TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Henri Keerutaja 210738IADB

Veebirakendus eraisiku investeeringute halduseks

Bakalaureusetöö

Juhendaja: Lembit Viilup
Doktorikraad

Tallinn 2023

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: Henri Keerutaja

15.05.2023

Annotatsioon

Investeerimisportfelli tootlus ja varaklasside vahelised jaotused on investori jaoks olulise väärtusega info. Hetkel puudub Eesti inimesel, kes omab investeeringuid erinevates varaklassides, hea võimalus jälgida investeeringute tootlust ja jaotust mugavalt ühest kohast.

Käesoleva bakalaureusetöö eesmärk on luua veebirakenduse prototüüp, mis võimaldab kasutajal koondada erinevate varaklasside investeeringud ühte keskkonda, et näha nende tootlust ja jaotust. Prototüübi funktsionaalsus arvestab, et sihtgrupiks on Eesti füüsilisest isikust investor.

Bakalaureusetöö käigus kaardistatakse Eesti füüsilisest isikust investori vajadused, analüüsitakse juba olemasolevate lahenduste ja alternatiivide sobivust sihtgrupile, kirjeldatakse nõuded uuele lahendusele, määratakse prototüübi skoop ning analüüsitakse tehnoloogiaid rakenduse loomiseks. Arendusprotsessi käigus luuakse veebirakenduse prototüüp vastavalt kirjeldatud nõuetele. Valminud prototüüpi võrreldakse olemasolevate lahendustega ning hinnatakse selle sobivust probleemi lahendamiseks.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 43 leheküljel, kuus peatükki, 17 joonist, viis tabelit.

Abstract

Web Application for Private Investment Management

Investment portfolio returns and allocations are valuable information for investors. Currently, there is no optimal solution for tracking returns and allocations for an Estonian investor with investments across multiple assets.

The Aim of this Bachelor´s thesis is to develop a prototype of a web application that allows the possibility of consolidating information from various assets into a single environment, enabling investors to monitor the performance and allocations of their investments.

In the process of the thesis, the needs of the Estonian private investors will be identified, existing solutions and alternatives will be analyzed, and their suitability for the target group will be assessed. Additionally, the requirements for new solutions will be set and the scope of the prototype will be determined. Suitable technologies for creating a web application prototype will be also analyzed. Throughout the development process, a web application will be created according to the requirements. The created prototype will be compared to existing solutions, and its effectiveness in solving the problem will be evaluated.

The thesis is in Estonian and contains 43 pages of text, six chapters, 17 figures, five tables.

Lühendite ja mõistete sõnastik

401(k)	*Company-sponsored retirement account,* Ameerika Ühendriikides tööandja toetatud pensionikonto.
API	*Application Programming Interface*, rakendusliides.
CSS	*Cascading Style Sheets,* märgistuskeel peamiselt veebilehe kujundamiseks.
CSV	*Comma-separated values,* piiritletud tekstifail, mis kasutab koma väärtuste eraldamiseks.
DBMS	*Database Management Systems,* andmebaasi haldussüsteem, andmete haldamiseks andmebaasis.
DTO	*Data Transfer Objects*, andmeedastusobjekt.
HTML	*Hypertext Markup Language*, märgistuskeel veebilehe sisu struktureerimiseks.
HTTP	*HyperText Transfer Protocol,* protokoll andmete edastamiseks arvutivõrkudes.
HTTPS	*Hypertext Transfer Protocol Secure,* protokoll krüpteeritud andmete edastamiseks arvutivõrkudes.
IRA	*Individual Retirement Account,* individuaalne pensionikonto, mis on kasutusel Ameerika Ühendriikides ja kuhu saab kanda teatud osa palgast maksusoodustusega.
JDBC	*Java Database Connectivity*, Java liides, mis pakub universaalset suhtlust ja ühendus andmebaaside ja Java rakenduse vahel.
JPA	*Java Persistence API*, spetsifikatsioon, mis on mõeldud andmete ligipääsuks, sisestamiseks ja haldamiseks Java rakenduse ja andmebaasi vahel, kasutades ORM raamistiku.
JSON	*JavaScript Object Notation,* tekstivormingus andmetüüp andmete edastamiseks ja salvestamiseks.
JSONB	*JavaScript Object Notation Binary,* andmetüüp JSON formaadis andmete hoidmiseks binaarsel vormingul.
JVM	*Java virtual machine*, Java virtuaalne masin, mille abil saab jooksutada arvutis Java ja teiste JVM keelte programme.
NoSQL	*Not only SQL,* mitterelatsiooniliste andmebaaside meetod.

ORM	*Object-Relational Mapping,* tehnika mille abil rakendus suhtleb andmebaasiga.
REST	*Representational State Transfer*, arhitektuuriline lahendus API loomiseks.
S&P 500	*Standard and Poor´s* börsiindeks, mis koosneb Ameerika Ühendriikide 500 suurima börsiettevõtte aktsiast.
SOA	*Service-oriented Architecture,* teenustele orienteeritud arhitektuur tarkvaraarenduses, kus süsteem koosneb iseseisvatest teenustest.
SQL	*Structured Query Language,* andmebaasi päringukeel relatsiooniliste andmebaaside jaoks.
SQL injektsioon	*SQL injection*, veebirakenduse turvanõrkus, mis võimaldab ründajal sisestada soovimatuid andmebaasi päringuid.
XSS	*Cross-Site Scripting*, veebirakenduste rünnaku tüüp.

Sisukord

1 Sissejuhatus 11
2 Lahendatav probleem 12
2.1 Probleemi aktuaalsus 13
2.2 Olemasolevad lahendused 14
2.2.1 Empower 14
2.2.2 Simplify 14
2.2.3 Mint 15
2.2.4 Morningstar Portfolio Manager 15
2.2.5 Quicken 15
2.2.6 Ziggma 15
2.2.7 Yahoo Finance 16
2.2.8 Sharesight 16
2.2.9 Alternatiivsed lahendused 16
2.3 Eesmärgi püstitus 17
2.4 Prototüübi skoop 18
3 Loodava veebirakenduse analüüs 19
3.1 Nõuded rakendusele 19
3.2 Veebirakenduse arhitektuur 21
3.2.1 Klient-server arhitektuur 21
3.2.2 Kolmeastmeline arhitektuur 21
3.2.3 SOA 22
3.2.4 Arhitektuuri kokkuvõte 23
3.3 Arendustehnoloogiate valik 23
3.3.1 Tagarakenduse tehnoloogiad 24
3.3.2 Eesrakenduse tehnoloogiad 27
3.3.3 Andmebaas 28
3.3.4 Andmebaasi haldussüsteem 29
3.3.5 Versioonihaldussüsteem 31
3.4 Turvariskid 34

3.4.1 HTTPS 34
3.4.2 Sisendandmed 35
3.4.3 Kasutaja salasõna hoidmine 36
4 Veebirakenduse loomine 38
4.1 Andmemudel 38
4.2 Tagarakenduse loomine 39
4.2.1 Tagarakenduse struktuur 40
4.2.2 RESTful API 43
4.3 Eesrakenduse loomine 44
4.3.1 Eesrakenduse struktuur 44
5 Prototüübi kirjeldus 47
5.1 Kasutajaliidese disain 47
5.2 Prototüübi testimine 49
5.3 Edaspidine tegevus 51
6 Kokkuvõte 52
Kasutatud kirjandus 54
Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks 58

Jooniste loetelu

Joonis 1. Eesti kodumajapidamiste hoiused 13
Joonis 2. Suurimad varaklassid eestlaste investeerimisportfellides 18
Joonis 3. Kolmeastmeline arhitektuur 22
Joonis 4. Programmeerimiskeelte populaarsus 23
Joonis 5. Git harud 32
Joonis 6. Versioonihalduskeskkondade populaarsus 32
Joonis 7. Parameetrite sidumine JDBC liidese näitels 36
Joonis 8. Prototüüp lahenduse andmemudel 39
Joonis 9. Domeeni klassi näide 41
Joonis 10. Näide JpaRepository liidese kasutamisest 41
Joonis 11. Lihtsustatud näide REST API kontrollerist 42
Joonis 12. Eesrakenduse struktuur 45
Joonis 13. Lihtsustatud näide marsruutimis komponendis asuvast massiivist 46
Joonis 14. Kogu portfelli ülevaade 48
Joonis 15. Portfelli detailne vaade 48
Joonis 16. Investeeringute lisamise aeg keskkonda 49
Joonis 17. Uue lahenduse kasutatavuse rahulolu 50

Tabelite loetelu

Tabel 1. Tagarakenduse programmeerimiskeele võrdlus 25
Tabel 2. Tagarakenduse raamistiku valik 26
Tabel 3. Eesrakenduse raamistiku võrdlus 27
Tabel 4. Andmebaasi haldussüsteemi võrdlus 30
Tabel 5. Versioonihalduskeskkondade võrdlus 34

1 Sissejuhatus

Investeerimisportfelli tootlus, varaklasside vaheline jaotus ja dünaamika on investori jaoks olulise väärtusega informatsioon. Ülevaade investeerimisportfelli muutuste ja sisu kohta võimaldab hinnata ja võrrelda tehtud investeeringuid ning selle põhjal teha tuleviku investeerimisotsuseid. Samuti aitab ülevaade tootlusest ning jaotustest hinnata ja maandada portfelli riske.

Kuna erinevate varaklassidega tehakse tehinguid erinevates keskkondades, on vajalikud andmed mitmes kohas laiali. Samuti saab ühe varaklassi investeeringuid hoida mitmes keskkonnas. Näiteks aktsiaid erinevate pankade kontodel. Eesti investoritele, kes omavad investeeringuid mitmetes varaklassides või kasutavad tehingute tegemiseks erinevaid keskkondi, ei ole hetkel rahuldavat lahendust mis võimaldaks koondada investeeringutega seotud andmed ühte kohta.

Bakalaureusetöö käigus kaardistatakse Eesti füüsilisest isikust investori vajadused, analüüsitakse juba olemasolevate lahenduste ja alternatiivide sobivust sihtgrupile, kirjeldatakse nõuded uuele lahendusele ja määratakse uue lahenduse prototüübi skoop. Samuti analüüsitakse tehnoloogiaid veebirakenduse loomiseks, et valida prototüübi valmistamiseks sobivamad. Arendusprotsessi käigus luuakse veebirakenduse prototüüp vastavalt kirjeldatud nõuetele. Valminud prototüüpi võrreldakse olemasolevate lahendustega ning hinnatakse selle sobivust probleemi lahendamiseks.

Veebirakenduse peamine eesmärk on võimaldada kasutajal koondada Eesti investorite seas populaarsemate varaklasside investeeringud võimalikult mugavalt ühte kohta, et saada ülevaade millest portfell koosneb ja mis on selle tootlus. Veebirakendus näitab kasutajale investeeringute tootlust ja jaotust nii varaklasside kui ka üksikute finantsinstrumentide lõikes. Sekundaarseks eesmärgiks on teha investeeringute sisestamine kasutajale võimalikult mugavaks.

2 Lahendatav probleem

Investeeringutega seotud riskide vähendamiseks ning tootluse suurendamiseks on vajalik teada, kuidas investeeringud on jaotunud ning mis on olnud nende tootlus. Arvutatud tootlus aitab investoril hinnata oma tehtuid otsuseid ja võrrelda muutusi erinevates varaklassides. Lisaks aitab tootluse teadmine võrrelda kuidas on investori investeerimisportfell muutunud võrreldes teiste investoritega või väärtpaberituru keskmisega. Ülevaade portfelli jaotusest on vajalik riskide maandamiseks ja valitud investeerimisstrateegiast kinnipidamiseks.

Majandusteadlane Harry Markowitz töötas välja kaasaegse portfelliteooria mille peamine komponent on just mitmekesistamine. Markowitz sai teooria väljatöötamise eest ka Nobeli majanduspreemia [1]. Maailma üheks edukamaks peetav investor Warren Buffet on oma testamendis ette näinud, et kogu tema raha investeeritakse tema naise nimele selliselt, et 10% pannakse valitsuse lühiajalistesse võlakirjadesse ja ülejäänud 90% madalate kuludega S&P 500 (*Standard and Poor's 500*) indeksfondi [2]. Erinevaid teooriaid on veel, kuid olenemata milline teooria valida, on nende praktiseerimiseks oluline teada kuidas investeeringud on jaotunud.

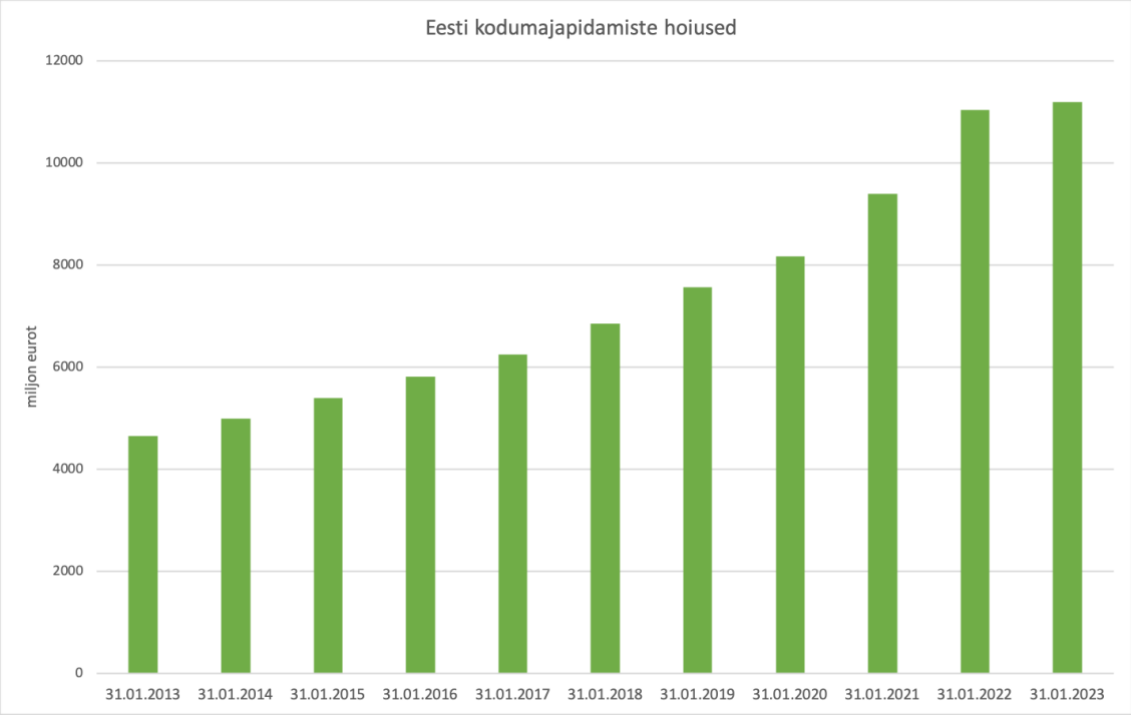
Selleks, et erinevate investeeringute tootlus oleks omavahel võrreldav on oluline määrata täpne ajaperiood, mille jooksul tootlust arvutatakse. Kõige levinum viis on arvutada aastatootlust [3]. Samuti on täpse tootluse arvutamiseks vajalik arvestada nii kapitali kasvu kui ka investeeringutega seotud rahavoogudega, sealhulgas intresside ja dividendidega. See on ka üheks põhjuseks, miks ei sobi internetipangas väärtpaberite kõrval kuvatav kogutootlus investeeringute tootlikkuse hindamiseks. Samuti ei näe internetipangas juba müüdud finantsinstrumentide tootlust.

