Tallinna Ülikool

Digitehnoloogiate Instituut

Ettevõtete veebiprofiilide automaatseks võrdlemiseks tarkvararakenduse loomine

Bakalaureusetöö

Autor: Madis Uibo

Juhendaja: Jaagup Kippar

Autor: ..... „ ..... „ 2018

Juhendaja: ..... „ ..... „ 2018

Instituudi direktor: ..... „ ..... „ 2018

Tallinn 2018

Autorideklaratsioon

Deklareerin, et käesolev bakalaureusetöö on minu töö tulemus ja seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on viidatud.

.....	.....
(kuupäev)	(autor)

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina Madis Uibo (sünnikuupäev: 18.01.1994)

1. Annan Tallinna Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Ettevõtete veebiprofiilide automaatseks võrdlemiseks tarkvararakenduse loomine“, mille juhendaja on Jaagup Kippar, säilitamiseks ja üldsusele kättesaadavaks tegemiseks Tallinna Ülikooli Akadeemilise Raamatukogu repositooriumis.
2. Olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest tulenevaid õigusi.

Tallinnas, _____

(digitaalne) allkiri ja kuupäev

Sisukord

Sõnastik

API	*Application Programming Interface*, rakendusliides
CAPTCHA	*Completely Automated Public Turing test to tell Computers and Humans Apart* (Gil, 2018) – Täiesti automatiseeritud avalik Turingi test, millega eristatakse arvuteid inimestest
CSS	*Cascading Style Sheets*, veebilehtede kujunduseks kasutatav märgistuskeel
CSV	*Comma-seperated values*, komaga eraldatud väärtused
DOM	*Document Object Model*, veebilehe dokumendi objektmudel
HTML	*Hypertext Markup Language*, hüperteksti märgistuskeel ehk dokumendi paigutuse keel
JSON	*JavaScript Object Notation*, JavaScript'i andmete esituse formaat
Jquery	JavaScript'i raamistik
PDF	*Portable Document Format*, teisaldatav dokumendi formaat (Hüüs, 2009)
SQL	*Structured Query Language*, struktuurpäringukeel
URL	*Uniform Resource Locator*, internetiaadress

Sissejuhatus

Automatiseerimine toob märkimisväärse kasu ettevõtetele ja majandusele, kuid see on protsess, mida ühe öö jooksul täide ei viida (McKinsey Global Institute, 2017). Paljud tarkvaraarendusega tegelevad ettevõtted just keskenduvadki automatiseerimisele, et leida lahendusi klientidele, kes soovivad oma tööd lihtsustada ning keskenduda kasulikumatele tegevustele.

Käesoleva töö eesmärgiks on koostada rakendus, mis aitaks kasutajal leida teatud ettevõtte veebiprofiili põhjal sarnaseid ettevõtteid. Rakendusel peab olema minimaalne kuid kasutajasõbralik kasutajaliides, kuna tulevased kasutajad kasutavad rakendust ärilistel eesmärkidel ning ei pruugi omada teadmisi programmeerimisest.

Töö jaguneb kolmeks peatükiks. Esimeses peatükis koostab autor lühikese ülevaate rakendustest, mis oma töö poolest on sarnased. Autor uurib kuidas need rakendused töötavad ning milliseid teadmisi oma rakenduse prototüübis kasutada saaks. Esimese peatüki teises osas kirjeldatakse vahendeid, mida arendatavas rakenduses kasutatakse.

Teises peatükis kirjeldatakse rakenduse prototüübi arendust: milliseid ülesandeid rakendus täitma peab, kuidas rakenduse parameetreid konfigureerida, millistest osadest kasutajaliides koosneb ning antakse ülevaade rakenduse keerulisematest taustaprotsessidest.

Kolmandas peatükis koostas autor testilood rakenduse erinevate osade tööst ning andis rakenduse kohta hinnangu, mis tingimustel prototüüp annab paremaid tulemusi ning mis tingimustel esineb takistusi prototüübi töös.

Töö teema pakkus autorile juhendaja, kelle tuttav edastas talle rakenduse idee, et tõhustada eksportmüügi teostamist.

1 Eeluuring

Käesoleva peatüki esimeses osas on välja toodud oma töö poolest sarnased rakendused. Autor uurib kuidas need rakendused töötavad ning milliseid uusi teadmisi nende kasutamisega sai. Teises osas on nimetatud peamised vahendid, mida arendatavas rakenduses kasutusele võetakse.

1.1 Sarnaste eesmärkidega rakendused

Käesolevas alapeatükis on loetletud oma töö ja ehituse poolest sarnased rakendused. Autor uurib kuidas need rakendused töötavad ning milliseid uusi teadmisi nende kasutamisega sai.

1.1.1 Webscraper.io brauseri laiendus

Webscraper.io (edaspidi Web Scraper) on Google Chrome ja Mozilla Firefox brauserite laiendus, mis võimaldab andmete kraapimist erinevatelt veebilehtedelt. Antud laiendusega saab luua plaani (ingl. *sitemap*), millega kasutaja kirjeldab, mis viisil veebisait läbitakse ning millised andmed sealt kogutakse. Kasutades neid plaane, Web Scraper läbib antud lehe vastavalt ja kraabib kõik valitud andmed. Kraabitud andmeid on hiljem võimalik alla laadida CSV formaadis. (Balodis, 2018)

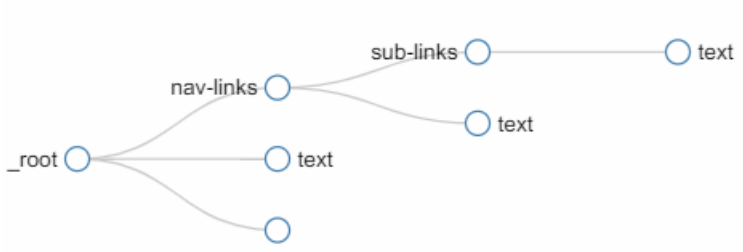
Lisaks tasuta Chrome laiendile pakub Web Scraper ka pilve-teenusena töötavat veebikraapijat, mis võimaldab suuremahulist andmete kogumist, mitme veebikraapija isendi käivitamist ning neid on võimalik käivitada ajakava järgi. Pilve-teenusena töötava veebikraapija miinuseks on maksumus – plaane saab küll tasuta testida, kuid andmete kogumiseks tuleb maksta vastav summa, mis on Web Scraper'i kodulehel[1] kirjeldatud.

Tarkvaraarendaja vaates on Web Scraper'it lihtne kasutada. Olles installinud laiendi, on seda võimalik kasutada arendaja tööriistade aknas (ingl. *developer tools*), mille jaoks tekib eraldi vahekaart. Selles vahekaardis on võimalik luua eelnevalt kirjeldatud plaan, andes parameetriks veebilehe URL-i. Järgmisena on võimalik lisada soovitud

[1] https://www.webscraper.io/service

HTML elementide selektorid (ingl. *selector*), mis on kasutajale võimalikult lihtsaks tehtud, kuna neid saab valida brauseri poolt kuvatud kasutajaliideses. Graafilise selektori kasutamisel on üks miinus, kui vaja on töötada hõljuknuppudega – need ei taha lehekülgedel avaneda kui selektor on kasutaja poolt kasutuses ning selle tõttu ei saa soovitud elemente valida. Kui kasutaja oskab sobivad HTML elemendid tekstiliselt lisada, siis see probleemi ei tekita. Lisaks võimaldab antud laiend ka regulaaravaldiste kasutamist. Tervikliku teksti saab jaotada sõnadeks, tähtedeks või millekski muuks, kui regulaaravaldis seda lubab.

Loodud plaanist on võimalik graaf koostada, mis annab üldise pildi sellest, kuidas veebilehti läbitakse ja kust andmeid kraabitakse (Joonis 1).



Joonis 1 Web Scraper plaani graaf

1.1.2 ParseHub

ParseHub on rakendus, mida saab kasutada keeruliste veebikraapijate koostamiseks ilma koodi kirjutamata. Tegemist on iseseisva tarkvaraga, mida on võimalik installida erinevatele operatsioonisüsteemidele nagu Windows, MacOS ja Linux. Rakendusele on sisse ehitatud brauser, mida kasutatakse andmete kraapimisel ning seetõttu ei sõltu teiste brauserite tööst. ParseHub'il on olemas nii tasuta versioon, mida kasutajad saavad esialgseks testimiseks proovida ning mitu tasulist plaani, millega saab tutvuda rakenduse kodulehel[2].

