioonifaile. Võrreldes JPA abil andmemudeli genereerimisega annab Flyway arendajale suurema kontrolli andmebaasi täpse struktuuri üle. Lisaks tagab Flyway migratsioonifailide versioneeritus, et andmebaasi muudatused tehakse alati samas järjekorras. See lihtsustab potensiaalsete vigade leidmist ning annab tehtud muudatustest parema ülevaate [68].

Kuigi andmebaasi struktuur on kirjeldatud Flyway migratsioonifailides, peavad rakenduse domeeniobjektid endiselt antud struktuurile vastama. Hibernate pakub võimalust säärast vastavust kontrollida. Selleks tuleb lisada *application.properties* konfiguratsioonifaili rida *hibernate.ddl-auto=validate*, misjärel ei loo Hibernate enam

domeeniobjektide põhjal uusi tabeleid, vaid valideerib need olemasoleva andmebaasistruktuuri vastu, visates mittevastavuse korral erindi.

5.1.3 REST API

Tulenevalt rakenduse arhitektuuri analüüsist (vt peatükki 4.5) on tagarakendusega suhtlemiseks kasutusele võetud REST API arhitektuurilaad. Selle realiseerimiseks rakenduses on loodud REST kontrollerid, mis määravad ära väljaspoolt ligipääsetavad API lõpp-punktid, nendele vastavad HTTP meetodid, tagastatava informatsiooni struktuuri ning vajadusel ka tagastuskoodi.

Spring raamistik võimaldab luua REST kontrollereid, lisades vastavatele klassidele annotatsiooni *@RestController*. Iga kontrolleri lõpp-punkt vastab kindlale HTTP meetodile, milleks on kas GET, POST, PUT või DELETE. Kasutatava meetodi defineerimiseks tuleb lisada klassile vastav annotatsioon, milleks on näiteks GET meetodi puhul *@GetMapping*. Vajadusel on ka võimalik seadistada kontroller vastama igale HTTP meetodile, lisades selleks klassile annotatsiooni *@RequestMapping*. Meetodi annotatsioon võtab argumendina sisse kontrolleri lõpp-punkti. Rakenduses algavad kõik lõpp-punktid aadressiga */api*, millele järgneb kontrolleri nimi ning seejärel täpsem lõpp-punkti aadress.

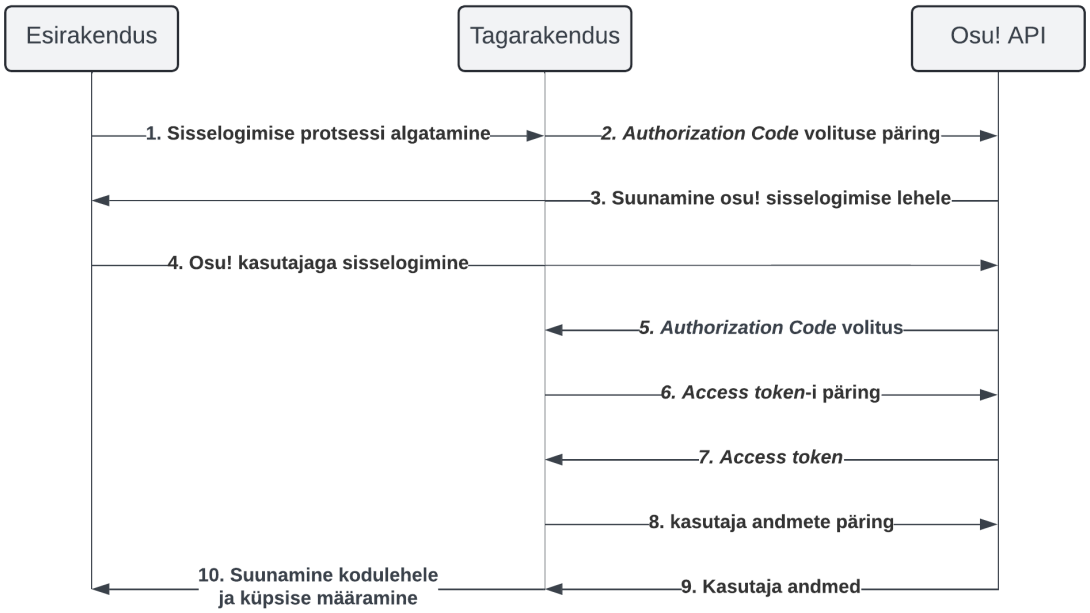
Väljastatavate andmete struktuuri määramiseks on loodud kaust *dto* ning sinna lisatud kõik andmevahetuseks vajalikud DTO klasside definitsioonid. Rakenduse DTO-de struktuur vastab enamjaolt domeeniobjektide struktuurile, kuid kuna tihti sisaldavad andmebaasist tulevad objektid liigset informatsiooni, ei ole need alati DTO-dega üheses vastavuses. Klientrakendusele väljastatakse andmeid JSON-i kujul.

5.1.4 OAuth2 protokolliga sisselogimine

Lähtuvalt rakenduse nõuetest peab see võimaldama osu! kasutajaga sisselogimist. Selle saavutamiseks on ära kasutatud osu! API poolt pakutavat OAuth2 protokolliga sisselogimise võimalust [26], mille implementeerimiseks on rakenduses kasutusele võetud Spring Security raamistik. Spring security pakub mugavaid lahendusi nii autentimiseks ehk kasutaja identiteedi tuvastamiseks, kui ka autoriseerimiseks ehk kasutajal olevate õiguste kontrollimiseks. Spring Security on väga konfigureeritav ning toetab mitmeid erinevaid autentimis- ja autoriseerimisviise, sealhulgas ka OAuth2 protokolli [69].

Spring Security raamistikus OAuth2 protokolli kasutamine nõuab esmalt vastava konfiguratsiooni loomist. Selleks on kõigepealt vaja registreerida osu! kodulehel klientrakendus (vt peatükki 4.2). Seejärel tuleb konfigureerida Spring Security poolt kasutatav osu! OAuth2 pakkuja, luues selleks spetsiaalse objekti ning lisades sinna kõik osu! API kasutamiseks vajalikud andmed. Siia kuuluvad loodava rakenduse registreerimisel saadud *Client ID* ja *Client Secret*, osu! API peamine lõpp-punkt ning kõik muud vajalikud lõpp-punktid sisselogimise protsessi läbiviimiseks.

Kasutajate sisselogimiseks kasutatakse rakenduses OAuth2 *Authorization Code* voogu. Selle tulemusena väljastatakse osu! API poolt *access token* ehk tõend, mille saamiseks on kasutatud OAuth2 *Authorization Code* volitust ning mis võimaldab rakendusel pärida sisselogitud kasutajaga seotud andmeid [29, pp. 24-31]. Sisselogimise protsessi algatamiseks suunatakse kasutaja vastavale aadressile, milleks on käesolevas rakenduses */oauth2/authorization/osu*. Suurem osa protsessi käigus sooritatavatest päringutest on Spring Security poolt automatiseeritud, leides vajalikud API lõpp-punktid eelnevalt loodud OAuth2 pakkuja objektis olevast konfiguratsioonist. Allpool on nähtav kasutaja sisselogimise voo skeem (vt Joonis 6).



Joonis 6. Kasutaja sisselogimise voo skeem.

Pärast kasutaja sisselogimist suunatakse ta esirakenduse kodulehele ning määratakse küpsis. Tegu on väikese andmepaketiga, mida hoitakse veebibrauseri sessioonis. Sisselogimise käigus luuakse tagarakenduses uus sessioon ning kasutajale määratakse

unikaalne sessiooni ID, mis salvestatakse kasutaja veebibrauserisse küpsise kujul. Kõikidele järgnevatele tagarakenduse API päringutele pannakse kaasa ka salvestatud küpsis ning tagarakendus saab selles sisalduva sessiooni ID abil kasutaja tuvastada [70]. Tagarakenduse sessioonis hoitakse ka kasutajate OAuth2 tõendeid, mis aitab tõsta rakenduse turvalisust [71]. Kasutajale määratud küpsisele ei lisata aga tema andmeid, mistõttu tuleb need pärast sisselogimist tagarakenduse API-lt eraldi pärida.

Sisselogimise käigus päritakse osu! API-lt vastava kasutaja andmed, mille abil on võimalik kas salvestada kasutaja rakenduse andmebaasi või uuendada seal juba olemasoleva kasutaja andmeid. See eemaldab eraldi kasutajate registreerimise vajaduse ning aitab säilitada võimalikult head vastavust rakenduse andmebaasis ja osu! serveris olevate kasutajate andmete vahel.

5.1.5 Kasutaja rollid

Kõikide rakenduse kasutajatega on seotud rollid ning nendega kaasnevad õigused. Sellest tulenevalt peab olema tagatud, et kasutaja ei saaks näha informatsiooni ega sooritada tegevusi, mille jaoks tal sobivad õigused puuduvad. Olukorda teeb keerulisemaks ka asjaolu, et kasutajate rollid on alati seotud turniiridega ning ühel kasutajal võib olla erinevates turniirides mitmeid rolle (vt peatükki 3.1.3). Sellest tulenevalt tuleb kasutaja rollide ja õiguste kontrollimisel silmas pidada, millise turniiriga on tegemist.

Kasutaja rollide seotus turniiridega tekitab vajaduse neid käsitsi kontrollida, kuna Spring Security ei sisalda antud olukorraks hästi sobivat lahendust. Selleks on kasutajate halduse eest vastutavasse teenusesse lisatud meetodid, mis võimaldavad tuvastada kõik kasutaja rollid kindlas turniiris ning kontrollida seeläbi tema õigusi. Loodud meetodeid saab seejärel teistes teenustes kasutada. Spring Security pakub ka võimalust lisada autoriseerimine otse kontrolleri lõpp-punkti meetodile, kasutades selleks annotatsiooni *@PreAuthorize*, mis käivitatakse enne väljakutsutavat meetodit ning mis võimaldab samuti rollide kontrollimiseks loodud meetodeid välja kutsuda.

5.1.6 Osu! API kasutamine

Rakendus sisaldab lisaks sisselogimisele mitmeid funktsionaalsuseid, mis nõuavad osu! API käest andmete pärimist. Näiteks on see vajalik uute mängitavate tasemete loomisel, kus on kasutajal võimalus sisestada vaid taseme identifikaator ning kõik muu luuakse automaatselt. Kõik osu! API kasutamist nõudvad funktsionaalsused eeldavad sisselogitud

kasutaja olemasolu ning saavad seega päringute saatmiseks kasutada sisselogimise käigus tagarakenduses olevasse sessiooni salvestatud OAuth2 tõendit.

Vajaliku integratsiooni realiseerimiseks on ära kasutatud Spring poolt pakutavat WebClient HTTP klienti, läbi mille on võimalik saata API päringuid otse tagarakenduse koodist [71]. WebClienti üheks suureks eeliseks on selle ühilduvus juba olemasoleva Spring Security OAuth2 konfiguratsiooniga, mis võimaldab hankida päringuteks vajalikke andmeid eelnevalt loodud osu! OAuth2 pakkuja objektilt ning lisab olemasolu korral päringule automaatselt ka OAuth2 tõendi. Näitena on allpool välja toodud osu! API poolt taseme pärimise kood (vt Joonis 7), millest on selguse huvides eemaldatud erinditega tegelemise osa.

```
private OsuBeatmapDto fetchBeatmapFromOsu(
        @PathVariable("beatmapId") Integer beatmapId
) {
    return webClient
            .get()
            .uri(uriBuilder -> uriBuilder
                    .path("/beatmaps/{beatmapId}")
                    .build(beatmapId)
            )
            .retrieve()
            .bodyToMono(OsuBeatmapDto.class)
            .block();
}
```

Joonis 7. Osu! API poolt taseme pärimise kood.

Päringute vastusteks on JSON objektid, mille struktuur on kirjeldatud osu! API dokumentatsioonis [26]. Antud objektid ei ole aga täpses vastavuses loodava rakenduse domeeniobjektide struktuuriga ning sisaldavad palju ebavajalikke lisaandmeid. Sellest tulenevalt on loodud nende ajutiseks hoidmiseks eraldi DTO objektid, mis konverteeritakse andmebaasi salvestamiseks sobivale kujule.

