TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Rauno Lõhmus 175899IDDR

Rallitulemuste kuvamise ja võrdlemise rakenduse arendus

Diplomitöö

Juhendaja: Meelis Antoi
Magister

Tallinn 2021

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: Rauno Lõhmus

04.12.2020

Annotatsioon

Antud diplomitöö eesmärk oli valmis arendada rakendus, kust on lihtne leida rallitulemusi. Rallitulemuste leidmiseks tehti erinevaid otsingufiltreid ning loodi võimalus võrrelda ühe rallietapi kõikide sõitjate etapi läbimise keskmist kiirust. Rakenduse arenduse eesmärgiks oli lihtsustada tulemuste otsimist. Rakenduse lähtekood on avalik ning selle arendamise juures peeti silmas rakenduse jätkusuutlikust.

Arenduse käigus loodi kaks rakendust – veebiteenus ning veebirakendus. Kuna rallitulemusi algselt väga ei olnud, siis loodi võimalus neid käsitsi muuta, lisada ja kustutada. Samuti käib rakendus teistelt veebilehtedelt andmeid küsimas. Rakendus tagastab ühe lehena rallitulemused, kust on võimalk otsida nii võistleja, etapi nime kui ka ralli nime järgi. Arendus jagati kolmeks tükiks – veebiteenus, veebirakendus ja testimine.

Rakendust arendades peeti silmas rakenduse jätkusuutlikust, kuid samas ka seda, et see oleks võimalikult uudne. Veebirakenduse arendamisel lähtuti sellest, kuidas oleks kasutajal seda hea ja mõnus kasutada. Rakenduste arenduses on välja toodud üldine protsess kogu arendusest. Diplomitöö lõpptulemuseks on töötav rakendus.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 35 leheküljel, 6 peatükki, 16 joonist, 2 tabelit.

Abstract

Development of Rally Results Layout and Comparison Application.

The aim of this thesis was to develop an application where it is easy to find rally results or filter them. Purpose of this applciation is to make easier to find results of your favorite competitior or find easily specific rally results. Application is open source.

During application development, author did not have any rally results, so he included solution to modify add or delete rally results, and application goes from time to time to find new rally results from other websites. Solution was separated into two applications – web services and web application. Application shows rally results as single page and offers filters to search faster and more convenient way to find results of favorite comeptitor or just specific event results.

On development process, author has kept in mind, that application should be easily usable, sustainable and modern. Development of this application was seperated into 3 pieces – development of web services, development of web application and testing.

The thesis is in Estonian and contains 35 pages of text, 6 chapters, 16 figures, 2 tables.

Lühendite ja mõistete sõnastik

AOP	*Aspect Oriented Programming* – aspekt orienteeritud programmeerimine
API	Application Programming Interface – Rakenduse liidestus
CD	*Continues delivery* – peale igat muudatust tehakse rakendusele valmis tarne
CI	*Continues integration* – Automaatne protsess, kus peale igat muudatust kood kompileeritakse, testitakse ning siis viiakse üle üldisesse repositooriumi
criteria API	Java rakenduse liidestus, millega saab teha andmebaasi päringuid
cron-avaldis	6 sõne, mis on omavahel tühikutega eraldatud ning määrab ära ajamomendi käsu täitmiseks
CRUD	Lisamise, otsimise, uuendamise ning kustutamise tegevused
DI	*Dependcy Injection* – sõltuvuste süstimine objekti
DTO	*Data Transfer Object* – Rakenduse objekt, mida kasutakse andmete tagastamiseks veebiteenusest
HTML	*Hypertext Markup Language* – keel, millega saab kirjutada veebilehe välja nägemist
IOC	*Inversion of Control* – Objektide mälus hoidmine ning teistele objektidele selle ette andmine
JSON	*Javascript Object Notation* ehk andmevahetusvorming
JSP	*Java Server Page* – tehnoloogia, millega saab arendada veebilehti kasutades Java programmeerimiskeelt
kompileerimine	Protsess, kus viiakse kood kõrgema taseme programmeerimiskeelest madalama taseme programmeerimiskeelele
localstorage	Brauseris olev salvestusruum
mapper	Meetod, mis viib objekti andmed teise objekti. Näiteks olem objekt konverditakse rakenduse objektiks
MVC	*Model View Controller* – arendusmuster, kus eraldatakse rakenduse väljanägemine, interaktsioonid ning andmeobjekti mudel

POJO	*Plain Old Java Object* – Java objekt, mis hoiab andmeid
võrguport	Liides, mille kaudu saab suhelda programmiga üle võrgu
REST	*Representational state transfer* – arhitektuuri stiil andmete vahetamise kihi jaoks
SPA	*Single Page Application* – üheleherakendus
SQL	*Structured Query Language* – andmete päringukeel
URL	Interneti aadress

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Käesoleva diplomitöö raames luuakse veebirakendus rallisõitude tulemuste kuvamiseks. Antud tööga luuakse rakendus, mis suudab filtreerida Eestis toimunud rallide tulemusi nii sõitja, ralli või ralli ja sõitja järgi ning lisada rallide tulemusi ning kuvada statistikat.

Töö motivatsiooniks oli raskendatud Eesti rallisõidutulemuste leidmine ning võrdlemine. Ainsad veebilehed, kus on võimalik leida kokkuvõtlike Eesti rallitulemusi on autosport.ee, rally.ee ning ralliportaal.ee. Veel on võimalik leida kodumaiseid rallitulemusi erinevatest spordiblogidest, näiteks Postimehe spordiblogist või Delfi spordiblogist, kuid kummaski pole tulemused kokkuvõtlikult kuvatud. Kõigist nimetatud allikatest on keeruline otsida või võrrelda erinevate sõitjate tulemusi.

Antud lõputöö teema valimise ajal olemasolevad rallitulemuste allikad ei ole väga kasutajasõbralikud ning rallitulemuste leidmine ja võrdlemine ei ole kõige mugavam. Lehel kuvatakse suuremate Eesti rallide tulemused tabelina ning tulemusi saab otsida ainult rallietappide ja võistlusklasside järgi. Veebilehel ei ole võimalik mugavalt võrrelda lemmiksõitjat mõne muu sõitjaga või otsida lemmiksõitja rallitulemusi.

Rallisõitude tulemuste leidmise ning võrdlemise lihtsustamiseks tehakse rakendus, mis suudab rallide tulemusi andmebaasi salvestada ning neid kasutajale kuvada. Otsimise lihtsustamiseks on võimalik kasutada erinevaid filtreid. Antud lahendus teeb rallitulemuste otsimise mugavamaks ja lihtsamaks. Rakendus võimaldab lisada infot rallide kohta nii käsitsi kui ka oskab otsida muudest rallitulemuste allikatest. Rakendus suudab saata andmeid kui ka neid vastu võtta teistest rakendusest.

Arendatud rakendus on suunatud kõikidele rallihuvilistele ning ralliga tegelevatele inimestele.

1.1 Metoodika

Antud lõputöö raames selgitakse teema aktuaalsust. Vaadeltakse lähemalt olemasolevaid rallitulemuste kuvamise rakendusi ning analüüsitakse, mis on nende negatiivsed ja

positiivsed aspektid. Tuuakse välja nii nende lahenduste eelised kui ka puudused ning selgitakse välja, mis võimalused peaksid olema arendataval rakendusel.

Järgmises osas analüüsitakse, millised võiksid olla rakenduse nõuded ehk mida peab olema rakendus suuteline tegema. Järgnevalt leitakse ning põhjendatakse tehnoloogiate valikut rakenduse arendamiseks. Samuti tuuakse välja üldine arhitektuuri pilt ning selle kirjeldus.

Rakenduse arenduses on ära kirjeldatud nii serverrakenduse arendamise kui ka klientrakenduse arendamise protsess. Kokkuvõtlikus osas võetakse kokku valminud rakendus ning tuuakse välja, kuidas tuleks rakendust edasi arendada.

2 Probleemi ülevaade

Iga spordihuviline või sportlase tuttav soovib näha sporditulemuste andmeid oma lemmiku kohta ning selle jaoks kasutakse erinevaid veebilehti, et näha tulemusi nii oma lemmikute kui ka lihtsalt sportlaste kohta. Hasartmängurid kasutavad tulemusi väljastavaid veebilehti, et saada lisainformatsiooni, kelle peale panustada.

Teema idee tekkis sellest, et autor on ise suur spordihuviline ning jälgib palju rallisporti. Enamasti leiab rallitulemusi suuremate rallide kohta, kasutades mõnda kohaliku ralliportaali. Sellest tekkis idee luua veebirakendus, mis oleks tänapäevane ning, mis suudaks kuvada nii väiksemate kui ka suuremate rallide tulemusi igal pool maailmas. Kindlasti oleks mõistlik siin kuvada, mitte ainult Eesti rallitulemusi, vaid ka näiteks Soome või Poola rallitulemusi. Samuti peaks selline veebirakendus suutma otsida tulemusi, kas võistleja, etapi või hoopis võistluse järgi. Olenemata sellest, kas tegemist on sportlase fänniga või lihtsalt spordihuvilisega, soovitakse igal juhul näha rallitulemusi kindla etapi või sõitja kohta võimalikult lihtsa vaevaga. Samuti võiks olla võimalus sportlase andmeid võrrelda, näiteks rallietappide läbimise keskmiseid kiiruseid. Kuna rallis sõidetakse ühte rada läbi mitmel etapil, siis oleks hea võrrelda kuidas sõitja aeg muutub samal rajal, aga erinevatel etappidel.

2.1 Olemasolevad lahendused

Veebilehed, mis tagastavad rallitulemusi on juba tehtud. Antud tööga tehti sarnane lahendus olemasolevate veebilehtedega, aga lisati lisaväärtusi, mis võimaldavad rallitulemuste otsijal kergemini leida rallide tulemusi. Edaspidi tuuakse välja olemasolevate lahenduste positiivsed ning negatiivsed aspektid.

2.1.1 Eesti autospordi liidu leht

Eesti autospordi liidu veebirakendus annab küll välja erinevaid eesti rallitulemusi, aga seal pole väiksemaid rallisid ning seal on võimalik näha ainult 2020. aasta rallitulemusi.

Kuigi seal on tehtud lahendus eelmiste aastate tulemuste otsimistega, siis antud veebileht on vigane ning eelmiste aastate tulemuste otsimisel tuleb veateade. Tulemused on seal veebilehel hästi struktureeritud tabelitesse. Tulemuste veebilehte ning õiget linki rallitulemuste jaoks on keeruline leida. Tulemuste lehekülge oleks süsteemil lugeda raske, kuna seal ei ole kasutatud ära veebilehe arendamise häid tavasid.

2.1.2 Rally.ee

Rally.ee veebilehel on ka tehtud võimalus kuvada erinevaid rallisid – nii Eesti kui ka välismaa rallisid. Suurema osa rallide lingid on tühjad ning kuvatakse ainult toimumise informatsiooni, aga mitte tulemusi. Seal on andmed halvemini struktureeritud ja visualiseeritud kui Eesti autospordi liidu lehel. Nagu ka Eesti autospordi liidu veebilehel ei ole siin ka ära kasutatud häid veebilehe arendamise tavasid.

2.1.3 Spordiblogid

Spordiblogidesse pannakse jooksvalt ralliga tulemusi ning võistlusega soetud teateid. Blogidest on sõitja tulemuse otsimine keeruline nii süsteemile kui ka otsijale. Blogides kajastatakse peamiselt suuremaid rallisid.

2.2 Loodava rakenduse aspektid

Lõputöö jooksul loodava rakenduse eesmärgiks oli välja arendada funktsionaalsused, millega leiab lihtsasti rallitulemused.

Rallitulemuste lihtsamaks leidmiseks tehti 3 otsingu filtrit – ralli sõitja või kaardilugeja järgi, ralli nime järgi ning etapi nime järgi. Rakendus ei ole võimeline kuvama mitte ainult Eesti rallitulemusi, vaid erinevate riikide rallitulemusi. Andmete sisestamiseks loodi vaated ning funktsionaalsused, et neid käsitsi sisestada. Rakendusele tehti kaks erinevat automaatset rallitulemuste sisestamise võimalust – olemasolevtest veebilehtedelt otsimine ja rakenduse integratsiooni liidese (API) kaudu, kus teised rakendused saadavad uusi tulemusi kasutades RESTful API-t. RESTful API turvamiseks kasutatakse integratsiooni võtit, mis teeb kindlaks, kust rakendusest tulid tulemused ning määrab ära, mis rakendused tohivad saata tulemusi. Automaatne veebilehelt tulemuste otsimine käib läbi ühelt eesti rallitulemuste veebilehelt, kus oleks andmeid leida kõige lihtsam. Antud lahendus on algse andmehulga saamiseks ning autor leiab, et tulevikus peaksid kõik uued andmed tulema kas läbi API või tuleb neid manuaalset sisestada. Tulemuste võrdlemiseks