Rakenduse esialgsel avamisel on vaja registreerida kasutaja, kuhu hiljem veebikraapija projektid lisatakse. ParseHub'i projekt on sarnane webscraper.io plaaniga, kus esmalt antakse projekti parameetriks veebilehe URL, mida projekti siseselt kasutatakse. Pärast projekti loomist saab HTML selektoreid kasutama hakata. Selektorid on klassifitseeritud kolmeks erinevaks tüübiks: *select* käsuga saab valida

[2] https://www.parsehub.com/pricing

ühe või mitu elementi ning automaatselt kraabib andmed kui võimalik, *relative select* käsuga on võimalik eelnevalt valitud elemente seostada teiste elementidega ja võimaluse korral kraabib andmed automaatselt ning *click* käsk vajutab eelnevalt selekteeritud elemendi peale, mida on hea kasutada näiteks töötamiseks rippmenüüdega.

ParseHub'il on kraapimise seadistuseks rohkem valikuid kui Web Scraper'il. *Load JavaScript & Images* valik annab veebikraapijale teada, kas oodata JavaScript'i ja piltide laadimist enne kui andmeid koguma hakatakse. *Rotate IP Addresses* valikuga on võimalik kraapija IP aadressi muuta, mida on hea kasutada juhul kui veebisait, millelt andmeid kraabitakse on ParseHub serverid ära blokeerinud. *Max Workers* arvu suurendamisel saab lisada veebikraapija isendeid juurde, millega saab andmete kraapimise protsessi kiirendada. ParseHub'i toe lehel[3] on kirjas, et üks isend läbib umbes viis lehte minutis. *Max Pages* parameeter ütleb kraapijale, kui mitu veebilehte see läbida võib enne oma töö lõpetamist.

Üldiselt on ParseHub'i mugav kasutada, isegi kui kasutajal on veebilehtede arendusega vähe kogemust. Kuid vähese kasutusaja jooksul leidis autor rakenduses mitu defekti, näiteks sissejuhatuse läbimisel tekkis selline olukord, kus rakendus keelas ära kasutajal ükskõik millise nupu vajutuse.

1.1.3 Scraper API

Scraper API käsitleb vahendusservereid, brausereid ja CAPTCHA kontrolle sinu eest, et saaksid võimalikult lihtsa API päringuga kõik vajalikud HTML andmed soovitud veebilehelt (Scraper API, kuupäev puudub). Scraper API tasuta versiooniga on võimalik teha 1000 päringut iga kuu, kuid tihedamaks kasutuseks on vaja maksta igakuiselt sõltuvalt päringute arvust, mis on Scraper API kodulehel[4] välja toodud.

Nagu eelmiste veebikraapijatega, Scraper API on võimalikult kasutuskindlaks tehtud. IP blokeeringute vastu kasutab rakendusliides tuhandeid erinevaid vahendusservereid, et kasutaja soovitud andmed kätte saaks. API-le on ka sisse ehitatud automaatne CAPTCHA lahendaja, mille tõttu kasutaja ei pea muretsema, et mõni päring

[3] https://help.parsehub.com/hc/en-us/sections/360000197673-Features-Tools
[4] https://www.scraperapi.com/pricing

automaatkontrolli pärast ebaõnnestuks. JavaScript'i renderdamiseks on võimalik päringutele lisada parameeter, mis annab veebikraapijale teada, et enne töö algust ootaks ära JavaScript'i komponentide laadimise. Kuna API kasutab kaheksat erinevat interneti-teenuse pakkujat ja ülemaailmseid vahendusservereid, ei tohiks rakenduse kiirus sõltuda kasutaja asukohast.

Huvitaval kombel ei suutnud autor Scraper API-lt korrektset vastust saada. Kasutades Scraper API dokumentatsiooni, proovis autor pärida andmeid kasutades käsuviipa ja brauserit, kuid mõlemal juhul tagastati kogu veebilehe sisu. Lisaks eelnevale on Scraper API dokumentatsioon väga minimaalses vormis ning täpsustavaid andmeid sealt ei leia, nagu näiteks kuidas päringut täpsustada, milliseid andmeid leheküljelt kraapida jne. Tekib segadus, kas antud API ainult võimaldabki ühe veebilehe pealt kogu HTML sisu kraapimist, et kust kasutaja hiljem peab ise sobivad andmed välja filtreerima või kas liidesel on lisaks muud funktsionaalsust ka juures, mis võimaldaks valitud andmete kogumist ning töötamist veebisaidi erinevate lehekülgedega.

1.1.4 MetricSpot

MetricSpot on rakendus, millega on võimalik koostada analüüsi ja optimiseerimise hinnang veebisaidist, mille abil saab parendada veebisaidi struktuuri. Hinnangu tulemustele on võimalik juurde lisada oma logo ning seda on võimalik alla laadida PDF formaadis.

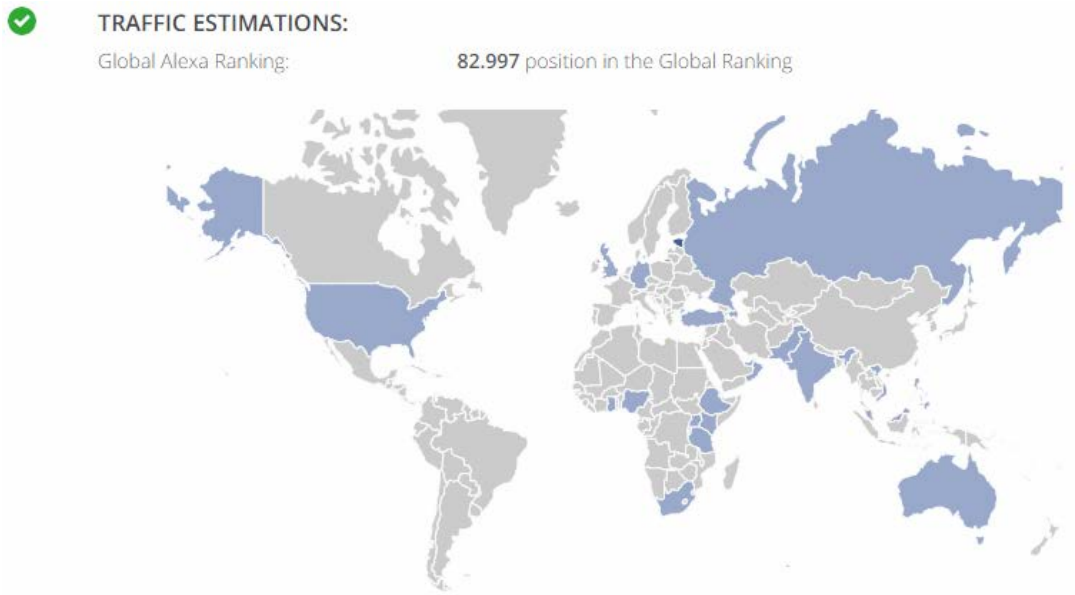
Kui osta tasuline versioon, pakub MetricSpot rohkem võimalusi. Kvaliteedi ja hinnangu järgi saab analüüsida 200 erinevat linki, mis viitavad sinu veebisaidile. Lisades oma veebisaidile märksõnu, saab MetricSpot rakendusega hinnata veebisaidi SERP[5] järku. Konkurentsi turustamisstrateegia analüüsimiseks on MetricSpot'il enda veebisaitide võrdlemise tööriist. Uute klientide meelitamiseks saab tasulise versiooniga koostada kohandatud SEO[6] aruandeid. (MetricSpot, kuupäev puudub)

MetricSpot rakenduse testimiseks kasutasin Tallinna Ülikooli lehekülge. Tulemused on kuvatud brauseris hästi loetaval kujul. Esmalt on välja toodud üldine hinnang

[5] SERP (*Search Engine Result Pages*) – Otsingumootori päringu vastuseks koostatud tulemuste leht
[6] SEO (*Search Engine Optimization*) – Protsess, mis mõjutab kui tihti on üks veebisait või veebileht otsingumootori tulemustes nähtav

protsentuaalselt järgmiste kategooriate järgi: SEO kompetentsus, elementaarne SEO, sisu, kasutatavus, tehnilised aspektid ja sotsiaalmeedia. Kategooriad on jaotatud eraldi sektsioonideks, kus iga kategooria kohta on toodud välja täpsemad detailid.