Erinevate varaklassidega tehakse tehinguid erinevates keskkondades. Investor võib osta aktsiaid kasutades internetipanka, anda välja laene läbi ühisrahastus-platvormi ja omada üürikorterit, millega seotud info on sisestatud mõnda tabelarvutustarkvarasse. Samuti saab ühe varaklassi investeeringuid hoida mitmetel kontodel, näiteks aktsiaid erinevate pankade kontodel või hoopis maaklerfirma kontol. Sellest tulenevalt on vajalikud andmed

mitmes kohas laiali. Portfellist ülevaate saamiseks on investoril vaja andmed koondada ühte keskkonda kokku. Eesti inimesel selleks hetkel mugavat lahendust ei ole.

2.1 Probleemi aktuaalsus

Eesti kodumajapidamiste hoiuste muutust krediidiasutustes iseloomustab joonis 1 [4]. Jooniselt on näha, et Eesti Panga andmetel on hoiused aastast aastasse kasvanud. Sellest tulenevalt jääb Eesti inimesel igal aastal järjest rohkem kapitali ka investeerimiseks. Investeerimishuvi kasvu eestlaste seas näitab see, et kodanike arv, kellel on väärtpaberikontol aktsiaid, ületas 2022 aastal 100 000 piiri. 2015 aastal oli väärtpaberikontode arv ligikaudu 20 000.



Joonis 1. Eesti kodumajapidamiste hoiused [4]

2.2 Olemasolevad lahendused

Olemasolevaid rakendusi investeeringute jälgimiseks ning haldamiseks on olemas mitmeid. Järgnevalt on võrreldud äriajakirja Forbes soovitatud investeeringute halduseks mõeldud rakendusi [5]. Lisaks on võrdlusesse lisatud ka Yahoo Finance ja Sharesight lahendused. Neid ei ole küll ajakirjas Forbes välja toodud, kuid Yahoo Finance on finantsuudiste ja andmete keskkond, mis on SimilarWebi poolt järjestatud üheks populaarsemaks uudiste veebileheks [6], ning pakub võimalust ka investeeringute jälgimiseks [7]. Sharesight on üks suurimaid investeeringute jälgimiseks loodud lahendusi, millel on kasutajaid üle maailma [8]. Samuti eelnevalt nimetatud lahendustele, proovis autor bakalaureusetöö käigus kasutada veel mitmeid olemasolevaid rakendusi. Kõik katsetatud lahendused olid sarnased järgnevalt väljatoodud lahendustele nii funktsionaalsuselt kui ka puuduste poolest ja seetõttu ei ole need eraldi väljatoodud.

2.2.1 Empower

Empower on üle 3 miljoni kasutajaga üks enim kasutatavam veebirakendus kuhu kasutajal on võimalus sisestada enamus varaklasside investeeringud. Pärast andmete sisestamist kuvab rakendus investeeringute tootlused ja jaotuse, samuti kuvatakse selgesti arusaadavad graafikud ning diagrammid. Lisaks on veebirakendusel veel väga palju funktsionaalsusi, sealhulgas võimalus oma igapäevaseid kulutusi ja sissetulekuid jälgida [5], [9]. Paraku takistavad veebikeskkonda sisselogimist teenusepakkuja poolt seatud piirangud, mis lubavad konto luua ainult Ameerika Ühendriikide kodanikel. Mainitud kitsendus piirab Eesti investoril keskkonda kasutada. Samuti on ka veebirakenduse funktsionaalsus suunatud just Ameerika Ühendriikide kodanikule. Näiteks uuendab veebirakendus hindasid ainult Ameerika Ühendriikide aktsiaturgudel kaubeldavatel finantsinstrumentidel.

2.2.2 Simplify

Simplify on suunatud pigem kulutuste ja säästude jälgimiseks, kuid sisaldab ka lihtsustatud võimalust investeeringute halduseks. Keskkonda saab lisada Ameerika Ühendriikides kasutusel olevad pensionikontosid IRA (*Individual Retirement Account*) ja 401(k) (*company-sponsored retirement account*) ning samuti ka väärtpaberikontol hoitavaid varasid [5]. Keskkond on pigem suunatud Ameerika Ühendriikides elavale

inimesele. Eesti investori vajadusi täidab antud lahendus vaid juhul kui investor omab ainult väärtpaberikontol hoitavaid varasid.

2.2.3 Mint

Sarnaselt Simplify lahendusele on ka Mint suunatud kulutuste ja säästude jälgimiseks. Investeeringute halduseks mõeldud funktsionaalsus on sarnaselt piiratud väärtpaberikontol olevate investeeringutega. Suurimaks märgatavaks erinevuseks võrreldes Simplify lahendusega on Bitcoini investeeringute jälgimise võimalus [10]. Eesti investori vajadustele, kellel on investeeringuid ka mujal kui väärtpaberikontol või Bitcoinis, see lahendus ei vasta.

2.2.4 Morningstar Portfolio Manager

Morningstar on tuntud finantsteenuste ettevõtte. Nende peamine toode pakub ligipääsu suurele hulgale andmetele paljude aktsiate ja fondide kohta. Morningstar Portfolio Manager võimaldab kasutajal süsteemi sisestada nii aktsiate kui ka fondide investeeringud ning kuvab seejärel varade jaotuse, tootluse ning kulud [11]. Toode sobib kasutajale, kes omab investeeringuid ainult aktsiates ja fondides. Teisi varaklasse süsteemi lisada ei ole võimalik.

2.2.5 Quicken

Quicken on keerukas eelarvestamise ja finantsplaneerimise keskkond, mis sobib hästi ettevõtetele. Tegu on töölaua rakendusega, mis tähendab, et rakendust saab kasutada ainult arvutis kuhu see on paigaldatud. Kuigi lahendust saavad kasutada ka füüsilisest isikust investorid, on põhjalikud raporteid investeeringute halduseks ning eelarvestamiseks suunatud just ettevõtetele [5]. Sarnaselt Empower lahendusele ei saa kasutajad väljaspool Ameerika Ühendriike keskkonnaga liituda, teenusepakkuja poolt seatud piirangute tõttu. Sellest tulenevalt ei sobi lahendus Eestis tegutsevale füüsilisest isikust investorile.

2.2.6 Ziggma

Ziggma on mõeldud aktiivsele investorile kellel on investeeringuid börsil kaubeldavates aktsiates, fondides või võlakirjades. Rakendus kuvab kasutajale portfelli kogutootlust, pakub aktsiate analüüsi ja näitab erinevaid suhtarvusid aktsiate kohta. Lisaks on võimalus seadistada erinevaid märguandeid, näiteks kui mõne aktsia hind tõuseb üle teatud taseme

või kui finantsinstrumendi jaotus kerkib üle määratud protsendi [12]. Sellised funktsionaalsused on kasulikud pigem kauplejale kui pikaajalisele investorile. Lisaks on kõik need funktsionaalsused kasutatavad ainult börsil kaubeldavate finantsinstrumentide puhul.

2.2.7 Yahoo Finance

Yahoo Finance on lihtsasti kasutatav veebirakendus, kus saad jälgida aktsia, võlakirjade ning valuuta investeeringuid. Investeeringuid on võimalik lisada nii käsitsi ja CSV-vormingus (*comma-separated values*) tekstifaili abil. Börsil kaubeldavate finantsinstrumentide puhul kuvatakse hetkehinnad reaalajas ja arvutatakse ka tootlused. Seda isegi Balti börsil kaubeldavate aktsiate, võlakirjade ja fondide puhul. Võimalus on näha kogutootlust, aastatootlust, liikuvaid keskmisi, suhtarvusid ja palju muud investorile kasuliku [7]. Tegemist on väga hea lahendusega Eesti investori jaoks kelle investeeringud piirduvad aktsiate, võlakirjade või valuutaga. Lahenduse puuduseks võib välja tuua, et kui kasutaja soovib jälgida ka teiste varaklasside investeeringuid, nagu näiteks laenude ja kinnisvara, siis peab ta seda tegema mõnel teisel platvormil. Ülevaate saamiseks kõigist varaklassidest on vajalik kasutusele võtta veel kolmas keskkond.

2.2.8 Sharesight

Sarnaselt Yahoo Finance veebirakendusele on ka Sharesight mõeldud aktsiate, võlakirjade ja valuuta investeeringute jälgimiseks [9]. Võrreldes Yahoo Finance lahendusega ei uuenda Sharesight hindasid automaatselt Balti börsil kaubeldavate finantsinstrumentide puhul. See tähendab, et kasutaja peab need käsitsi sisestama ja pidevalt uuendama. Investeeringuid saab lisada ka automaatselt, kuid seda vaid populaarsemate USA finantsasutuste puhul. Kuna CSV-vormingus tekstifaili abil lisamine puudub, siis mugavat võimalust Eesti pankade kontol olevate investeeringute lisamiseks ei ole.

2.2.9 Alternatiivsed lahendused

Üheks levinud vahendiks investeerimisportfelli jälgimiseks ja haldamiseks on Microsoft Excel või mõni muu sarnane tabelarvutustarkvara.

Tabelarvutustarkvara kasutamise eeliseks on paindlikkus. Kasutaja saab luua just enda vajadustest lähtuva dokumendi. Kasutajal on võimalus luua tabelid just nende

varaklasside kohta, kus ta omab investeeringuid, arvutada välja endale olulised näitajad, kujundada tabelid enda käejärgi ja palju muud.

Teisalt nõuab korraliku dokumendi koostamine keskmisest rohkem teadmisi tabelarvutustarkvarast. Kui kasutajal puudub pädevus luua keerukamaid tabelarvutustarkvara dokumente, on võimaluseks otsida internetist juba teiste investorite poolt valmis tehtud dokumendi põhjad. Saadaval on nii tasuta kui tasulisi dokumente. Tasuta pakutavate kvaliteet on sageli kehvem. Nii tasuta kui tasuliste puhul kulub aega, et neid kasutama õppida.

Samuti peab kasutaja väga täpselt teadma, mida ta arvutada tahab ning kuidas seda korrektselt teha. Oht on sisestada mõni valem valesti ning saadavad arvutused ei ole korrektsed. Sarnaselt on oht sisestada andmed valesti, sest kasutaja peab kõik sisestused tegema käsitsi. Ajapikku muutuvad tabelid aina mahukamaks ja nendes orienteerimine läheb seetõttu järjest ebamugavamaks.

Kokkuvõttes sobib tabelarvutustarkvara investeeringute jälgimiseks kasutajale, kes on keskmiselt asjatundlikum tabelarvutustarkvara kasutaja ning omab häid finantsteadmisi.

2.3 Eesmärgi püstitus

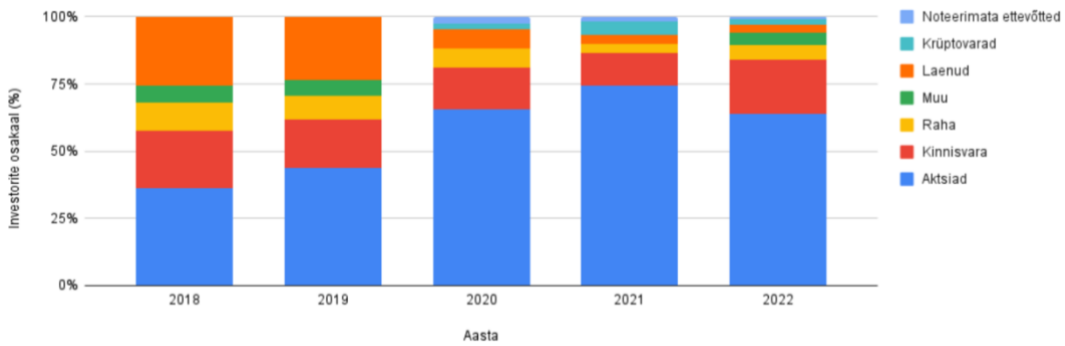
Eksisteerivad lahendused on fokuseeritud peamiselt kas kindlatele varaklassidele, kindla riigi kodanikule või mõlemat kombineerides. Kindlatele varaklassidele fokuseeritud keskkonnad pakuvad väga laia funktsionaalsust ja on seetõttu väga head tööriistad neile kes investeerivad just vastavatesse varaklassidesse. Kindla riigi kodanikule suunatud lahendused muudavad andmete sisestamise just selle riigi kasutajale väga mugavaks. Teiste riigi kodanikele need lahendused väga mugavad kasutada ei ole. Ühe riigi kodanikule suunatud lahendused võivad olla kitsendatud ka kindlatele varaklassidele.

Probleemi lahenduseks pakub autor välja veebirakenduse, kuhu on kasutajal võimalikult mugav sisestada investeeringuid erinevatest varaklassidest, ning samuti nendega seotud tehingud. Pärast andmete sisestamist näeb kasutaja investeeringutega seotud olulist informatsiooni, sealhulgas jaotust ja tootluseid. Kasutajasõbraliku lahenduse loomiseks on vajalik, et see oleks suunatud kindlale sihtgrupile, kelle investeerimisportfellid ja vajadused on sarnased. Antud lahenduse loomisel lähtutakse Eesti inimesest, kellel on investeeringuid erinevates varaklassides. Kasutaja saab andmed sisestada liidestades

rakenduse Eesti suuremate pankadega, importides CSV-vormingus tekstifaili või käsitsi lisades. Lisaks pankadele saab liidestuse lisada Eesti investorite seas populaarsete maaklerfirma platvormidega, ühisrahastus-keskkondadega või mõne muu populaarse finantsteenuse keskkonnaga. Veebirakendus uuendab automaatselt hindasid nii finantsinstrumentidel mis kauplevad maailma suurimatel börsidel kui ka Balti ja Skandinaavia börsidel. Kinnisvara puhul uuendatakse turuhinda vastavalt piirkonna keskmisele müügihinnale. Lisaks on lahendus eestikeelne. Kuna finantssektoriga mitte kokkupuutuva inimese jaoks võivad paljud veebirakenduses esinevad terminid olla keerulised, on emakeelse keskkonna kasutamine arusaadavam.

2.4 Prototüübi skoop

Uue lahenduse täielik väljatöötamine ületab antud bakalaureusetöö mahtu. Seetõttu valmib lõputöö raames lahenduse prototüüp, mis võimaldab kasutajal ühte keskkonda koondada eestlaste seas kolm kõige populaarsemat varaklassi, milleks on aktsiad, kinnisvara ja raha (Joonis 2) [13]. Need varaklassid kokku moodustavad üle 90% Eesti investorite portfellist. Prototüübi eesmärk on pakkuda Eesti investoritele keskkond, kuhu koondada sihtgrupi hulgas populaarsemad varaklassid ning näha nende tootlust ja jaotust. Andmeid saab prototüübis lisada vaid käsitsi. Pärast andmete sisestamist näeb kasutaja kahte peamist näitajat, milleks on portfelli tootlus ja jaotus. Automaatset hindade uuendamist protüübi puhul kasutusele ei võeta ja valmiv lahendus on eestikeelne. Selline prototüüp võimaldab valideerida lahenduse kasulikkust ja kasutajate vajadusi, et planeerida tulevikus lisatavaid funktsionaalsusi.



Joonis 2. Suurimad varaklassid eestlaste investeerimisportfellides [13]

3 Loodava veebirakenduse analüüs

Järgnevalt määratakse lahenduse valmimiseks vajalikud nõuded, analüüsitakse veebirakenduste loomiseks kasutatavaid arhitektuuri viise ja tehnoloogiaid ning valitakse sobivamad prototüübi valmimiseks. Samuti analüüsitakse veebirakendustes enim levinumaid ründevektoreid ning meetmeid nende eest kaitseks.

3.1 Nõuded rakendusele

Nõuete määramisel on lähtutud lahenduse kitsendustest ning on kirjutatud kasutajalugudena. Iga kasutajalugu selgitab mõnda lahenduse funktsionaalsust lõppkasutaja vaatenurgast. Välja on toodud funktsionaalsed ja mitte funktsionaalsed nõuded prototüübi loomiseks. Lisaks on tuleviku nõuete juures väljatoodud sellised nõuded, mis loovad kasutajale küll väärtust, kuid ei mahu bakalaureusetöö raames valminud prototüüpi.