5.2 Esirakenduse arendus

Järgnevalt kirjeldatakse rakenduse kliendipoolse osa ehk esirakenduse arendusprotsessi ja ülesehitust. Lähtuvalt analüüsist on esirakenduse arenduseks kasutatud TypeScript programmeerimiskeelt ja React raamistikku. Kasutajaliidese struktuuri ja välimuse planeerimisel võeti arvesse eelnevalt analüüsitud disainiprintsiipe (vt peatükki 4.6).

5.2.1 Rakenduse loomine ja struktuur

Projekti loomisel kasutati ära Create React App funktsionaalsust. See võimaldab luua mugavalt React baasrakenduse ning lisada kõik peamised rakenduse toimimiseks vajalikud teegid, kasutades selleks käsku *npx create-react-app*. Pärast baasrakenduse loomist eemaldati ebavajalikud genereeritud komponendid ning kõik muud vajalikud teegid lisati arendusprotsessi käigus käsitsi.

Rakenduse failid on jaotatud erinevatesse kaustadesse. Struktuuri loomisel on tähelepanu pööratud sellele, et kõik failid ja komponendid oleksid funktsionaalsuse järgi ära jaotatud. Enamik allmainitud kaustadest on jagatud ka alamkaustadeks, kuid neid on siinkohal väljatoomiseks liiga palju. Rakenduse peamised kaustad on järgnevad:

- *Routes* – Sisaldab komponente, mida kasutatab ruuter navigeerimiseks.
- *Components* – Sisaldab komponente, mida ruuter navigeerimiseks ei kasutata.
- *Services* – Sisaldab kõiki teenuseid ja baasteenuseid, mis võimaldavad andmevahetust tagarakenduse API-ga.
- *Domain* – Sisaldab domeeniobjektide liideseid, mis on esirakenduse spetsiifilised ning mida ei kasutata tagarakenduse API-ga suhtlemiseks.
- DTO – Sisaldab DTO objektide liideseid, mida kasutatakse tagarakenduse API-ga suhtlemiseks.

5.2.2 Kasutatud teegid ja komponendid

Esirakenduse arendusprotsessi kiirendamiseks ja lihtsustamiseks on kasutusele võetud järgnevad teegid ja komponendid:

- Material UI – React komponentide kogum, mis põhineb Material Design printsiipidel. Material UI pakub suurel hulgal kergesti kohandatavaid komponente nagu näiteks nuppe, tekstielemente ja tabeleid, lihtsustades seeläbi rakenduse kasutajaliidese loomist. Material UI komponentide välimus põhineb üldisel stiilil, mida on võimalik konfigureerida. Näiteks saab muuta rakenduses kasutatavaid baasvärve, tüpograafiat ning palju muud [73].

- React Router – Teek eri lehtede vahelise navigatsiooni loomiseks ilma veebilehe värskendamise vajaduseta, kasutades selleks ruuteri komponenti [74].

- React Quill – Komponent rikastekstiredaktori loomiseks [75]. Tekstiredaktor on rakenduses kasulik turniiride informatsiooni sisestamise hõlbustamiseks, mille puhul kasutatakse tihti erinevaid fontide suurusi, värve, nimekirju ja muud.

- Formik ja Yup – Teegid vormide loomiseks ning kasutaja sisendi valideerimiseks. Formik võimaldab mugavat ja paindlikku React vormide loomist, nendega sonduvate andmete ja veateadete haldamist ning vormide esitamist. Yup võimaldab defineerida skeemi kasutaja sisendite valideerimiseks ning lihtsustab keerulisemate valideerimisreeglite ülesseadmist [76].

- JSON Server – Teek RESTful API käitumise simuleerimiseks. JSON serveri abil on võimalik testida tagarakendusega analoogse API kasutamist, luues selleks faili *db.json* ning lisades sinna testandmed [77], mida saab seejärel esirakendusest pärida. Antud teeki kasutati vaid esirakenduse arendamise ajal.

5.2.3 Suhtlemine tagarakendusega

Tagarakendusega suhtlemiseks on kasutusel Axios HTTP klient, mis võimaldab konfigureerida ja saata HTTP päringuid [78]. Korduva koodi vähendamiseks on kasulik panna kõik teenused kasutama sama Axios konfiguratsiooni. Selle saavutamiseks on loodud Axios baasinstants, mille kaudu kõik päringuid tehakse. Kasutaja tuvastamiseks tuleb konfigureerida instants päringutega kaasa panema ka küpsiseid, milleks tuleb lisada konfiguratsioonile rida *withCredentials: true*. Allpool on välja toodud Axios baasinstantsi loomise kood (vt Joonis 8). Andmevahetus tagarakendusega toimub JSON objektide kujul.

```
export default axios.create({
    baseURL: API_BASE_URL,
    headers: {
        'Content-Type': APPLICATION_JSON,
        'Accept': APPLICATION_JSON
    },
    withCredentials: true
});
```

Joonis 8. Axios baasinstantsi loomise kood.

Päringute saatmise meetodid on kogutud kaustas *services* olevatesse teenuste klassidesse. Korduva koodi vähendamiseks on loodud baasteenus, mis kasutab päringute saatmiseks eelnevalt loodud Axios baasinstantsi ning sisaldab seega nii API baasaadressi kui ka põhilisi meetodeid vastava API lõpp-punktidega suhtlemiseks. Kõik ülejäänud teenused pärinevad baasteenusest, andes konstrukorile sisse vaid loodava teenusega seonduva API lõpp-punkti aadressi, mis lisatakse baasaadressile. Näiteks luuakse turniiride teenus aadressiga */tournaments*.

API päringute saatmisel on oluline ka saadetavate ning tagastatavate andmete struktuur. Sellest tulenevalt on baasteenus loodud geneerilisena ning uue teenuse loomisel tuleb täpsustada sellega seotud objektide liidesed. Antud lahendus võimaldab teenuse loomisel veenduda, et päringutes kasutatakse õige struktuuriga andmeid. Andmete struktuuri määramiseks on loodud ka DTO liidesed, mis järgivad tagarakenduse DTO objektide struktuuri.

Rakenduses esineb mitmeid kohti, kus kasutajal on võimalik *tab*-ide ehk vahekaartide abil eri vaadete vahel liikuda. Iga uue vahekaardi peale vajutades tehakse aga sellega seonduvate andmete laadimiseks uus Axios päring, mis võib kiire vahekaartide vahel liikumise korral põhjustada päringute kuhjumist. Probleemi leevendamiseks on ära kasutatud Axios võimalust peatada päring enne vastuse saamist. Selleks on teenust välja kutsuvas komponendis loodud *AbortController* objekt, mis on võimeline väljastama signaali päringu katkestamiseks. Kui kasutaja vajutab uuele vahekaardile, saadab *AbortController* signaali päringut sooritavale teenusele, mis katkestab kõik eelnevad sama API lõpp-punkti poole tehtud päringud.

5.2.4 Turvalisus

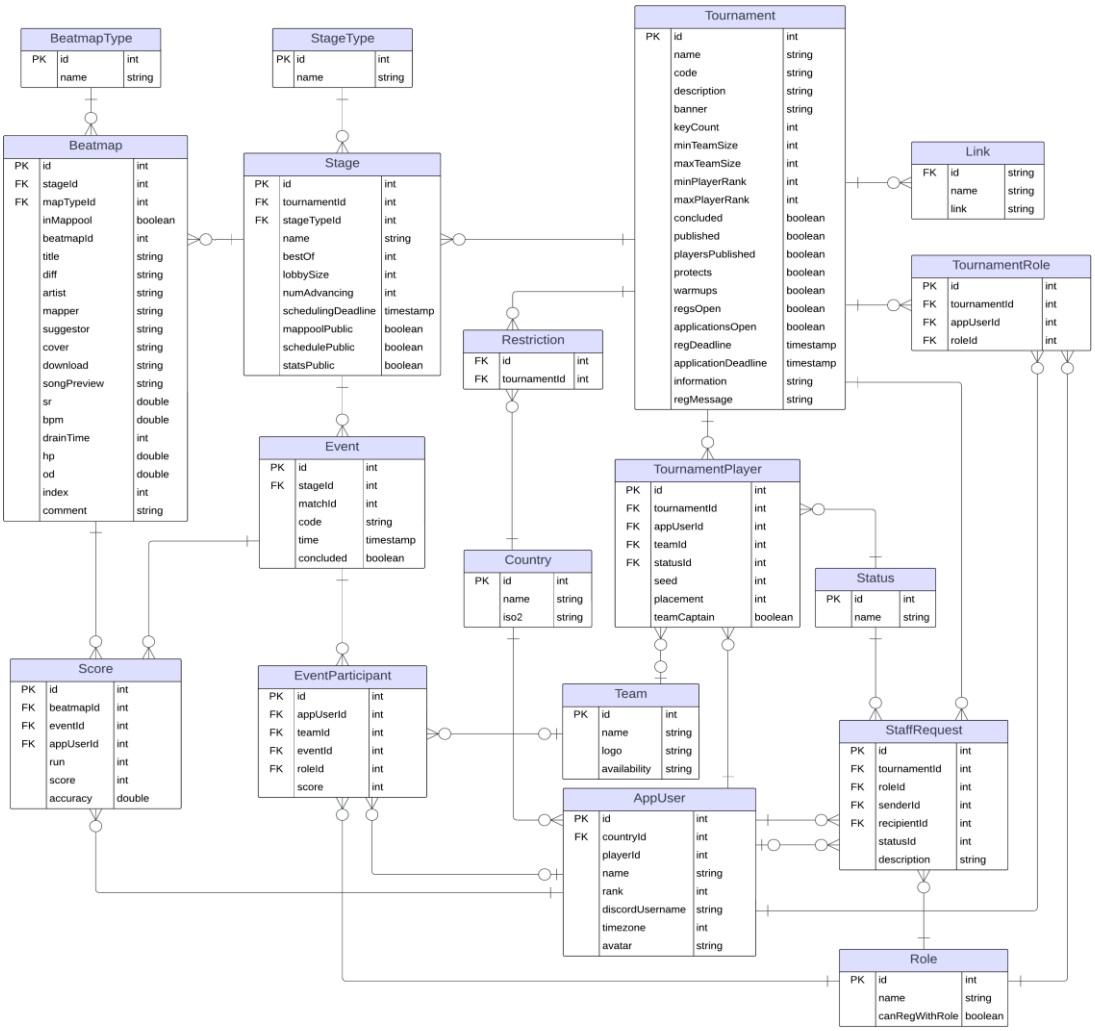
Turniiride haldamisega seotud tegevused nõuavad kasutaja autentimist, mistõttu tuleb hoida esirakenduses viisi kasutajat tuvastada. Selle saavutamiseks kasutatakse küpsiseid. Sisselogimisel suunatakse kasutaja osu! sisselogimisteenuse poole. Pärast andmete sisestamist suunatakse kasutaja tagasi rakenduse kodulehele ning määratakse küpsis. Tagarakenduse poole API päringute tegemisel pannakse seejärel iga päringuga kaasa ka küpsis, kasutades selleks Axios *withCredentials* parameetrit.

Kuna kasutaja õigused on turniiripõhised, on vajalik hoida esirakenduses ka andmeid kõikide temaga seotud rollide kohta. Lisaks peaks parema kasutajakogemuse

saavutamiseks jääma kasutaja sisselogituks ka lehe värskendamisel. Selle saavutamiseks päritakse pärast sisselogimist tagarakenduse API käest kasutaja andmed ning vastus salvestatakse *localStorage* andmehoidlasse, mis säilitab sinna salvestatud väärtuseid ka lehe värskendamisel. Kasutaja andmed sisaldavad endas kõiki tema rolle ning mängija staatustega seotud informatsiooni, mida on võimalik autoriseerimiseks kasutada. Väljalogimisel või küpsise aegumisel andmehoidla tühjendatakse.