SEO kompetentsus kirjeldab väliste tegurite mõju veebisaidi järgule otsingumootorites. Selles sektsioonis on ka välja toodud ülemaailmne vaade, kus Tallinna Ülikooli lehekülge kõige tihedamini külastatakse (Joonis 2).



Joonis 2 Tallinna Ülikooli veebisaidi ülemaailmsed külastused

Elementaarse SEO sektsioon kirjeldab veebisaidil leiduvad faktoreid, mis samuti mõjutavad veebisaidi järku otsingumootorites. Need faktorid on seotud veebisaidi tehniliste omadustega, mida on võimalik lühiaja jooksul parendada. Näiteks Tallinna Ülikooli lehelt on puudu metaandmete kirjeldus ja märksõnad. MetricSpot'i soovituste järgi aitavad need kaks mainitud HTML elementi parendada veebisaidi otsitavust otsingumootorites.

Sisu sektsioonis on andmed veebisaidi struktuuri kohta. Üldiselt on seal kirjeldatud milliseid HTML elemente võiks veebisaidil vähem olla, kas ettevõttega seotud blogi on leitav ning millised märksõnad tihti korduvad.

Kasutatavuse sektsioonis on andmed veebisaidi optimaalsuse kohta. Tallinna Ülikooli leht sai kasutatavuse sektsioonis hea tulemuse, sest veebisaidil on olemas üldine vealeht, otsingumootor ning seda on mugav mobiilsete seadmetega kasutada. Ainuke

takistus oli allalaadimise kiirus, aga kuna MetricSpot'i server asub Põhja-Ameerikas, siis võib järeldada, et selle põhjuseks on serveri füüsiline asukoht.

Isegi kui lähtekood pole nähtav kasutajale, omab see suurt mõju veebisaidi kiiruses ja jõudluses ning neid kahte omadust kasutavad otsingumootorid veebisaidi hindamisel (MetricSpot, kuupäev puudub). Tehnilise aspektide sektsioon kirjeldab just neid omadusi ning lisaks kuvatakse veebisaidi andmeid turvalisuse, HTML korrektsuse, serveri asukoha ja kasutatud tehnoloogiate kohta.

1.2 Töö tarbeks valitud vahendid

Käesolevas peatükis on kirjeldatud peamised vahendid, mida rakenduse arendamisel on kasutatakse. Lisaks nimetatud tehnoloogiatele on arenduse käigus võetud kasutusele teisi vahendeid, sest arenduse käigus tihtipeale selgub, et midagi on esialgsest planeerimisest puudu jäänud.

1.2.1 Java

Autor on oma seminaritöös (Uibo, 2018) koostanud lühikese kirjeldava alapeatüki Java kohta. Java on 1995. aastal välja lastud objektorienteeritud programmeerimiskeel, mis oli esialgselt James Goslingu algatatud ettevõtte Sun Microsystems poolt arendatud. Java põhineb ideel „*Write Once, Run Anywhere*“, tagades tõhusa käitusaja populaarsetel platvormidel. 13. november 2006. aastal Sun avalikustas suurema osa Java lähtekoodist. 8. mail 2007. aastal avalikustati kogu Java kood, välja arvatud väike osa sellest, mille üle õigusi ei omatud. (Tutorialspoint, 2017). 2010. aastal Oracle Corporation soetas Sun Microsystems'i ning sellest saadik on nad Java koodi edasi arendanud (Oracle Corporation, 2010).

Java kui programmeerimiskeelel on mitmeid kasutuseelised. Java on kavandatud lihtsaks kasutuseks ja seetõttu on seda kergem kirjutada, kompileerida, siluda ja õppida kui teisi programmeerimiskeeli. Objektorienteeritud ehituse tõttu on võimalik luua modulaarseid programme ja taaskasutatavat koodi. Java on platvormist sõltumatu, mille tõttu on võimalik kolida kergelt ühelt operatsioonisüsteemilt teise peale (IBM, 2017).

1.2.2 Spring MVC

Rakenduse üldraamistikuks otsustas autor kasutada Spring MVC raamistikku. Spring MVC on avatud lähtekoodiga Java raamistik, mille abil on võimalik luua veebirakendusi lihtsalt ja kiirelt. Hetke seisuga on mõistlik kasutada Spring MVC versiooni 5 ja Java versiooni 8, kuna Spring MVC 5 on loodud kasutamaks Java 8. Isegi kui veebiraamistikul on Java 9 ühilduvus lisatud, on see varajase ligipääsu seisundis ning seetõttu võib rakenduse arengu ajal esineda palju vigu. Autoril on varasem kogemus Spring tehnoloogia kasutamisega ning see võib rakenduse arenduse ajal kasuks tulla.

1.2.3 Apache Tomcat

Apache Tomcat on avatud lähtekoodiga veebiserver Java rakendusteks. Autor otsustas kasutada Tomcat'i, kuna see on paindlik, usaldusväärne, kergekaaluline ning lihtsalt konfigureeritav (Davis, 2014). Rakendust arendatakse Tomcat versioon 9 veebiserveris.

1.2.4 Apache Maven

Projekti haldamiseks kasutatakse Apache Maven'i, mille abil saame lisada projektile kõik vajalikud raamistikud ja andmeteegid. Lisaks on Maven'il moodul, mis kompileerib kogu projekti üheks paketiks, mille saab hiljem veebiserverile paigaldada.

1.2.5 PostgreSQL

Andmete salvestamiseks kasutatakse PostgreSQL versiooni 9.5. PostgreSQL on avatud lähtekoodiga objektrelatsiooniline andmebaasi süsteem, mis on viimased 30 aastat olnud aktiivses arenduses ning teeninud hea maine usaldusväärsuse, stabiilsuse ja jõudluse poole pealt (PostgreSQL, kuupäev puudub). Nagu ka Java, jookseb PostgreSQL erinevatel operatsioonisüsteemidel ilma takistusteta.

1.2.6 Hibernate

Hibernate on objektrelatsiooniline vastendamisraamistik ehk ORM, mille eesmärgiks on arenduse hõlbustamine. Lauri Elias (Elias, 2011) kirjutas oma seminaritöös, et Hibernate kasutab POJO (ingl. *Plain Old Java Object*) mudelit, et vastendada Java objektid relatsioonilisse andmebaasi, mille läbi on tagatud objektide püsiesitus. Püsiesitatavus võimaldab objektil säilida ka pärast seda, kui protsess, mis ta tekitas, on lõppenud (Christian Bauer, 2005). Võib ka öelda, et Hibernate on sild Java objektide ja andmebaasi vahel (Joonis 3).

Joonis 3 Hibernate ühendus Java objektide ja andmebaasi vahel (Tutorialspoint, kuupäev puudub)

Hibernate eelised (Tutorialspoint, kuupäev puudub):

- Hibernate kujutab Java objektid relatsioonilisteks andmemudeliteks
- Andmebaasi skeemi muudatustega tuleb ainult muuta Hibernate'i konfiguratsiooni
- Java objektidega töötamiseks ei ole vaja koodis kasutada SQL andmetüüpe, muutujate põhjal oskab Hibernate ise määrata, mis andmetüüpidega on tegemist
- Hibernate ei vaja töötamiseks rakenduslikku serverit
- Keerukate objektide vaheliste ühenduste loomiseks saab kasutada erinevaid annotatsioone, lihtsustades programmerija tööd
- Andmete pärimiseks saab kasutada erinevaid strateegiaid, mille abil optimiseeritakse programmi tööd

1.2.7 Jsoup

Et elimineerida võimalikud sõltuvused maksustatud veebikraapija API-st, otsustas autor kasutada andmete kraapimiseks Java andmeteeki Jsoup, mis võimaldab HTML elementide töötlemist. Jsoup varustab rakenduse mugava API-ga, mida saab kasutada andmete kogumiseks ja käsitlemiseks, kasutades parimaid DOM, CSS ja Jquery sarnaseid meetodeid. Jsoup liidestub WHATWG HTML5[7] spetsifikatsiooni ja parsib HTML-i sama DOM-i peale (Jsoup, kuupäev puudub). Sama põhimõttega töötavad ka tänapäeva brauserid. Jsoup'i kasutades suudab rakendus koguda veebisaitidelt märksõnu, mida hiljem kasutatakse sarnaste veebisaitide leidmiseks.

