TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Kristo Isberg 179171IADB

KASUTAJALIIDESE TESTIDE VAHENDI VALIK JA JUURUTAMINE TARKVARAARENDUSETTEVÕTTE NÄITEL

Bakalaureusetöö

Juhendaja: Maili Markvardt
MSc

Tallinn 2020

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: Kristo Isberg

23.02.2020

Annotatsioon

Käesoleva bakalaureusetöö eesmärgid olid sobivaima automaatse kasutajaliidese testimise vahendi leidmine ettevõtte jaoks, kus automaatseid kasutajaliidese teste varasemalt ei kasutatud, ning kasutajaliidese testide näidislahenduse loomine lihtsustamaks ettevõttes töötavate tarkvaraarendajate õpikõverat.

Töö käigus katsetati seitset automaatset kasutajaliidese testimise vahendit: CodeceptJS, Cypress, Nightwatch.js, Selenide, Serenity BDD, TestCafe ja WebdriverIO. Vahendite Saaty meetodi abil võrdlemise tulemusena selgus, et ettevõtte vajadustele vastas vahenditest enim TestCafe.

Töö teise tulemusena loodi kasutajaliidese testide näidislahendus ettevõtte siseveebi näitel ning seeläbi parandati ka rakenduse töökindlust.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 31 leheküljel, 7 peatükki, 7 joonist, 8 tabelit.

Abstract

Selecting a Tool for and Implementing End-to-End Tests for a Software Development Company

The objectives of this bachelor's thesis were to select the best suiting automated end-to-end testing tool for a company that had previously not used automated end-to-end tests and to provide the software developers of the company with a sample implementation of automated end-to-end tests to decrease their learning curve.

In order to select the best suiting tool for the company, seven tools were selected into the comparison: CodeceptJS, Cypress, Nightwatch.js, Selenide, Serenity BDD, TestCafe and WebdriverIO. The tools were compared by their reliability, performance, ease of installation and use, browser support, support for concurrent execution, remote browsers and BrowserStack, CrossBrowserStack and SauceLabs services. As a result of the comparison, TestCafe was selected as the best suiting tool for the company using Saaty's method.

As the second result of the thesis, a sample implementation of automated end-to-end tests was created by covering the company's intranet application with end-to-end tests using TestCafe. This also improved the reliability of the application by helping discover several bugs in it.

The thesis is in Estonian and contains 31 pages of text, 7 chapters, 7 figures, 8 tables.

Lühendite ja mõistete sõnastik

API	rakendusliides, *application programming interface* Liides, mis võimaldab reeglistikule vastaval rakendusprogrammil teise programmi teenuseid kasutada.
Chromium	Avatud lähtekoodiga veebilehitseja projekt, millel põhinevad mitmed veebilehitsejad, näiteks Google Chrome.
DOM	dokumendi objektimudel, *document object model* Programmeerimisliides HTML ja XML dokumentidele, mis kujutab dokumente sõlmede ja objektidena.
eesrakendus	*front-end application* Rakendus või rakenduse osa, mida kasutaja näeb.
JVM	Java Virtual Machine Virtuaalmasin, mis käitab Java baitkoodiks kompileeritud rakendusi.
käsureakest	*command line shell* Rakendus käsureakäskluste operatsioonisüsteemile täitmiseks edastamiseks.
npm	Node Package Manager Paketihaldur Node.js jaoks.
panuk	*commit* Faili(desse) korraga tehtud muudatused.
püsiseade	*test fixture* Püsivad andmed, millest testid sõltuvad.
tagarakendus	*back-end application* Rakendus või rakenduse osa, mida kasutaja ei näe ning mis teeb äriloogilisi otsuseid.
töölõim	*worker thread* Meetod Node.js-is JavaScripti koodi paralleelselt mitmes lõimes käitamiseks.
vastupidine puhverserver	*reverse proxy* Server, mis vahendab kliendi poolt tehtavaid päringuid teistele veebiserveritele.

vastuvõtutestid	*acceptance tests* Testid, mis kontrollivad, kas testitav süsteem vastab ärilistele nõuetele.
voolav süntaks	*fluent syntax* Inimkeele sarnaselt loetav süntaks.
väidete teek	*assertion library* Teek, mille abil kontrollitakse, kas testide tulemused vastavad väidetele.

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Käesoleva bakalaureusetöö teemaks on automaatsete kasutajaliidese testide vahendi valik ja juurutamine tarkvaraarendusettevõtte näitel. Antud tarkvaraarendusettevõte tegeleb IT-lahenduste disainimise ja ehitamisega pankadele ja finantstehnoloogiaettevõtetele. Suurema osa ettevõtte poolt arendatavatest lahendustest moodustavad võrgurakendused.

Ettevõttel on vähemalt üks klient, kellel on nõue, et rakenduse ühe osa uuendamise korral tuleb kogu rakendus täielikult kasutajaliidese kaudu läbi testida. Praegu tehakse seda käsitsi, mis olenemata sellest, kas testijad on ettevõtte- või kliendipoolsed, on pikas perspektiivis ajakulukas ja seeläbi ka ressursinõudlik tegevus. Seetõttu soovib klient lähitulevikus kasutusele võtta automaatsed kasutajaliidese testid.

Teadaolevalt ei kasutata ettevõttes praegusel hetkel automaatseid kasutajaliidese teste mitte kusagil ning seetõttu soovib ettevõte saada ülevaadet kaasaegsetest kasutajaliidese testimise vahenditest. Tööl on kaks eesmärki:

- sobivaima automaatse kasutajaliidese testimise vahendi valimine ettevõtte jaoks,
- automaatsete kasutajaliidese testide näidislahenduse loomine lihtsustamaks ettevõttes töötavate tarkvaraarendajate õpikõverat.

Töö koosneb neljast osast. Töö esimeses osas tutvustatakse kasutajaliidese teste, nende vajalikkust, nende eeliseid ja puuduseid ning soovituslikku mahtu. Töö teises osas tutvustatakse töö metoodikat: rakendust ja kasutajavoogusid, mille alusel vahendeid võrreldakse ning sobivaima vahendi leidmiseks kasutatavat Saaty meetodit. Töö kolmandas osas leitakse võrreldavad vahendid, analüüsitakse neid ning analüüsi põhjal leitakse sobivaim vahend ettevõttes kasutamiseks. Töö neljandas osas kirjeldatakse töö tulemusena valminud kasutajaliidese testide näidislahendust ning selle loomise töökäiku. Töö viiendas osas arutletakse töö võimalike edasiarenduste üle.

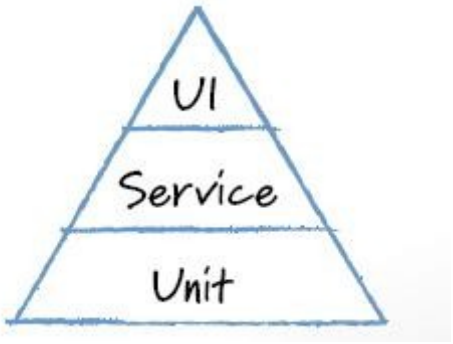
2 Kasutajaliidese testid

Kasutajaliidese testid (ingl. *end-to-end tests, e2e tests*) on testid, mille käigus testitakse rakenduse kasutajavoogusid algusest lõpuni jäljendades päris kasutaja käitumist rakenduses kontrollimaks rakenduse toimimist tervikuna. Kasutajaliidese testid kontrollivad, et rakenduse kõik kihid ning nendevahelised ja välised ühendused toimiksid vigadeta. Samuti on võimalik kasutajaliidese testide abiga kontrollida, et kasutajaliides toimiks kõigil toetatavatel seadmetel ning veebilehitsejatel. [1]

Kasutajaliidese testidel on mitmeid puudusi, millega tuleks testimisstrateegiat koostades arvestada. Kasutajaliidese testid on aeglased, kuna need nõuavad tegevusi nagu veebilehitseja käivitamine, testitava veebilehekülje avamine ja veebileheküljel päris kasutaja tegevuste jäljendamine [2]. Lisaks on kasutajaliidese testid rabedad, kuna negatiivse testi tulemuse võivad lisaks reaalsetele vigadele põhjustada ka näiteks oodatust pikem koodi käitusaeg, ebastabiilne internetiühendus ja väliste sõltuvuste poolsed probleemid [2]. Kasutajaliidese testide põhjal on ka raskem vigu siluda, kuna pole teada konkreetne rakenduse komponent, kus viga tekkis [3].

Et minimeerida kasutajaliidese testide puudustest tulenevaid probleeme, ei tohiks kogu rakenduse testimine toimuda kasutajaliidese testide kaudu. Rakenduse üksikute komponentide toimimist peaksid kontrollima ühiktestid ning üksikute komponentide koostoimimist peaksid kontrollima integratsioonitestid. Kasutajaliidese testid peaksid olema mõeldud ainult kogu rakenduse tervikuna toimimise kontrollimiseks. [3]

Erinevat liiki testide arvude ligikaudset suhet aitab väljendada joonisel 1 nähtav algselt 2004. aastal tarvaraarendaja Mike Cohni poolt kirjeldatud ning tarkvaraarendajate seas laialdast kasutust leidnud testipüramiid: püramiidi tipu ehk mahu poolest väikseima osa moodustavad kasutajaliidese testid, püramiidi põhja ehk mahu poolest suurima osa moodustavad ühiktestid ning nende kahe vahel paiknevad integratsioonitestid [4] [5].



Joonis 1. Mike Cohni testipüramiid

Näiteks USA tehnoloogiaettevõttes Google soovitatakse, et kõigist testidest 70% moodustaksid ühiktestid, 20% integratsioonitestid ja vaid 10% kasutajaliidese testid [3].

3 Metoodika

Töö töökäik koosneb kuuest etapist:

- testitava rakenduse leidmine,
- näidistestide abil testitavate kasutajavoogude leidmine,
- kõigile nõuetele vastavate ja potentsiaalselt sobivate testimisvahendite leidmine,
- testimisvahendite individuaalne analüüs ning näidistestide kirjutamine eelnevalt välja selgitatud kasutajavoogudele,
- testimisvahendite hindamine ning sobivaima testimisvahendi leidmine Saaty meetodi abil,
- testitava rakenduse katmine kasutajaliidese testidega.

3.1 Testitava rakenduse valik ja kirjeldus

Testitavaks rakenduseks valiti ettevõtte siseveeb, mille kasuks otsustati, kuna tegemist on võrdlemisi mahuka rakendusega, mille ülesehitus ja kasutatavad tehnoloogiad sarnanevad suuresti tolle kliendi, kelle rakendusele tulevikus kasutajaliidese teste kirjutama hakatakse (edaspidi kliendi), omadele. Lisaks omati juba varasemalt piisavaid ligipääse ettevõtte siseveebi arendus- ja tootekeskkondadele, mis muutis ettevõttesisese asjaajamise kiireks ja probleemivabaks.

Alternatiivina kaaluti kliendi rakenduste arhitektuuri võimalikult täpselt jäljendava testrakenduse loomist. Analüüsi käigus leiti aga, et ettevõtte siseveebi arhitektuur sarnaneb kliendi rakenduste arhitektuurile piisavalt ja arhitektuuri võimalikult täpne jäljendamine ei olegi kõige olulisem, kuna töö tulemusi peaks saama kasutada lisaks kliendi rakendustele ka mujal. Samuti välditi nõnda teoreetilist olukorda, kus testrakenduse funktsionaalsus valitakse nõnda, et see soodustab mõne kindla testimisvahendi kasutamist. Lisaks annab töö ettevõttele nõnda ka kohest ja otsest kasu, kuna ettevõtte siseveebile kasutajaliidese testide kirjutamine parandab selle kui ettevõttele strateegiliselt olulise rakenduse töökindlust.

