TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Rasmus Janson 185676IADB

Raamatute otsimise ja hinnavõrdluse veebirakenduse arendus

Bakalaureusetöö

Juhendaja: Kersti Antoi
Teadusmagister

Tallinn 2022

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Rasmus Janson

25.04.2022

Annotatsioon

Käesoleva bakalaureusetöö eesmärgiks on luua töötav veebirakendus BookSearch, mis otsib raamatuid Eesti viiest suurimast raamatupoest.

Arendusprotsessi käigus luuakse veebirakendus, mis võimaldab kasutajatel otsida raamatuid otsingutermini järgi. Leitud tooted kuvatakse ühel veebilehekülgel, et kasutaja saaks raamatute hindu võrrelda ja leida odavaima hinnaga raamatu. Kui toodet parasjagu poes müügil ei ole, saab kasutaja selle lisada soovinimekirja ja saada selle poodi ilmumisel teavitusi. Arenduse loogika on peamiselt jaotatud kolme ossa: veebiteenuse ja klientrakenduse ehitamine ning nende paigaldamine serverisse.

Lisaks sisaldab lõputöö kasutatud tehnoloogiate kirjeldusi ja nende seast kõige optimaalsema valikut, süsteemi arhitektuuri kirjeldust ning automaatteste. Arendusprotsessi tulemuseks on töötav veebirakendus.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 47 leheküljel, 6 peatükki, 11 joonist, 2 tabelit.

Abstract

Development of Book Search and Price Comparison Web Application

The aim of current thesis is to create a working web application BookSearch, which searches for books in the five largest bookstores in Estonia.

During the development process, a web application is created that is available to users to search for books by search term. The found products are displayed on one website so that the user can compare the prices of books and find the book with the cheapest price. If the product is not currently available in the store, the user can add it to the wish list and receive notifications when product appears in store. The development logic is mainly divided into three parts: building a web service and a client application and installing them on a server.

In addition, the thesis includes a description of the technologies used, including the most optimal selection, a description of the system architecture, and automated tests. The results of the development process are a working web application.

The thesis is in Estonian and contains 47 pages of text, 6 chapters, 11 figures, 2 tables.

Lühendite ja mõistete sõnastik

API	*Application Programming Interface* – rakendusliides (protokollide ja tööriistade komplekt rakendustarkvara ehitamiseks)
CDK	*Component Dev Kit* – kogum primitiividest, mõeldud Angulari komponentide loomiseks
CLI	*Command-line interface* – käsurea liides
CSS	*Cascading Style Sheets* – veebilehtede küljendamisel kasutatav märgenduskeel
DOM	*Document Object Model* – dokumendi objektimudel on platvormist ja keelest sõltumatu XML, XHTML ja HTML dokumentidega suhtlemise liides
DNS	*Domain Name System* – domeeninimede süsteem
DPI	*Dots per inch*, punkti tolli kohta
FaaS	*Function as a Service* - Pilvepõhine teenus, sarnane PaaS-ile, ning automaatselt skaleeruv
HTML keel	*Hyper Text Markup Language* – veebilehe märgendamise
IA	Arvutisüsteemide instituut
IaaS	*Infrastructure as a Service* - pilvepõhine teenus, mis haldab infrastruktuurilisi elemente
IDE	*Integrated Development Environment* – koodikirjutamiskeskkond
ISBN	*International Standard Book Number* – rahvusvaheline raamatu standardnumber
JSON	*Javascript Object Notation* – andmevahetusvorming
JWT	JSON *Web Token* - internetistandard andmete loomiseks koos digitaalallkirja ja/või krüpteerimisega
MPA	*Multi Page Application* – mitmelehe-rakendus
PaaS	*Platform as a Service* - pilvepõhine teenus, mis haldab kõike, mis ei ole seotud rakenduse ning äriloogikaga
RDBMS	*Relational Database Management* System – relatsioonilise andmebaasi haldussüsteem

RDBMS	*Relational Database Management System* - relatsioonandmebaasi haldussüsteem
REST	*Representational State Transfer* - tarkvaraarhitektuuri laad, mis seab veebirakenduse loomisele kindlad piirid
SaaS	*Software as a Service* - pilvepõhine teenus, mille puhul haldab kõiki aspekte teenusepakkuja
SPA	*Single Page Application* – üheleherakendus
Stack Overflow	küsimuste ja vastuste veebisait programmeerijatele.
URL	*Uniform Resource Locator* - internetiaadress
XML	*Extensible Markup Language* – dokumentide laiendatav märgendamiskeel
XPath	Väljenduskeel, mis on loodud toetama XML-dokumentide päringuid või teisendamist. Saab kasutada navigeerimiseks XML dokumendis

Sisukord

1 Sissejuhatus 11
1.1 Ülevaade probleemist 12
1.2 Metoodika 12
2 Eksisteerivad lahendused ja skoop 14
2.1 BookFinder 14
2.2 AddAll 14
2.3 Loodava lahenduse skoop 15
3 Loodava veebirakenduse analüüs 17
3.1 Nõuete määramine 17
3.1.1 Funktsionaalsed nõuded 17
3.1.2 Mittefunktsionaalsed nõuded 18
3.1.3 Tulevikus loodavad funktsionaalsused 18
3.2 Tehnoloogia valik 18
3.2.1 Teenusepoolse programmeerimiskeele valik 19
3.2.2 Teenusepoolsed raamistikud 21
3.2.3 Kliendipoolne tehnoloogia (frontend) 22
3.3 Andmebaasi valik 23
3.4 Arenduskeskkonna valik 24
3.5 Veebirakenduse haldus 25
3.6 Arhitektuur 27
3.7 Veebirakenduse disain 28
3.7.1 Kasutajakogemuse disain 28
3.7.2 Andmebaasi disain 30
3.8 Analüüsi kokkuvõte 31
4 Veebirakenduse arendus 33
4.1 Veebiteenuse-poolne lahendus 33
4.1.1 Spring Boot rakenduse arhitektuur 33
4.1.2 Veebikaabitsa teegid 34
4.1.3 REST API 37
4.1.4 Veebiteenuse turvalisus 39
4.1.5 Konfiguratsioon 40

4.2 Klientrakenduse lahendus ... 40
4.2.1 Komponentide planeerimine ... 40
4.2.2 Angular – *Single Page Application* (SPA) ... 41
4.2.3 Angular rakenduse struktuur ... 42
4.2.4 Angular Material ... 43
4.2.5 Angular rakenduse serveerimine brauserile ... 43
4.2.6 Klientrakenduse turvalisus ... 43
4.3 Testimine ... 44
4.3.1 Kliendipoolsed automaattestid ... 44
4.4 Pidev integratsioon ... 45
4.4.1 Heroku ühendus GitHubiga ... 46
5 Hinnang loodud veebirakendusele ... 47
5.1 Saavutatud kasutatavus ... 47
5.2 Võimalused edasiarenduseks ... 49
6 Kokkuvõte ... 50
Kasutatud kirjandus ... 51
Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks ... 54
Lisa 2 – Nutiseadme brauseri vaated ... 55
Lisa 3 – Esilehe ja progressiooni laadimisrõnga näitamine ... 58

Jooniste loetelu

Joonis 1. Veebirakenduse arhitektuur 27
Joonis 2. Esilehe kasutajakogemuse diagramm 29
Joonis 3. Soovinimekirja kasutajakogemuse diagramm 30
Joonis 4. Soovinimekirja olemid 31
Joonis 5. Spring Initializr kasutajaliides 34
Joonis 6. Veebikaapimine Seleniumiga 36
Joonis 7. Veebikaapimine HtmlUnit-iga 37
Joonis 8. Otsingutulemuste lehe komponentide paigutus 41
Joonis 9. Angulari automaattestid, kontrollimaks komponendi näitamist õige parameetri korral 45
Joonis 10. Otsingutulemuste kuvamine 48
Joonis 11. Soovinimekirja vaade 48

Tabelite loetelu

Tabel 1. Programmeerimiskeelte võrdlus 21
Tabel 2. Veebiteenuse API päringud 39

1 Sissejuhatus

Tänapäeval tellitakse aina rohkem asju veebikeskkondadest, mis vähendab tarvidust minna poodi sisseoste tegema. Sama saab öelda ka raamatute tellimise kohta. Kõigil Eesti suurimatel raamatupoodidel on oma veebilehekülg, kust saab raamatuid osta. Erinevalt raamatupoodides kohapeal käimisest saab kasutaja veebist vaadata, millises poes ja millise hinnaga raamat on. Mugavat viisi Eesti raamatupoed selleks ei paku. Ainuke võimalus raamatute hinnavõrdluseks on mitu erinevat veebilehte lahti võtta ja siis hindu vaadata, tingimusel, et antud raamatupoodides otsitav raamat üldse parasjagu müügil on.

Käesoleva lõputöö raames luuakse veebirakendus raamatute otsimiseks ja hinnavõrdluseks. Eesmärgiks on töötav rakendus, mis kogub infot Eesti suurimatest veebis olevatest raamatupoodidest, ning kuvab kogutud info ühel leheküljel. Loodav lahendus muudaks raamatute otsimise ja ostmise veebist mugavamaks ning odavamaks.

Autor otsustas keskenduda ainult Eesti raamatupoodidele, kuna välismaalt tellimine võib võtta nädalaid. Eestis, kui raamat on laos, võtab tellimine üldjuhul 3-5 tööpäeva. Selle poolest erinebki lõputöös käsitletav rakendus juba eksisteerivatest lahendustest, mis on välismaised ja ei paku kiiret raamatute tellimist Eestisse. Kuigi lisaks raamatutele saaks mõnest raamatupoest osta ka muid asju, nagu lauamänge või arvutimänge, siis antud lõputöö piiritleb ainult raamatutega. Mängude või muude tarvete ostmiseks on teised poed ning enamasti külastatakse raamatupoode raamatute pärast.

1.1 Ülevaade probleemist

Eestis on viis suuremat raamatupoodi. Nendeks on: Apollo, Rahva Raamat, Krisostomus, Raamatukoi ja Varrak. Tänapäeval on igal suuremal raamatupoel oma veebilehekülg, ning ka need Eesti raamatupoed pole erandiks.

Rohke veebipoodide arv tähendab suuremat valikuvõimalust kliendile. Suurem valikuvõimalus paneb aga mõtlema, kust oleks kõige mõttekam raamatuid tellida. Kõik raamatupoed müüvad raamatuid erineva hinnaga, mis üldjuhul ei erinegi tavaliselt konkurendi hinnast, kuid selleks, et kliente juurde meelitada tehakse toodetele ka soodustusi. Kui võrrelda mõne raamatupoe raamatu tavahinda teise poe sama raamatu soodushinnaga, siis on hinnavahe juba märgatavam. Tänapäevases tehnoloogia maailmas ei ole muidugi hinnavõrdluse sooritamises midagi rasket. Tuleb lihtsalt veebibrauseris mitu akent korraga lahti võtta ja vaadata, millises poes on raamatul soodsam hind. Sellest hoolimata võiks see protsess olla kiirem ja mugavam.

Olenemata hinnavahest on tõenäoliselt inimestel aja jooksul välja kujunenud oma lemmikraamatupoed. Seetõttu ei pruugi hinnavahe neid nii väga kõigutada. Sellegipoolest on nagu mündil ka sellel probleemil teine pool. Nimelt kui raamatut parasjagu otsitavas raamatupoes müügil ei ole, siis esimene instinkt on ikka vaatama minna teisest raamatupoest.

Raha saaks kokku hoida, võrreldes kõikide raamatupoodide raamatu hindu korraga, ning valides nende seast soodsaima. Sama saab öelda aja kokkuhoiu kohta, kui ei pea korraga mitme erineva raamatupoe lehekülgi korraga lahti võtma ja sealt otsima hakkama.

1.2 Metoodika

Antud lõputöö raames analüüsitakse esmalt eksisteerivaid lahendusi, selgitatakse, miks need süsteemid ei sobi lahenduseks ja pakutakse välja probleemile sobiv IT lahendus, mis jääks lõputöö skoopi, kuid võimaldaks edasiarendamist tulevikus.

Analüüsi käigus tuuakse välja süsteemi funktsionaalsed ja mittefunktsionaalsed nõuded. Samuti analüüsitakse erinevaid tehnilisi lahendusi ning põhjendatakse teenusepoolsete ja kliendipoolsete tehnoloogiate valikut ning ka arendus- ning haldusvahendite valikut.

Vastavalt nõuetele luuakse rakenduse arhitektuur ning kirjeldatakse rakenduse kasutajakogemuse disain.

Rakenduse arenduse protsessi on kirjeldatud nii teenuse- kui ka kasutajaliidese poolt. Kirjeldatakse analüüsi osas valitud raamistike arhitektuuri, konfiguratsiooni ja teisi teenuse- kui ka kasutajaliidese poolega kaasa tulevaid lisavahendeid. Lõpuosas on kirjeldatud ka rakenduse kliendipoolset testimist ja pidevat integratsiooni. Lahenduse valmimisel sõnastatakse tulemus, hinnatakse, kuivõrd sobilik on lõpplahendus ja kirjeldatakse edasiarendus võimalusi.

2 Eksisteerivad lahendused ja skoop

2.1 BookFinder

BookFinder.com on raamatute otsingumootor, mis loodi juba 1997. aastal. BookFinder otsib rohkem kui 150 miljoni müügisoleva raamatu seast [1]. Müügisolevate raamatute alla kuuluvad nii uued, kasutatud kui ka õpikud. BookFinderi eesmärk on sama, mis lõputöös kirjeldatud – säästa inimeste aega ja raha. Sarnaselt lõputööle saab seejärel otsitud raamatu osta otse algselt müüjalt ilma juurdehindluseta.