1.2.8 Google Custom Search JSON API

Google Custom Search JSON API võimaldab rakendusel programmeeritult pärida Google otsingumootori tulemusi. Teiseks võimaluseks oleks kasutada Jsoup'i enda funktsionaalsust ning kraapida andmed Google otsingumootori kasutajaliidese pealt, kuid seal võivad tekkida probleemid CAPTCHA kontrollidega. Nende kontrollide vältimiseks ning mugavuse tõttu on mõistlikum kasutada Google poolt pakutud API-t. Custom Search API miinuseks on päringute arv – tasuta versiooni kasutamiseks on ühe päeva jooksul lubatud 100 otsimispäringut. Tihedamaks kasutuseks on vaja Google konsoolis lisada arvelduskonto, millelt laekuvad maksed sõltuvalt päringute arvuga.

1.2.9 Bootstrap

Kasutajasõbraliku ja minimaalse kasutajaliidese loomiseks kasutatakse rakenduses Bootstrap'i. Bootstrap on üks populaarsematest HTML, CSS ja JavaScripti raamistikest, mida on mugav kasutada adaptiivsete veebilehtede loomisel.

[7] https://html.spec.whatwg.org/multipage/

2 Rakenduse arendus

Käesolevas peatükis on kirjeldatud kuidas uurimistöö jaoks rakenduse prototüüpi arendati. Igas alapeatükis on kirjeldatud rakenduse erinevaid komponente ning nende loogikat. Rakenduse koodidetailidele tähelepanu pööratud ei ole, kuid soovi korral on võimalik rakenduse lähtekood leida autori koodirepositooriumist https://github.com/madis121/CompanyWebProfileComparator.

2.1 Rakenduse struktuur

Käesolevas alapeatükis kirjeldatakse rakenduse struktuuri, kuidas failid on jaotatud erinevatesse kaustadesse ning kuidas need omavahel seoses on.

Juurkaustas leidub src kaust, kus asub POM ehk *Project Object Model* ja rakenduse koodi kaust. POM-is on informatsioon selle kohta, kuidas Maven rakendust kompileerib ning mis raamistikud ja teegid rakenduse töötamiseks vajalikud on. Koodi kaust jaguneb kolmeks erinevaks osaks:

- `src/main/java` – Siin asuvad projekti Java koodifailid, mis on esialgselt jaotatud pakettideks ning paketid omakorda jaotuvad erinevateks Java klassideks ja liidesteks.
- `src/main/resources` – Siin asub rakenduse konfiguratsioonifail, logimisteegi Logback konfiguratsioonifail ning tõlketekstide kogumikud.
- `src/main/webapp/WEB-INF` – Siin asuvad kasutajaliidese poolt kasutatud vaadete koodifailid, vormingu ja ilme kirjeldamiseks CSS failid ning erinevad skriptifailid, mida kasutatakse kasutajaliidese poolseks programeerimiseks. Lisaks eelnevale on `WEB-INF` kausta lisatud andmebaasi skript, mida saab kasutada andmebaasi tabelite loomiseks.

2.2 Rakenduse konfigureerimine

Rakenduse konfigureerimiseks kasutatakse faili nimega `app.properties`. Konfiguratsioonis on võimalik muuta:

- Andmebaasi ühenduse andmeid
- Google Custom Search API parameetreid

Näidiskonfiguratsiooni leiab koodirepositooriumist. Parameetrite täpsemad kirjeldused on välja toodud järgmises tabelis (Tabel 1).

Tabel 1 Rakenduse konfiguratsiooni parameetrid

Parameeter	**Kirjeldus**
`postgresql.database.url`	Andmebaasi URL
`postgresql.database.username`	Andmebaasi kasutaja
`postgresql.database.password`	Andmebaasi kasutaja parool
`postgresql.database.schema`	Andmebaasis kasutatav skeem
`google.cse.id`	Google Custom Search Engine ID
`google.cse.api.request.url`	Google Custom Search API päringu URL
`google.cse.api.key`	Google Custom Search API võti
`google.cse.api.requests`	Tulemuste lehtede arv, mida Google Custom Search API tagastab

2.3 Rakenduse täidetavad ülesanded

Rakendus peab suutma täita järgmisi ülesandeid:

1. Kasutaja loob otsitava ideaalettevõtte profiili. Seda on võimalik teha automaatselt andes süsteemile sisendiks sarnaste ettevõtete veebisaitide aadressid. Süsteem teeb sellest kokku ühe profiili baseerudes korduvatele märksõnadele. Kasutaja saab seda profiili lõpus korrigeerida, lisada ja eemaldada märksõnu. Teiseks võimaluseks on kohe sobivad märksõnad ise sisestada, jättes vahele märksõnade kogumise protsessi.
2. Kasutaja valib sihtriigi ning mis taseme kontakte ta otsib: *chief executive, HR manager, CFO,* jne. Süsteem läbib sihtriigis asuvate ettevõtete veebilehti ja leiab üles need, mis kattuvad näiteks vähemalt 70% ulatuses ideaalse profiiliga.
3. Süsteem kraabib nendelt lehtedelt infot: veebiaadressi, ettevõtte nime, üldnumbri, sihtisiku nime ja ametikoha ning telefoninumbri.
4. Vastuseks loob rakendus tabeli, kus eesotsas on suurema ja allpool väiksema kattuvusega ettevõtted.

2.4 Rakenduse kasutajaliides

Käesolevas alapeatükis kirjeldatakse rakenduse kasutajaliidesega hõlmavaid komponente.

2.4.1 Üldine kujundus

Isegi kui kujundusele suurt tähelepanu ei pööratud, oleks kasutajatel mugav rakendust kasutada, kui kasutajaliides oleks võimalikult minimaalne ning arusaadav. Bootstrap'i raamistik pakub arendajale mugava kontrolli HTML elementide manipuleerimiseks. Selle asemel, et kulutada tunde CSS stiilide koostamiseks ja HTML elementide paigutamiseks, saab suurema töö tehtud Bootstrap'i poolt defineeritud stiiliklasside kasutamisega. Plaanis on ka kasutada modaalaknaid, mis on samuti Bootstrap'i raamistikku kaasatud ning nende kasutamiseks on vaja ainult paar rida HTML koodi kirjutada. Kuna Bootstrap on loodud adaptiivse kujunduse kasutamiseks, on enamik

elemente võimelised end erinevate ekraani resolutsioonidega ümber paigutama. Selle tulemusena saab luua lihtsa, kuid arusaadava kujunduse nii kasutajatele kui ka arendajale.

Rakenduse üldine välimus koosneb vasakust navigatsioonireast, kuhu lisatakse rakenduse erinevate vormide lingid ning päisest, kus on võimalik rakenduse keelt vahetada ning sätete vormi avada. Erinevate lehekülgede sisu mahutatakse nende kahe kasutajaliidese komponendi vahele.

2.4.2 Profiilide halduse vorm

Profiilide halduse leheküljel kuvatakse eelnevalt koostatud profiile. Profiilid koosnevad märksõnadest, mis on kas kasutaja enda poolt ise lisatud või kasutaja poolt sisestatud veebisaitide andmete kraapimise tulemusena saadud. Profiile on hiljem võimalik muuta või kustutada.

Uut profiili on võimalik luua all vasakus servas nähtava nupuga, mille vajutusel avaneb liideses uus modaalaken. Avanenud modaalaknas on kasutajal võimalik valida profiili loomiseks kas kasutada olemasolevaid veebisaite või alustada tühja profiiliga.

Olemasolevate veebisaitide kasutamisega läbib veebikraapija soovitud veebisaitide erinevad leheküljed ning kogub sealt enimkorduvad märksõnad, mis veebikraapija töö lõppedes lisatakse profiilile. Veebisaitide sisendväljadesse on lisatud regulaaravaldise kontrollid, et need vastaksid veebiaadressi kujule, sisaldades ka protokolli. Genereeritud profiili märksõnu on hiljem võimalik kasutaja poolt eemaldada või ise uusi juurde lisada. Tühja profiili valikuga kuvatakse koheselt märksõnade sisestamisvormi, jättes vahele veebikraapija töö protsessi. Soovi korral saab kasutaja salvestada või puhastada loodud profiili ning alustada algusest. Salvestamise korral peab profiil sisaldama vähemalt kolme märksõna.

2.4.3 Sarnaste ettevõtete otsingu vorm

Ettevõtete otsingu lehel saab eelnevalt koostatud profiilidele sarnaseid ettevõtteid leida. Olles valinud rippmenüüst ühe profiili ja vajutanud ava otsing nupule, kuvatakse kasutajale modaalakent, mis on sarnane profiili loomise vormile, kus on võimalik

märksõnu lisada või eemaldada, kuid lisaks sellele on kasutaja kohustatud täpsustama otsitava ettevõtete sihtriigi. Otsingu alustamiseks tuleb vajutada otsi nuppu ning protsessi ajal kuvatakse kasutajale kerivat animatsiooni.

