TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Karl Mihkel Pohga 193464IADB

Veebirakendus jalgpalliväljakute broneeringute haldamiseks

Bakalaureusetöö

Juhendaja: German Mumma
Magistrikraad

Tallinn 2022

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: Karl Mihkel Pohga

16.05.2022

Annotatsioon

Käesoleva bakalaureusetöö eesmärk on luua töötav veebirakendus jalgpalliväljakute broneeringute haldamiseks. Rakendus on mõeldud nii erinevatele väljakute omanikele kui ka harrastajatele, kes soovivad väljakut broneerida. Veebirakenduse eesmärk on teha lihtsamaks praegune väljakute broneerimise süsteem, mis toimub endiselt telefoni või e-mailiga suheldes.

Arendusprotsessi käigus luuakse veebirakendus, mis võimaldab klientidel registreeruda platvormile ning teha uusi broneeringuid. Rakendusel on ka teise rolliga kasutajad ehk väljakute omanikud, kes saavad veebirakendusele lisada uusi väljakuid ning igale väljakule uusi vabu aegu. Samuti saavad nad hallata ka enda väljakute broneeringuid.

Töö esimene peatükk on lõputöö sissejuhatus. Töö teises peatükis kirjeldatakse lähemalt probleemi ja eesmärki. Töö kolmandas peatükis viiakse läbi veebirakenduse analüüs, mille käigus pannakse paika rakenduse nõuded ning analüüsitakse erinevaid tehnoloogiaid. Töö neljandas peatükis viiakse läbi veebirakenduse arendus, mille käigus tuuakse välja rakenduse tähtsamad nüansid. Töö viiendas peatükis tehakse veebirakenduse arendamise käigust järeldused. Töö kuues peatükk on kokkuvõte.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 40 leheküljel, kuut peatükki, 24 joonist, kolme tabelit.

Abstract

Web Application for Managing Football Field Bookings

The aim of this bachelor's thesis is to create a working web application for managing football field reservations. The application is intended for both, football field owners and enthusiasts wanting to book fields for their usage. The purpose of the web application is to simplify the current reservation process, which still takes place by telephone or e-mail.

For this purpose, a web application will be created that allows new user to register on the platform and book available fields. The application will also include user role for field owners, who are able to add new fields to the application, enter available times for bookings for each field and manage current bookings regarding their fields.

The first chapter of the thesis is an introduction. The second chapter describes the problem and the objective of the thesis in more detail. In the third chapter an analysis of the web application is performed, during which the requirements of the application are set and different technologies are analyzed. In the fourth chapter, the software development of the web application is described including the results of the development process. In the fifth chapter, conclusions are drawn from the development of the application. The sixth chapter of the thesis is a summary.

The thesis is in Estonian and contains 40 pages of text, six chapters, 24 figures, three tables.

Lühendite ja mõistete sõnastik

DPI	*Dots per inch*, punkti tolli kohta
API	*Application Programming Interface*, rakendusliides
CSS	*Cascading Style Sheets*, kaskaadilaadistik
DBMS	*Database Management System*, andmebaasi haldussüsteem
DDL	*Data Definition Language*, andmete määratlemise keel
DOM	*Document Object Model*, dokumendi objektmudel
ECMAScript	*JavaScript* programmeerimiskeele standard, mis on mõeldud veebilehtede koos töötamise tagamiseks erinevates brauserites
HMAC	*Hash-Based Message Authentication Code*, räsipõhine sõnumi autentimiskood
HTML	*Hyper Text Markup Language*, veebilehe märgendamise keel
HTTP	*HyperText Transfer Protocol*, veebilehe teabe edastamiseks mõeldud protokoll
JPA	*Java Persistence API*, objekt-relatsioonilise andmebaaside lähenemisviis
JSON	*JavaScript Object Notation*, andmevahetusvorming
JSX	*JavaScript XML;*Eelprotsessor, mis tagab XML kasutamise
JWT	*JSON Web Token*, autentimistvorm
MVC	*Model View Controller*, mudel-vaade-kontroller
ORM	*Object-Relational Mapping*, objekt-relatsioonile kaardistamine
PYPL	*Popularity of Programming Language Index*, programmeerimiskeele populaarsuse indeks
REST	*Representational State Transfer*, veebiteenusega suhtlemise liidese arhitektuur
SQL	*Structured Query Language*, struktureeritud andmebaasipäringu keel
SSL	*Secure Sockets Layer*, turvaline internetiühenduse kiht
ZIP	Faili formaat
tagarakendus	serveripoolne loogika, mis juhib erinevaid rakendusi nende taustal. Enamasti hõlmab tagarakendus andmebaasi, serverit ja rakenduse koodi.

TDD	*Test Driven Development*, testi põhine arendus
UEFA	*The Union of European Football Associations*, Euroopa Jalgpalliliitude Liit
UML	*The Unified Modeling Language*, ühtne modelleerimiskeel
URI	*Uniform Resource Identifier*, unikaalne resurssi eristaja
URL	*Uniform Resource Locator*, internetiaadress
XML	*Extensible Markup Language*, dokumentide märgendamiskeel

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Eestis on väga palju jalgpalliklubisid, kellel pole oma koduväljakuid. Iga aasta tuleb juurde ka mitteprofessionaalseid jalgpalliklubisid, kes otsivad jalgpalli mängimiseks väljakuid. Samas pole ühtset süsteemi, kus väljakuid broneerida, vaid üldjuhul käib broneerimine endiselt helistamise või meili teel, mis on paraku väga aeganõudev ning keeruline.

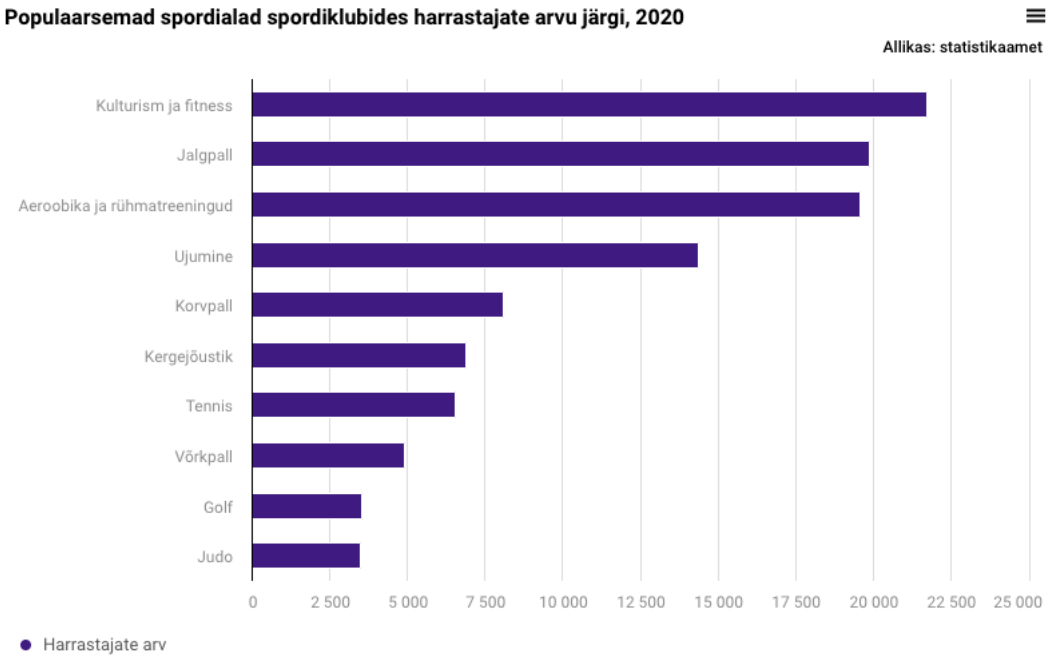
Lõputöö eesmärk on luua veebipõhine jalgpalliväljakute broneerimise haldamiseks mõeldud rakendus. Rakendus peaks olema kasulik nii väljakute omanikele kui ka klientidele. Väljakute omanikud saavad enda vabu aegu märkida ning kliendid neid broneerida. Töö esimeses faasis kirjeldatakse loodava rakenduse tausta, selle aktuaalsust ning miks on loodav rakendus vajalik. Autor kirjeldab ka probleemi, eesmärki ning põgusalt rakenduse ärilisi nõuded.

Töö teises faasis, analüüsi peatükis, kirjeldab autor täpsemalt rakenduse nõudeid, toob välja erinevad loodud kasutajalood ning millised on rakenduse kasutajad. Samuti kirjeldab autor erinevaid tehnoloogiaid, mis on abiks lõputöö rakenduse loomisel. Peatükis keskendutakse ees- ja tagarakenduse arendamiseks valitud tehnoloogiatele, analüüsitakse alternatiive ning põhjendatakse, mille alusel said valikud tehtud. Samuti kirjeldatakse, milline on andmebaasi ülesehitus ning mis tehnoloogia sai valitud.

Töö kolmandas faasis, rakenduse arendamise peatükis, kirjeldab autor erinevaid arendusnüansse. Tuuakse välja tagarakenduse arhitektuur ning kuidas see toimub. Samuti ka eesrakenduse arhitektuur koos kasutajaliidesega. Joonistega tuuakse välja ka erinevaid näiteid rakenduse koodist. Kirjeldatakse ka lõpptulemust, mis toob välja veebirakenduse funktsionaalsuse ja erinevad võimalused.

2 Ülevaade probleemist

2020. aasta seisuga oli jalgpall, spordiklubis harrastajate arvu järgi, populaarsuselt teine spordiala. Jalgpalli harrastab pea 20000 inimest (Joonis 1) ning sinna hulka pole arvestatud need inimesed, kes ei kuulu ühtegi spordiklubisse, vaid mängivad ainult omaenda lõbuks. Samuti pole sinna hulka arvestatud ka professionaalseid jalgpallureid. Jalgpall on populaarseim spordiala 12 erinevas maakonnas ning ka väga populaarne 20-aastaste ja vanemate meeste seas. [1]



Joonis 1. Populaarsemad spordialad spordiklubides harrastajate arvu järgi. [1]

Arvestades, et tegu on harrastajate poolest Eesti kõige populaarsema pallispordialaga, on hetkel infrastruktuuri tase kehv. Vähe väljakuid, kehvad tingimused ning pole e-riigile kohast internetipõhist broneerimissüsteemi.

2.1 Idee

Lõputöö analoogne idee tekkis 2021. aastal läbitud Tarkvaratehnika aine käigus, kui pidi looma enda valitud meeskonnaga veebirakenduse analüüsi projekt. Tolles veebirakenduse analüüsis hoomati kõiki erinevaid spordialasid ning kõiki erinevaid väljakuid. Kuna autor on ise ühe mitteprofessionaalse jalgpalliklubi üks eestvedajaid, siis märkas ta, et selline probleem reaalselt eksisteerib Eesti jalgpallimaastikul. Oma klubile treeninguks väljaku leidmiseks kulub tihti nädalaid ning lugematul hulgal e-maile, mis muudab protsessi ebamugavaks. Autor otsustas lõputöö kontekstis kitsendada veebirakenduse skoopi ainult jalgpalliväljakute broneerimiseks, arvestades enda kogemuse käigus saadud erinevaid teadmisi.

2.2 Probleem

Hetkel puudub Eestis selline keskkond, kus broneerida kõikide jalgpalliväljakute vabu aegu. Samuti ei ole ka *online*-broneerimist mitte ühelgi eraldiseisval jalgpalliväljaku haldaja kodulehel. See teeb keeruliseks harrastajatel jalgpalliväljakute broneerimise, sest vaba aja leidmiseks peab tihtipeale saatma mitmeid meile ja tegema palju telefonikõnesid. Samuti ehitatakse UEFA (*The Union of European Football Associations*) organisatsiooni toetusega uusi jalgpallihalle ja väljakuid Eestisse, kus soovivad ka harrastajad vabu aegu broneerida.

2.3 Eesmärk

Lõputöö eesmärk on luua vastavalt seatud nõuetele sobiv veebirakendus. Loodaval veebirakendusel saavad kliendid mugavalt väljakuid ja nende vabu aegu broneerida ning väljakute omanikud saavad enda väljakute vabu aegu üles märkida. Rakenduse loomisel on prioriteet rakendusele seatud kliendipoolsed ja väljakute omanike poolsed nõuded. Loodav veebirakendus peaks aitama tõsta Eesti jalgpalli harrastajate taset, sest lihtsam on teha ka ühekordseid broneeringuid. Samuti aitab väljaku omanikel teenida tulu aegade pealt, mis muidu ei ole suudetud maha müüa ning mille ajal väljak seisab tühjalt.

Veebirakendusel on kaks peamist kasutajat: klient ja väljaku omanik. Kliendid peaksid saama broneerida ühekordseid väljaku aegu kui ka mitmeid aegu korraga. Kliendid saavad näha ka oma tehtud broneeringuid ning kas nende broneeringud on väljakute poolt

aktsepteeritud või mitte. Samas saavad väljakute omanikud hallata oma väljakuid, lisada väljakuid juurde kui ka hallata väljakule tehtud broneeringuid. Selle jaoks, et kogu rakendust hallata on ka administraator, kes saab määrata kasutajate rolle. Registreerida saavad vaid kliendid, väljakute omanikud registreeritakse administraatori poolt.

3 Veebirakenduse analüüs

Lõputöö praktiliseks väljundiks on luua hajus veebirakendus, mis on ülesehitatud kasutades REST (*Representational State Transfer*) API (*Application Programming Interface*) arhitektuuri. Rakenduse analüüsi peatükis kirjeldatakse lähemalt rakenduse nõudeid, projektianalüüsimise tarkvara valikut, ees- ja tagarakenduse tehnoloogiate valikuid kui ka andmebaasi tehnoloogiate valikut. Samuti analüüsitakse erinevate tarkvarade ja tehnoloogiate valikut ning tuuakse välja miks sai mõni valitud.

3.1 Veebirakenduse nõuded

Veebirakenduse nõuete koostamisel on võetud arvesse asjaolu, et probleemi lahendatakse esialgu ainult Eesti kontekstis. Nõuded on jaotatud kasutajalugudeks, mis on omakorda jaotatud kolme rakenduses oleva rolli järgi: administraator, klient ja väljaku omanik. Need on valminud koostöös erinevate väljakute omanikega ning potentsiaalsete klientidega suheldes. Erinevate funktsionaalsete nõuete kohta loodi ka UML (*The Unified Modeling Language*) *Use Case* diagramm (Joonis 2).