BookFinderi lehel saab raamatuid otsida nii autori, pealkirja kui ka ISBN (rahvusvaheline raamatu standardnumber) järgi. Samuti saab otsida võtmesõnade, hinnavahemiku, ilmumisaasta ja köite järgi. Lisaks saab otsingusse lisada ka sihtkohariigi. BookFinder keskendub peamiselt hollandi-, inglise-, prantsuse-, saksa-, itaalia- ja hispaaniakeelsetele raamatutele.

2.2 AddAll

AddAll on nii mõtte kui ka lahenduse poolest väga sarnane BookFinder-ile. Põhierinevuseks on saatmise puhul väiksem sihtkohtade arv. Näiteks erinevalt BookFinderist ei paku AddAll saatmist Eestisse. Sihtkohta valides näitab lehekülg sellega kaasnevat postikulu ja oodatavat saatmisaega. Sellegipoolest, kui otsida raamatut ja pakutavate poodide seas on näiteks Amazon, siis saab ikkagi raamatu Eestisse tellida.

Hoolimata AddAlli ja BookFinderi sarnasustest, pakub AddAll veel enne raamatu ostmist võimaluse vaadata selle arvustusi. Arvustajate seas on tuntud autorid, kuid ka AddAlli enda toimetus, mis loob enamasti raamatute edetabeleid. Edetabelid koosnevad näiteks kümnest parimaist fantaasiaraamatust jms. Sellegipoolest pole arvustuste vaatamine väga kasulik, kuna otsitava raamatu kohta suure tõenäosusega eraldi arvustust pole. Mõne väga tuntud raamatu arvustuse võib leida nende edetabelitest [2].

2.3 Loodava lahenduse skoop

Eksisteerivate lahenduste suurim puudus on see, et kuigi tegemist on rakendustega, mis otsivad raamatuid tuhandetelt raamatumüüjatelt üle maailma ja miljonite raamatute seast, ei ole nende seas Eesti raamatumüüjaid. Sellega kaasneb mitu probleemi:

- Ükski välismaine raamatumüüja ei paku raamatuid eesti keeles.
- Tarneaeg – välismaalt tellides võib raamatu kohalejõudmiseks minna nädalaid ja isegi kuid. Eestist tellides, kui raamat on laos, kulub tavaliselt 3-5 tööpäeva. Lisaks on alati võimalus raamatupoodi kohapeale minna, kui see on läheduses.
- Mõned eksisteerivad lahendused ei paku üldse tarnimist Eestisse.

Lahendust, mis võrdleks Eesti poodide raamatu hindu, ei eksisteerigi. Hinnavõrdluseks ainuke moodus on võtta korraga lahti erinevate poodide veebilehed ning sealt raamatud välja otsida ja siis hindu vaadata. Sellegipoolest on tõenäoliselt inimestel oma eelistatud raamatupood, kust nad alati raamatuid ostavad, ning hinnavõrdlus neid nii väga ei huvitagi. Siinkohal võib aga probleemiks osutuda, et selles raamatupoes ei pruugi antud toodet parasjagu olla ning lõpuks peab ikka kuskilt mujalt soovitud raamatut otsima.

Et raamatute otsimine ja tellimine oleks kiire ning kasutajal oleks võimalik sooritada hinnavõrdlust, pakub antud töö autor luua veebirakendus, mis keskendub Eesti raamatupoodidest raamatute otsimisele ja otsingu tulemuste kuvamisele ühel veebilehel. Loodaval rakendusel saab olema otsinguväli, kuhu saab sisestada otsingutermini. Sisestatud tekst lisatakse kõikide teiste raamatupoodide aadressiribale parameetriks ning saadud tulemused kogutakse kuvamiseks kokku. Kui ühtegi tulemust ei leita või kasutaja soovib just temale meeldivast raamatupoest raamatut osta, aga seda parasjagu müügis ei ole, saab kasutaja raamatu lisada oma soovinimekirja. Vastavalt soovinimekirjas olevatele kirjetele käib veebikaabits märgitud raamatupoed kord päevas läbi. Kui soovinimekirjas olevale kirjele leitakse vaste, saadetakse kasutajale teavitus.

Kuigi mõnest poest saaks lisaks raamatutele otsida ka muid asju, näiteks lauamänge või arvutimänge, piirdub lahendus ainult raamatute otsimise ja kuvamisega. Samuti jääb lahenduse skoobist välja veebikaabitsate monitooring, mis võimaldaks rakenduse

administraatoril saada teavitusi veebikaabitsate katkiminemise kohta (juhul kui HTML peaks muutuma).

Lõputöö kontekstis piirdutakse vaid brauseripõhise lahendusega. Eelkõige seetõttu, et raamatupoodide veebilehekülјed on brauseripõhised ning brauseripõhisel lahendusel on parim võimalus jõuda kõige laiema publikuni. Sellegipoolest peab arvestama, et tänapäeval kasutatakse mobiilseid seadmeid juba pea rohkem kui arvuteid ning seetõttu peaks pakutav lahendus olema kasutatav ka nutitelefonis.

Antud lõputöö raames arendatava lahenduse eesmärgiks pole raha teenimine, vaid veebist raamatute otsimise ja ostmise mugavamaks ning odavamaks muutmine.

3 Loodava veebirakenduse analüüs

3.1 Nõuete määramine

Nõuete määramisel võetakse arvesse loodava lahenduse skoop. Rakenduses on ainult üks kasutajagrupp (edaspidi tavakasutaja). Traditsiooniline administraatori roll puudub. See oleks vajalik ehk ainult tavakasutajate kustutamiseks, mis on aga teisejärgulisem ülesanne ja jääb edasiarenduse võimaluseks.

Nõuded põhinevad kasutajalugudel ning jagunevad funktsionaalseteks- ja mittefunktsionaalseteks nõueteks. Nõudeid, mida soovitakse edasi arendada peale lõputöö valmimist, on käsitletud töö lõpus.

3.1.1 Funktsionaalsed nõuded

Tavakasutaja-põhised funktsionaalsed nõuded:

- Raamatute otsimine otsingutermini järgi
- Otsingu tulemuste seas peab kuvama raamatu pealkirja, pilti, hinda, soodushinda, autorit ja linki poe veebilehele
- Otsingu tulemuste puudumisel näidata selle kohta vastavat teadet
- Raamatute otsimine, mis võivad olla nii eesti- kui ka inglise keelsed
- Veebilehel registreerumine
- Registreeritud kontoga sisse logimine
- Välja logimine
- Kirjete lisamine soovinimekirja
- Kirjete kustutamine soovinimekirjast
- Teate saamine, kui soovinimekirjas olev toode poodi ilmub

3.1.2 Mittefunktsionaalsed nõuded

- Nutiseadme brauseri vaade
- Süsteem kuvab maksimaalselt kümme tulemust poe kohta
- Soovinimekirjas olevaid tooteid otsitakse märgitud poodidest kord päevas
- Kui soovinimekirjas olevale kirjele leidub vaste, saadetakse kasutajale teavitus meilile

3.1.3 Tulevikus loodavad funktsionaalsused

Järgnevalt kirjeldatud funktsionaalsused jäävad lõputöö praegusest skoobist välja. Siia kuulub veebikaabitsate monitooring, mis tähendab, et kui veebikaabits ei suuda veebilehtedelt mingitel põhjustel enam infot lugeda, saadetakse selle kohta detailne teave administraatorile. Kiire teabe edastamine aitaks kiiremas korras rakendus tagasi töökorda saada. Peamiseks põhjuseks, miks veebikaabits ei saaks enam veebilehelt infot kätte, on veebilehe kujunduse muutumine. Selliseid juhtumeid üldjuhul väga tihti ette ei tohiks tulla, kuid sellegipoolest tuleks selle võimalusega arvestada.
Teiseks funktsionaalsuseks, mille saaks tulevikus lisada, on detailsem toodete otsing. Antud lõputöö raames keskendutakse ainult raamatute ja e-raamatute otsingule, kuid mõned raamatupoed müüvad ka muid asju peale raamatute. Antud lõputöö skoopi arvestades arvab autor, et muude toodete otsimise lisamine rikuks loodava rakenduse mõtte. Sellegipoolest on see funktsionaalsus, mida saaks antud rakendusele tulevikus juurde lisada.

3.2 Tehnoloogia valik

Antud lõputöö eesmärgiks on luua ühelehe veebirakendus. Veebirakendus on rakendusprogramm, mis töötab kaugserveri peal ja mida edastatakse interneti ning brauseri liidese kaudu. Rakenduse toimimiseks vajab see veebiserverit, rakendusserverit ja enamikel juhtudel ka andmebaasi. Veebiserverid edastavad kliendi poolt tulnud päringud rakendusserverile ja rakendusserver täidab need päringud. Kui rakendusse on kaasatud ka andmebaas, siis rakendusserver suhtleb andmebaasiga lugedes või muutes seal infot. [3]

Loodaval veebirakendusel on vaja kõike kolme ülalmainitud tehnoloogiat. Kuna tänapäeval on veebirakenduse arendustehnoloogiaid väga palju (erinevad programmeerimiskeeled, programmeerimiskeelte raamistikud, andmebaasid), tuleb nende kõigi seast valida sobivaimad. Valikute tegemisel on arvestatud lõputöö nõuete ja skoobiga.

- Paindlikkus ja skaleeritavus. Eelnevalt sai mainitud, et tänapäeval on palju erinevaid tehnoloogiaid. Pidevalt tuleb juurde mõni uus, kuid hääbuvad mõned teised. Seega tasub kursis olla viimaste trendidega ja vaadata, et valitud tehnoloogiatel oleks vähemalt mõneaastane tugi. Kui pikaajalist tuge tehnoloogiatele ei pakuta, on neid riskantne rakenduse ehitamisel kasutada. [4]

- Turvalisus – kuna antud rakenduses saab kasutajaid luua ja kasutajad saavad lisada ainult neile nähtavaid kirjeid soovinimekirja, tuleb arvestada ka turvalisuse aspektiga.

- Dokumentatsioon – iga rakenduse arendamisel tuleb ette erinevaid probleeme, sellepärast tuleks valida tehnoloogiad, millel on piisav dokumentatsioon ning kogukond. [4]

- Kogemus – lõputöö tegemiseks on piiratud aeg, sellepärast tuleb tehnoloogiate valimisel lähtuda ka autori oskustest ja kogemusest nendega. Võib ette tulla, et mõnda teist tehnoloogiat oleks mingis aspektis mõistlikum kasutada, kuid kui pole sellega varem kokku puutunud, pikendaks see kindlasti lõputöö valmimise aega.

3.2.1 Teenusepoolse programmeerimiskeele valik

Stack Overflow[1] viis 2021. aasta mais arendajate seas läbi küsitluse, milles osales üle 80 000 arendaja. Küsitlusest tuli välja, et kõige populaarsemad teenusepoolsed programmeerimiskeeled on Java, C#, PHP, Python ja Node.js (pole päris programmeerimiskeel, vaid programmeerimiskeskkond, mis laseb arendajatel kasutada JavaScripti teenusepoolse programmeerimiskeelena). [5] Järgnevalt on toodud välja nende keelte lühikirjeldused:

[1] Veebisait

- Java - tüübi- ja klassipõhine objektorienteeritud keel. Tavaliselt kompileeritakse baitkoodi käsukomplekti ja binaarvormingusse, mis on määratletud Java virtuaalmasina spetsifikatsioonis. [6]

- C# - objektorienteeritud ja tüübikindel programmeerimiskeel. Võimaldab arendajatel luua erinevaid tüüpe turvalisi ja töökindlaid rakendusi, mis töötavad .NET-is. [7]

- PHP – avatud lähtekoodiga serveripoolne skriptikeel, mida sageli kasutatakse veebiarenduseks. Üldkasutatav keel, mida saab kasutada paljude arenduste, sealhulgas graafiliste kasutajaliideste tegemiseks. [8]

- Python – avatud lähtekoodiga tõlgendatud, objektorienteeritud kõrgetasemeline programmeerimiskeel, millel on dünaamiline semantika. Selle kõrgetasemelised sisseehitatud andmestruktuurid koos dünaamilise tippimise ja sidumisega, muudavad selle väga atraktiivseks nii rakenduste kiireks arendamiseks kui ka kasutamiseks skripti- või liimkeelena olemasolevate komponentide ühendamiseks. [9]

- Node.js – avatud lähtekoodiga ja platvormist sõltumatu JavaScripti käivituskeskkond. Node.js on ehitatud Google Chrome JavaScripti V8 mootori peale. [10]

Arvestades seda, et kõik väljatoodud programmeerimiskeeled on paindlikud, turvalised ja väga hea dokumentatsiooniga, siis valiku tegemisel lähtutakse autori eelnevast kogemusest ja õppimiskeerukusest. Järgnevalt on välja toodud teenusepoolsete keelte võrdlus (Tabel 1).

Tabel 1. Programmeerimiskeelte võrdlus

Keel	Kogemus	Õppimiskeerukus
Java	Hea	Madal [11]
C#	Hea	Madal [12]
PHP	Nõrk	Keskmine [13]
Python	Nõrk	Madal [14]
Node.js	Rahuldav	Keskmine [15]

Võttes arvesse, et autoril on juba hea kogemus Java ja C#, siis piirdutakse edasiselt ainult nende keeltega, kuna teistega pole piisavalt kogemust ja juurdeõppimine võtaks liiga palju aega.
Java ja C# on süntaksi mõttes väga sarnased. Ainukeseks Java eeliseks autori arvates on selle suurem varajane kogemus.