Otsing koosneb kolmest protsessist: esmalt päritakse otsingumootori API-lt märksõnade järgi sarnaseid veebisaite. Tagastatud veebisaidid töötleb veebikraapija sarnaselt, mida tehakse uue profiili loomise ajal. Viimase protsessina võrreldakse ükshaaval otsingumootori API-lt saadud veebisaitide märksõnu otsingus kasutatud profiili märksõnadega.

Kui rakendus on suutnud otsingu lõpetada, kuvatakse kasutajale modaalaknas tabelit pealkirjaga „Otsingu detailid“, kus on välja toodud loend ettevõtete veebisaitidest, mis otsingumootori vastusest saadi. Loend on järjestatud kahanevas järjekorras, ehk kõige rohkem kattuvad ettevõtted on tabeli eesotsas. Otsingu tulemustele on saab anda nimetuse ning andmete talletamiseks on vaja see salvestada. Salvestamise järgselt suunatakse kasutaja tulemuste vormile.

2.4.4 Tulemuste vorm

Tulemuste vormil kuvatakse varasemalt salvestatud otsingu tulemuste loendit. Tulemuste detailandmete vaatamiseks on võimalik avada modaalaken, kus kuvatakse kõikide veebisaitide aadresse, mis olid otsingus kasutatud profiiliga sarnased. Otsingu tulemusi on võimalik kasutajaliideses kustutada.

2.4.5 Sätete vorm

Sätete vormil saab kasutaja muuta veebikraapija parameetreid. Veebikraapija parameetrid on kirjeldatud järgmises tabelis (Tabel 2).

Tabel 2 Veebikraapija parameetrid

Parameeter	**Kirjeldus**
Ühe veebisaidi otsitavate lehtede arv	Kui mitu lehekülge veebikraapija ühel veebisaidil läbib
Märksõna minimaalne pikkus	Kui märksõna on antud arvust väiksem või võrdne, siis veebikraapija ei lisa seda tulemusse
Eiratud HTML elemendid	Komaga eraldatud elementide loend, mida veebikraapija eirab
Eiratud märksõnad	Komaga eraldatud sõnade loend, mida veebikraapija tulemustes ei kuvata

2.5 Keerulisemad taustaprotsessid

Käesolev alapeatükk kirjeldab rakenduse keeruliste taustaprotsesside tööd.

2.5.1 Veebikraapija

Veebikraapija on loodud Jsoup andmeteegi põhjal. Üks veebikraapija isend sõltub viiest parameetrist:

1. Loend - Veebisaitide aadressid
2. Täisarv - Ühe veebisaidi läbimise maksimaalne arv enne kui järgmist veebisaiti külastama hakatakse
3. Täisarv – Ühe märksõna minimaalne pikkus
4. Loend - Eiratud HTML elemendid
5. Loend - Eiratud märksõnad

Veebikraapija töö toimub rekursiivselt. Esmalt võetakse veebisaitide aadresside loendist esimene URL ning kontrollitakse, kas selles leidub üleliigseid parameetreid, mis võivad veebikraapija tööd kas aeglustada või takistada.

Veebikraapija tööd aeglustavaks URL-i parameetriks on näiteks numbrisümbol #, mida veebilehe linkidel kasutatakse sama lehe teise sektsiooni viimiseks. Numbrisümbol otseselt veebikraapija tööd ei takista, kuid tekib võimalus, et leitakse korduvaid linke, mis on loodud kasutajatele kergemaks navigeerimiseks ühel leheküljel. Seda lehekülge hakatakse korduvalt külastama ning selle tõttu lõpptulemuse andmed muutuksid ebatäpseks.

Veebikraapija tööd takistavateks elementideks on ohtlikud tähemärgid veebilehe URL-is. Jeff Star'i artiklis „*(Please) Stop Using Unsafe Characters in URLs*" on välja toodud hulk tähemärke, mis pole veebistandardite järgi ohutud ning tuleks kindlasti ära kodeerida. Need tähemärgid tekitavad rakenduses turvaaugu, mida võivad pahatahtlikud tegurid ära kasutada (Star, kuupäev puudub).

Kui töötlusse võetav URL on ära puhastatud, kogutakse sealt kõik lingid ning lisatakse hulk (ingl. *set*) tüüpi kollektsiooni. *Set* kollektsiooni eeliseks on see, et sinna ei ole võimalik lisada duplikaate. Koos URL-i puhastamise ja *Set* kollektsioon kasutusega eemaldatakse võimalus, et ühte kindlat veebilehte ei külastata rohkem kui ühe korra.

Järgmise astmena korjatakse lehekülje keha sisu, millest välja jäetakse need HTML elemendid, mis veebikraapija isendi loomise ajal üheks parameetriks antud oli. Kuna arendatavas rakenduses ei ole kasutajal võimalik ise täpsustada, milliseid HTML elemente täpselt valida, tekib võimalus, et keha elemendis võib leida JavaScript'i ja teisi ebavajalikke komponente. Selle parameetri abil eiratakse neid elemente ning tulemuseks saadakse täpsemad andmed. Leheküljelt kogutud tekstid jaotatakse märksõnadeks ning kontrollitakse kas nende pikkus on suurem kui veebikraapija isendi üheks parameetriks määratud arv. Kui tingimus on täidetud, siis need märksõnad lisatakse loendisse, kus jäetakse meelde ka kui mitu korda antud märksõna kogu veebisaidi peal kordub. Kui veebikraapija on ühte veebisaiti külastanud vähem või võrdselt veebikraapija teise parameetri väärtusega, siis lõpetab veebikraapija selle töötlemise ning hakkab töötlema järgmist veebisaiti.

2.5.2 Otsingumootori kasutamine

Otsingumootoriks kasutatakse Google Custom Search API-t. Otsingumootori kasutamiseks peab rakenduse konfiguratsioonifailis leiduma Google Custom Search API-ga seotud parameetrid:

1. String - Google Custom Search Engine ID
2. String - Google Custom Search API päringu URL
3. String - Google Custom Search API võti
4. Täisarv - Tulemuste lehtede arv, mida Google Custom Search API tagastab

Google Custom Search Engine ID saamiseks on kasutajal vaja luua Google konto ning luua uus kohandatud otsingumootor Google Custom Search Engine lehel[8]. Otsingumootori loomisel on vaja anda väärtused parameetritel *Sites to search* ja *Name of the search engine*. *Sites to search* väärtuseks võib anda mõne Google poolt pakutud näidisaadressi, sest hiljem muudame selle täpsemate seadete all ümber ning *Name of the search engine* väärtuseks määrata soovitud otsingumootori nime. Kui kohandatud otsingumootor on loodud, tuleks see avada redigeerimiseks ning muuta *Search the entire web* väärtuseks *ON*. Sellega lubame otsingumootoril tulemusi leida üle kogu veebi, selle asemel, et otsing oleks piiratud ühe kindla URL-i peale. Google Custom Search API-ga seotud 1. parameetri väärtuseks sobibki loodud otsingumootori *Search Engine ID*.

Google Custom Search API päringu URL-i väärtuseks on teenuse päringu URL ilma parameetriteta. Antud URL-i leiab Google Custom Search API testimislehelt[9].

Google Custom Search API võtme saab luua Google Custom Search JSON API lehel[10] vajutades nupule GET A KEY, kus tuleb võtme genereerimiseks luua uus projekt, millega loodud võti seostatakse. Google Custom Search API-ga seotud 3. parameetri väärtuseks sobibki loodud võti.

Kuna otsingumootori API tagastab korraga ainult 10 lehekülge, oleks vaja suuremamahulise otsingu korral saata mitu päringut. Google Custom Search API-ga

[8] https://cse.google.com/cse/all
[9] https://developers.google.com/custom-search/v1/cse/list
[10] https://developers.google.com/custom-search/v1/overview#api_key

seotud 4. parameetri väärtuses on võimalik täpsustada, kui mitu päringut tehakse ühe otsingu kohta. Vaikimisi on selle parameetri väärtuseks üks.