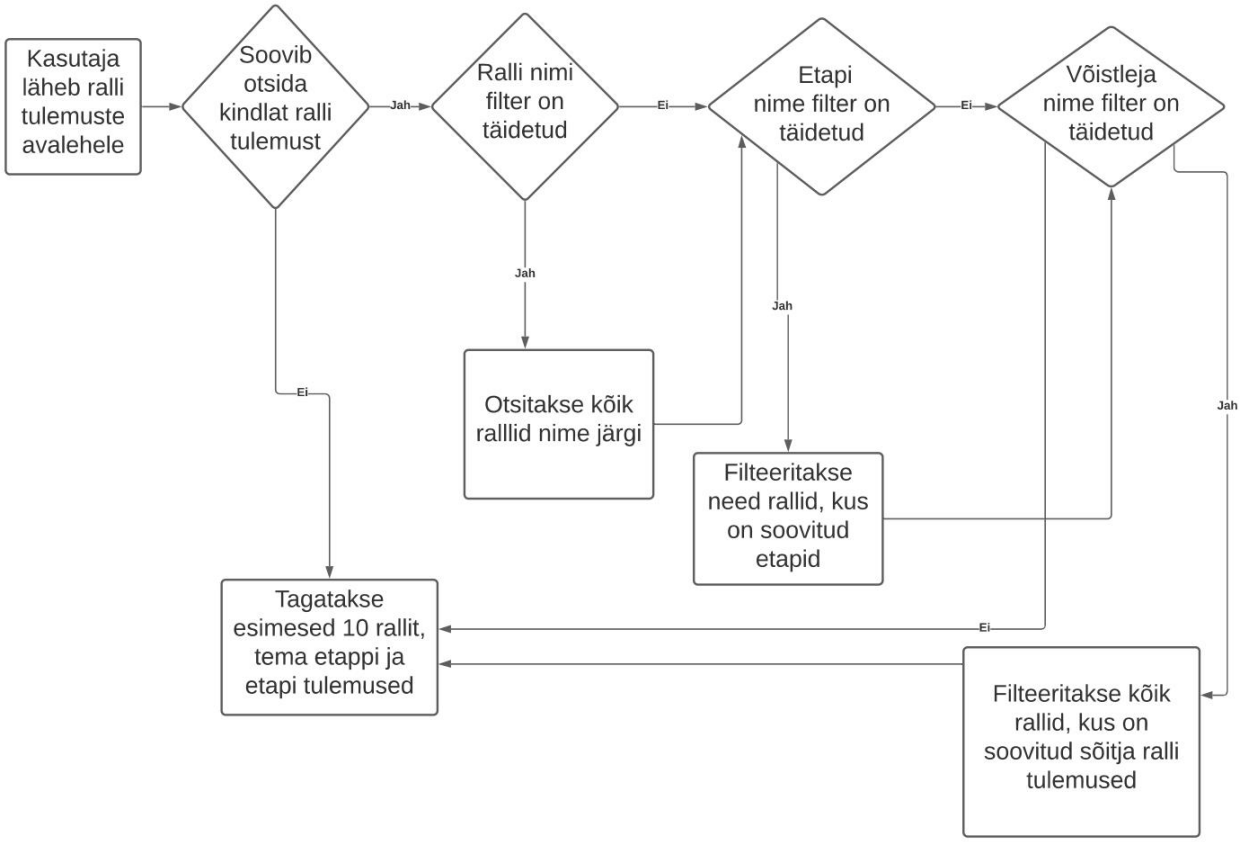
genereeritakse graafikuid. Kuna üldjuhul sõidetakse ühte ja sama rada ühe ralli jooksul mitu korda, siis on võimalik võrrelda kui palju muutus sõitja keskmine kiirus erinevatel läbimistel. Samuti on võimalik võrrelda sama etapi erinevate sõitjate keskmised kiirused.

Välja toodud rakenduse aspektid võimaldavad rakenduse kasutamist mitmes erinevas riigis, importida uusi rallitulemusi automaatselt kui ka manuaalselt, kuvada rallitulemusi ning on võimalus leida neid kiirelt ja lihtsalt.

3 Rakenduse analüüs

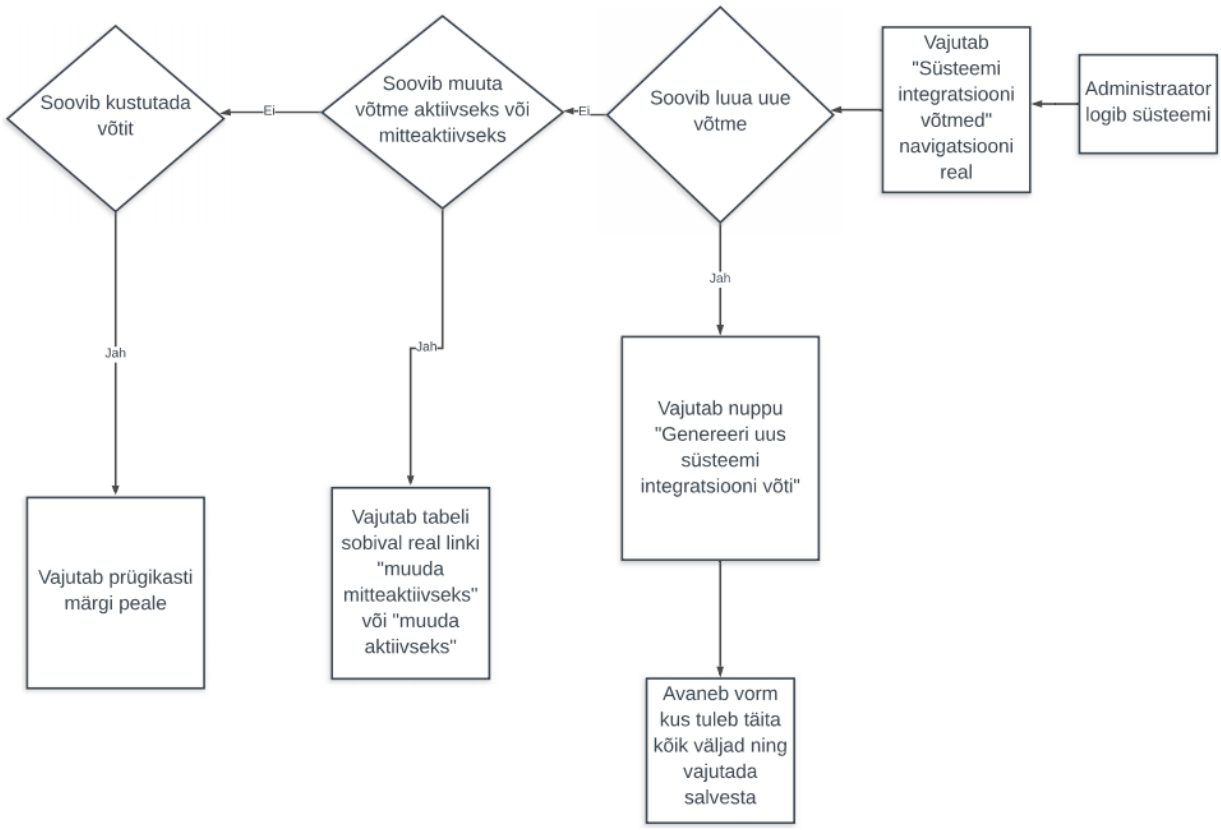
3.1 Rakenduse nõuded

Rakenduse nõuded on kirjeldatud plokkskeemidena. Plokkskeeme koostati kasutades *Lucidchart* diagrammide tarkvara. *Lucidchart* diagrammide koostamise tarkvaraga saab lihtsalt ja kiirelt koostada väiksemaid skeeme. Plokkskeemid on jaotatud kolmeks erinevaks tükiks – rallitulemuste leidmine, rallitulemuste salvestamine ning süsteemi integratsiooni (*API*) võtme genereerimine. Süsteemi integratsiooni liidese ehitamine jäeti antud diplomitöö skoobist välja, kuna töö tegemise aeg on piiratud ning autor ei oleks jõudnud seda töö tähtajaks valmis.



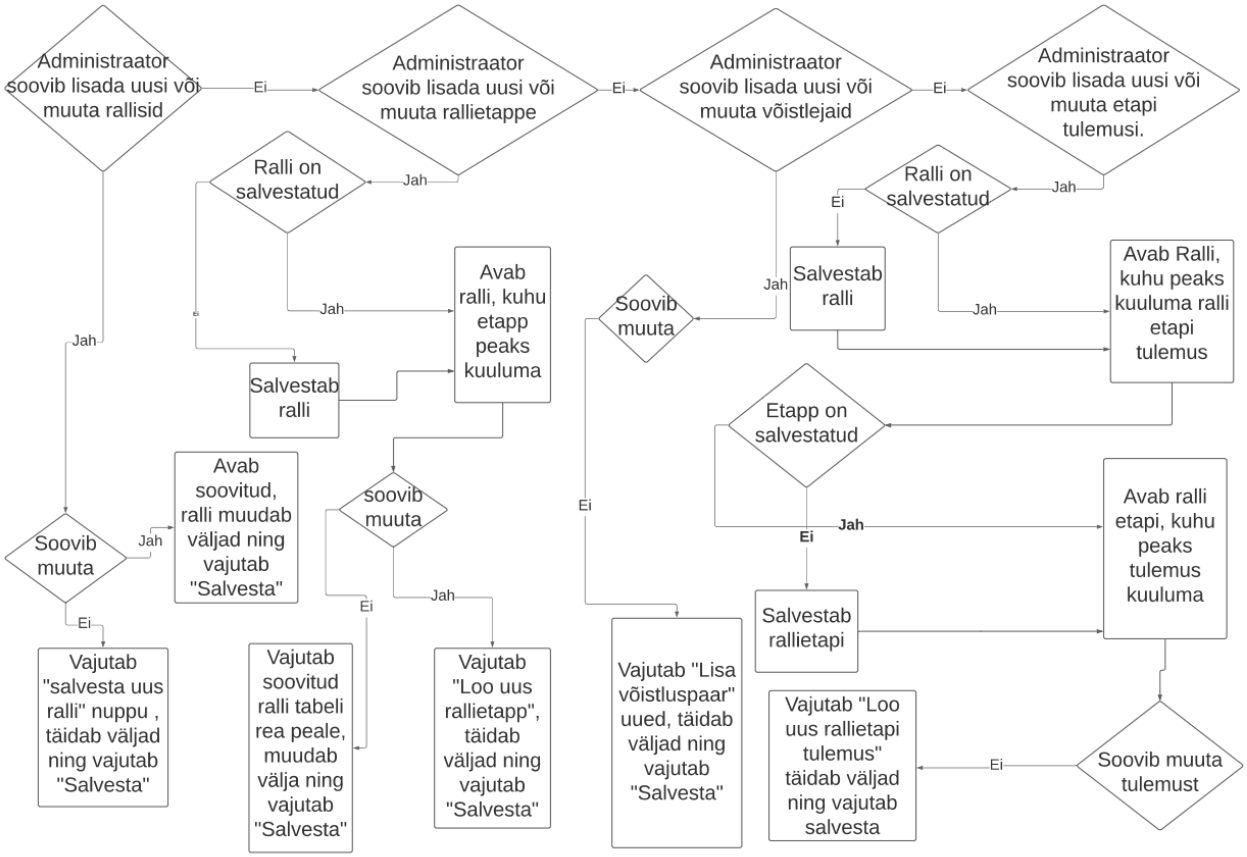
Joonis 1. Rallitulemuste otsimine.

Joonisel 1 oleval plokkskeemil on näha, et rallitulemuste otsimise funktsionaalsuseks arendati rakendusele kolm erinevat otsimise filtrit – võistleja nime, etapi nime ning ralli nime filter. Võistleja nime filter võimaldab otsida nii rallisõitja kui ka kaardilugeja järgi. Etapi ja ralli filter võimaldavad otsida nende nimede järgi.



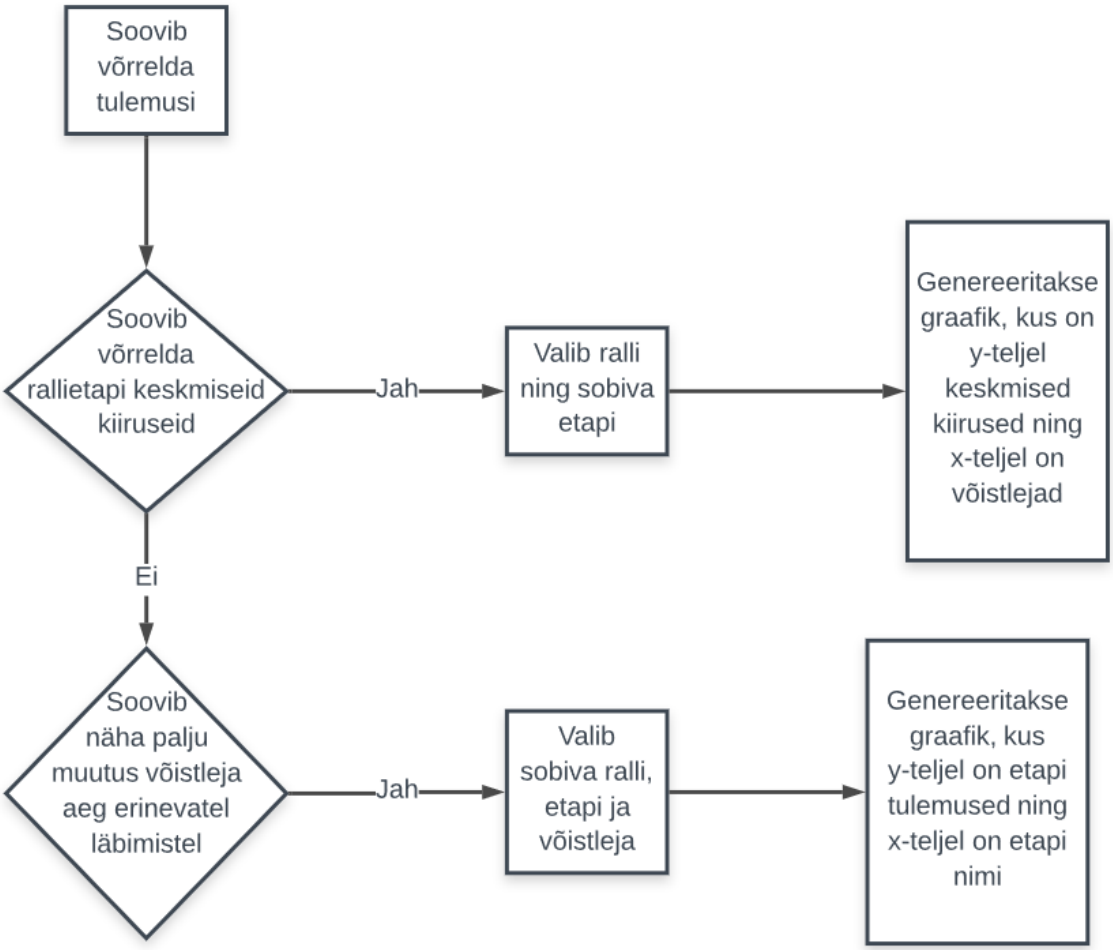
Joonis 2 Süsteemi integratsiooni võtme genereerimine ning aktiivseks või mitteaktiivseks muutmine.