Funktsionaalsed nõuded:

- Kasutajana soovin luua portfelli enda investeeringute jälgimiseks.
- Kasutajana soovin portfelli lisada oma aktsiate investeeringud käsitsi.
- Kasutajana soovin portfelli lisada kinnisvara investeeringuid käsitsi.
- Kasutajana soovin portfelli lisada käsitsi raha varu.
- Kasutajana soovin näha portfelli lisatud varaklasside jaotust.
- Kasutajana soovin näha portfelli lisatud üksikute finantsinstrumentide jaotust.
- Kasutajana soovin näha kogu portfelli tootlust.
- Kasutajana soovin näha portfelli tootlust varaklasside lõikes.
- Kasutajana soovin näha tootlust üksik instrumendi lõikes.

Mittefunktsionaalsed nõuded:

- Kasutajana soovin, et rakendus oleks lihtsasti õpitav ja kasutatav.
- Kasutajana soovin aru saada rakenduse sisust, teadmata inglise keelseid finantstermineid.

Tuleviku funktsionaalsused:

- Kasutajana soovin portfelli lisada oma aktsiate ja fondide tehingud erinevates Eesti suuremates internetipankades API (*Application Programming Interface*) kaudu.
- Kasutajana soovin portfelli lisada oma aktsiate ja fondide tehingud internetipangas CSV-vormingus tekstifaili väljavõtte abil.
- Kasutajana soovin, et Balti börsil kaubeldavate aktsiate ja fondide hinnad uuendatakse igapäevaselt automaatselt.
- Kasutajana soovin, et USA börsidel kaubeldavate aktsiate ja fondide hinnad uuendatakse igapäevaselt automaatselt.
- Kasutajana soovin näha investeeringute osakaalu ja tootlust sektorite kaupa.
- Kasutajana soovin raha varaklassi lisada erinevaid valuutasid.
- Kasutajana soovin, et erinevates valuutades investeeringud konverteeritakse automaatselt valitud valuutaks igapäevaselt.
- Kasutajana soovin portfelli lisada laenude varaklassiga seotud tehingud.
- Administraatorina soovin näha isikustamata andmeid rakenduse aktiivsuse ja kasutamise kohta.
- Administraatorina soovin uuendada selliste finantsvarade hindasid, mis pole automaatselt kättesaadavad.

3.2 Veebirakenduse arhitektuur

Järgnev peatükk kirjeldab veebirakenduste puhul kasutatavaid arhitektuuri viise ja analüüsib nende sobivust lõputöö käigus valmiva prototüübi jaoks.

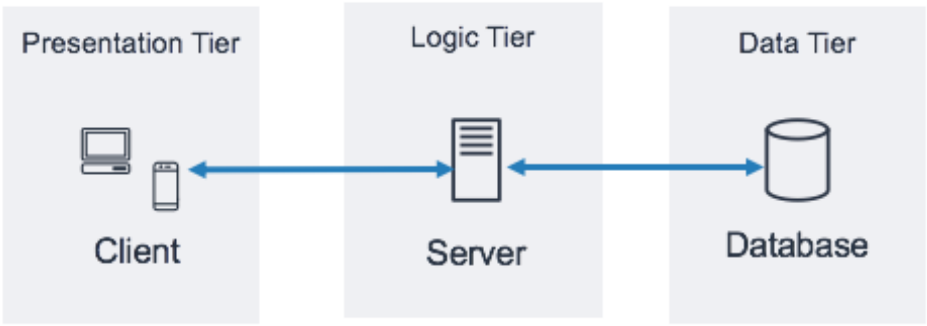
3.2.1 Klient-server arhitektuur

Veebirakendused kasutavad klient-server arhitektuuri. Kliendi poolne rakendus vastutab peamiselt kasutajaliidese kuvamise ja funktsionaalsuse eest [14], sealhulgas milline näeb välja mõni nupp ja mis juhtub kui kasutaja sellele vajutab. Serveri poolne rakendus vastutab andmete lisamise, töötlemise ja salvestamise eest.

Klient-server arhitektuuri puhul suhtlevad klient ja server enamasti päring-vastust (*request-response*) mudelil [14]. Näiteks kui kasutaja vajutab nupule, saadab kliendi poolne rakendus päringu serveri poolsesse rakendusse. Seejärel serveri poolne rakendus töötleb päringut ja saadab vastuse tagasi.

3.2.2 Kolmeastmeline arhitektuur

Kolmeastmeline (*three-tier*) arhitektuur on enim levinud viis klient-server arhitektuuri praktiseerimiseks. Joonisel 3 on näha kuidas kolmeastmeline arhitektuur jagab rakenduse kolmeks loogiliseks ja füüsiliseks komponendiks: esitlus (*presentation tier*), loogika (*logic tier)* ja andmebaas (*database tier*) [15]. Sellise arhitektuuri peamine eelis on see, et igal komponendil on oma vastutusala. Tänu sellele saab komponente üheaegselt erinevate arendajate poolt arendada ilma, et see mõjutaks teisi komponente. Iga komponent saab kasutada alumise kihi teenuseid aga mitte vastupidi. Selline struktuur tagab skaleeritavuse, kuna ükski kiht ei ole teineteisest sõltuv. Samuti on lihtne komponente taaskasutada [16].



Joonis 3. Kolmeastmeline arhitektuur [15]

Esitlus komponendi puhul on tegemist kasutajaliidesega mille kaudu kasutaja rakendusega suhtleb, ning mis vastutab info esitlemise eest. Veebirakenduse puhul töötab esitlus kiht veebilehitsejas ning kolm peamist kasutusel olevat tehnoloogiat selles komponendis on: HTML, CSS ja JavaScript [16].

Loogika komponendis toimub kogu rakenduse äriloogika. Esitlus kihist saadud andmed töödeldakse vastavalt rakenduse äriloogikale [16]. Samuti toimub siin ka vigade haldus. Lisaks saadetakse siit päringud andmete lisamiseks, kustutamiseks ja muutmiseks andme astmes. Loogika kihi rakendus ei tööta enam kasutaja arvutis. Veebirakenduse puhul asub see enamasti pilveteenuse pakkuja serveris.

Andmebaasi komponent vastutab andmete hoidmise, haldamise ja ligipääsetavuse eest. Selles kihis asub ka andmebaasi haldussüsteem (*Database Management System*) mis vastutab andmebaasiga suhtluse eest [16].

3.2.3 SOA

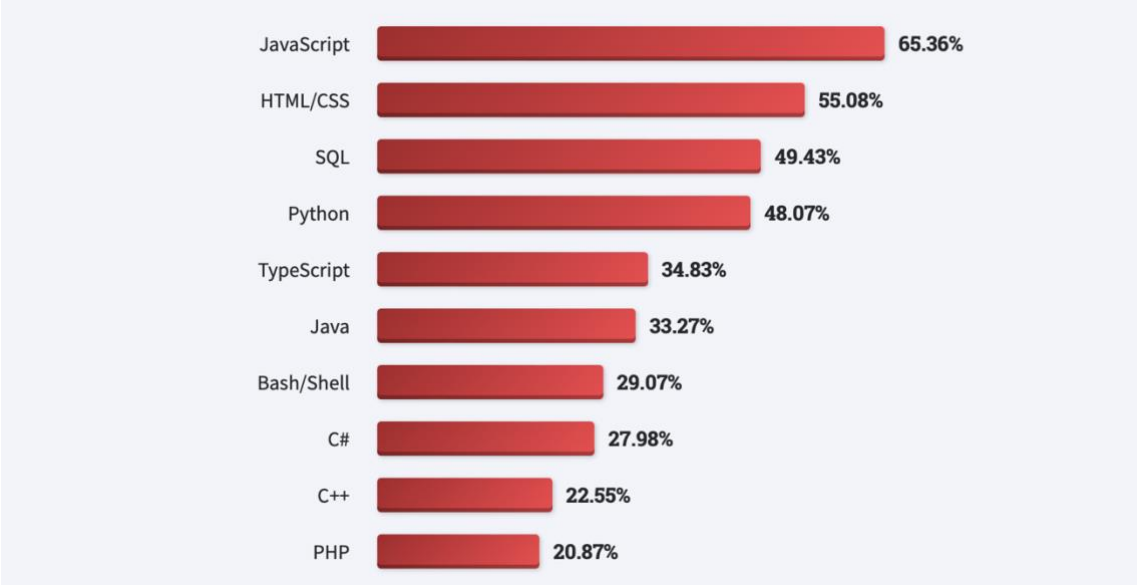
Kuigi aastakümneid oli kolmeastmeline arhitektuur peamisi viise klient-server arhitektuuri praktiseerimiseks, siis tänapäeval muutub aina populaarsemaks SOA (*Service-oriented Architecture*) [16]. SOA puhul on rakenduse funktsionaalsused jagatud eraldi iseseisvateks mooduliteks ehk teenusteks. Teenus on enamasti väga kindlalt määratletud funktsionaalsus. Kõik teenused koos moodustavad ühtse terviku. Selline lahendus võimaldab mugavamalt rakendust hooldada, lisada funktsionaalsust ja testida. Kuna kõiki teenuseid peab haldama ning iga teenus peab kirjeldama oma liidest, on sellise struktuuri kavandamine ja esmase versiooni tööle saamine ajakulukam [17].

3.2.4 Arhitektuuri kokkuvõte

Kuna SOA puhul puudub autoril isiklik praktiline kogemus ja selle korrektse teostamise jaoks tuleb kulutada täiendavat ressurssi, siis ei ole see antud lõputöö puhul mõistlik valik. Seetõttu valmib protüüp kolmeastmelisel arhitektuuril, millega on kasutajal ka isiklik kogemus ning seetõttu on selle kasutamine efektiivsem.

3.3 Arendustehnoloogiate valik

Järgnevalt analüüsitakse rakenduse loomiseks erinevaid arendustehnoloogiaid ning valitakse neist välja sobivad. Tulenevalt tänapäeva väga laiast arendustehnoloogiate võimalustest on analüüsimiseks valitud vaid enim levinud lahendused. Valides tehnoloogiaid vaid enim levinute seast, tõstab see tõenäosust leida tulevikus meeskonda uusi pädevaid liikmeid. Samuti on sellisel juhul lihtsam ettetulevatele probleemidele lahendusi leida. Analüüsi valimi tegemisel on toetatud Stack Overflow iga-aastasele tarkvaraarendajate küsitlusele, kus osales 2022 aastal üle 70 000 vastaja rohkem kui 180 riigist [18]. Joonisel 4 on toodud välja enim kasutatavate programmeerimiskeelte võrdlus populaarsuse järgi. Lisaks populaarsusele arvestatakse sobiva tehnoloogia valikul ka autori kogemuse, õppekeerukuse ja hinnaga. Tulenevalt bakalaureusetöö ajalistest piiridest on üks olulisemaid kriteeriumeid autori varasem kogemus.



Joonis 4. Programmeerimiskeelte populaarsus [18]

3.3.1 Tagarakenduse tehnoloogiad

Stack Overflow küsitlusest selgunud kümne kõige populaarsema programmeerimiskeele seast sobib tagarakenduse arendamiseks JavaScript, Python, TypeScript, Java, C#, C++ ja PHP [18].

JavaScript on rohkem tuntud kui veebirakenduste skriptimiskeel veebilehitsejas. Samas on võimalik JavaScripti koodi käivitada ka serveris, kasutades selleks näiteks Node.js käituskeskkonda [19].

Python on kergesti õpitav ja lihtsasti loetav populaarne kõrgetasemeline programmeerimiskeel [20]. Pythonit kasutatakse väga erinevates valdkondades, sealhulgas veebirakenduste arenduses, andmete analüüsis, masinõppes, automatiseerimises ja andmete visualiseerimises.

TypeScript võimaldab erinevalt JavaScriptist staatilist tüüpimist (*static typing*), mis tähendab, et andmetüübid määratakse ja kontrollitakse juba kompileerimisel [21]. Selline funktsionaalsus aitab vähendada potentsiaalsete vigade arvu. Kuna TypeScript kompileeritakse tavaliseks JavaScriptiks, saab seda jooksutada samades keskkondades kus JavaScripti.

Java on objekt-orienteeritud programmeerimiskeel. Kompileeritud Java koodi saab jooksutada kõikidel platvormidel mis toetavad Javat, ilma vajaduseta uuesti koodi kompileerida. Kompileeritud Java koodi interpreteerib JVM (*Java virtual machine*) ehk Java virtuaalne masin [22].

C# on objekt-orienteeritud keel mis on loodud Microsofti pool. C# on mõeldud töötama vaid platvormidel, mis toetavad .NET keskkonda. Koodi jooksutamiseks teistel platvormidel on vajalik .NET Native koodi teisaldamise tööriist [23].

C++ on madalatasemeline objekt-orienteeritud programmeerimiskeel, mis programmeerimiskeele C edasiarendus. C++ võimaldab kontrolli süsteemi ressursside ja mälu üle [24].

PHP on skriptimiskeel mis sobib väga hästi just veebilehtede arenduseks. PHP kood jooksutatakse serveris ning genereeritakse HTML mis saadetakse klientrakendusele [25]. PHP keelt on võimalik ka HTMLiga kombineerida.

Tagarakenduse arenduse jaoks sobilike programmeerimiskeeli vastavalt eelnevalt seatud kriteeriumitele on võrreldud tabelis 1. Võrdluses ei ole hinna komponenti, sest kõik eelnevalt mainitud programmeerimiskeeled on kasutamiseks tasuta. Populaarsuse võrdlemisel on lähtutud eelnevalt mainitud Stack Overflow uuringul [18].

Tabel 1. Tagarakenduse programmeerimiskeele võrdlus

Keel	**Autori kogemus**	**Õppekeerukus [26]**	**Populaarsus [18]**
JavaScript	Väike	Madal	65,36%
Python	Väike	Madal	48,07%
TypeScript	Rahuldav	Madal	34,83%
Java	Hea	Keskmine	33,27%
C#	Rahuldav	Keskmine	27,98%
C++	Puudub	Kõrge	22,55%
PHP	Väike	Madal	20,87%

Võrdlusest on näha, et kõige suurem kogemus on programmeerimiskeelega Java. Järgnevad TypeScript ja C#. Mõlemad keeled on kogemuse poolest võrdsed, kuid õppekeerukus on C# puhul suurem. Populaarsuse poolest on nii Java, TypeScript kui ka C# üsna võrdsed.

Veebirakenduse arendamisel on oluline ka raamistik, mis tihti on programmeerimiskeele põhine. Tarkvaraarenduses nimetatakse raamistikuks eelnevalt valmis kirjutatud koodi, mis lihtsustab teatud funktsionaalsuse loomist, ilma et iga tarkvaraarendaja peaks seda nullist kirjutama. Kuigi raamistike pakutavad võimalused omavahel erinevad, on nende kõigi peamine eesmärk lihtsustada arendajata tööd enamikes rakendustes ettetulevates probleemides. Samas võivad erinevad raamistikud lahendada sama probleemi erinevalt. Raamistiku kasutamise eeliseks on arendamise mugavus, kiirus ja turvalisus. Seetõttu on järgnevalt võrreldud tagarakenduse jaoks sobilike raamistike. Valikusse on võetud populaarsemad raamistikud mis sobivad kas Java, TypeScript või C# programmeerimiskeele jaoks, sest eelnevalt selgus, et just nende keeltega on autoril kõige suurem kogemus. Ka raamistike populaarsuse hindamisel on lähtutud Stack Overflow küsitlusest [18].

Veebirakenduste arendamiseks Java baasil on populaarsemaks raamistikuks Spring. Spring raamistik teeb veebirakenduse arendamise lihtsamaks, kiiremaks ja turvalisemaks [27]. Raamistikul on suur kasutajaskond.

Üks populaarseimad valikuid tagarakenduse loomiseks TypeScript keeles on Express.js, mis on paindlik ja minimalistlik, võimaldades luua lihtsa vaevaga veebirakendusi. Express.js omab suurt kasutajaskonda [28].