2.5.3 Otsingumootori töökäik

Otsingumootori töö algab korrektse teenuse päringu URL-i koostamisega. Päring peab sisaldama kolme kohustuslikku parameetrit:

1. `cx` – Kohandatud otsingumootori ID, mida päringus kasutatakse (Google Custom Search Engine ID väärtus)
2. `q` – Päringus kasutatavad otsingu tingimused
3. `key` - Google Custom Search API võti

Esimene ja kolmas parameeter on eelmise alapeatüki läbimisega olemas, aga otsingu tingimus on veel puudu. Otsingu tingimuse operaatorite kohta leiab täpsemat informatsiooni artiklist „The Ultimate Guide to the Google Search Parameters“ (Watson-Wailes, 2008).

Otsingu tingimuse koostamiseks tuleb kõik otsinguvormiga edastatud märksõnad ühendada `OR` operaatoriga, mille tulemusena Google Search Engine API tagastab meile kõik võimalikud veebisaidid, mis on seotud ühe või mitme märksõnaga. Kui otsingus kasutatud profiili loomisel oli kasutatud veebisaite, siis tuleks need tulemustest eemaldada, sest tulemustes on vaja kuvada sarnaseid veebisaite, mitte samu. Seda on võimalik teha operaatoriga `-site:...`, kus kolm punkti on asendatud kasutatud veebisaidi aadressiga. Kui veebiprofiili loomisel oli kasutatud rohkem kui üks veebisait, tuleb antud tingimust korrata iga veebisaidi kohta.

Kuna otsinguvormis tuleb täpsustada, mis sihtriigi kohta tulemusi otsitakse, tuleb päringusse lisada üks mittekohustuslik parameeter. Parameeter `cr` abil saab API-le selle informatsiooni edastada, mis koosneb prefiksist `country` ning prefiksile on juurde lisatud ISO Alpha-2[11] riigikood. Kui sihtriigiks on Eesti, siis oleks parameetri `cr` väärtuseks `countryEE`.

[11] ISO Alpha-2 – Kahetäheline kood, mis on seotud ühe riigiga (Nations Online , kuupäev puudub)

Koostatud päringu URL kodeeritakse ning rakendus pärib andmeid Google Custom Search API-st. Vastuses leiduvad veebisaitide URL-id lisatakse loendisse ning need edastatakse veebikraapijale märksõnade kogumiseks. Juhul kui päring peaks ebaõnnestuma, kuvatakse kasutajale veateade Google Custom Search Engine'i otsingul esines tehniline tõrge. Kui Google Custom Search Engine API tasuta versiooni päringute liimit on ületatud, kuvatakse selle kohta vastav veateade.

2.5.4 Märksõnade võrdlemine

Märksõnade ehk string-tüüpi väärtuste võrdlemiseks on kasutatud Apache Commons Text andmeteeki, mis sisaldab erinevaid algoritme nende töötlemiseks. Commons Text dokumentatsioonis on kirjeldatud andmeteegis leiduvaid tekstide võrdlemisalgoritme järgmiselt (The Apache Software Foundation, kuupäev puudub):

- Cosine Distance – Arvutab koosiinus kauguse kahe erineva stringi vahel
- Fuzzy Score – Sobitamisalgoritm, mis on sarnane otsimisalgoritmidele, mida kasutavad erinevad tekstiredaktorid nagu Sublime Text, TextMate ja Atom
- Hamming Distance – Hammingi kaugus kahe erineva sama pikkusega teksti vahel on arv, mis koosneb indeksitest, kus tähemärgid erinesid üksteisest
- Jaccard Distance – Arvutab protsentuaalselt Jaccardi kauguse kahe erineva stringi vahel
- Jaccard Similarity – Arvutab protsentuaalselt Jaccardi sarnasuse kahe erineva stringi vahel
- Jaro-Winkler Distance – Sarnasusalgoritm, mis protsentuaalselt arvutab kui mitu tähemärki asuvad samal indeksil võrreldes kahte erinevat stringi
- Levenshtein Distance – Kahe erineva stringi erinevuse arvutamise algoritm

Kuna märksõnade erinevust oleks vaja saada protsentuaalselt, siis kaks sobivat algoritmi nendest oleks Jaccard Similarity ja Jaro-Winkler Distance. Mõlemad algoritmid töötavad põhimõttel, et mida sarnasemad tekstid on, seda suurem protsent väärtuseks on. Jaccard similarity asemel võiks ka Jaccard Distance kasutada, aga siis tulemuseks oleks vastupidine väärtus, ehk väiksema protsendi korral on tekstid sarnasemad.

Et otsustada kumba algoritmi rakenduses mõistlikum kasutada oleks, katsetas autor mõlemaid algoritme kasutades erinevaid märksõnu. Esimeses katses võrreldavaks märksõnaks oli jäätis (Tabel 3). Mõlemad algoritmid andsid suurima tulemuse täpselt samale sõnale. Erinevus seisneb aga teiste märksõnadega võrdlemises, üldiselt annab Jaro-Winkleri algoritm palju suurema tulemuse ülejäänud märksõnade kohta, kui Jaccardi algoritm.

Tabel 3 Märksõna jäätis võrdlemise tulemused

Märksõna	**Jaccard Similarity**	**Jaro-Winkler Distance**
sorbetid	0,300000	0,527778
marjad	0,110000	0,000000
pelmeenid	0,090000	0,425926
köögiviljad	0,170000	0,419192
jäätis	1,000000	1,000000

Teises katses võrreldavaks märksõnaks on jäätised, mis on sarnane märksõnale jäätis, kuid erineb tähemärkide pikkuse poolest (Tabel 4). Isegi kui mõlemad algoritmid andsid suurima tulemuse märksõnale jäätis, on Jaro-Winkleri algoritm antud olukorras parem, kuna ainult suurim tulemus jäetakse ühe märksõna kohta meelde.

Tabel 4 Märksõna jäätised võrdlemise tulemused

Märksõna	**Jaccard Similarity**	**Jaro-Winkler Distance**
sorbetid	0,500000	0,541667
marjad	0,200000	0,527778
pelmeenid	0,270000	0,458333
köögiviljad	0,230000	0,477273
jäätis	0,710000	0,950000

Kolmandas katses võrreldavaks märksõnaks on toit, mida märksõnade tabelis pole (Tabel 5). Kuna antud sõna puudub võrreldavate märksõnade tabelis, siis pole võimalik saada tulemust, mis saja protsendiliselt klapiks. Mõlemad algoritmid andsid sarnasemaks sõnaks sorbetid, mis tegelikult iseloomu poolest on toit küll, kuid kui võrrelda kahe erineva kategooria sõnu, siis tulemus ei saa väga usaldusväärne olla. Isegi kui võrrelda sõna tema sünonüümiga, võib mõni teine märksõna suurema tulemuse anda. Parema lahenduse saamiseks oleks võimalik märksõnade võrdlemiseks kasutada masinõpet, kuid siis oleks vaja masina treenimiseks väga suurt andmekogu, kus on olemas kõik võimalikud sõnad koos nende sünonüümidega ja sõnad peaksid veel lisaks olema erinevates keeltes. Kuna autoril selline andmekogu puudub, siis hetkeseisuga parim lahendus märksõnade võrdlemiseks on Jaro-Winkleri algoritm.

Tabel 5 Märksõna toit võrdlemise tulemused

Märksõna	**Jaccard Similarity**	**Jaro-Winkler Distance**
sorbetid	0,380000	0,583333
marjad	0,000000	0,000000
pelmeenid	0,110000	0,000000
köögiviljad	0,090000	0,446970
jäätis	0,330000	0,444444

3 Rakenduse testimine

Rakenduse testimiseks on autor koostanud testilood erinevate lehekülgede osadest ning nende funktsionaalsuse toimimisest. Testide eesmärgiks on kinnitada rakenduses korrektselt töötavad funktsioone kuid ka leida võimalikult palju defekte, mille tõttu rakendus ei tööta nagu planeeritud. Testilugude jaoks on koostatud tabelid, mis koosnevad tegevustest ja oodatavatest tulemustest. Iga testiloo lõpus kirjeldatakse, mis testimise käigus õnnestus, milliseid takistusi esines ning mille tõttu probleemid esinesid.

Enne testimist peab kindlaks tegema, et rakendus oleks korrektselt konfigureeritud, mille kohta leiab informatsiooni peatükist 2.1. Lisaks rakenduse konfigureerimisele on vaja kasutajaliideses kindlaks teha, et sätete vormil on veebikraapijale sobivad parameetrite väärtused määratud. Järgnevate testilugude ajal olid veebikraapija parameetrite väärtusteks:

- Ühe veebisaidi otsitavate lehtede arv - 25
- Märksõna minimaalne pikkus - 3
- Eiratud HTML elemendid - button,iframe,script
- Eiratavad märksõnad - kui,ning,mis

3.1 Uue profiili koostamine veebisaitide põhjal

Käesolevas alapeatükis testitakse uue profiili loomise funktsionaalsust (Tabel 6).