5.3 Andmemudel

Enne rakenduse arendusprotsessi alustamist on mõistlik panna paika kasutatavate andmete struktuur. Ühtse andmestruktuuri jälgimine hõlbustab rakenduse erinevate osade vahelist suhtlust. Andmete struktuuri visualiseerimiseks on loodud olemi-suhte diagramm (vt Joonis 9), kasutades selleks tööriista Lucidchart.



Joonis 9. Rakenduse olemi-suhte diagramm.

Andmemudel määrab ära läbiviidavate turniiride struktuuri. Turniiride peamised andmed on välja toodud tabelis *Tournament*. Turniirid koostnevad tabelis *Stage* kirjeldatud etappidest. Igal etapil on ka tüüp, mis on määratud tabelis *StageType*. Etapid koosnevad omakorda nendes mängitavatest tasemetest (tabel *Beatmap*) ning nendes toimuvatest sündmustest (tabel *Event*). Viimase alla kuuluvad nii matšid kui ka kvalifikatsioonietapi *lobby*-d. Sarnaselt etappidega on ka igal mängitaval tasemel kindel tüüp, mis on kirjeldatud tabelis *BeatmapType*.

Mängitavate tasemete ja sündmustega on seotud ka skoorid, mis on kirjeldatud tabelis *Score* ning mis on olulised statistika väljanäitamiseks ja *seeding*-u genereerimiseks. Igal sündmusel on mitu osalejat, kelleks võib olla kas üksik mängija, meeskond või personaliliige. Vastav suhe määratakse tabelis *EventParticipant*, mis seob omavahel sündmuse ning sellel osalejad. Seoses määratakse ka osapoolte rollid antud sündmuses.

Kasutajate andmeid hoitakse tabelis *AppUser*. Kuna rakenduses on sisselogimiseks kasutusel OAuth2 protokoll, ei ole tarvis hoida andmebaasis kasutajate paroole. Sellest tulenevalt ei sisalda *AppUser* tabel kasutaja kohta tundlikku informatsiooni.

Kasutajate rollid ning mängija staatused erinevates turniirides on ära kirjeldatud kahes tabelis, milleks on *TournamentRole* ja *TournamentPlayer*. Neist esimene määrab ära suhte kasutaja, turniiri ja nendega seotud rolli vahel ning on aluseks ligipääsuõiguste määramisele. Tabel *TournamentPlayer* sisaldab kasutaja mängija rolliga seotud staatusteid turniirides, kuhu kasutaja end mängijana registreerinud on. Siia kuuluvad seos tiimiga, seeme, lõpp-koht turniiris ning mängija staatus.

AppUser tabel on seotud ka tabeliga *StaffRequest*, mis kirjeldab ära kasutajaga seotud personalitaotlused ja -kutsed. Suhe kasutajaga on määratud läbi kahe seose, *senderId* ja *recipientId*. Antud seosed määravad ära personalikutse saatja ning selle kättesaaja. Personalitaotluse puhul on *recipientId* väli tühi, kuna need saadetakse kindla kasutaja asemel turniirile.

5.4 Rakenduse testimine

Rakenduse testimine viidi läbi käsitsi. Automaattestide kirjutamine jäi tulenevalt lõputöö kirjutamiseks antud aja piiratusest ja loodava rakenduse suurest mahust skoobist välja. Käsitsi testimise läbiviimise hõlbustamiseks sisestati eelnevalt andmebaasi suurel hulgal

testandmeid, mis sisaldasid nii osu! kasutajaid kui ka valmis turniire koos etappide, mängitavate tasemete ning kõigi muude vajalike komponentidega.

Testimise käigus veenduti, et kõik rakenduse nõutetest lähtuvalt loodud funktsionaalsused toimiksid korrektselt. Testiti uue turniiri loomist ning selle väärtuste ja staatuse muutmist, mängijate ja personali registreerimist ning etappide, mängitavate tasemete ja etappides toimuvate sündmuste haldamist. Testiti ka matšide ja kvalifikatsiooni *lobby*-de läbiviimist ning veenduti, et pärast iga sündmuse läbiviimist toimuks ka turniiri statistika uuenemine. Viimaks testiti *seeding*-u genereerimist ning veenduti, et mängijatele arvutatud seeme oleks vastavuses turniiri statistikas nähtava kvalifikatsiooni pingereaga. Kõik eelmainitud tegevused viidi läbi nii soolo- kui ka meeskonnaturniirides, kuna rakendus võimaldab mõlemat tüüpi turniiride läbiviimist.

Lisaks funktsionaalsuste testimisele kontrolliti ka kõiki kasutaja rollidega seonduvaid piiranguid. Veenduti selles, et kasutaja saaks näha vaid andmeid ja teha vaid tegevusi, mille tarvis on tal sobivad õigused. Kontrolliti ka turniiri ja selle osade nähtavuse taseme korrektset toimimist, kuna privaatseks märgitud andmeid peavad saama vaadata vaid kindla rolliga kasutajad. Viimaks testiti ka vigaste päringute saatmist, mille käigus kontrolliti tagarakenduse poolt õige erindi viskamist ning esirakenduse poolel vastava veateate kasutajale väljanäitamist.

6 Tulemuste analüüs

Käesolevas peatükis analüüsitakse lõputöö raames loodud rakenduse funktsionaalsuseid ja selle mittefunktsionaalseid aspekte ning uuritakse erinevaid viise rakenduse edasiarenduseks. Funktsionaalsuste loomisel lähtuti analüüsi käigus paika pandud nõuetest (vt peatükki 4.1).

6.1 Loodud funktsionaalsused

Rakenduse olulisimaks funktsionaalsuseks on osu!mania turniiride organiseerimine, läbi viimine ning nendega seotud andmete haldamine. Turniiri organiseerimine algab selle loomisega, mida saab sisse logitud kasutaja teha otse kodulehelt. Pärast turniiri loomist saab organiseerija lisada sellele etappe ning neid hallata. Etappidesse saab seejärel lisada mängitavaid tasemeid ning sündmuseid, milleks on sõltuvalt antud etapi tüübist kas matšid või kvalifikatsiooni *lobby*-d. Viimaks pakub rakendus nii eelmainitud sündmuste kui ka erinevate turniiriga seonduvate protsesside, nagu näiteks *seeding* ja *mappooling*, läbiviimist. Eelmainitud tegevuste läbiviimine on tehtud võimalikult lihtsaks ega nõua suurel hulgal olemasolevaid teadmisi.

Etappides toimuvate sündmuste läbiviimiseks on loodud veebilehed, mille ülesehituse ja disaini määramisel on võetud inspiratsiooni osu! turniirides tihti kasutatavatest kohtunike lehtedest [10]. Antud lehtedel saavad kohtunikud kergesti sündmuste tulemusi kirja panna ning nende kulgu jälgida. Sündmuse lõpetamisel tuleb lisada sellega seonduv osu! matši identifikaator, misjärel salvestatakse tulemused ning uuendatakse automaatselt turniiri statistikat. Kogu sündmuse läbiviimise protsess on tehtud nii lihtsaks kui võimalik ning kohtuniku poolt käsitsi sisestatavate andmete hulk on viidud võimalikult madalale. Näiteks ei pea kohtunik sisestama käsitsi mängijate skoore, vaid ainult tasemete võitjad. Mängijate skoorid päritakse automaatselt osu! API poolt. Samuti ei pea kohtunik käsitsi importima mängitavate tasemete nimekirja ega muid vajalikke algandmeid.

Rakendus toetab turniiri mängijate, meeskondade ja personaliliikmete registreerimist ja haldamist. Kasutaja saab registreerida end mõne nupuvajutusega turniiri mängijaks.

Meeskonnaturniiri puhul tuleb korraga registreerida kõik meeskonna mängijad, mida teeb meeskonna kapten. Pärast registreerimist lisatakse kasutajad automaatselt turniiri mängijate nimekirja. Säärane lahendus eemaldab vajaduse kasutada registreerimiseks eraldi vormide loomise keskkondi ega nõua mängijate andmete käsitsi töötlemist.

Personalirolli saamiseks tuleb kasutajal saata turniirile personalitaotlus, milles on märgitud soovitud roll ning mida saab organiseerija seejärel kas vastu võtta või tagasi lükata. Vajaduse korral on organiseerijal võimalik personaliliikmeid ka turniirilt eemaldada. Lisaks saab organiseerija välja saata välja individuaalseid personalikutseid. Antud lahendus võimaldab registreerida personaliliikmeid isegi siis, kui turniir pole veel avalikustatud. Personalitaotluste ja -kutsete kasutamine aitab lihtsustada personali kogumise protsessi.

Rakenduse kodulehel on nähtaval kõik hetkel käivad ja varem toimunud turniirid, mis pole märgitud organiseerija poolt privaatseks. Lisaks on kasutajal võimalus turniire erinevate parameetrite alusel filtreerida, mis lihtsustab turniiride leidmise ja jälgimise protsessi. Kasutajad saavad jälgida ka nii enda kui ka teiste kasutajate rolle ja mängijate staatusi, avades selleks vastava kasutaja profiili. Omaenda profiililt näeb kasutaja lisaks eelmainitule ka tema poolt saadetud personalitaotluseid koos staatustega ning talle saadetud personalikutseid.

Rakendus on integreeritud osu! API-ga, võimaldades osu! kasutajaga sisselogimist, läbi API uute tasemete loomist või nende muutmist osu! identifikaatori abil, ning etappides toimuvate matšide ja kvalifikatsiooni *lobby*-de tulemuste salvestamist. Rakenduse osu! API-ga integreerimine vähendab märkimisväärselt käsitsi andmete sisestamise vajadust ning automatiseerib turniiri statistika loomise protsessi. Päringute hulga minimeerimise huvides tehakse neid vaid eelmainitud tegevuste puhul. Kõikide muude tegevuste sooritamiseks ja turniiriga seonduvate andmete välja näitamiseks kasutatakse rakenduse andmebaasi. See aitab vältida osu! API liigset koormamist, mis on üks rakenduse mittefunktsionaalsetest nõuetest.

Tulenevalt lõputöö eesmärgist mängib rakenduses olulist rolli kasutajaliides, mille üldised disainiprintsiibid määrati analüüsi käigus (vt peatükki 4.6). Kasutajaliidese puhul keskenduti lähtuvalt mittefunktsionaalsetest nõuetest informatsiooni leitavusele ning tegevuste sooritamise intuitiivsusele ja automatiseeritusele. Ülesehituse loomisel võeti

inspiratsiooni erinevatest olemasolevatest osu! turniiride kodulehtedest. Kasutajaliidese välimus on loodud ka võimalikult sidus, et lihtsustada selle õpitavust. Rakenduse kasutajaliides sisaldab palju vaateid, millest olulisemad on välja toodud peatükis Lisa 2.

6.2 Edasiarenduse võimalused

Lõputöö raames valminud rakenduse puhul on tegu prototüübiga ning seda on võimalik veel mitmel viisil täiendada. Autori hinnangul kõige olulisemaks edasiarenduseks oleks rakenduse laiendamine peale osu!mania ka teistele osu! mängurežiimidele, mis suurendaks oluliselt rakenduse potensiaalsete kasutajate arvu. Eriti kasulik oleks lisada osu!standard turniiride loomise tugi, kuna see on neljast osu! mängurežiimist kõige populaarsem ning sellele korraldatakse kõige rohkem turniire. Rakenduse säärasel laiendamisel on oluline pidada silmas vastava mängurežiimi ja selle turniiride eripärasid.

Rakendus võimaldab organiseerida vaid lihtsama ülesehitusega turniire ning seega oleks autori hinnangul oluline lisada tuge suuremale hulgale turniiriformaatidele. Süsteemi loomine, mis võimaldaks mistahes meelevaldsete reeglitega turniiri organiseerimist, ei ole aga realistlik. Sellele vaatamata on võimalik katta ära peaaegu kõik vähegi populaarsemad formaadid. Siia kuuluvad näiteks *mapping* turniirid ehk osu! tasemete loomise turniirid, segaturniirid (võistlevad nii soolomängijad kui ka meeskonnad) ning mitmed muud formaadid. Rohkematele turniiriformaatidele toe lisamine aitaks jällegi suurendada rakenduse potensiaalsete kasutajate hulka.