.NET on kõige populaarsem raamistik veebirakenduste arendamiseks C# programmeerimiskeeles. Microsofti poolt arendatud raamistik on avatud lähtekoodiga ning seda saab jooksutada erinevatel platvormidel. Lisaks on .NET raamistikul hea dokumentatsioon ja suur kasutajaskond [29]. Raamistiku abil saab ehitada veebirakendusi, mobiilirakendusi, pilveteenuseid ning palju muud.

Tabel 2 näitab eelnevalt mainitud tagarakenduse raamistike võrdlust vastavalt eelnevalt seatud kriteeriumitele. Võrdluses ei ole hinna komponenti, sest kõik eelnevalt mainitud raamistikud on kasutamiseks tasuta.

Tabel 2. Tagarakenduse raamistiku valik

Raamistik	**Autori kogemus**	**Õppekeerukus**	**Populaarsus [18]**
Spring	Hea	Keskmine	16,13%
Express.js	Puudub	Madal	22,99%
.NET	Rahuldav	Keskmine	34,55%

Populaarsuse poolest mahuvad kõik eelnevas tabelis väljatoodud raamistikud esimese viie hulka. Seetõttu hindab autor tõenäosust leida lahendust ettetulevatele väljakutsetele suhteliselt võrdseks. Varasemalt selgus, et programmeerimiskeeltest tagarakenduse loomiseks on autoril suurim kogemus Java keelega. Eelnevas tabelis võrreldud raamistikest on autoril suurim kogemus Spring raamistikuga. Kuna Spring raamistik võimaldab luua rakendusi Java baasil, siis on Java programmeerimiskeel ja Spring raamistik hea valik bakalaureusetöö käigus analüüsitava probleemi lahendamiseks.

3.3.2 Eesrakenduse tehnoloogiad

Eesrakendus vastutab kasutajaliidese kuvamise eest. Kliendipoolse eesrakenduse loomiseks kasutatakse peamiselt JavaScript programmeerimiskeelt. Erinevus tuleb raamistiku kasutamisest. Stack Overflow küsitluse tulemusteste kohaselt on kolm populaarsemat eesrakenduse raamistiku React, Angular ja Vue [18]. Kasutades raamistike on võimalus kasutada ka TypeScript programmeerimiskeelt.

React on teek millega saab luua veebirakendusele komponentidest koosnevat kasutajaliidest ning mille komponente saab taaskasutada [30]. Kasutades Reacti teeki saab luua keerukaid veebirakendusi ja see toetab TypeScript keele kasutamist.

Angular on raamistik millega saab arendada keerukaid ja skaleeritavaid veebirakendusi. Raamistik sisaldab suurt valikut teeke mis hõlmab suurt hulka funktsionaalsust, näiteks vormihaldust, klient-server suhtlust ja palju muud. Angulari raamistikus loodav rakendus koosneb komponentidest mis on kirjutatud TypeScript programmeerimiskeeles [31].

Vue.js on avatud lähtekoodiga progressiivne JavaScripti raamistik, mis on mõeldud kasutajaliideste loomiseks. Raamistik sisaldab peamisi vahendeid mis on vajalikud eesrakenduse loomiseks [32]. Vue.js toetab TypeScript programmeerimiskeele kasutamist.

Tabelis 3 on võrreldud eelnevalt väljatoodud kolme eesrakenduse arendamiseks mõeldud raamistiku vastavalt eelnevalt seatud kriteeriumitele. Võrdluses ei ole hinna komponenti, sest kõik mainitud raamistikud on kasutamiseks tasuta. Populaarsuse võrdlemisel on arvestatud varasemalt mainitud Stack Overflow uuringuga [18].

Tabel 3. Eesrakenduse raamistiku võrdlus

Raamistik	**Autori kogemus**	**Õppekeerukus [33]**	**Populaarsus [18]**
React	Rahuldav	Keskmine	42,62%
Angular	Hea	Kõrge	20,39%
Vue.js	Väike	Keskmine	18,82%

Tulenevalt autori vähesest kogemusest Vue.js raamistikuga, ning samuti asjaolust, et tegu on väljatoodud raamistikest kõige vähem populaarsemaga, ei osutu Vue.js sobilikuks valikuks. Populaarsuse poolest on React esimesel kohal. Samuti on React teegi

õppekeerukus madalam kui Angulari puhul. Teisalt on Angular raamistikuga autoril kõige suurem kogemus, mistõttu õppekeerukus ei ole nii oluline komponent. Lähtudes asjaolust, et üheks olulisemaiks kriteeriumiks on autori varasem kogemus, osutub sobivaimaks valikuks Angular.

Eesrakenduses on kasutusel veel ka HTML ja CSS. Tegemist ei ole programmeerimiskeelte vaid märgistuskeeltega mille abil saab luua veebilehe struktuuri ja kujundust.

3.3.3 Andmebaas

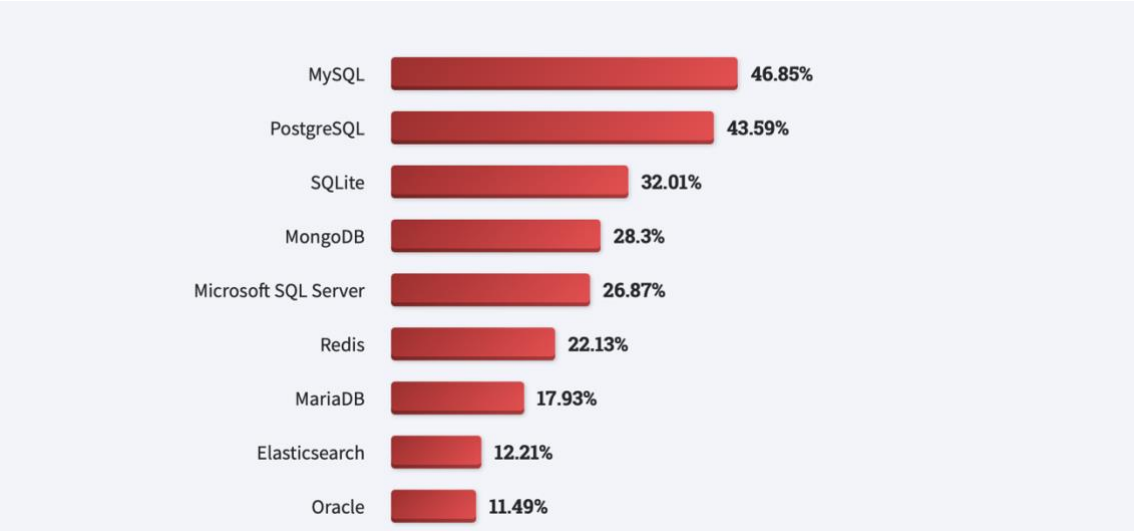
Suurem osa rakendusi kasutab tänapäeval relatsioonilist andmebaasi. Selline andmebaas koosneb relatsioonide kogumist. Seoseid tabelite vahel kirjeldatakse relatsioonilises andmebaasis andmeväljade väärtuste kaudu. Sellist andmevälja kutsutakse võtmeks. Primaarvõti (*primary key*) on unikaalne ja selle eesmärk on identifitseerida andmebaasi kirje. Välisvõtit (*foregin key*) kasutatakse seoste loomiseks. Välisvõtmena kasutatakse enamasti seostatava kirje primaarvõtit, sest on oluline, et seos oleks unikaalne. Andmete pärimiseks kasutatakse enamasti SQLi (*Structured Query Language*). Relatsiooniline andmebaas võimaldab andmeid hoida struktureeritult. See tähendab, et andmeid hoitakse organiseeritult, kindlalt määratud andmetüübina mis vastab andmemudelile, ning jälgib kindlat järikorda. See tagab andmete terviklikkuse, lihtsa juurdepääsetavuse ja kasutatavuse [34].

Mitterelatsioonilisi andmebaase kutsutakse ka NoSQL (*Not only SQL*) andmebaasideks. Seda seetõttu, et enamasti ei kasutata andmete määramiseks ja pärimiseks SQLi. Nende peamiseks eeliseks võrreldes relatsiooniliste andmebaasidega on struktureerimata andmete hoidmine. See tagab küll parema paindlikkuse, kuid võib põhjustab ka andmete terviklikkuse kadu. Samuti on mitterelatsioonilised andmebaasid horisontaalselt skaleeritavamad. See tähendab, et koormuse jaotamiseks ja jõudluse tõstmiseks saab paralleelselt käivitada mitu sõlme. Seda võimaldab andmehulkade iseseisvus. Relatsioonilise andmebaasi puhul on see raskem, sest seoseid erinevate sõlmede vahel jaotada ja säilitada on keeruline. Samuti on NoSQL andmebaaside puhul seosed nõrgemad kui relatsioonilastel andmebaasidel. Enim kasutatavad mitterelatsioonilased andmebaasid on võti-väärtus, dokument-orienteeritud ja veerupõhised andmebaasid [35].

Lähtuvalt andmete terviklikkuse olulisusest ning struktureeritud lähteandmetest on antud bakalaureusetöö käigus analüüsitava veebirakenduse prototüübi puhul eelistatud valik relatsiooniline andmebaas. Seetõttu mitterelatsiooniliste andmebaaside tüüpe edasi ei analüüsita.

3.3.4 Andmebaasi haldussüsteem

Andmebaasihaldus süsteem (*Database Management Systems*) ehk DBMS on tarkvara andmebaaside loomiseks, haldamiseks ja kasutamiseks [35]. Sellise tarkvara valik on veebirakenduse puhul kriitilise tähtsusega, sest sellest sõltub nii rakenduse jõudlus kui ka turvalisus. Joonis 7 iseloomustab Stack Overflow 2022 aasta küsitluse tulemusi andmebaasi haldussüsteemi populaarsuse kohta [18]. Jooniselt selgub, et neli kõige populaarsemat valikut on MySQL, PostgreSQL, SQLite ja MongoDB. Kuna MongoDB puhul on tegemist mittereatsioonlise andmebaasi haldussüsteemiga, siis seda antud töö jooksul edasi ei uurita.



Joonis 7. Andmebaasi haldussüsteemide populaarsus [18]

MySQL on avatud lähtekoodiga relatsiooniline andmebaasihaldus süsteem. MySQL arendamisel loobuti SQL standardi täielikust kinnipidamisest, et saavutada parem kiirus ja töökindlus. Selle negatiivseks küljeks on mõningad funktsionaalsed piirangud. Andmebaasi haldussüsteemi kasutamine kommertslikel eesmärkidel on tasuline [36].

PostgreSQL on samuti avatud lähtekoodiga relatsiooniline andmebaasi haldussüsteem mis suudab tõhusalt käsitleda mitut ülesannet samaaegselt. PostgreSQL on üritanud

võimalikult täpselt jälgida SQL standardit ning toetab enamus SQL funktsionaalsust [37]. Lisaks toetab PostgreSQL keerukaid andmetüüpe, sealhulgas JSONB [38] ja mitmemõõtmeline massiiv. Kuigi PostgreSQL jääb enamasti jõudluselt alla MySQL andmebaasi haldussüsteemile [36], ei saa jõudluse teste andmebaasi haldussüsteemide puhul võtta absoluutse tõena, sest jõudlus sõltub ka konkreetsetest päringutüüpidest. Paremat jõudlust näitab PostgreSQL näiteks suurte andmehulkade ja keeruliste päringutega. PostgreSQL on täielikult tasuta kasutamiseks.

SQLite on iseseisev relatsiooniline andmebaasi haldussüsteem, mille üheks peamiseks omaduseks on töökindlus ning jõudlus isegi vähese mäluga keskkondades. Iseseisvuse all mõeldakse seda, et töötamiseks läheb vaja väga vähe sõltuvusi, ning see töötab kõigis operatsioonisüsteemides. Lisaks puudub kasutuselevõtmisel konfiguratsiooni vajadus. SQLite ei sisalda kasutajate haldamise võimalust ja toetab väga piiritletud andmetüüpe. SQLite on avatud lähtekoodiga ja mõeldud tasuta kasutamiseks [39]. Kuna SQLite kasutab vähe mälu, on see väga hea valik näiteks mobiilirakendustes. Väheste andmetüüpidega piiritletusse tõttu ei sobi see suuremate ja keerukamate lahenduste jaoks. Kuna puudub kasutajate haldamise võimalus, ei ole see hea valik juhul kui on vaja lisada erinevaid kasutajaid mitmete erinevate kasutajaõigustega.

Tabelis 4 on toodud andmebaasi haldussüsteemi erinevused. Kuna andmebaasi haldussüsteemi valiku puhul on kriitilise tähtsusega ka selle turvalisus ja jõudlus, siis on ka neid kriteeriumeid võrdluses arvesse võetud.

Tabel 4. Andmebaasi haldussüsteemi võrdlus

Andmebaasi-haldur	**Autori kogemus**	**Õppe-keerukus [36]**	**Turvalisus [36]**	**Jõudlus [36]**	**Populaarsus [18]**	**Hind**
MySQL	Rahuldav	Madal	Hea	Hea	46,85%	Tasuline [36]
PostgreSQL	Hea	Rahuldav	Hea	Rahuldav	43,49%	Tasuta [37]
SQLite	Rahuldav	Madal	Väike	Hea	32,01%	Tasuta [39]

Kuna SQLite andmebaasihaldussüsteemil puudub kasutajate haldamise võimalus, ei ole see hea valik lõputöö käigus valmiva veebirakenduse jaoks. Tuleviku funktsionaalsuse

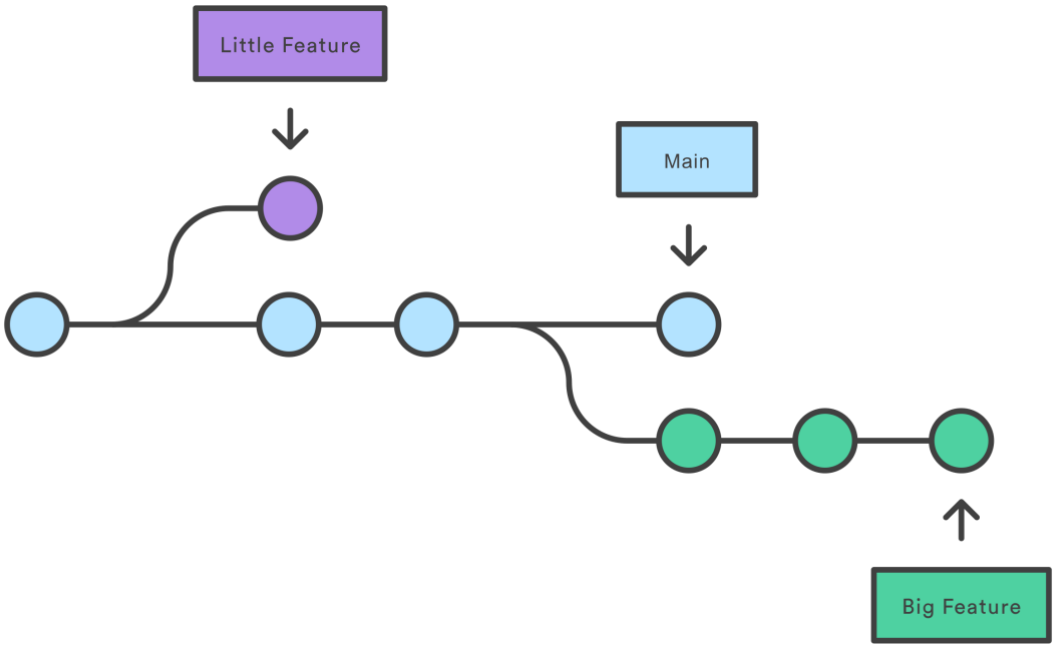
arendamisel näeb autor vajadust anda erinevatele andmebaasi kasutajatele erinevad õigused. MySQL andmebaasihaldussüsteem ei osutu valituks seetõttu, et kommertslikel eesmärkidel on selle kasutamine tasuline. PostgreSQL on parim valik, sest on ka kommertslikel eesmärkidel kasutamiseks tasuta ja lisaks on autori kogemus sellega kõige parem.