Joonisel 2 on välja toodud ning ära defineeritud, milline näeb välja süsteemi integratsiooni võtme genereerimine veebirakenduses. Antud võtit läheb vaja selleks, et teised rakendused saaksid saata rallitulemusi antud töö käigus valmis arendatud rakendusele ning selleks, et rakendus teaks, kelle poolt tulevad uued rallitulemused.



Joonis 3. Rallitulemuste lisamine.

Joonisel 3 on välja toodud 4 erinevat tüüpi objekte, mida võimaldab süsteem lisada – uued rallid, uued etapid, mis on seotud ralliga, uued võistlejad ning võistleja tulemused rallietappidel.



Joonis 4. Rallitulemuste võrdlemine.

Jooniselt 4 on näha, et rallitulemusi on võimalik kahte moodi võrrelda. Üheks võimaluseks on võtta üks rallietapp ning võrrelda erinevate sõitjate keskmist kiirust antud etapil. Kuna ühte rada sõidetakse ühel rallil mitmel erineval etapil, siis teises võrdlemise viisis saab võrrelda ühe sõitja erinevate etappide läbimist, kus kasutakse sama rada.

Üheks rakenduse nõudeks on andmete otsimine teistest süsteemidest, aga selle lahendamist kirjeldatakse peatükis 3.5.3.

Kasutades plokkskeeme on ära kirjeldatud millised nõuded rakendusel on ning kuidas käib rakenduses rallitulemuste otsimine, lisamine ning võrdlemine. Lisaks sellele on veel lisatud süsteemi integratsiooni võtme genereerimise loogika.

3.2 Rakenduse lahenduse arhitektuuri kirjeldus

Diplomitöö käigus arendati valmis veebirakendus, kus on võimalik näha rallivõistluste tulemusi ning neid võrrelda.

Rakendus peab olema suuteline nii andmeid välja kuvada veebilehtedena, rallitulemusi edastada kui ka neid vastu võtta. Autor leidis, et sellist rakendust saab arendada kahel viisil – kas arendada dünaamilise veebilehe rakendus või arendada kaks rakendust, kus üks rakendus on veebiteenus, mis tagastab andmeid ning teine on SPA rakendus, mis pärib andmeid ning kuvab neid.

Dünaamilise veebilehe rakenduse jõudlus on halvem võrreldes SPA rakendusega, kuna iga kord kui vajutatakse nupule, tagastatakse serveri poolt terve veebilehe HTML kood, aga SPA puhul laetakse uuesti ainult need HTML komponendid, mis vajavad uuendamist. SPA lahenduse puhul on lihtsam pärida andmeid erinevatest API-dest, kuid dünaamilise veebliehe rakendusega on see raskem ja tülikam. Kahe rakenduse puhul on võimalik mõlemat rakendust eraldi arendada, mis teeb arendusprotsessi paindlikumaks ning lihtsamaks. Samuti on võimalik viia sisse muudatusi ühte rakendusse nii, et teist ei peaks testima, näiteks muutes SPA rakenduse visuaalset väljanägemist, ei pea veebiteenust muutma ega testima.

Rakenduse lahenduse arhitektuuriks valiti ehitada kaks rakendust, millest üks on SPA ning teine on veebiteenuse rakendus, sest see on kiirem ning seda on lihtsam arendada võrreldes dünaamilise veebilehe rakendusega [1].

3.3 Andmebaasisüsteemi valik

Rallitulemuste kuvamiseks on vaja andmeid salvestada. Tulemuste salvestamiseks on mitmeid erinevad võimalusi. Üheks võimaluseks oleks salvestada kõik andmed rakenduse mällu, aga iga kord kui rakendust uuesti käivitatakse, läheksid vanad andmed kaotsi [2]. Andmeid on võimalik veel salvestada, kas failidesse või andmebaasi. Kuigi andmete salvestamine failidesse võib olla mõnes olukorras kiirem, annab andmebaasi salvestamine

andmetele parema ülevaate ning on võimalik teha paremini loetavaid päringud. Andmebaasi salvestamise jaoks on olemas raamistikud, mis võimaldavad lihtsa vaevaga salvestada andmeid andmebaasi. Samuti on andmebaasisüsteemidel palju tööriistu ja haldamisvõimalusi, näiteks nii varukoopia tegemine kui ka varukoopia peale panemine. Andmete salvestamiseks valiti andmebaas, kuna see on eelnevalt nimetatud põhjustel kõige mõistlikum [3].

3.3.1 Andmebaasi mudeli valik

Tänapäeva rakendustes on kasutusel kas relatsiooniline, graaf või dokumendi andmebaasi mudel. Autor on tegelenud nii relatsiooniliste andmebaasidega kui ka graafandmebaasidega. Allpool tuuakse välja graafandmebaasi ja relatsioonilise andmebaasi erinevused ning eelised.

Graafandmebaas on andmebaasi mudel, mis koosneb graafide andmestruktuuridest, kus olemid ja tema väljad on tipud ning tippude vahelised seosed on servad.

Relatsioonilise andmebaasi mudeli puhul on sarnased olemid jaotatud tabelitesse ning tabelite vahel on seosed. Tabeli read on olemid ning tulbad olemi väljad.

Antud rakenduse puhul kasutatakse ära relatsioonilist andmebaasi, kuna relatsioonilisest andmebaasist on kergem aru saada. Graafandmebaasil on hea kiirus analüüsides suurel hulgal andmeid, aga kuna antud rakenduse puhul peamiselt kuvatakse tulemusi, siis graafandmebaasi eelised ei ole vajalikud ning seetõttu jäi valikuks relatsiooniline andmebaas [4].

3.3.2 Andmebaasihalduri valik

Andmebaasihalduritena vaadeldakse kolme populaarsemat *PostgreSQL*, *SQL Server* ning *Oracle*. Kuigi antud andmebaasid on sarnased ning kõik kasutavad SQL päringukeelt, siis neil on eelised olenevalt projektist [5].

Oracle on relatsiooniline andmebaasihaldur. CRUD operatsioonide jaoks kasutab see SQL päringukeelt. Protseduuride jaoks on andmebaasi sisse ehitatud PL/SQL programmeerimiskeel. *Oracle* kasutamiseks tuleb osta litsents. *Oracle* andmebaasihalduril on mitmeid erinevaid litsentse ning need erinevad aspektide poolest, mida andmebaasihaldur suudab pakkuda [6], [7].

PostgreSQL on relatsiooniline andmebaasihaldussüsteem, mis kasutab pärimiseks SQL päringukeelt ning erinevate protseduuride jaoks on sellel PL/pgSQL programmeerimiskeel. Antud andmebaas on avatud lähtekoodiga, kaitstud *PostgreSQL* litsentsiga ning tasuta kasutamiseks kõigile [8].

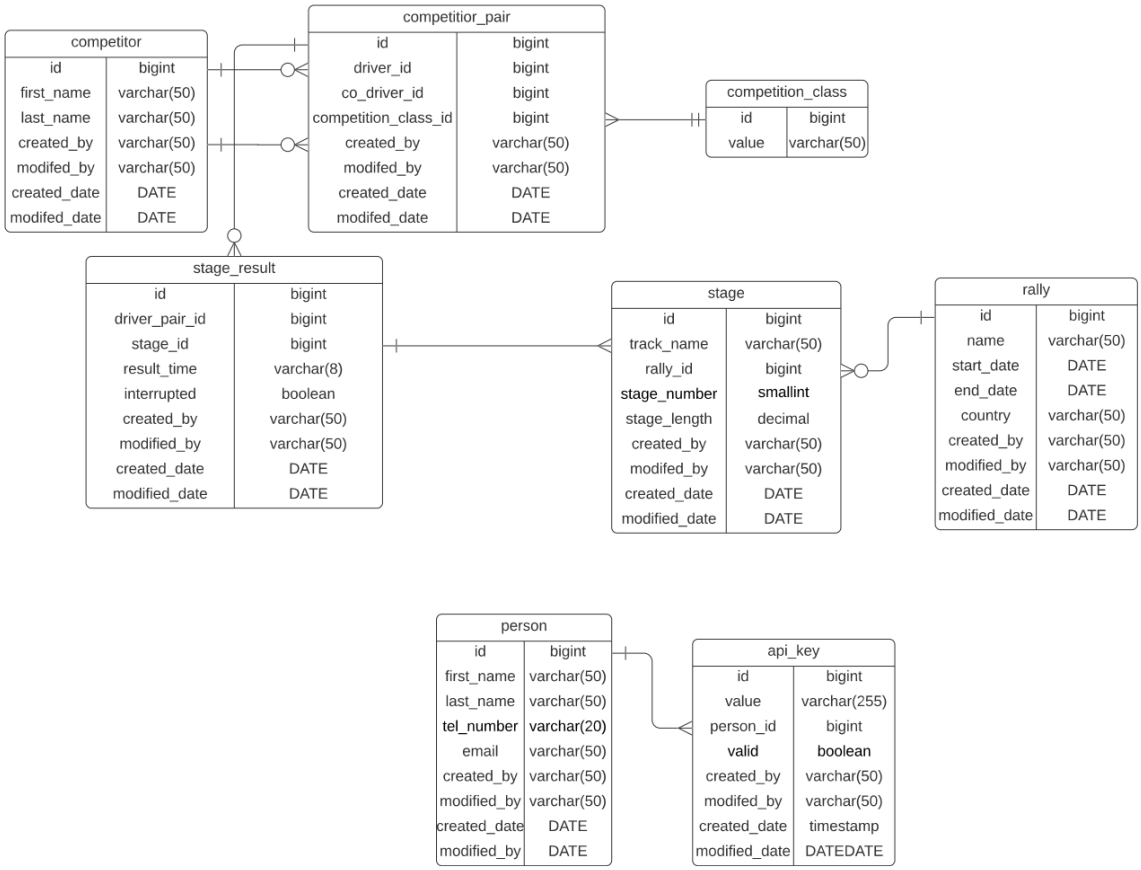
SQL Server on samuti relatsiooniline andmebaasihaldussüsteem. SQL Server kasutab päringute jaoks SQL baseeruvat päringu keelt Transact-SQL. Erinevalt PostgreSQL andmebaasist on *SQL Server* tasuline ehk selle kasutamiseks tuleb osta litsents [8].

Lõputöö lahenduse jaoks võeti kasutusele *PostgreSQL*, kuna töö autor on kogu oma töökogemuse ajal kasutanud relatsioonilist andmebaasi *PostgreSQL* ning erinevalt teistest on see tasuta. Andmeid ei kogune nii palju, et tulevikus võiks olla probleemi andmebaasi jõudlusega.

3.3.3 Andmebaasi skeem

Rallitulemuste salvestamiseks relatsioonilisse andmebaasi on vaja skeemi, mille alusel saab tulemusi andmebaasi salvestada. Rakenduste nõuete järgi on koostatud andmebaasiskeem. Rakenduses kasutakse administraatori kasutajat küll, aga seda kasutajat ei salvestada andmebaasi, kuna rakenduses on mõeldud, et on ainult üks kasutaja. Kasutajanimi ja parool konfigureeritakse rakenduse käivitamisel.

Skeemi koostamiseks kasutati *Lucidchart* diagrammide tarkvara.



Joonis 5. Andmebaasi skeem.

3.4 Veebiteenuse tehnoloogia valik

Veebiteenuste arendamise teevad lihtsamaks erinevad tehnoloogiad ja raamistikud. Kõik tehnoloogiad valiti valimisse nende kasutavatuse, funktsionaalsuse ning autori kogemuse järgi.

3.4.1 Veebiteenuse programmeerimiskeele valik

Veebiteenuste arendamiseks on palju erinevaid raamistike, aga neid raamistikke saab kasutada ainult ühe või mõne kindla programmeerimiskeelega. Töö kirjutamise hetkel on diplomitöö kirjutajal kogemust ainult Java, C# ning JavaScript programmeerimiskeeltega veebiteenuse arendamisel.

Java – platvormist sõltumatu, objektorienteeritud programmeerimiskeel, mis loodi Sun Microsystemsi poolt, aga nüüd arendab keelt edasi Oracle. Java keelt saab kasutada nii erinevate töölauarakenduste, Android mobiilirakenduste kui ka serverrakenduste arendamiseks [9].

C# – objektorienteeritud keel, mis on arendatud Microsofti poolt. Kasutades erinevaid raamistikke on võimalik C#-is arendada mobiilirakendusi, Windowsi töölauarakendusi kui ka serverirakendusi [10].

JavaScript – interpeeritud objektorienteeritud skriptimiskeel, kus saab ära kasutada nii imperatiivset, objektorienteeritut kui ka funktsionaalset programmeerimise paradigmat [11]. JavaScripti kasutati algselt veebilehtedele funktsionaalsuse lisamiseks, aga erinevate raamistike arenguga on tänapäeval võimalik kasutada seda ka serverrakenduste, veebirakenduste või mobiilirakenduste arendamiseks [12].

Arvestades, et diplomitöö tegemine on piiratud ajaga, tuleks valida keel, mis suudaks veebiteenust võimalikult kiiresti ja lihtsalt arendada, oleks kaasaegne ning autoril oleks selle keelega kogemust. JavaScripti keelt oli autor kasutanud ainult ühes kooliprojektis ehk selle kogemus oli minimaalne ning selle kasutamine oleks nõudnud raamistiku tundma õppimist. Seega sobivate keelte valimist eemaldati JavaScript, kuna autor oleks pidanud hakkama õppima uusi raamistikke ning see oleks olnud aeganõudev.