Lisaks turniiride üldisele formaadile on ka oluline, et organiseerijad saaksid turniiride reegleid ja nendes kasutatavaid arvutusmeetodeid võimalikult palju enda vajadustele vastavalt kohandada. Näiteks on *seeding*-u genereerimiseks võimalik kasutada mitmeid erinevaid arvutusmeetodeid, kuid hetkel kasutab rakendus neist vaid ühte lihtsamat. Paindlikkuse suurendamiseks saaks pakkuda organiseerijatele võimalust valida erinevate *seeding*-u arvutusmeetodite vahel. Sarnaselt eelmainitule oleks kasulik suurendada paindlikkust ka turniiride reeglite määramise osas, et anda organiseerijatele nende üle võimalikult suur kontroll.

Loodud rakenduses on ainsaks turniiride välimuse kohandamise võimaluseks bänneri pildi muutmine, mistõttu näevad kõik turniirid väga sarnased välja. Kuigi veebilehtede ülesehitust ei ole võimalik drastiliselt muuta, oleks siiski võimalik anda turniiride

organiseerijatele mõningaid viise turniiride välimuse omanäolisemaks muutmiseks, võimaldades näiteks turniiride alamlehtedel kasutatavate värvide muutmist, antud lehtedele taustapildi lisamist või muud sarnast lahendust.

Viimaks saaks täiendada rakendust erinevate väiksemate lisafunktsionaalsustega, mis suurendaks selle kasulikkust ja paindlikkust turniiride organiseerimisel ning parandaks kasutajakogemust. Mõned näited võimalikest lisafunktsionaalsustest oleks turniiridele põhjalikuma statistika loomine, mängijate skooridele detailsema vaate lisamine ning *seeding*-u tulemuse põhjal matšide ajakava genereerimine. Lisaks eelmainitule oleks rakenduse kasutatavuse parandamiseks mõistlik lisada ka heleda ja tumeda kasutajaliidese vahel lülitumise funktsionaalsus ning parandada kasutajaliidese kohanduvust mobiilsetes seadmetes.

7 Kokkuvõte

Käesoleva lõputöö eesmärgiks oli veebirakenduse prototüübi loomine, mis lihtsustaks osu!mania turniiride organiseerimist ja läbiviimist ning võimaldaks mugavalt jälgida hetkel käivaid ja varem toimunud turniire. Rakendus oleks kasulik väiksemate turniiride korraldajatele, kelle jaoks on olemasolevad lahendused liigselt keerulised või ajakulukad.

Lõputöö tulemusena valmis veebirakenduse prototüüp, mis võimaldab organiseerida ja läbi viia lihtsama ülesehitusega osu!mania turniire ning hallata nendega seotud mängijaid ja personali. Loodud rakendus viib kokku kõik analüüsi käigus uuritud olemasolevate lahenduste peamised funktsionaalsused läbi ühtse kasutajaliidese. Lisaks lihtsustab see paljude turniiride organiseerimisega seonduvate tegevuste sooritamist ning vähendab tänu mitmete protsesside automatiseeritusele ka käsitsi tehtava töö hulka.

Valminud rakendus täidab kõiki analüüsi käigus seatud nõudeid, vastab määratud skoobile ning lahendab seega autori hinnangul lõputöö raames püstitatud probleemi. Sellele vaatamata on tegu prototüübiga, mida on võimalik veel mitmel moel täiendada. Näiteks saaks laiendada rakendust lisaks osu!maniale ka teistele osu! mängurežiimidele, lisada rohkem paindlikkust erinevate turniiriformaatide loomisel ja turniiride reeglite määramisel ning pakkuda erinevaid viise turniiride välimuse kohandamiseks.

Kasutatud kirjandus

[1] D. Herbert, „Welcome | osu!,“ [Võrgumaterjal]. Saadaval: https://osu.ppy.sh/. [Kasutatud 10. jaanuar 2024].

[2] D. Herbert, „Game Mode / osu!mania · wiki | osu!,“ [Võrgumaterjal]. Saadaval: https://osu.ppy.sh/wiki/en/Game_mode/osu%21mania. [Kasutatud 10. jaanuar 2024].

[3] Whisper, „Rhythm Games,“ Rainy Whispers, 7. Veebruar 2019. [Võrgumaterjal]. Saadaval: https://rainywhispers.home.blog/2019/02/07/vsrg/. [Kasutatud 10. jaanuar 2024].

[4] X. Li, M. Zheng, Y. Zhang, Y. Wang, L. Nie, Y. Yuan, T. Qian ja Y. Ku, „Music-based casual video game training alleviates symptoms of subthreshold depression,“ *Front. Public Health*, kd. 10, 3. August 2022.

[5] STL Rock School, „Are Rhythm Games Good for You? Unveiling the Benefits of the Beat,“ Medium, 29. Juuni 2023. [Võrgumaterjal]. Saadaval: https://medium.com/@adam_8474/are-rhythm-games-good-for-you-unveiling-the-benefits-of-the-beat-d152fb66b1de. [Kasutatud 10. jaanuar 2024].

[6] Reality Remake, „Beat Saber Is A Good Workout - Beat Saber Is Great Arm Exercise,“ [Võrgumaterjal]. Saadaval: https://www.realityremake.com/articles/beat-saber-is-a-good-workout-beat-saber-is-great-arm-exercise. [Kasutatud 10. jaanuar 2024].

[7] D. Herbert, „osu!mania World Cup / osu!mania World Cup 2014 · wiki | osu!,“ [Võrgumaterjal]. Saadaval: https://osu.ppy.sh/wiki/en/Tournaments/MWC/2014. [Kasutatud 10. jaanuar 2024].

[8] D. Herbert, „Tournaments | osu!,“ [Võrgumaterjal]. Saadaval: https://osu.ppy.sh/community/tournaments. [Kasutatud 10. jaanuar 2024].

[9] „6K AST Master Sheet,“ [Võrgumaterjal]. Saadaval: https://docs.google.com/spreadsheets/d/1WWtioo56lCEcQzLxSDQp2DbtfiLyBQtflmDLnrC_Gk8/edit. [Kasutatud 10. jaanuar 2024].

[10] IceDynamix, „Ref Sheet Template v2,“ 6. mai 2022. [Võrgumaterjal]. Saadaval: https://docs.google.com/spreadsheets/d/1lZjp9EFqAFCgoAV1ZoOxlxg5uhEATEI_ZPCC-i_6QFM/edit#gid=262741559. [Kasutatud 12. jaanuar 2024].

[11] D. Herbert, „Tournaments · forum | osu!,“ [Võrgumaterjal]. Saadaval: https://osu.ppy.sh/community/forums/55. [Kasutatud 10. jaanuar 2024].

[12] Ill onion, „Tournament Hosting Guide,“ [Võrgumaterjal]. Saadaval: https://docs.google.com/document/d/1aveFDrzwC9TiRrHAsDfRW0bVSKs3JY-v8TNmN0kB484/edit. [Kasutatud 12. jaanuar 2024].

[13] Nathaniel, „osu! Comprehensive Tournament Role Guide,“ [Võrgumaterjal]. Saadaval:

https://docs.google.com/document/d/1ynEItqDBZYp9CVuFuJAJ6WBPLm20AacOrdiRGolUpEA/edit. [Kasutatud 12. jaanuar 2024].

[14] Underjoy, „osu!mania Mappool Creation: Analysis and Opinion,“ [Võrgumaterjal]. Saadaval: https://docs.google.com/document/d/13J3D2ZBAJHccwBQd3OyvBf8qU5PKQUbWaAG4M9ruesc/edit. [Kasutatud 12. jaanuar 2024].

[15] „Global Taiko Showdown,“ [Võrgumaterjal]. Saadaval: https://gtsosu.com/. [Kasutatud 11. jaanuar 2024].

[16] D I O, „The Perennial,“ [Võrgumaterjal]. Saadaval: https://theperennial.net/. [Kasutatud 16. detsember 2024].

[17] W. Chai, „Google Sheets,“ TechTarget, mai 2021. [Võrgumaterjal]. Saadaval: https://www.techtarget.com/whatis/definition/Google-Spreadsheets. [Kasutatud 11. jaanuar 2024].

[18] L. Henderson, „Google Sheets vs. Excel: 2024 Unbiased Comparison (+ User Reviews),“ Nifty, 29. november 2023. [Võrgumaterjal]. Saadaval: https://niftypm.com/blog/google-sheets-vs-excel/. [Kasutatud 11. jaanuar 2024].

[19] M. Adielsson, R. Barnes, P. Kupfer, I. Roberts ja J. H. Weber, „Google Sheets function list,“ Google, [Võrgumaterjal]. Saadaval: https://support.google.com/docs/table/25273?hl=en. [Kasutatud 11. jaanuar 2024].

[20] Omkelderman, „osu! api fetch stuff for google scripts,“ [Võrgumaterjal]. Saadaval: https://gist.github.com/omkelderman/037342ca6612140197d0bb6f19328884. [Kasutatud 11. jaanuar 2024].

[21] A. Kayode-Sanni, „Google Forms: Advantages & Disadvantages In 2022,“ Formplus, 22. november 2022. [Võrgumaterjal]. Saadaval: https://www.formpl.us/blog/google-forms-advantages-disadvantages-in-2022. [Kasutatud 11. jaanuar 2024].

[22] L. Tennyson, „How to connect Google Forms to Google Sheets,“ Sheetgo, [Võrgumaterjal]. Saadaval: https://blog.sheetgo.com/google-sheets-features/how-to-connect-google-forms-to-google-sheets/. [Kasutatud 11. jaanuar 2024].

[23] ITrobes, „What are the Advantages and Disadvantages of Web Applications?,“ 6. aprill 2023. [Võrgumaterjal]. Saadaval: https://www.itrobes.com/what-are-the-advantages-and-disadvantages-of-web-applications/. [Kasutatud 13. jaanuar 2024].

[24] D. Herbert, „osu!api · wiki | osu!,“ [Võrgumaterjal]. Saadaval: https://osu.ppy.sh/wiki/en/osu%21api. [Kasutatud 19. jaanuar 2024].

[25] Ppy, „osu!web,“ [Võrgumaterjal]. Saadaval: https://github.com/ppy/osu-web. [Kasutatud 19. jaanuar 2024].

[26] „osu!web Documentation,“ [Võrgumaterjal]. Saadaval: https://osu.ppy.sh/docs/index.html. [Kasutatud 19. jaanuar 2024].

[27] A. Starreveld, „Understanding OAuth2,“ Medium, 2. august 2023. [Võrgumaterjal]. Saadaval: https://medium.com/web-security/understanding-oauth2-a50f29f0fbf7. [Kasutatud 19. jaanuar 2024].

[28] B. Gorman, „Why Your Organization Should be Using OAUTH 2.0,“ Clowder, 30. juuli 2019. [Võrgumaterjal]. Saadaval: https://www.clowder.com/post/why-your-organization-should-be-using-oauth-2.0. [Kasutatud 19. jaanuar 2024].

[29] D. Hardt, „The OAuth 2.0 Authorization Framework,“ Internet Engineering Task Force, oktoober 2012. [Võrgumaterjal]. Saadaval: https://datatracker.ietf.org/doc/html/rfc6749. [Kasutatud 19. jaanuar 2024].

[30] B. Anderson ja B. Nicholson, „SQL vs. NoSQL Databases: What’s the Difference?,“ IBM, 12. juuni 2022. [Võrgumaterjal]. Saadaval: https://www.ibm.com/blog/sql-vs-nosql/. [Kasutatud 25. jaanuar 2024].

[31] DB-Engines, „DB-Engines Ranking,“ Solid IT, jaanuar 2024. [Võrgumaterjal]. Saadaval: https://db-engines.com/en/ranking. [Kasutatud 25. jaanuar 2024].