3.3.5 Versioonihaldussüsteem

Versioonihaldus on tänapäeva tarkvaraarenduses olulisel kohal. Selle abil on võimalik jälgida muudatusi mida failides tehakse. Näha ei ole mitte ainult muudatused vaid ka nendega seotud info, sealhulgas muudatuste autor. Samuti on failide varasemad versioonid taastatavad. Versioonihaldussüsteem (*version control system*) on tarkvara mis aitab seda teha.

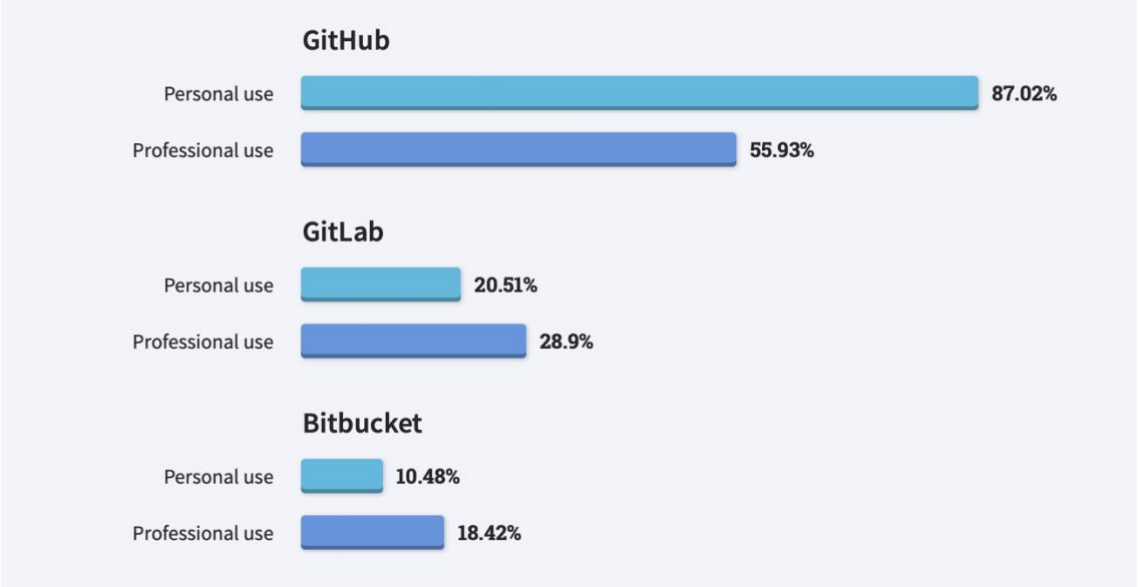
Git on enim kasutatavam versioonihaldussüsteem muudatuste jälgimiseks failides ning samuti töö koordineerimiseks inimeste vahel. Tegu on vabavaralise ja avatud lähtekoodiga hajutatud versioonihaldussüsteemiga. Hajutatud tähendab, et koodihoidlat (*repository*) ei hoita ainult ühe kohas, vaid iga Giti kasutaja kloonib selle endale arvutisse. Hoidlaks nimetatakse failide kogumit mis sisaldab kogu muudatuste ajalugu. Arvutisse salvestatud failide hoidla on koopia originaalist. Originaal hoidlat hoitakse teenusepakkuja serveris või vajadusel enda serveris. Kui arenduse käigus peaks keegi tegema muudatuse, mis varasemalt töötavas rakenduses midagi katki teeb, võimaldab Git minna tagasi töötavale versioonile. Samuti aitab Git hoida koodi uuendatuna erinevate meeskonna liikmete vahel [40].

Erinevad harud (*branches*) on võimalus jagada kood iseseisvateks osadeks. Joonis 5 iseloomustab kuidas erinevate uuenduste tarvis on loodud uued harud [41]. Selliselt jääb koodibaas, mis asub peamise (*main*) harul puutumatuks seni kuni uus haru uuesti peamise haruga ühendatakse. Selliselt on võimalus arendada ja testida igat uuendust eraldi haru peal.



Joonis 5. Git harud [41]

Kuna versioonihalduskeskkonna abil on võimalus koodi hoiustada teenusepakkuja serveris, siis on koodibaasile ligipääs kõikjalt tagatud ja lisaks on ka seda mugav kõigiga jagada. Lisaks on versioonihalduskeskkonnad loonud mitmeid võimalusi mis laiendavad versioonihaldussüsteemi funktsionaalsust. Jooniselt 6 on näha, et enim kasutatavad Giti peale loodud versioonihalduskeskkonnad on GitHub, GitLab ja Bitbucket [18].



Joonis 6. Versioonihalduskeskkondade populaarsus [18]

GitHub on enim kasutatav Giti peale loodud versioonihalduskeskkond, pakkudes lisaks koodihoidla võimalusele ka muud funktsionaalsust, näiteks graafilist kasutajaliidest, mis teeb Giti kasutamise mugavamaks ja arusaadavamaks. Palju lisa funktsionaalsust sõltub sellest, kas kasutada tasuta või tasulist toodet. Üks olulisi erinevusi on, et tasuta versioonis saab hoida ainult avaliku lähtekoodi ehk koodihoidlas hoitav kood on kõigile nähtav [42].

GitLab on kasutajate arvu poolest suuruselt teine Giti peale loodud versioonihalduskeskkond. Sarnaselt GitHub keskkonnale pakub ka GitLab graafilist kasutajaliidest, kuid võimaldab hoida lähtekoodi privaatsena ka tasuta versiooni puhul [43].

Bitbucket on sarnaselt eelnevalt väljatoodud lahendustele graafilist kasutajaliidest pakkuv versioonihalduskeskkond, mis on samuti loodud Git versioonihaldussüsteemi peale. Bitbucketi peamine erinevus võrreldes eelnevalt mainitud lahendustega on lihtne integreeritus enim kasutatava projektide haldamis keskkonnaga Jira [44]. Kuigi Jira integreerimist on võimalik teha ka teiste keskkondadega, siis tänu sellele, et Jira on loodud sama ettevõtte poolt kui Bitbucket, on need loodud koos töötama väga sujuvalt ning mugavalt [45]. Sarnaselt GitLab keskkonnale on võimalus lähtekoodi hoida privaatsena ka tasuta versiooni puhul.

Tabeli 5 on võrreldud eelnevalt mainitud versioonihalduskeskkondasid vastavalt eelnevalt seatud kriteeriumitele. Stack Overflows uuringus on populaarsus eraldi väljatoodud nii isiklikuks- kui ka professionaalseks tarbeks kasutajate seas [18]. Tulenevalt sellest, et üheks populaarsuse mõõtmise põhjuseks on võimaliku tööjõu leidmise tõenäosus tulevikus, kasutame võrdluses just professionaalseks tarbeks kasutajate tulemust.

Tabel 5. Versioonihalduskeskkondade võrdlus

Versioonihalduse-keskkond	**Autori kogemus**	**Õppekeerukus**	**Hind**	**Populaarsus [18]**
GitHub	Väike	Madal	Tasuta/Tasuline [42]	55,93%
GitLab	Rahuldav	Keskmine	Tasuta/Tasuline [43]	28,9%
Bitbucket	Hea	Madal	Tasuta/Tasuline [44]	18,42%

Kõigist võrreldud keskkondades on olemas nii tasuta kui ka tasuline versioon. Samas ei võimalda GitHubi tasuta versioon hoida lähtekoodi privaatsena. Kuna privaatsus ja esialgsete kulude madalal hoidmine on olulised kriteeriumid, siis ei ole antud töö puhul GitHub hea valik. Lähtudes sellest, et üks olulisemaid kriteeriumeid on autori kogemus, siis Bitbucket on sobivam valik kui GitLab. Bitbucketi kasuks räägib ka eelnevalt mainitud mugav ühildamine projekti haldamise keskkonnaga Jira. Nendel põhjustel osutub kõige sobilikumaks valikuks Bitbucket.

3.4 Turvariskid

Kuigi lõputöö raames valmiv lahendus ei paku võimalust väärtpaberite või muu finantsvara hoidmiseks vaid investeeringutega seotud tehingute haldamiseks, võib siiski selliste tehingutega seotud informatsiooni jõudmine kolmandate isikute kätte tekitada palju kahju. Käesolevas peatükis on kirjeldatud peamised veebirakenduste turvariskid ja praktikad [46], mille praktiseerimine on vajalik riskide maandamiseks.

3.4.1 HTTPS

Veebirakenduse puhul on vajadus liigutada andmeid kasutaja veebilehitseja ja serveri vahel. Kasutades selleks tavalist HTTP (*Hypertext Transfer Protocol*) protokolli, edastatakse andmed lihttekstina. Sellisel juhul ei ole kaitstud andmete privaatsus ja turvalisus, sest kui keegi juhtub ühendust pealt kuulama saab ta ligi andmete sisule. Sellist rünnakut nimetatakse ka vahendajarünne (*man-in-the-middle attack*). Kuna

veebirakenduse omanikul pole võimalus kontrollida millist ühendust kasutaja kasutab, on oluline tagada andmete terviklikus ja konfidentsiaalsus. Parim lahendus on selleks kasutada HTTPS (*Hypertext Transfer Protocol Secure*) protokolli. Krüpteerimise abil kaitseb see andmeid, mis liiguvad veebilehitseja ja serveri vahel [46].

3.4.2 Sisendandmed

Üks võimalike riske on kasutaja veebilehel sisestatud info. Kuna kasutajatel võivad olla erinevad kavatsused, siis peab olema kontroll selle üle mida kasutaja sisestada saab. HTML vormid lubavad küll sisestatud infot valideerida, aga see ei kaitse päriselt riskide eest. Seda seetõttu, et kasutaja saab väga kergelt päringu saata ka ilma HTML vormi kasutamata. Üks võimalus selleks on kasutades API klient rakendusi, näiteks Postmani [47]. Samuti võib kasutaja täiesti tahtmatult sisestada andmeid läbi pahatahtlikel eesmärkidel loodud HTML vormi, mis muudab sisestatud infot [46]. Selliselt on võimalik rakenduses välja kutsuda soovimatut käitumist või andmete lekkimist.

Sisendandmed mis ei vasta rakenduse ootustele, võivad põhjustada rakenduses mitte terviklike andmete kuvamist, vigasid äriloogikas või isegi saada kõrvalisel isikul kontroll rakenduse üle. Näiteks võib kasutaja sisestada andmed, mis server rakenduse jaoks on teatud funktsionaalsusega kood. Seetõttu on vajalik sisestatud andmed valideerida lisaks klient rakendusele ka serveris. Valideerides kasutaja sisendit serveri poolses rakenduses, saame vähendada tõenäosust, et sisendandmed ei vasta rakenduse ootustele. Lisaks valideerimisele on oluline ka saneerimine, mis hõlmab soovimatute andmete eemaldamist või asendamist selliselt, et need ei kujuta rakendusele ohtu.

Kõige lihtsam viis mainitud riski vähendada on serveris sisendandmete sisestamist piirata äriloogiliste kriteeriumite järgi. Selleks võib olla näiteks sisestatud numbri vahemik või sõne pikkus. Sellist valideerimist nimetakse ka positiivseks valideerimiseks [46]. Näiteks vanuse sisestamisel ei ole loogiline kui sisestatud väärtus on alla nulli.

Järgmine samm võiks olla keelata sisendid mis sisaldavad potentsiaalselt ohtlike väärtusi. Üheks selliseks väärtuseks on HTML märgend <script>. Sellise märgendi kasutamine võimaldab XSS (*Cross-Site Scripting*) rünnakut. See tähendab seda, et kasutaja saadab sisendina pahatahtliku koodi, mis võib põhjustada sensitiivsetele andmetele ligipääsu, häirida rakenduse tööd või võtta kontrolli rakenduse üle. Potentsiaalselt ohtlike sisendite piiramist nimetatakse ka negatiivseks valideerimiseks. XSS rünnaku vastu aitab ka

HTML väljundi kodeerimine. See tähendab, et sümbolid ja erimärgised mis HTML koodis midagi tähendavad, konverteeritakse tekstiks selliselt, et veebilehitseja tõlgendab neid märke lihttekstina, mitte HTML koodina [46]. Paljud kaasaegsed raamistikud pakuvad võimalust selliseks HTML väljundi kodeerimiseks, on seda turvameedet lihtne kasutusele võtta.

Samuti võivad sisendandmed sisaldada ka SQL päringuid. Sellist ründevektorit nimetatakse SQL injektsioon (*SQL injection*), ning sel viisil on võimalik rikkuda andmete terviklikust, kustutada andmeid või saada ligi teiste kasutajate isiklikele andmetele. Sellist rünnakut on võimalik ennetada parameetrite sidumisega (*parameter binding*). Joonis 7 iseloomustab parameetrite sidumist JDBC (*Java Database Connectivity*) liidese abil. Selline lahendus käsitleb kasutaja sisestatud parameetrid eraldi SQL päringust, tänu millele on palju keerulisem süsteemi sisestada pahatahtlike SQL päringuid [46].

```
void addStudent(String lastName, String firstName) {
        PreparedStatement stmt = getConnection().prepareStatement(
            "INSERT INTO students (last_name, first_name) VALUES (?, ?)");
        stmt.setString(1, lastName);
        stmt.setString(2, firstName);
        stmt.execute();
 }
```

Joonis 7. Parameetrite sidumine JDBC liidese näitel [46]

Enamasti võib öelda, et mida rohkem piiranguid valideerida, seda väiksem tõenäosus on eelmainitud turvariskideks.

3.4.3 Kasutaja salasõna hoidmine

Veebirakendustes kus saab luua kasutajaid, on oluline, et kasutajate salasõnad oleks turvaliselt hoiustatud. Salasõna korrektne hoiustamine ei kaitse vaid väliste rünnakute eest. Ka rakenduse arendajatel ja haldajatel ei tohi olla võimalust näha kasutaja salasõna. Selliste riskide maandamiseks ei tohi kunagi hoida andmebaasis salasõna ennast, vaid salasõna räsi (*hash*). Räsi on kindlate matemaatiliste valemite järgi arvutatud teabekogum. Salasõna ja räsi seos on ühesuunaline, mis tähendab, et räsist ei ole võimalik arvutada salasõna ennast. Salasõna valideerimiseks arvutatakse igakord uuesti räsi ja võrreldakse seda andmebaasis hoiustatud räsiga. Selleks, et kaks kasutajat sama

salasõnaga ei saaks tulemuseks sama räsiväärtust, tuleb enne räsi arvutamist lisada salasõnale ka sool(*salt*). Soolaks krüptograafias nimetatakse salasõnale lisatavat juhusliku bitijada [46].