Autor valis kõige sobivamaks keeleks Java, kuna töö kirjutaja on töötanud üle 2 aasta arendajana, kus ta kasutab igapäevaselt Java programmeerimiskeelt. Autor on oma töökogemuse jooksul kasutanud erinevaid raamistikke ning tööriistu, mis vähendavad vajadust uusi raamistikke õppida. Javas on kasutusel projekti kokku ehitamise tööriistad, näiteks *Gradle* või *Maven*. Nende ülesanne Javas on kokku ehitada ette antud struktuuri järgi rakendus, käivitada ning lisada projekti välised teegid, mida on arendamiseks vaja. Nii *Gradle* kui ka *Maven* tööriistad koosnevad nii projekti kokku ehitamiste kui ka käivitamise skriptidest. Neid skripte on võimalik juurde teha [13]. C#-is on ka olemas tööriistad, mis võimaldavad rakenduse arendust lihtsamaks teha. C#-is on kasutusel *NuGet*, mis lisab välised teeke projekti ning erinevalt *Gradle* või *Maven*-ist saab sealt otsida teeke läbi kasutajaliidese [14]. Projekti ehitamiseks saab kasutada *MSBuild* programmi [15].

Antud töö jaoks valiti projekti ehitamise tööriistaks *Gradle*, kuna see on uuem ja paindlikum võrreldes *Maven*-iga ning autoril on rohkem kogemust *Gradle* tööriistaga kui *Maven* tööriistaga.

3.4.2 Veebiteenuse raamistiku valik

Kõik Java rakendused, mis on veebiteenused, mis tagastavad andmeid või on ehitatud dünaamilise veebilehe rakendusena ehk tagastavad veebilehti, kasutavad *Jakarta EE* teeki. *Jakarta EE* teek teeb võimalikuks läbi URL-i pärida serverrakenduselt andmeid. Veebiteenust arendada kasutades *Jakarta EE* teeki, on tülikas ja keeruline, aga tänaseks on Javas mitmeid raamistikke, mis kasutavad *Jakarta EE* funktsionaalsusi ning teevad serverrakendus arenduse lihtsamaks nagu näiteks *Spring* raamistik või *Micronaut* raamistik.

Spring raamistik loodi selleks, et teha arendajatele Java rakenduste arendamise võimalikult kiireks ning lihtsaks. Algselt *Spring* raamistik pakkus lihtsasti kasutatavat DI, IOC konteinerit ning AOP-i. *Spring* raamistikku kuulvad ka mitmed väiksemad moodulid, mis lihtsustavad erinevate probleemide lahendamist. Näiteks *Spring Data Access* moodul võimaldab Java arendajal võimalikult lihtsasti luua ühendus relatsioonilse andmebaasiga ning teha sinna erinevaid päringuid. Autor on kasutanud *Spring* raamistikku kõigis tööalastes projektides, kus on kasutusel Java programmeerimiskeel ehk ta tunneb end selle raamistikuga töötades mugavalt [16].

Micronaut raamistik on sarnaneb *Spring* raamistikule, aga erinevalt *Spring* raamistikust, on seda raamistiku lihtsam kasutada kui on vaja arendada pilverakendusi. Antud raamistikuga autoril kogemus puudub [17].

Veebiteenuse rakenduse raamistikuks valiti *Spring*, kuna autoril on suur kogemus antud raamistikuga ja *Spring* raamistik on olnud kasutusel Java projektides pikka aega ehk probleemide korral leiab lihtsasti *Google*-st vastused oma küsimustele.

3.4.3 Veebiteenuse rakenduse teekide valik

Spring raamistiku kuulub mitmeid erinevaid teeke ning selleks, et nende teekide haldamist *Spring* raamistiku rakenduses lihtsamaks teha, on valmis arendatud *Spring Boot* Raamistik. *Spring Boot* raamistik hõlmab kõiki *Spring* raamsitiku mooduleid ehk teeke ning projekti lisades annab *Spring Boot* just need *Spring* raamisitku moodulite versioonid, mis töötavad hästi omavahel.

Spring Boot on mikro raamistik, mille mõte on, lisada erinevaid *Spring* raamistiku mooduleid neid nii, et nad hästi omavahel töötaksid [18]. Allpool on välja toodud nii kõik

Spring raamistiku kuuluvad moodulid kui ka teised vajalikud teegid ning mis on nende ülesanne projektis.

Spring Data JPA moodul võimaldab väga lihtsasti nii luua ühendust relatsioonilise andmebaasiga, pärida, uuendada, lisada kui ka kustutada andmeid andmebaasist. Moodul võimaldab hästi pärida andmeid lehekülgede kaupa. Samuti võimaldab lihtsalt teha transaktsioone andmebaasi päringutel. Dünaamiliste päringute tegemiseks saab kasutada, kas *QueryDSL* teeki või *Spring Data JPA Specification API*-t [19], [20].

Spring Web moodul annab ette baasklassid ning automaatselt genereeritud konfiguratsiooni veebirakenduse arendamiseks. Sinna kuulub veel sisseehitatud rakendusserver, kontrollorite defineerimiseks annotatsioonid, mis kasutavad ära *Servlet API*-t. *Spring Web* moodulisse kuulub ka *Jackson* teek, mis aitab viia objektid JSON kujule ning vastupidi [21].

Liquidbase teeki kasutatakse projektis andmebaasi skriptide haldamiseks. *Liquidbase* võimaldab mugavalt lisada juurde andmebaasi skripte kas tabeli lisamiseks või mõne muu otsetarbe jaoks. *Liquidbase* on hea kasutada ka CI ja CD protsesside jaoks. Näiteks kui rakendus viiakse testkeskkonda, siis ehitakse konfiguratsioonile vastav andmebaas ehk andmebaasis on ainult nii palju funktsionaalsust kui serverrakendus vajab. Raamistik aitab rallitulemuste rakendusel andmebaasi inkrementaalselt arendada, mis lihtsustab terve rakenduse arendust [22].

Spring HATEOAS on *Springi* moodul, mis võimaldab lisada RESTful-i päringutele lingid, millega on võimalik mõnda täpsemat objekti otsida. Antud teek võimaldab lihtsasti juurde arendada klientrakendust ning igalt veebiteenuse päringult saab linke kas sama päringu kohta või mõne objekti sees oleva objekti pärimise lingi. Kui pärida veebiteenuselt kõik ühe ralli etapid, siis seal on link, kuidas leida ühte etapi või mõne etapi sõitja kohta infot [23].

Mapstructi kasutakse rakenduse arendamisel objektide ümber teisendamisel. Antud teek genereerib valmis koodi, mis aitab andmebaasi objekti teisendada rakenduse objektiks või vastupidi. Samuti saab teegiga teisendada rakenduse objekti ümber DTO-ks. Teek on oluline, kuna aitab kiiremini objektide teisendamise funktsionaalsust teha [24].

Lombok teeki kasutakse lihtsate ja sarnaste Java klassi meetodite genereerimiseks, näiteks objekti välja saamiseks suudab *Lombok* genereerida vastavaid meetodeid objekti klassi. *Lomboki* teek aitab arendajal kiiresti lisada tavapäraseid meetodeid, mis peaks olema igal Java klassil [25].

Paremaks ja turvalisemaks andmebaaside päringute tegemiseks, kasutakse ära *Criteria API*-t, mis võimaldab teha turvaliselt andmebaasi päringute kitsendusi dünaamiliselt ja lihtsalt [26]. *Criteria API* kasutamiseks koos *Spring Data JPA* teegiga on *Spring Data JPA* moodulis loodud *Specifications* klass, mis võimaldab dünaamiliselt luua andmebaasi päringu kitsendusi ning lisada need andmebaasi päringutele [20]. Andmebaasi kitsenduse päringutel tuleb kasutada andmebaasi tabeli objekti ehk olemi väljade nimesid. Välja nimed kirjutatakse sõnedega ning selleks, et olla kindel, et sõnes pole väikest kirjaviga, kasutatakse projektis *JPA Static Metamodel Generator*-it, mis genereerib igale olemi väljale konstanti, mis on väärtustatud olemi välja nimega [27].

Kõik teegid ning raamistikud, mis eelnevalt välja toodud olid plaanis kasutada, aga arenduse ajal võib ette tulla ka olukord, kus peab veel mõne teegi lisama. Kõik teegid, mis pole siin välja toodud, aga on arendusel kasutusel tuuakse välja arenduse peatükis.

3.4.4 Veebiteenuse rakendusserveri valik

Server rakenduse arendamiseks on vaja anda Java rakendusele rakendusserver. Rakendusserver lihtsustab serverrakenduse arendamist ehk arendaja ei pea arendama serverrakenduse funktsionaalsusi. Arendaja ülesandeks on äriloogika valmis arendamine ning muu vajalik, mis serverrakendusel peaks olema tehakse rakendusserveri poolt ära. Rakendusserverit võib võtta ka kui rakenduse raamistiku ehk kõik, mis on igal serverrakendusel sama tehakse ära raamistiku poolt ning kõik, mis on antud rakenduse spetsiifiline osa tehakse arendaja poolt valmis [28].

Java rakendusserveri valmisse võetakse kaks rakendusserverit – *Jetty* ja *Tomcat*. Valimisse võeti just need rakendusserverid, kuna nendega on autoril kogemust.

Tomcat on avatud lähtekoodiga rakendusserver, mis kasutab *Apache 2* litsentsi ning see arendati Apache Software Foundation poolt. *Tomcat* rakendusserver kasutab *Jakarta EE* spekifikatsiooni. *Tomcat* kasutab viimased JSP ja *Servlet API* standardeid. *Tomcat* on enim tuntud ja kasutakse suurema osa *Spring*-i rakendustes [28], [29].

Jetty on ka avatudlähtekoodiga rakendusserver, mis arendati Eclipse Foundation poolt. *Jetty* kasutab nii *Jakarta EE* spekifikatsiooni, uusimat *Servlet API*-t kui ka JSP nagu ka *Tomcat* rakendusserver. *Jetty* kasutab nii *Apache 2* litsentsi kui ka *Eclipse* litsentsi. *Jetty* võtab vähem mälu kui *Tomcat* [28], [29].

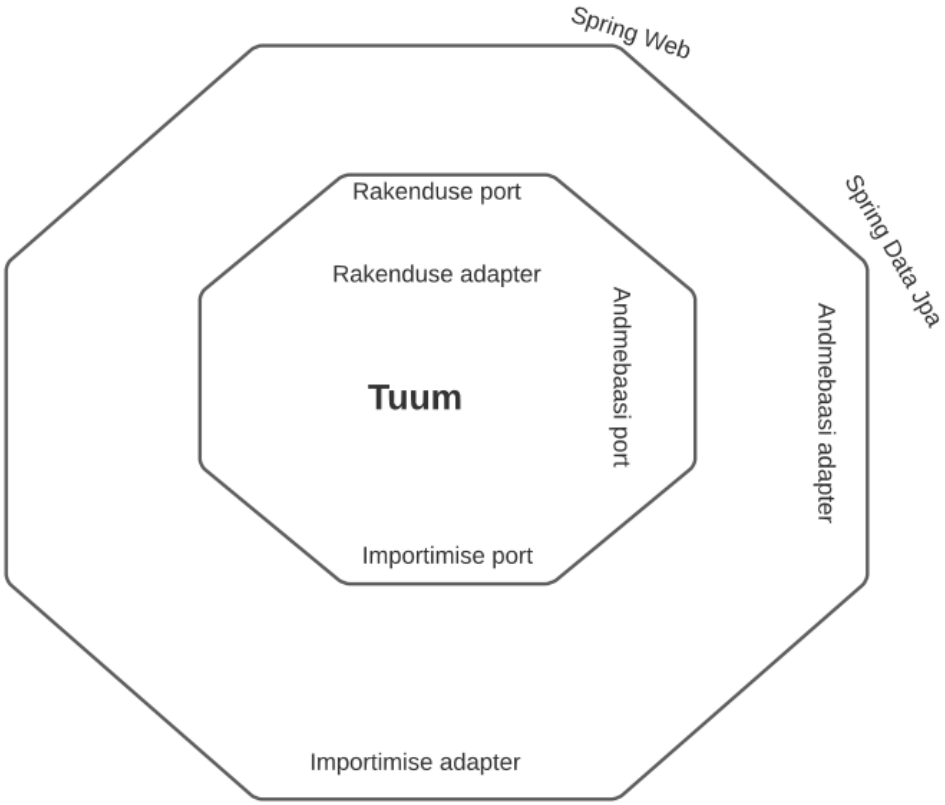
Mõlemad rakenduseserverid on väga sarnased ning täidavad sama eesmärki. *Tomcat*-i kasutatakse *Spring*-i rakendustes vaikimisi, aga on võimalik ka minna üle *Jetty* rakendusserverile. *Tomcat* on võrreldes *Jetty*-iga rohkem tuntud. *Jetty* rakendusserver võtab vähem mälu võrreldes *Tomcat*-iga. Autori enamus kogemust tuleb töötades koos *Tomcat* rakendusserveriga ning seetõttu valiti rakendusserveriks *Tomcat*.

3.4.5 Veebiteenuse arhitektuur

Veebiteenuse arendamise jaoks tuli paika panna rakenduse arhitektuur, mis võimaldab rakendust lihtsasti edasi arendada. Selle rakenduse puhul kasutakse portide ning adapterite arhitektuurimustrit. Selle peamine eesmärk on lüüa lahku rakenduse komponendid.

Rakenduse tuum ehk pordid on arhitektuuriline kiht, kus defineeritakse üldine rakenduse loogika ning millised teenused on rakendusel kasutusel. Portides määratakse enamasti spekifikatsioon ja rakenduse mudelid. Väljaspool tuuma kihti asuvad detailsemad kihid ehk adapterid, kus implementeeritakse portides defineeritud spekifikatsioonid. Siin on oluline see, et portide kihis ei tohi kasutada raamistike ega teeke, välja arvatud koodi genereerimise teegid, mis teevad arenduse kiiremaks. Siin jagunevad pordid kaheks – primaarsed pordid ning sekundaarsed pordid. Rallitulemuste veebiteenuse rakenduses on primaarsed pordid rakenduse pordid ning sekundaarsed pordid on andmebaasi pordid ja tulemuste importimise pordid. Rakenduse porte saab ka nimetada ka rakenduse äriloogikaks. Antud lähenemise puhul on lihtne testida arendatud komponente. Samuti see võimaldab vahetada välja rakenduse raamistiku ilma vajaduseta kirjutada uuesti rakenduse äriloogikat [30].