3.1.1 Funktsionaalsed nõuded

Kliendi-põhised funktsionaalsed nõuded:

- Kliendina soovin ma registreerida veebirakenduses, et broneerida vabu aegu.
- Kliendina soovin ma sisse logida veebirakendusse, et broneerida vabu aegu.
- Kliendina soovin ma näha erinevaid jalgpalliväljakuid ning uurida nende detailsemat infot.
- Kliendina soovin ma näha jalgpalliväljakul olevaid vabu aegu ning teha ühekordseid broneeringuid.
- Kliendina soovin ma näha jalgpalliväljakul olevaid vabu aegu ning teha mitmekordseid broneeringuid.

- Kliendina soovin ma näha oma tulevasi broneeringuid ning näha, kas need on kinnitatud.
- Kliendina soovin ma näha oma toimunud broneeringuid ning näha, kas nende eest on makstud.

Väljakute omanike põhised funktsionaalsed nõuded:

- Väljakute omanikuna soovin ma sisse logida veebirakendusse, et hallata oma väljakuid.
- Väljakute omanikuna soovin ma lisada jalgpalliväljakuid ning nende kohta käivat detailset infot.
- Väljakute omanikuna soovin lisada jalgpalliväljakule ühte vaba aega.
- Väljakute omanikuna soovin lisada jalgpalliväljakule korduvat vaba aega.
- Väljakute omanikuna soovin ma näha enda väljakul tehtud broneeringuid, neid kinnitada või tühistada ja märkida, kas nende eest on tasutud.

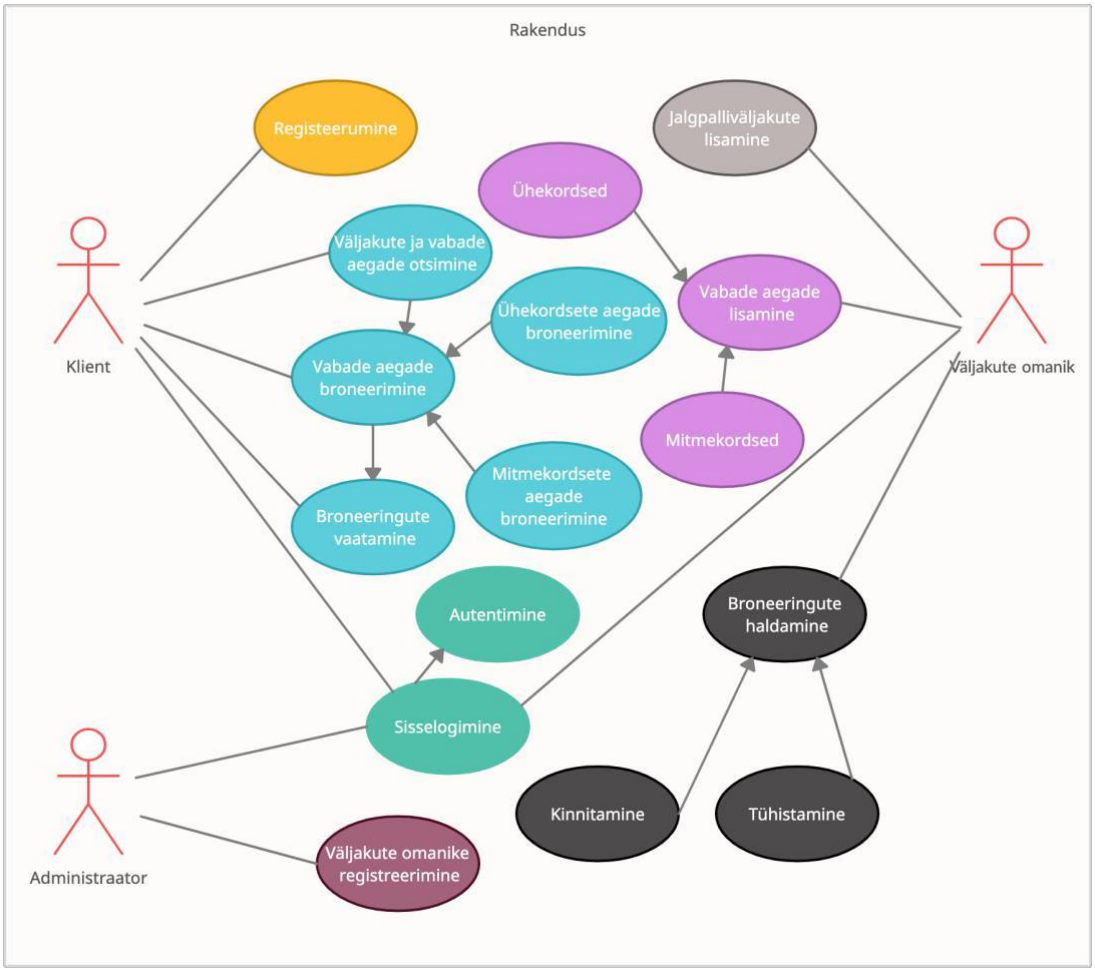
Administraatori-põhised funktsionaalsed nõuded:

- Administraatorina soovin ma teha kõiki eelnevaid mainitud funktsionaalseid protsesse.
- Administraatorina soovin ma registreerida veebirakendusele väljakute omanike kasutajaid.

3.1.2 Mittefunktsionaalsed nõuded

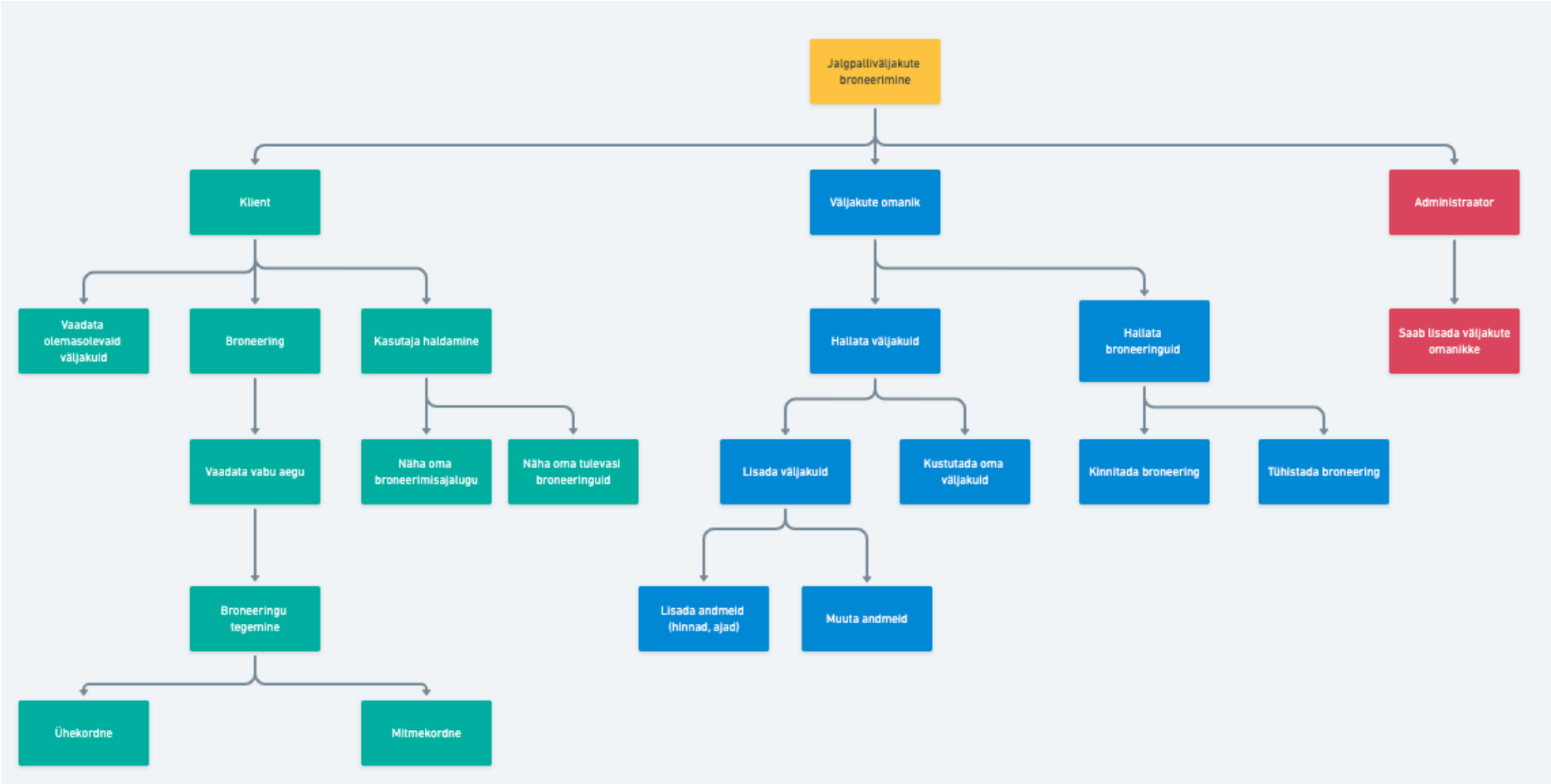
Kõikide kasutajate põhised mittefunktsionaalsed nõuded:

- Veebirakendus peab olema kasutatav ka mobiilis.
- Veebirakendus peab olema kasutatav ükskõik missugusel veebibrauseril.
- Veebirakenduse kasutajate andmed peavad olema turvalised.



Joonis 2. UML Use Case diagramm veebirakenduse funktsionaalsete nõuete kohta

Samuti luuakse rakenduse lihtsamaks mõistmiseks rakenduse mõttekaart (Joonis 3). Mõttekaart luuakse peamiste funktsionaalsuste kohta rakendusel. Mõttekaart on hea viis, kuidas rakendus väiksemateks tükkideks jaotada. Samuti hoiab see visuaalselt hea pildi, mis on juba arendatud ning mis vajab veel rakenduses arendamist. [2]



Joonis 3. Mõttekaart rakenduse funktsionaalsete nõuete kohta.

3.2 Projekti analüüsimise tarkvara

Tänapäeval algab projekti tegemine analüüsist. Analüüs on väga oluline osa projekti valmimisest, sest täpne analüüs aitab hoida kokku palju aega ning muid ressursse. Õnneks on loodud ka mitmeid erinevaid lahendusi, mis aitavad kiiremini ja tõhusamalt projektianalüüsi läbi viia. [3] Autor on valinud enda projektihaldus tarkvaraks Jira.

Algselt loodi Jira, et tuua esile projekti käigus tekkinud vead, kuid nüüdseks on see arenenud tööhaldustööriistaks, mis on sobilik agiilseks arenduseks. Jira pakub nii Scrum kui ka Kanban metoodika jaoks vajalike aluspõhjasid. Tänu Jira tarkvarale on lihtsam projekti tööde ajaplaneerimine ning saab ka näha projekti tootlikkust. [4]

Jirasse on samuti ka väga lihtne importida kolmanda osapoole tarkvara. Näiteks saab Jira siduda Slackiga, et tiimi suhtlus oleks mugavam. Kokku on erinevaid kolmanda osapoole tarkvarasid, mida Jiraga saab siduda, üle 3000. [5] Samuti saavad Jirat kasutada erinevates rollides olevad isikud. Näiteks saavad seal tööd kaardistada nii projektijuhid, analüütikud kui ka arendajad ja testijad. [4]

Jira kasutamisoskus tuleb kindlasti kasuks ka tööturul, sest seda kasutab üle 65000 erineva ettevõtte üle kogu maailma. Jiral on ka väga mugav vigade analüüsimise funktsionaalsus, mis on eriti populaarne tarkvaraarendajate seas. Lisaks on Jiral tähtajatu tasuta tarkvara kasutamise võimalusi, seniks kuni sinu projektis pole rohkem kui kümme inimest. [5]

3.3 Tagarakendus

Siin peatükis toob autor välja mõned populaarsemad programmerimiskeeled, analüüsib neid ning põhjendab oma lõplikku valikut. Samuti lähtuvalt programmeerimiskeelest, teeb autor valiku raamistiku osas ning selgitab uurimistöö tagarakenduseks valitud metoodikat.

3.3.1 Tagarakenduse programmeerimiskeel

Maailmas on väga mitmeid erinevaid programmeerimiskeeli, mille vahel saab tarkvaraarendaja valida oma projekti kirjutamisel. Siiski enne keele valimist tuleb analüüsida erinevaid valikuid ning teha nende põhjal valik just selle järgi, mis on rakenduse jaoks oluline. [6]

PHP

PHP on väga populaarne programmeerimiskeel just seetõttu, et seda on algajal arendajal väga lihtne õppida ning see on väga stabiilne. Samas tähendab ka see seda, et PHP kohta võib internetis leiduda väga palju kehvasid õpetusi. Üks populaarsemaid veebirakendusi, mis on ehitatud PHP programmeerimiskeelele, on Facebook. [7]

PHP on väga stabiilne ja kasutajasõbralik kood. Samuti on PHP väga paindlik keel, mis võimaldab erinevates keeltes rakendusi siduda. Samas ei ole see väga turvaline keel ning vajab väga palju õppimist, et seda korrektselt kirjutada. Samuti on PHP võrdlemisi kehva jõudlusega keel, mis tähendab seda, et seal on keeruline kasutada erinevaid funktsionaalsusi korraga. [8]

Python

Veebirakenduste arendamisel peetakse *Python* programmeerimiskeelt tihti üheks parimaks keeleks. Seda on lihtne õppida, sellel on väga lihtne süntaks ning samuti on kerge integreerida teiste programmeerimiskeeltega. *Python* keele kohta on ka palju abistavaid õppematerjale. [7] *Python* keelel on ka väga palju erinevaid teeke, mis aitavad arendajal koodi kirjutada. Samas on *Python* peamiselt suunatud masinõppe ja andmeteaduse rakendustele. [9]

Java

Java on arendajate seas populaarne programeerimiskeel olnud juba üle 20 aasta. *Java* on väga stabiilne objektorienteeritud keel, mis annab arendajatele väga hea kontrolli keele üle. Samuti on *Java* kohta internetis väga palju abimaterjale. [10] *Java* on ka väga turvaline keel. *Spring* raamistikuga saab tagada väga hea turvalisuse oma programmile. Üleüldse on *Java* keelel mitmeid raamistikke ning teeke, mis teevad arenduse oluliselt mugavamaks. Samas ei ole jõudlus nii hea, kui on näiteks *C* programmeerimiskeelel. [7]

C#

C# on Microsofti poolt loodud objektorienteeritud keel, mille süntaks on *Java* omaga võrdlemisi sarnane. *C#* on osa *.NET Core* raamistikuks ning programmeerimiskeele jõudlus on väga hea. Samuti on keelel väga suur kogukond nii, et internetis on leida palju

abistavat materjali. *C#* keelega saab luua väga mitmekülgseid rakendusi, nii veebirakendusi, mobiilirakendusi kui ka erinevaid mänge. [11]