[32] S. Ravoof, „PostgreSQL vs MySQL: Explore Their 12 Critical Differences,“ Kinsta, 29. detsember 2023. [Võrgumaterjal]. Saadaval: https://kinsta.com/blog/postgresql-vs-mysql/. [Kasutatud 25. jaanuar 2024].

[33] Qasim, „SQLite vs MySQL: Find The Best Database Management System For 2024,“ RedSwitches, 27. november 2023. [Võrgumaterjal]. Saadaval: https://www.redswitches.com/blog/sqlite-vs-mysql/. [Kasutatud 25. jaanuar 2024].

[34] TIOBE, „TIOBE Index for January 2024,“ [Võrgumaterjal]. Saadaval: https://www.tiobe.com/tiobe-index/. [Kasutatud 22. jaanuar 2024].

[35] Y. Shiotsu, „What Is PHP and Why Should You Use It?,“ Upwork, 21. detsember 2021. [Võrgumaterjal]. Saadaval: https://www.upwork.com/resources/why-use-php. [Kasutatud 23. jaanuar 2024].

[36] Javatpoint, „Advantages and disadvantages of Java,“ [Võrgumaterjal]. Saadaval: https://www.javatpoint.com/advantages-and-disadvantages-of-java. [Kasutatud 23. jaanuar 2024].

[37] Gradle Inc., „Gradle Build Tool Features,“ [Võrgumaterjal]. Saadaval: https://gradle.org/features/. [Kasutatud 23. jaanuar 2024].

[38] AltexSoft, „The Good and the Bad of C# Programming,“ 29. oktoober 2021. [Võrgumaterjal]. Saadaval: https://www.altexsoft.com/blog/c-sharp-pros-and-cons/. [Kasutatud 23. jaanuar 2024].

[39] Y. Gavrilova, „Pros and Cons of Python Programming Language,“ Serokell, 31. oktoober 2023. [Võrgumaterjal]. Saadaval: https://serokell.io/blog/python-pros-and-cons. [Kasutatud 23. jaanuar 2024].

[40] DDI Development, „Advantages and disadvantages of Laravel Framework for web Development,“ juuli 2020. [Võrgumaterjal]. Saadaval: https://ddi-dev.com/blog/programming/pros-and-cons-of-laravel-framework-for-web-app-development/. [Kasutatud 23. jaanuar 2024].

[41] Microsoft, „What is Java Spring Boot?,“ [Võrgumaterjal]. Saadaval: https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-java-spring-boot. [Kasutatud 23. jaanuar 2024].

[42] VMware Tanzu, „Spring Security,“ [Võrgumaterjal]. Saadaval: https://spring.io/projects/spring-security/. [Kasutatud 23. jaanuar 2024].

[43] K. Kara, „Authentication and Authorization in ASP.NET Core: A Comprehensive Guide,“ Medium, 16. detsember 2023. [Võrgumaterjal]. Saadaval: https://medium.com/@kerimkkara/authentication-and-authorization-in-asp-net-core-a-comprehensive-guide-dfb8fb806ac7. [Kasutatud 23. jaanuar 2024].

[44] M. Deery, „What Is the Django Framework? The Complete Beginner's Guide," Careerfoundry, 30. august 2023. [Võrgumaterjal]. Saadaval: https://careerfoundry.com/en/blog/web-development/django-framework-guide/. [Kasutatud 23. jaanuar 2024].

[45] MDN, „What is JavaScript?," Mozilla, [Võrgumaterjal]. Saadaval: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript. [Kasutatud 24. jaanuar 2024].

[46] TypeScript, „TypeScript for the New Programmer," Microsoft, [Võrgumaterjal]. Saadaval: https://www.typescriptlang.org/docs/handbook/typescript-from-scratch.html. [Kasutatud 24. jaanuar 2024].

[47] S. Greif ja E. Burel, „State of JavaScript 2022: Front-end Frameworks," [Võrgumaterjal]. Saadaval: https://2022.stateofjs.com/en-US/libraries/front-end-frameworks/. [Kasutatud 22. jaanuar 2024].

[48] A-Team Global, „WHY IS REACT SO POPULAR THESE DAYS?," 31. märts 2023. [Võrgumaterjal]. Saadaval: https://a-team.global/blog/why-is-react-so-popular/. [Kasutatud 24. jaanuar 2024].

[49] AltexSoft, „The Good and the Bad of Vue.js Framework Programming," 28. september 2022. [Võrgumaterjal]. Saadaval: https://www.altexsoft.com/blog/pros-and-cons-of-vue-js/. [Kasutatud 24. jaanuar 2024].

[50] AltexSoft, „The Good and the Bad of Angular Development," 6. september 2022. [Võrgumaterjal]. Saadaval: https://www.altexsoft.com/blog/the-good-and-the-bad-of-angular-development/. [Kasutatud 24. jaanuar 2024].

[51] React, „Using TypeScript," Meta, [Võrgumaterjal]. Saadaval: https://react.dev/learn/typescript. [Kasutatud 24. jaanuar 2024].

[52] Vue.js, „Using Vue with TypeScript," [Võrgumaterjal]. Saadaval: https://vuejs.org/guide/typescript/overview#. [Kasutatud 24. jaanuar 2024].

[53] S. Chacon ja B. Straub, „Getting Started - About Version Control," *Pro Git*, 2nd ed. Apress, 2014, pp. 10-13.

[54] G2, „Best Version Control Hosting Software," [Võrgumaterjal]. Saadaval: https://www.g2.com/categories/version-control-hosting. [Kasutatud 26. jaanuar 2024].

[55] V. Ilyukha, „GitHub vs GitLab vs BitBucket," Jelvix, [Võrgumaterjal]. Saadaval: https://jelvix.com/blog/bitbucket-vs-github-vs-gitlab. [Kasutatud 26. jaanuar 2024].

[56] A. Manchanda, „What is Software Architecture in Software Engineering?," Net Solutions, 12. aprill 2023. [Võrgumaterjal]. Saadaval: https://www.netsolutions.com/insights/why-software-architecture-matters-to-build-scalable-solutions/. [Kasutatud 20. jaanuar 2024].

[57] Bharath, „The Clean Architecture—Beginner's Guide," Medium, 4. jaanuar 2022. [Võrgumaterjal]. Saadaval: https://betterprogramming.pub/the-clean-architecture-beginners-guide-e4b7058c1165. [Kasutatud 20. jaanuar 2024].

[58] Amazon Web Services, „What is a RESTful API?," [Võrgumaterjal]. Saadaval: https://aws.amazon.com/what-is/restful-api/. [Kasutatud 20. jaanuar 2024].

[59] M. Richards, *Software Architecture Patterns*. Sebastopol, CA: O'Reilly Media, 2015.

[60] F. Salas, „Efficient Data Transfer in REST APIs: A Deep Dive into the DTO Pattern with Spring Boot and MySQL,“ Medium, 11. juuni 2023. [Võrgumaterjal]. Saadaval: https://blog.stackademic.com/efficient-data-transfer-in-rest-apis-a-deep-dive-into-the-dto-pattern-with-spring-boot-and-mysql-df2bdf1ece74. [Kasutatud 20. jaanuar 2024].

[61] Design Studio, „Dark Mode UI Design Best Practices,“ 27. november 2023. [Võrgumaterjal]. Saadaval: https://www.designstudiouiux.com/blog/dark-mode-ui-design-best-practices/. [Kasutatud 21. jaanuar 2024].

[62] „Figma: The Collaborative Interface Design Tool,“ Figma, [Võrgumaterjal]. Saadaval: https://www.figma.com/. [Kasutatud 21. jaanuar 2024].

[63] MUI, A. Stefan ja O. Tassinari, „Material UI for Figma (and MUI X),“ Figma, [Võrgumaterjal]. Saadaval: https://www.figma.com/community/file/912837788133317724/material-ui-for-figma-and-mui-x. [Kasutatud 21. jaanuar 2024].

[64] Ganeshchowdharysadanala, „Spring Initializr,“ GeeksforGeeks, 8. november 2023. [Võrgumaterjal]. Saadaval: https://www.geeksforgeeks.org/spring-initializr/. [Kasutatud 18. oktoober 2024].

[65] JigNect Technologies Pvt Ltd, „Lombok Unleashed: Elevating Java Efficiency with Getters, Setters, Constructors, Builders, and More,“ [Võrgumaterjal]. Saadaval: https://jignect.tech/lombok-unleashed-elevating-java-efficiency-with-getters-setters-constructors-builders-and-more/. [Kasutatud 17. oktoober 2024].

[66] Baeldung, „Java Bean Validation Basics,“ 15. juuni 2024. [Võrgumaterjal]. Saadaval: https://www.baeldung.com/java-validation. [Kasutatud 17. oktoober 2024].

[67] M. Tyson, „What is JPA? Introduction to Java persistence,“ InfoWorld, 20. mai 2022. [Võrgumaterjal]. Saadaval: https://www.infoworld.com/article/2259807/what-is-jpa-introduction-to-the-java-persistence-api.html. [Kasutatud 17. oktoober 2024].

[68] V. Mihalcea, „Schema Migration with Hibernate and FlywayDB,“ SitePoint, 3. oktoober 2016. [Võrgumaterjal]. Saadaval: https://www.sitepoint.com/schema-migration-hibernate-flywaydb/. [Kasutatud 18. oktoober 2024].

[69] M. Behler, „Spring Security: Authentication and Authorization In-Depth,“ 30. mai 2022. [Võrgumaterjal]. Saadaval: https://www.marcobehler.com/guides/spring-security. [Kasutatud 9. oktoober 2024].

[70] CyberChimps Inc., „What Are Session Cookies? – A Comprehensive Guide for 2024,“ 18. jaanuar 2024. [Võrgumaterjal]. Saadaval: https://cyberchimps.com/blog/session-cookies/. [Kasutatud 18. oktoober 2024].

[71] H. Bhattbhatt, „Securing Front-End Applications: The Case for Server-Side Token Management,“ Medium, 3. märts 2024. [Võrgumaterjal]. Saadaval: https://harish-bhattbhatt.medium.com/securing-front-end-applications-the-case-for-server-side-token-management-e0ec69de3a8c. [Kasutatud 18. oktoober 2024].

[72] Broadcom, „WebClient,“ [Võrgumaterjal]. Saadaval: https://docs.spring.io/spring-framework/reference/web/webflux-webclient.html. [Kasutatud 9. oktoober 2024].

[73] C. Thai, „Mastering Modern UI Development: A Comprehensive Guide to Using Material-UI with React,“ DEV, 1. juuni 2024. [Võrgumaterjal]. Saadaval: https://dev.to/christopherthai/mastering-modern-ui-development-a-comprehensive-guide-to-using-material-ui-with-react-9d6. [Kasutatud 18. oktoober 2024].

[74] Simplilearn, „Why Should You Use a Router in React.js?,“ 25. juuli 2024. [Võrgumaterjal]. Saadaval: https://www.simplilearn.com/tutorials/reactjs-tutorial/routing-in-reactjs. [Kasutatud 18. oktoober 2024].

[75] Zenoamaro, „react-quill,“ [Võrgumaterjal]. Saadaval: https://github.com/zenoamaro/react-quill. [Kasutatud 18. oktoober 2024].

[76] M. Kosisochukwu, „Integrating Formik & Yup for React Form Validation,“ Pieces, 2. jaanuar 2023. [Võrgumaterjal]. Saadaval: https://pieces.app/blog/react-form-validation-formik-yup. [Kasutatud 18. oktoober 2024].

[77] J. Ofoegbu, „How to Use JSON Server for Front-end Development,“ freeCodeCamp, 21. august 2023. [Võrgumaterjal]. Saadaval: https://www.freecodecamp.org/news/json-server-for-frontend-development/. [Kasutatud 16. oktoober 2024].

[78] J. J. Sarjeant, „Axios,“ [Võrgumaterjal]. Saadaval: https://axios-http.com/. [Kasutatud 14. oktoober 2024].

Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[1]

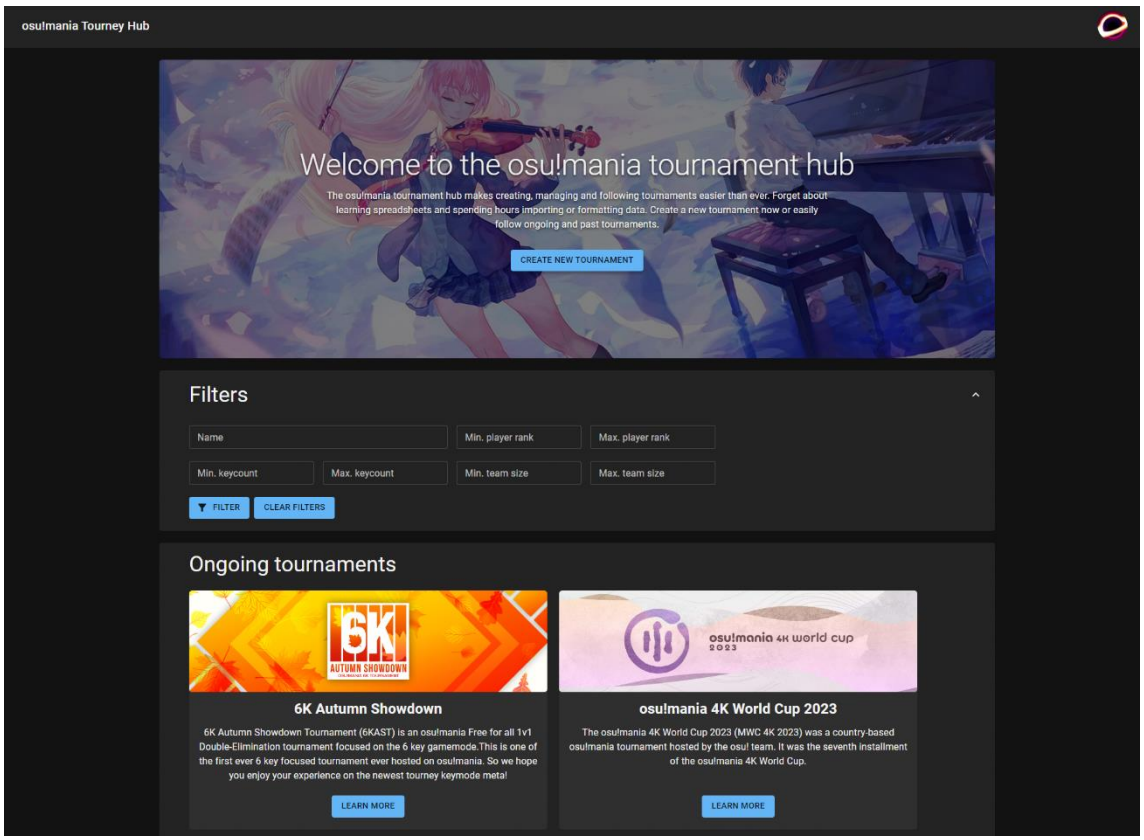
Mina, Erik Martin Murde

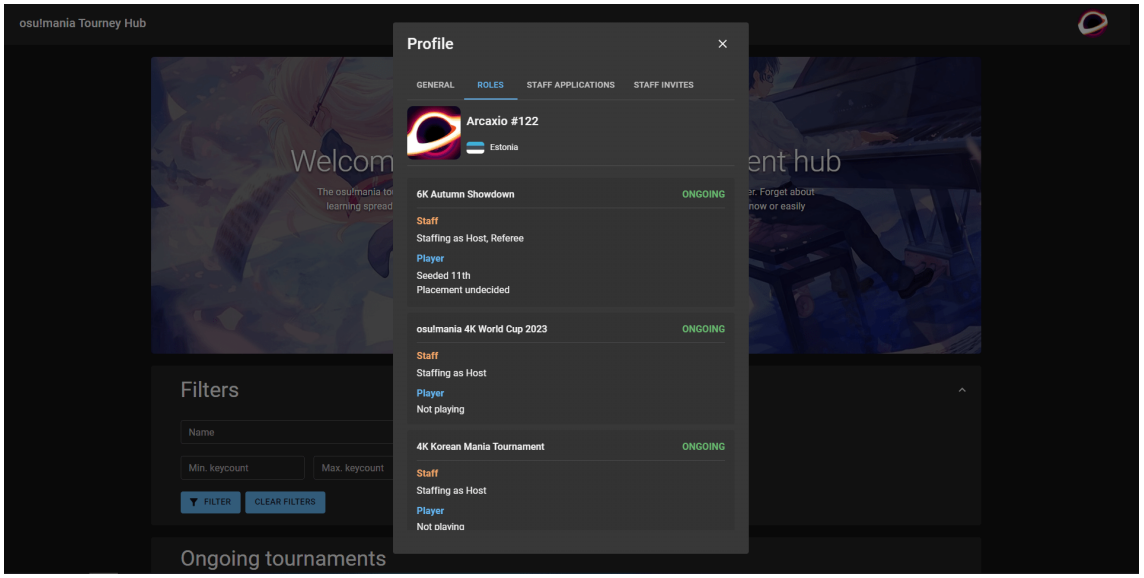
1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Veebirakendus rütmimänguturniiride organiseerimiseks“, mille juhendaja on Meelis Antoi
   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

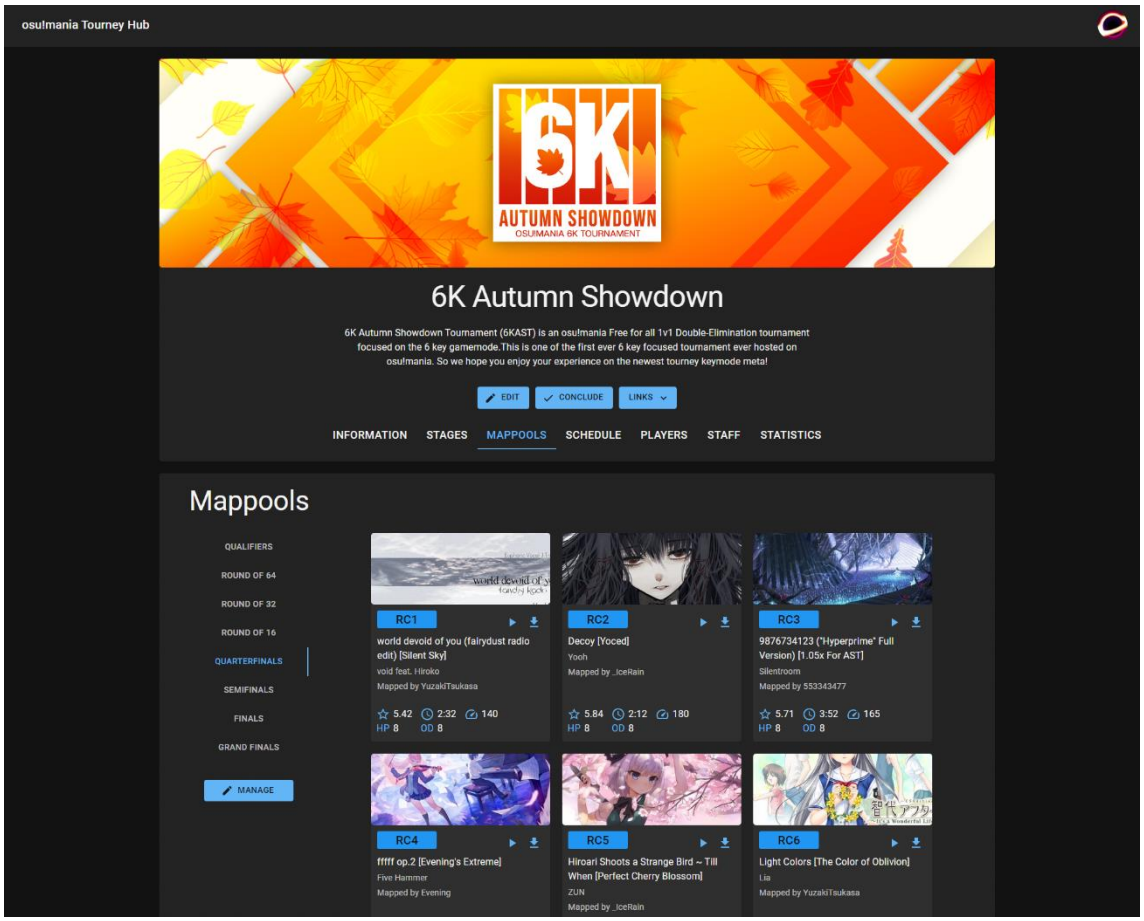
02.01.2025

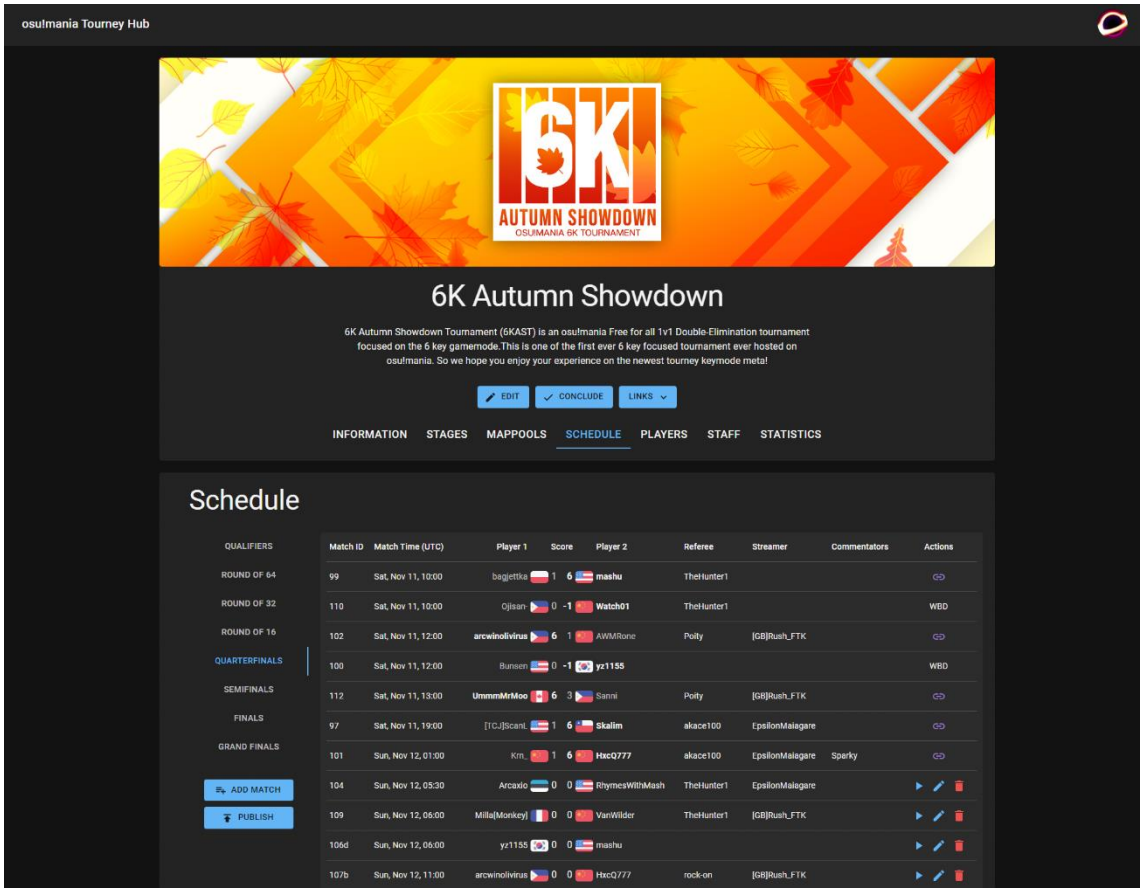
1 Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või enam isikut oma ühise loomingulise tegevusega ning lõputöö kaas- või ühisautor(id) ei ole andnud lõputööd kaitsvale üliõpilasele kindlaksmääratud tähtajaks nõusolekut lõputöö reprodutseerimiseks ja avalikustamiseks vastavalt lihtlitsentsi punktidele 1.1. ja 1.2, siis lihtlitsents nimetatud tähtaja jooksul ei kehti.