3.2.2 Teenusepoolsed raamistikud

Iga suurema teenusepoolse keelega käib kaasas ka vastav raamistik. Raamistik on tööriistade ja moodulite kogum, mis aitab rakenduse arhitektuuri üles ehitada. Põhiliselt aitavad raamistikud arendajate elu lihtsamaks muuta. Nende peamised eelised on turvalisuse tagamine, aja kokkuhoid, integratsioonid ja paindlikus. [16]

Kuna teenusepoolse keele valikutesse jäid Java ja C#, siis järgnevalt tuuakse välja nende keelte raamistikud.

Microsofti C# üheks tuntuimaks ja enim kasutatavaks raamistikuks on .NET Framework. Kuigi äsja mainitud raamistik on C# maailmas veel kõige laialdasemalt kasutatud, on ka veel teine raamistik. Stack Overflow 2021 arendajate uuringust tuli välja, et arendajate lemmik raamistikuks on hoopis .NET Core/.NET 5. [5] Kuna .NET Core on võrreldes .NET Framework-iga uuem ja üha rohkem populaarsust koguv ning .NET Frameworki edasiarendus, siis valikusse otsustati võtta .NET Core.

Samamoodi nagu C# maailmas, on ka Javas valik erinevate raamistike vahel mitmekesine. Java suurimateks raamistikeks on Spring, Struts ja Hibernate. [17] Siiski paistab nende kolme seast enim välja Springi raamistik, mida nimetatakse raamistike raamistikuks. Põhjuseks on see, et Spring toetab ka teist kahte raamistiku. [18] Kuna

autoril on mainitud raamistikest kõige suurem kogemus Springiga, siis valitakse see raamistik.

Just nagu Java ja C# on süntaksi ja semantika poolest väga sarnased, on ka nende suurimad raamistikud sarnased. Nii Spring kui ka .NET Core on mõlemad väga hea dokumentatsiooni ja laialdase kasutajaskonnaga. Mõlemad on paindlikud ja turvalisuse aspektis võrdväärsed. Samuti on autoril kogemust mõlema keele ja raamistikuga. Arvestades aga lõputöö teemat ja skoopi, otsustati Java ja Springi raamistiku kasuks. Tulenevalt lõputöö teemast on autoril teema ja Java Springiga mõnevõrra suurem kogemus, mis sai kahe keele ja raamistiku vahel määravaks faktoriks.

3.2.3 Kliendipoolne tehnoloogia (frontend)

Kuigi teenusepool on rakenduse töötamise põhitalaks, ei vaataks kasutajad kahte kordagi rakendust, millel pole visuaalset atraktiivset graafilist liidest. [19]

Kliendipool koosneb graafilisest disainist (välimus) ja kasutajakogemusest (lehe funktsionaalsus). [20] Disaini ja kasutajakogemuse moodustavad peamiselt HTML, CSS ja JavaScript. HTML-iga luuakse lehele elemendid, CSS-iga antakse loodud elementidele välimus ja JavaScript lisab funktsionaalsuse. Kuigi veebirakenduse saaks valmis teha ainult neid kolme tehnoloogiat kasutades, siis JavaScripti puhul on mõttekas abiks võtta ka mõni selle raamistik, mis aitaks arenduse teha paindlikumaks ja paremini hallatavamaks.

Järgnevalt on välja toodud populaarsemad raamistikud ja paketid:

- Angular – komponendipõhine arendusplatvorm, mis on üles ehitatud TypeScriptile[1]. Tuleb kaasa suur ja hästi integreeritud teekide kogu. [21] Peamiselt mõeldud suurte rakenduste tegemiseks.

- React – pole päris raamistik, aga JavaScripti paketid, mis on mõeldud kasutajaliideste ehitamiseks. [22] Enim kasutatav kliendipoolne raamistik. [5]

- Vue.js – sarnaselt Angularile on JavaScripti raamistik, mõeldud kasutajaliideste ehitamiseks. [23]

[1] JavaScriptil põhinev tüübipõhine programmeerimiskeel, mis kompileerib JavaScriptiks

Kuna tegemist on suurte ja laialdaselt kasutatavate tehnoloogiatega, on neil kõigil hea dokumentatsioon, skaleeruvus ja turvalisus. Kõige populaarsem kolme seast on React. Sellegipoolest otsustas autor Angulari kasuks. Kuigi React on populaarsem ja väiksema mahulise projekti jaoks parem, on autoril Angulariga rohkem kogemust ja ta kasutab seda paralleelselt igapäeva töös.

3.3 Andmebaasi valik

Tavalise veebirakenduse kriitilisemaid osi on andmebaas. Seepärast on oluline, et valitaks sobiv andmebaas, mis on rakenduse nõuetele kohane. Andmebaasi saab aga valida üle 300 andmebaasihaldussüsteemi seast. [24] Seega, enne kui saab valida konkreetset andmebaasi, on mõistlik vaadata andmebaasitüüpe. Erinevate andmebaasi tüüpide valik on toodud järgnevalt: relatsioonilised, SQL-ta, pilve, veergudega, laia veergudega, objekt-orienteeritud, võti-väärtus, hierarhilised, dokument-orienteeritud, graaf ning aegrida andmebaasid. [25] Kõige kasutatavamad „andmebaasitüübid“ on relatsioonilised, dokument-orienteeritud, ja võti-väärtus andmebaasid. [26]

Lõputöö raames vaadatakse kõige suurema populaarsuse, kasutajabaasiga ja ülikoolis varasemalt kasutatud andmebaase. Andmebaasi valimisel tuleb arvesse võta ka selle hinda ja süsteemi nõudeid.

2021. aasta andmete põhjal on arendajate seas kõige kasutatavamaks andmebaasiks MySQL. [5] MySQL on relatsiooniline andmebaas, mille põhirõhk on stabiilsusel ja töökindlusel. Samuti kasutatakse seda enim veebiarenduslahenduste jaoks. [27]

Rääkides andmebaasidest ei saa mainimata jäta Oracle andmebaasi. Oracle relatsioonandmebaasi haldussüsteem (*RDBMS*), loodud 1979. aastal, on maailma esimene. [28] Kuigi Oracle on üks populaarsemaid ja suuremaid, on see ka üks kallimaid andmebaase. Samuti kasutatakse Oraclet pigem suurtes ettevõtetes, kus on vaja hallata suurt hulka andmeid. [29]

PostgreSQL, samuti relatsiooniline andmebaas, on sarnane MySQL-ile. Erinevuseks on Postgre vabavaralisus. [28]

MongoDB on oma mitte-SQL andmebaaside valdkonnas üks suurimaid. Seda on sobilik kasutada, kui andmed on poolstruktureeritud (JSON, XML) kujul ja andmebaasi skeemi on vaja tihti muuta. [28]

Redis on võti-väärtus andmebaas. Redise peamiseks eeliseks on kiirus. Seda kasutatakse enamasti, kui on vaja salvestada andmeid ilma keerulise struktuurita. [29]

3.4 Arenduskeskkonna valik

Arenduskeskkonna valikud võib jagada kaheks: koodihaldus ja koodi arenduskeskkond.

Koodihalduskeskkondadest võrreldakse kolme, ühtlasi ka kõige populaarsemat keskkonda, millega on autoril varasemalt olnud ka kokkupuude. Nendeks on Github, GitLab ja Bitbucket.

Github on eelmainitud keskkondadest suurim. Hetkel on sellel üle 73 miljoni kasutaja ja üle 200 miljoni repositooriumi. [24]

GitLab sarnaneb sisult Githubile ja on uusim neist. Hetkel on sellel üle 30 miljoni registreeritud kasutaja. [25]

Bitbucketit omab Atlassian, mis arendab tarkvaratööristu nagu Jira, Trello ja Confluence. Hea integratsiooni olemasolu selliste tööriistadega on olnud Bitbucketi suureks eeliseks. [26]

Koodi arenduskeskkondade valikul tuleb arvesse võtta seda, et need peaksid toetama Javat ja Angulari. Tuntuimad Java arenduskeskkonnad on Intellij ja Eclipse. Suurimaks erinevuseks nende vahel on hind. Kuigi Intellij’l on ka tasuta versioon, omab see vähem funktsionaalsusi. Eclipse on täiesti tasuta. [27] Angulari põhilisteks arenduskeskkondadeks on Visual Studio Code ja WebStorm. Mõlemad keskkonnad on sarnaste funktsionaalsustega. WebStormi üheks eeliseks on parem koodi refaktoreerimine. [28]

3.5 Veebirakenduse haldus

Et kasutajad internetis saaks loodavale veebirakendusele ligi on vaja serverlahendust. Tänapäeval saab valida kahe lahenduse vahel – pilveteenus ja füüsiline serveri omamine.

Füüsilised serverid on spetsiaalselt isiklikuks kasutamiseks mõeldud arvutid. Seda tüüpi serverid on spetsiaalselt konfigureeritavad. Serveri omanikul on privilegeeritud juurdepääs kogu süsteemile, mis tähendab, et serverile saab paigaldada mistahes tarkvara.

Pilveteenused ei paikne kindla füüsilise tarkvara peal, vaid asetsevad mitmes erinevas virtuaalses seadmes erinevates asukohtades. Pilveteenused on mingil määral juba teenusepakkuja poolt ette konfigureeritud. See tähendab, et kasutajal ei ole nii suurt vabadust võrreldes füüsilise serveriga. Omakorda teeb see aga pilveteenuse kasutuselevõtu palju lihtsamaks, kuna palju asju on juba teenusepakkuja poolt ette ära tehtud. Samuti on üheks suurimaks erinevuseks pilve- ja füüsilise serveri vahel hinnakiri. Pilveteenuse puhul sõltub hind kasutajate arvust. Rohkemate kasutajate puhul läheb teenus rohkem maksma. Füüsilise serveri puhul on hind fikseeritud.

Pilveteenused jagunevad omakorda veel nelja kategooriaasse: IaaS, PaaS, SaaS, FaaS. Nagu juba eelnevalt mainitud, on füüsilise serveri puhul omanikul täielik kontroll, siis pilveteenuste puhul on igal kategoorial teatud osas õigusi teenusepakkuja poolt ära võetud.

IaaS ehk *Infrastructure as a Service*[1] pakub baasinfrastruktuuri (virtuaalne masin, arvutivõrk, salvestusruum). Kasutaja peab platvormi ja keskkonna konfigureerima ning seda haldama. Tuntumad näited on Amazon Web Services ja Microsft Azure. [35]

SaaS ehk *Software as a Service*[2], mida nimetatakse vahel ka tellitavaks tarkvaraks. Selle puhul ei vastuta kasutaja ei riistvaralise ega ka tarkvaraliste muudatuste eest. Teenusepakkujad vastutavad kõige eest. Tuntumaiks näiteks on GMAIL. [35]

[1] Pilvepõhine teenus, mis haldab infrastruktuurilisi elemente

[2] Pilvepõhine teenus, mille puhul haldab kõiki aspekte teenusepakkuja

PaaS ehk *Platform as a Service* [1] pakub platvormi, mis võimaldab kasutajal rakendusi arendada, käitada ja hallata ilma infrastruktuuri ehitamise ja hooldamiseta. Seda on kõige lihtsam kasutada lihtsa veebirakenduse ülespanemiseks. Tuntumateks näideteks Google App Engine ja Heroku. [35]

FaaS ehk Function as a Service[2] on sarnane PaaS-ile, kuid on veel lihtsamini kasutatav. Põhjuseks on FaaS-i automaatne skaleeruvus. Seega põhineb ka hinnakiri veebirakenduse tegelikul kasutatavusel, mitte deklareeritud ressursivajadusel. Tuntumad näited on AmazonWeb Services Lambda ja Google Cloud Functions. [36]