Järgmisel lehel on kujutatud skeemi, mis illustreerib veebiteenuse arhitektuuri.



Joonis 6. Veebiteenuse arhitektuur.

Kui tuleks implementeerida ralli otsimine, siis tuleks teha kõigepealt otsimise port ning otsimise pordis kasutatakse andmebaasi porti. Rakendus olla sõltuvuses raamistikest, mida rakenduses kasutatakse. Need on lihtsalt abistavad tegurid, aga raamistiku vahetamine peaks olema võimalikult lihtne. Raamistikud ning teegid pandi skeemil välise kaheksanurga otstesse.

core	Rakenduse tuuma kiht, siin määratletakse kõik pordid, mida iga adapter peab implementeerima. Joonisel on see tuuma kihina. Siia kuuluvad nii sekundaarsed kui ka primaarsed pordid, rakenduse mudelid ning implementeeritud primaarsed pordid ehk äriloogika. Joonisel on "TUUM" kihina.
persistence	Andmebaasi kiht, tehakse andmebaasi päringud, olemid ning objektide teisendamised. Joonisel on andmebaasi adapter kihina.
importer	Rallitulemuste importimise kiht, meetodid, kuidas saada tulemusi teistest süsteemidest ning nende viimine rakenduse objektide kujule. Joonisel on tulemuste importimise adapter kihina.
core.service	Rakenduse primaarsete portide implementatsioon ehk äriloogika. Joonisel kui rakenduse adapterina.

Tabel 1 Veebiteenuste peamiste pakettide selgitus.

3.5 Klientrakenduse tehnoloogia valik

Klientrakendus võib olla nii veebirakendus, mobiilirakendus kui ka töölauarakendus. Töölauarakendus klientrakenduseks väga hästi ei sobiks, kuna selleks on vaja kasutada arvutit, aga tahvelarvutis või telefonis seda kasutada ei saaks. Ka mobiilirakendus ei sobiks hästi, kuna seda ei oleks võimalik avada arvutis. Veebirakendus on kõige mõistlikum valik, kuna seda saab avada tahvelarvutis, telefonis ja arvutis.

3.5.1 Klientrakenduse raamistiku valik

Klientrakenduse kõige kiiremaks ja parimaks lahenduseks on mõistlik kasutada veebirakenduse raamistike. Küll on võimalik ehitada klientrakendust kasutades ainult JavaScript, HTML, CSS programmeerimiskeeli, aga nii on seda keerulisem ja veaohtlikum teha ning see nõuab rohkem aega. Veebirakenduse ehitamiseks on tänapäeval kasutusel kolm populaarsemat JavaScript-i raamistiku – *React*, *Angular* ning *Vue*.

Angular oli esimene JavaScripti veebirakenduse raamistik. See loodi Google-i poolt aastal 2012. See loodi kasutades MVC arendusmustrit. Sinna lisati andmete dünaamiline muutmine ehk kui veebilehel on kujutatud andmed muutuvad, siis veebilehel on automaatselt muudatusi näha. Seal raamistikus on võimalik teha oma komponente, mis näevad välja nagu HTML, mis parandab koodi loetavust ning kvaliteeti ning neid komponente saab kasutada mitmes kohas. Raamistikus on peale MVC arendusmustri kasutusel ka DI [31].

React raamistiku arendati Facebooki poolt. Kui *Angular* kasutas MVC arendusmustrit, et luua erinevaid komponente, siis kasutab *React JSX* tehnoloogiad, kus on HTML ja JavaScripti kood ühes kohas koos, kus kontrollitakse erinevate vaadete loogikat. *React* hoiab mälus virtuaalselt veebilehe vaadet. Veebilehel muudatusi tehes kõigepealt uuendatakse virtuaalsest veebilehe vaadet, leitakse minimaalsed elemendid, mis tuleb muuta ning siis uuendatakse veebilehe vaadet, mida kasutaja näeb. Selline lähenemine võimaldab muudatusi võimalikult kiirelt veebilehel kuvada [32] [33].

Vue.js on neist kolmest raamistikkudest kõige uuem. Selle arendas välja Evan You, kes varasemalt töötas Google-s. *Vue.js* raamistku idee oli luua võimalik väikse suurusega ja kiiresti vaateid laadiv raamistik, mis baseerub *Angular*-il. *Vues* nagu ka *Angular*-is on võimalik teha taaskasutavaid komponente ja neid kasutada ning need näevad välja nagu HTML elemendid [34].

Raamistiku nimi	Arendaja kogemus	Selgitus
Angular	Suur	Töö autor on kasutanud raamistiku nii kooli töödes, hobiprojektides kui ka töö projektides.
React	Väike	Kasutanud hobiprojektides ning tööl paar kuud.
Vue	Pole kokku puutunud	Autor pole kokku puutunud *Vue.js* raamistikuga.

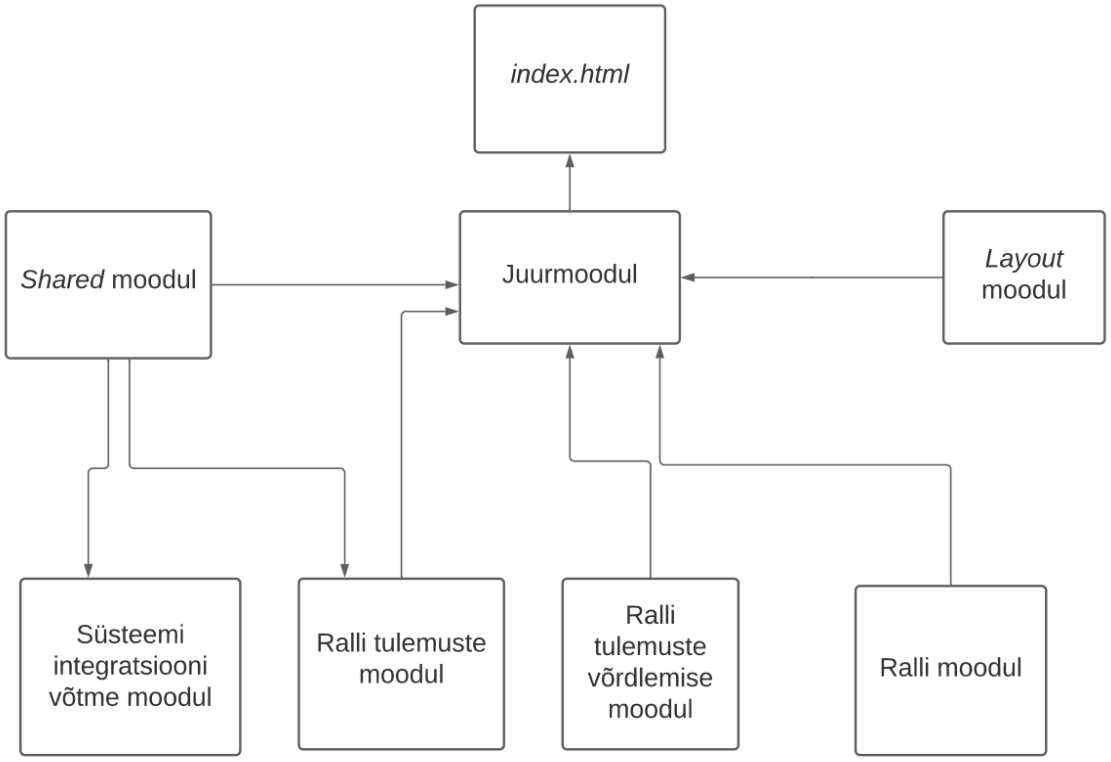
Tabel 2. Võrdlus autori kogemusest JavaScripti raamistikutega.

Diplomitöö autor ei ole varasemalt *Vue.js* raamistikuga tegelenud ning kuna töö tegemise aeg on piiratud, siis *Vue.js* ja *React* ei ole kõige mõistlikumad valikud. Klientrakenduse raamistikuks valis autor *Angular*-i, kuna tal oli sellega kõige rohkem kogemust. Kõik need raamistikud on küll sarnased ning täidavad ühte ülesannet – arendada võimalikult kiiresti mitmele platvormile rakendus. *Angular*-i arendamisel kasutatakse *Angular* 10 versiooni.

Kiiremaiks veebirakenduse väljanägemise arendamiseks kasutatakse *Bootstrap* raamistiku. Raamistik pakub hea väljanägemisega komponente ning võimaldab muuta veebilehed skaleeruvaks [35].

3.5.2 Klientrakenduse arhitektuur

Klientrakenduse arendamisel kasutatakse komponent arhitektuuristiili. Rakendus koosneb mitmest erinevast moodulist, mis imporditakse juurmoodulisse. Iga moodul koosneb ühest või mitmest sarnasest komponendist, mooduli definitsioonist ning komponentide interneti aadressi definitsioonist.. Moodulite seas on kaks erilist moodulit *shared* ning *layout*. *Shared* moodulis on abikomponendid, mis aitavad teistel komponentidel paremini oma funktsionaalsust saavutada ning *layout* moodulis on lehekülje välised elemendid nagu näiteks jalus ja päis [36].



Joonis 7. Klientrakenduse arhitektuur.

Jooniselt 7 on näha milline näeb välja kogu klientrakenduse arhitektuur. Skeemilt on näha, et *shared* moodulit ei impordita ainult juurmoodulisse, vaid ka väiksematesse moodulitesse. Kui ralli moodulil on vaja mõnda abifunktsiooni, mis *shared* moodulil on, siis tuleks tekitada joon ralli mooduli ja *shared* mooduli vahel.

3.5.3 Olemasolevate rallitulemuste leidmine

Rakenduse üks nõutest oli, et rakendusel peaks olema funktsionaalsus rallitulemusi ise otsida. Rakendus peaks iga nädal käima uusi tulemusi küsimas sellisel ajahetkel, kus rakenduse kasutamine on minimaalne. Ralli olemasolu veebiteenuse rakenduse andmebaasis kontrollitakse ralli toimumise kuupäeva ja ralli nime järgi.

Autor välistas rallitulemuste otsimise spordiblogidest, kuna sealt oleks keerulline otsida. Näiteks Postimehe spordiblogi ei lae kõiki tulemusi korraga vaid viimased tulemusi. Kerides blogi allapoole tulevad varasemad rallitulemused, aga süsteemil oleks keeruline

kõiki rallitulemusi otsida. Kõik tulemused on seal kuvatud piltidena ning pildilt tulemuste otsimine oleks liiga keeruline.

Diplomitöö arenduse jaoks kulutatud aeg on piiratud ning seetõttu töö raames arendatav rakendus otsib ainult Eesti rallide tulemusi. Nimelt on kaks veebirakendust, mis annavad välja Eesti rallide tulemusi. Nendeks on rally.ee veebirakendus ning Eesti autospordi liidu veebirakendus.

Eesti autospordi liidu veebirakenduses on andmed jaotaud tabelisse ning on paremini organiseeritud võrreldes rally.ee veebirakendusega. Eesti rallide tulemuste otsimiseks kasutatakse antud töös Eesti autospordi liidu veebirakendust, kuna sealt on võimalik tulemusi kõige lihtsamini leida. Andmed on jaotatud kindla struktuuri järgi tabelisse, kust saab kogu vajaliku info kergesti kätte ning siis saab neid andmebaasi salvestada. Rakendus kasutab „http://autosport.ee/rallyreg/index.php?page=22&“ linki, et otsida rallitulemusi.

3.6 Versioonihaldustarkvara ning versioonihalduskeskkonna valik

Versioonihalduskeskkonna parimaks valikuks võeti valimisse kolm erinevat süsteemi: Bitbucket, GitLab ning Github [37].

Github platvorm loodi aastal 2007 ning 2011. aastal oli see kõige kasutatavam platvorm. Microsoft ostis GitHub-i aastal 2018. Nüüd on seal võimalik luua nii avalike kui ka privaatseid koodihoidlaid (repositooriumeid). Pakub tasuta lisavõimalusi, kui on tudengi litsents. Github teeb rakenduse teekidele turvalisuse kontrolle ning kui projektis on kasutusel mõni teek, mis ei ole väga turvaline, siis Github teavitab sellest. Üheks miinuseks võib tuua, et platvorm toetab ainult *Git* versioonihaldustarkvara [37]. Väiksemate projektide ning tiimide ja ilma eriliste lisaväärtusteta saab kasutada GitHub-i tasuta [38]. Toetab ka *Mercurial* versioonihaldustarkvara [39].

GitLab platvormi on võimalik püsti panna oma serveris ning hoida kõiki projekti koodihoidlaid seal. GitLab on avatud lähtekoodiga. GitLab-il on küll tasuta versioon, aga seal pole palju lisaväärtusi ning võrreldes GitHub-iga ei ole see nii populaarne. Samuti on GitLab-il keeruline kasutajaliides, mida kasutada [37].

Bitbucket platvorm loodi algselt ainult *Mercurial* projektide jaoks. Aastal 2011 hakati toetama *Git* versioonihaldustarkvara koos *Mercurial*-iga ning tänaseks toetakse ainult *Git*-i. Üheks suureks plussiks Bitbucketi juures on see, et ta töötab Atlassian-i teenustega nagu näiteks Jira [37].