Tagarakenduse programmeerimiskeele valimisel on autor kitsendanud oma valiku nelja ülaltoodud keele vahele. Üleval toodud analüüsi arvestades, koostas autor tabeli, kus arvestas kolme kriteeriumi:

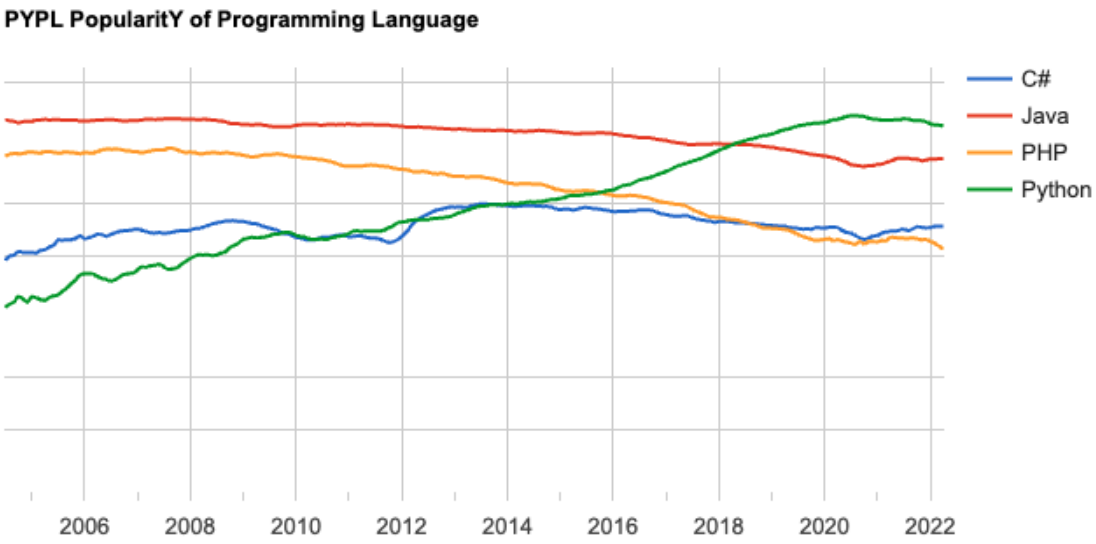
- Populaarsus – kui populaarne on antud keel hetkel maailmas?
- Kogemus – kui suur kogemus on autoril keele kirjutamisel?
- Õppimise keerukus – kui lihtne on leida programmeerimiskeele kohta materjali ning seda talletada?

Igale kriteeriumile antakse hindeid 4 punkti skaalas, mida parem, seda väiksem punktide arv. Lõpuks teeb autor valiku selle järgi, milline programmeerimiskeel kogus kokku kõige vähem punkte.

Tabel 1. Tagarakenduse programmeerimiskeele kriteeriumite võrdlus.

	PHP	Python	Java	C#
Populaarsus	4	1	2	3
Kogemus	4	3	1	2
Õppimise keerukus	4	3	1	2
Kokku	**12**	**7**	**4**	**7**

Populaarust hinnates kasutati PYPL (*Popularity of Programming Language*) indeksit (Joonis 4). PYPL indeks analüüsib kui tihti mingi keele õpetusi otsitakse Google otsingumootoriga. Selles võrdluses tuli kõige populaarsemaks keeleks *Python*, mida otsitakse 27,95% kordadest ning mille trend on viimase paari aastaga oluliselt kasvanud. Teisena on *Java*, mida otsitakse 18,09% kordadest, seejärel C# 7,39% ning PHP 5,48% kordadest. [12]



Joonis 4. Tagarakenduse programmeerimiskeelte populaarsus PYPL järgi. [12]

Kogemus ja õppimise keerukus on küllaltki individuaalsed kriteeriumit. Kogemuse kriteeriumi arvestuses tuli parimale kohale *Java*, sest seda praktiseerib autor igapäevaselt tööl. Samuti tuli ka õppimise keerukuse arvestuses parimale kohale Java, sest Javal on väga suur kommuun ning palju abistavaid materjale. Kõikidest nendest aspektidest lähtudes otsustas autor valida tagarakenduse programmeerimiskeeleks *Java*.

3.3.2 Tagarakenduse raamistik

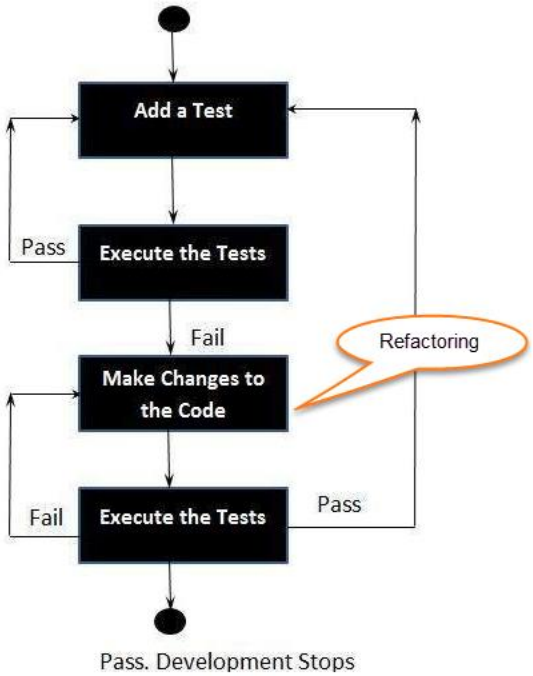
Kuna Java keel on üks maailma populaarseimaid programmeerimiskeeli, siis on talle loodud ka mitmeid erinevaid raamistike. Raamistiku kasutamine arendamise käigus ei ole ilmtingimata kohustuslik, aga seda saab kasutada mitmel erineval põhjusel. Raamistik aitab arendajal jälgida koodikirjutamisel ühtset struktuuri. Samuti toetab raamistik rakendust mitmete erinevate teekidega. [13] Arvestades erinevaid aspekte otsustas autor tagarakenduse raamistikuna kasutada Spring raamistikku.

Spring raamistik on *Java* programmeerimiskeelel põhinev raamistik, mis loodi 2002. aastal. Raamistik on jagatud erinevateks mooduliteks, millest rakenduse arendamise käigus kasutatakse *Spring MVC* (*Model View Controller*) moodulit. Springi projekti saab luua kasutades *Spring Initializrit*, mis loob sulle sinu valitud sõltuvustega uue projekti. Samuti on *Spring* raamistikul väga mugav turvalisuse moodul, mis aitab tagada turvalise rakenduse. [13]

3.3.3 Tagarakenduse arenduse metoodika

Tagarakenduse arenduse metoodikas valis autor testipõhise arenduse ehk TDD (*Test Driven Development*). TDD on testidel põhinev metoodika, mille puhul kirjutatakse enne loogika valmimist testid. Selle tehnika peamine eesmärk muuta koodi ainult siis, kui test ebaõnnestub. Seega tekib vähem üksteist kordavaid testmeetodeid. Testipõhist arendust kasutatakse tihti agiilse arenduse puhul. [14] Testipõhise arenduse põhimõtted on järgmised (Joonis 5):

1. Enne loogika valmimist kirjutatakse testid.
2. Kirjutatakse lihtne loogika, mis läbib kõik testid.
3. Kui test ebaõnnestub, parandatakse ja täiustatakse koodi, kuni test on jälle töökorras.



Joonis 5. Testipõhise arenduse metoodika. [14]

Testipõhisel arenduse metoodikal on väga palju eeliseid. See metoodika aitab kiiresti jõuda rakenduse vigadele jälile. Samuti vähendab see oluliselt koodi ümbertöötlemiseks kuluvat aega. Metoodika tagab ka arendajale kindlustunde, et tema kood on alati testitud ning korrektne. Samuti aitab see luua loetava ning eeskujuliku struktuuriga testmeetodid. [14]

3.4 Eesrakendus

Eesrakendus ehk kliendirakendus on kõik see, mida on visuaalselt näha, nagu näiteks nupud, lingid, animatsioonid jne. REST API arhitektuuri puhul on eesrakenduse ülesanne koguda informatsiooni ning saata see tagarakendusele. Samuti on ka ülesandeks pärida informatsiooni tagarakenduselt. [15] Siin alampeatükis toob autor välja mõned populaarsemad eesrakenduse raamistikud, analüüsib neid ning põhjendab oma lõplikku valikut. Lisaks kirjeldab autor valitud programmeerimiskeelt ning kuidas luuakse eesrakendusele disain.

3.4.1 Eesrakenduse programmeerimiskeel

Eesrakenduse programmeerimiskeele valik piirdub rakenduse arendamisel kahe keelega: *JavaScript* ja *TypeScript*. Need kaks keelt said valitud, kuna nende näol on tegu arendajate seas kõige populaarsemate eesrakenduse keeltega. [2] HTML (Hyper Text Markup Language) ja CSS (*Cascading Style Sheets*) puhul pole tegemist konkreetsete programmeerimiskeeltega. Need on pigem projekti struktuuri abistavad võimalused. *JavaScript* on 1995. aastal loodud programmeerimiskeel, mida kasutab 97% veebilehekülgedest. *JavaScript* on dünaamiline keel, mis vastab *ECMAScript* standardile. [16]

TypeScript on *JavaScript* keele edasiarendus ning sellel on palju eeliseid. Üks peamisi eeliseid on see, et *TypeScript* on range keel. See tähendab, seda et igal muutujal on rangelt määratletud, mida ta võib sisaldada. See muudab koodi kirjutamise oluliselt lihtsamaks ning väldib mitmeid anomaaliaid, mis võivad tekkida *JavaScriptiga*. [17] Just selle suure erinevuse pärast, et *TypeScript* on range keel, valis autor eesrakenduse programmeerimiskeeleks *TypeScript* keele.

3.4.2 Eesrakenduse raamistik

Nii nagu maailmas on väga mitmeid programmeerimiskeeli, on ka väga palju erinevaid eesrakenduse raamistikke. Peamiselt kasutatakse eesrakenduse raamistikke erinevate tööriistade ja komponentide loomiseks ning ka lihtsamaks arendamiseks. Siiski tuleb ka nende vahel teha kaalutletud valik ning arvesse tuleb võtta just arendatava projekti eripärasid. [18]

Angular

Angular on 2016. aastal loodud raamistik, mis põhineb eelkõige *TypeScript* keelel. *Angulariil* on kahesuunaline sidumise süsteem, mis tähendab seda, et toimub sünkroonimine vaate ja mudel vahel reaalajas. *Angular* on suure jõudlusega keel, mis on väga hea raamistik veebirakenduste arendamiseks. Samuti on sellel ka väga suur kogukond, mis toetab arendajaid abimaterjalidega. Samas on *Angulari* küllatki keeruline õppida. [18]

Vue

Vue raamistikku peetakse tänapäeval üheks lihtsaimaks eesrakenduse raamistikuks. Sarnaselt Angularile on sellel kahesuunaline sidumise süsteem. *Vue* on eelkõige suunatud suuremate projektide arendamiseks, sest ta kasutab lehtede loomiseks *MVC* struktuuri. *Vuel* on väga põhjalik dokumentatsioon, mis teeb arendamise üsnagi lihtaks ja selgeks. Samuti on seal väga lihtne koodi korduvkasutada ning ka integreerida. [18]

React

React on avatud lähtekoodiga raamistik, mille on loonud Facebooki arendustiim. Peamiseks põhjuseks, miks *React* raamistik loodi, oli see, et lahendada koodi hallatavusega seotud probleemid. *React* eristub teistest raamistikest oma DOM (*Document Object Model*) mudeli poolest. *React* on ka väga kasutajasõbralik ning sellel on väga põhjalik dokumentatsioon. Samuti on *React* keeles väga lihtne komponente taaskasutada. Samas on JSX (*JavaScript XML*) õppimine algajale arendajale võrdlemisi keeruline. [18]

Eesrakenduse raamistiku valimisel on autor kitsendanud oma valiku kolmele ülal toodud raamistiku vahele. Üleval toodud analüüsi arvestades, koostas autor tabeli, kus arvestas kolme kriteeriumi:

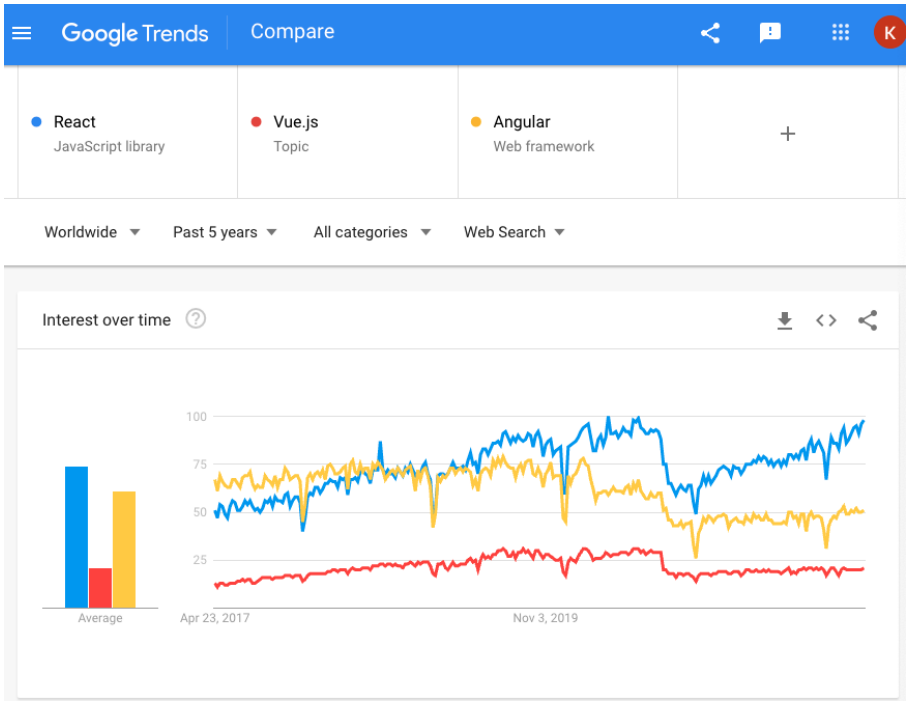
- Populaarsus – kui populaarne on antud raamistik hetkel maailmas?
- Kogemus – kui suur kogemus on autoril raamistiku kirjutamisel?
- Õppimise keerukus – kui lihtne on leida raamistiku kohta materjali ning seda talletada?

Igale kriteeriumile antakse hindeid 3 punkti skaalas, mida parem, seda väiksem punktide arv. Lõpuks teeb autor valiku selle järgi, milline raamistik kogus kokku kõige vähem punkte.