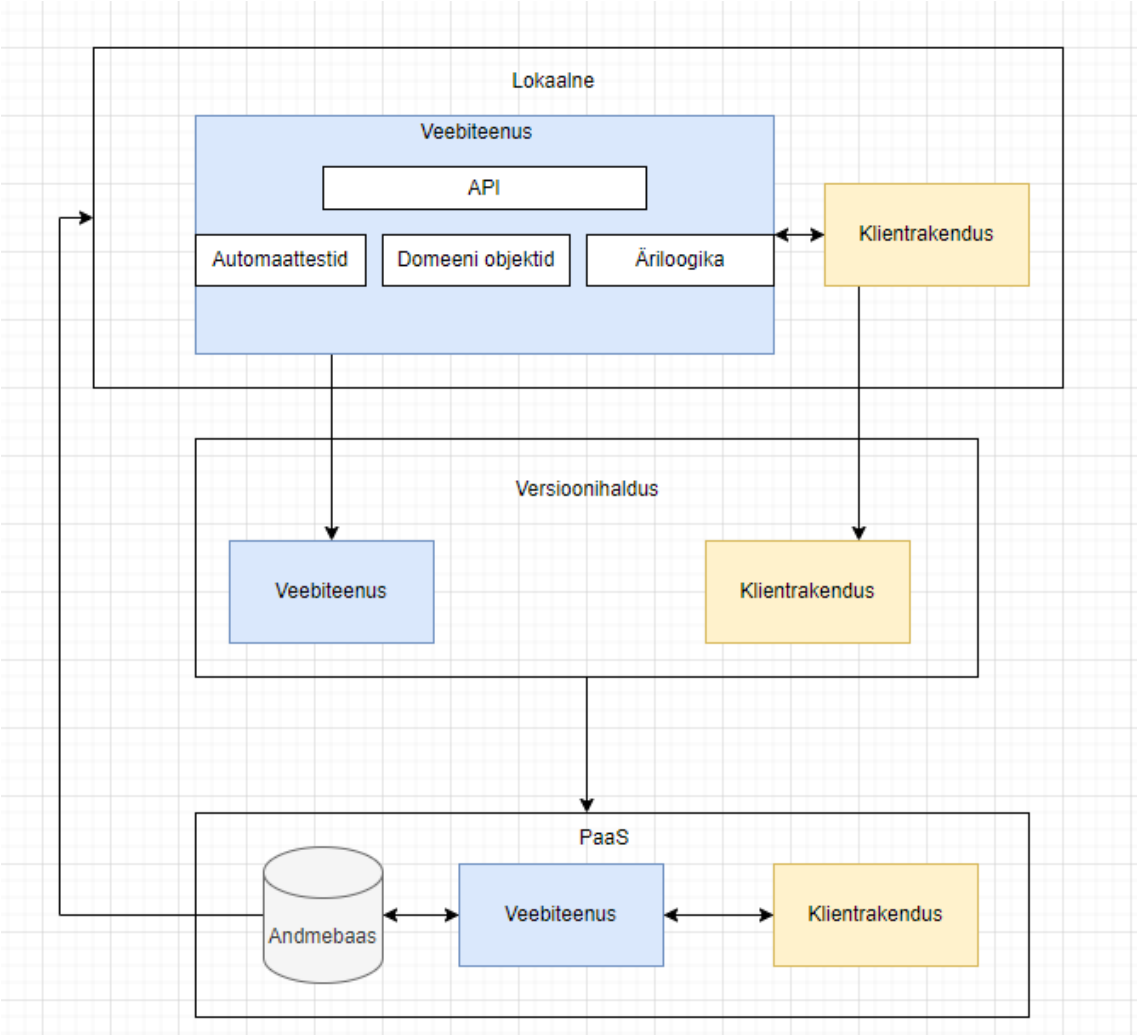
Võttes arvesse lõputöö skoopi ja nõudeid, on heaks lahenduseks PaaS. Kuigi FaaS on automaatselt skaleeruv, otsustas autor PaaS-i kasutks, kuna on sellega (Herokuga) varem kokku puutunud ja see on põhimõttelt sarnane FaaS-ile.

[1] Pilvepõhine teenus, mis haldab kõike, mis ei ole seotud rakenduse ning äriloogikaga

[2] Pilvepõhine teenus, sarnane PaaS-ile, ning automaatselt skaleeruv

3.6 Arhitektuur

Võttes arvesse, et veebirakenduse serveerimiseks on valitud PaaS, koosneb veebirakenduse arhitektuur järgnevatest komponentidest: Lokaalne arenduskeskkond, versioonihaldus ja PaaS. Rakenduse lokaalne versioon laetakse GitHubi ülesse. GitHub ja Heroku (PaaS) on omavahel seotud. Iga kord, kui veebiteenuses või klientrakenduses tehakse muudatusi kajastuvad need ka Herokus.



Joonis 1. Veebirakenduse arhitektuur

Antud veebirakenduse puhul arendatakse veebiteenust ja klientrakendust eraldi. Veebiteenus suhtleb andmebaasiga ja klientrakendusega, mille omavaheliseks suhtluseks kasutatakse *API-t* [1]. Mõlemal on oma eraldi versioonihaldus.

Lõppkasutaja suhtleb veebirakendusega läbi klientrakenduse, mida serveerib PaaS.

3.7 Veebirakenduse disain

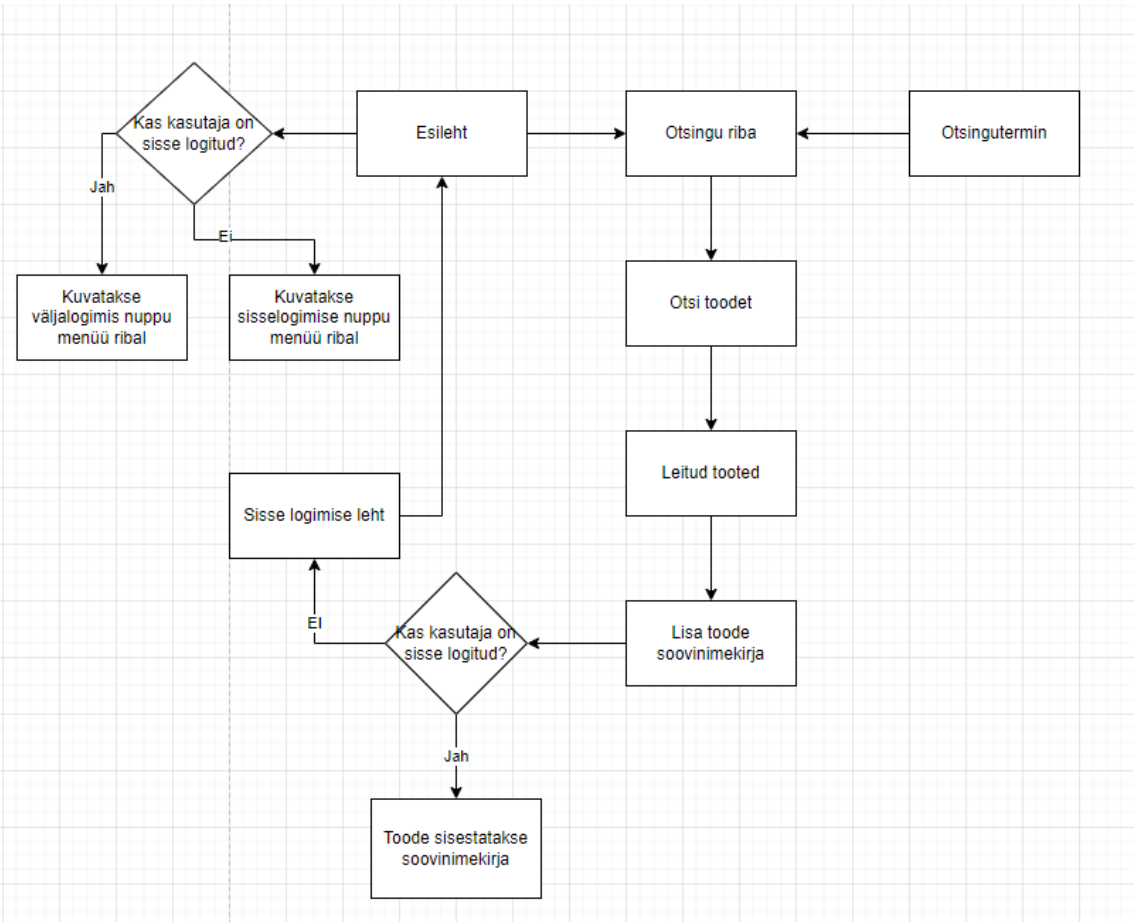
Veebirakenduse loomisel alustatakse kasutajakogemuse disainist. Kasutajakogemuse disaini loomine lubab paremini hinnata, mis funktsionaalsusi on vaja luua nii klientrakenduses kui ka veebiteenuses. Funktsionaalsuste selgumisel luuakse sellele vastav andmebaasi skeem.

3.7.1 Kasutajakogemuse disain

Tulenevalt süsteemi nõuetest võib veebilehel navigeerimise jagada kahte ossa: esileht ja soovinimekiri.

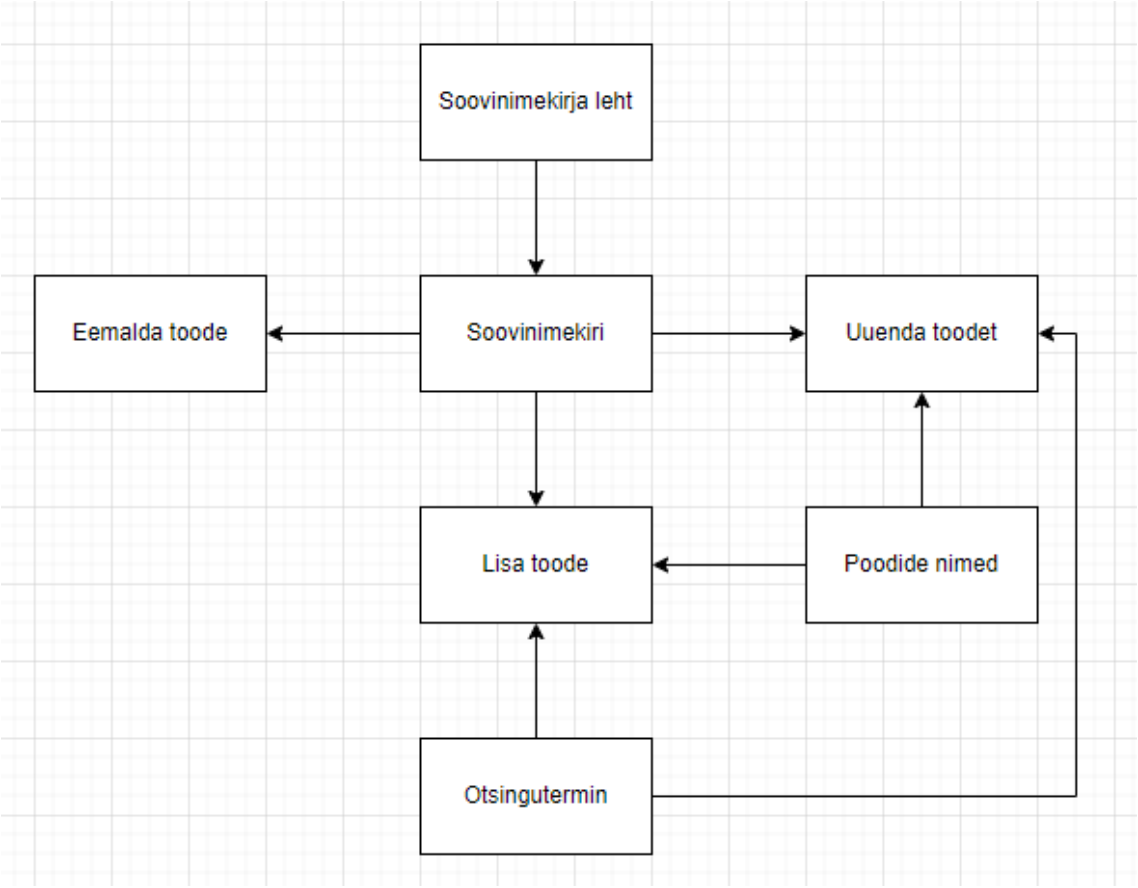
Joonisel 2 on välja toodud esilehe kasutajakogemus, mis keskendub põhiliselt toodete otsimisele ning kuvamisele. Kui toode peaks olema välja müüdud ja kasutaja on sisse logitud, on kasutajal võimalus lisada see toode otse esilehelt soovinimekirja. Välja müüdud toodete all kuvatakse soovinimekirja lisamise nuppu, millele vajutades lisatakse toode koos valitud poega soovinimekirja.

[1] Rakendusliides, mis võimaldab veebiteenusel ja klientrakendusel omavahel suhelda



Joonis 2. Esilehe kasutajakogemuse diagramm

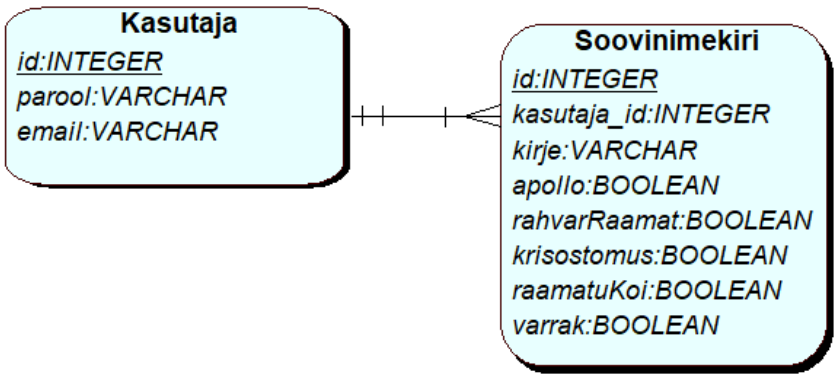
Joonisel 3 on ära toodud soovinimekirja kasutajakogemus. Lisaks sellele, et kasutaja saab ka otse esilehelt otsitud tooteid soovinimekirja lisada, saab seda teha ka soovinimekirja lehel, kui kasutaja on sisse logitud. Soovinimekirja lisamisel tuleb määrata otsingutermin, mille järgi poodidest otsitakse. Samuti tuleb määrata poed, kust soovitakse toodet otsida. Veel saab juba soovinimekirjas olevate toodete kirjeldusi muuta ja ka eemaldada.



Joonis 3. Soovinimekirja kasutajakogemuse diagramm

3.7.2 Andmebaasi disain

Lähtuvalt süsteemi nõuetest ei vaja antud veebirakendus keerulist andmebaasi. Andmebaasi salvestatakse ainult kasutajad ja soovinimekirja kirjed. Joonisel 4 on esitatud soovinimekirja funktsionaalsuse toimimiseks vajalikud olemid.