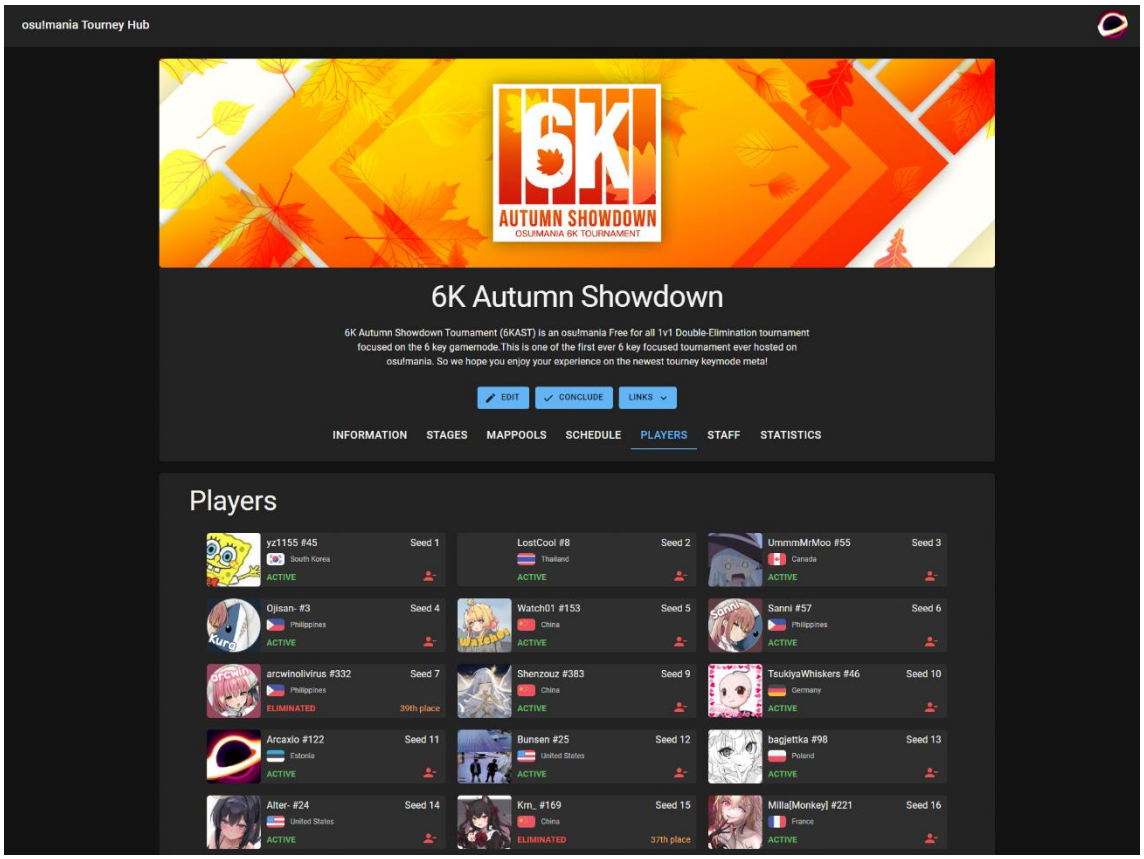
Lisa 2 – Esirakenduse vaated

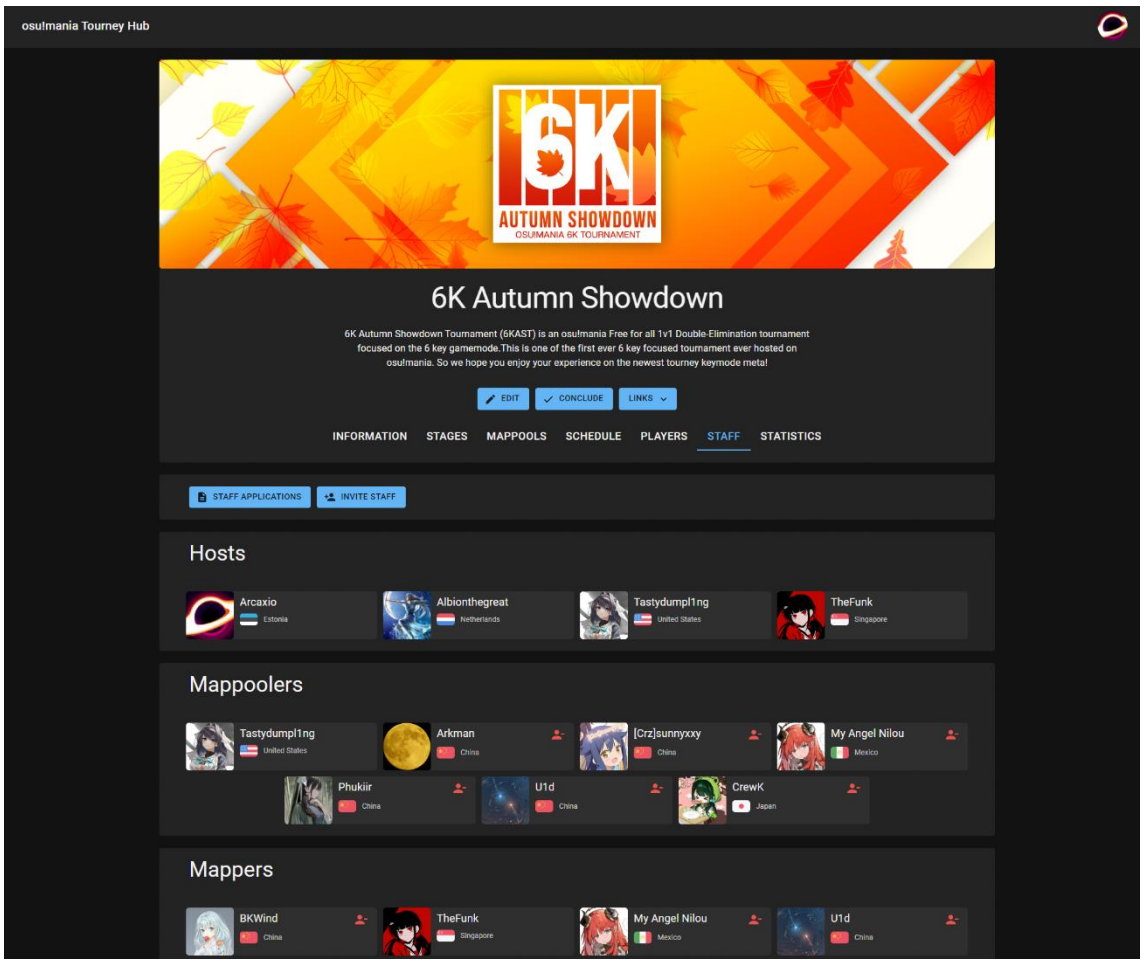


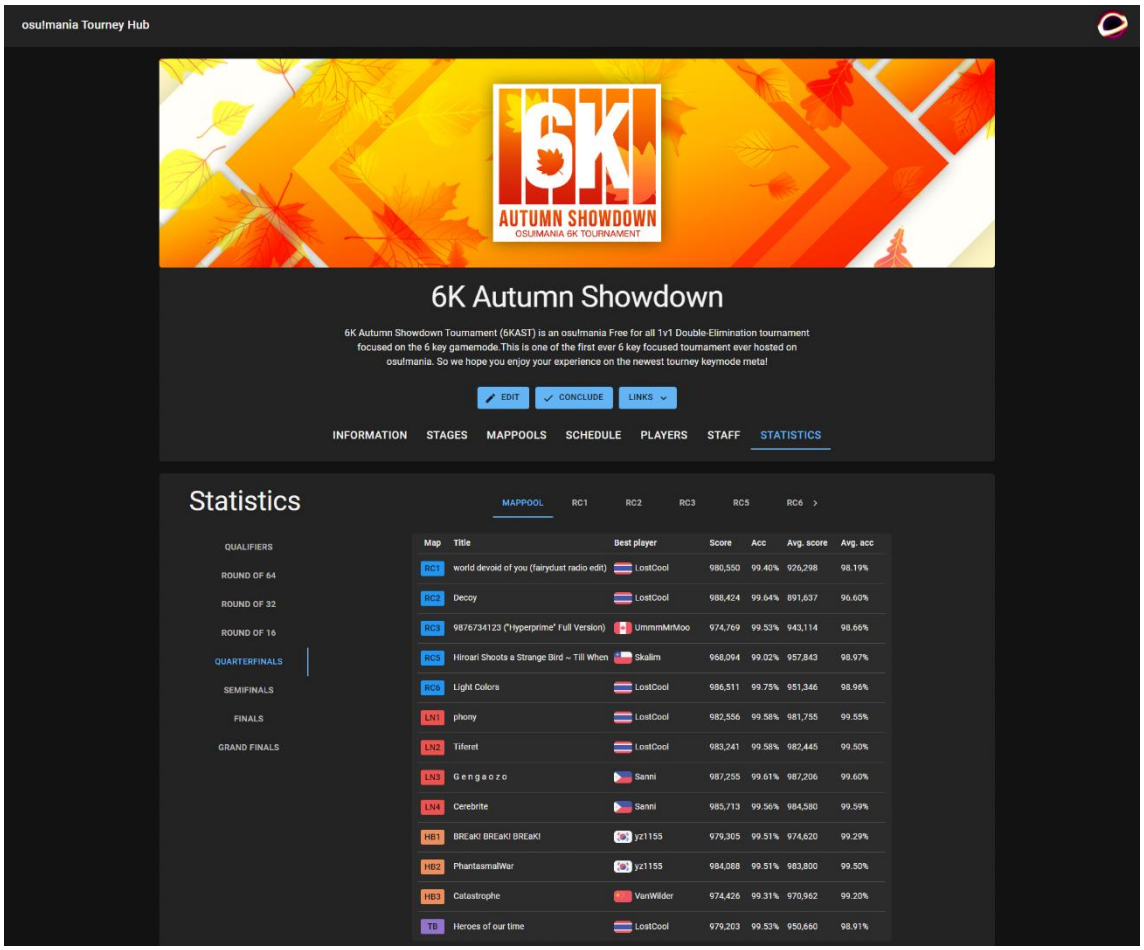


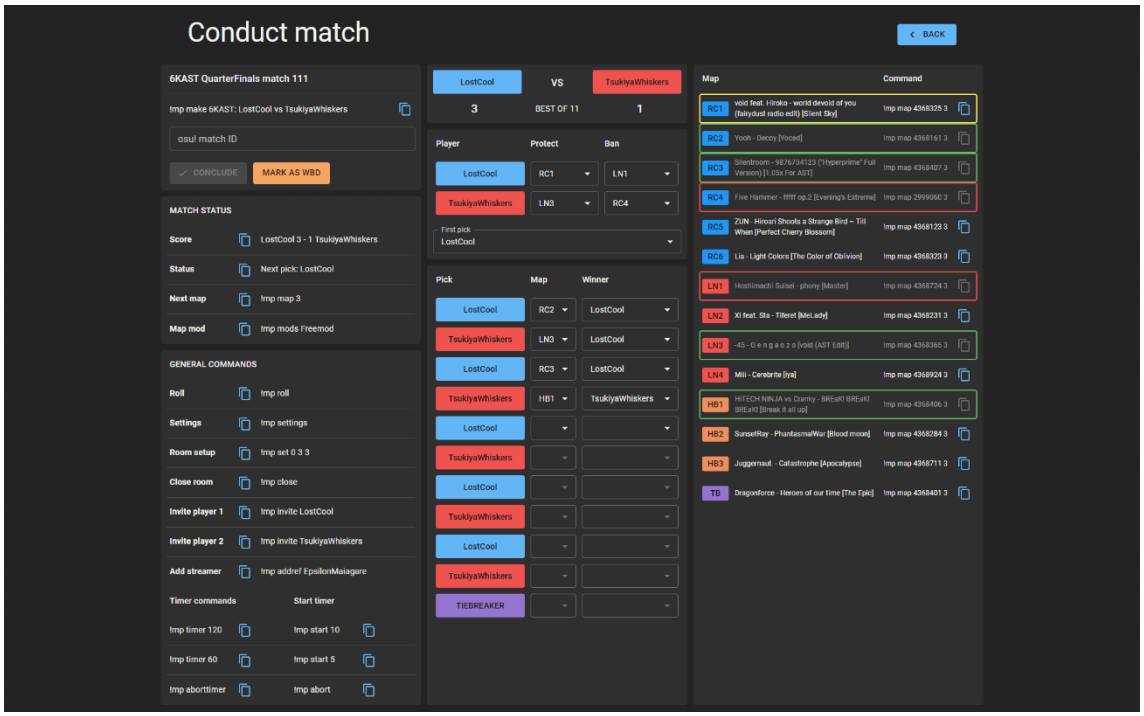

osu!mania Tourney Hub
6K
AUTUMN SHOWDOWN
OSU!MANIA 6K TOURNAMENT
6K Autumn Showdown
6K Autumn Showdown Tournament (6KAST) is an osu!mania Free for all 1v1 Double-Elimination tournament focused on the 6 key gamemode.This is one of the first ever 6 key focused tournament ever hosted on osu!mania. So we hope you enjoy your experience on the newest tourney keymode meta!
EDIT
CONCLUDE
LINKS
INFORMATION
STAGES
MAPPOOLS
SCHEDULE
PLAYERS
STAFF
STATISTICS
Mappools
QUALIFIERS
ROUND OF 64
ROUND OF 32
ROUND OF 16
QUARTERFINALS
SEMIFINALS
FINALS
GRAND FINALS
MANAGE
RC1
world devoid of you (fairydust radio edit) [Silent Sky]
void feat. Hiroko
Mapped by YuzakiTsukasa
5.42 2:32 140
HP 8 OD 8
RC2
Decoy [Yoced]
Yooh
Mapped by _IceRain
5.84 2:12 180
HP 8 OD 8
RC3
9876734123 ("Hyperprime" Full Version) [1.05x For AST]
Silentroom
Mapped by 553343477
5.71 3:52 165
HP 8 OD 8
RC4
fffff op.2 [Evening's Extreme]
Five Hammer
Mapped by Evening
RC5
Hiroari Shoots a Strange Bird ~ Till When [Perfect Cherry Blossom]
ZUN
Mapped by _IceRain
RC6
Light Colors [The Color of Oblivion]
Lia
Mapped by YuzakiTsukasa


osu!mania Tourney Hub
6K
AUTUMN SHOWDOWN
OSU!MANIA 6K TOURNAMENT
6K Autumn Showdown
6K Autumn Showdown Tournament (6KAST) is an osu!mania Free for all 1v1 Double-Elimination tournament focused on the 6 key gamemode.This is one of the first ever 6 key focused tournament ever hosted on osu!mania. So we hope you enjoy your experience on the newest tourney keymode meta!
EDIT
CONCLUDE
LINKS
INFORMATION
STAGES
MAPPOOLS
SCHEDULE
PLAYERS
STAFF
STATISTICS
Schedule
QUALIFIERS
ROUND OF 64
ROUND OF 32
ROUND OF 16
QUARTERFINALS
SEMIFINALS
FINALS
GRAND FINALS
ADD MATCH
PUBLISH
Match ID Match Time (UTC) Player 1 Score Player 2 Referee Streamer Commentators Actions
99 Sat, Nov 11, 10:00 bagjettka 1 6 mashu TheHunter1
110 Sat, Nov 11, 10:00 Ojisan 0 -1 Watch01 TheHunter1 WBD
102 Sat, Nov 11, 12:00 arcwinolivirus 6 1 AWMRone Poity [GB]Rush_FTK
100 Sat, Nov 11, 12:00 Bunsen 0 -1 yz1155 WBD
112 Sat, Nov 11, 13:00 UmmmMrMoo 6 3 Sanni Poity [GB]Rush_FTK
97 Sat, Nov 11, 19:00 [TC.]Scani. 1 6 Skalim akace100 EpsilonMalagare
101 Sun, Nov 12, 01:00 Km. 1 6 HxcQ777 akace100 EpsilonMalagare Sperky
104 Sun, Nov 12, 05:30 Arcaxle 0 0 RhymesWithMash TheHunter1 EpsilonMalagare
109 Sun, Nov 12, 06:00 Milla[Monkey] 0 0 VanWilder TheHunter1 [GB]Rush_FTK
106d Sun, Nov 12, 06:00 yz1155 0 0 mashu
107b Sun, Nov 12, 11:00 arcwinolivirus 0 0 HxcQ777 rock-on [GB]Rush_FTK


osu!mania Tourney Hub
6K
AUTUMN SHOWDOWN
OSUMANIA 6K TOURNAMENT
6K Autumn Showdown
6K Autumn Showdown Tournament (6KAST) is an osu!mania Free for all 1v1 Double-Elimination tournament focused on the 6 key gamemode.This is one of the first ever 6 key focused tournament ever hosted on osu!mania. So we hope you enjoy your experience on the newest tourney keymode meta!
EDIT
CONCLUDE
LINKS
INFORMATION
STAGES
MAPPOOLS
SCHEDULE
PLAYERS
STAFF
STATISTICS
Players
yz1155 #45 Seed 1 South Korea ACTIVE
LostCool #8 Seed 2 Thailand ACTIVE
UmmmMrMoo #55 Seed 3 Canada ACTIVE
Ojisan- #3 Seed 4 Philippines ACTIVE
Watch01 #153 Seed 5 China ACTIVE
Sanni #57 Seed 6 Philippines ACTIVE
arcwinolivirus #332 Seed 7 Philippines ELIMINATED 39th place
Shenzouz #383 Seed 9 China ACTIVE
TsukiyaWhiskers #46 Seed 10 Germany ACTIVE
Arcaxio #122 Seed 11 Estonia ACTIVE
Bunsen #25 Seed 12 United States ACTIVE
bag|ettka #98 Seed 13 Poland ACTIVE
Alter- #24 Seed 14 United States ACTIVE
Km_ #169 Seed 15 China ELIMINATED 37th place
Milla[Monkey] #221 Seed 16 France ACTIVE


osu!mania Tourney Hub
6K
AUTUMN SHOWDOWN
OSUMANIA 6K TOURNAMENT
6K Autumn Showdown