Tabel 2. Eesrakenduse raamistiku kriteeriumite võrdlus.

	Angular	Vue	React
Populaarsus	2	3	1
Kogemus	1	3	2
Õppimise keerukus	2	3	1
Kokku	**5**	**9**	**4**

Populaarsust hinnates kasutati *Google Trendsi* poolt loodud graafikut (Joonis 6). *Google Trendsi* abil saab sisestada kolm *Googles* otsivatavat märksõna ning seejärel leitakse sulle millised neist on kõige populaarsemad märksõnad. Raamistike võrdluses kujunes kõige populaarsemaks *React*, mis on viimased neli aastat olnud antud raamistike võrdluses kõige populaarsem, seejärel *Angular* ning kolmandana *Vue*. [19]



Joonis 6. Eesrakenduse raamistike populaarsus Google Trendsi järgi. [19]

Sarnaselt tagarakenduse programmerimiskeele valikul, on ka eesrakenduse raamistiku valikul kogemus ja õppimise keerukus individuaalsed kriteeriumid. Kogemuse kriteeriumit arvestuses, tuli parimale kohale Angular, sest seda raamistikku praktiseerib autor igapäevaselt tööl. Samas õppimise keerukuse arvestuses tuli parimale kohale *React*, sest selle kohta leidub internetis kõige rohkem abimaterjale. Arvestades kõiki neid aspekte, otsustas autor valida eesrakenduse raamistikuks *Reacti*.

3.4.3 Eesrakenduse disain

Eesrakenduse disaini arendamiseks on autor valinud Bootstrapi raamistiku. Bootstrap on tasuta avatud lähtekoodiga raamistik, mis on loodud veebilehe lihtsamaks arendamiseks. See koosneb HTML-, CSS- ja JavaScript-põhistest skriptidest funktsioonide ja komponentide näol. [20]

Bootstrapi peamine eesmärk on aidata arendajatel luua kohalduva disainiga veebilehti. See tähendab seda, et kõik komponendid töötavad optimaalselt igal erineval ekraanisuurusel. Bootstrap on väga lihtne raamistik, mida õppida. Sellel on väga suur kogukond ning põhjalik dokumentatsioon. Sellel on ka väga lihtne piltide lisamise süsteem, mis automaatsel arvutab välja pildi suuruse veebilehel, arvestades ekraani suurust. [21]

3.5 Andmebaas

Andmebaas on organiseeritud ning struktureeritud teabe või andmete kogum, mis salvestatakse tavaliselt elektrooniliselt arvutisüsteemi. Andmebaasi juhib andmebaasihaldussüsteem ehk DBMS (*Database Management System*). Tänapäeval kasutusel olevate andmebaaside tüüpide andmed modelleeritakse tavaliselt tabeliteks, koos ridade ja veergudega. Nii saab andmebaasis olevatele andmetele lihtsamini juurde pääseda, neid hallata, muuta ja värskendada. Enamik andmebaase kasutab andmete töötlemiseks SQL (*Structured Query Language*) keelt. [22]

Korrektse andmebaasi olemasolu on rakenduse töötamisele väga oluline. Kehvalt struktureeritud andmebaas võib olla aeglane, mis raskendab rakenduse tööd. Samuti võivad sellised andmebaasid tagastada vale informatsiooni ning käivitada mittevajalikke funktsionaalsusi. [23] Seega on õige andmebaasi valik väga oluline ühe veebirakenduse töö tagamisel.

Lõputöö arendamise käigus kasutatavaks andmebaasiks valis autor *PostgreSQL* andmebaasi. *PostgreSQL* on objekt-relatsiooniline andmebaas, mis tähendab seda, et kõik andmebaasi andmed on objekti vormis. Tegu on väga kiire ja hea jõudlusega andmebaasiga, mis on täiesti tasuta. [24] *PostgreSQL* pakub ka väga häid turvalisuse tagamise võimalusi andmebaasis. Sellel on SSL (*Secure Sockets Layer*) -tugi ning ka koos andmebaasiga tuleb *SE-PostgreSQL* täiendus, mis võimaldab täiendavaid juurdepääsukontrolle. [25]

4 Veebirakenduse loomine

Selles peatükis kirjeldatakse veebirakenduse loomise protsessi. Peatükk on jaotatud neljaks alampeatükiks, mille esimeses osas kirjeldatakse projekti ettevalmistust. Luuakse kasutajalood *Jira* keskkonnas, samuti pannakse paika ka andmebaasi skeem koos omavaheliste seostega.

Peatüki teises osas keskendutakse tagarakenduse arendamisele. Kirjeldatakse tagarakenduse arhitektuuri, tuuakse näiteid metoodika kasutamisest ning kirjeldatakse, kuidas tagatakse rakenduse turvalisus.

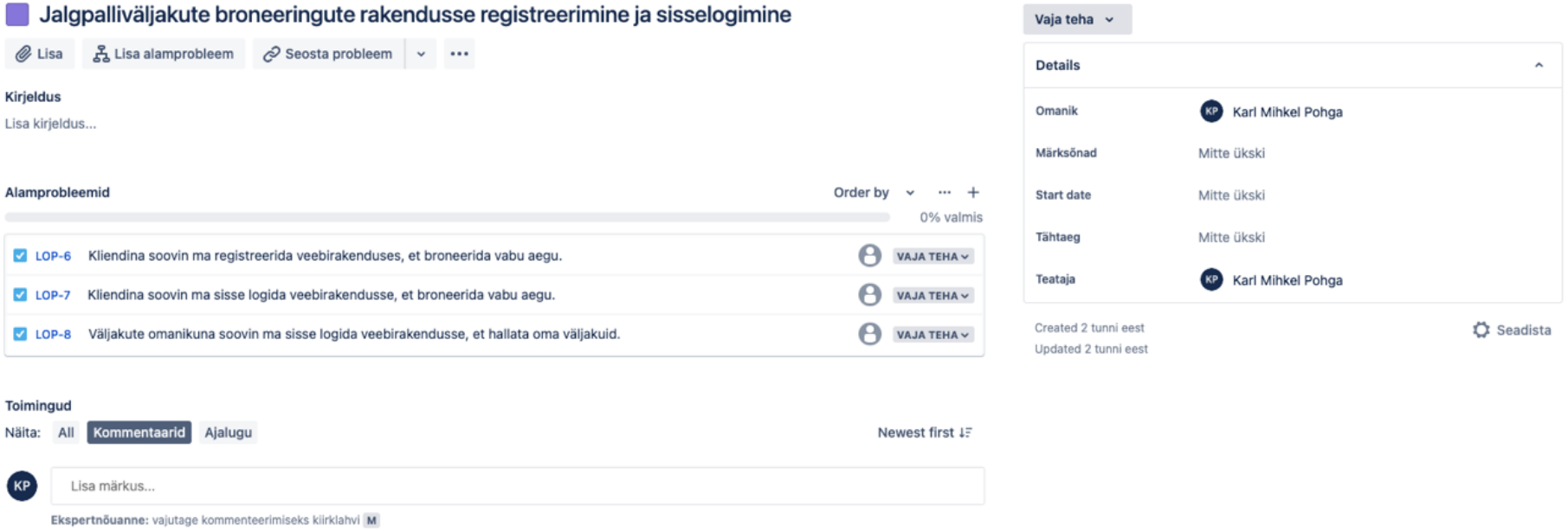
Peatüki kolmandas osas keskendutakse eesrakenduse arendamisele. Kirjeldatakse samuti eesrakenduse arhitektuuri ning tuuakse näiteid erinevate kasutajate võimalustest veebirakendusel.

4.1 Projekti ettevalmistus

Selles peatükis kirjeldatakse projekti ettevalmistust. Kõigepealt tehakse läbi põhjalik analüüs eelmises peatükis valituks osutunud *Jira* platvormil. Seejärel luuakse projekt *Spring Initializri* abil ning lõpuks luuakse andmebaasimudel ning seadistatakse see kohalikuks arendamiseks.

4.1.1 Tööde planeerimine

Selle jaoks, et kaardistada eelseisvad tööd loodi *Jira* keskkonnas uus projekt. Projekti loodi omakorda veebirakenduse arendamise käigus vajaminevad kasutajalood (Joonis 7). Kasutajalood on hea viis oma töö planeerimiseks ning korraliku struktuuri loomiseks. [26] Kasutajalood on agiilses arenduses väga olulised. Need aitavad luua analüüsi käigus arendajale konkreetse arendamise struktuuri. Kasutajalugude eesmärk on sõnastada kindlalt ja üheselt mõistetavalt arenduse töö. Need on üldiselt kuni paarilauselised kirjeldused soovitavast tulemusest. Küll aga ei sisalda need kindlaid arenduse nõudeid. [27] Kõik loodud rakenduse arendamiseks vajaminevad kasutajalood on toodud välja teises peatükis.

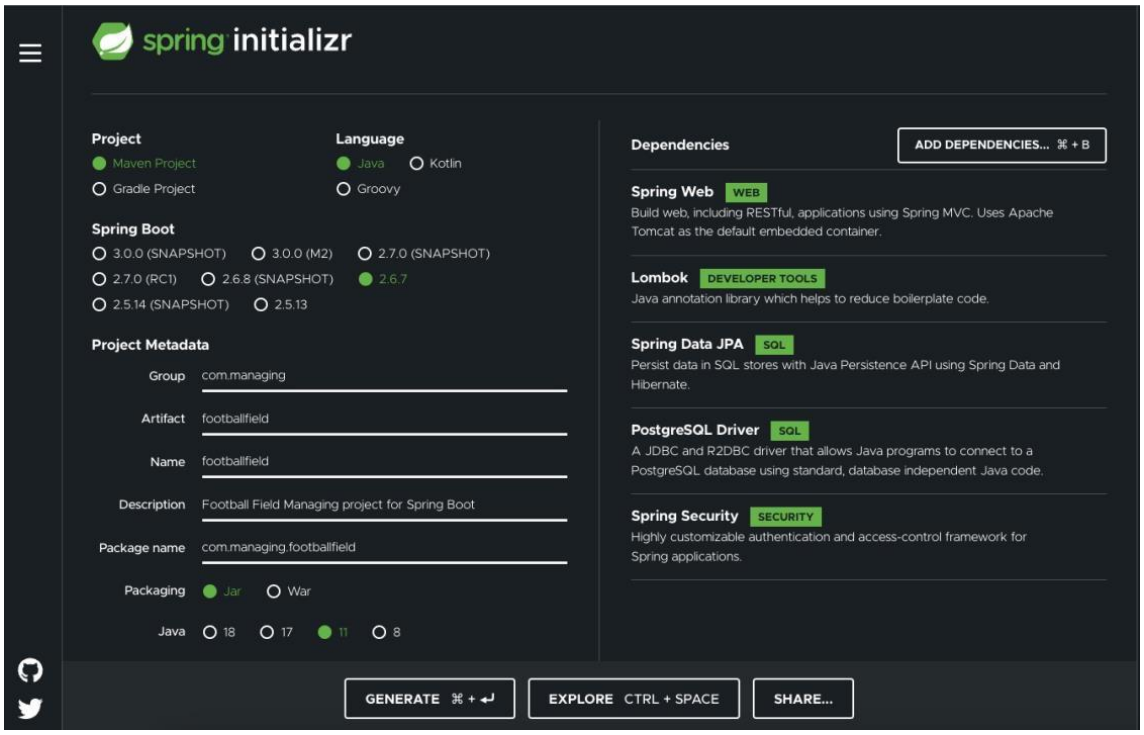


Joonis 7. Jira platvormil loodud kasutajaloo näide.

4.1.2 Spring Boot projekti seadistamine

Spring Boot projekti seadistamine on tänu *Spring Initializrile* tehtud tänapäeval väga lihtsaks. Selleks kulub umbes 15 minutit. *Spring Initialzr* loob eelinstallitud projekti, mille saab *ZIP*-formaadis allalaadida. Sealt saab ka valida kohe rakenduses kasutatavad sõltuvused. [28]

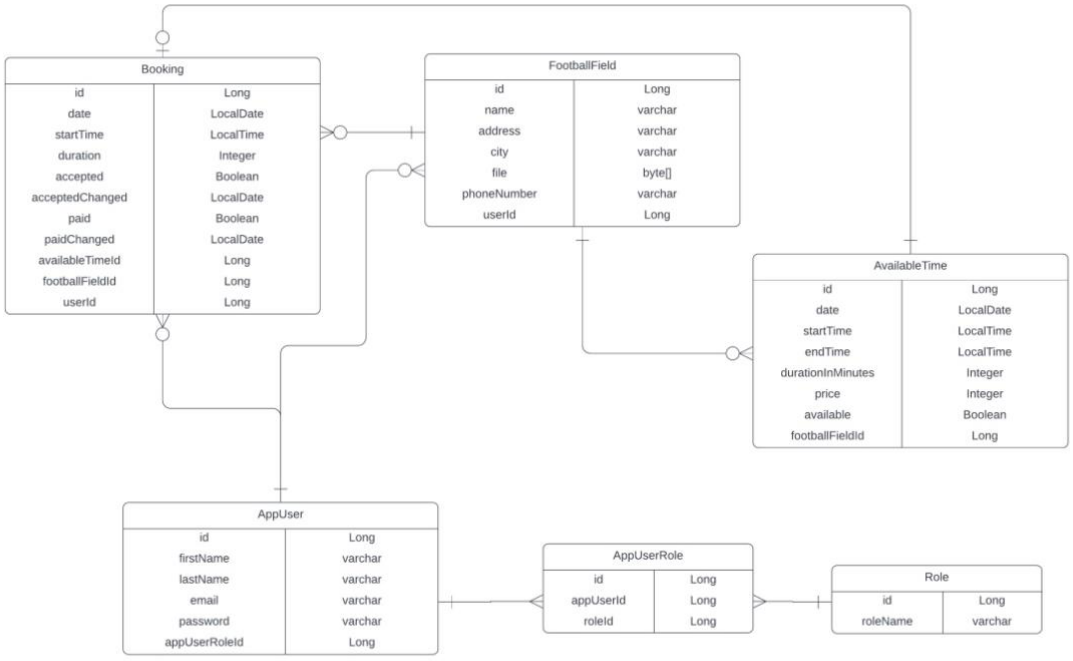
Projekti loomiseks tuleb sisestada metaandmed. Valida endale korrektne keel, projektitüüp ning ka Spring Boot raamistiku versioon (Joonis 8). Projekt ise sisaldab põhiklassi, testklassi ning rakenduse parameetrite kausta.