Joonis 4. Soovinimekirja olemid

3.8 Analüüsi kokkuvõte

Analüüsi osas käsitleti süsteemi funktsionaalseid ja mittefunktsionaalseid nõudeid. Tehnoloogia poole pealt sai valitud teenusepoole arenduseks Java ja selle raamistik Spring. Need on valimis aspekte arvestades sobivad vahendid serveripoole arenduseks ning autoril on nendega juba ka varasem kokkupuude. Kliendipoolseks arenduseks sai valitud Angular, mis on enamasti mõeldud küll suuremate rakenduste jaoks, kuid kuna see on üsna populaarne, võimekas ja autoril on hea kogemus sellega, otsustati jääda selle juurde.

Serverilahenduseks on võetud PaaS koos Herokuga, kuna see on üks lihtsamaid viise veebirakenduse serveerimiseks. Antud lahenduse puhul toimub kogu haldus teenusepakkuja poolt.

Andmebaasiks valiti pilvepõhine PostgreSQL, peamiselt serverilahenduse valiku alusel. Herokul on hea integratsioon Postgre-ga, mis tähendab, et Postgre andmebaasi saab hoida Heroku pilves. Kuna serveripoolt ja kliendipoolt plaanitakse nagunii juba Herokus hoida, ning sellel on Postgre integratsioon, on mõistlik ka andmebaasi Herokus hoida.

Koodi kirjutatamiseks on kasutatud kahte IDE-t[1]. Serveripoolset koodi kirjutatakse IntelliJ IDE programmis ja kliendipoolt Visual Studio Code programmis. Koodihalduskeskkonnaks valiti GitHub.

[1] Integrated development environment – integreeritud arenduskeskkond

4 Veebirakenduse arendus

Käesoleva veebirakenduse arenduse kirjeldus on jaotatud nelja suuremasse peatükki: veebiteenus, klientrakendus, testimine ja pidev integratsioon. Järgnevates peatükkides keskendutakse kasutatavatele tehnoloogiatele ja arenduse meetoditele.

4.1 Veebiteenuse-poolne lahendus

Veebirakenduse veebiteenuse osa on ehitatud Spring Boot raamistiku peale. Spring Boot on Springi enda laiendus, et muuta arendus, testimine ja rakenduse kasutuselevõtt veelgi lihtsamaks. Veebiteenuse kiht vastutab nii äriloogika kui ka andmebaasiga suhtlemise eest. Järgnevates peatükkides on lähemalt kirjeldatud veebiteenuse tehnoloogilisi lahendusi ja arenduse aspekte.

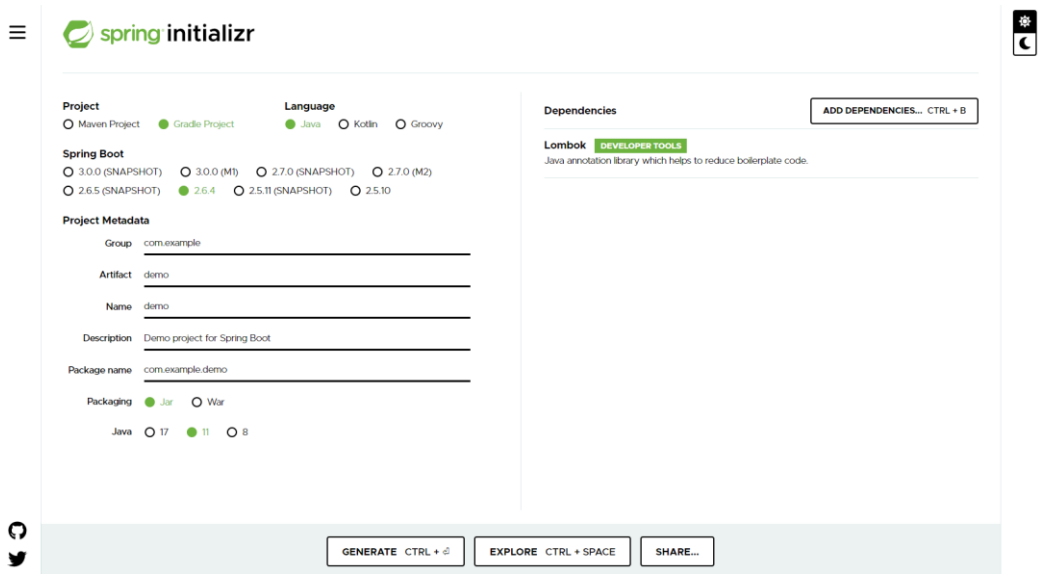
4.1.1 Spring Boot rakenduse arhitektuur

Spring Booti suureks plussiks on see, et koodi kirjutamisega saab pea koheselt alustada. See tähendab, et ei pea aega raiskama keskkonna ettevalmistamisele ja seadistamisele. Erinevalt teistest Java raamistikest pakub see paindlike XML[1]-i konfiguratsioone, andmebaasiga suhtlemist, lihtsat töövoogu ja laia valikut arendustööristu. [37] Samuti kaasneb Spring Bootiga sisseehitatud Tomcat, Jetty või Undertow server. [38]

Nagu eelnevalt sai mainitud on Spring Boot projekti loomine väga kerge. Spring initializr pakub projekti loomiseks mugavat ja arusaadavat kasutajaliidest, mis lubab luua uue Spring Boot projekti vajalike tööriistade ja teekidega.

Joonisel 5 on välja toodud Spring Boot projekti loomise kasutajaliides, kus saab valida projekti ehitamise ja haldamise tööriista (Gradle/Maven), programmeerimiskeelt, soovitud Spring Boot versiooni, projekti metaandmeid, ja väliseid teeke. Samuti saab enne projekti loomist vaadata, milline loodav projekt välja näeks.

[1] Extensible Markup Language – dokumentide laiendatav märgendamiskeel



Joonis 5. Spring Initializr kasutajaliides

Projektis loodavate klasside haldamiseks on failid jaotatud järgnevatesse kaustadesse:

- Configuration – konfiguratsioonifailid.
- Controllers – REST API kontrollerid.
- Models – domeeniobjektid äriloogika perspektiivis.
- Security – turvalisuse loogikaga seotud klassid.
- Repository – andmebaasiga suhtlemiseks.

4.1.2 Veebikaabitsa teegid

Loodava veebirakenduse veebiteenuse suurem äriloogiline aspekt tuleneb veebikaabitsa tööst. Veebikaapimine on meetod, mida kasutatakse andmete kogumiseks veebilehekülgedelt. Andmed kogutakse kokku ja muudetakse sobivasse formaati. Antud lõputöös on vaadatud kolme erinevat veebikaabitsa teeki: JSoup, Selenium, HTML Unit.

4.1.2.1 Jsoup

Jsoup on Java teek HTML-iga töötamiseks. Jsoup pakub väga mugavat API-t veebikaapimiseks kasutades parimaid HTML5 DOM[1] meetodeid ja CSS-i valijaid.

[1] Document Object Model – keelest sõltumatu dokumentidega suhtlemise liides

Samuti on Jsoupil väga hea dokumentatsioon ja see on üks populaarsemaid Java veebikaapimisteeke. [39]

4.1.2.2 Selenium

Selenium on veebirakenduste testimise raamistik. Peale testimise kasutatakse Seleniumit veel veebikaapimise tööriistana. Kuna Selenium on üsna suur raamistik mitmete komponentidega, siis lõputöös käsitletakse ainult Seleniumi WebDriver komponenti, mida kasutatakse veebikaapimiseks. Erinevalt teistest veebikaabitsatest tuleb Selenium hästi toime ka dünaamiliste veebilehekülgedega. [39] Dünaamiliste veebilehekülgede puhul luuakse veebilehekülgede sisu reaalajas JavaScripti poolt. Dünaamiliste lehekülgede aspekti tuleb veebikaabitsa valimisel silmas pidada, kuna teised lõputöös käsitletavad teegid dünaamiliste lehekülgede kaapimist ei paku. Seleniumi miinuseks on aga selle kiirus. Paljude andmete laadimisel võib laadimine rohkelt aega võtta.

Kuna Rahva Raamatu veebilehekülg on dünaamiline, tuleb sealt andmete kogumiseks kasutada Seleniumit, kuna teised teegid ei tule toime dünaamiliste lehekülgedega.

Joonisel 6 on välja toodud andmete kogumine Rahva Raamatu leheküljelt. Kõigepealt otsitakse üles konteiner, kus on kõik tooted sees. Konteinerist otsitakse välja kõik pealkirjad ja autori nimed, mis on listi kujul. List käiakse läbi, et pealkiri ja autori nimi omavahel kokku sobitada ja nendest objekt teha.

```
searchParam = searchParam.replaceAll(" ", "%20");
        String URL = "https://rahvaraamat.ee/s/" + searchParam +
"/et?productType=BOOK&page=1";

        driver.get(URL);
        WebElement products =
driver.findElementByClassName("styles_gridViewContainer__3y5Q5");

        List<WebElement> titles =
products.findElements(By.className("styles_productTitle__205Z0"));
        List<WebElement> authors =
products.findElements(By.className("styles_productSubtitle__1Bhcq"));
        var productsToGet = Math.min(authors.size(), 10);

        List<DataModel> dataModelList = new ArrayList<>();

        for (int i = 0; i < productsToGet; i++) {
            var title = titles.get(i).findElement(By.tagName("a")).getText();
            var author = authors.get(i).getText();
             DataModel data = new DataModel();
            data.setBookTitle(title);
            data.setAuthorName(author);
            dataModelList.add(data);

}
```

Joonis 6. Veebikaapimine Seleniumiga

4.1.2.3 HtmlUnit

HtmlUnit on kasutajaliideseta brauser Java programmide jaoks. See modelleerib HTML-dokumente ja pakub API-t, mis võimaldab avada veebilehti, täita vorme, klõpsata linkidele jne. Enamasti kasutatakse seda testimiseks, kuid sobib ka veebikaapimiseks. [41] Eelmises peatükis sai mainitud, et dünaamiliste lehekülgede pärast tuleb kasutada Seleniumit. Kuna Selenium on aga pea kaks korda aeglasem kui HtmlUnit, võetakse ülejäänud nelja poe kaapimisel kasutusse HtmlUnit ja Rahva Raamatu lehekülje kaapimiseks kasutatakse Seleniumit.

Joonisel 7 on näidatud samade andmete otsimist nagu joonisel 6. Selenium pakkus andmete otsimiseks paindlikumaid CSS valijaid. HtmlUnit selliseid pandlikke valijaid ei paku ja sellepärast on kasutusele võetud XPath[1].

[1] Väljenduskeel, mis on loodud toetama XML-dokumentide päringuid või teisendamist

```
WebClient webClient = new WebClient();
        webClient.getOptions().setCssEnabled(false);
        webClient.getOptions().setJavaScriptEnabled(false);

        searchParam = searchParam.replaceAll(" ", "%20");
        String URL =
"https://www.apollo.ee/catalogsearch/result/index/?cat=default-category%7Ce-
raamatud,default-category%7Craamatud&q=" + searchParam;
        HtmlPage page = webClient.getPage(URL);

        List<HtmlElement> products =
page.getByXPath("/html/body/div[2]/div/div/div[4]/div/div[2]/div[2]/div[2]/di
v/div");
        var productsToGet = Math.min(products.size(), 10);
        List<DataModel> dataModelList = new ArrayList<>();

        for (int i = 1; i <= productsToGet; i++) {
            var title =
((HtmlAnchor)page.getFirstByXPath(String.format("/html/body/div[2]/div/div/di
v[4]/div/div[2]/div[2]/div[2]/div/div[%s]/div[3]/h2/a",
i))).getTextContent().trim();
            var author =
((HtmlElement)page.getFirstByXPath(String.format("/html/body/div[2]/div/div/d
iv[4]/div/div[2]/div[2]/div[2]/div/div[%s]/div[3]/div[1]",
i))).getTextContent().trim();

            DataModel data = new DataModel();
            data.setBookTitle(title);
            data.setAuthorName(author);
            dataModelList.add(data);
        }
```

Joonis 7. Veebikaapimine HtmlUnit-iga

4.1.3 REST API

REST[1] API[2] on rakenduste programmeerimisliides, mis järgib REST-i arhitektuurilisi piiranguid. [39] Koos HTTP-protokolliga, moodustab REST lihtsa päringu/vastuse mehhanismi. Antud lõputöös kasutatakse kliendi ja serveripoole vaheliseks suhtluseks REST API-t.

REST API kasutab suhtluseks erinevaid päringuid, mis jagunevad tüüpide poolest järgnevalt: GET, POST, PUT ja DELETE. REST võib läbi HTTP edastada

[1] Representational state transfer – tarkvaraarhitektuuri laad

[2] Application programming interface

informatsiooni mitmel kujul: JSON[1], HTML[2], *plain text* jne. Kuna JSON on üks populaarsemaid andmeedastusformaate, kasutatakse seda. Samuti on JSON lihtsasti loetav ja arusaadav.

Päringud jaotatakse kontrollerite põhjal erinevatesse gruppidesse. Gruppides on välja toodud päringu kirjeldus, tüüp ja ressurss, mis tähistab kontrolleri aadressi, kuhu päring läheb. Tabelis 2 on välja toodud kõige olulisemad päringud, mis kirjeldavad teenuse funktsionaalsusi.