3.1.1 Arhitektuur ja kasutatavad tehnoloogiad

Testimise seisukohast olulised tehnoloogiad on ettevõtte siseveebis ning kliendi rakendustes identsed. Eesrakendused (ingl. *front-end application*) kasutavad React vaatemootorit ning nende sõltuvused paigaldatakse npm (Node Package Manager) repositooriumist. Tagarakendused (ingl. *back-end application*) kasutavad Spring Boot raamistikku ning nende sõltuvused paigaldatakse Gradle automatiseerimistööriista abil Maven Central repositooriumist. Rakendused ehitatakse valmis ja paigaldatakse arendus- ja tootekeskkondadesse kasutades Jenkins automatiseerimisserverit.

Ainus suurem erinevus ettevõtte siseveebi ja kliendi rakenduste vahel, mis testimisprotsessil rolli mängida võib, on see, et kui kliendi rakenduste ees- ja tagarakendus paiknevad eraldi projektides ning kahe projekti paigaldamine ja käivitamine toimub eraldi, siis ettevõtte siseveebi ees- ja tagarakendus paiknevad ühes projektis ja nende paigaldamine ja käivitamine toimub korraga.

3.1.2 Testitavad kasutajavood

Testitavate kasutajavoogude leidmiseks selgitati välja ettevõtte siseveebi populaarseimad kasutajavood, analüüsides rakenduse vastupidise puhverserveri (ingl. *reverse proxy*) logisid. Testitavad kasutajavood otsustati leida logide analüüsimise teel, et valitud kasutajavood sarnaneksid päris kasutajate toimingutele rakenduses ning kasutajavoogude valik oleks võimalikult erapooletu. Välja valiti kolm kõige populaarsemat kasutajavoogu, mis on kättesaadavad töö lisast 1.

3.1.3 Kasutajaliidese testide maht testrakenduses

Kuna ettevõtte siseveebis puudusid töö kirjutamise hetkel lisaks kasutajaliidese testidele ka ühik- ja integratsioonitestid, ei olnud võimalik kasutajaliidese testide mahu hindamisel aluseks võtta ühik- ja integratsioonitestide mahtu.

Seetõttu võeti testide loomisel aluseks põhimõte, et kasutajaliidese testid peaksid kontrollima vaid rakenduse toimimist tervikuna ning vaikimisi otsustati iga kasutajavoo testimiseks luua vaid üks test. Kasutajavoogude puhul, mis hõlmasid mõnda komponenti, mille kasutamisel saab tekkida äriloogiline viga, otsustati luua ka test ühe sellise vea kontrollimiseks.

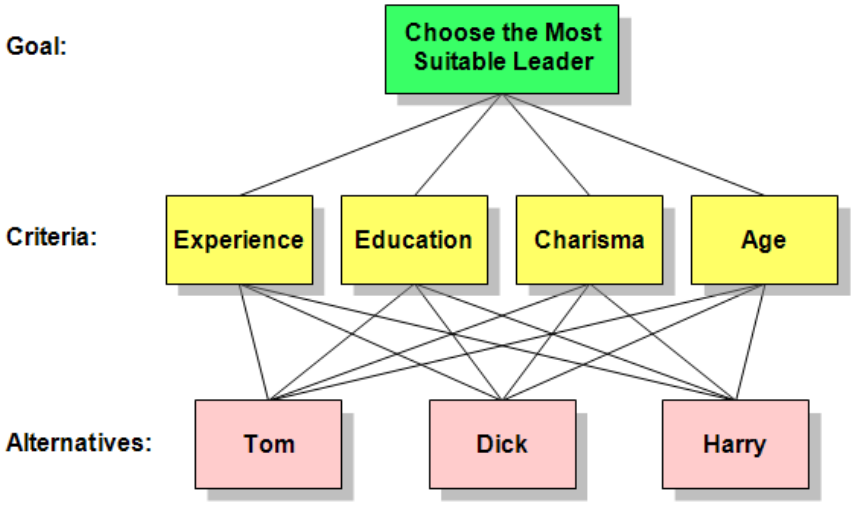
3.1.4 Kasutajavoogude põhjal loodavad testid

Välja selgitatud kasutajavoogude põhjal otsustati luua kokku 10 näidistesti: testid eelnevalt sõnastatud kasutajavoogudele ning lisaks eraldi testid sisselogimisprotsessile, mis on kõigi ülejäänud kasutajavoogude osaks. Näidisteste luues lähtuti üldjuhul punktis 3.1.3 sõnastatud põhimõttest, ainsaks erandiks oli rakenduse sisselogimise protsess, millele loodi neli äriloogilise vea testi, kuna tegemist on rakenduse jaoks kriitilise tähtsusega komponendiga ning sellest sõltuvad ka kõik ülejäänud testid. Näidistestide kirjeldused on kättesaadaval töö lisas 2.

3.2 Saaty meetod

Saaty meetod ehk analüütiliste hierarhiate meetod (ingl. *Analytic Hierarchy Process*, AHP) on USA matemaatiku Thomas L. Saaty poolt 1970. aastate esimeses pooles välja töötatud meetod paari kaupa võrdlemise kaudu suhteskaalade leidmiseks [6]. Kõige sagedamini kasutatakse seda mitme kriteeriumi ja mitme alternatiiviga otsuste tegemiseks, sealhulgas ka subjektiivsete hinnangute põhjal objektiivsete tulemuste leidmiseks [6]. Saaty meetodit on kasutatud näiteks sobivaima tuumajaama liigi väljaselgitamiseks võttes arvesse nii majanduslikke kui ka näiteks sotsiaalseid tegureid ning Microsoftis tarkvaraliste süsteemide üldise kvaliteedi hindamiseks [7] [8].

Saaty meetod põhineb hierarhilisel mudelil, mille tipuks on mingisugune eesmärk [6]. Eesmärgi all paiknevad kriteeriumid, mis võivad jaguneda alamkriteeriumiteks, mille all paiknevad omakorda alternatiivid [6]. Näidis nelja kriteeriumi ja kolme alternatiiviga Saaty meetodi hierarhiast on nähtaval joonisel 2 [9].



Joonis 2. Saaty meetodi hierarhia ettevõtte juhi valimise näitel

Kuna Saaty meetodi kasutamine on mahukas ja keerukas protsess, on selle protsessi lihtsustamiseks loodud mitmeid tööriistu.

Kõige tuntum taoline tööriist on Aalto ülikooli süsteemianalüüside laboratooriumis loodud Web-HIPRE [10]. Paraku on tegemist Java rakendusega, mida pole uuendatud alates 2003. aastast ning mida saab kasutada vaid veebilehitsejaga Microsoft Internet Explorer, kuna kõigist tänapäevastest veebilehitsejatest on rakenduse kasutamiseks vajalik Java tugi eemaldatud [11] [12].

Alternatiivsete tööriistade uurimise järel otsustati Saaty meetodi kasutamiseks kasutada tasuta veebipõhist teenust 123ahp, mis sisaldab kogu vajalikku funktsionaalsust ning annab suhteliselt põhjaliku ülevaate Saaty meetodi protsessi vahesammudest [13]. Ainsad puudujäägid 123ahp teenuse juures on alamkriteeriumite toe puudumine ning ebamugav kasutajaliides.

4 Kasutajaliidese testimise vahendid

Võrdlusesse vahendite leidmiseks kasutati erinevaid veebiallikaid:

- GitHubi repositoorium "Awesome Selenium", kuhu kogutakse vabavarakogukonnas populaarset *awesome* manifesti järgides kokku kõik kvaliteetsed Seleniumi ja WebDriveriga seotud projektid ja ressursid [14] [15];
- GitHubi teemad "end-to-end-testing", "e2e", "e2e-testing", "e2e-tests", "webdriver", "selenium" [16] [17] [18] [19] [20] [21];
- Google-i otsingud märksõnadega "end-to-end testing JavaScript" ja "end-to-end testing Java" [22] [23].

4.1 Vahendite võrdlusesse valimise kriteeriumid

Vahendite võrdlusesse valimiseks sõnastati järgnevad kriteeriumid:

- ei tohi nõuda eraldiseisvat paigaldust, kasutavad paigaldamiseks juba kasutusel olevaid meetodeid (npm või Gradle) ja paigaldamine saab toimuda rakenduse paigaldamise ajal;
- testide loomine peab toimuma arendusprotsessi osana ning teste peab saama kirjutada juba kasutusel olevas programmeerimiskeeles (JavaScript või Java);
- peab toimuma aktiivne arendus: arendust ei ole ametlikult lõpetatud ning viimasest arendusaktiivsusest (väljalase npm-is või Mavenis, panuk (ingl. *commit*) GitHubis) pole möödas üle poole aasta;
- ei tohi olla lihtsalt veebilehitseja automatiseerimise vahend, peab sisaldama kasutajaliidese testide jaoks vajalikku funktsionaalsust;
- ei tohi olla raamistikuspetsiifilised, peavad töötama kõigi veebirakendustega, mis kasutavad paigaldamiseks ja käivitamiseks eelnevalt nimetatud tehnoloogiaid.

4.2 Võrdlusest välistatud vahendid

Võrdlusesse vahendeid otsides leiti hulgaliselt vahendeid, millega oleks õnnestunud vähemalt osaliselt soovitud lõpptulemuseni jõuda, ent mille vastavus punktis 4.1 sõnastatud kriteeriumitega oli liiga väike nende võrdlusesse kaasamiseks. Järgnevas nimekirjas on välja toodud mõningaid näiteid taolistest vahenditest koos võrdlusest välistamise põhjustega.

- **Conductor:** ebaaktiivne arendus (viimane arendustegevus aastal 2017) [24];
- **DalekJS:** arendus lõpetati ametlikult aastal 2017, reaalsuses lõppes arendustegevus aastal 2015, autorid soovitavad kasutajaliidese testimiseks kasutada TestCafe-d [25];
- **Protractor:** mõeldud spetsiifiliselt Angulari ja AngularJS-i rakenduste testimise jaoks [26];
- **Galen:** mõeldud spetsiifiliselt kohanduva disaini testimiseks, ei sisalda kogu vajalikku funktsionaalsust [27];
- **Zombie.js, CasperJS:** kasutavad päris veebilehitsejate asemel eraldiseisvaid graafilise kasutajaliideseta veebilehitsejaid [28] [29];
- **Puppeteer, Playwright:** veebilehitsejate automatiseerimise teegid, ei ole mõeldud testimise jaoks ja ei sisalda ise vajalikku funktsionaalsust [30] [31];
- **Karma:** mõeldud eelkõige veebilehekülgede ühiktestimiseks, autorid soovitavad kasutajaliidese testide jaoks kasutada Protractorit [32];
- **SlimerJS, PhantomJS, Nightmare:** eraldiseisvad graafilise kasutajaliideseta veebilehitsejad, ei ole mõeldud testimise jaoks [33] [34] [35].

4.3 Võrreldavad kriteeriumid

Vahendite võrdlemiseks ja Saaty meetodis kasutamiseks sõnastati ettevõttesiseselt kogutud informatsiooni põhjal viis kriteeriumit:

- **Seadistamise lihtsus:** Minimaalse töötava lahenduse seadistamine peaks toimuma suurema vaevata. Pole kõige olulisem, kuna tegemist on suuresti ühekordse tegevusega.

- **Õpikõvera suurus:** Testide kirjutamine peaks olema sobilik ka eesrakenduste arendajatele, kelle peamine töövahend on JavaScript. Testide kirjutamiseks kasutatav süntaks peaks olema võimalikult ühtlane. Testide korrektne õnnestumine peaks olema saavutatav võimalikult lihtsalt. Ei tohiks eksisteerida vigu, mis nõuavad raskendatud lahendusi (ingl. *workarounde)*.
- **Veebilehitsejate tugi:** Ettevõtte üheks eesmärgiks kasutajaliidese testide kasutamisel on kontrollimine, kas rakendus töötab samaväärselt kõigil veebilehitsejatel. Vahend peaks toetama kõiki laialt levinud veebilehitsejaid. Teste peaks saama käitada ka üle võrgu. Eksisteerida võiks ka BrowserStack, CrossBrowserTesting ja SauceLabs teenuste toed.
- **Jõudlus:** Ei ole kõige olulisem, kuid peaks olema arvestataval tasemel. Peaks eksisteerima ka testide paralleelselt mitmes lõimes käitamise võimalus.
- **Stabiilsus:** Testide käitus peaks alati lõppema sama tulemusega, ei tohiks esineda ebastabiilsusi, mis ei tulene ebakvaliteetsetest testidest.