Tabel 6 Uue profiili koostamise testilugu

Tegevus	**Oodatav tulemus**
Kasutaja avab lehekülje „Profiilide haldus“.	Avaneb lehekülg „Profiilide haldus“, kus kuvatakse eelnevalt loodud profiilide nimekirja.
Kasutaja vajutab nuppu „Loo profiil“.	Avaneb uue profiili loomise modaalaken.
Kasutaja märgib valiku „Kasuta veebiprofiile“, kirjutab sisendvälja väärtuseks https://www.premia.ee/ ning vajutab nupule „Loo profiil“.	Töö ajal kuvatakse modaalaknas kerimise animatsiooni. Töö lõppedes kuvatakse kasutajale kogutud märksõnade välja.
Kasutaja vajutab ühe märksõna märgendi peal nuppu „X“.	Loendist eemaldati märksõna märgend.
Kasutaja lisab loendi ühe märksõna juurde.	Loendis kuvatakse uut märksõna märgendit.
Kasutaja kirjutab nime välja „Premia test“ ning vajutab nupule „Salvesta profiil“.	Suunatakse tagasi leheküljele „Profiili haldus“. Nimekirja on lisatud salvestatud profiil nimega „Premia test“.

Rakendus suutis lühikese aja jooksul sisestatud veebisaidist koguda 25 enim korduvat märksõna. Märksõnu on võimalik juurde lisada ja eemaldada, kuid olemasoleva märksõna juurde lisamisel peidetakse see loendist. Tegemist on JavaScript’i lisamooduli defektiga, mida on rakenduses kasutatud märksõnade märgendite kuvamiseks. Isegi kui antud sõna on loendist peidetud, jääb see tegelikult sisendvälja nähtamatul kujul ning salvestamisel jäävad andmed alles.

3.2 Olemasoleva profiili muutmine

Käesolevas alapeatükis testitakse olemasoleva profiili muutmise funktsionaalsust (Tabel 7). Testilugu eeldab, et rakenduse andmebaasi on salvestatud profiil nimega „Premia test“.

Tabel 7 Olemasoleva profiili muutmise testilugu

Tegevus	**Oodatav tulemus**
Kasutaja avab lehekülje „Profiilide haldus“.	Avaneb lehekülg „Profiilide haldus“, kus kuvatakse eelnevalt loodud profiilide nimekirja.
Kasutaja vajutab profiili muutmise nuppu (paberi ja pliiatsi ikoon).	Avaneb profiili muutmise modaalaken.
Kasutaja vajutab ühe märksõna märgendi peal nuppu „X“.	Loendist eemaldati märksõna märgend.
Kasutaja lisab loendi ühe märksõna juurde.	Loendis kuvatakse uut märksõna märgendit.
Kasutaja annab profiilile uueks nimeks „Premia“ ning vajutab nupule „Salvesta“.	Suunatakse tagasi leheküljele „Profiili haldus“. Nimekirjas asendati profiil nimi „Premia test“ nimega „Premia“.

Olemasoleva märksõna lisamisel tekib sama defekt, mis tekkis ka uue profiili koostamises. Salvestamisel takistusi ei esinenud.

3.3 Sarnaste ettevõtete otsing

Käesolevas alapeatükis testitakse ettevõtete otsingu funktsionaalsust (Tabel 8). Testilugu eeldab, et rakenduse andmebaasi on salvestatud profiil nimega „Premia“ ja konfiguratsioonifailis on parameetri `google.cse.api.requests` väärtuseks 1.

Tabel 8 Sarnaste ettevõtete otsingu testilugu

Tegevus	**Oodatav tulemus**
Kasutaja avab lehekülje „Ettevõtete otsing“.	Avaneb lehekülg „Ettevõtete otsing“, kus kuvatakse olemasolevate profiilide rippmenüüd.
Kasutaja valib rippmenüüst profiili nimega „Premia“ ning vajutab nupule „Ava otsing“.	Avaneb otsingu detailide modaalaken.
Kasutaja vajutab ühe märksõna märgendi peal nuppu „X“.	Loendist eemaldati märksõna märgend.
Kasutaja lisab loendi ühe märksõna juurde.	Loendis kuvatakse uut märksõna märgendit.
Kasutaja valib riigiks „Estonia“, jätab sisendvälja „Kontaktid“ tühjaks ning vajutab nupule „Otsi“.	Töö ajal kuvatakse modaalaknas kerimise animatsiooni. Töö lõppedes kuvatakse kasutajale märksõnade alusel sarnaseid ettevõtteid.
Kasutaja kirjutab tulemuste nimeks „Premia tulemused“ ning vajutab nupule „Salvesta“.	Kasutaja suunatakse otsingu tulemuste vormile, mille nimekirja lisati salvestatud otsingu tulemus.

Olemasoleva märksõna lisamisel tekib sama defekt, mis tekkis ka uue profiili koostamises. Sarnaste ettevõtete otsing võtab mitu minutit aega võtta, sõltuvalt interneti kiirusest. Otsingus kasutatud märksõnade hulgale võivad otsingu tulemused

segased olla. Täpsemaks otsinguks on soovitav kasutada vähem kui 10 märksõna. Salvestamisel takistusi ei esinenud.

3.4 Tulemuste vaatamine ja kustutamine

Käesolevas alapeatükis testitakse tulemuste vaatamise ja kustutamise funktsionaalsust (Tabel 9). Testilugu eeldab, et andmebaasi on salvestatud otsingu tulemused, mille nimeks on „Premia tulemused“.

Tabel 9 Tulemuste vaatamise ja kustutamise testilugu

Tegevus	**Oodatav tulemus**
Kasutaja avab lehekülje „Tulemused“.	Avaneb lehekülg „Tulemused“, kus kuvatakse salvestatud otsingute tulemusi.
Kasutaja vajutab otsingu tulemuste „Premia tulemused“ detailvaate nuppu (info ikoon).	Avaneb otsingu tulemuste „Premia tulemused“ detailvaate modaalaken.
Kasutaja kontrollib, kas veebisaitide nimekirjas olevad lingid avanevad.	Lingid avanevad ilma probleemideta.
Kasutaja vajutab akna üleval paremas servas „X“ nupule.	Detailvaate modaalaken suletakse.
Kasutaja vajutab otsingu tulemuste „Premia tulemused“ kustutamise nuppu (prügikasti ikoon).	Avaneb otsingu tulemuste kustutamise modaalaken.
Kasutaja vajutab nupule „Kustuta“.	Suunatakse tagasi leheküljele „Tulemused“. Nimekirjast on eemaldatud otsingu tulemused nimega „Premia tulemused“.

Tulemuste vaatamise ja kustutamise testilugu läbiti ilma takistusteta.

3.5 Profiili kustutamine

Käesolevas alapeatükis testitakse profiili kustutamise funktsionaalsust (Tabel 10). Testilugu eeldab, et rakenduse andmebaasi on salvestatud profiil nimega „Premia“.

Tabel 10 Profiili kustutamise testilugu

Tegevus	**Oodatav tulemus**
Kasutaja avab lehekülje „Profiilide haldus“.	Avaneb lehekülg „Profiilide haldus“, kus kuvatakse eelnevalt loodud profiilide nimekirja.
Kasutaja vajutab profiili kustutamise nuppu (prügikasti ikoon).	Avaneb profiili kustutamise modaalaken.
Kasutaja vajutab nupule „Kustuta“.	Suunatakse tagasi leheküljele „Profiili haldus“. Nimekirjast on eemaldatud profiil nimega „Premia“.

Profiili kustutamise testilugu läbiti ilma takistusteta.

3.6 Hinnang

Rakendus suudab täita enamjaolt kõik ülesanded, mis vajalikud on. Hetkel ei suuda rakendus kuvada veebisaitidelt leiduvaid kontaktandmeid, kuna see funktsionaalsus on arenduses. Kuna märksõnade võrdlemisega ei saa väga täpset tulemust, ei kuvata otsingu tulemustes ainult 70% kattuvusega veebisaite, vaid kuvatakse kõiki tulemusi, millest eespool on suurema kattuvusega veebisaidid.