[1] Javascript Object Notation - andmevahetusvorming

[2] HyperText Markup Language – veebilehtede märgenduskeel

Tabel 2. Veebiteenuse API päringud

Päringu kirjeldus	Päringu tüüp	Ressurss
Autentimiskontroller		
Sisse logimine	POST	/login
Registreerimine	POST	/signup
Soovinimekirja kontroller		
Soovinimekirja lisamine	POST	/wishlist/{id}
Soovinimekirja pärimine	GET	/wishlist/{id}
Soovinimekirjast kustutamine	DELETE	/wishlist/{id}

4.1.4 Veebiteenuse turvalisus

Koos valitud Springi raamistikuga tuleb kaasa ka alamraamistik Spring Security. Spring Security on autentimis- ja autoriseerimisraamistik. [40] Läbi autentimise eristatakse süsteemis kasutajaid ja autoriseerimine vastutab kasutajatele määratud õiguste kontrolli eest. Kuna aga antud süsteemis on ainult tavakasutajad, ei ole autoriseerimisel tarvidust, kuna kõigil kasutajatel on samad õigused.

Olemuselt kujutab Spring Security endast filtrite ahelat, mis seisab klienditeenuse poolt tulevate päringute ja veebirakenduse kontrollerite vahel. Kuigi Spring Security-t saaks implementeerida ka ainult ühe filtri kujul, on soovituslik luua filtrite ahel, kus igal filtril on oma ülesanne. Näiteks sisselogimise puhul läheb sissetulev päring kõigepealt läbi sisselogimisfiltri, siis läbi autentimisfiltri ja lõpuks jõuab päring kontrollerini. [41]

Autentimise jaoks on erinevaid võimalusi, kuid kõige populaarsem moodus on kasutada JSON Web Token-it (JWT). JWT on standard, mida kasutatakse teabe turvaliseks edastamiseks, kuid põhiliselt kasutatakse seda autentimiseks. [42]

JWT koosneb kolmest osast: *header* ehk päis, *payload* ehk edastatav info ja *signature* ehk allkiri. JWT päis sisaldab sellele rakendavate krüptograafiliste toimingute loendit. See võib olla allkirjastamistehnika, sisutüübi metaandmete teave jms. Allkirja osa kasutatakse kontrollimiseks, et teavet ei ole vahepeal muudetud. Allkiri luuakse kasutades päises kirjeldatud algoritmi, *payloadi's* olevat teavet ja privaatset saladust, mis on privaatvõtme sarnane. [42]

4.1.5 Konfiguratsioon

Teenuse seadistus käib läbi konfiguratsiooniklasside ja failide, mis asetsevad *Configuration* kaustas. Seal on kirjeldatud üldine andmebaasipõhine konfiguratsioon. Samuti kasutatakse Resources kaustas olevat *application.properties* faili, mis sisaldab andmebaasi parameetreid.

4.2 Klientrakenduse lahendus

Klientrakenduse arendamiseks valiti Angulari raamistik, mis põhineb TypeScriptil. Angular on täismahuline veebiarendusraamistik, mis kasutab komponendipõhist arhitektuuri, mis soodustab koodi taaskasutust, loetavust ja hooldatavust. Järgenvates peatükkides on süvitsi kirjeldatud, kuidas on kasutatud Angulari klientrakenduse loomiseks.

4.2.1 Komponentide planeerimine

Komponendid on põhiosa Angulari rakenduses. Iga komponent koosneb neljast osast:

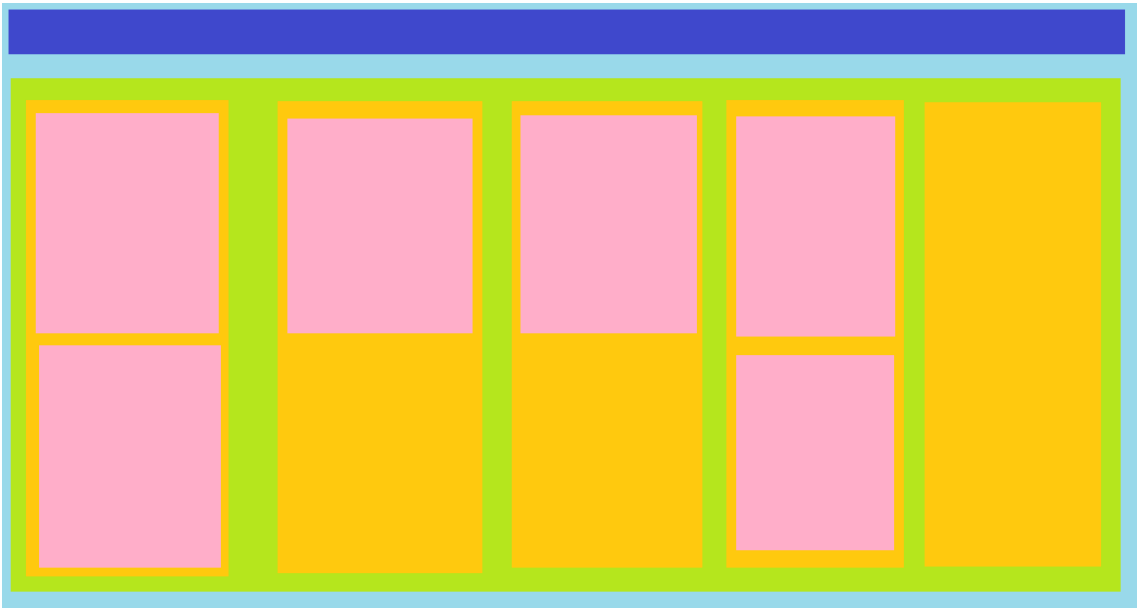
- HTML mall - deklareerib, mida lehel renderdatakse
- CSS valija – HTML mallile rakendatud kujundus
- TypeScripti klass – määrab komponendi käitumise
- Spec fail – komponentide testimiseks

Kokkuvõtvalt on rakendusel vaja 8 komponenti, mis on järgnevad:

- Põhikomponent – selle peale kuvatakse kõik teised komponendid
- Navigeerimiskomponent – navigeerimislingid ja otsinguriba
- Otsingutulemuste komponent – otsinguribasse sisestatud ja leitud toodete kuvamine
- Registreerimiskomponent – kasutaja registreerimine

- Sisselogimiskomponent – kasutaja sisselogimine
- Soovinimekirja komponent – toodete lisamiseks ja kustutamiseks soovinimekirjast
- Tutvustav komponent – veebilehekülge tutvustav info
- Laadimiskomponent – näidatakse kasutajale, kui tooteid laetakse

Joonisel 8 on toodud klientrakenduse vaade pärast raamatute otsimist ja vaate kuvamiseks vajalikud komponendid. Helesinine kast hõlmab kogu lehte ja tähistab põhikomponenti, mille peal kuvatakse kõik teised komponendid. Lilla kast on navigeerimismenüü. Roheline kast on antud näite puhul otsingutulemuste komponent, mis kuvab kõik poed ja poodidest leitud raamatud. Oranžid kastid tähistavad viite erinevat poodi ja roosad kastid vastavalt otsingutulemustele leitud tooteid.



Joonis 8. Otsingutulemuste lehe komponentide paigutus

4.2.2 Angular – *Single Page Application* (SPA)

Traditsiooniliselt olid veebirakendused MPA (*Multi-Page Application),* mis tähendab, et iga hiireklõpsuga laaditi serverist uus leht. See lähenemisviis aga suurendab serveri koormust ja muudab veebilehe laadimise aeglasemaks. SPA ehk *Single page application* tähendab ühe lehe veebirakendust, mis laeb korraga alla ainult ühe HTML-lehe ja lehel navigeerides värskendatakse ainult mingit osa sellest, mis tähendab, et

serverist ei pea tervet lehte uuesti alla laadima. See omakorda tagab veebilehe kiirema töö. [43]

Parandamaks kasutajakogemust on ka antud lõputöös klientrakenduse veebilehe formaadiks võetud SPA. Kuigi mahukate API päringute tegemisel võib lehe laadimine ikka kauem aega võtta, saab senikaua kasutajale näidata progressiooni laadimisrõngast.

4.2.3 Angular rakenduse struktuur

Komponentide planeerimise peatükis (4.2.1) sai juba mainitud, et Angulari rakenduse põhiosaks on komponendid, kuid tegelikkuses pole see kõik. Komponendid jaotatakse omakorda veel moodulitesse. Moodulid koguvad seotud koodi (mitte ainult komponendid) funktsionaalseteks komplektideks, ehk siis moodulid või Angulari keeles *NgModules* on Angulari rakenduse hierarhilise struktuuri tipus. Igal Angulari rakendusel on juurmoodul ehk *AppModule*, mis sisaldab tavaliselt palju teisi mooduleid. See omakorda tähendab, et mooduleid saab importida ja eksportida, ehk siis mooduleid saab kasutada teistes moodulites. Koodi jaotamine erinevatesse moodulitesse aitab koodi hoida organiseeritult ja loetavalt ning samuti saab koodi nii paremini taaskasutada. [44]

Lisaks moodulitele ja komponentidele kasutatakse tavapärases Angular rakenduses veel teenuseid (*services*) ja sõltuvuse süstimist (*dependency injection*). Andmed või koodi loogika, mis pole konkreetse komponendiga seotud või mida saaks mitme komponendi vahel jagada, paigutatakse teenustesse. Koos sõltuvussüstimistehnikaga, saab teenused sisestada komponentidesse, mida komponendid saavad näiteks andmete pärimiseks kasutada. Seeläbi välditakse koodi kordumist ja eraldatakse kood loogilistesse osadesse. [44]

Viimaseks Angular rakenduse struktuuri põhiosaks on *routing* ehk navigeerimine. Tegelikkuses on *Angular Router* lihtsalt üks eraldiseisev moodul, kus saab defineerida navigatsiooni tee. Tavaliselt, kui vajutada navigeerimis lingile, laeks brauser uue lehe, kuid Angulari ruuter peatab sellise brauseri käitumise ja olenevalt navigeerimislingi URL-ist kuvab õige komponendi vaate. [44]

4.2.4 Angular Material

Angular Material on kasutajaliidese komponentide teek Angulari arendajatele, ehk valmis loodud komponentide teek. Angular Material aitab arendajatel keskenduda rohkem rakenduse arenduse funktsionaalsele poolele ja vähem disainile. [45] Materiali kasutamiseks Angulari projektis tuleb see projekti lisada, kuna algselt projekti luues ei ole see sinna kaasa lisatud. Lisaks, et Material komponente kasutada, tuleb need importida vastavasse moodulisse. Kuigi Material on valmis komponentide kogum, on võimalus neid komponente ka oma tahtmise järgi kujundada, mis pakub lisapaindlikust.

Hetkel pakub Angular Material 36 erinevat valmiskomponenti. Samuti pakub Material *Component Dev Kit* (CDK), mis on kogum primitiividest. CDK võimaldab lihtsa vaevaga arendajal endal komponente luua. [45]

Kuna Material pakub ilusa disainiga valmiskomponente on suur osa klientrakenduse kasutajaliidesest ehitatud kasutades neid komponente.

4.2.5 Angular rakenduse serveerimine brauserile

Varasemalt on mainitud, et rakenduse nähtavaks tegemiseks ülejäänud internetile kasutatakse Heroku platvormi. Selleks, et Angulari klientrakendus Herokusse üles laadida, on vaja teha paar seadistust. Kõigepealt tuleb projekt ehitada, mis tähendab, et vajalikud failid kompileeritakse brauserile arusaadavasse formaati. Kompileeritud failid paigutatakse jaotuskausta (*distribution folder*), mida hiljem Heroku kasutab veebirakenduse serveerimiseks. Teiseks tuleb projekti lisada Node server, kuna Angulari projekti ei ole võimalik ilma selleta Herokusse laadida. Serveri lisamine käib läbi ühe javascripti faili, kus kirjeldatakse ümbersuunamine http-lt https-ile, päringute ümbersuunamine index.html faili (Angular router hoolitseb selle eest, millist komponenti kuna näidata) ja kuulatav port, mis on tavaliselt 8080. Pärast kirjeldatud sammude tegemist on võimalik projekt laadida Herokusse, mis serveerib rakendust ülejäänud internetile. [46]

4.2.6 Klientrakenduse turvalisus

Teenusepoolel kasutatakse autentimiseks JWT, mis tuleb implementeerida ka klientrakenduse poolel. Selleks tuleb klientrakenduses kõigepealt sisestada vajalikud andmed sisselogimiseks (kasutajanimi, parool), mis saadetakse teenusepoolele. Kui

andmed on õiged, saadab teenusepool tagasi JWT, mis tuleb klientrakenduse puhul meelde jätta. Meeldejätmiseks kasutatakse *local storage* võimalust, mis lubab JWT brauserisse salvestada. Järgmiseks tuleb luua *HttpInterceptor*, mis kontrollib, kas kasutaja on autenditud, ning seejärel lisab kõigile väljaminevatele päringutele JWT. Sedasi ei pea teenusepool iga tuleva päringu peale uut JWT genereerima, vaid kontrollib saadetud JWT õigsust ja seejärel saadab vastuse. [47]

4.3 Testimine

Käesolevas lõputöös on keskendutud ainult kliendipoole testimisele, kuna töö kontekstis pole teenusepoole testide järgi suurt vajadust.