Joonis 8. Spring Boot projekti loomiseks kasutatud metaandmed

4.1.3 Andmebaasimudel ja selle seadistamine

Teise peatüki analüüsi tulemustest saab autor luua andmebaasi olemi-suhte diagrammi. Diagramm kirjeldab andmebaasis olevaid tabeleid ning nende vahelisi seoseid. Välja on toodud ka atribuudid koos oma tüüpidega. Kokku sai loodud 6 tabelit (Joonis 9), mille kõigi primaarvõti on *Long* tüüpi. Peamisteks tabeliteks on broneeringute tabel, vabade aegade tabel ja jalgpalliväljakute tabel. Lisaks on defineeritud ka kasutajate tabel, rollide tabel ning nende omavahelise seose tabel.



Joonis 9. Andmebaasi olemi-suhte diagramm

Analüüsi peatükis sai andmebaasiks valitud *PostgreSQL*. See on relatsiooniline andmebaas, mis hoiab andmeid JPA (*Java Persistence API*). JPA on ORM (*Object-Relational Mapping*) raamistik, mis pakub erinevaid klasse ja meetodeid, et andmebaasiga suhelda. JPA kasutamiseks sai eelnevalt lisatud ka sellele vastav sõltuvus.

Selle jaoks, et luua ühendus andmebaasiga, tuleb ressursside kataloogis application.properties faili luua andmebaasiühenduse spetsifikatsioon. Spetsifikatsiooni kirjeldamiseks kasutatakse *Hibernate* rakendust, mis aitab rakendusel luua ühendus andmebaasiga. Failis täpsustatakse ära formaat, dialekt kui ka andmeallikas. Täpsustatakse ka DDL (*Data Definition Language*) formaat, mis tagab selle, kuidas andmebaas rakenduse käimisel käitub. Arendamise käigus kasutatakse tavaliselt *create-drop* või *update* väärtust.

```
spring.datasource.url=jdbc:postgresql://localhost:5433/book
ing

spring.jpa.hibernate.ddl-auto=create-drop

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.dialect=org.hibernate.dia
lect.PostgreSQLDialect

spring.jpa.properties.hibernate.format_sql=true
```

Koodinäide 1. Andmebaasi konfiguratsioon *application.properties* failis

4.2 Tagarakenduse arendamine

Tagarakenduse arendamine on jaotatud nelja peatükki: tagarakenduse arhitektuur ja struktuur, tagarakenduse metoodika kasutamine, REST API otsad ning tagarakenduse turvalisus. Järgnevates peatükkides on lähemalt kirjeldatud veebirakenduse tagarakenduse arenduse erinevaid aspekte.

4.2.1 Tagarakenduse arhitektuur ja struktuur

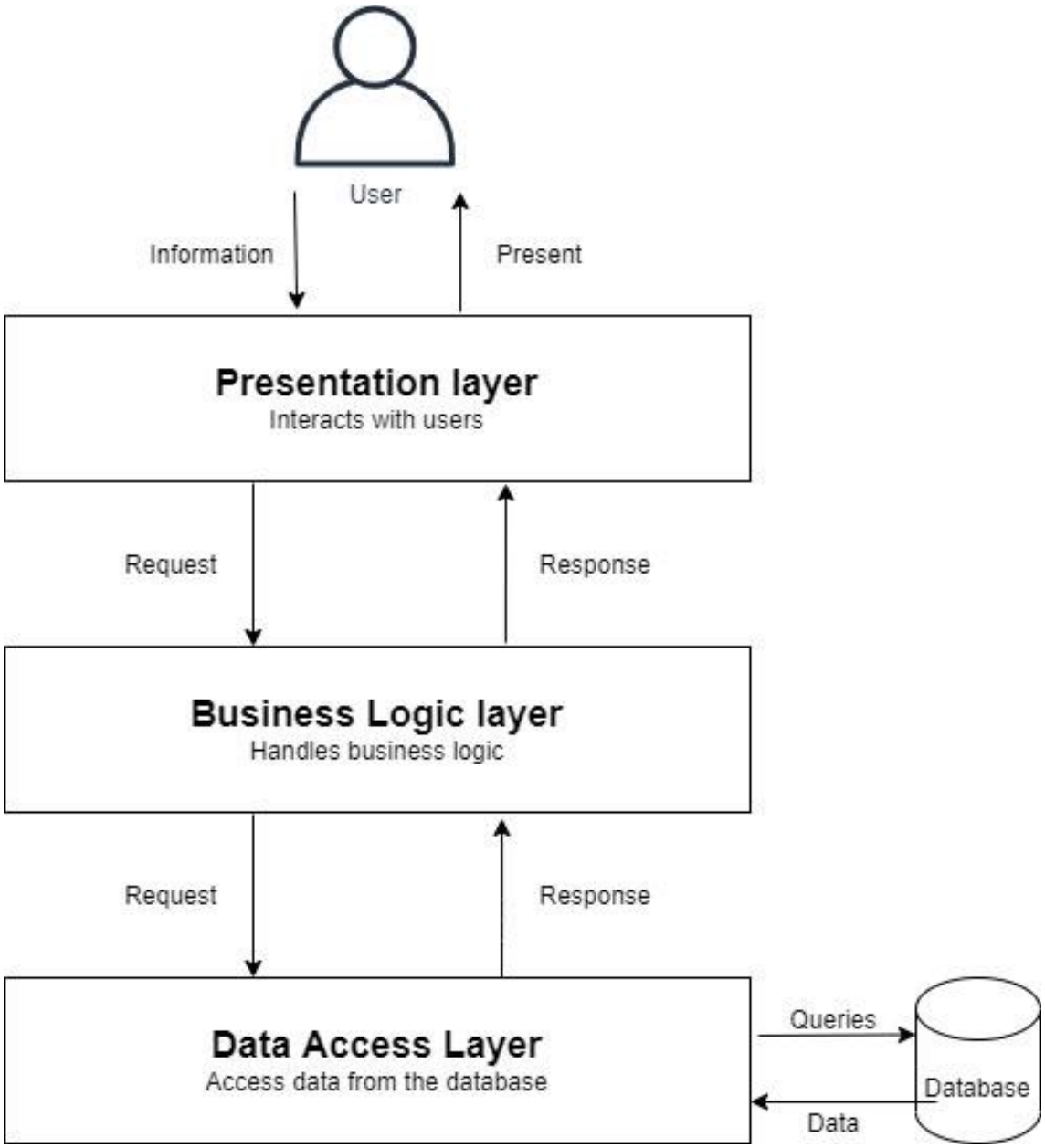
Spring Initialzrit kasutades on baasprojektis olemas juba mõned klassid, et rakendust jooksutada. Projektis kasutatav struktuur on siiski vaja ise välja mõelda. Selle tagarakenduse arendamise käigus kasutatakse kolme-kihilist arhitektuuri. Kolm kihti on esitluskiht, äriloogika kiht ning andmebaasiga suhtlemise kiht (Joonis 10).

Esitluskiht on tagarakenduse arhitektuuri kõige kõrgem kiht. Selle kihiga suhtleb kasutaja ning see saadab rakenduse teise kihtide poolt saadud andmeid edasi eesrakendusele. Esitluskihis asuvad kõik rakenduse veebikontrollerid, mis eesrakenduse pöördumisel teiste kihtidega suhtleb. [29]

Rakenduse keskmes asub tagarakenduse äriloogika kiht. Äriloogika kiht suhtleb nii temast kõrgemal oleva esitluskihi kui ka madalama andmebaasi suhtluskihiga. Äriloogika üks oluline eesmärk on tagada koodi hea loetavus. See tähendab seda, et kogu rakenduse loogika poleks samas kohas, kus on rakenduse kontrollerid. Esitluskiht saadab äriloogika

kihile päringu, selle peale suhtleb äriloogika kiht andmebaasi kihiga, kust pärib vajalikud andmed. Teinekord ei ole vaja eesrakendusele kuvada kõiki andmeid, mis on andmebaasis. Selle jaoks tehakse andmetega vajalikud ärilised muudatused ning saadetakse tagasi esitluskihile. [30]

Rakenduse kõige madalam kiht on andmebaasiga suhtlemise kiht. Seda kasutatakse andmebaasi kirjete lugemiseks, sisestamiseks, uuendamiseks ja kustutamiseks. Andmebaasi kiht suhtleb ainult endast kõrgemal oleval äriloogika kihiga. Äriloogika kiht saadab andmebaasi kihile päringu ning seejärel otsitakse andmebaasi konfiguratsiooni abil välja vajalikud andmed. [30]

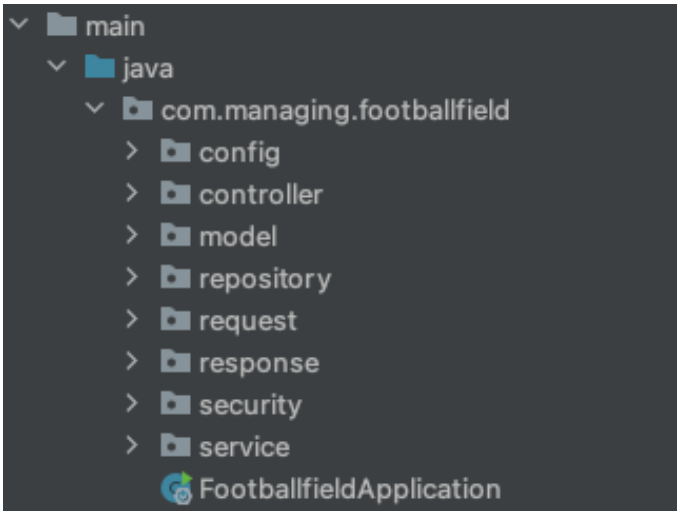


Joonis 10. Kolme-kihiline arhitektuur [30]

Kolme-kihilisel arhitektuuril on palju eeliseid, kuid peamine eelis on koodi loetavus. Iga kiht tegeleb täpselt selle jaoks mõeldud tegevusega ning koodist on lihtne aru saada ka kolmandatel isikutel. Seda on ka väga lihtne muuta ja arendada. Kuna igal kihil on oma ülesanne, siis on lihtsam leida vigu ning neid parandada. Samuti väldib see kiht koodikorduste tekkimist, sest kõiki erinevaid klasse saab kihtides kasutada. [31]

Projektis kasutatavate klasside haldamiseks on struktureeritud ka erinevad kaustad (Joonis 11), mis kasutavad kolme-kihilise arhitektuuri meetodit. Loodud kasutad on:

- Config – selles kasutas asuvad rakenduse konfiguratsioonifailid.
- Controller – selles kaustas asuvad esitluskihi klassid koos veebikontrolleritega.
- Model – selles kaustas asuvad rakenduse andmemudelid.
- Repository – selles kaustas asuvad rakenduse andmebaasi kihiga seotud liidesed.
- Request – selles kaustas asuvad rakenduse veebikontrolleritele saadetud päringute klassid.
- Response – selles kaustas asuvad rakenduse veebikontrollerite tagasi saadetavate päringute klassid.
- Security – selles kaustas asuvad rakenduse turvalisusega seotud klassid.
- Service – selles kaustas asuvad rakenduse äriloogika kihiga seotud klassid.



Joonis 11. Tagarakenduse paketistruktuur

4.2.2 Tagarakenduse turvalisus

Spring Booti kasutamise suur eelis on Spring Security raamistik, mis aitab tagada rakenduse turvalisust. See on väga laialdaselt kasutusel just sellepärast, et on hästi kohandatav iga rakenduse eripärade järgi. Raamistik keskendub rakenduse autentimisele ja autoriseerimisele. Autentimisel kontrollitakse kasutaja tunnuseid ning autoriseerimisel kontrollitakse, kas vastavale kasutajale on soovitud tegevused lubatud. [32]

Rakenduse autenditud kasutajate eristamiseks kasutatakse JSON (*JavaScript Object Notation*) Web Tokenit ehk JWT. JWT on loodud selle jaoks, et edastada turvaliselt andmeid JSON-objektina ühelt osapoolelt teisele. See võib kasutada erinevaid algoritme, kuidas andmeid salastada, kuid antud rakenduse käigus kasutati HMAC (*Hash-Based Message Authentication Code*) algoritmi, täpsemalt HS512. JWT koosneb enamasti kolmest osast: päisest, sisust ja allkirjast. Päis sisaldab enamasti JWT algoritmi tüüpi. Sisu võib olla väga erinev ning seda saab iga kasutaja ise määrata. Antud rakenduse JWT sisuks on kasutajanimi ja kasutaja rollid. Allkiri luuakse päise ja sisu allkirjastamisest valitud algoritmiga. Allkirja kasutatakse, et kontrollida, kas sisu on muudetud. [33]

Mõnede rakenduste puhul piisab ainult autentimisest, kuid antud rakendusel on erinevatel rollidel teistsugused võimalused. Seega tuleb kasutada ka autoriseerimist. Autoriseerimisel kasutatakse JWT, mille küljes on kasutaja erinevad rollid. Enne päringu jõudmist kontrollerini, jõuab JWT turvalisuse kihti. JWT küljest dekodeeritakse kasutaja erinevad rollid ning kontrollitakse, kas tal on ligipääs antud päringule või mitte. [34]

4.2.3 REST API

REST API on rakenduse liides, mis kasutab HTTP-päringuid (*HyperText Transfer Protocol*), et tuvastada GET, POST, PUT ja DELETE meetoditel andmete edastamist. [35] Rakenduse REST API päringute defineerimisel on kasutatud eelnevalt koostatud kasutajalugusid. Tabelis 3 on välja toodud rakenduse päringud koos tüübi ja *URI-ga* (*Uniform Resource Identifier*). Rakenduse radade moodustamisel on kasutatud ühtset struktuuri. Rajad algavad *api/v1/,* kus api sümboliseerib REST API arhitektuuri ning *v1* sümboliseerib rakenduse versiooni.

Spring Boot raamistik tunneb ära REST API kontrollerite klassid esitluskihis *@RestController* annotatsiooni järgi. Iga klass on annoteeritud ka *@RequestMapping* annotatsiooniga, mis täpsustab selle konkreetse klassi raja. Igas klassis on meetodid, kus

on sellele kontrollerile omased API otsad. Eraldi meetodid tunneb Spring Boot raamistik ära *@Getmapping, @Putmapping, @Postmapping ja @Deletemapping* annotatsioonidega. Parameetrid annoteeritakse *@PathVariable* annotatsiooniga. Mõnikord on saadetud suuremad päringud, need on annoteeritud *@RequestBody* annotatsiooniga.