Kõik need kolm versioonihalduskeskkonda toetavad *Git* versioonihaldustarkvara, aga kõik nendest ei toeta *Mercurial* tarkvara. Autori kogemus piirdub ka ainult *Git* versioonihaldustarkvaraga ning seetõttu valiti tööks teostavaks versioonihaldustarkvaraks *Git*. Versioonihalduskeskkonnaks valiti GitHub, kuna autor kasutab enda isiklikuks tarbeks ning seda on mugav kasutada. GitHub-il on suurem kasutajaskond ehk kui mingi probleem peaks tekkima GitHub-iga, siis sellele on võimalik kergelt lahendust leida.

Rakenduse lähtekoodi leiab *„https://github.com/raunlo/RallyResultsApp“* lingilt.

3.7 Analüüsi kokkuvõte

Analüüsi käigus leiti kõige kaasaegsemad ning mugavamad tehnoloogiad. Lahendust tehti kahe rakendusega – veebiteenus ja klientrakendus. Veebiteenus arendati kasutades Java programmeerimiskeelt ja *Spring Boot* raamistikku. Veebiteenus disainiti kasutades portide ja adapterite arhitektuuri. Klientrakendus, mis on veebirakendus tehti *Angular* raamistikus. Klientrakendus kuvab andmeid pärides neid veebiteenuselt. Rakendusel on üks kasutaja, kes suudab käsitsi lisada, muuta ja kustutada rallitulemusi ning genereerida võtit, et teised süsteemid saaksid saata uusi tulemusi. Rakenduste arendamisel arvestati, et edasiarendus oleks võimalikult lihtne. Andmete salvestamiseks kasutati relatsioonilist andmebaasi ning *PostgreSQL* andmebaasihaldussüsteemi. Lahenduse lähtekoodi hoitakse GitHub versioonihalduskeskkonnas ning kasutakse *Git* versioonihaldustarkvara.

4 Rakenduste arendus

Käesolevas peatükis seletatakse lahti kuidas rallitulemuste veebirakendus arendati, suuremad komistuskivid ning nende lahendused.

Rakenduse arendamisel arvestati, et tulevikus oleks võimalik kasutada seda rakendust erinevates riikides, aga antud lõputöö raames lähtuti ainult, et seda kasutatakse ainult Eestis. Rakenduse arendus on jaotatud neljaks alampeatükiks – veebiteenuse arendus, esirakenduse arendus, mõlema rakenduse testimine ning edasiarendus.

4.1 Veebiteenuse arendus

Autor tegi ainult need päringuid veebiteenusele, mis täidaksid analüüsis püstitatud rakenduse nõudeid. Kuigi analüüsis toodi välja, et kasutatakse *Spring HATEOAS* raamistiku, mis lisab lingid tagastatavatele andmetele, siis ajalise piirangu tõttu autor ei jõudnud neid realiseerida, aga tulevikus on autoril plaan need siiski lisada.

Veebiteenuses kasutati kolme tüüpi POJO-sid – rakenduse objektid, andmebaasi objektid ning andmeid tagastavad objektid. Kõikidel objektid, millel on järelliide „Dbo“ on andmebaasi objektid ja kõik objektid, millel puudub järelliide on rakenduse objektid ning millel on järelliide „Dto“ on andmeid tagastavad objektid. Samuti on eraldi „Import“ järelliide, mida kasutatakse rallitulemuste importimise objektidel. Näiteks ralli andmete puhul on kolm erinevat objekti *Rally*, *RallyDbo*, *RallyDto*. *RallyDto* on ralli andmeid tagastav objekt ja *RallyDbo* on ralli andmebaasi tabeli objekt ning *Rally* on rakenduse objekt. Erinevate objektide tegemine oli mõistlik, kuna andmebaasis on välju, mida rakenduses ei peaks kasutama. Samamoodi on ka import ja DTO objektides välju, mida ei oleks mõistlik hoida rakenduse objektides. Andmete kirjutamiseks ühest objektist teise, kasutati *Mapstruct* teeki. *Mapstruct* võimaldab lihtsa vaevaga defineerida, kuidas tuleb viia andmed ühest objektist teise ning selle peale genereeritakse koodijupp, mida kasutab rakendus andmete teisendamiseks.

Rakenduse arendamisel kasutati Java 11 versiooni ning *Spring Boot* 2.3.4 versiooni.

4.1.1 Andmebaasi käivitamine

Rakenduse arendamisel kasutati *PostgreSQL* andmebaasisüsteemi ning andmebaasi ülesseadmiseks kasutati *dockerit* ning *docker-compose'i*.

```
version: '3'

services:
  database:
    image: "postgres"
    environment:
      - POSTGRES_DB=rally-results
      - POSTGRES_USER=postgres
      - POSTGRES_PASSWORD=postgres
    ports:
    - 5432:5432
```

Joonis 8. *Docker* konfiguratsioon andmebaasi jaoks.

Jooniselt 8 on näha andmebaasi käivitamise *docker-compose* skripti. Jooniselt on näha, et andmebaasi nimi on „rally-results", andmebaasi administraatori kasutajanimi on „postgres" ning parool „postgres". Andmebaas kasutab 5432 võrguporti. Antud skripti käivitamiseks tuleb luua fail nimega „docker-compose.yml" ning käsureal käivitada käsk „docker-compose up" eeldusel, et *docker* ja *docker-compose* on arvutisse installitud ning käsk käivitatakse samas asukohas, kus on „docker-compose.yml" fail.

4.1.2 Veebiteenuses loodud päringud

Veebirakenduse arendamisel loodi kahte tüüpi päringuid – avalikud päringuid ning turvatud päringud. Avalikeks päringuteks on rallitulemuste otsimine ning rallitulemuste võrdlemise andmed. Turvatud päringuid on sellised, mida saab läbi viia ainult administraatori kasutaja. Kõik turvatud päringud on peamiselt CRUD päringud ehk loomise, pärimise, muutmise ning kustutamise päringud ning neid kasutatakse rallide, rallietappide, võistlejate, etapi tulemuste ning süsteemi integratsiooni võtme objektidega.

4.1.3 Veebiteenuse päringute turvamine

Kuna andmete näitamine ning andmete pärimine on tehtud erinevates rakendustes, siis on mõistlik luua lahendus, kus turvatud päringu puhul antakse kaasa väärtus, mis lubaks päringut teostada. Sellise lahenduse jaoks kasutas autor oma lahenduses *oauth password grant_type* skeemi.

See lahendus koosneb kahest komponendist – autoriseerimisserverist ning ressursiserverist. Autoriseerimisserverit kasutatakse selleks, et valideerida kasutajanimi ja tema parool ning tagastada talle kasutaja turvamärk. Ressursiserver kontrollib antud turvamärgi valiidust, leiab antud kasutaja läbi turvamärgi ning kontrollib, kas antud kasutajal on õigusi selleks, et saada kätte soovitud andmeid.

Autor võttis rakenduses kasutusele *oauth2 springi* mooduli, mis aitab sellist lahendust teostada. Samuti võeti kasutusele *autoncofigure-processor* moodul, mis võimaldab luua objekte rakenduse konfiguratsioonist.

```
POST /oauth/token HTTP/1.1
Content-type: application/x-www-form-urlencoded; charset=UTF-8
Authorization: Basic cmFsbHlfcmVzdWx0czpyYWxseV9zZWNyZXRz


HOST: localhost:8080


grant_type=password
&username=admin
&password=admin
```

Joonis 9. Näidis päring turvamärgi saamiseks.

Joonisel 9 on näha päringut millega saadakse turvamärk. Tegemist on POST tüüpi päringuga ning sellele antakse andmed, mis peaks olema *application/x-www-form-urlencoded* formaadis ning see sisaldab kolme välja – „username“, „password“, „grant_type“. Samuti tuleb kaasa autoriseerimisserveri turvamärk. Päringult saadakse tagasi kasutaja turvamärk koos selle eluajaga ning tüübiga. Kasutaja identifitseerimisel kasutatakse „Bearer“ turvamärgi tüüpi. Järgnevalt salvestatakse kasutaja turvamärk

localstorage-sse ning kui on vaja teha päring veebiteenusele, siis antakse kaasa saadud turvamärk.

Rakenduses on ainult üks kasutaja, kes saab andmeid luua, muuta ja kustutada ning seetõttu on rakenduse kasutaja defineeritud rakenduse konfiguratsioonis. Rakenduse konfiguratsioonis tuleb määrata administraatori kasutaja ja parool. Kui rakendusele antakse ette kasutajanimi ja parool, siis sisselogimise puhul tehakse paroolist räsi ning kontrollitakse räsi järgi kas antud parool on õige. Parooli räsi moodustamiseks kasutatakse *bcrypt* algoritmi.

4.1.4 Olemasolevate rallitulemuste importimine

Üheks keerulisemaks ülesandeks käesoleva diplomitöö veebirakenduse arendamisel oli leida olemasolevad rallitulemused ning need rakenduse andmebaasi importida. Antud ülesanne muutus päris keeruliseks, kuna veebilehed, kust autor tahtis tulemusi importida ei olnud hästi tehtud ning andmete leidmine veebilehelt ostus keerukaks. Keerukas muutus veebilehelt andmete otsimine, kuna HTML elementidel ei ole kasutatud „id“ atribuute. HTML-ist andmete lugemiseks kasutas autor *Jsoup* teeki, mis lihtsustab HTML-i tagide otsimise. Antud teegi API meenutab *Jquery* teeki, mida kasutatakse JavaScripti programmeerimiskeeles, et muuta või leida HTML elemente.

Rallitulemuste importimisel kasutatakse cron-avaldist, millega määratletakse millal käivitatakse rallitulemuste importimine. Selleks avaldiseks on „0 0 2 * * SUN“. Antud avaldis määrab, et rallitulemuste importimist käivitatakse iga nädal pühapäeval kell 2 öösel.

4.1.5 Veebiteenuse konfiguratsioon

Veebiteenuse konfiguratsiooni faili jaoks kasutati YML failiformaati ning kõik konfiguratsiooni parameetrid on välja toodud lisas 1.

4.1.6 Andmebaasi skeemi loomine

Andmebaasi skeemi loomise jaoks kasutame *Liquidbase* teeki. *Liquidbase* teek aitab teha valmis andmebaasi tabeli loomise skriptid ning neid käivitada kui rakendust käivitatakse.

4.1.7 Andmebaasi päringud

Andmebaasi päringute tegemiseks kasutati *Specification API*-t. Antud API võimaldab teha kiirelt ja mugavalt dünaamilisi andmebaasi päringuid.

```
Specification.<StageDbo>where((root, query, criteriaBuilder) ->
criteriaBuilder.equal(root.get(StageDbo_.RALLY).get(RallyDbo_.ID), rallyId));
```

Joonis 10. Näidis andmebaasi kitsendusest kasutades *Specification JPA API*-t, et otsida rallietappe, kui on olemas ralli id

Jooniselt 10 on näha näidet *Specification API* kasutusest. Seal luuakse andmebaasi päringule kitsendus, kus otsitakse kõik ühe ralli etapid. „StageDbo“ ja“ RallyDbo“ on objektide nimed, mis esindavad andmebaasi tabeleid. „StageDbo_.Rally“ ja „RallyDbo_.ID“ on abimuutujad, mis viitavad andmebaasi tabeli objekti väljale. Need on genereeritud kasutades *hibernate jpagenmodel* teeki. Muidu tuleks kasutada sõnesid, aga kuna sõnedega võib arendaja teha kirjavigu, siis kasutatakse antud teeki, mis muudab arenduse kiiremaks ning vähem veaohtlikuks.

4.1.8 Vigade käsitlemine veebiteenuses

Mõne veebiteenuse päringu puhul visatakse erind. Veebiteenuses loodi lahendus, kus püütakse erind kinni ning tagastatakse päringule vastus koos vea sõnumiga ja koodiga. Kui üritatakse lisada rallile uut etappi, aga sama numbriga etapp on juba olemas, siis süsteem viskab 409 veakoodi, mis tähendab, et tegemist on duplikaadiga ning sellise numbriga etappi ei saa rallile lisada. Selleks, et sellist lahendust saavutada, kasutati Java koodis *controller advice* klassi, mis püüab erindid kinni ning tagastab vea päringu tegijale.

4.2 Klientrakenduse arendus

4.2.1 Andmete lugemine JSON formaadist

Klientrakenduses tuleb päringu vastused teisendada TypeScript-i objektideks. Selle jaoks kasutas autor *Decorator* arendusmustrit. *Decorator* arendusmustris lisatakse TypeScript klassi väljadele juurde metaandmeid, mis aitavad moodustada TypeScript-i objekte JSON-ist. Metaandmed, mida hoitakse objekti väljade kohta on: elemendi nimi JSON-is, kas tegemist on massiiviga , mis tüüpi see objekti väli on ning funktsioon, millega saab

JSON-i osa teisendada. TypeScript-is on hea sellise arendusmustri jaoks kasutada annotatsioone. Need lisatakse klassi väljadele metaandmetena.

```
export class RallyStageResults {
        public length: string = undefined;
        public time: string = undefined;
        public interrupted: boolean = undefined;
        @JsonProperty({clazz: CompetitorPair})
        public competitor: CompetitorPair = undefined;
}
```

Joonis 11. Näidis TypeScript-i klassist, kus antakse objekti väljale metaandmeid.

Jooniselt 11 on näha kuidas väljale „competitor“ on lisatud annaotatsioon „@JsonProperty“, millele on lisatud ainult üks metaandme väli ehk välja tüüp. Tehti abimeetod, kus asub loogika JSON sõne TypeScripti objektiks teisendamiseks kasutades eespool mainitud metaandmeid. Autor sai selle idee antud blogipostitusest „http://cloudmark.github.io/Json-Mapping/“.