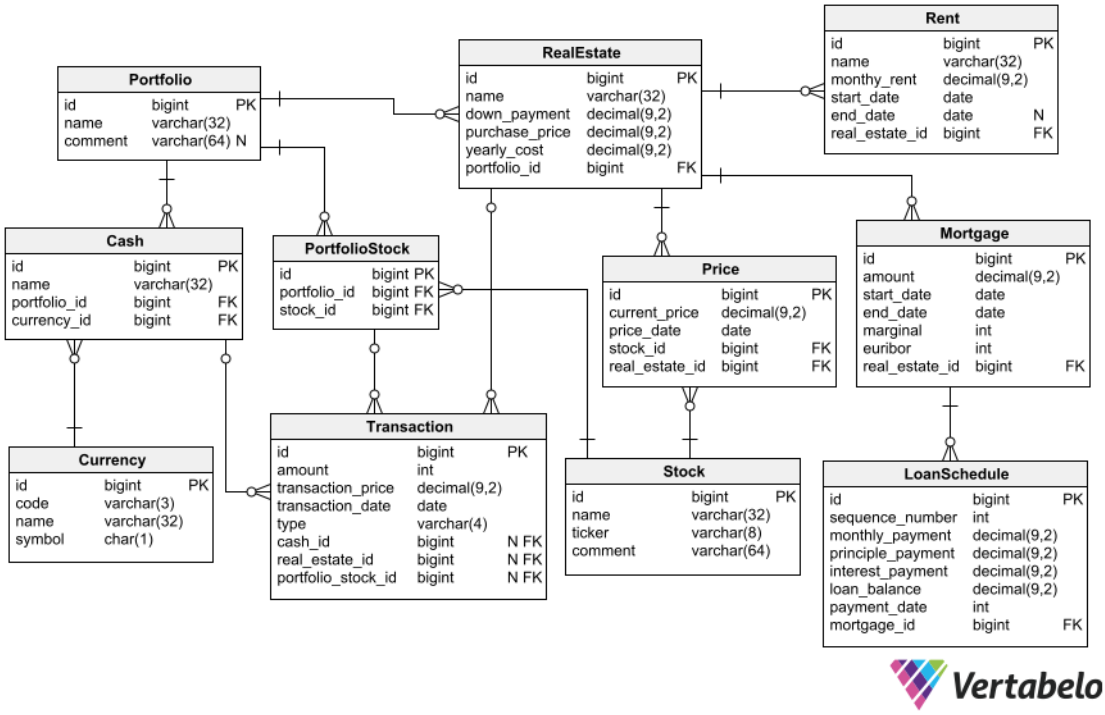
4 Veebirakenduse loomine

Käesolevas peatükis kirjeldatakse veebirakenduse prototüübi arendust. Arendus on jaotatud kolmeks osaks: andmemudeli koostamine, tagarakenduse loomine ja eesrakenduse loomine.

4.1 Andmemudel

Andmemudeli koostamine aitab mõista milliseid andmeid arendatav veebirakendus vajab ning kuidas need omavahel seotud on. Lisaks võib andmemudelit käsitleda kommunikatsiooni vahendina. See tagab selle, et kõik meeskonnast omaksin ühtset arusaama arendatavast rakendusest. Bakalaureusetöö käigus luuakse andmemudeli järgi rakenduse andmebaas. Seetõttu on andmemudeli loomine kriitilise tähtsusega. Andmebaasi on arenduse hilisemas faasis kulukas muuta ja see toob enamasti kaasa ka vajaduse muuta rakendusekoodi. Lisaks mõjutab andmebaas ka kogu rakenduse jõudlust.

Joonisel 8 on näha töö uue lahenduse andmebaasi andmemudel. Andmemudel on loodud Verabelo keskkonnas, kus tudengid saavad luua akadeemilise konto, ning seejärel keskkonda täiesti tasuta kasutada [48]. Andmemudelist on näha, et portfelli saab lisada kolme tüüpi finantsvarasid: aktsiad, kinnisvara ja raha. Kuna börsil kaubeldavad aktsiad on enamasti paljude investorite portfellides, siis on loodud vahetabel *PortfolioStock*, mis kirjeldab seost kindla portfelli ja aktsia vahel. Aktsiaga seotud hinna puhul on tegu turuhinnaga. Kõikide kolme varaklassi tabelitega on seotud *Transaction* tabel, kus hoitakse kindla investeeringu tehinguga seotud informatsiooni, mis on oluline tootluse arvutamiseks. Sealhulgas tehingu kuupäeva, tehingu hinda, kogust ja tüüpi. Kinnisvara investeeringutega seotud tabelis *RealEstate* on lisaks seostatud tabelitega *Rent* ja *Mortage*. Nendes tabelites hoitakse infot vastavalt üürihinna ja hüpoteeklaenu kohta. Mõlemad näitajad on olulised kinnisvaraga seotud tootluse arvutamiseks.



Joonis 8. Prototüüp lahenduse andmemudel

Vertabelo keskkonnas on lisaks andmemudel koostamise võimalusele olemas ka võimalus andmemudeli järgi automaatselt luua SQL skript andmebaasi tabelite ja seoste loomiseks. Koostatud skripti jooksutamine andmebaasis on mugav ja kiire lahendus tabelite ja seoste loomiseks. Kindlasti tuleb seosed üle kontrollida, sest automaatselt koostatud skriptid võivad sisaldada vigasid või mitte arvestada kõiki nüansse.

4.2 Tagarakenduse loomine

Tagarakendus on veebirakenduse osa, mis töötab serveris ning selle ülesanne on andmeid töödelda ja hoiustada. Siin toimub ka kogu äriloogika. Bakalaureusetöö käigus valminud lahendus on loodud Java programmeerimiskeeles ja Spring raamistikuga. Lisaks on kasutatud Spring Boot raamistikku, mis seob kokku mitmed Spring raamistikud ja kolmandata osapoolte tööriistad [49]. Tänu sellele võtab rakenduse arendusega alustamine vähe aega ja ei nõua palju seadistamist. Kuigi Spring Booti kasutamine vähendab paindlikust, ei oma see prototüübi valmistamisel suurt tähtsust, kuna valmiv lahendus ei nõua väga spetsiifilist või keerukat konfiguratsiooni.

4.2.1 Tagarakenduse struktuur

Tagarakendus on jaotatud järgmistesse pakettidesse (*packages*):

- Config
- Controller
- Domain
- DTO
- Mapper
- Repository
- Service

Domain pakett sisaldab domeeni klasse, mis defineerivad andmebaasi struktuuri. Seetõttu luuakse domeeni klassid vastavalt varasemalt kirjeldatud andmemudelile. Kui lisatakse uus domeeni klass, siis lisatakse ka andmebaasi uus tabel. Kui domeeni klassi väljasid uuendatakse, uuenevad ka andmebaasitabeli tulbad. Olemite suhted defineeritakse samuti annotatsioonidega, mis määravad ära ka andmebaasitabelite omavahelised seosed.

Selleks, et märgistada mis on domeeni klass, on vaja Java klassile lisada annotatsioon *@Entity*. Selline annotatsioon on osa JPA (*Java Persistence API*) spetsifikatsioonist, mis on mõeldud kasutamaks ORM (*object-relational mapping*) tehnikat Java rakendustes. JPA pakub standardiseeritud liideseid ja annotatsioone. ORM on tehnika, mille abil rakendus suhtleb andmebaasiga. Kuigi JPAd on võimalik kasutada erinevate ORM raamistikega, kasutatakse projektis Hibernate raamistikku, mis on Spring Boot raamistiku puhul üks enim kasutatavaim ORM raamistik.

Joonis 9 on näide Portfolio nimelisest domeeni klassist veebirakenduse prototüübis. *@Table* annotatsiooniga saab määrata millisesse tabelisse domeeniobjekt vastendatakse. Kui klassi ja andmebaasitabeli nimed on samad, ei ole *@Table* annotatsiooni tegelikult vaja. Selleks, et kood oleks arusaadavam on see siiski lisatud. *@Id* annotatsioon annab JPAle teada, et tegu on klassi identifikaatoriga. *@GeneratedValue* määrab, et identifikaator genereeritaks automaatselt Annotatsiooniga *@Column* saab määrata tabeli tulba omadused, sealhulgas tulba nime.

```
@Entity
@Table(name = "portfolio")
public class Portfolio {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id", nullable = false)
    private Long id;
    @Column(name = "name", nullable = false, length = 32)
    private String name;
    @Column(name = "comment", length = 64)
    private String comment;
    @OneToMany(mappedBy = "portfolio")
    private Set<PortfolioStock> portfolioStocks =
            new LinkedHashSet<>();
    @OneToMany(mappedBy = "portfolio")
    private Set<Cash> cash = new LinkedHashSet<>();
    @OneToMany(mappedBy = "portfolio")
    private Set<Realestate> realestates = new
LinkedHashSet<>();
}
```

Joonis 9. Domeeni klassi näide

DTO pakett sisaldab andmeedastusobjekte (*Data Transfer Objects*). Need on klassid, mis on mõeldud andmete edastamiseks klientrakendusele. DTO klass määrab andmete struktuuri mida kliendile edastatakse. Selleks, et domeeniobjekt vastendada DTO objektiks ja vastupidi, on loodud Mapper klassid, mis asuvad Mapper paketis.

Repository pakett sisaldab JpaRepository klasse, mis vastutavad andmebaasiga suhtluse eest. JpaRepository liidest saab kasutada näiteks peamiste andmebaasi operatsioonide jaoks, milleks on lugemine, loomine, uuendamine ja kustutamine. Kuna need funktsionaalsused on liidese poolt valmis, ei ole vaja kasutajal neid ise kirjutada. Joonis 10 iseloomustab JpaRepository liidese kasutamist. Tüübi parameetritena on määratud domeeni klass, mille jaoks liidest kasutatakse. Samuti domeeni klassi identifikaatori andmetüüp, mis on antud näite puhul 64-bitine pikk täisarv (*Long*).

```
public interface PortfolioRepository extends
        JpaRepository<Portfolio, Long> {
}
```

Joonis 10. Näide JpaRepository liidese kasutamisest

Service pakett sisaldab klasse, milles on kirjeldatud veebirakenduse äriloogika. Nendes klassides asuvad meetodid tagastavad ainult DTO klasse. Service paketis asuvatele klassidele saab päringuid kontrolleri kihis asuvad klassid. Vastavalt päringule saadetakse andmed edasi Repository klassile.

Controller pakett sisaldab REST API kontrollereid. Kontroller võtavad vastu ja töötleb API päringuid, ning saadab need edasi Service klassi, et andmeid vastavalt äriloogikale töödelda. Päringule vastatakse HTTP olekukoodiga (*status code*). Oluline on see, et siin kihis ei ole ligipääsu domeeni klassidele ega Repository liidestele. Samuti tagastavad siin klassides olevad meetodid vaid DTO klassi objekte. Hea tava on lisada kontrolleritele versioonid. See aitab uuendada ja muuta kontrollereid selliselt, et tagatud on ka vanemaid versioone kasutatavate teenuste korrektne toimimine. Joonis 11 iseloomustab lihtsustatud REST API kontrollerit. Jooniselt on näha, et hetkel on kasutusel versioon v1.

```
@RestController
@RequestMapping("/api/v1/portfolios")
public class PortfolioController {
    private final PortfolioService portfolioService;
    public PortfolioController(PortfolioService
    portfolioService) {
        this.portfolioService = portfolioService;
    }
    @GetMapping("")
    public List<PortfolioDto> getAllPortfolios() {
        return portfolioService.findAll();
    }
    @GetMapping("/{id}")
    public PortfolioDto getPortfolioById(
            @PathVariable Long id) {
        return portfolioService.getPortfolioById(id);
    }
}
```

Joonis 11. Lihtsustatud näide REST API kontrollerist

4.2.2 RESTful API

RESTful API on veebirakenduste puhul populaarne lähenemine võimaldamaks suhtlust serverrakenduse ja klientrakenduse vahel. REST on arhitektuuriline lahendus, mis sisaldab hulk nõudeid ja ka piiranguid, standardiseerimaks rakenduste omavahelise suhtluse. API on rakendusliides, mis määrab reeglid rakenduste vaheliseks suhtluseks. API mis jälgib REST nõudeid nimetatakse RESTful API. Üheks olulise omaduseks on võimalus eesrakendus ja tagarakendus arendada iseseisvalt, üksteisest sõltumata. See tähendab, et kui kummaski tehakse muudatus, siis teise koodi pole vaja muuta. Oluline on vaid teada millises formaadis sõnumeid saata. Samuti on RESTful API olekuta (*stateless*), mis tähendab, et päringud ei sõltu eelnevalt vahendatud sõnumitest. Seetõttu peab iga päring sisaldama kogu infot mis on vajalik päringu täitmiseks [50].

RESTful API päringus sisalduvad peamised komponendid:

- HTTP verb: määrab millist operatsiooni väljakutsutakse.
- HTTP päis (*header*): sisaldab päringu kohta käivat informatsiooni, näiteks andmete tüüp.
- Unikaalne internetiaadress (*uniform resource locator*): määrab päringu lõpp-punkti.
- Andmed: mõned HTTP verbid võivad nõuda korrektseks töötamiseks, et päring sisaldaks andmeid.

RESTful API vastuses sisalduvad peamised komponendid:

- Olekurida (*status line*): koosneb kolmes numbrist mis annab märku kas päring õnnestus või mitte.
- Sõnumi keha (*message body*): andmed vastavalt päringu päises defineeritud vormingule.
- Päis: sisaldab infot vastuse kohta, näiteks kuupäev ja andmete tüüp.

Neli enim levinud HTTP verbi on:

- GET: kasutatakse andmete lugemiseks.
- POST: kasutatakse andmete sisestamiseks.
- PUT: kasutatakse olemasolevate andmete uuendamiseks.
- DELETE: kasutatakse andmete kustutamiseks.

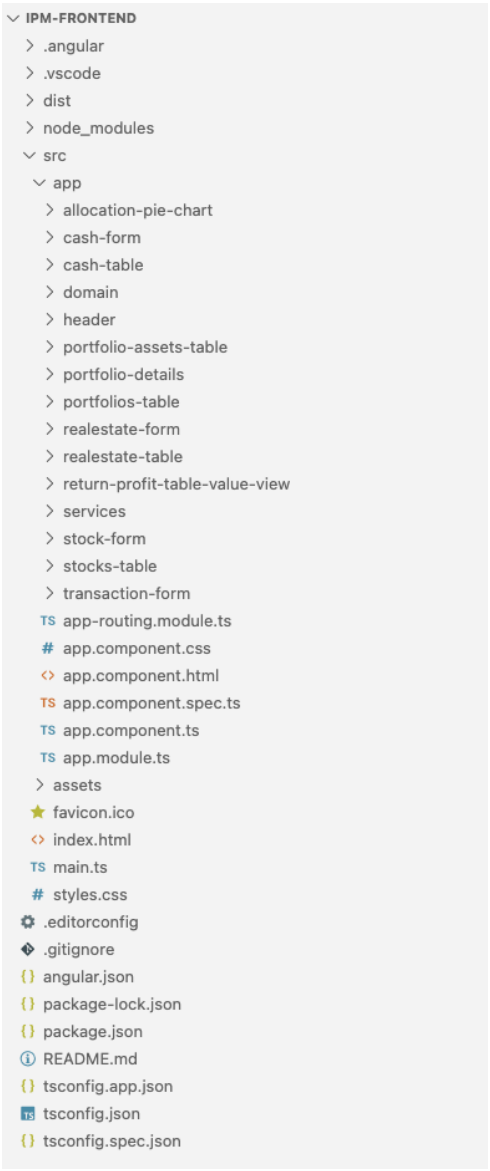
Käesolevas töös kasutatakse JSON-vormingut, mis määratakse HTTP päises *Content-Type* väljal. JSON on ka enim kasutatavaim tüüp, sest see on loetav nii inimesele kui ka arvutile.

4.3 Eesrakenduse loomine

Eesrakendus on veebirakenduse see osa, mida kasutaja näeb ning mille kaudu ta rakendusega suhtleb. Bakalaureusetöö käigus valminud lahenduses on eesrakenduse puhul kasutusel TypeScript programmeerimiskeel ja Angular raamistik.

4.3.1 Eesrakenduse struktuur

Eesrakendus koosneb Angularis komponentidest (Joonis 12). Komponendiks nimetatakse iseseisvat ja korduvkasutatavat koodilõiku, mis koosneb TypeScript klassist, HTML maillist ja CSS-failist. Iga komponent võib kasutada ka teisi komponente. TypeScript klassis defineeritakse komponendi funktsionaalsus ja loogika, HTML mall määrab komponendi väljanägemise ja struktuuri ning CSS-fail kirjeldab komponendi stiili.



Joonis 12. Eesrakenduse struktuur

Lisaks Angulari komponentidele on rakenduses domeeni klassid ja teenuse klassid. Domeeni klassid kirjeldavad rakenduse tööks vajalike andmete struktuurid ja nende vahelised seosed. Teenuse klassides on kirjeldatud funktsioonid tagarakendusele päringu saatmiseks.