Lõputöös on testid kirjutatud ühiktestidena. Ühiktestimine on tarkvara testimise meetod, mille käigus testitakse tarkvara üksikuid komponente. Ühiktestid aitavad veenduda koodi korrektses toimimises ning võimaldavad tulevikus vigu kiiremini tuvastada. Samuti on ühiktestid hea viis koodi dokumenteerimiseks, kuna need kirjeldavad hästi, mida teatud koodijupp peaks tegema. [51]

4.3.1 Kliendipoolsed automaattestid

Angulari projektis genereeritakse iga komponendi või teenuse loomisel ka sellega seostuv *spec* ehk testimisfail. Angular kasutab oma testide jooksutamiseks Karmat ja Jasminit.

Jasmine on testimise raamistik ja Karma jooksutab loodud teste. Tavaline Angulari komponendi testfail koosneb üldjuhul kolmest osast:

- describe – testbloki algus, mis saab sisendiks testitava komponendi nime.
- beforeEach – asünkroonne funktsioon, mis jookseb enne igat testi ja mille eesmärgiks on lasta kogu võimalikul asünkroonsel koodil lõppeda enne järgmise testi juurde minemist.
- it – testi sisu, kus kirjeldatakse oodatav sisend ja väljund.

Lisaks neile kolmele osale tuleb enne ka kõik komponendiga seonduvad sõltuvused importida. [52]

Joonis 9 kujutab kahte automaattesti, mis testivad lapskomponendi näitamist vanemkomponendi sees. Üks test kontrollib, et komponenti ei näidataks, kui nähtavus on valeks määratud, ning teine kontrollib vastupidist.

```
describe('AppComponent', () => {
  beforeEach(async () => {
    await TestBed.configureTestingModule({
      imports: [
        RouterTestingModule
      ],
      declarations: [
        AppComponent,
        WelcomeComponent
      ],
    }).compileComponents();
  });

  it('should hide welcome component if showWelcome is false', () => {
    const fixture = TestBed.createComponent(AppComponent);
    const appComponent = fixture.componentInstance;
    appComponent.showWelcome = false;
    fixture.detectChanges();
    expect(fixture.debugElement.query(By.css('app-welcome'))).toBeNull();
  });

  it('should show welcome component if showWelcome is true', () => {
    const fixture = TestBed.createComponent(AppComponent);
    const appComponent = fixture.componentInstance;
    appComponent.showWelcome = true;
    fixture.detectChanges();
    expect(fixture.debugElement.query(By.css('app-welcome'))).toBeTruthy();
  });
});
```

Joonis 9. Angulari automaattestid, kontrollimaks komponendi näitamist õige parameetri korral

4.4 Pidev integratsioon

Pidev integratsioon on tarkvara tava, kus kood laetakse üles jagatud hoidlasse nagu GitHub. Pideva integratsiooni eesmärgiks on vigade varajane tuvastamine. Koodi laadimisel hoidlasse saab jooksutada ka koodiga kaasas olevaid teste. Nii saab muudatuste tegemisel jälgida rakenduse korrasolekut. Samuti saab kontrollida koodi stiili, rakenduse turvaaspekte jms. [48]

4.4.1 Heroku ühendus GitHubiga

Heroku integreerub GitHubiga, mis tähendab, et laadides uue muudatuse GitHubi, kajastub see automaatselt ka Herokus. Kasutades ära seda integratsiooni, ei pea iga kord uut juurutust Herokusse tegema, vaid piisab muudatuste laadimisest GitHubi. GitHubi integratsiooni konfigureerimiseks on vajalik repositooriumi URL-i sisestamine Heroku projekti. [49]

5 Hinnang loodud veebirakendusele

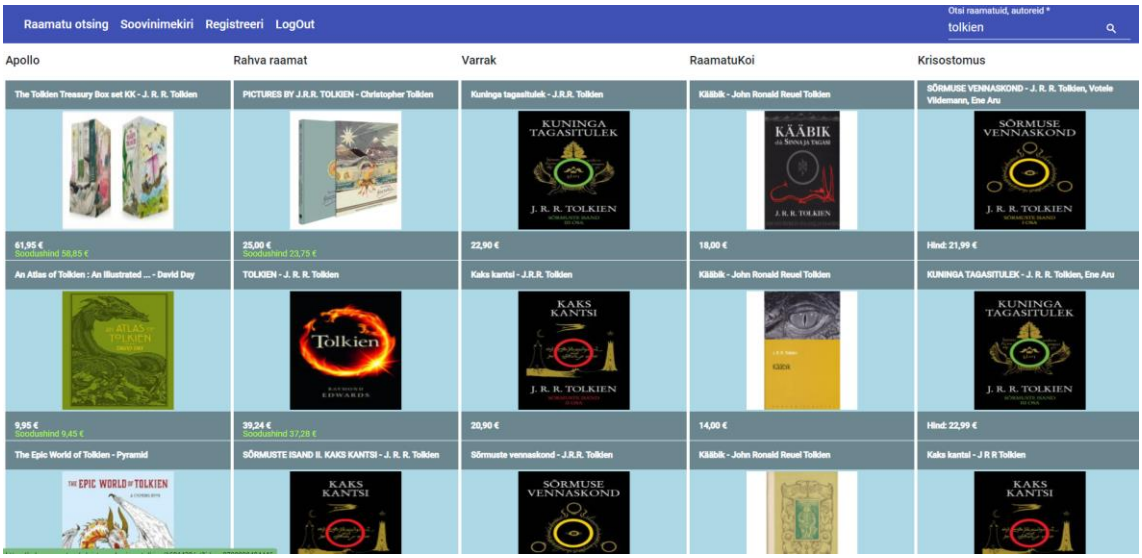
Hinnangu loodud veebirakendusele saab anda oodatud ja saavutatud funktsionaalsuse alusel. Samuti saab hinnangu anda edasiarendatavusele.

5.1 Saavutatud kasutatavus

Kõik määratletud funktsionaalsed ja mittefunktsionaalsed nõuded ei saanud projekti arendamise jooksul aja puuduse tõttu realiseeritud.

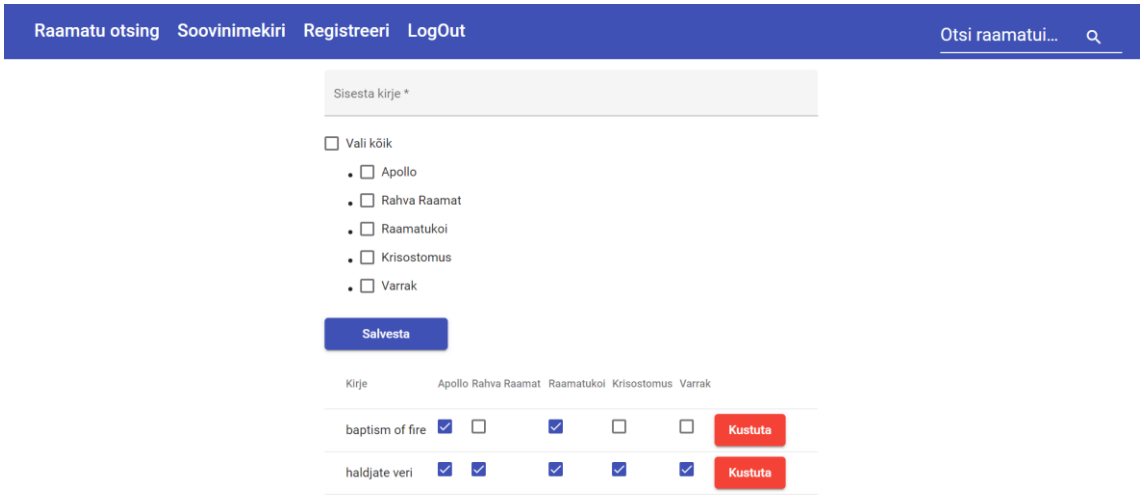
Suurimaks probleemiks on toodete otsimise kiirus. Otsimise ajal näidatakse kasutajale progressiivset spinnerit, mis parandab kasutajakogemust, kuid sellegipoolest on kiirus probleemiks. Probleem tuleneb peamiselt toodete otsimisel Rahva Raamatu poest. Kuna Rahva Raamatu pood kasutab toodete laadimiseks JavaScripti, ehk veebilehe sisu laetakse dünaamiliselt, siis ei suuda veebikaabits alati tooteid üles leida, kuna veebilehe sisu pole veel laetud. Selle probleemi lahendamiseks tuleb lisada ooteaeg, et veebikaabits hakkaks tooteid alles siis otsima, kui veebileht on täielikult laetud.

Joonisel 10 on kuvatud otsingutulemuste lehekülge. Lehel on kuvatud viie raamatupoe otsingutulemused vertikaalselt. Välja on toodud raamatu pealkiri, autori nimi, hind ja heleroheliselt soodushind. Tootele vajutades suunatakse kasutaja vastava raamatupoe toote leheküljele. Funktsionaalsus, mis sai algselt plaanitud, kuid aja puuduse tõttu jääb edasiarenduseks, on toote lisamine otse sellelt leheküljelt soovinimekirja.



Joonis 10. Otsingutulemuste kuvamine

Joonisel 11 on kuvatud soovinimekiri. Kasutaja saab soovinimekirja lisada kirje ja poed, kust vastavat kirjet otsida. Samuti saab kasutaja tooteid soovinimekirjast eemaldada. Pärast kirje lisamist soovinimekirja, käib veebikaabits kord päevas kõik valitud poed läbi. Kui poodidest leitakse vastav kirje(d), saadetakse kasutajale meili peale teade.



Joonis 11. Soovinimekirja vaade

Töö mittefunktsionaalsetes nõuetes on kirjas nutiseadme brauseri vaade. Vaated on kuvatud lisas 2. Samuti on kuvatud esilehe vaade ja progressiooni laadimisrõnga näitamine lisas 3.

5.2 Võimalused edasiarenduseks

Üheks suurimaks edasiarenduse eesmärgiks on lisada veebikaabitsate monitooring. Monitooring on vajalik, kuna raamatupoodide veebilehekülje võivad muutuda. Veebilehekülje suuremat sorti muutumisel lähevad tõenäoliselt veebikaabitsad katki ja sellest peaks rakenduse arendajat võimalikult ruttu teavitama, et arendaja saaks muudatused kiirelt sisse viia.

Teiseks eesmärgiks on ka juba eelmises peatükis mainitud toote lisamine otse otsingutulemuste lehelt soovinimekirja. Kuigi kasutaja saab minna soovinimekirja lehele ja seal toote väga lihtsalt lisada nimekirja, parandaks lisatav funktsionaalsus kasutajakogemust.

Kolmandaks eesmärgiks oleks lisada üldisem toodete otsimise võimalus, kuna mõned poed müüvad peale raamatute ka muid tooteid. Kasutaja saaks spetsiaalselt valida, mis tooteid ta otsida soovib.

Rakendus on suunatud ainult Eesti raamatupoodidele ja seetõttu on veebilehekülg ka eestikeelne, kuid ei teeks paha, kui oleks võimalus valida ka inglisekeelne kasutajaliides, mis oleks neljandaks eesmärgiks.

6 Kokkuvõte

Bakalaureusetöö käigus loodi veebirakendus, mis kuvab kasutajale raamatud koos hindadega erinevatest raamatupoodidest. Rakenduse töövoog on järgmine: kasutaja sisestab leheküljel olevasse otsingumootorisse otsingutermini. Otsingul käivituvad veebikaabitsad, mis otsivad Apollo, Rahva Raamat, Krisostomus, Varrak ja Raamatukoi veebilehekülgedelt raamatuid. Tulemuste lehel kuvatakse korraga 10 toodet (kui on nii palju), kuna need on kõige täpsemad tulemused kasutaja poolt sisestatud otsingule. Tooted kuvatakse raamatu pealkirja, autori, pildi, hinna ja soodushinnaga. Tootele peale vajutades suunatakse kasutaja poe toote leheküljele. Lisaks saab kasutaja lisada kirjeid soovinimekirja. Soovinimekirjas märgitud poed käiakse kord päevas läbi ja kui poodidest leitakse vastav kirje(d), saadetakse selle kohta kasutajale meili peale teavitus.

Veebirakenduse loomiseks on loodud veebiteenuse pool (backend) ja kliendipool (frontend). Veebiteenus loodi kasutades Java Springi raamistikku kogumaks infot erinevatelt veebilehekülgedelt (veebikaabits) ja edastamaks infot API kaudu. Kliendipoole tegemisel kasutati Javascripti Angular raamistiku, mis tegi päringuid veebiteenuse vastu ning kuvab API-st saadud infot.

Veebirakendusega on saavutatud töö alguses püstitatud eesmärk osaliselt. Esiteks ei pea kasutaja raamatute otsimiseks mitut erinevat veebilehte lahti tegema. Teiseks ei pea kasutaja erinevate lehekülgede pakutud raamatute hindu võrdlema. Kõik raamatud ja nende hinnad kuvatakse ühel leheküljel, mis kokkuvõttes säästab kasutaja aega ja teeb raamatute otsimise ning ostmise mugavamaks. Samas jäi aja puuduse tõttu soovinimekirja funktsionaalsus poolikuks. Kasutaja peaks saama lisada tooteid soovinimekirja otse otsingutulemuste lehelt. Hetkel jääb see aga edasiarendusse.

Kasutatud kirjandus

[1] „BookFinder.com,“ AbeBooks (Amazon), 30 Jaanuar 1997. [Võrgumaterjal]. Available: https://www.bookfinder.com/?ref=bf_s1_hd_1. [Kasutatud 6 Märts 2022].

[2] „AddAll,“ [Võrgumaterjal]. Available: https://www.addall.com/. [Kasutatud 6 Märts 2022].