Veebilehitsejate, mida testitavad vahendid toetama peaks, välja selgitamisel tugineti StatCounteri 2020. aasta veebruari andmetele Eestis kasutatavate veebilehitsejate turuosa kohta. Nõudmiseks seati, et vahendid peaksid toetama kõiki töölaua veebilehitsejaid, mille turuosa on vähemalt üks protsent. Nõudmisele vastasid järgnevad veebilehitsejad:

- Google Chrome – 66,97%,
- Apple Safari – 15,38%,
- Mozilla Firefox – 7,34%,
- Internet Explorer – 2,42%,
- Microsoft Edge (vana versioon) – 2,00%,
- Opera – 1,80% [36]

Toetatavaid veebilehitsejaid kasutas kokku 95,91 protsenti Eesti veebikasutajatest. Ülejäänud 4,09 protsendist 2,83 oli Samsung Interneti ehk Samsungi Androidi telefonide vaikimisi veebilehitseja turuosa ning järelejäänud turuosa kuulus peamiselt väiksema kasutajaskonnaga Chromiumil põhinevatele veebilehitsejatele (Vivaldi, Microsoft Edge uus versioon, Yandex Browser jt). [36]

4.4 Võrreldavad vahendid

Punktis 4.1 sõnastatud kriteeriumitele vastas ja võrdlusesse valiti seitse kasutajaliidese testimise vahendit:

- CodeceptJS (versioon 2.5.0),
- Cypress (versioon 4.2.0),
- Nightwatch.js (versioon 1.3.4),
- Selenide (versioon 5.10.0),
- Serenity BDD (versioon 2.1.13),
- TestCafe (versioon 1.8.2),
- WebdriverIO (versioon 6.0.8).

4.5 Võrreldavate vahendite poolt kasutatavad liidesed

Seitsmest võrdlusesse valitud vahendist neli (Nightwatch.js, Selenide, Serenity BDD ja WebdriverIO) kasutasid veebilehitsejaga ühendumise liidesena WebDriverit, kaks (Cypress ja TestCafe) kasutasid oma liidest ning üks (CodeceptJS) toetas mitut erinevat liidest. Cypressi ja TestCafe liidesed on mõlemad veebilehitseja-siseselt töötavad liidesed, mistõttu kategoriseeriti liidesed järgmiselt: WebDriver ning veebilehitseja-sisesed liidesed [37] [38].

4.5.1 WebDriver

WebDriver on W3C (World Wide Web Consortium) veebilehitsejate testimise ja tööriistade töörühma poolt välja töötatud standard platvormist ja programmeerimiskeelest sõltumatuks veebilehitsejate jälgimiseks ja kontrollimiseks [39]. Kuna WebDriveri puhul on tegemist ametliku ja laialtlevinud standardiga, millele pandi kaudne alus juba 2004. aastal, on selle põhjal loodud draiverid kõigile tänapäevastele veebilehitsejatele [40].

WebDriveri poolt sooritatavad tegevused ei ole veebilehitseja mõistes simuleeritud ning need on võimalikult lähedased päris kasutaja poolt sooritatavatele tegevustele. See tähendab seda, et WebDriver saab alati lehel mingisugust tegevust sooritada vaid siis kui see on võimalik ka päris kasutajal. [37]

WebDriveri suurimaks puuduseks nõudmistest lähtuvalt on erinevad draiverite paigaldusviisid. Kui näiteks ChromeDriveri ja GeckoDriveri, mida on vaja WebDriveri kasutamiseks vastavalt Google Chrome-i ja Mozilla Firefoxiga, paigaldamine toimub standardsel viisil, siis vana Microsoft Edge-i kasutamiseks vajaliku Microsoft WebDriveri paigaldamine käib läbi Windowsi Deployment Image Servicing and Management käsurearakenduse ning Apple Safari kasutamiseks vajalik SafariDriver on viimaste macOS versioonide puhul vaikimisi juba paigaldatud, ent tuleb käsurea kaudu aktiveerida. [41]

Lisaks peavad WebDriverit kasutavad vahendid draiveritega suhtlema üle rakendustevahelise liidese, mis tähendab, et väliste teguritena võivad testide tulemust mõjutada rakendustevahelise liidese latentsus ja potentsiaalne haprus. [37]

Kõigi WebDriverit kasutavate vahendite puhul avaldus testimisel ühine viga, kus funktsioon, mis peaks tekstilahtri kõigepealt tühjendama ning seejärel sinna uue teksti sisestama, jättis tühjendamise sammu vahele ning lisas uue teksti eelneva järele. Internetist uurides selgus, et tegemist on konfliktiga ChromeDriveri ja React vaatemootori vahel ning ühtegi laialdast heakskiitu leidnud lahendust sellele ei eksisteerinud [42]. Ajutise lahendusena saadeti kõigi vahendite puhul WebDriverile käsklus enne uue teksti sisestamist varasema teksti pikkusega võrdse arvu kordi kustutusklahvi vajutamiseks.

4.5.2 Veebilehitseja-sisesed liidesed

Veebilehitseja-sisene liides on mõiste kategoriseerimaks Cypress ja TestCafe veebilehitsejaga ühendumise liideseid. Veebilehitseja-sisesed liidesed on liidesed, mille puhul veebilehitsejale käskluseid jagav rakendus asub veebilehitseja poolt avatud veebilehe sees, mitte eraldi rakenduses. [37] [38]

Veebilehitseja-siseste liidestega vahenditel puuduvad välised sõltuvused erinevalt WebDriverit kasutavatest vahenditest, mis vajavad iga veebilehitseja kasutamiseks erinevat draiverit. Lisaks ei sõltu taoliste vahendite abil kirjutatud testide tulemused veebilehitseja draiveriga suhtlemiseks vajaliku rakendustevahelise liidese latentsusest ja potentsiaalsest haprusest. [37]

Lisaks on taolise lähenemise korral vähemalt kõigi võrreldavate vahendite puhul testid üksteisest isoleeritud, mis tähendab, et testid ei sõltu üksteise tulemustest, kuna veebilehitseja küpsised, mälu jms hävitatakse testide vahel automaatselt. [38] [43]

Taoliste vahendite poolt sooritatavad tegevused on aga veebilehitseja mõistes simuleeritud ning seetõttu võib tekkida erandlikke olukordi, kus vahend saab sooritada tegevusi, mida päris kasutaja sooritada ei saaks, näiteks vajutada nuppu, millele hiirekursoriga ligipääs puudub. [37]

Lisaks on veebilehitseja-sisese lähenemise korral testide käigus sooritatavad tegevused piiratud veebilehesiseste tegevustega, näiteks ei suuda testid kasutada mitut veebilehitseja sakki. [37] [38]

4.6 Võrreldavate vahendite analüüs

Vahendite veebilehitsejate, BrowserStack, CrossBrowserTesting ja SauceLabs teenuste ning veebilehitsejate üle võrgu juhtimise tugede võrdlemisel tugineti vahendite veebilehekülgedelt pärinevale informatsioonile. Ülejäänud analüüsi jaoks loodi kõigi vahendite abil näidistestid ettevõtte siseveebile. Kuna vahendid kasutasid vaikimisi erinevaid seadeid, võeti vastu järgnevad otsused, et kõigi vahendite abil loodud testid sarnaneksid semantika ja struktuuri poolest võimalikult ligilähedaselt üksteisele ning neid oleks võimalik paremini võrrelda:

- Ühegi vahendi abil testide loomisel ei kasutatud leheküljeobjekti mudelit.
- Kõigi vahendite abil loodavad testid isoleeriti üksteisest, vajadusel tühjendati iga testi eel veebilehitseja küpsised ja mälu ning iga testi alguses tuli sisselogimisprotsess uuesti läbida.
- Kõigi vahendite abil loodavaid teste käitati jõudluse testimisel paralleelselt vaid ühes lõimes. Selle võimaluse olemasolu vahendite puhul aga analüüsiti ja selle kasutamist katsetati ning see väljendub ka võrdluse lõpptulemustes.

Kõigi vahendite näidistestide kirjutamiseks ja käitamiseks kasutati sama füüsilist masinat Windows 10 operatsioonisüsteemiga, stabiilset kaabliga internetiühendust ning Google Chrome (versioon 81.0.4044.92) ja Mozilla Firefox (versioon 75.0)

veebilehitsejaid. Testide kirjutamisel võeti eesmärgiks, et need töötaks samaväärselt mõlema veebilehitsejaga ning nii kasutajaliidesega kui ilma. Antud veebilehitsejate kasuks otsustati, kuna tegemist oli ainsate veebilehitsejatega (välja arvatud ülejäänud Chromiumil põhinevad veebilehitsejad, mille kasutuskogemus oleks olnud peaaegu identne Google Chrome-i omaga), mida toetasid kõik võrdlusesse valitud vahendid ning tegemist oli kahe kõige populaarsema veebilehitsejaga Eestis kui välistada Apple Safari veebilehitseja, mida saab kasutada vaid macOS operatsioonisüsteemiga arvutitel [36] [44].

Testide jõudluse ja stabiilsuse hindamiseks käitati iga vahendi abil kirjutatud näidisteste 100 korda ning peeti arvet õnnestunud ja ebaõnnestunud käituskordade arvu kohta ning õnnestunud käituskordade puhul ka käitusaja kohta. Arvu 100 kasuks otsustati, kuna kasutajaliidese testid on suhteliselt aeglased (keskmiseks näidistestide käitusajaks arvestati 60 sekundit), aga tegemist on piisavalt suure valimiga, et selle käigus peaksid avalduma kõik ebastabiilsused. Testide jõudluse mõõtmiseks kasutati PowerShell käsureakesta (ingl. *command line shell*) käsklust Measure-Command, millega mõõdeti aega testide käivitamise käskluse sisestamisest vahendi protsessi elutsükli lõpuni. Ebaõnnestunud testide korral analüüsiti neid. Kui ebaõnnestunud testid tulenesid testide poolsetest vigadest, parandati need ja käitati testid uuesti.

Kõigi vahendite jõudlustestide käitamiseks kasutati graafilise kasutajaliideseta Google Chrome veebilehitsejat, mille kasuks otsustati, kuna tegemist on Eestis ülekaalukalt kõige populaarsema veebilehitsejaga [36]. Ideaalis oleks soovitud jõudlusteste käitada ka Mozilla Firefox veebilehitsejaga, kuid testide käitamise protsess osutus planeeritust ajanõudlikumaks (testide keskmiseks käitusajaks osutus planeeritud 60 sekundi asemel 95,6 sekundit ehk üle 59% rohkem, lisaks tuli osade vahendite puhul vigaste testide poolt põhjustatud stabiilsusprobleemide parandamise järel teste uuesti käitada).

4.6.1 CodeceptJS

CodeceptJS toetab mitut päris veebilehitsejatega ühendumise liidest: WebDriver (läbi WebdriverIO), TestCafe, Protractor, Playwright ning Puppeteer, lisaks ka eraldiseisvat Nightmare graafilise kasutajaliideseta veebilehitsejat ja mobiilirakenduste testimiseks

mõeldud Appiumi ja Detoxit [37]. Tulevikus plaanitakse tugi lisada ka Cypress liidesele [45].