4.2.2 Klientrakenduse turvalisus

Eelpool on mainitud, et veebiteenus kasutab *oauth2 password grant_type* autentimise skeemi. Veebiteenusele tehakse päring kasutaja turvamärgi saamiseks, kui sisselogimise vormil on täidetud nii kasutajanimi kui ka parool. Kasutajanime ja parooli õigsuse puhul saadakse tagasi turvamärk, mis salvestatakse brauseri *localstorage*-sse. Autor leidis, et kõige mõistlikum oleks salvestada *localstorage*-sse. Üheks alternatiiviks oleks *sessionstorage*, aga see tühjendatakse iga kord kui suletakse brauser või selle vaheleht. Rakendusse konstandina ei oleks ka mõistlik salvestada, kuna kui värskendatakse veebilehte, siis läheb antud turvamärk kaotsi ning kasutaja peab ennast uuesti autentima. Küpsistesse ei ole kerge salvestada, kuna TypeScript-iga ei ole võimalik saada ligi päringute küpsistele. *Localstorage*-iga võib tekkida väike turvarisk, et antud turvamärki on võimalik kätte saada kasutades JavaScripti või TypeScripti. Aga kuna rakenduses on üks administraatori kasutaja, siis see risk on minimaalne, et vale inimene saab ligi administraatori kasutajale. Kõikidele turvatud päringutele lisatakse juurde turvamärk, et veebiteenuse päringud läbiksid turvakontrolli.

Kui üritatakse minna lehele, kus saab olla ainult administraatori kasutajaga sisse loginud ning administraatori kasutajaga pole sisse logitud, siis näidetakse 401 vea lehte. Veebilehel on kirjas, et tal pole õiguseid selle lehe nägemiseks ning link, mis viib rallitulemuste lehele.

4.2.3 Klientrakendusest päringute tegemine veebiteenusele

Klientrakenduses üheks komistuskiviks oli see, kuidas anda kaasa kasutaja turvavõti ainult neile päringutele, mis vajavad seda. Selle jaoks kasutas autor *Angular* klientrakenduse baasklassi *HttpInterceptor*. Antud klassi „intercept“ funktsioon käivitatakse enne http päringu tegemist ning seal on võimalik muuta päringute andmeid. Autor kasutas seda klassi, et anda õiged baas URL-id ning päringute päised kaasa päringutele. Kasutaja turvamärgi saamise päringu jaoks tuleb anda kaasa autoriseerimisserveri turvamärk. Selline lahendus võimaldab ka konfigureerida veebiteenuse URL-i. Selle asemel, et teha päring aadressil „http://localhost:8080/api/v1/results/search“, tehti päring „public-api:/results/search“, ning *HttpInterceptor* asendas „public-api:/“ konfigureeritud URL-iga. Samamoodi käitutakse ka kasutaja turvavõtme ja turvatud API päringute puhul. Kui tegemist on turvatud päringuga või turvamärgi päringuga, siis lisatakse ka http päringu päisesse vastav turvamärk. Turvatud päringu puhul on selleks kasutaja turvamärk ja turvavõtme saamise päringu puhul on selleks autoriseerimisserveri turvamärk.

4.2.4 Klientrakenduse konfiguratsioon

```
export const environment = {
  production: false,
  api :{
    url: "http://localhost:8080/api/v1/"
  },
  oauth: {
    secret : "cmFsbHlfcmVzdWx0czpyYWxseV9zZWNyZXRz",
    url: "http://localhost:8080/oauth/"
  }
};
```

Joonis 12. Klientrakenduse konfiguratsioon.

Klientrakenduse konfiguratsiooni jaoks on vaja seadistada veebiteenuse baas internetiaadress, kasutaja turvamärgi saamise baas internetiaadress ning turvamärgi saamise kliendi salavõti.

Samuti on konfiguratsioonis ka väli „production“, mis tuleb muuta „true“-ks klientrakenduse produktsiooni viimisel.

4.2.5 Rallitulemuste kuvamine ning otsimine

Erinevalt olemasolevatest rallitulemuste kuvamise rakendusest, kus kasutati rallitulemuste kuvamiseks mitut erinevat lehte, siis antud töö puhul kasutati ainult ühte veebilehte. Autor valis sellise lahenduse, kuna nii on lihtsam, kiirem ja mugavam leida rallitulemusi. Autor lahendas selle kasutades hüpikakent ning *accordion Boostrap* elementi. Samuti on lehele lisatud kolm otsimise võimalust – võistleja nime järgi, ralli nime järgi ning etapi nime järgi. Lisas 2 on kaks joonist, mis illustreerivad rallitulemuste kuvamist ühel lehel.

4.2.6 Rallietapi keskmiste kiiruste graafik

Rakenduste nõuetes oli välja toodud, et peaks olema kaks graafikut - keskmise kiiruse graafik ning tulemuste võrdlemise graafik. Kuna autor kasutas graafikute tegemiseks *ng2-chart* teeki ning sellega ei olnud võimalk hästi teha mõistliku aja jooksul rallitulemuste

graafikut, siis jäeti see tegemata. Keskmiste kiiruste graafiku jaoks on kasutusel kaks otsinguvälja – ralli nimi ja etapi nimi. Ralli nimi on tekstitüüpi sisendelement, aga nime kirjutades pakutakse kirjutatule sarnaseid rallide nimesid. Kui kasutaja valib sealt ühe, siis teises valiksisend HTML elemendis väärtustakse kõik selle ralli etapid ning kui kasutaja valib sobiva etapi, siis joonistatakse kasutajale graafik, kus on kõik selle etapi tulemused. Lisas 3 on võimalik näha joonised selle veebilehe vaadest.

4.3 Edasiarendus

Kõiki funktsionaalsusi ei jõutud antud diplomitöö jooksul valmis arendada. Rakendust tuleks edasi arendada. Rakendusele tuleks lisada rallitulemuste lisamine kasutades päringuid (API-t). Selle jaoks tuleks teha vastavad päringud ning päringul autentimine läbi süsteemi integratsiooni võtme. Rakenduses puudub ka graafik raja korduvate läbimiste kohta, mis tuleks lisada. Kuna rakendus kasutab mitmeid vabatarkvaralisi teeke ning tegemist on avalikult kättesaadava rakendusega, siis tuleks lisada rakendusele ka vastavad litsentsid. Rakendusele tuleks lisada ka erinevad keeled, et seda oleks võimalik kasutada väljaspool Eestit.

5 Rakenduse testimine

Rakenduse testimiseks kasutati manuaalset testimist, kuna autoril ei jäänud aega, et kirjutada automaatteste, aga ta soovib seda tulevikus teha. Kõik testimised tehti lähtuvalt analüüsis välja toodud rakenduste nõuetest. Mõlemat rakendust testiti korraga.

Kõik testid, mis tehti, jaotati seitsmesse gruppi – rallid, rallietapid, võistlejad, etapi tulemused, süsteemi integratsiooni võtmed, rallitulemuste otsimine, rallietapi keskmiste kiiruste leidmine, graafiku genereerimine, administraatori testid. Iga grupp koosneb mitmest testist. Kõik testigrupid ning testid on välja toodud lisas 4.

6 Kokkuvõte

Diplomitöö käigus arendati valmis rallitulemuste kuvamise ning võrdlemise rakendus. Rakendus võimaldab otsida tulemusi nii ralli, sõitja kui ka ralli ja sõitja järgi. Erinevate sõitjate tulemuste võrdlemiseks kasutatakse graafikuid. Samuti on võimalik näha etapi keskmisi kiiruseid. Rakendusel on olemas administraatori kasutaja, kes saab manuaalselt lisada rallitulemusi süsteemi. Rakendus on võimeline saatma rallitulemusi teisele rakendustele. Rakendus suudab ka otsida olemasolevatelt veebilehtedelt Eesti rallitulemusi. Veebirakendus aitab leida andmeid ning toob võrdlusi nii toimuvate kui ka toimunud rallitulemuste kohta. Rakendust testiti manuaalselt. Rakendus vajab edasiarendust.

Püstitatud eesmärk täideti ning saad valmis rakendus, kust on võimalik lihtsasti otsida rallitulemusi kasutades otsingufiltreid. Samuti on võimalik genereerida rallietappide graafikuid. Lahendus lihtsustab kindla tulemuse otsimist ning annab ülevaate etapi tulemustest.

Kasutatud kirjandus

[1] P. Wójcik, „Why Single-page Applications + REST are better than Dynamic web pages,“ [Võrgumaterjal]. Available: https://applandeo.com/blog/single-page-applications-rest-better-dynamic-web-pages/. [Kasutatud 11 10 2020].

[2] M. Samanpour, „Caching in Web Applications,“ [Võrgumaterjal]. Available: https://www.codementor.io/@meysamsamanpour/caching-in-web-applications-fz1gzizpa. [Kasutatud 17 10 2020].

[3] „File System vs DBMS: Key Differences,“ [Võrgumaterjal]. Available: https://www.guru99.com/difference-between-file-system-and-dbms.html. [Kasutatud 17 10 2020].

[4] T. Manrai, „Graph database vs Relational database,“ [Võrgumaterjal]. Available: https://medium.com/dev-genius/graph-database-vs-relational-database-70f6156f7415. [Kasutatud 17 10 2020].

[5] M. Chand, „What are the Most Popular Relational Databases,“ [Võrgumaterjal]. Available: https://www.c-sharpcorner.com/article/what-are-the-most-popular-relational-databases/. [Kasutatud 17 10 2020].

[6] „Oracle Database,“ [Võrgumaterjal]. Available: https://www.pcmag.com/encyclopedia/term/oracle-database. [Kasutatud 17 10 2020].

[7] „How Much Does Oracle License Cost,“ [Võrgumaterjal]. Available: https://costaide.com/oracle-license-cost/. [Kasutatud 01 12 2020].

[8] „PostgreSQL vs. SQL Server (MSSQL) - Extremely Detailed Comparison,“ [Võrgumaterjal]. Available: https://www.enterprisedb.com/blog/microsoft-sql-server-mssql-vs-postgresql-comparison-details-what-differences. [Kasutatud 17 10 2020].

[9] „What is Java?,“ [Võrgumaterjal]. Available: https://opensource.com/resources/java. [Kasutatud 11 10 2020].

[10] „What is C# used for?,“ [Võrgumaterjal]. Available: https://www.educative.io/edpresso/what-is-c-used-for. [Kasutatud 11 10 2020].

[11] „What is JavaScript?,“ [Võrgumaterjal]. Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript. [Kasutatud 11 10 2020].

[12] „What is JavaScript Used For?,“ [Võrgumaterjal]. Available: https://www.hackreactor.com/blog/what-is-javascript-used-for. [Kasutatud 11 10 2020].

[13] „Getting Started With Java Build Tools,“ [Võrgumaterjal]. Available: https://www.jrebel.com/blog/getting-started-with-java-build-tools. [Kasutatud 11 10 2020].

[14] „NuGet is now fully integrated into MSBuild,“ [Võrgumaterjal]. Available: https://devblogs.microsoft.com/nuget/nuget-now-fully-integrated-into-msbuild/. [Kasutatud 11 10 2020].

[15] „MSBuild,“ [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/visualstudio/msbuild/msbuild?view=vs-2019. [Kasutatud 11 10 2020].

[16] „Spring Framework,“ [Võrgumaterjal]. Available: https://www.journaldev.com/16922/spring-framework. [Kasutatud 11 10 2020].

[17] „What is Micronaut?,“ [Võrgumaterjal]. Available: https://hackernoon.com/what-is-micronaut-37a6565f217d. [Kasutatud 11 10 2020].

[18] „What is Spring Boot?,“ [Võrgumaterjal]. Available: https://stackify.com/what-is-spring-boot/. [Kasutatud 11 10 2020].

[19] „Spring Data JPA,“ [Võrgumaterjal]. Available: https://spring.io/projects/spring-data-jpa. [Kasutatud 11 10 2020].

[20] „Advanced Spring Data JPA - Specifications and Querydsl,“ [Võrgumaterjal]. Available: https://spring.io/blog/2011/04/26/advanced-spring-data-jpa-specifications-and-querydsl/. [Kasutatud 11 10 2020].

[21] „Modules of the Spring Architecture,“ [Võrgumaterjal]. Available: https://dzone.com/articles/4-modules-of-spring-architecture. [Kasutatud 11 10 2020].

[22] „What is Liquibase and why use it?,“ [Võrgumaterjal]. Available: https://philtasticguy.com/devops-for-databases/what-is-liquibase-and-why-use-it/. [Kasutatud 11 10 2020].

[23] „Applying HATEOAS to a REST API with Spring Boot,“ [Võrgumaterjal]. Available: https://dzone.com/articles/applying-hateoas-to-a-rest-api-with-spring-boot. [Kasutatud 11 10 2020].

[24] „MapStruct—An Entity Mapping Solution,“ [Võrgumaterjal]. Available: https://medium.com/@tushar.sharma118/mapstruct-an-exploratory-example-51047d97352d. [Kasutatud 11 10 2020].

[25] „REDUCING BOILERPLATE CODE WITH PROJECT LOMBOK,“ [Võrgumaterjal]. Available: https://objectcomputing.com/resources/publications/sett/january-2010-reducing-boilerplate-code-with-project-lombok. [Kasutatud 11 10 2020].

[26] „JPA Criteria API Queries,“ [Võrgumaterjal]. Available: https://www.objectdb.com/java/jpa/query/criteria. [Kasutatud 11 10 2020].

[27] „JPA Static Metamodel Generator,“ [Võrgumaterjal]. Available: https://docs.jboss.org/hibernate/orm/current/topical/html_single/metamodelgen/MetamodelGenerator.html. [Kasutatud 11 10 2020].

[28] „Top 10 Open Source Java and JavaEE Application Servers,“ [Võrgumaterjal]. Available: https://blog.idrsolutions.com/2015/04/top-10-open-source-java-and-javaee-application-servers/. [Kasutatud 30 10 2020].

[29] „Tomcat vs Jetty - Two Great ServletContainers. Which One to Choose?,“ [Võrgumaterjal]. Available: https://www.dailyrazor.com/blog/tomcat-vs-jetty/. [Kasutatud 30 10 2020].

[30] „Ports-And-Adapters / Hexagonal Architecture,“ [Võrgumaterjal]. Available: http://www.dossier-andreas.net/software_architecture/ports_and_adapters.html. [Kasutatud 30 10 2020].