[3] T. Contributor, „SearchSoftwareQuality,“ 01 August 2019. [Võrgumaterjal]. Available: https://searchsoftwarequality.techtarget.com/definition/Web-application-Web-app. [Kasutatud 12 Märts 2022].

[4] A. Horiachko, „Softermii,“ Softermii, 27 Oktoober 2021. [Võrgumaterjal]. Available: https://www.softermii.com/blog/10-tips-in-choosing-the-best-tech-stack-for-your-web-application. [Kasutatud 12 Märts 2022].

[5] S. Overflow, „2021 Developer Survey,“ Stack Overflow, 2021.

[6] Oracle, „Oracle Java documentation,“ Oracle, [Võrgumaterjal]. Available: https://docs.oracle.com/javase/8/docs/technotes/guides/language/index.html. [Kasutatud 12 Märts 2022].

[7] Microsoft, „Microsoft docs,“ Microsoft, 30 November 2021. [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/. [Kasutatud 12 Märts 2022].

[8] K. Chris, „freeCodeCamp,“ freeCodeCamp, 30 August 2021. [Võrgumaterjal]. Available: https://www.freecodecamp.org/news/what-is-php-the-php-programming-language-meaning-explained/. [Kasutatud 12 Märts 2022].

[9] „Pythond docs,“ [Võrgumaterjal]. Available: https://www.python.org/doc/essays/blurb/. [Kasutatud 12 Märts 2022].

[10] „Nodejs,“ [Võrgumaterjal]. Available: https://nodejs.dev/learn/introduction-to-nodejs. [Kasutatud 12 Märts 2022].

[11] „Student Scholarships,“ [Võrgumaterjal]. Available: https://studentscholarships.org/articles/22/how-hard-is-it-to-learn-java-learn-through-our-java-tutorial. [Kasutatud 12 Märts 2022].

[12] M. Aina, „Career Karma,“ 08 Jaanuar 2022. [Võrgumaterjal]. Available: https://careerkarma.com/blog/is-c-sharp-hard-to-learn/. [Kasutatud 12 Märts 2022].

[13] „ILoveLanguages,“ [Võrgumaterjal]. Available: https://www.ilovelanguages.com/is-php-a-hard-language-to-learn/. [Kasutatud 12 Märts 2022].

[14] Thinkful, „Thinkful,“ [Võrgumaterjal]. Available: https://www.thinkful.com/blog/how-hard-is-it-to-learn-python/. [Kasutatud 12 Märts 2022].

[15] A. Cassity, „Scholarlyoa,“ 15 November 2021. [Võrgumaterjal]. Available: https://scholarlyoa.com/how-difficult-is-node-js-for-beginner/. [Kasutatud 12 Märts

2022].

[16] R. patel, „mindInventory,“ [Võrgumaterjal]. Available: https://www.mindinventory.com/blog/best-backend-frameworks/. [Kasutatud 13 Märts 2022].

[17] R. Shankar, „hackr.io,“ 30 November 2021. [Võrgumaterjal]. Available: https://hackr.io/blog/java-frameworks. [Kasutatud 13 Märts 2022].

[18] K. Gupta, „FreelancingGig,“ 26 Aprill 2018. [Võrgumaterjal]. Available: https://www.freelancinggig.com/blog/2018/04/26/spring-popular-java-frameworks/. [Kasutatud 13 Märts 2022].

[19] C. Brewster, „TRIO,“ [Võrgumaterjal]. Available: https://trio.dev/blog/front-end-technologies. [Kasutatud 13 Märts 2022].

[20] „airfocus,“ [Võrgumaterjal]. Available: https://airfocus.com/glossary/what-is-a-front-end/. [Kasutatud 13 Märts 2022].

[21] „angular.io,“ [Võrgumaterjal]. Available: https://angular.io/guide/what-is-angular. [Kasutatud 13 Märts 2022].

[22] „reactjs,“ [Võrgumaterjal]. Available: https://reactjs.org/docs/getting-started.html. [Kasutatud 13 Märts 2022].

[23] „Vue.js,“ [Võrgumaterjal]. Available: https://vuejs.org/guide/introduction.html. [Kasutatud 13 Märts 2022].

[24] „GeeksforGeeks,“ 23 Detsember 2020. [Võrgumaterjal]. Available: https://www.geeksforgeeks.org/how-to-choose-the-right-database-for-your-application/. [Kasutatud 19 Märts 2022].

[25] „Matillion,“ 18 November 2020. [Võrgumaterjal]. Available: https://www.matillion.com/resources/blog/the-types-of-databases-with-examples. [Kasutatud 19 Märts 2022].

[26] „Statistics&Data,“ Mai 2021. [Võrgumaterjal]. Available: https://statisticsanddata.org/data/the-most-popular-databases-2006-2021/. [Kasutatud 19 Märts 2022].

[27] A. Mehta, „appinventiv,“ 8 Märts 2022. [Võrgumaterjal]. Available: https://appinventiv.com/blog/top-web-app-database-list/. [Kasutatud 19 Märts 2022].

[28] M. Kamaruzzaman, „towardsdatascience,“ 20 Jaanuar 2021. [Võrgumaterjal]. Available: https://towardsdatascience.com/top-10-databases-to-use-in-2021-d7e6a85402ba. [Kasutatud 19 Märts 2022].

[29] L. Harukushko, „Yalantis,“ [Võrgumaterjal]. Available: https://yalantis.com/blog/how-to-choose-a-database/. [Kasutatud 19 Märts 2022].

[30] „github,“ [Võrgumaterjal]. Available: https://github.com/about. [Kasutatud 13 Märts 2022].

[31] „About Gitlab,“ [Võrgumaterjal]. Available: https://about.gitlab.com/company/. [Kasutatud 13 Märts 2022].

[32] A. Kumar, 09 Veebruar 2022. [Võrgumaterjal]. Available: https://disbug.io/en/blog/github-vs-gitlab-vs-bitbucket. [Kasutatud 13 Märts 2022].

[33] „InterviewBit,“ 5 Oktoober 2021. [Võrgumaterjal]. Available: https://www.interviewbit.com/blog/intellij-vs-eclipse/. [Kasutatud 13 Märts 2022].

[34] „Robert Cooper,“ 25 Juuli 2021. [Võrgumaterjal]. Available: https://robertcooper.me/post/vscode-vs-webstorm. [Kasutatud 13 Märts 2022].

[35] N. Suryavanshi, „medium,“ 8 November 2017. [Võrgumaterjal]. Available: https://medium.com/@nnilesh7756/what-are-cloud-computing-services-iaas-caas-paas-faas-saas-ac0f6022d36e. [Kasutatud 19 Märts 2022].

[36] Piotr, „Brainhub,“ 27 Juuli 2021. [Võrgumaterjal]. Available: https://brainhub.eu/library/cloud-architecture-saas-faas-xaas/. [Kasutatud 19 Märts 2022].

[37] „scand,“ 26 Juuni 2020. [Võrgumaterjal]. Available: https://scand.com/company/blog/pros-and-cons-of-using-spring-boot/. [Kasutatud 20 Märts 2022].

[38] „spring,“ [Võrgumaterjal]. Available: https://spring.io/projects/spring-boot. [Kasutatud 20 Märts 2022].

[39] „jsoup,“ [Võrgumaterjal]. Available: https://jsoup.org/. [Kasutatud 2 Aprill 2022].

[40] K. P, „Towards Data Science,“ 4 August 2020. [Võrgumaterjal]. Available: https://towardsdatascience.com/web-scraping-with-selenium-d7b6d8d3265a. [Kasutatud 2 Aprill 2022].

[41] „HtmlUnit,“ 19 Märts 2022. [Võrgumaterjal]. Available: https://htmlunit.sourceforge.io/. [Kasutatud 2 Aprill 2022].

[42] „RedHat,“ 8 Mai 2020. [Võrgumaterjal]. Available: https://www.redhat.com/en/topics/api/what-is-a-rest-api. [Kasutatud 26 Märts 2022].

[43] „Spring Security,“ [Võrgumaterjal]. Available: https://spring.io/projects/spring-security. [Kasutatud 26 Märts 2022].

[44] M. Behler, „Macrobehler,“ 21 August 2020. [Võrgumaterjal]. Available: https://www.marcobehler.com/guides/spring-security. [Kasutatud 26 Märts 2022].

[45] „Tutorialspoint,“ [Võrgumaterjal]. Available: https://www.tutorialspoint.com/spring_security/spring_security_with_jwt.htm. [Kasutatud 26 Märts 2022].

[46] „GeeksforGeeks,“ 23 Juuli 2020. [Võrgumaterjal]. Available: https://www.geeksforgeeks.org/what-is-spa-single-page-application-in-angularjs/. [Kasutatud 27 Märts 2022].

[47] „Introduction to Angular concepts,“ [Võrgumaterjal]. Available: https://angular.io/guide/architecture. [Kasutatud 27 Märts 2022].

[48] „Angular Material,“ [Võrgumaterjal]. Available: https://material.angular.io/. [Kasutatud 27 Märts 2022].

[49] K. Omeri, „Better Programming,“ 17 Veebruar 2020. [Võrgumaterjal]. Available: https://betterprogramming.pub/how-to-deploy-your-angular-9-app-to-heroku-in-minutes-51d171c2f0d. [Kasutatud 27 Märts 2022].

[50] „Java in use,“ [Võrgumaterjal]. Available: https://www.javainuse.com/spring/ang7-jwt. [Kasutatud 27 Märts 2022].

[51] O. Mikhalchuk, „Forte Group,“ 10 Detsember 2020. [Võrgumaterjal]. Available: https://fortegrp.com/the-importance-of-unit-testing/. [Kasutatud 3 Aprill 2022].

[52] S. G. d. Rosa, „Medium,“ 30 November 2017. [Võrgumaterjal]. Available: https://medium.com/swlh/angular-unit-testing-jasmine-karma-step-by-step-e3376d110ab4. [Kasutatud 03 Aprill 2022].

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

Mina, Rasmus Janson

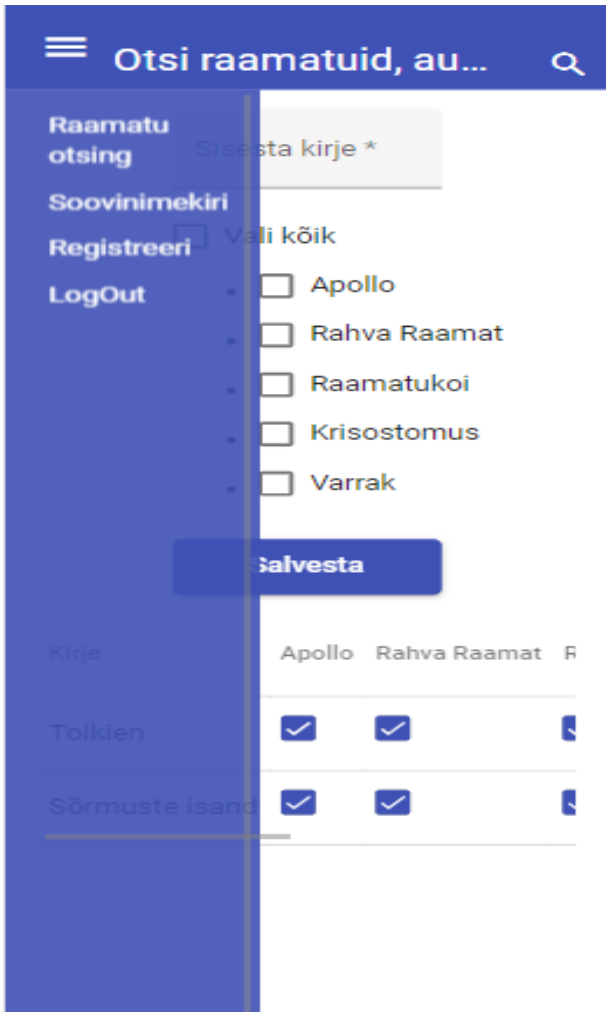
1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Raamatute otsimise ja hinnavõrdluse veebirakenduse arendus“, mille juhendaja on Kersti Antoi
   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse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

25.04.2022

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

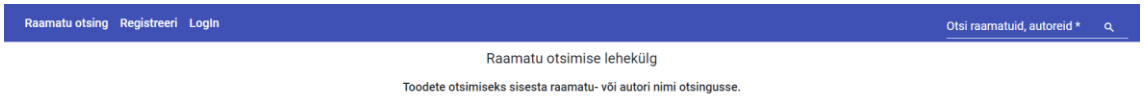
Lisa 2 – Nutiseadme brauseri vaated


Otsi raamatuid, autoreid *
tolkien
Sisesta kirje *
Vali kõik
Apollo
Rahva Raamat
Raamatukoi
Krisostomus
Varrak
Salvesta
Kirje
Apollo
Rahva Raamat
Tolkien
Sõrmuste isand


Otsi raamatuid, au...
Raamatu otsing
Soovinimekiri
Registreeri
LogOut
Sisesta kirje *
Vali kõik
Apollo
Rahva Raamat
Raamatukoi
Krisostomus
Varrak
Salvesta
Kirje
Apollo
Rahva Raamat
Tolkien
Sõrmuste isand


Otsi raamatuid, au...
Email
Parool
Logi sisse

Lisa 3 – Esilehe ja progressiooni laadimisrõnga näitamine



Raamatu otsimise lehekülg

Toodete otsimiseks sisesta raamatu- või autori nimi otsingusse.

Palun oodake! Laeme raamatuid.