Toetatud liidestest toetavad kõiki soovitud veebilehitsejaid vaid WebDriver, TestCafe ja Protractor. Kuna Protractor on mõeldud eelkõige AngularJS rakenduste testimiseks, tehti valik WebDriveri ja TestCafe vahel [26]. CodeceptJS dokumentatsioonis soovitatakse mitme veebilehiteja toe saavutamiseks kasutada esialgu liidesena TestCafed ning vahetada see välja WebDriveri vastu kui TestCafe isepärasused muutuvad piiravaks või põhjustavad vigu, mistõttu kasutati näidistestide kirjutamisel liidesena TestCafed [37].

CodeceptJS paigaldamine ja käivitamine toimub npm-i kaudu, mis tähendab, et testide kirjutamiseks saab kasutatada JavaScripti ning JavaScriptiks transpileeritavaid programmeerimiskeeli [37]. Vaikimisi ei toetanud CodeceptJS modernset JavaScripti süntaksit, mistõttu tuli selle tugi ise lisada. CodeceptJS-i süntaks on väga puhas ja loogiline. Lisaks soodustab CodeceptJS koodi ja andmete taaskasutust: CodeceptJS-i I objekti, mis väljendab testiva kasutaja minaperspektiivi, saab laiendada oma meetodite ning väärtustega [37].

Jõudlustestide käitamisel tekkisid üksikud juhtumid, kus test ei leidnud üles DOM (dokumendi objektimudel, ingl. *document object model*) elementi, mis tekkis mõne pikemaajalise protsessi (näiteks andmete laadimise või vormi esitamise) tulemusena. Uurimisel selgus, et erinevalt suuremast osast ülejäänud vahenditest ei oota CodeceptJS enne elemendi kasutamist selle tekkimise järgi ning kui elementi selle kasutamise ajal ei eksisteeri, test ebaõnnestub [37]. Segadust tekitas aga asjaolu, et erinevalt näiteks WebdriverIO-st, mille testid ebaõnnestusid kõigi taoliste olukordade puhul ning viga avaldus juba teste kirjutades, oli CodeceptJS puhul ebaõnnestumine harv nähtus, viga avaldus alles jõudlustestide käitamisel ning jäi mulje, et vahend ootab elementide tekkimise järgi, ent väga lühiajaliselt. Vahendi dokumentatsioonist taolise käitumise kohta TestCafe liidese korral aga informatsiooni ei leitud. Peale antud probleemi lahendamist läbis CodeceptJS jõudlustestid probleemideta, testide käitamine võttis aega keskmiselt 190,3 sekundit, käitusaegade standardhälve oli 5,4 sekundit.

Teste saab üle võrgu käitada nii WebDriver kui TestCafe liidest kasutades, kuna CodeceptJS seadistusfaili kaudu on võimalik WebdriverIO ja TestCafe liideste

seadistusi muuta ja mõlemal teegil on olemas nii testide üle võrgu käivitamise võimalus kui ka BrowserStack, CrossBrowserTesting ja SauceLabs teenuste tugi [37] [43] [46] [47].

CodeceptJS toetab testide paralleelset käitamist kahel viisil: kasutades NodeJS töölõimi (ingl. *worker thread*) ning alamprotsesse. Töölõimede kasutamine on lihtsam ning nõuab testide käivitamise käskluses vaid ühe kuni kahe parameetri muutmist, kuid toetab testide paralleelset käitamist vaid lokaalselt ning samas veebilehitsejas, lisaks ei kuva see testide logisid. Alamprotsesside kasutamine nõuab ulatuslikumat seadistamist, ent toetab ka testide käitamist üle võrgu ning mitmes erinevas veebilehitsejas. [37]

Testimise käigus üritati kasutada töölõimi, ent see ei õnnestunud, kuna iga veebilehitseja protsess viis läbi vaid ühe testi ning lõpetas seejärel töö, mispeale lõpetas töö ka põhiprotsess. Sooviti katsetada ka alamprotsesside kasutamist, ent teadaolevalt on alamprotsesside kasutamine TestCafe liidesega samuti vigane [48].

4.6.2 Cypress

Cypress kasutab veebilehitsejatega ühendumiseks oma veebilehitseja-siseselt töötavat liidest, mis toetab veebilehitsejatest täielikult Google Chrome-i ja uut Microsoft Edge-i, beta-tugi on olemas Mozilla Firefoxil ning teoreetiliselt toetab see ka kõiki teisi Chromiumil põhinevad veebilehitsejaid, näiteks Operat, Brave-i ja Vivaldit, kuid teiste soovitud veebilehitsejate tugi puudub Cypressil täielikult [38]. Internet Exploreri ja Apple Safari veebilehitsejate toe lisamine on arutluse all, kuid pole hetkeseisuga arenduses [49] [50].

Cypressi paigaldamine ja käivitamine toimub npm-i kaudu, mis tähendab, et testide kirjutamiseks saab kasutatada JavaScripti ning JavaScriptiks transpileeritavaid programmeerimiskeeli. Testide struktureerimiseks kasutatakse Mocha testimisraamistikku ja Chai väidete teeki (ingl. *assertion library*). Cypress soodustab koodi ja andmete taaskasutust: Cypressi objekti saab laiendada oma meetoditega ning andmeid saab hoida JSON formaadis püsiseadetena (ingl. *test fixture*). [38]

Cypressi testide kirjutamise süntaks on kompaktne ja loogiline, ainus küsimusi tekitav asjaolu oli see, et Cypressi should ja and funktsioonides asuvad Chai väited sõnena,

mitte objekti omadustena nagu Chai teeki otse kasutades. Cypressi seadistamisel oli vaid üks küsimusi tekitav asjaolu: kui kõigi muude seadistuste seadistamine, sealhulgas testiprotsessist videote salvestamise välja lülitamine, toimus läbi seadistusfaili, siis vea korral ekraanist kuvatõmmise salvestamise välja lülitamine toimus programmaatiliselt. Tegemist on asjaoluga, mis on häirinud ka mitmeid teisi kasutajaid, kuid mida veel lahendatud pole [51].

Testide käitamisel Google Chrome veebilehitsejaga ei esinenud ühtegi probleemi, ent testide käitamine Mozilla Firefox veebilehitsejaga ebaõnnestus täielikult, kuna Cypress ei suutnud kahe testifaili vahel veebilehitseja süsteemiprotsessi sulgeda, mistõttu teatas veebilehitseja uue protsessi käivitamise asemel, et eelnevalt tuleb varasem protsess sulgeda. Tegemist on levinud veaga, millest on ka Cypressi arendajaid teavitatud, kuid mis testimise ajal veel parandatud ei olnud [52]. Jõudlustestid läbis Cypress probleemideta, testide käitamine võttis aega keskmiselt 76,3 sekundit, käitusaegade standardhälve oli 3,6 sekundit.

Cypress toetab testide paralleliseerimist vaid üle mitme masina ning nõuab selleks kas ametlikku (Cypress Dashboard Service) või mitteametlikku (näiteks sorry-cypress) töölauateenust. Töölauateenuseta Cypress ei toeta ka testide käitamist üle võrgu ning puuduvad ka CrossBrowserTesting, BrowserStack ja SauceLabs teenuste toed, ent vähemalt kahe viimase lisamine on tulevikus plaanis. [38]

4.6.3 Nightwatch.js

Nightwatch.js kasutab veebilehitsejatega ühendumise liidesena WebDriverit [53]. Nightwatch.js paigaldamine ja käivitamine toimub npm-i kaudu, mis tähendab, et testide kirjutamiseks saab kasutatada JavaScripti ning JavaScriptiks transpileeritavaid programmeerimiskeeli [53]. Vaikimisi ei toetanud Nightwatch.js modernset JavaScripti süntaksit, mistõttu tuli selle tugi ise lisada. Nightwatch.js-i süntaks on suhteliselt kompaktne ning loogiline, peamiselt on kasutusel voolav süntaks (ingl. *fluent syntax*). Väidete kirjutamiseks toetatakse mitut erinevat stiili, näidistestide kirjutamisel kasutati Chai stiilis väiteid, mis paraku voolavat süntaksit ei toetanud.

Aeg-ajalt tekkis Nightwatch.js-il probleeme tekstiväljadega, mis automaatselt fokusseeriti – vahend ei suutnud selliseid välju mõnikord täita. Selle olukorra

parandamiseks pandi vahend enne välja täitmist käsitsi 0,25 sekundiks ootama. Testide käitamisega ei tekkinud probleeme ei Google Chrome ega Mozilla Firefox veebilehitsejaga. Jõudlustestid läbis Nightwatch.js probleemideta, testide käitamine võttis aega keskmiselt 61,6 sekundit, käitusaegade standardhälve oli 2,3 sekundit.

Nightwatch.js toetab testide paralleliseerimist nii samal masinal kui üle võrgu ning selle võimaluse aktiveerimiseks tuli lisada vahendi seadistusfaili üks rida. Vaikimisi käitab see paralleelselt maksimaalselt nii palju veebilehitseja protsesse kui on masina protsessoril lõimi. Lisaks on olemas ka tugi veebilehitsejate käitamiseks üle võrgu ning BrowserStack, CrossBrowserTesting ja SauceLabs teenuste jaoks. [53]

4.6.4 Selenide

Selenide kasutab veebilehitsejatega ühendumise liidesena WebDriverit [54]. Selenide-i paigaldamine toimub Maven repositooriumi kaudu ning selle paigaldamiseks ja käivitamiseks saab kasutada erinevaid automatiseerimistööriistu (Maven, Gradle, Ant jt) [54]. Testide kirjutamiseks saab kasutada kõiki JVM-i (Java Virtual Machine) keeli (Java, Kotlin, Scala jt) ning Selenide toetab kõiki levinumaid JVM testimisraamistikke (JUnit, TestNG, Cucumber jt) [54] [55]. Kuna projektis olid varasemalt kasutusel Java, Gradle ning JUnit, eelistati Selenide-i testide kirjutamiseks neid.

Selenide toetab testide käitamist üle võrgu ning ka BrowserStack, CrossBrowserTesting ja SauceLabs teenuseid [54] [56]. Selenide-ile endale testide paralleliseerimise võimalust sisse ehitatud pole, kuid sellega saab kasutada testimisraamistike ja kolmandate osapoolte poolt pakutavaid testide paralleliseerimise võimalusi. Töö käigus üritati kasutada JUniti eksperimentaalset testide paralleliseerimise võimekust, mis põhimõtte poolest küll toimis, ent muutis testid oluliselt hapramaks – Selenide ei suutnud mõnes kohas tegelikult eksisteerivaid DOM elemente üles leida.

Peale punktis 4.5.1 nimetatud probleemi seoses tekstilahtrite tühjendamisega ei esinenud ühtegi probleemi testide käitamisel ei Google Chrome ega Mozilla Firefox veebilehitsejaga. Selenide-i abil kirjutatud testid nõuavad Java koodile iseloomulikult küll võrreldes JavaScriptil põhinevate vahenditega palju korduvat koodi, kuid Selenide-i enda süntaks on kompaktne ning ühtlane. Jõudlustestid läbis Selenide probleemideta,

testide käitamine võttis aega keskmiselt 64,6 sekundit, käitusaegade standardhälve oli 5,7 sekundit.

4.6.5 Serenity BDD

Serenity BDD, varasema nimega BDD Thucydides, kasutab veebilehitsejatega ühendumise liidesena WebDriverit [57]. Serenity BDD paigaldamine toimub Maven repositooriumi kaudu ning selle paigaldamiseks ja käivitamiseks saab kasutada erinevaid automatiseerimistööriistu (Maven, Gradle, Ant jt) [57]. Testide kirjutamiseks saab kasutada kõiki JVM-i keeli (Java, Kotlin, Scala jt) ning see toetab JUnit, Cucumber ja JBehave testimisraamistikke [57]. Kuna projektis olid varasemalt kasutusel Java, Gradle ning JUnit, eelistati Serenity BDD testide kirjutamiseks neid.