Erinevate komponentide vaadete vahel liikumiseks on kasutusel Angulari marsruutimise moodul. Moodulis on defineeritud massiiv, mille iga element on JavaScripit objekt, mis sisaldab kahte atribuuti, rajanime ja komponenti. Rajanimi määrab ära marsruudi internetiaadressi ning komponent määrab, millise komponendi vaadet antud aadressi puhul kasutada.

Joonis 13 iseloomustab marsruutimise moodulis defineeritud marsruutimis massiivi. Kui rajanimele lisada *portfolios* siis kuvatakse kasutajale vaadet mis on seotud *PortfoliosTableComponent* komponendiga. Kui aga sisestada rajanimi *portfolios/1*, siis kuvatakse *PortfolioDetailsComponent* komponendiga seotud vaade. Lisaks edastaks rajaparameeter, mille abil komponent saab pärida vastava portfelliga seotud informatsiooni.

```
const routes: Routes = [
      {path: 'portfolios',
      component:PortfoliosTableComponent},
      {path: 'portfolios/:id',
      component:PortfolioDetailsComponent},
];
```

Joonis 13. Lihtsustatud näide marsruutimis komponendis asuvast massiivist

Tagarakendusega suhtlemiseks on kasutusel Angural raamistiku sisseehitatud *HttpClient* moodul, mis võimaldab teha HTTP päringuid. Samuti võimaldab teenus muid päringutega seotuid tegevusi, näiteks veatöötlust, päringule päiste lisamist, rajaparameetrite lisamist ja palju muud.

5 Prototüübi kirjeldus

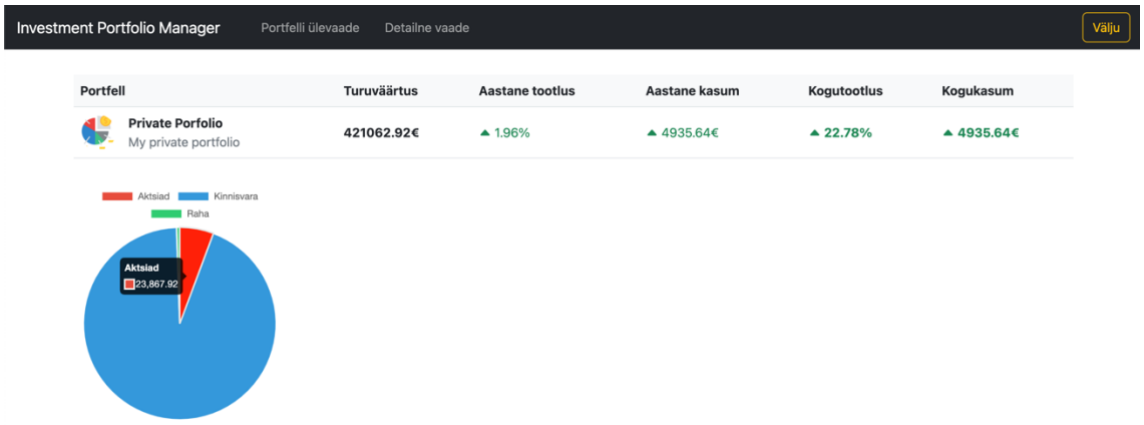
Lõputöö käigus valmis veebirakenduse prototüüp, mis võimaldab kasutajal sisestada aktsiate ja kinnisvaraga seotud investeeringud. Samuti on võimalus lisada rahavaru. Need kolm varaklassi on Eesti investorite seas kõige populaarsemad, kattes 90% investorite portfellist. Peale andmete sisestamist kuvab rakendus kasutajale tootlused ja jaotused nii varaklasside kui ka üksikute finantsinstrumentide lõikes. Valminud prototüüp vastab seatud funktsionaalsetele ja mittefunktsionaalsetele nõuetele.

Veebirakenduse prototüübi arendamiseks kasutatud tehnoloogiad valiti laialt kasutatavate seast. Lahendus valmis jälgides kolmeastmelist arhitektuuri. Tagarakenduse arendamiseks kasutati Spring Boot raamistikku. Eesrakenduse puhul kasutati Angular raamistikku. Andmebaasihaldussüsteemiks valiti PostgreSQL.

5.1 Kasutajaliidese disain

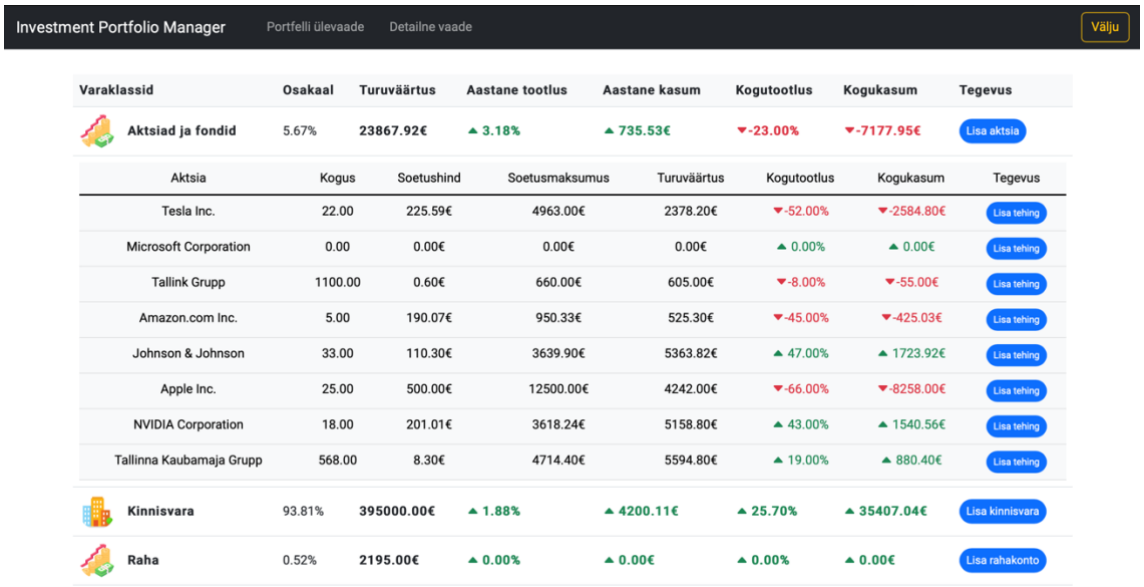
Järgnevalt tutvustatakse veebirakenduse vaateid ning kirjeldatakse nende funktsionaalsust. Veebirakenduse prototüübil on kaks peamist vaadet: kogu portfelli ülevaade ja finantsinstrumentide detailne ülevaade. Erinevate varaklasside investeeringute lisamiseks ja ülevaatamiseks on esmalt vajalik luua portfell.

Joonisel 14 on kujutatud valminud veebirakenduse prototüübi kogu portfelli ülevaadet. Siin vaates näeb kasutaja enda portfelli kogu turuväärtust, jooksva aasta tootlust ja kasumit, kogutootlust ning kogukasumit. Samuti on antud vaates näha sektordiagramm, mis iseloomustab visuaalselt varaklasside jaotust portfellis.



Joonis 14. Kogu portfelli ülevaade

Portfelli detailset vaadet on kujutatud joonisel 15. Selles vaates on näha kõikide varaklasside osakaalud, turuväärtus, aastane tootlust ja kasum, kogutootlus ning kogukasum. Vajutades mõne varaklassi nimele, kuvatakse selles varaklassis sisalduvad finantsinstrumendid ja nendega seotud informatsioon. Samuti on võimalik siin portfelli lisada finantsinstrumente ja nendega seotud tehinguid.

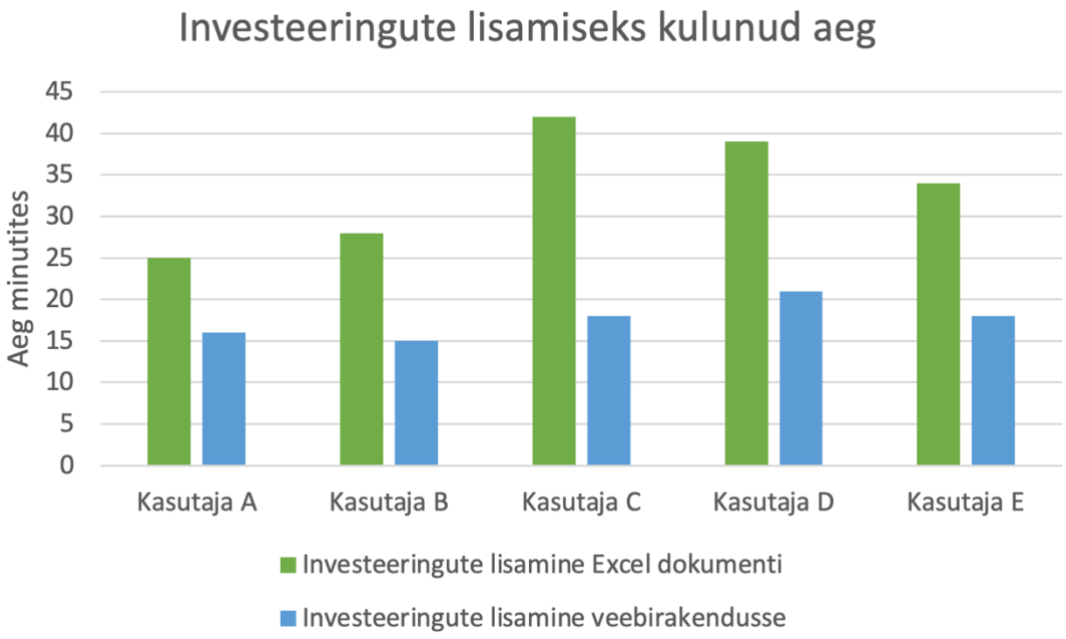


Joonis 15. Portfelli detailne vaade

5.2 Prototüübi testimine

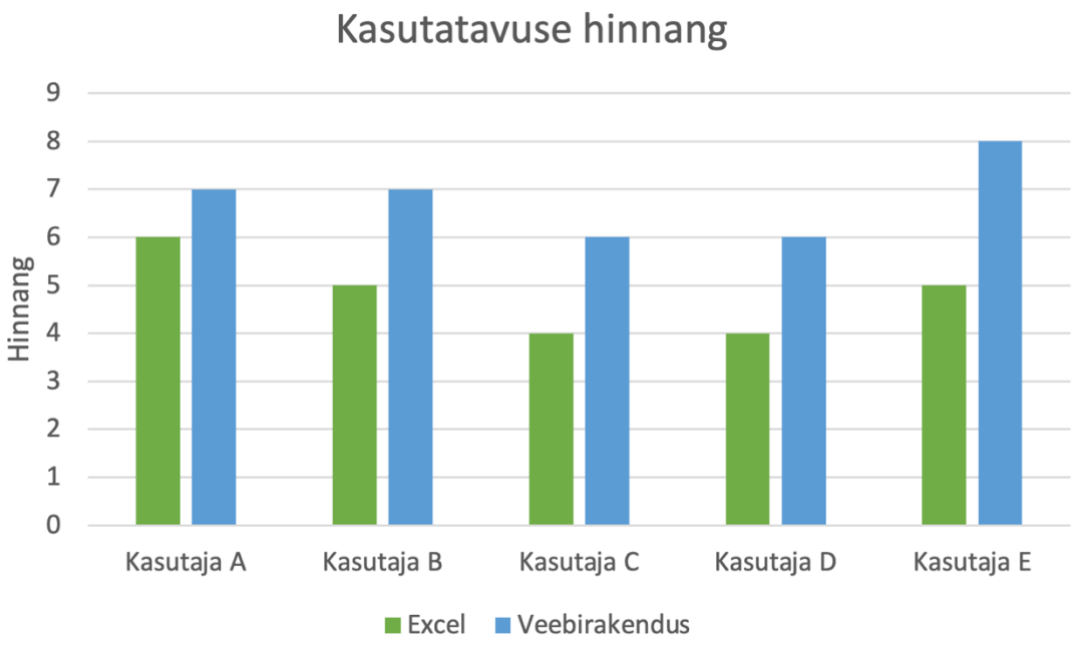
Tulenevalt asjaolust, et tabelarvutustarkvaras koostatud dokument on Eesti investori jaoks ainus seatud tingimustele vastav lahendus, kuhu kasutaja saab investeeringutega seotud informatsiooni kokku koondada, on lõputöö käigus valminud veebirakenduse prototüüpi võrreldud just selle lahendusega. Peamiseks võrdluse kriteeriumiks on investeeringute keskkonda lisamiseks kulunud aeg. Kõigil testijatel paluti nii tabelarvutustarkvara dokumenti kui ka lõputöö käigus valminud lahendusse lisada etteantud investeeringud. Samuti paluti testkasutajatel hinnata mõlema lahenduse kasutatavust kümne punkti skaalal. Iga kasutaja sai peale testimist anda lisaks ka vabas vormis tagasisidet. Kokku osales testimisel viis kasutajat, kes on varasemalt investeerimisega kokku puutunud.

Jooniselt 16 on näha, et uue lahendusega kulus investeeringute keskkonda lisamiseks oluliselt vähem aega. Keskmiselt kulus testkasutajal tehingute sisestamiseks veebirakendusse 17,6 minutit ja keskmine ajasääst kasutaja kohta oli 16 minutit. Veebirakenduse puhul andmete sisestamise aega erinevate kasutajate lõikes suhteliselt sarnane. Suurem aja erinevus Exceli dokumendi puhul tuleneb kasutajate erinevast kogemusest tabelarvutustarkvaradega.



Joonis 16. Investeeringute lisamise aeg keskkonda

Joonis 17 iseloomustab, et uue lahenduse kasutatavust hindasid kasutajad paremaks kui tabelarvutustarkvara puhul. Peamisteks põhjustekst tõid testijad välja, et veebirakenduses on kergem orienteeruda ja andmete sisestamiseks ning ülevaate saamiseks kulub vähem aega.



Joonis 17. Uue lahenduse kasutatavuse rahulolu

Kuigi valminud veebirakendus pakub Eesti investorile head lahendust investeeringutega seotud informatsiooni koondamiseks ühte keskkonda ning annab ülevaate olulistest näitajatest, ei ole prototüüp veel autori hinnangul piisavalt esinduslik avalikustamiseks. Üheks põhjuseks on asjaolu, et sarnaselt tabelarvutustarkvarale, on täpsete tootluste arvutamiseks vaja andmeid pidevalt uuendada. Selle puuduse tõid välja ka testkasutajad. Oluliste andmete mitte uuendamine toob kaasa ebakorrektse informatsiooni kuvamise. Seda saab tulevikus vältida osade andmete automaatse uuendamisega. Samuti on oluline lisada teavitused sellise informatsiooni kohta, mis kuvatakse ebakorrektselt, et kasutaja oleks teadlik andmete uuendamise vajalikust ja ebaõigete väärtuste kuvamisest.

Lisaks selgus lahenduse väljatöötamise käigus, et investeeringute puhul võib esineda mitmeid erijuhte. Aktsiatega võivad olla seotud erinvad korporatiivsed sündmused,

näiteks tükeldamine (*split*), vastupidine poolitus (*reverse split*) ja pakutavate aktsiate vahetamine mitme ettevõtte ühinemise tõttu (*merge*). Korrektse informatsiooni kuvamiseks peab rakendus sisaldama funktsionaalsuseid neid sündmuseid arvestada.

5.3 Edaspidine tegevus

Bakalaureusetöö käigus valmis lahendus, mis vastab kõigile eelnevalt seatud nõuetele. Samuti näitas testkasutajate peal prototüübi valideerimine, et tegemist on efektiivse lahedusega, mis sai positiivset tagasisidet. Seetõttu on teemaga kasulik edasi tegeleda.

Samas selgus, et veebirakenduse prototüüp ei ole siiski valmis laialdaselt levitamiseks. Veebirakendust hakkab kasutama väikene grupp testkasutajaid, kelle eesmärgiks on tuvastada lahenduses leiduvaid vigasid ja teha ettepanekuid rakenduse paremaks muutmiseks.