Tabel 3. Rakenduses kasutatavad API päringud

Päring	**Tüüp**	**Rada**
Vabade aegade leidmine jalgpalliväljaku järgi	GET	/available-time/{footballFieldId}/{date}
Vabade aegade leidmine kindlate kuupäevade vahel	GET	/available-time/available/{footballFieldId}/{startDate}/{endDate}/{startTime}
Vabade aegade uuendamine	PUT	/available-time/{availableTimeId}/{available}
Vabade aegade lisamine	POST	/available-time/set-new
Kõik toimunud broneeringud kasutaja järgi	GET	/booking/user/{userId}/{pastBookings}
Kõik toimunud broneeringud väljakute omaniku järgi	GET	/booking/manager/{managerId}/{pastBookings}
Broneeringu salvestamine	POST	/booking/save
Broneeringu kustutamine	DELETE	/booking/{bookingId}
Broneeringu aktsepteerimise muutmine	PUT	/booking/accepted/{bookingId}
Broneeringu tasumise muutmine	PUT	/booking/paid/{bookingId}
Kõik jalgpalliväljakud	GET	/football-fields
Jalgpalliväljak id järgi	GET	/football-fields/{footballFieldId}
Jalgpalliväljaku salvestamine	POST	/football-fields/save-new

4.2.4 Tagarakenduse metoodika kasutamine

Tagarakenduse arendamise käigus sai valitud metoodikas testipõhine arendus. Testipõhise metoodika erinevad nüansid on välja toodud peatükis 3.3.3. Selle metoodika kasutamise juures on väga oluline, et enne koodi kirjutamist, mõtleks arendaja detailselt läbi, kuidas saab rakendus üles ehitatud. Vastasel juhul on metoodika kasutamine ebaefektiivne.

Metoodika kasutamise alguses lõi autor väga lihtsa testklassi, mis koosnes peamiselt soovitud meetoditest. Seejärel kirjutas autor valmis testi väga lihtsa loogika. Pärast seda tuli luua ka loogikaklassid ning veenduda, et testid läbivad. Kui testid läbisid, sai edasi täiendada loogikat. Loogika täiendamisel pidi ka kogu aeg täiendama teste, sest muidu ei läbinud testmeetodid kontrolli.

Sõltuvalt loogika keerukusest, sai mõne meetodi puhul mitu korda täiendatud testmeetodeid. Sellise metoodika kasutamine oli väga efektiivne, sest tagarakendusest suurem osa meetodeid sai kaetud (Joonis 12).

Element	Class, %	Method, %	Line, %
model	100% (5/5)	57% (31/54)	66% (47/71)
service	100% (5/5)	76% (26/34)	65% (122/187)
controller	100% (4/4)	100% (18/18)	100% (18/18)
FootballfieldApplication	100% (1/1)	0% (0/1)	50% (1/2)
repository	100% (0/0)	100% (0/0)	100% (0/0)

Joonis 12. Tagarakenduse loogikaklasside katvuse statistika

4.3 Eesrakenduse arendamine

Eesrakenduse arendamine on jaotatud kolme peatükki: eesrakenduse struktuur, eesrakendus klientidele ja eesrakendus väljakute omanikele. Järgnevates peatükkides on lähemalt kirjeldatud veebirakenduse eesrakenduse arenduse erinevaid aspekte.

4.3.1 Eesrakenduse struktuur

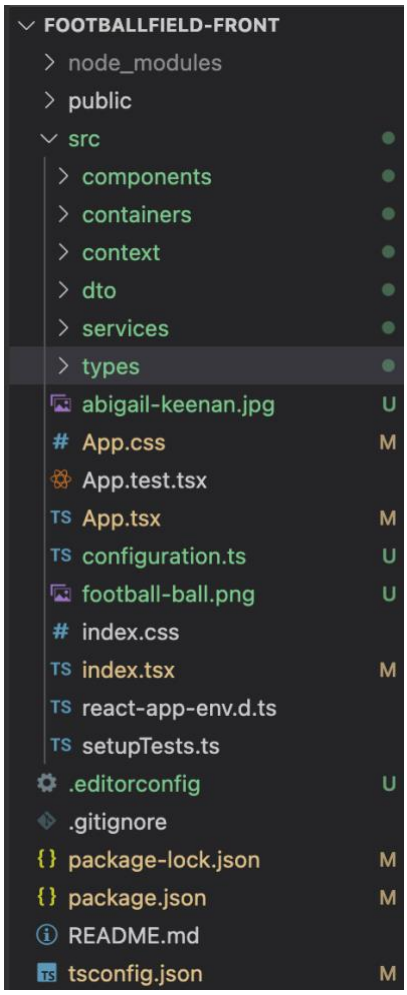
Eesrakendus on kasutajaliides, mis on peamine viis, kuidas kasutaja suhtleb rakendusega. Nagu analüüsi peatükis selgus, siis eesrakenduse loomiseks kasutatakse *TypeScript* programmeerimiskeelt ning *React* raamistikku. Lisaks kasutatakse disaini tegemiseks Bootstrapi disainiraamistikku.

Võrreldes tagarakendusega, on eesrakenduse struktuur oluliselt lihtsam kui tagarakenduse puhul. Reacti projekti saab luua sarnaselt Spring Boot projektile lihtsalt kasutades *Create React App* lahendust. Selle jaoks on vajalik *Node Package Manager*, mis haldab projekti ning käivitab projekti. Samuti sisaldab *Create React App* lahendus juba vajalikke teeke nagu *React-DOM* ja *React-Scripts.* [36]

Projekti loomisel on olemas juba mõned klassid. Olemas on näiteks index.html, App.tsx ja index.tsx klassid. Enamik olemasolevates klassidest ning rakenduse arendamise käigus loodavad klassid on JSX klassid. JSX lubab kasutada XML süntaksit *TypeScript* koodis ning on hea viis hoida ühes kohas nii rakenduse loogikat kui ka disaini. [37]

Projektis kasutatavate klasside haldamiseks on struktureeritud ka erinevad kaustad (Joonis 13). Loodud kasutad on:

- Components – selles kaustas on rakenduse üldkasutatavad komponendid.
- Containers – selles kaustas on täiendavad kaustad iga veebirakenduse lehe eraldiseisvate klasside kohta.
- Context – selles kaustas on rakenduse olekuga seotud klass.
- DTO – selles kaustas on rakenduse andmemudelid.
- Services – selles kaustas on rakenduse üldkasutatavad loogika klassid andmete pärimiseks tagarakendusest.
- Types – selles kaustas on rakenduse üldkasutatavad mudelid.

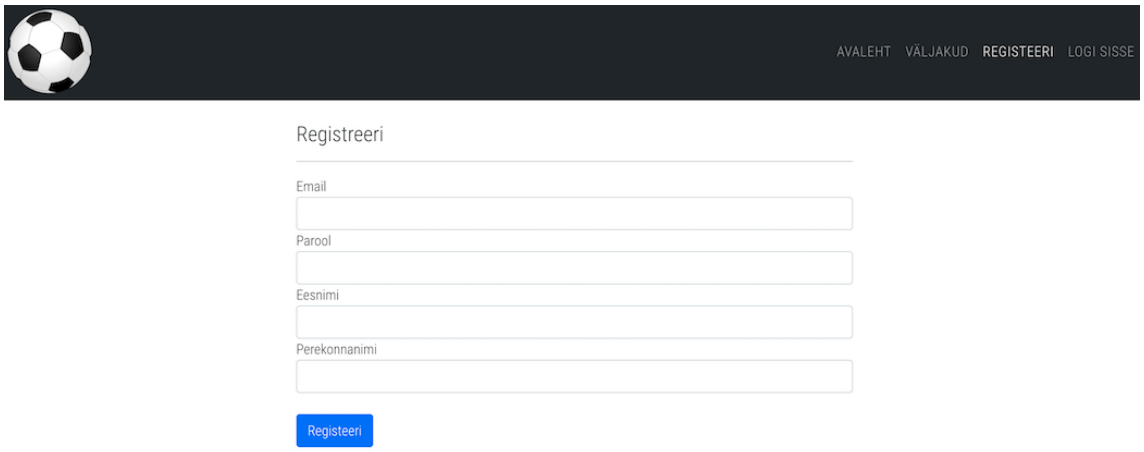


Joonis 13. Eesrakenduse paketistruktuur

Eesrakenduse käivitamisel käivitatakse esimesena index.tsx fail. See renderdab omakorda App.tsx komponendi. Selles komponendis hoitakse rakenduse erinevaid radasid (inglise keeles *route*). Need on eraldatud *Routes* komponendiga ning suunavad korrektse URL (*Uniform Resource Locator*) minemisel õigele komponendile. Samuti hoitakse siin komponendis ka rakenduse olekut, mis koosneb JWT žetoonist, nimest, rollist ja kasutaja ID-st.

4.3.2 Eesrakendus klientidele

Veebirakendusele minnes kuvatakse esmalt veebirakenduse avalehte. Paremal üleval on navigatsiooniriba, vasakul üleval logo ning keskel on rakenduses kuvatav info. Samuti on terve lehekülg kohanduva disainiga ehk on endiselt korrektne brauseri akna suuruse muutmisel. Navigatsiooniriba paremal üleval asub nupp **Registreeri**, mis viib registreerimisvormile (Joonis 14). Registreerimise õnnestumisel suunatakse kasutaja tagasi esilehele.

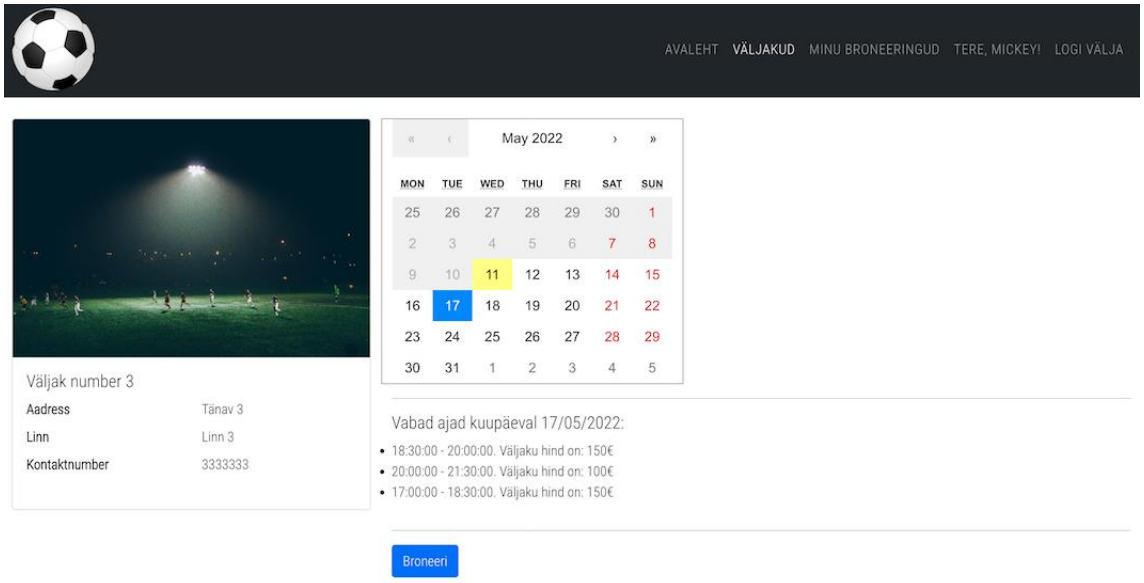


Joonis 14. Veebirakendusele registreerumise leht

Pärast registreerimist logitakse kasutaja automaatselt sisse ning tagarakendus tagastab kasutajale omase JWT tähise ja kasutaja andmed. Navigatsiooniribal asetsev **Registreeri** nupp asendub kasutajanime ning tervitusega ja **Logi sisse** nupp asendub **Logi välja** nupuga. **Logi välja** nupu vajutamisel suunatakse kasutaja tagasi esilehele. Navigatsiooniriba kaudu saab klient navigeerida end väljakute vaatesse (Joonis 15). Väljakute vaates tagastab tagarakendus kõik rakenduses olevad väljakuid, klient saab näha erinevaid väljakuid ja nende andmeid.

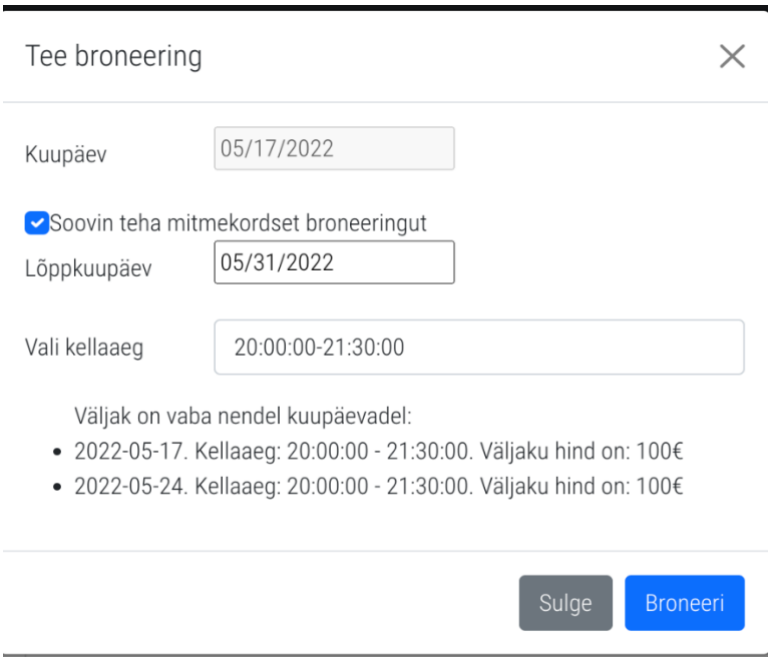
Joonis 15. Veebirakenduse väljakute leht kliendile

Väljakute lehelt saab klient navigeerida vabade aegade lehele (Joonis 16). Vabade aegade lehel kuvatakse kliendile samu väljaku andmeid ning lisaks kalendrit. Kalendris saab klient valida kuupäeva ning selle all kuvatakse selle peale sellel kuupäeval olevad vabad ajad, juhul kui need eksisteerivad. Iga kuupäeva muutuse peale, päritakse tagarakenduselt sellel kuupäeval olevad vabad ajad.