Serenity BDD on võrdluse kõige laiaotstarbelisem vahend, kuna see on mõeldud vastuvõtutestide (ingl. *acceptance tests*) kirjutamiseks üldiselt, mitte pole mõeldud spetsiifiliselt kasutajaliidese testide jaoks [58]. Kasutajaliidese testimise funktsionaalsus pole Serenity BDD-le sisse ehitatud, sellel on olemas vaid Selenium teegi integratsioon [58]. Testide kirjutamisel tuli kasutada Selenium teeki otse, mis tegi Serenity BDD-st võrdluse kõige madalama abstraktsiooni tasemega vahendi.

Serenity BDD testide käitamine ebaõnnestus, kuna vahend ei suutnud teadmata põhjusel WebDriveri instantsi JUniti testidesse sisestada. Kuna Serenity BDD on nõnda laiaotstarbeline vahend ja kasutajaliidese testid JUniti ja Seleniumi abil on suhteliselt spetsiifiline kasutusjuhtum, ei leitud internetist probleemile sobilikku lahendust, mistõttu otsustati Serenity BDD edasisest võrdlusest välistada.

4.6.6 TestCafe

TestCafe kasutab veebilehitsejatega ühendumiseks oma liidest: TestCafe-d kasutades avab veebilehitseja testitava veebilehekülje läbi puhverserveri, mis sisestab testitavasse veebilehekülge kasutaja tegevusi emuleeriva koodi. TestCafe töötab kõigil veebilehitsejatel, soovitud veebilehitsejatest toetatakse ametlikult kõiki peale Opera, mille vastu TestCafe-d aktiivselt ei testita. [43]

TestCafe paigaldamine ja käivitamine toimub npm-i kaudu, mis tähendab, et testide kirjutamiseks saab kasutatada JavaScripti ning JavaScriptiks transpileeritavaid programmeerimiskeeli [43]. Kasutusjuhendis kasutatakse testide kirjutamisel pisut ebatavalist süntaksit ühe sõneargumendiga funktsioonide väljakutsumiseks, mis tekitab segadust koodi vormindamise vahendites nagu Prettier, ent selle saab kergesti asendada levinuma süntaksiga.

TestCafe süntaks on kompaktne, kasutatakse voolavat süntaksit. Teste kirjutades tekkis tunne, et testide kirjutamise API-l (rakendusliidesel, ingl. *application programming interface*) võiks olla lisaks üks abstraktsioonikiht, kuna hetkel erineb väidete süntaks DOM elementide kasutamise süntaksist märgatavalt. Testide käitamisega ei tekkinud probleeme ei Google Chrome ega Mozilla Firefox veebilehitsejaga. Jõudlustestid läbis TestCafe probleemideta, testide käitamine võttis aega keskmiselt 107,9 sekundit, käitusaegade standardhälve oli 3 sekundit.

TestCafe toetab testide käitamist üle võrgu ning olemas on ka ametlikud liidesed BrowserStack ja SauceLabs teenuste ja CrossBrowserTestingu poolt pakutav liides nende teenuse jaoks. [43] [46]

Testide paralleliseerimist toetab TestCafe nii samal masinal kui üle võrgu kõigil veebilehitsejatel peale vana Microsoft Edge, kuna seal puudub teadaolev võimalus uue veebilehitseja akna avamiseks ja seal navigeerimiseks. Testimise käigus käitati TestCafe teste probleemideta samal masinal neljas paralleelses veebilehitseja protsessis, mille jaoks tuli testide käivitamise käsklusele juurde lisada üks parameeter paralleelsete protsesside arvuga. [43]

4.6.7 WebdriverIO

WebdriverIO kasutab veebilehitsejatega ühendumise liidesena WebDriverit. WebdriverIO paigaldamine ja käivitamine toimub npm-i kaudu, mis tähendab, et testide kirjutamiseks saab kasutatada JavaScripti ning JavaScriptiks transpileeritavaid programmeerimiskeeli. Testimisraamistikest toetab WebdriverIO Mochat, Cucumberi ja Jasmine-i. Näidistestide kirjutamisel kasutati paigaldustööriista poolt vaikimisi paigaldatud Mocha testimisraamistikku koos Chai väidete teegiga. [47]

Vaikimisi Mocha seadistus ei toetanud modernset JavaScripti süntaksit, mistõttu tuli see ise seadistada. Näidistestide kirjutamisel ilmnes, et kirjutatud testid olid oluliselt rabedamad kui ülejäänud vahendite abil kirjutatud testid. See tulenes sellest, et erinevalt suuremast osast ülejäänud vahenditest ei oodanud WebdriverIO enne DOM elemendi kasutamist selle ekraanile tekkimise järgi enne selle kasutamist ja luges testi ebaõnnestunuks kui seda ekraanil veel polnud. Selle probleemi parandamiseks tuli WebdriverIO-le iga sellise olukorra ees anda käsk ootamaks, et element ekraanile tekiks enne kui vahend seda kasutada üritab.

Peale vahendi käsitsi ootama panemise vajaduse ning punktis 4.5.1 nimetatud probleemi seoses tekstilahtrite tühjendamisega ei esinenud ühtegi probleemi testide käitamisel ei Google Chrome ega Mozilla Firefox veebilehitsejaga. WebdriverIO testide kirjutamise süntaks koos Chai väidetega on kompaktne ja suhteliselt ühtlane. Jõudlustestid läbis WebdriverIO probleemideta, testide käitamine võttis aega keskmiselt 73,1 sekundit, käitusaegade standardhälve oli 3,1 sekundit.

WebdriverIO toetab testide käitamist üle võrgu: liidesed on olemas nii Selenium serveri kui ka BrowserStack, CrossBrowserTesting ja SauceLabs teenuste jaoks. WebdriverIO toetab testide paralleliseerimist nii samal masinal kui üle võrgu, antud võimalus oli testimisel vaikimisi aktiveeritud ja töötas probleemideta. [47]

4.7 Sobivaima vahendi valimine Saaty meetodi abil

Saaty meetodi kasutamiseks vahendite hindamisel kasutati tasuta veebipõhist teenust 123ahp [13]. Iga järgneva tabeli juurde on lisatud võrdluse järjepidevuse suhe (ingl. *consistency ratio*, edaspidi CR), mida kasutatakse võrdluse tulemuse kontrollimiseks [6]. Et võrdlus oleks kasutamiseks sobilik, peab võrdluse CR olema väiksem kui 10% [6].

Esmalt hinnati teenuse abil omavahel punktis 4.3 sõnastatud kriteeriumite tähtsuse suhteid, kasutades Saaty fundamentaalskaalat (lisa 3) ning tulemustest koostati risttabel (tabel 1). Kuna stabiilsus osutus kõigi vahendite puhul võrdseks (kõik vahendid läbisid jõudlustesti vahendipoolsete probleemideta), otsustati antud kriteerium edasisest analüüsist välistada. Tähtsaimaks kriteeriumiks peeti veebilehitsejate tuge, väga

oluliseks peeti ka õpikõvera suurust, ülejäänud kriteeriumid ei olnud nõnda suure tähtsusega.

Tabel 1. Kriteeriumite tähtsuste suhted

	Seadistamise lihtsus	**Õpikõvera suurus**	**Veebilehitsejate tugi**	**Jõudlus**	**Kaal**
Seadistamise lihtsus	1	1/6	1/8	1/3	0,0453
Õpikõvera suurus	6	1	1/4	5	0,2596
Veebilehitsejate tugi	8	4	1	7	0,6107
Jõudlus	3	1/5	1/7	1	0,0844

Kriteeriumite tähtsuste suhete võrdluse CR oli 9,77%, mis on väiksem kui 10%, mis tähendab, et võrdlus on kasutuskõlblik.

Seejärel hinnati teenuse abil omavahel vahendeid iga kriteeriumi kohta, kasutades Saaty fundamentaalskaalat (lisa 3) ning tulemustest koostati risttabelid (tabelid 2-5). Vahendite tähistamiseks kasutati tabelis järgnevaid tähiseid:

- **A** – CodeceptJS,
- **B** – Cypress,
- **C** – Nightwatch.js,
- **D** – Selenide,
- **E** – TestCafe,
- **F** – WebdriverIO.

Seadistamise lihtsuse hindamisel võeti arvesse minimaalse töötava lahenduse saavutamiseks vaja läinud seadistuste mahtu, seadistamisel tekkinud probleeme ning lisaülesandeid ning asjaolu, et WebDriveril põhinevad vahendid võivad vajada eraldiseisvat lisapaigaldust.

Tabel 2. Vahendite hinnangute suhted seadistamise lihtsuse kohta

	A	**B**	**C**	**D**	**E**	**F**	**Kaal**
A	1	1/3	3	5	1/3	7	0,1638
B	3	1	5	5	1/3	7	0,2617
C	1/3	1/5	1	1	1/7	3	0,0577
D	1/5	1/5	1	1	1/7	3	0,0542
E	3	3	7	7	1	9	0,4349
F	1/7	1/7	1/3	1/3	1/9	1	0,0276

Vahendite hinnangute suhete seadistamise lihtsuse kohta võrdluse CR oli 4,91%, mis on väiksem kui 10%, mis tähendab, et võrdlus on kasutuskõlblik.

Õpikõvera suuruse hindamisel võeti arvesse vahendi süntaksi ühtlust ning testide korrektselt toimima saamisel tekkinud probleeme. Lisaks võeti arvesse asjaolu, et testide kirjutamise keelena Javat kasutavate vahendite kasutamine võib olla eesrakenduste arendajate jaoks raskendatud.

Tabel 3. Vahendite hinnangute suhted õpikõvera suuruse kohta

	A	**B**	**C**	**D**	**E**	**F**	**Kaal**
A	1	3	3	3	3	3	0,2243
B	3	1	5	5	3	7	0,4100
C	1/3	1/5	1	3	1/3	3	0,0963
D	1/3	1/5	1/3	1	1/3	3	0,0661
E	1/3	1/3	3	3	1	5	0,1634
F	1/3	1/7	1/3	1/3	1/5	1	0,0400

Vahendite hinnangute suhete õpikõvera suuruse kohta võrdluse CR oli 8,01%, mis on väiksem kui 10%, mis tähendab, et võrdlus on kasutuskõlblik.

Veebilehitsejate toe osas hinnati võrdselt kõiki vahendeid peale Cypressi, kuna Cypress toetas praktikas vaid Google Chrome veebilehitsejat (ning eeldatavasti ka ülejäänud

Chromiumil põhinevaid veebilehitsejaid, sh. Operat), samas kui kõik ülejäänud vahendid toetasid kõiki soovitud veebilehitsejaid ja teenuseid.

Tabel 4. Vahendite hinnangute suhted veebilehitsejate toe kohta

	A	**B**	**C**	**D**	**E**	**F**	**Kaal**
A	1	9	1	1	1	1	0,1957
B	1/9	1	1/9	1/9	1/9	1/9	0,0217
C	1	9	1	1	1	1	0,1957
D	1	9	1	1	1	1	0,1957
E	1	9	1	1	1	1	0,1957
F	1	9	1	1	1	1	0,1957

Vahendite hinnangute suhete veebilehitsejate toe kohta võrdluse CR oli 0%, mis on väiksem kui 10%, mis tähendab, et võrdlus on kasutuskõlblik.

Jõudluse hindamisel võeti arvesse näidistestide ühes lõimes käitamise jõudlust ning testide paralleliseerimise võimaluse olemasolu ja toimivust.

Tabel 5. Vahendite hinnangute suhted jõudluse kohta

	A	**B**	**C**	**D**	**E**	**F**	**Kaal**
A	1	1/5	1/9	1/7	1/7	1/9	0,0215
B	5	1	1/5	1/3	1/3	1/5	0,0571
C	9	5	1	5	5	3	0,4251
D	7	3	1/5	1	1	1/5	0,1016
E	7	3	1/5	1	1	1/5	0,1016
F	9	5	1/3	5	5	1	0,2931

Vahendite hinnangute suhete jõudluse kohta võrdluse CR oli 9,35%, mis on väiksem kui 10%, mis tähendab, et võrdlus on kasutuskõlblik.