Samuti lisatakse järgmises arendusetapis liidestamise võimalus Eesti suuremate pankadega, et kasutaja saaks investeeringutega seotud andmed keskkonda sisestada automaatselt. Veel lisatakse automaatne hindade uuendame nii Balti kui ka USA börsidel kaubeldavate aktsiate ja fondide puhul. Selliselt on mainitud finantsinstrumentide tootluse ja kasumi numbrid alati korrektsed. Kasutaja peab ise käsitsi hindasid uuendama vaid selliste aktsiate ja fondide puhul, mis ei kauple eelnevalt mainitud börsidel või kauplevad börsi väliselt. Samuti jääb käsitsi hindade uuendamise vajadus kinnisvara investeeringute puhul.

Lisaks võetakse kasutusele praktikad varasemalt mainitud peamiste veebirakenduse turvariskide maandamiseks, lisatakse automaattestid, arendatakse finantsinstrumentide tegevusvaldkonna kategoriseerimise võimalus ja arendatakse laenude varaklassiga seotud investeeringute lisamise võimalus portfelli.

6 Kokkuvõte

Käesolevas bakalaureusetöös analüüsiti Eesti investori võimalusi investeeringutega seotud informatsiooni koondamiseks ühte keskkonda, eesmärgil saada ülevaade investeerimisportfellist. See on oluline selleks, et jälgida endale seatud eesmärke ja nendes püsimist. Samuti aitab selline ülevaade finantsriske maandada.

Analüüsi käigus selgus, et Eesti investorite seas populaarsemad varaklassid on aktsiad, kinnisvara ja raha. Samuti selgus, et olemasolevad lahendused on peamiselt fokuseeritud kas kindlale varaklassile või kindla riigi kodanikule. Kindlale varaklassile suunatud lahendused on sobivad tööriistad neile, kes omavad investeeringuid vaid selles varaklassis. Kindla riigi kodanikule suunatud lahedused on Ameerika Ühendriikide põhised, mis tähendab, et keskkonda saab konto luua vaid selle riigi kodanik. Lisaks on nende lahenduste puhul võimalik liidestada ainult Ameerika Ühendriikide pankadega ja aktsia hindade automaatne uuendamine toimus vaid USA börsil kaubeldavate väärtpaberite puhul.

Alternatiivseks lahenduseks investeerimisportfelli jälgimiseks on tabelarvutustarkvara kasutamine. Selle eeliseks on paindlikus luua dokument vastavalt investori vajadustele. Lahendus ei sobi kõigile investoritele, sest dokumendi koostamine nõuab asjatundliku tabelarvutustarkvara kasutamise oskust ja häid finantsteadmisi. Samuti on andmete sisestamine ajakulukas ning tabelid muutuvad ajapikku mahukamaks ja on selle tõttu raskesti hallatavad.

Bakalaureusetöö tulemusena töötati välja veebirakenduse prototüüp, mis vastab seatud funktsionaalsetele ja mittefunktsionaalsetele nõuetele. Rakenduse arendamiseks analüüsiti laialt kasutusel olevaid tehnoloogiaid ning valiti nendest prototüübi arendamiseks sobilikumad. Rakenduse kasutajal on võimalik sisestada aktsiate ja kinnisvaraga seotud investeeringud. Samuti on võimalus lisada rahavaru. Peale andmete sisestamist kuvab rakendus kasutajale tootlused ja jaotused nii varaklasside kui ka üksikute finantsinstrumentide lõikes.

Prototüübi testimisel testkasutajate peal selgus, et lahendus on kasutajasõbralikum ja võimaldab investeeringutega seotud informatsiooni kiiremini sisestada kui tabelarvutustarkvaras koostatud dokument. Samuti aitab valminud lahendus vältida vigasid, mis võivad tekkida tootluse arvutamise valemite koostamisel tabelarvutustarkvaras. Sellest tulenevalt on prototüüp täitnud idee valideerimise eesmärgi edukalt ning tõestanud uue lahenduse väärtust.

Kuigi valminud veebirakendus pakub Eesti investorile head lahendust investeeringutega seotud informatsiooni koondamiseks ühte keskkonda, ning annab ülevaate olulistest näitajatest, ei ole prototüüp autori hinnangul piisavalt esinduslik avalikustamiseks. Peamisteks põhjusteks on asjaolu, et sarnaselt tabelarvutustarkvarale, on täpsete tootluste arvutamiseks vaja andmeid pidevalt uuendada. Samuti selgus lahenduse väljatöötamise käigus, et investeeringute puhul võib esineda mitmeid erijuhte.

Järgmise tegevusena hakkab veebirakendust kasutama väikene grupp testkasutajaid, kelle eesmärgiks on tuvastada lahenduses leiduvaid vigasid ja teha ettepanekuid rakenduse paremaks muutmiseks. Samuti lisatakse liidestamise võimalus Eesti suuremate pankadega, et kasutaja saaks investeeringutega seotud andmed keskkonda sisestada automaatselt. Veel lisatakse automaatne hindade uuendame nii Balti kui ka USA börsidel kaubeldavate aktsiate ja fondide puhul. Lisaks võetakse kasutusele praktikad peamiste veebirakenduse turvariskide maandamiseks, lisatakse automaattestid, arendatakse finantsinstrumentide tegevusvaldkonna kategoriseerimise võimalus ja arendatakse laenude varaklassiga seotud investeeringute lisamise võimalus portfelli.

Kasutatud kirjandus

[1] Investopedia, „Modern Portfolio Theory: What MPT Is and How Investors Use It,“ 2021. [Võrgumaterjal]. Loetud aadressil: https://www.investopedia.com/terms/m/modernportfoliotheory.asp. [Kasutatud 13.02.2023].

[2] W. E. Buffet, „Berkshire Hathaway Inc. Shareholder Letter,“ 2013. [Võrgumaterjal]. Loetud aadressil: https://www.berkshirehathaway.com/letters/2013ltr.pdf. [Kasutatud 27.01.2022].

[3] N. Lioudis, „How to Calculate Your Portfolio's Investment Returns,“ 06.02.2022. [Võrgumaterjal]. Loetud aadressil: https://www.investopedia.com/ask/answers/062215/how-do-i-calculate-my-portfolios-investment-returns-and-performance.asp. [Kasutatud 27.01.2023].

[4] E. Pank, „Krediidiasutuste varade ja kohustuste peamised näitajad,“ 27.03.2023. [Võrgumaterjal]. Loetud aadressil: https://statistika.eestipank.ee/#/et/p/147/r/3610/3359. [Kasutatud 10.02.2023].

[5] B. Friedberg and M. Adams, "Best Investing Apps For Portfolio Management," 03.04.2023. [Online]. Loetud aadressil: https://www.forbes.com/advisor/investing/best-investment-managing-apps/. [Accessed 10.04.2023].

[6] SimilarWeb, „Top Website Ranking,“ 01 02 2023. [Võrgumaterjal]. Loetud aadressil: https://www.similarweb.com/top-websites/news-and-media/. [Kasutatud 10 02 2023].

[7] finance.yahoo.com, „Yahoo Finance,“ [Võrgumaterjal]. Loetud aadressil: https://finance.yahoo.com/. [Kasutatud 10.02.2023].

[8] Sharesight, „Sharesight,“ [Võrgumaterjal]. Loetud aadressil: https://www.sharesight.com. [Kasutatud 17.02.2023].

[9] Empower, „Empower,“ [Võrgumaterjal]. Loetud aadressil: https://www.empower.com. [Kasutatud 19.01.2023].

[10] Mint, „Mint,“ [Võrgumaterjal]. Loetud aadressil: https://mint.intuit.com. [Kasutatud 17.02.2023].

[11] „Morningstar,“ [Võrgumaterjal]. Loetud aadressil: https://www.morningstar.com/portfolio-manager. [Kasutatud 17.02.2023].

[12] Ziggma, „Ziggma,“ [Võrgumaterjal]. Loetud aadressil: https://ziggma.com. [Kasutatud 18.02. 023].

[13] T. Ilves, „Finantsvabaduse grupi 2022a. küsitluse kokkuvõte,“ 17.02.2023. [Võrgumaterjal]. Loetud aadressil: https://taavi.golive.ee/2023/02/17/finantsvabaduse-grupi-2022a-kusitluse-kokkuvote/. [Kasutatud 04 03 2023].

[14] aws.amazon.com, „What Is A Web Application?,“ [Võrgumaterjal]. Loetud aadressil: https://aws.amazon.com/what-is/web-application/. [Kasutatud 14.04.2023].

[15] aws.amazon.com, „Three-tier architecture overview,“ [Võrgumaterjal]. Loetud aadressil: https://docs.aws.amazon.com/whitepapers/latest/serverless-multi-tier-architectures-api-gateway-lambda/three-tier-architecture-overview.html. [Kasutatud 16.04.2023].

[16] www.ibm.com, „What is three-tier architecture?,“ [Võrgumaterjal]. Loetud aadressil: https://www.ibm.com/topics/three-tier-architecture. [Kasutatud 14.04.2023].

[17] G. Hohpe ja B. Woolf, Enterprise Integration Patterns, 2003, p. 36.

[18] Stack Overflow, „Stack Overflow,“ 2022. [Võrgumaterjal]. Loetud aadressil: https://survey.stackoverflow.co/2022/. [Kasutatud 06.03.2023].

[19] M. Foundation, „JavaScript,“ 05 04 2023. [Võrgumaterjal]. Loetud aadressil: https://developer.mozilla.org/en-US/docs/Web/JavaScript. [Kasutatud 13.03.2023].

[20] www.python.org, "Python," [Online]. Loetud aadressil: https://www.python.org/doc/essays/blurb/. [Accessed 10.03.2023].

[21] www.typescriptlang.org, „Typescript,“ [Võrgumaterjal]. Loetud aadressil: https://www.typescriptlang.org. [Kasutatud 10.03.2023].

[22] www.java.com, „What is Java technology and why do I need it?,“ [Võrgumaterjal]. Loetud aadressil: https://www.java.com/en/download/help/whatis_java.html. [Kasutatud 10.03.2023].

[23] Microsoft, „A tour of the C# language,“ 13.02.2023. [Võrgumaterjal]. Loetud aadressil: https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/. [Kasutatud 10.03.2023].

[24] www.w3schools.com, „C++ Introduction,“ [Võrgumaterjal]. Loetud aadressil: https://www.w3schools.com/cpp/cpp_intro.asp. [Kasutatud 10.03.2023].

[25] www.php.ne, „PHP,“ [Võrgumaterjal]. Loetud aadressil: https://www.php.net/manual/en/intro-whatis.php. [Kasutatud 10.03.2023].

[26] B. Semah, „Low-Level vs High-Level Language,“ 21.03.2023. [Võrgumaterjal]. Loetud aadressil: https://hackr.io/blog/best-programming-languages-to-learn. [Kasutatud 10.03.2023].

[27] spring.io, „Why Spring?,“ [Võrgumaterjal]. Loetud aadressil: https://spring.io/why-spring. [Kasutatud 10.03.2023].

[28] expressjs.com, „expressjs.com,“ [Võrgumaterjal]. Loetud aadressil: https://expressjs.com. [Kasutatud 10.03.2023].

[29] Microsoft, „.NET,“ [Võrgumaterjal]. Loetud aadressil: https://dotnet.microsoft.com/en-us/. [Kasutatud 10.03.2023].

[30] React, „React,“ [Võrgumaterjal]. Loetud aadressil: https://react.dev. [Kasutatud 10.03.2023].

[31] angular.io, „What is Angular?,“ [Võrgumaterjal]. Loetud aadressil: https://angular.io/guide/what-is-angular. [Kasutatud 10.03.2023].

[32] vuejs.org, „Vue.js,“ [Võrgumaterjal]. Loetud aadressil: https://vuejs.org. [Kasutatud 11.03.2023].

[33] A. School, „React, Vue or Angular,“ [Võrgumaterjal]. Loetud aadressil: https://www.alidaschool.com/blog/react-vue-or-angular-the-best-javascript-framework-to-learn-to-get-a-front-end-job. [Kasutatud 10.03.2023].

[34] E. Navathe, Fundamentals of Database Systems, 2015, pp. 60-63.

[35] M. Drake, „An Introduction to Databases,“ 20.07.2022. [Võrgumaterjal]. Loetud aadressil: https://www.digitalocean.com/community/conceptual-articles/an-introduction-to-databases. [Kasutatud 01.03.2023].

[36] M. Drake, „A Comparison Of Relational Database Management Systems,“ 09.03.2022. [Võrgumaterjal]. Loetud aadressil: https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-management-systems. [Kasutatud 01.03.2023].

[37] www.postgresql.org/, „What is PostgreSQL?,“ [Võrgumaterjal]. Loetud aadressil: https://www.postgresql.org/about/. [Kasutatud 01.03.2023].

[38] PostgreSQL, „JSON Types,“ [Võrgumaterjal]. Loetud aadressil: https://www.postgresql.org/docs/current/datatype-json.html. [Kasutatud 17.03.2023].

[39] sqlite.org, „What Is SQLite?,“ [Võrgumaterjal]. Loetud aadressil: https://sqlite.org/index.html. [Kasutatud 01.03.2023].

[40] git-scm.com, „Branching and Merging,“ [Võrgumaterjal]. Loetud aadressil: https://git-scm.com/about. [Kasutatud 05.04.2023].

[41] Atlassian, „Bitbucket,“ [Võrgumaterjal]. Loetud aadressil: https://www.atlassian.com/git/tutorials/using-branches. [Kasutatud 05.04.2023].

[42] GitHub, „GitHub’s products,“ [Võrgumaterjal]. Loetud aadressil: https://docs.github.com/en/get-started/learning-about-github/githubs-products. [Kasutatud 05.04.2023].

[43] K. Karin, "What is GitLab and How to Use It?," 16.01.2023. [Online]. Loetud aadressil: https://www.simplilearn.com/tutorials/git-tutorial/what-is-gitlab. [Kasutatud 05.04.2023].

[44] Bitbucket, „A brief overview of Bitbucket,“ [Võrgumaterjal]. Loetud aadressil: https://bitbucket.org/product/guides/getting-started/overview#a-brief-overview-of-bitbucket. [Kasutatud 05 04. 023].

[45] Atlassian, „Jira Software,“ [Võrgumaterjal]. Loetud aadressil: https://www.atlassian.com/software/jira?&aceid=&adposition=&adgroup=140479881486&campaign=18442480203&creative=639487383319&device=c&keyword=jira&matchtype=e&network=g&placement=&ds_kids=p73335832032&ds_e=GOOGLE&ds_eid=700000001558501&ds_e1=GOOGLE&gclid=C. [Kasutatud 06.04.2023].

[46] C. Cairns ja D. Somerfield, „The Basics of Web Application Security,“ 01.05.2017. [Võrgumaterjal]. Loetud aadressil: https://martinfowler.com/articles/web-security-basics.html. [Kasutatud 12.04.2023].

[47] Postman, „About Postman,“ [Võrgumaterjal]. Loetud aadressil: https://www.postman.com/company/about-postman/. [Kasutatud 07.04. 023].

[48] Verabelo, „Verabelo,“ [Võrgumaterjal]. Loetud aadressil: https://vertabelo.com. [Kasutatud 24.03.2023].

[49] Spring, „Spring Boot,“ [Võrgumaterjal]. Loetud aadressil: https://spring.io/projects/spring-boot#overview. [Kasutatud 26.03.2023].

[50] Codeacademy Team, „What is REST?,“ [Võrgumaterjal]. Loetud aadressil: https://www.codecademy.com/article/what-is-rest. [Kasutatud 06.03. 023].

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

Mina, Henri Keerutaja

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Veebirakendus eraisiku investeeringute halduseks“, mille juhendaja on Lembit Viilup
   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.

15.05.2023

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.