6K Autumn Showdown Tournament (6KAST) is an osu!mania Free for all 1v1 Double-Elimination tournament focused on the 6 key gamemode.This is one of the first ever 6 key focused tournament ever hosted on osu!mania. So we hope you enjoy your experience on the newest tourney keymode meta!
EDIT
CONCLUDE
LINKS
INFORMATION
STAGES
MAPPOOLS
SCHEDULE
PLAYERS
STAFF
STATISTICS
STAFF APPLICATIONS
INVITE STAFF
Hosts
Arcaxio Estonia
Albionthegreat Netherlands
Tastydumpl1ng United States
TheFunk Singapore
Mappoolers
Tastydumpl1ng United States
Arkman China
[Crz]sunnyxxy China
My Angel Nilou Mexico
Phukiir China
U1d China
CrewK Japan
Mappers
BKWind China
TheFunk Singapore
My Angel Nilou Mexico
U1d China


osu!mania Tourney Hub
6K
AUTUMN SHOWDOWN
OSU!MANIA 6K TOURNAMENT
6K Autumn Showdown
6K Autumn Showdown Tournament (6KAST) is an osu!mania Free for all 1v1 Double-Elimination tournament focused on the 6 key gamemode.This is one of the first ever 6 key focused tournament ever hosted on osu!mania. So we hope you enjoy your experience on the newest tourney keymode meta!
EDIT
CONCLUDE
LINKS
INFORMATION
STAGES
MAPPOOLS
SCHEDULE
PLAYERS
STAFF
STATISTICS
Statistics
MAPPOOL
RC1
RC2
RC3
RC5
RC6
QUALIFIERS
ROUND OF 64
ROUND OF 32
ROUND OF 16
QUARTERFINALS
SEMIFINALS
FINALS
GRAND FINALS
Map Title Best player Score Acc Avg. score Avg. acc
RC1 world devoid of you (fairydust radio edit) LostCool 980,550 99.40% 926,298 98.19%
RC2 Decoy LostCool 988,424 99.64% 891,637 96.60%
RC3 9876734123 ("Hyperprime" Full Version) UmmmMrMoo 974,769 99.53% 943,114 98.66%
RC5 Hiroari Shoots a Strange Bird ~ Till When Skalim 968,094 99.02% 957,843 98.97%
RC6 Light Colors LostCool 986,511 99.75% 951,346 98.96%
LN1 phony LostCool 982,556 99.58% 981,755 99.55%
LN2 Tiferet LostCool 983,241 99.58% 982,445 99.50%
LN3 G e n g a o z o Sanni 987,255 99.61% 987,206 99.60%
LN4 Cerebrite Sanni 985,713 99.56% 984,580 99.59%
HB1 BREaK! BREaK! BREaK! yz1155 979,305 99.51% 974,620 99.29%
HB2 PhantasmalWar yz1155 984,088 99.51% 983,800 99.50%
HB3 Catastrophe VanWilder 974,426 99.31% 970,962 99.20%
TB Heroes of our time LostCool 979,203 99.53% 950,660 98.91%


Conduct match
BACK
6KAST QuarterFinals match 111
!mp make 6KAST: LostCool vs TsukiyaWhiskers
osu! match ID
CONCLUDE
MARK AS WBD
MATCH STATUS
Score LostCool 3 - 1 TsukiyaWhiskers
Status Next pick: LostCool
Next map !mp map 3
Map mod !mp mods Freemod
GENERAL COMMANDS
Roll !mp roll
Settings !mp settings
Room setup !mp set 0 3 3
Close room !mp close
Invite player 1 !mp invite LostCool
Invite player 2 !mp invite TsukiyaWhiskers
Add streamer !mp addref EpsilonMaiagare
Timer commands Start timer
!mp timer 120 !mp start 10
!mp timer 60 !mp start 5
!mp aborttimer !mp abort
LostCool VS TsukiyaWhiskers
3 BEST OF 11 1
Player Protect Ban
LostCool RC1 LN1
TsukiyaWhiskers LN3 RC4
First pick
LostCool
Pick Map Winner
LostCool RC2 LostCool
TsukiyaWhiskers LN3 LostCool
LostCool RC3 LostCool
TsukiyaWhiskers HB1 TsukiyaWhiskers
TIEBREAKER
Map Command
RC1 void feat. Hiroko - world devoid of you (fairydust radio edit) [Silent Sky] !mp map 4368325 3
RC2 Yooh - Decoy [Yoced] !mp map 4368161 3
RC3 Silentroom - 9876734123 ("Hyperprime" Full Version) [1.05x For AST] !mp map 4368407 3
RC4 Five Hammer - fffff op.2 [Evening's Extreme] !mp map 2999060 3
RC5 ZUN - Hiroari Shoots a Strange Bird ~ Till When [Perfect Cherry Blossom] !mp map 4368123 3
RC6 Lia - Light Colors [The Color of Oblivion] !mp map 4368323 3
LN1 Hoshimachi Suisei - phony [Master] !mp map 4368724 3
LN2 Xi feat. Sta - Tiferet [MeLady] !mp map 4368231 3
LN3 -45 - G e n g a o z o [void (AST Edit)] !mp map 4368365 3
LN4 Mili - Cerebrite [lya] !mp map 4368924 3
HB1 HiTECH NINJA vs Cranky - BREaK! BREaK! BREaK! [Break it all up] !mp map 4368406 3
HB2 SunsetRay - PhantasmalWar [Blood moon] !mp map 4368284 3
HB3 Juggernaut. - Catastrophe [Apocalypse] !mp map 4368711 3
TB Dragonforce - Heroes of our time [The Epic] !mp map 4368401 3