Testimise käigus tuvastati probleem osade veebisaitidega, mille aadressi ei ole võimalik osade keelte parameetreid lisada. Näiteks üheks selliseks veebisaidiks on https://www.balbiino.ee, mille lõppu saab lisada parameeteri `/en` (https://www.balbiino.ee/en), et kuvada veebisait inglise keeles, kuid eesti keelseks tõlgeteks puudub parameeter `/et` (https://www.balbiino.ee/et). Sellise veebisaidi omapära tõttu võib märksõnade kogumise protsessi ajal veebikraapija avada inglise keelseid lehti ja koguda nendelt märksõnu, sest veebikraapija läbib kõiki lehti, mis

sisaldavad kasutaja poolt sisestatud URL-i. Tulemusena kuvatakse segakeelseid märksõnu ning kasutaja ei saa loodud profiili mõistlikuks otsinguks kasutada.

Kuna lõpptulemuses kuvatakse märksõnu, mida veebisaidil on tihti kasutatud, võivad need märksõnad olla väga üldised, mis mitte midagi veebisaidi kohta ei ütle. Kasutajal tuleks võimalikult palju täiendada veebikraapija sätetes eiratavate märksõnade loendit, et neid lõpptulemuses kasutajale ei kuvataks.

Sarnaste ettevõtete otsingus on soovitatav kasutada võimalikult vähe märksõnu ning need võiksid seostuda mingi kindla valdkonnaga. Kasutades tähtsusetuid märksõnu võib otsingumootor tagastada erinevate valdkondadega seotud ettevõtete veebilehti, mis pole kuidagi omavahel seoses.

Kokkuvõte

Käesoleva töö eesmärgiks oli koostada rakendus, mis aitaks kasutajal leida teatud ettevõtte veebiprofiili põhjal sarnaseid ettevõtteid. Töö sisaldab lühikest ülevaadet rakendustest, mille eesmärgiks on veebilehtedelt andmete kogumine ja nende töötlemine. Ülevaate põhjal koostati rakendusele veebikraapija, mis suudab erinevatelt veebisaitidelt koguda korduvaid märksõnu ning nende põhjal luua profiil.

Töö lõpuks valmis rakenduse prototüüp, mis lisaks andmete kogumisele suudab otsingumootorist sarnaseid veebisaite leida ning neid märksõnade põhjal võrrelda. Märksõnade võrdlemiseks uuriti erinevaid tekstivõrdlusalgoritme nagu Jaccard Similarity, Jaro-Winkler Distance, Levenshtein Distance, jne. Lõpuks otsustati kasutusele võtta Jaro-Winkler Distance algoritm, mis andis katsetuste käigus parima tulemuse. Sellegipoolest ei ole see ideaalne algoritm, sest võrreldakse kahe string andmetüübi väärtuse sarnasust, aga sünonüümide ja sarnaste tähendustega sõnade võrdlemisel ei ole tulemus optimaalne.

Rakendus sisaldab minimaalset kasutajaliidest, et kasutajad kes ei oma teadmisi programmeerimises, suudaksid rakendusega toime tulla ning kasutada seda abivahendina vajalike eesmärkide täitmisel.

Uurimistöö lõppemisega ei ole rakenduse arendus veel läbi. Kuna testimise käigus selgus mitmeid erinevaid olukordi, mille tõttu rakenduse töö ei pruugi kasutajale olla kasulik, oleks vaja veebikraapija ja otsingumootori tööd parendada. Autor soovib siiski rakenduse arendada sellise seisuni, et see hõlbustaks kasutajate tööd usaldusväärsete tulemustega. Rakenduse edaspidised muudatused on kirjeldatud projekti koodirepositooriumis[12].

[12] https://github.com/madis121/CompanyWebProfileComparator

Allikate loetelu

Balodis, M. (9. August 2018. a.). *Extensions*. Allikas: Chrome Web Store: https://chrome.google.com/webstore/detail/web-scraper/jnhgnonknehpejjnehehllkliplmbmhn

Christian Bauer, G. K. (2005). *Java Persistence with Hibernate*. Greenwich: Manning.

Davis, M. (17. Juuni 2014. a.). *Five Reasons You Should Use Tomcat*. Allikas: Future Hosting: https://www.futurehosting.com/blog/five-reasons-you-should-use-tomcat/

Elias, L. (2011). *Hibernate raamistiku õppematerjal*. Tallinn: Tallinna Ülikool.

Gil, P. (24. Oktoober 2018. a.). *What Is a CAPTCHA Test? How Do CAPTCHAs Work?* Allikas: Lifewire: https://www.lifewire.com/what-is-a-captcha-test-2483166

Hüüs, S. (22. Mai 2009. a.). *Mis on PDF ja kuidas sellega ümber käia?* Allikas: Digitark: https://digitark.ee/mis-on-pdf-ja-kuidas-sellega-umber-kaia/

IBM. (2017). *Advantages of Java*. Allikas: IBM: https://www.ibm.com/support/knowledgecenter/en/ssw_aix_71/com.ibm.aix.performance/advantages_java.htm

Jsoup. (kuupäev puudub). *Java HTML Parser, with best of DOM, CSS and jquery*. Allikas: Jsoup: https://jsoup.org/

McKinsey Global Institute. (Jaanuar 2017. a.). *Harnessing automation for a future that works*. Allikas: McKinsey: https://www.mckinsey.com/featured-insights/digital-disruption/harnessing-automation-for-a-future-that-works

MetricSpot. (kuupäev puudub). *Website Analytics and SEO Tools*. Allikas: Metric Spot: https://metricspot.com/

Nations Online . (kuupäev puudub). *ISO Alpha-2, Alpha-3, and Numeric Country Codes*. Allikas: Nations Online: https://www.nationsonline.org/oneworld/country_code_list.htm

Oracle Corporation. (2010). *Oracle and Sun Microsystems*. Allikas: Oracle: https://www.oracle.com/sun/index.html

PostgreSQL. (kuupäev puudub). *The world's most advanced open source database*. Allikas: PostgreSQL: https://www.postgresql.org/

Scraper API. (kuupäev puudub). *Helping developers easily build scalable web scrapers*. Allikas: Scraper API: https://www.scraperapi.com/

Star, J. (kuupäev puudub). *(Please) Stop Using Unsafe Characters in URLs*. Allikas: Perishable Press: https://perishablepress.com/stop-using-unsafe-characters-in-urls/

The Apache Software Foundation. (kuupäev puudub). *Package org.apache.commons.text.similarity*. Allikas: Apache Commons Text Documentation: https://commons.apache.org/proper/commons-text/apidocs/org/apache/commons/text/similarity/package-summary.html

Tutorialspoint. (2017). *Java - Overview*. Allikas: Tutorialspoint: https://www.tutorialspoint.com/java/java_overview.htm

Tutorialspoint. (kuupäev puudub). *Hibernate Overview*. Allikas: Tutorialspoint: https://www.tutorialspoint.com/hibernate/hibernate_overview.htm

Uibo, M. (2018). *Java 8 ja Spring 5 koostöö.* Tallinn: Tallinna Ülikool.

Watson-Wailes, P. (10. Juuli 2008. a.). *The Ultimate Guide to the Google Search Parameters*. Allikas: Moz: https://moz.com/blog/the-ultimate-guide-to-the-google-search-parameters

Summary

Software Development for Automatic Matching of Company Web Profiles

The purpose of this thesis was to develop an application that would help the user find similar company web profiles to the one that user had specified. This project includes a short overview of similar applications which purpose is to collect data from different websites and process it. With this knowledge, a web scraper was developed for the application which could collect repetitive keywords from different websites and create a profile based on those keywords.

As the result, a prototype was created that could also find similar websites and compare them by keywords, in addition to collecting data. Different text comparison algorithms were researched such as Jaccard Similarity, Jaro-Winkler Distance, Levenshtein Distance and so on. In the end, Jaro-Winkler Distance algorithm was used as it proved to give the best results. Even then, the algorithm is not ideal as it compares two different string-type values, but it lacks the knowledge of synonyms and terms that could have a similar meaning. Due to that, the results might not be optimal.

Application includes a minimal user interface, which will help potential users utilise this application for their needs and do not need previous knowledge in programming to do so.

The application development is not yet complete. During the testing phase, many different circumstances were found that proved the application not to be useful. Because of that, the functionality of web scraper and search engine still needs improving. The author wishes to improve the application to the stage where it assists users and provides them with reliable results. Future changes related to the application are described at the author's code repository[13].

[13] https://github.com/madis121/CompanyWebProfileComparator