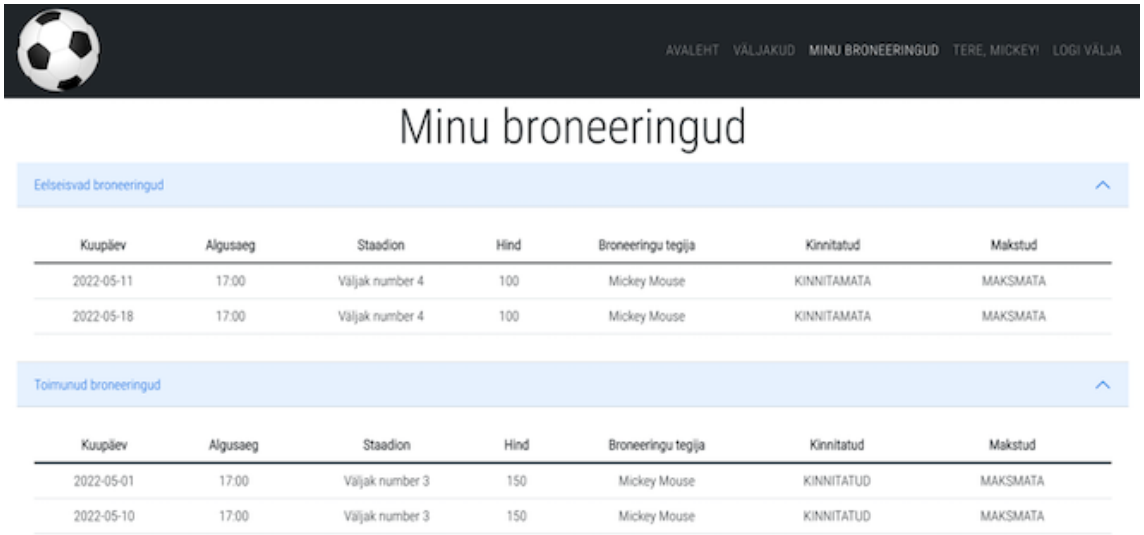
Joonis 16. Veebirakenduse väljaku vabade aegade leht kliendile

Vabade aegade vaatest saab klient broneerida vabu aegu. Selle jaoks tuleb vajutada **Broneeri** nupule ning kliendile avatakse modaal. Modaalis saab valida ka mitmekordse broneeringu tegemise valiku. Mitmekordset broneeringut tehes, kuvatakse kõik selles vahemikus olevate samade nädalapäevade vabad ajad (Joonis 17). Seejärel saab klient kas oma broneeringu kinnitada või modaalist lahkuda.



Joonis 17. Veebirakenduse väljaku vabade aegade broneerimise modaal kliendile

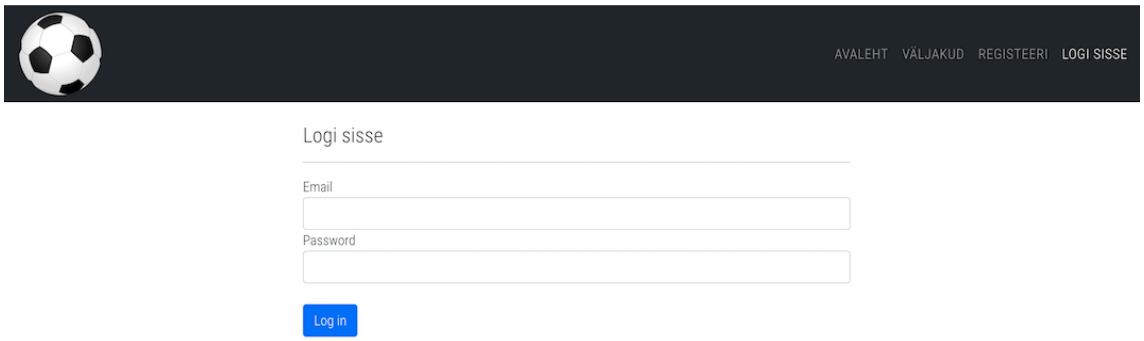
Enda tehtud broneeringuid näeb klient navigatsiooniribalt **Minu broneeringute** lehele navigeerides. Kliendile avaneb broneeringute vaade, kus on nii Eelseisvad broneeringud kui ka Toimunud broneeringud (Joonis 18). Broneeringute all näeb klient oma broneeringu spetsiifilisi andmeid. Samuti kuvatakse kliendile, kas tema broneering on väljaku omaniku poolt kinnitatud ning kas broneeringu eest on juba tasutud.



Joonis 18. Veebirakenduse broneeringute leht kliendile

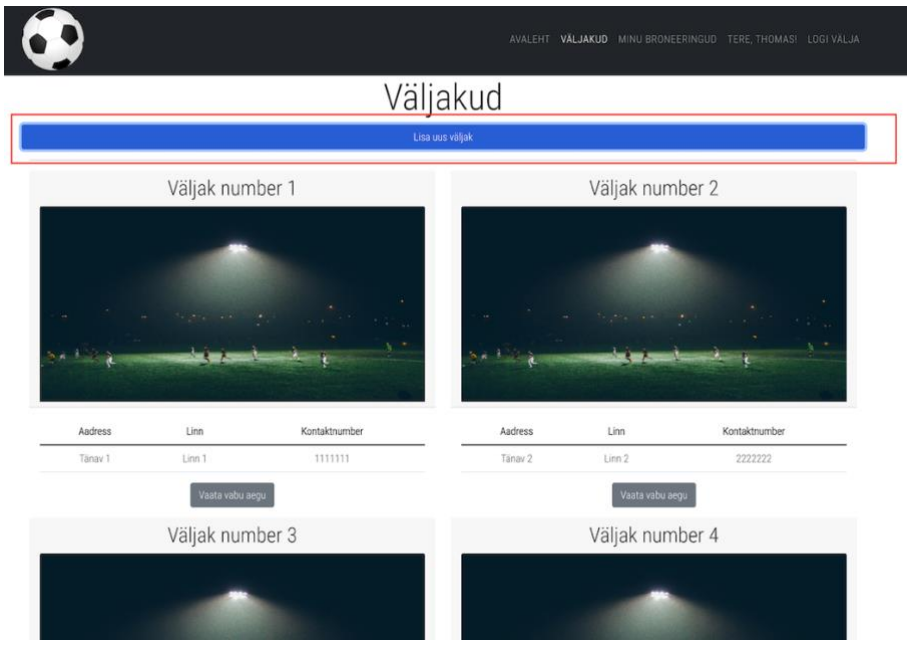
4.3.3 Eesrakendus väljakute omanikele

Veebirakendusele minnes kuvatakse esilehte, millel on navigatsiooniriba. Navigatsiooniribal Logi sisse nupule vajutamisel suunatakse kasutaja sisselogimisvormile (Joonis 19). Sisselogimisvormil tuleb täita e-posti aadress ja salasõnaga. Kui vorm täidetakse korrektse infoga, tagastab tagarakendus JWT ning kasutaja andmed. Kasutaja suunatakse esilehele.

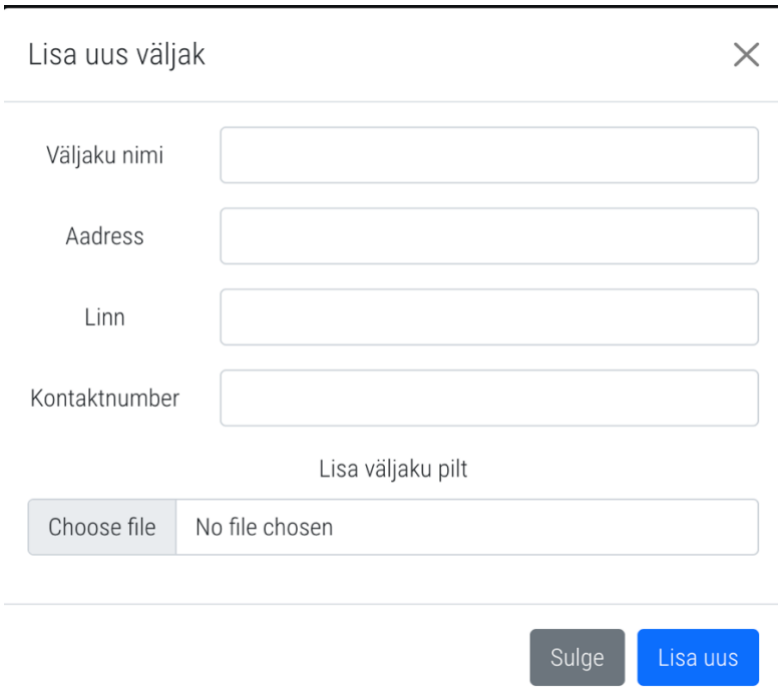


Joonis 19. Veebirakenduse sisselogimise leht kõikidele kasutajatele

Sarnaselt kliendi vaatega on väljakute omanikul navigatsiooniriba samade nuppudega. Küll aga on igalt lehelt mõned nupud peidetud kliendi eest, mis on aga lubatud väljakute omanikele. Väljakute omanikud saavad väljakute vaates lisada uusi väljakuid (Joonis 20), mille peale kuvatakse modaali (Joonis 21), kuhu tuleb sisestada väljaku andmed. Modaali kinnitamisel salvestab tagarakendus andmebaasi uue väljaku.

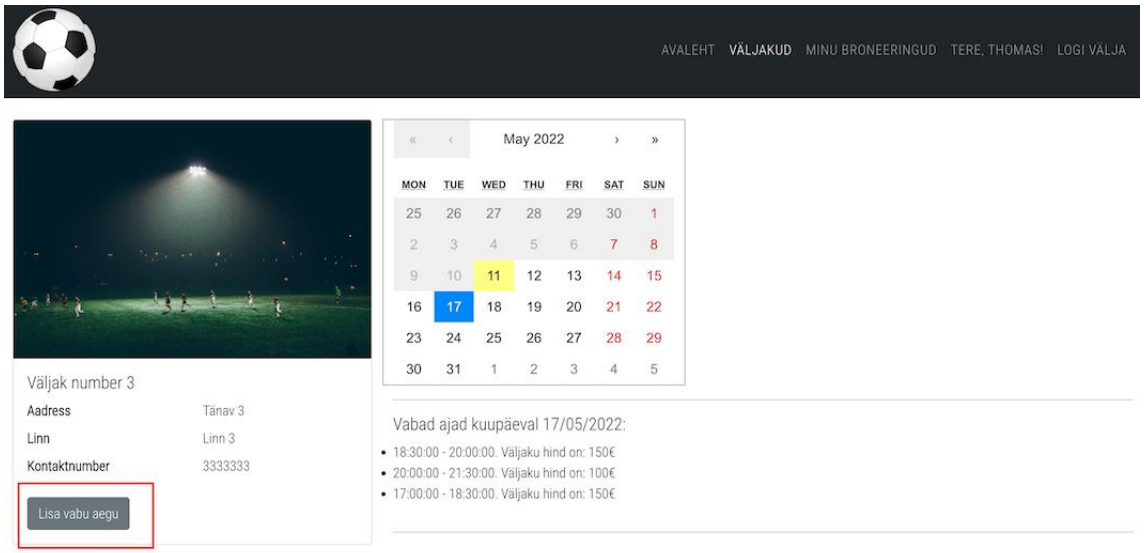


Joonis 20. Veebirakenduse väljakute leht väljakute omanikele

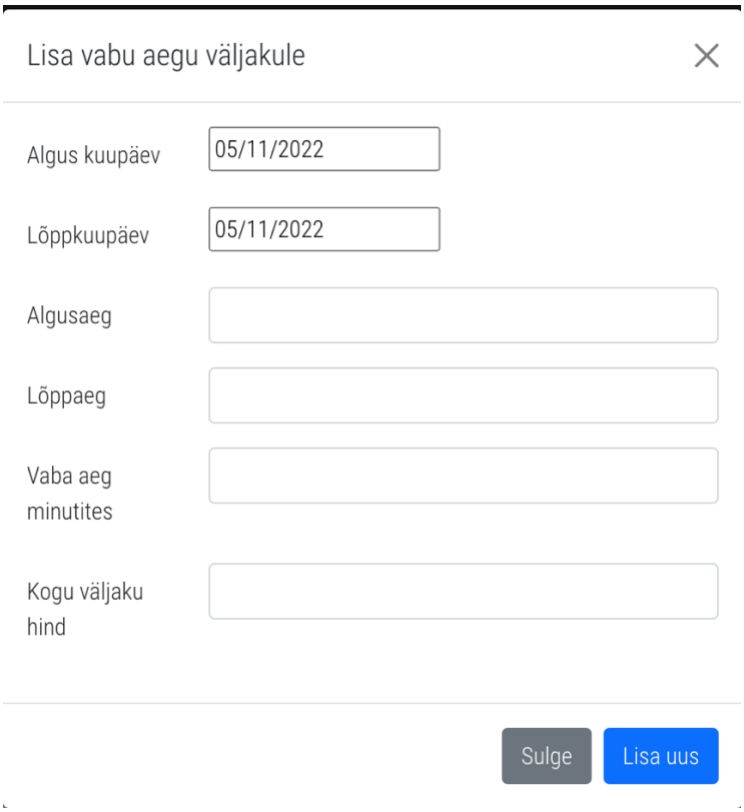


Joonis 21. Veebirakenduse väljakute lisamise modaal väljakute omanikele

Samuti on väljakute omanikel ka vabade aegade lehel nupp Lisa vabu aegu (Joonis 22). Sellele nupule vajutades kuvatakse modaal (Joonis 23), kus väljakute omanik saab lisada vabu aegu konkreetsele väljakule. Aegu saab lisada ka korduvaid. Korduva aja valimisel lisatakse tagarakenduse poolt baasi iga vahemikku jääva valitud nädalapäeva vabad ajad.

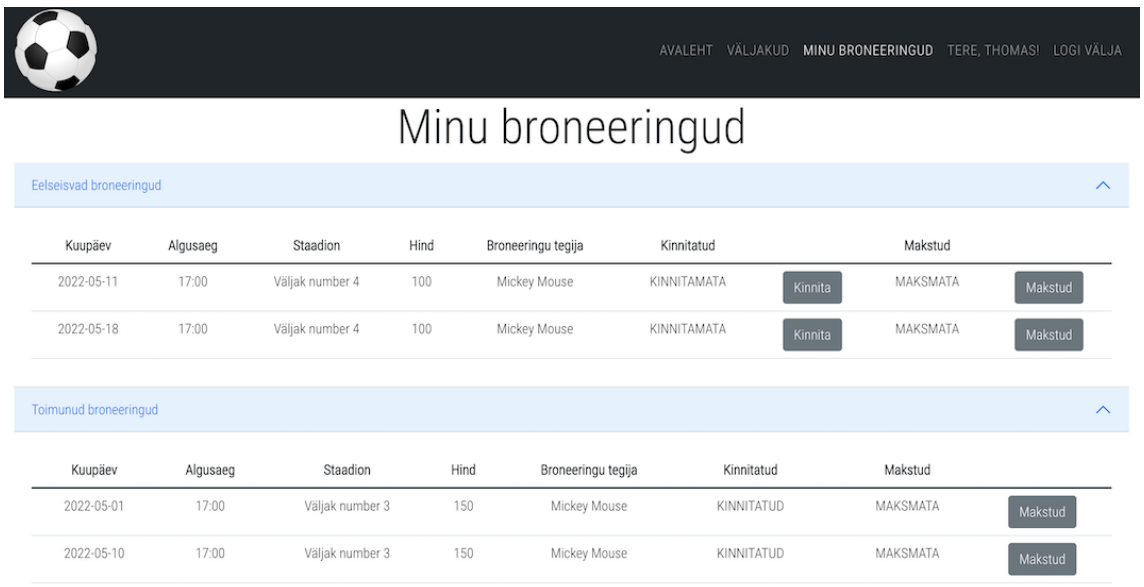


Joonis 22. Veebirakenduse väljaku vabade aegade leht väljakute omanikele



Joonis 23. Veebirakenduse vabade aegade lisamise modaal väljakute omanikele

Minu broneeringud lehel näeb väljakute omanik kõiki oma erinevatel väljakutel tehtud broneeringuid (Joonis 24). Samuti on väljakute omanikul võimalik kinnita broneeringuid ning märkida, et broneeringute eest on tasutud.