Seejärel leiti kriteeriumite ja vahendite hinnangute põhjal vahendite koguhinnangud (tabel 6).

Tabel 6. Saaty meetodi lõpptulemus

	Seadistamise lihtsus	**Õpikõvera suurus**	**Veebilehitsejate tugi**	**Jõudlus**	**Koguhinnang**
CodeceptJS	0,0074	0,0582	0,1195	0,0018	0,1870
Cypress	0,0119	0,1064	0,0133	0,0048	0,1364
Nightwatch.js	0,0026	0,0250	0,1195	0,0359	0,1830
Selenide	0,0025	0,0172	0,1195	0,0086	0,1477
TestCafe	0,0197	0,0424	0,1195	0,0086	**0,1902**
WebdriverIO	0,0013	0,0104	0,1195	0,0247	0,1559

Lõpptulemuste CR oli 5,87%, mis on väiksem kui 10%, mis tähendab, et lõpptulemused on sobilikud.

Lõpptulemustest selgus, et suurima koguhinnanguga vahend oli TestCafe, ent vahed CodeceptJS-i ning Nightwatch.js-iga olid suhteliselt väikesed. Lõpptulemuste lihtsustamiseks eemaldati sealt kõik ülejäänud vahendid ning veebilehitsejate toe kriteerium, mille tulemus oli kõigi allesjäänud vahendite puhul sama (tabel 7).

Tabel 7. Saaty meetodi lihtsustatud lõpptulemus

	Seadistamise lihtsus	**Õpikõvera suurus**	**Jõudlus**	**Koguhinnang**
CodeceptJS	0,0074	0,0582	0,0018	0,0674
Nightwatch.js	0,0026	0,0250	0,0359	0,0635
TestCafe	0,0197	0,0424	0,0086	**0,0707**

Lihtsustatud tulemustest selgus, et CodeceptJS-i tulemus oli TestCafe tulemusest 4,67% väiksem ning Nightwatch.js-i tulemus oli TestCafe tulemusest 10,18% väiksem. Üle poole Nightwatch.js-i tulemusest pärines vahendi tulemusest jõudluse arvestuses (tegemist oli võrdluse kiireima vahendiga ning selle paralleliseerimise võimekus töötas

probleemideta), samas CodeceptJS ja TestCafe ületasid seda seadistamise lihtsuse ja õpikõvera suuruse osas.

TestCafe puhul oli tegemist võrdluse variantidest parima kompromissiga – selle jõudlus oli küll alla keskmise, kuid see toetas testide paralleliseerimist (va. vana Microsoft Edge veebilehitseja puhul), see toetas kõiki soovitud veebilehitsejaid, selle seadistusprotsess oli võrdluse sujuvaim ning selle kasutamine oli suhteliselt lihtne. Seetõttu nõustuti Saaty meetodi tulemustega ning sobivaimaks kasutajaliidese testimise vahendiks valiti TestCafe.

5 Kasutajaliidese testide juurutamine ettevõtte siseveebis

Võrdluse võitjaks osutunud TestCafe abil koostati kasutajaliidese testid kogu ettevõtte siseveebi jaoks. Testid loodi TestCafe abil loodud näidistestide põhjal, mistõttu jätkati seal alustatud stiili, mille kohaselt testide ehitamisel leheküljeobjekti mudelit ei kasutatud.

Lisaks punktis 3.1.3 sõnastatud kasutajaliidese testide liikidele lisandus vajadus testida, et kasutajad ei näeks valikuid, mida nad näha ei tohi. Selleks loodi kaks testkasutajat: kasutaja A, kellel olid tavakasutaja õigused, ning kasutaja B, kellel olid kõik administratiivsed õigused. Administratiivseid toiminguid sooritati testides kasutajaga B ning kasutajaga A kontrolliti, et tavakasutajad ei saaks neid administratiivseid toiminguid sooritada. Seetõttu jagunesid loodud testid üldjoones kolmeks: kasutajavoo õnnestumist kontrollivad testid, veaolukorra tekkimist kontrollivad testid ning õiguste piiranguid kontrollivad testid. Kokku koostati 92 kasutajaliidese testi.

Testifailide struktuuri kirjeldamiseks kasutatakse näidisena osa testifailist, mida kasutatakse puhkuste funktsionaalsuse testimiseks tavakasutaja õigustega. Lisas 4 on nähtav testifaili algus, mis algab testifaili deklareerimisega fixture funktsiooni abil, millele edastatakse parameetrina testifaili nimetus. Seejärel kasutatakse meetodit page, millele edastatakse parameetrina veebilehekülje aadress, millel testimise protsess toimub. Seejärel kasutatakse meetodit beforeEach, millele parameetrina edastatav funktsioon kutsutakse välja enne igat antud failis paiknevat testi. Kuna TestCafe testid on üksteisest isoleeritud, on iga testi eel tarvis uuesti sisse logida, mistõttu on näidise puhul veebilehekülje aadressina kasutusel rakenduse avalehekülg ning enne igat testi logitakse sisse tavakasutajana, peale mida navigeeritakse puhkuste leheküljele ning oodatakse puhkuste tabeli tekkimist, mille olemasolu on vajalik kõigi antud failis paiknevate testide jaoks.

Lisas 5 nähtavas õiguste piirangute testis kontrollitakse, et tavakasutaja õigustega kasutaja ei näeks puhkuste tabeli kohal asuvas menüüs nelja raamatupidajatele ja tiimijuhtidele mõeldud valikut.

Lisades 6 ja 7 nähtavates testides sooritatakse uue puhkuse salvestamise protsess, ent lisas 6 nähtavas testis valitakse puhkuse algus- ja lõppkuupäevadeks korrektsed (tulevikus paiknevad) kuupäevad ning lisas 7 nähtavas testis vigased (minevikus paiknevad) kuupäevad, mistõttu nende andmetega puhkuse salvestamine vastavalt õnnestub ja ebaõnnestub. Õnnestuva testi puhul kontrollitakse, et kasutajale kuvatakse salvestamise õnnestumise teavitus, salvestatud puhkus lisatakse puhkuste tabelisse ja ei kuvata veateadet. Ebaõnnestuva testi puhul kontrollitakse, et kasutajale kuvatakse veateade ning ei kuvata salvestumise õnnestumise teavitust.

DOM elementide leidmiseks leheküljelt kasutati CSS (Cascading Style Sheets) valijaid (ingl. *selector*). Üksikute elementide puhul kasutati elementide identifitseerimiseks identifikaatori omadust, millele anti ainulaadne nimi, näiteks rakenduse menüüribal asuva sisselogimisnupu identifikaatori omaduseks määrati „login-button”. Elemendi leidmiseks mitme samalaadse elemendi seast kasutati kombinatsiooni klassi omadusest ja andmeomadustest, näiteks töötajate nimekirjas määrati kõigi töötajate nimede klassi omaduseks „person-name-link” ning nende andmeomaduseks „data-person” määrati töötaja siseveebi kasutaja unikaalne identifikaator.

Probleemseteks osutusid kolmandate osapoolte poolt loodud teekidest pärinevad Reacti komponendid, millest paljudele polnud võimalik soovitud omadusi määrata. Näiteks lisades 6 ja 7 nähtavate testide puhul ei olnud võimalik puhkuseliigi rippmenüü valikutele määrata identifikaatori omadust ja andmeomadusi, mistõttu määrati nende valikute eristamiseks neile unikaalne klassi omadus. Samuti kasutas nuppude eristamiseks klassi omadust kalendrist kuupäevade valimiseks kasutatav React Datepicker teek.

Testide käitamine toimub käsurealt, näiteks testide käitamiseks graafilise kasutajaliideseta Google Chrome veebilehitsejaga kasutatakse käsklust „npm run testcafe:chrome”, mis on rakenduse seadistusfailis määratud lühinimi käsklusele „testcafe "chrome:headless" src/tests/*.js -c 4”, kus parameeter „chrome:headless”

tähistab graafilise kasutajaliideseta Google Chrome veebilehitsejat ning parameeter „-c 4” tähistab testide jooksutamist paralleelselt neljas lõimes.

Testide käitamise ajal koostatakse testiraport vaikimisi formaadis, mille lihtsustatud näidis eelnevalt näidisena kasutatud testide kohta on nähtaval lisas 8. Testide tulemused rühmitatakse testifailide järgi, testide nimede juures kuvatakse märge vastava testi õnnestumise või ebaõnnestumise kohta, ebaõnnestunud testi puhul kuvatakse täpne rida testi lähtekoodis, mille tõttu test ebaõnnestus. TestCafe toetab ka mitmeid alternatiivseid testiraporti formaate [43]. Juhul kui vea silumisel peaks tekkima vajadus jälgimaks veebilehekülje visuaalset olekut, on võimalik luua alternatiivne käsklus testide käitamiseks kasutajaliidesega veebilehitsejaga.

Koostatud testide abil leiti ettevõtte siseveebist mitmeid vigu, mis tulenesid sellest, et testimisvahend sooritas veebileheküljel tegevusi kiiremini kui päris kasutajad seni suutnud olid. Seetõttu tekkis olukordi, kus osad andmed polnud veel sisse laetud, kuid test üritas sooritada tegevust, mis neid andmeid kasutas, mistõttu tekkis eesrakenduses veaolukord.

Koostatud testid asuvad ettevõtte koodivaramus ning on vajadusel kättesaadavad kõigile ettevõtte töötajatele. Testide koostamisel kasutati suurt osa TestCafe API-st, mistõttu on koostatud testid heaks näidiseks nii TestCafe kasutamise kohta kui ka kolmanda osapoole teekide kasutamise kohta kasutajaliidese testidega.

6 Võimalikud edasiarendused

Kuna ettevõttes ei kasutatud varasemalt kasutajaliidese teste, tuleks täieliku kasutajaliidese testimise lahenduse välja töötamiseks teha täiendavat uurimistööd, mis ei mahtunud käesoleva bakalaureusetöö skoopi.

Näiteks tuleks välja töötada metoodika kasutajaliidese testide käitamiseks järjest mitme veebilehitsejaga nõnda, et ühe veebilehitseja testide lõpptulemus ei mõjutaks teise veebilehitseja testide käiku ning testid ise ei peaks vastutama algandmete taastamise eest, mis poleks paljudel juhtudel isegi võimalik.

Tasuks uurida ka leheküljeobjekti mudeli kasutamist antud ettevõtte kontekstis ja TestCafe vahendi spetsiifiliselt ning hinnata, kas selle kasutamise eelised oleksid suuremad kui selle puudused.

Samuti võib mõne aasta pärast tekkida vajadus töö protsessi kordamiseks, kuna mitmed võrdluses olnud vahendid jõudsid stabiilse versioonini alles mõne viimase aasta jooksul ning nende areng on siiani kiire. Näiteks võrdluses viimaseks jäänud Cypressi suurimaks probleemiks oli veebilehitsejate ja teenuste tugi, ent Cypressi arendajad plaanivad tulevikus selles valdkonnas olukorda parandada. Lisaks võib tekkida ka uusi konkureerivaid kasutajaliidese testimise vahendeid.

7 Kokkuvõte

Käesoleval bakalaureusetööl oli kaks eesmärki: sobivaima automaatse kasutajaliidese testimise vahendi leidmine ettevõtte jaoks, kus automaatseid kasutajaliidese teste varasemalt ei kasutatud, ning kasutajaliidese testide näidislahenduse loomine lihtsustamaks ettevõttes töötavate tarkvaraarendajate õpikõverat.