[31] „Angular Introduction: What It Is, and Why You Should Use It,“ [Võrgumaterjal]. Available: https://www.sitepoint.com/angular-introduction. [Kasutatud 30 10 2020].

[32] „What Is the React.js Framework? When and Why Should I Use React.js in My Project?,“ [Võrgumaterjal]. Available: https://medium.com/@marslan.ali/what-is-the-react-js-framework-when-and-why-should-i-use-react-js-in-my-project-1f606186b58. [Kasutatud 30 10 2020].

[33] „What is a Virtual DOM and why does React use it?,“ [Võrgumaterjal]. Available: https://morioh.com/p/267cde97a158. [Kasutatud 30 10 2020].

[34] „Why Vuejs is so Awesome?,“ [Võrgumaterjal]. Available: https://medium.com/jrc-tech-drive/why-vuejs-is-so-awesome-926e6afed41. [Kasutatud 30 10 2020].

[35] „What is Bootstrap: A Beginners Guide,“ [Võrgumaterjal]. Available: https://careerfoundry.com/en/blog/web-development/what-is-bootstrap-a-beginners-guide/. [Kasutatud 04 11 2020].

[36] „Component architecture with Angular,“ [Võrgumaterjal]. Available: https://blog.angulartraining.com/component-architecture-with-angular-6f7bc9165443. [Kasutatud 04 11 2020].

[37] „Gitlab VS Github VS BitBucket. Which one deserve your time ?,“ [Võrgumaterjal]. Available: https://dev.to/zechtyounes/gitlab-vs-github-vs-bitbucket-which-one-deserve-your-time-2npm. [Kasutatud 30 10 2020].

[38] „Pricing,“ [Võrgumaterjal]. Available: https://github.com/pricing. [Kasutatud 30 10 2020].

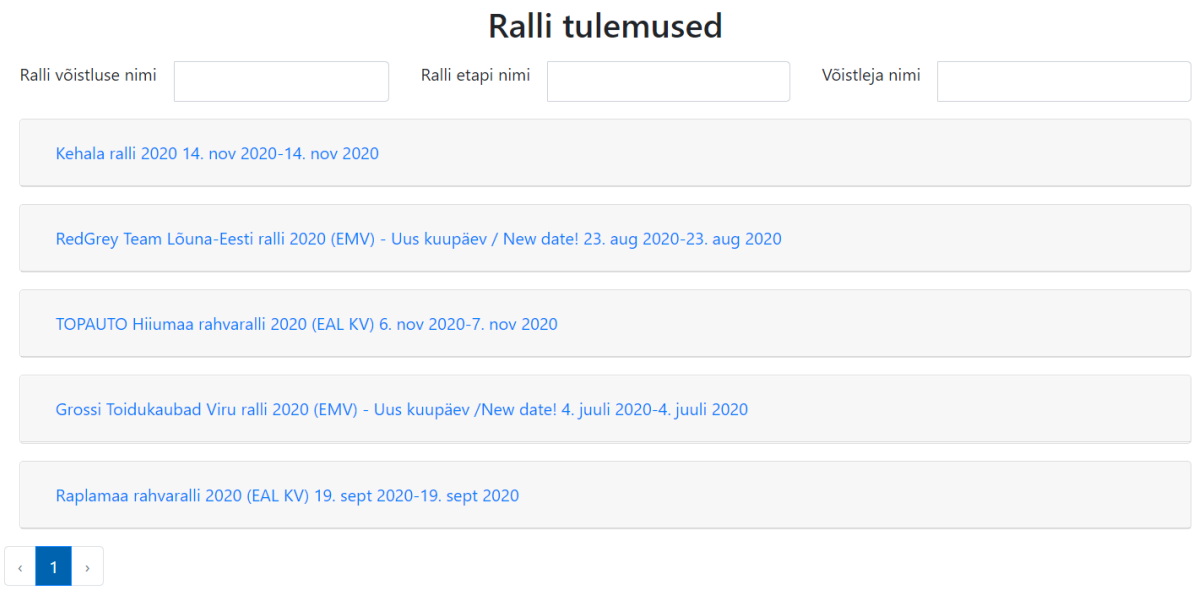
[39] „Using GitHub with Mercurial and TortoiseHG,“ [Võrgumaterjal]. Available: https://www.ryadel.com/en/using-github-with-mercurial-and-tortoisehg/]. [Kasutatud 30 10 2020].

Lisa 1 – Veebiteenuse konfiguratsioon

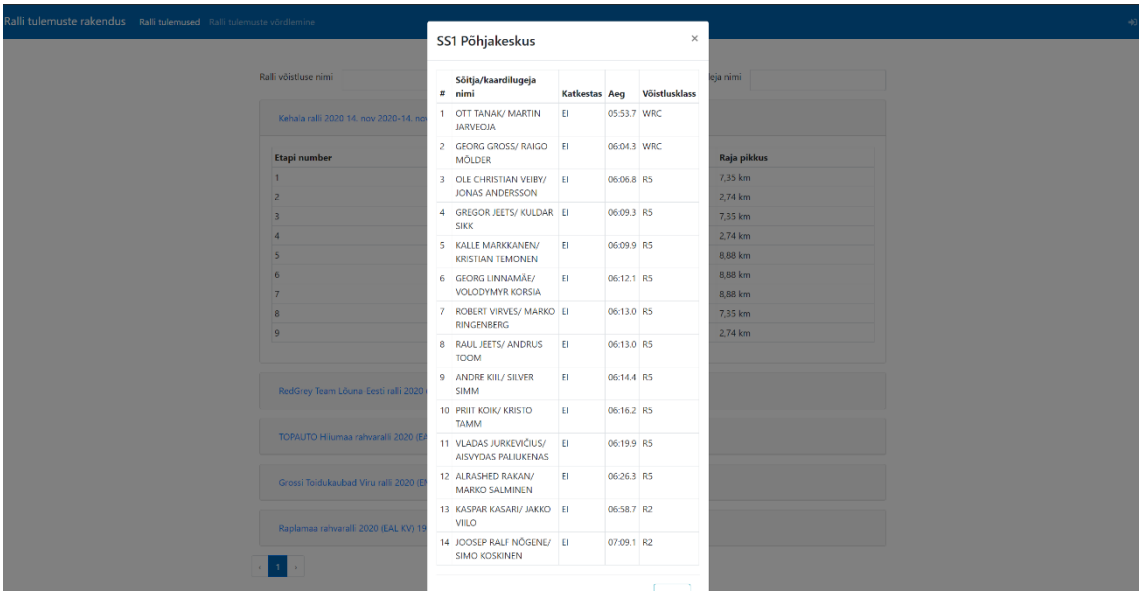
```
spring:
  jackson:
    deserialization:
      USE_BIG_DECIMAL_FOR_FLOATS: true
    default-property-inclusion: non_null
  datasource:
    initialization-mode: always
    url: jdbc:postgresql://localhost:5432/rally-results
    username: postgres
    password: postgres
  liquibase:
    clear-checksums: true
security:
  clientId: rally_results
  clientSecret: rally_secrets
  adminUsername: admin
  adminPassword: admin
  scopes:
    - read
    - write
  grantType: password
  tokenValiditySeconds: 500000
rallies:
  import:
    cron: 0 0 2 * * SUN
```

Joonis 13. Veebiteenuse konfiguratsioon.

Lisa 2 – Rallitulemuste veebilehe vaade

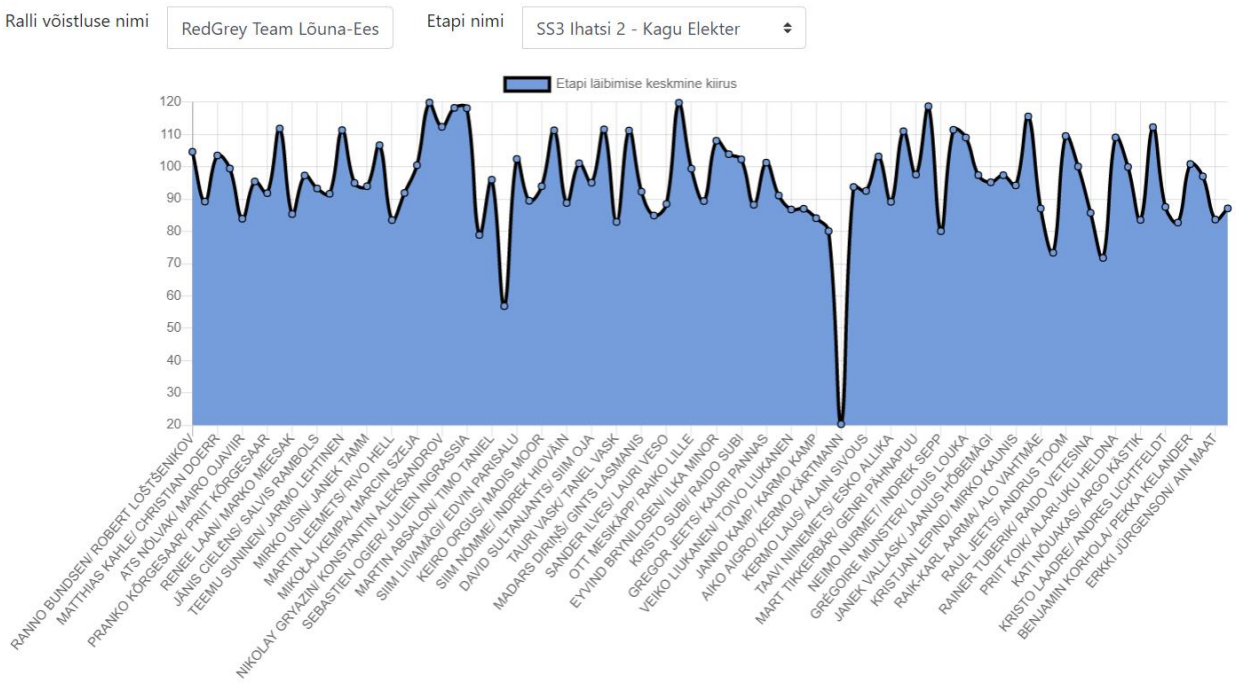


Joonis 14. Rallitulemuste vaade.



Joonis 15. Rallitulemuste vaade koos etappide, ja tulemustega.

Lisa 3 – Rallietapi keskmiste kiiruste graafik



Joonis 16. Rallitulemuste võrdlemise vaade.

Lisa 4 – Läbiviidud testid

1. Rallid

    - Kui kõik väljad on täidetud, siis on salvestamine edukas.
    - Annab kasutajale vea, kui üritatakse salvestada rallit, kus ralli nimi, algus- ja lõppkuupäevad on juba olemas. Veebiteenuses tagastatakse veakood 409.
    - Rallietappe saab edukalt kustutada.
    - Vajutades maantee märgi peale viiakse etappide lehele.

2. Rallietapid

    - Kui lisatakse rallietapp numbriga, mis on juba olemas, siis tagastatakse veebiteenusest veakood 409 ning kuvatakse viga kasutajale, et antud rallil on juba sellise numbriga etapp olemas.
    - Rallietappe saab kustutada vajutades prügikasti märgi peale.
    - Lisamisel antakse viga kasutajale kui etapi pikkus pole number või pole positiivne arv.
    - Kui vajutada „tulemused“, siis viiakse tulemuste lehele.

3. Etappide tulemused

    - Tulemuste lisamisel tuleb täita võistleja nimi ning valida, kas võistleja katkestas või panna tema etapi läbimise tulemus.
    - Etapi läbimise tulemus peab olema formaadis „minutid:sekundid, millisekundid“ ning kui sellist formaati ei ole ette antud, siis viskab veebirakendus veateate.
    - Kui etapi tulemus või katkestamist ning võistluspaari pole märgitud, antakse viga kasutajale.

- Tulemusi saab edukalt kustutada.

4. Võistlejad

    - Lisades uue võistluspaari, kontrollitakse, kas andmebaasis on selline võistleja juba olemas. Kui on olemas, siis kasutatakse olemasolevaid võistlejaid. Vastasel juhul lisatakse uus võistleja koos võistluspaari lisamisega.

    - Võistlejaid ei saa muuta, aga saab kustutada.

5. Süsteemi integratsiooni võtmed

    - Võtmeid saab lisada süsteemi täites kohustuslikud väljad.

    - Kui tabelis on juba olemasolevad võtmed, siis on võimalik vajutada kopeerimise ikoonile ning võtme väärtus kopeeritakse lõikelauale.

    - Võtme väärtuseid saab kustutada ning muuta aktiivseks või mitteaktiivseks.

6. Rallitulemuste kuvamine

    - Kui kõik otsimise väljad on tühjad, siis näidetakse kõige hilisemat kümmet rallit koos etappide ning tulemustega.

    - Kui rallil pole ühtegi etappi või mõnel etapil pole ühtegi tulemust, siis neid ei näidata kasutajale.

    - Rallid on sorteeritud toimumise kuupäeva järgi.

    - Kui täita otsimise väljad, siis ainult need tulemused kuvatakse, mis klapivad otsitava argumentidega.

7. Rallitulemuste võrdlemine

    - Kui kirjutada sõne või täht ralli nime lahtrisse, siis veebirakendus pakub ralli nimesid, kus antud sõne või täht ralli nimes.

- Kui ralli nimi valitud, peaksid etapi lahtrisse tekkima rallietappide nimed
- Kui valida etapp, siis tekkib graafik koos etapi keskmiste kiirustega.

8. Administraatori testid
    - Kui administraator läheb lehele, mida ei ole olemas, saab ta 404 vealehe.
    - Kui administraator läheb mõnele lehele, kus on administraatori õigusi vaja, aga ta turvamärk on aegunud, siis viiakse ta 401 vealehele.

Lisa 5 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[1]

Mina, Rauno Lõhmus

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Rallitulemuste kuvamise ja võrdlemise rakenduse arendus“, 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 Tehinkaülikooli 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.

04.12.2020

[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.*