Joonis 24. Veebirakenduse broneeringute leht väljakute omanikele

5 Järeldused

Lõputöö tulemusena loodi peaaegu kõikidele funktsionaalsetele nõuetele vastav veebirakendus. Veebirakendus koosneb eesrakendusest ning funktsionaalsetele nõuetele vastavast loogikast koosnevast tagarakendusest. Samuti sai veebirakendusele üles seatud andmebaas andmete hoiustamiseks. Kõik mittefunktsionaalsed nõuded said täidetud. Täitmata jäi funktsionaalsetest nõuetest osa, kus väljakute omanik saaks tühistada broneeringuid.

Väga suur osa tagarakendusest on kaetud testidega ning seda just testipõhise lähenemise pärast. Testipõhise arendamise metoodika oli autorile alguses võrdlemisi keeruline, kuid samas laiendas oluliselt silmaringi. Samuti sunnib selline metoodika arendajat oluliselt täpsemalt läbi mõtlema, kuidas ja mida teha. Siiski tuleb arvestada, et ilmselt on sellise metoodika kasutamine kõige efektiivsem, kui rajada rakendust täiesti algusest. Rakenduse täiendamisel sellise metoodika kasutamine võib olla ebaefektiivne. Seda sellepärast, kuna rakenduse algusest arendamisel saab välja mõelda korrektse struktuuri, kuidas rajada nii testklassid kui ka loogikaklassid. Täiendamisel peab tihti palju uurima ning otsima, milles probleem seisneb. Eriti keeruline tundub testi põhise arendamise kasutamine rakendusel, mille koodi pole autor algusest kirjutanud.

Veebirakenduse arenduse käigus sai kasutatud sellel ajaperioodil uudseid tehnoloogiad. Java ja Spring Boot on üks populaarsemaid tagarakenduse raamistikke, mida kasutavad väga paljud ettevõtted tööturul. Samuti on viimastel aastatel väga populaarne eesrakenduses kasutatud *React* raamistik.

Peamine prioriteet rakenduse edasiarendamisel on täita funktsionaalsed nõuded. See tähendab seda, et väljakute omanikud peaksid saama tühistada nende väljakule tehtud broneeringuid. Seejärel saab mõelda rakenduse detailsemaks muutmisele.

Rakendusele tuleks lisada palju veateateid, mis muudaksid kasutajakogemuse paremaks. Veateated aitavad kasutajal mõista, mis on valesti läinud rakenduse kasutamisel. Veateateid saab olla igasuguseid ning enamasti on nad eristatud tagarakenduse päringu vastusest tulenevast staatuskoodist.

Samuti tuleb mõelda ka erinevatele broneerimisnüanssidele, millega esimeses versioonis polnud arvestatud. Näiteks võiks saada väljakut broneerida ka poole väljaku kaupa. Lisaks võiks olla võimalik otsida ja sorteerida erinevaid väljakuid ning ka kuvada veel rohkem detailset infot väljaku kohta.

6 Kokkuvõte

Käesoleva lõputöö eesmärk oli luua veebipõhine jalgpalliväljakute broneeringute haldamiseks mõeldud rakendus, mis lahendaks praeguse emaili ja telefoni teel broneerimise keerukuse. Veebirakendus peaks looma ühtse süsteemi, kus väljakuid broneerida ning vähendama oluliselt ajakulu erinevate vabade aegade leidmiseks väljakutel.

Rakenduse analüüsi käigus kaardistati funktsionaalsed ning mittefunktsioonalsed nõuded. Need said loodud suheldes erinevate klubi omanike ja väljakute omanikega. Rakenduse analüüsi käigus analüüsiti erinevaid võimalike tehnoloogiaid rakenduse arendamiseks. Rakenduse arendamiseks valiti tagarakenduse puhul *Java* ja *Spring Boot* raamistik ning eesrakenduse puhul *TypeScript* ja *React* raamistik. Andmebaasi tarkvaraks sai *PostgreSQL*.

Rakenduse arendamise peatükis kirjeldati esmalt rakenduse arendamiseks tehtud ettevalmistust. Seejärel kirjeldati nii tagarakenduse kui ka eesrakenduse arendamisel tekkinuid nüansse. Toodi välja rakenduste struktuur, kirjeldati erinevaid metoodikaid, toodi välja, kuidas tagati rakenduses turvalisus ning ka veebirakenduse väljanägemine jooniste abil. Lõputöö lõppes järelduste peatükiga, kus autor analüüsis lühidalt kogu projekti valmimist.

Loodud veebirakenduse projekti võib lugeda õnnestunuks, sest tehtud said peaaegu kõik planeeritud nõuded. Lõputöö tulemusena loodi veebirakendus, millel on kasutajaliides eesrakenduse näol ning serverliides tagarakenduse näol. Samuti sai loodud rakendusele andmebaas. Kuigi üks rakenduse nõuetest jäi tegemata, loob ülejäänud töö hea põhja, et alustada edasiarendustega rakenduses. Kuna rakendus ei ole veel ka avalikult kättesaadav, siis jätkub rakenduse arendustöö.

Kasutatud kirjandus

[1] Riina Leinbock, „Eestis on kõige enam kulturismi ja fitnessi harrastajaid,“ [Võrgumaterjal]. Available: https://www.stat.ee/et/uudised/eestis-koige-enam-kulturismi-ja-fitnessi-harrastajaid. [Kasutatud: 02.04.2022]

[2] StackOverflow, „2021 Developer Survey,“ [Võrgumaterjal]. Available: https://insights.stackoverflow.com/survey/2021#most-popular-technologies-languageprof. [Kasutatud 22.04.2022]

[3] Adam Mateja, „How to Start a Succesful IT Project: 6 Essential Steps,“ [Võrgumaterjal]. Available: https://studiosoftware.com/blog/how-to-start-a-successful-it-project-six-essential-steps/. [Kasutatud 03.04.2022]

[4] Atlassian, „What is Jira Used for?“ [Võrgumaterjal]. Available: https://www.atlassian.com/software/jira/guides/use-cases/what-is-jira-used-for. [Kasutatud 03.04.2022]

[5] Bugreporting, „Jira Pros And Cons | Things Jira is Great at,“ [Võrgumaterjal]. Available: https://www.bugreporting.co/pros-and-cons/jira. [Kasutatud 03.04.2022]

[6] Yoshitaka Shiotsu, „A Beginner’s Guide to Back-End Development,“ [Võrgumaterjal]. Available: https://www.upwork.com/resources/beginners-guide-back-end-development. [Kasutatud: 10.04.2022]

[7] Jose Gomez, „Best Back-end Languages: How to Choose the Perfect One?“ [Võrgumatejal]. Available: https://www.koombea.com/blog/best-back-end-languages-how-to-choose-the-perfect-one/. [Kasutatud 10.04.2022]

[8] AplusTopper, „PHP Advantages and Disadvantages | What is PHP Language? Merits and Demerits of PHP,“ [Võrgumatejal]. Available: https://www.aplustopper.com/php-advantages-and-disadvantages/. [Kasutatud 10.04.2022]

[9] Krzystof Basel, „Python Pros and Cons,“ [Võrgumatejal]. Available: https://www.netguru.com/blog/python-pros-and-cons. [Kasutatud 10.04.2022]

[10] DataFlair, „Pros and Cons of Java | Advantages and Disadvantages of Java,“ [Võrgumatejal]. Available: https://data-flair.training/blogs/pros-and-cons-of-java/. [Kasutatud 10.04.2022]

[11] Altexsoft, „The Good and the bad of C# Programming,“ [Võrgumatejal]. Available: https://www.altexsoft.com/blog/c-sharp-pros-and-cons/. [Kasutatud 10.04.2022]

[12] Pierre Carbonnelle, „PYPL PopularitY of Programming Language,“ [Võrgumatejal]. Available: https://pypl.github.io/PYPL.html. [Kasutatud 10.04.2022]

[13] Kumar Chandrakant, „Why Choose Spring as Your Java Framework?“ [Võrgumatejal]. Available: https://www.baeldung.com/spring-why-to-choose. [Kasutatud 10.04.2022]

[14] Jash Unadkat, „BDD vs TDD vs ATDD: Key Differences,“ [Võrgumatejal]. Available: https://www.browserstack.com/guide/tdd-vs-bdd-vs-atdd. [Kasutatud 11.04.2022]

[15] Jessiva Wilkins, „Front End Developer – What is Front End Development, Explained in Plain English,“ [Võrgumatejal]. Available: https://www.freecodecamp.org/news/front-end-developer-what-is-front-end-development-explained-in-plain-english/. [Kasutatud 11.04.2022]

[16] Wikipedia, „JavaScript,“ [Võrgumaterjal]. Available: https://en.wikipedia.org/wiki/JavaScript. [Kasutatud 11.04.2022]

[17] Stxnext, „TypeScript vs. JavaScript: What is TypeScript and how is it different from JavaScript?“ [Võrgumaterjal]. Available: https://www.stxnext.com/blog/typescript-pros-cons-javascript/. [Kasutatud 11.04.2022]

[18] Jeel Patel, „List of 10 Best Front end Frameworks to Use For Web Development,“ [Võrgumaterjal]. Available: https://www.monocubed.com/blog/best-front-end-frameworks/. [Kasutatud 12.04.2022]

[19] Google Trends, „Google Trends,“ [Võrgumaterjal]. Available: https://trends.google.com/trends/explore?date=today%205-y&q=%2Fm%2F012l1vxv,%2Fg%2F11c0vmgx5d,%2Fg%2F11c6w0ddw9. [Kasutatud 17.04.2022]

[20] Jordana A., „What is Bootstrap?“ [Võrgumaterjal]. Available: https://www.hostinger.com/tutorials/what-is-bootstrap/. [Kasutatud 12.04.2022]

[21] Alexandre Ouellette, „What is Bootstrap: A Beginner’s Guide,“ [Võrgumaterjal]. Available: https://careerfoundry.com/en/blog/web-development/what-is-bootstrap-a-beginners-guide/. [Kasutatud 12.04.2022]

[22] Oracle, „What is a Database?“ [Võrgumaterjal]. Available: https://www.oracle.com/database/what-is-database/. [Kasutatud 13.04.2022]

[23] Jason Byrd, „The Basics of Back End Development & Database Design,“ [Võrgumaterjal]. Available: https://www.3mediaweb.com/blog/back-end-database-design/. [Kasutatud 13.04.2022]

[24] Liliia Harkushko, „Vital Things to Consider When Choosing a Database for Your App,“ [Võrgumaterjal]. Available: https://yalantis.com/blog/how-to-choose-a-database/. [Kasutatud 13.04.2022]

[25] Sagar Bhatia, „PostgreSQL vs MySQL: Everything You Need to Know in 2022,“ [Võrgumaterjal]. Available: https://hackr.io/blog/postgresql-vs-mysql. [Kasutatud 13.04.2022]

[26] Max Rehkopf, „Agile epics: definition, examples, and templites,“ [Võrgumaterjal]. Available: https://www.atlassian.com/agile/project-management/epics. [Kasutatud 15.04.2022]

[27] Brian Burns, „MindMapping: 7 ways to use mindmapping as a developer,“ [Võrgumaterjal]. Available: https://www.netguru.com/blog/mindmapping-7-ways-to-use. [Kasutatud 15.04.2022]

[28] Spring, „Building an Application with Spring Boot,“ [Võrgumaterjal]. Available: https://spring.io/guides/gs/spring-boot/. [Kasutatud 16.04.2022]

[29] FineReport, „3-Tier Architecture: Everything You Need to Know,“ [Võrgumaterjal]. Available: https://www.finereport.com/en/product-functions/3-tier-architecture.html [Kasutatud 03.05.2022]

[30] Enlab, „How to build and deploy a three-layer architecture application with C#,“ [Võrgumaterjal]. Available: https://enlabsoftware.com/development/how-to-build-and-deploy-a-three-layer-architecture-application-with-c-sharp-net-in-practice.html [Kasutatud 03.05.2022]

[31] IBM, „Three-Tier Architecture,“ [Võrgumaterjal]. Available: https://www.ibm.com/cloud/learn/three-tier-architecture [Kasutatud 03.05.2022]

[32] Spring, „Spring Security,“ [Võrgumaterjal]. Available: https://spring.io/projects/spring-security [Kasutatud 04.05.2022]

[33] JWT, „Introduction to JSON Web Tokens,“ [Võrgumaterjal]. Available: https://jwt.io/introduction. [Kasutatud 04.05.2022]

[34] Marco Behler, „Spring Security: Authentication and Authorization In-Depth,“ [Võrgumaterjal]. Available: https://www.marcobehler.com/guides/spring-security. [Kasutatud 04.05.2022]

[35] Andy Lian, „Rest Controller – Building REST API in Microservices,“ [Võrgumaterjal]. Available: https://codeburst.io/rest-controller-building-rest-api-638d3ff4fa71. [Kasutatud 04.05.2022]

[36] React, „Create a New React App,“ [Võrgumaterjal]. Available: https://reactjs.org/docs/create-a-new-react-app.html. [Kasutatud 06.05.2022]

[37] React, „Introducing JSX,“ [Võrgumaterjal]. Available: https://reactjs.org/docs/introducing-jsx.html. [Kasutatud 06.05.2022]

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

Mina, Karl Mihkel Pohga

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "Veebirakendus jalgpalliväljakute broneeringute haldamiseks", mille juhendaja on German Mumma
   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.

16.05.2022

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 – Ees -ja tagarakenduse projektid

Tagarakendus - https://gitlab.cs.ttu.ee/kapohg/footballfield

Eesrakendus - https://gitlab.cs.ttu.ee/kapohg/footballfield-front