Töö esimeses osas võrreldi seitset automaatset kasutajaliidese testimise vahendit: CodeceptJS, Cypress, Nightwatch.js, Selenide, Serenity BDD, TestCafe ja WebdriverIO. Vahendeid võrreldi nende jõudluse, stabiilsuse, paigaldamise ja kasutamise mugavuse, veebilehitsejate toe, veebilehitsejate üle interneti juhtimise toe, testide paralleliseerimise toe ning BrowserStack, CrossBrowserTesting ja SauceLabs teenuste toe järgi. Sobivaim vahend valiti kasutades Saaty meetodit, mille tulemusena selgus, et ettevõtte vajadustele vastab vahenditest enim TestCafe.

Töö teise osa tulemusena loodi kasutades TestCafe-d kasutajaliidese testide näidislahendus ettevõtte siseveebi näitel. Ettevõtte siseveebi kasutajaliidese testidega katmise käigus avastati ka mitmeid vigu, misläbi parandati rakenduse töökindlust.

8 Kasutatud kirjandus

[1] S. Bose (2020). End To End Testing: A Detailed Guide. https://www.browserstack.com/guide/end-to-end-testing (15.04.2020)
[2] O. Emmanuel (2018). The Problem with End-to-End Tests. https://levelup.gitconnected.com/the-problem-with-end-to-end-tests-65509df4bc7a (16.04.2020)
[3] M. Wacker (2015). Just Say No to More End-to-End Tests. https://testing.googleblog.com/2015/04/just-say-no-to-more-end-to-end-tests.html (15.04.2020)
[4] M. Fowler (2012). TestPyramid. https://martinfowler.com/bliki/TestPyramid.html (15.04.2020)
[5] M. Cohn (2009). The Forgotten Layer of the Test Automation Pyramid. https://www.mountaingoatsoftware.com/blog/the-forgotten-layer-of-the-test-automation-pyramid (15.04.2020)
[6] R.W. Saaty (1987). The analytic hierarchy process—what it is and how it is used. https://www.sciencedirect.com/science/article/pii/0270025587904738 (02.04.2020)
[7] G. Locatelli, M. Mancini (2012). A framework for the selection of the right nuclear power plant. http://eprints.lincoln.ac.uk/id/eprint/7016/1/A_framework_for_the_selection_of_the_right_nuclear_power_plant.pdf (15.04.2020)
[8] J. McCaffrey (2005). Test Run: The Analytic Hierarchy Process. https://docs.microsoft.com/en-us/archive/msdn-magazine/2005/june/test-run-the-analytic-hierarchy-process (15.04.2020)
[9] L. Sander (2010). AHP TDHLeadImage.png. https://commons.wikimedia.org/wiki/File:AHP_TDHLeadImage.png (16.04.2020)
[10] R.P. Hämäläinen, J. Mustajoki (2007). Web-HIPRE. http://hipre.aalto.fi/ (20.04.2020)
[11] R.P. Hämäläinen, J. Mustajoki (2003). Web-HIPRE Version History. http://hipre.aalto.fi/VersionHistory.html (22.04.2020)
[12] R.P. Hämäläinen, J. Mustajoki (2019). Enabling Java for Web-HIPRE. http://hipre.aalto.fi/JavaTutorial/java-index.html (22.04.2020)
[13] 123ahp.com (2018). my choice, my decision. http://www.123ahp.com/ (15.04.2020)
[14] C. Bromann (2019). Awesome Selenium. https://github.com/christian-bromann/awesome-selenium (15.02.2020)
[15] S. Orhus (2019). The awesome manifesto. https://github.com/sindresorhus/awesome/blob/master/awesome.md (14.04.2020)
[16] GitHub (2020). end-to-end-testing. https://github.com/topics/end-to-end-testing (15.02.2020)
[17] GitHub (2020). e2e. https://github.com/topics/e2e (15.02.2020)
[18] GitHub (2020). e2e-testing. https://github.com/topics/e2e-testing (15.02.2020)
[19] GitHub (2020). e2e-tests. https://github.com/topics/e2e-tests (15.02.2020)
[20] GitHub (2020). webdriver. https://github.com/topics/webdriver (15.02.2020)
[21] GitHub (2020). selenium. https://github.com/topics/selenium (15.02.2020)
[22] Google (2020). end-to-end testing javascript. https://www.google.com/search?q=end-to-end+testing+javascript (15.02.2020)
[23] Google (2020). end-to-end testing java. https://www.google.com/search?q=end-to-end+testing+java (15.02.2020)
[24] Conductor (2017). io.ddavison : conductor. https://search.maven.org/artifact/io.ddavison/conductor (29.02.2020)
[25] DalekJS (2017). DalekJS Base Framework. https://github.com/dalekjs/dalek (15.02.2020)
[26] Protractor (2018). Protractor - end-to-end testing for AngularJS. https://www.protractortest.org/ (15.02.2020)
[27] Galen Framework (2017). Galen Framework | Automated testing of responsive design. http://galenframework.com/ (15.02.2020)
[28] Zombie (2018). Zombie.js. http://zombie.js.org/ (15.02.2020)
[29] CasperJS (2018). CasperJS. https://github.com/casperjs/casperjs (15.02.2020)
[30] Puppeteer (2020). Puppeteer. https://github.com/puppeteer/puppeteer (15.02.2020)

[31] Microsoft (2020). Playwright. https://github.com/microsoft/playwright (13.03.2020)
[32] Karma (2019). Frequently Asked Questions. https://karma-runner.github.io/4.0/intro/faq.html (19.04.2020)
[33] SlimerJS (2018). SlimerJS. https://slimerjs.org/ (15.02.2020)
[34] PhantomJS (2018). PhantomJS - Scriptable Headless Browser. https://phantomjs.org/ (15.02.2020)
[35] Segment (2017). Nightmare. http://www.nightmarejs.org/ (15.02.2020)
[36] StatCounter (2020). Browser Market Share Estonia. https://gs.statcounter.com/browser-market-share/all/estonia (02.03.2020)
[37] CodeceptJS (2020). CodeceptJS. https://codecept.io/ (13.03.2020)
[38] Cypress (2020). JavaScript End to End Testing Framework. https://www.cypress.io/ (15.02.2020)
[39] W3C Browser Testing and Tools Working Group (2020). WebDriver. https://www.w3.org/TR/webdriver/ (15.02.2020)
[40] Software Freedom Conservancy (2020). Selenium History. https://www.selenium.dev/history/ (04.04.2020)
[41] Pine View Software AS (2019). Getting Started. https://webdriver.io/docs/gettingstarted.html (15.02.2020)
[42] WebdriverIO community (2018). `browser.setValue` only does appending without clearing textfield on Chrome 70 on Chromdriver 2.43. https://github.com/webdriverio/webdriverio/issues/3024 (19.03.2020)
[43] TestCafe (2020). A node.js tool to automate end-to-end web testing | TestCafe. https://devexpress.github.io/testcafe/ (15.022020)
[44] Apple (2020). Safari. https://www.apple.com/safari/ (15.04.2020)
[45] CodeceptJS (2019). Roadmap. https://github.com/Codeception/CodeceptJS/wiki/Roadmap (22.04.2020)
[46] CrossBrowserTesting (2020). testcafe-browser-provider-crossbrowsertesting. https://github.com/crossbrowsertesting/testcafe-browser-provider-crossbrowsertesting (21.04.2020)
[47] OpenJS Foundation (2020). WebdriverIO · Next-gen browser automation test framework for Node.js. https://webdriver.io/ (15.02.2020)
[48] CodeceptJS community (2019). Process doesnt end when running in parallel. https://github.com/Codeception/CodeceptJS/issues/1997 (14.04.2020)
[49] Cypress community (2020). Proposal: IE 11 Support. https://github.com/cypress-io/cypress/issues/6423 (22.04.2020)
[50] Cypress community (2020). Proposal: macOS Safari Support. https://github.com/cypress-io/cypress/issues/6422 (22.04.2020)
[51] Cypress community (2019). Add configuration option for disabling screenshots in cypress.json. https://github.com/cypress-io/cypress/issues/5029 (26.03.2020)
[52] Cypress community (2020). 4.0.1 cypress run: Firefox instances not closed at end of spec. https://github.com/cypress-io/cypress/issues/6392 (27.03.2020)
[53] Pine View Software AS (2019). Getting Started. https://nightwatchjs.org/gettingstarted/configuration/ (13.03.2020)
[54] Codeborne (2020). Selenide: concise UI tests in Java. https://selenide.org/index.html (30.03.2020)
[55] Martinig & Associates (2013). Selenide - Concise UI Tests in Java. https://www.methodsandtools.com/tools/selenide.php (06.04.2020)
[56] Codeborne (2016). Selenide 4.2 released. https://selenide.org/2016/12/30/selenide-4.2/ (06.04.2020)
[57] J.F. Smart (2016). The Serenity Reference Manual. http://thucydides.info/docs/serenity-staging/ (15.04.2020)
[58] Serenity BDD (2018). What is Serenity. http://www.thucydides.info/#/whatisserenity (15.04.2020)

Lisa 1 – Näidistestidega testitavad kasutajavood

Ettevõtte töötajate nimekirja kaudu isiku profiilile liikumine ning tema istekoha vaatamine

1. Kasutaja avab rakenduse
2. Kasutaja vajutab ekraani üleval paremal nurgas olevat “Login” nuppu, mille peale avatakse aken sisselogimisvormiga
3. Kasutaja sisestab enda kasutajanime ja parooli ning vajutab “Submit” nuppu, mille peale kasutaja logitakse sisse
4. Kasutaja vajutab pealehel “People” kasti peale, mille peale kuvatakse talle ettevõtte töötajate nimekirja
5. Kasutaja vajutab ühe töötaja nime peale, mille peale kuvatakse talle töötaja profiil koos istumisplaaniga, mis sisaldab töötaja istekohta

Põhipuhkuse avalduse sisestamine vastavast menüüst

1. Kasutaja avab rakenduse
2. Kasutaja vajutab ekraani üleval paremal nurgas olevat “Login” nuppu, mille peale avatakse aken sisselogimisvormiga
3. Kasutaja sisestab enda kasutajanime ja parooli ning vajutab “Submit” nuppu, mille peale kasutaja logitakse sisse
4. Kasutaja vajutab pealehel “Personal” kasti peale, mille peale kuvatakse talle alamenüü kastide kujul
5. Kasutaja vajutab alamenüüs “Day off” kasti peale, mille peale kuvatakse talle leht tema puhkuste ülevaatega
6. Kasutaja vajutab “Add new vacation” nupu peale, mille peale kuvatakse talle uue puhkuse sisestamise vorm
7. Kasutaja valib puhkuse liigi (“Annual holiday”) ning algus- ja lõpukuupäeva, mille peale kuvatakse talle “Save” nupp

8. Kasutaja vajutab “Save” nuppu, mille peale salvestatakse ja lisatakse ülevaatesse ta puhkus

Viimase kuu töötundide sisestamine enda profiililt

1. Kasutaja avab rakenduse
2. Kasutaja vajutab ekraani üleval paremal nurgas olevat “Login” nuppu, mille peale avatakse aken sisselogimisvormiga
3. Kasutaja sisestab enda kasutajanime ja parooli ning vajutab “Submit” nuppu, mille peale kasutaja logitakse sisse
4. Kasutaja vajutab ekraani üleval paremal nurgas enda nime peale, mille peale kuvatakse talle tema profiil
5. Kasutaja valib lehe all otsas olevast menüüst valiku “Projects”, mille peale kuvatakse talle nimekiri klientidest, mille aktiivsete projektidega ta käesoleval aastal seotud olnud on
6. Kasutaja vajutab soovitud kliendi peale, mille peale kuvatakse talle antud kliendi aktiivsed projektid, millega ta käesoleval aastal seotud olnud on
7. Kasutaja vajutab projekti real olevat pliiatsikujulist nuppu, mille peale määratakse antud projekt muudetavasse olekusse
8. Kasutaja sisestab vastavasse lahtrisse eelneva kuu töötunnid ning vajutab linnukese-kujulist nuppu, mille peale antud projekti töötunnid salvestatakse ja uuendatakse tabelis

Lisa 2 – Näidistestide kirjeldused

Sisselogimine õige kasutajanime ja õige parooliga

1. Kasutaja avab rakenduse
2. Kasutaja vajutab sisselogimise vormi avamise nuppu
3. Kasutaja sisestab kasutajanime lahtrisse õige kasutajanime
4. Kasutaja sisestab parooli lahtrisse õige parooli
5. Kasutaja vajutab sisselogimise nuppu
6. Kontroll, et leheküljel ei eksisteeriks veateadet
7. Kontroll, et menüüribal eksisteeriks kasutaja täisnimi

Sisselogimine õige kasutajanime ja tühja parooliga

1. Kasutaja avab rakenduse
2. Kasutaja vajutab sisselogimise vormi avamise nuppu
3. Kasutaja sisestab kasutajanime lahtrisse õige kasutajanime
4. Kasutaja vajutab sisselogimise nuppu
5. Kontroll, et leheküljel eksisteeriks veateade
6. Kontroll, et menüüribal ei eksisteeriks kasutaja täisnime

Sisselogimine tühja kasutajanime ja õige parooliga

1. Kasutaja avab rakenduse
2. Kasutaja vajutab sisselogimise vormi avamise nuppu
3. Kasutaja sisestab parooli lahtrisse õige parooli
4. Kasutaja vajutab sisselogimise nuppu
5. Kontroll, et leheküljel eksisteeriks veateade
6. Kontroll, et menüüribal ei eksisteeriks kasutaja täisnime

Sisselogimine õige kasutajanime ja vigase parooliga

1. Kasutaja avab rakenduse
2. Kasutaja vajutab sisselogimise vormi avamise nuppu
3. Kasutaja sisestab kasutajanime lahtrisse õige kasutajanime
4. Kasutaja sisestab parooli lahtrisse vigase parooli
5. Kasutaja vajutab sisselogimise nuppu
6. Kontroll, et leheküljel eksisteeriks veateade
7. Kontroll, et menüüribal ei eksisteeriks kasutaja täisnime

Sisselogimine vigase kasutajanime ja õige parooliga

1. Kasutaja avab rakenduse
2. Kasutaja vajutab sisselogimise vormi avamise nuppu
3. Kasutaja sisestab kasutajanime lahtrisse vigase kasutajanime
4. Kasutaja sisestab parooli lahtrisse õige parooli
5. Kasutaja vajutab sisselogimise nuppu
6. Kontroll, et leheküljel eksisteeriks veateade
7. Kontroll, et menüüribal ei eksisteeriks kasutaja täisnime

Puhkuseavalduse sisestamine õigete andmetega

1. Sisselogimine õige kasutajanime ja õige parooliga
2. Kasutaja vajutab “Personal” kastile
3. Kasutaja vajutab “Day off” kastile
4. Kontroll, et leheküljel eksisteeriks kasutaja puhkuste tabel
5. Kontroll, et puhkuste tabelis ei eksisteeriks “Saved” staatusega puhkust
6. Kasutaja vajutab uue puhkuse lisamise nupule
7. Kasutaja vajutab puhkuseliigi rippmenüüst valiku “Põhipuhkus”
8. Kasutaja valib puhkuse alguskuupäevaks järgmise kuu 15. kuupäeva
9. Kasutaja valib puhkuse lõpukuupäevaks järgmise kuu 21. kuupäeva
10. Kasutaja vajutab puhkuse salvestamise nupule
11. Kontroll, et leheküljel eksisteeriks salvestamise õnnestumise teade

12. Kontroll, et leheküljel ei eksisteeriks veateadet
13. Kontroll, et puhkuste tabelis eksisteeriks “Saved” staatusega puhkus

Puhkuseavalduse sisestamine vigaste kuupäevadega (minevikku)

1. Sisselogimine õige kasutajanime ja õige parooliga
2. Kasutaja vajutab “Personal” kastile
3. Kasutaja vajutab “Day off” kastile
4. Kontroll, et leheküljel eksisteeriks kasutaja puhkuste tabel
5. Kasutaja vajutab uue puhkuse lisamise nupule
6. Kasutaja vajutab puhkuseliigi rippmenüüst valiku “Põhipuhkus”
7. Kasutaja valib puhkuse alguskuupäevaks eelmise kuu 15. kuupäeva
8. Kasutaja valib puhkuse lõpukuupäevaks eelmise kuu 21. kuupäeva
9. Kasutaja vajutab puhkuse salvestamise nupule
10. Kontroll, et leheküljel ei eksisteeriks salvestamise õnnestumise teadet
11. Kontroll, et leheküljel eksisteeriks veateade

Jooksva kuu töötundide sisetamine korrektse arvuga

1. Sisselogimine õige kasutajanime ja õige parooliga
2. Kasutaja vajutab menüüribal enda nime peale
3. Kasutaja valib profiili menüüst valiku “Projektid”
4. Kasutaja avab klientide valikust testkliendi
5. Kasutaja vajutab testprojekti töötundide muutmise nuppu
6. Kasutaja sisestab jooksva kuu töötundide lahtrisse väärtuse “123”
7. Kasutaja vajutab testprojekti töötundide salvestamise nuppu
8. Kontroll, et testprojekti jooksva kuu töötundide väärtus oleks “123”

Jooksva kuu töötundide sisestamine negatiivse arvuga

1. Sisselogimine õige kasutajanime ja õige parooliga
2. Kasutaja vajutab menüüribal enda nime peale

3. Kasutaja valib profiili menüüst valiku “Projektid”
4. Kasutaja avab klientide valikust testkliendi
5. Kasutaja vajutab testprojekti töötundide muutmise nuppu
6. Kasutaja sisestab jooksva kuu töötundide lahtrisse väärtuse “-123”
7. Kasutaja vajutab testprojekti töötundide salvestamise nuppu
8. Kontroll, et testprojekti jooksva kuu töötundide väärtus oleks “0”

Isiku istekoha leidmine tema profiililt

1. Sisselogimine õige kasutajanime ja õige parooliga
2. Kasutaja vajutab “People” kastile
3. Kasutaja vajutab töötajate nimekirjas testkasutaja nime peale
4. Kontroll, et profiilil eksisteeriks istekohtade kaart
5. Kontroll, et istekohtade kaardil eksisteeriks testkasutaja istekoht

Lisa 3 – Saaty fundamentaalskaala väärtused

Tabel 8. Saaty fundamentaalskaala väärtused

Allikas: [6]

Tähtsuse intensiivsus absoluutskaalal	**Tähendus**	**Selgitus**
1	Võrdne tähtsus	Kahe tegevuse mõju eesmärgini jõudmisel on sama suur
3	Mõõdukas tähtsus	Kogemus ja hinnang annavad ühele tegevusele eelise
5	Oluline või tugev tähtsus	Kogemus ja hinnang annavad ühele tegevusele tugeva eelise
7	Väga tugev tähtsus	Üks tegevus on teisega võrreldes tugevalt eelistatud ning see on praktikas tõestatav
9	Ekstreemne tähtsus	Tugevaim võimalik eelistus ühe tegevuse osas võrreldes teisega
2, 4, 6, 8	Kahe väärtuse vahelised väärtused	Kui kahe väärtuse vahel on vaja leida kompromiss

Lisa 4 – Ettevõtte siseveebi testifaili alguse näidis

```
import {Selector} from "testcafe";
import {INDEX_URL, REGULAR, login} from "./utils";

fixture("vacations regular")
    .page(INDEX_URL)
    .beforeEach(async t => {
        await login(t, REGULAR);

        await t
            .click("#personal-block")
            .click("#vacations-block")
            .expect(Selector("#vacations-table").exists)
            .ok();
    });
```

Joonis 3. Ettevõtte siseveebi testifaili alguse näidis

Lisa 5 – Ettevõtte siseveebi õiguste piirangute testi näidis

```
test("does not show admin buttons", async t => {
    await t
        .expect(Selector(`.tab-button[data-
tab="STAFF_VACATIONS_TAB"]`).exists)
        .notOk()
        .expect(Selector(`.tab-button[data-
tab="SICKNESS_LEAVES_TAB"]`).exists)
        .notOk()
        .expect(Selector(`.tab-button[data-
tab="PROCESS_TAB"]`).exists)
        .notOk()
        .expect(Selector(`.tab-button[data-
tab="BALANCE_TAB"]`).exists)
        .notOk();
});
```

Joonis 4. Ettevõtte siseveebi õiguste piirangute testi näidis

Lisa 6 – Ettevõtte siseveebi õnnestuva testi näidis

```
test("creates a new vacation appliction", async t => {
    await t
        .click("#add-regular-vacation-button")
        .click("#vacation-type-selection")
        .click(".vacation-type-POHI")
        .click(".vacation-period:first-of-type .start-date")
        .click(".react-datepicker__navigation--next")
        .click(".react-datepicker__day--015")
        .click(".vacation-period:first-of-type .end-date")
        .click(".react-datepicker__day--021")
        .click("#add-vacation-save")
        .expect(Selector(".ant-notification-notice-icon-
success").exists)
        .ok()
        .expect(Selector(".ant-notification-notice-icon-
error").exists)
        .notOk()
        .expect(Selector(`.vacations-table-row[data-
status="NEW"]`).exists)
        .ok();
});
```

Joonis 5. Ettevõtte siseveebi õnnestuva testi näidis

Lisa 7 – Ettevõtte siseveebi ebaõnnestuva testi näidis

```
test("does not allow creating an application in the past", async t =>
{
    await t
        .click("#add-regular-vacation-button")
        .click("#vacation-type-selection")
        .click(".vacation-type-POHI")
        .click(".vacation-period:first-of-type .start-date")
        .click(".react-datepicker__navigation--previous")
        .click(".react-datepicker__day--015")
        .click(".vacation-period:first-of-type .end-date")
        .click(".react-datepicker__day--021")
        .click("#add-vacation-save")
        .expect(Selector(".ant-notification-notice-icon-
error").exists)
        .ok()
        .expect(Selector(".ant-notification-notice-icon-
success").exists)
        .notOk();
});
```

Joonis 6. Ettevõtte siseveebi ebaõnnestuva testi näidis

Lisa 8 – Ettevõtte siseveebi testiraporti näidis

```
 Running tests in:
 - Chrome 81.0.4044.138 / Windows 10
 Vacations regular
 √ does not show admin buttons
 × creates a new vacation appliction
   1) AssertionError: expected false to be truthy
      Browser: Chrome 81.0.4044.138 / Windows 10
         38 |        .click(".react-datepicker__day--015")
         39 |        .click(".vacation-period:first-of-type .end-date")
         40 |        .click(".react-datepicker__day--021")
         41 |        .click("#add-vacation-save")
         42 |        .expect(Selector(".ant-notification-notice-icon-
success").exists)
       > 43 |        .ok()
         44 |        .expect(Selector(".ant-notification-notice-icon-
error").exists)
         45 |        .notOk()
         46 |        .expect(Selector(`.vacations-table-row[data-
status="NEW"]`).exists)
         47 |        .ok();
         48 |});
         at <anonymous> (src\tests\vacations_regular.js:43:10)
         at <anonymous> (src\tests\vacations_regular.js:29:1)
         at <anonymous>
      (node_modules\testcafe\src\api\wrap-test-function.js:17:28)
         at TestRun._executeTestFn
      (node_modules\testcafe\src\test-run\index.js:294:19)
         at TestRun.start
      (node_modules\testcafe\src\test-run\index.js:345:24)

 √ does not allow creating an application in the past

 1/3 failed (28s)
```

Joonis 7. Ettevõtte siseveebi testiraporti näidis