TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Liina Savtšik 164170IABB

UI TESTIMISE AUTOMATISEERIMISE TÖÖRIISTA VALIMINE ANGULAR'I PÕHILISELE IDUTOOTELE

bakalaureusetöö

Juhendaja: Deniss Kumlander
Doktorikraad

Tallinn 2019

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: Liina Savtšik

20.05.2019

Annotatsioon

Antud balalaureusetöö põhieesmärk on analüüsida ja valida sobivaim UI testimise automatiseerimise tööriist Angular'i põhilisele idutootele selle töö parimaks korraldamiseks. Analüüsimiseks ja võrdlemiseks olid valitud kõige efektiivsemad ja jõukohasemad vahendid: Selenium, Protractor, TestComplete, Screenster ja QF-Test.

Töö käigus tutvustatakse UI testimist ja selle automatiseerimist, kirjeldatakse selle üldist olulisust ja aktuaalsust vastavas projektis. Erinevate vahendite analüüsi käigus tuuakse välja nende positiivsed ja negatiivsed küljed koos praktiliste näidetega. Analüüsile järgeb vahendite võrdlus.

Võrdluse tulemusena valiti QF-Test tööriist, mis osutus kõige paremini iduettevõtte nõudeid rahuldatavaks tööriistaks UI testide automatiseerimiseks.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 26 leheküljel, 3 peatükki, 8 joonist, 2 tabelit.

Abstract

Selection the UI Test Automation Tool for Angular Based Startup Product

The main purpose of this bachelor's thesis is to analyze and select the most appropriate UI test automation tool for Angular based startup product which could afford better work management for the startup company. There were chosen the most effective and affordable testing tools like Selenium, Protractor, TestComplete, Screenster and QF-Test for analysis and comparison.

The beginning of this thesis gives an introduction to the main topic of UI testing and its automation, explains its importance and relevance in the given project.

In the course of analysing various tools there were elaborated their positive and negative aspects with the practical examples. When the analysis is done, resources comparison starts.

As a result of tools comparison, the QF-Test tool was chosen as a best option. This tool satisfies most of the given requirements for UI test automation tool for the current startup product.

The thesis is in Estonian and contains 26 pages of text, 3 chapters, 8 figures, 2 tables.

Lühendite ja mõistete sõnastik

AI	*Artificial Intelligence*, tehisintellekt
Angular 6	Platvorm 6. versiooni mobiil- ja töölaua veebirakenduste loomiseks
API	*Application Programming Interface*, funktsioonide ja protseduuride kogum rakenduste loomiseks
Azure	Microsofti pilveplatvorm
BDD	*Behavior-Driven Development*, käitumisel põhinev arendus
CSV	*Comma-Separated Values*, tekstiformaat tabelandmete esitamiseks
DDT	*Data-Driven Testing*, tarkvara testimismeetod, mis võimaldab lahutada andmeid testkäskudest
Debug	Silumine rakenduse vigade tabamiseks ja käitumise jälgimiseks
E2E	*End-To-End*, testimise tüüp kõikide rakendusse integreeritud komponentide töötamise kontrollimiseks
GUI	*Graphical User Interface*, graafiline kasutajaliides
HTML	*HyperText Markup Language*, veebilehtede märgistuskeel
Jira	Tarkvara arendusprojektide planeerimiseks ja jälgimiseks tiimis
KPI	*Key Performance Indicators*, tulemuslikkuse peamised näitajad
OCR	*Optical Character Recognition*, optiline märgituvastus
Open-source	Avatud lähtekoodiga tööriist
Selenium Server	Server, mille kaudu transporditakse käsud Selenium WebDriverist kaugarvuti veebibrauserisse
Startup	Iduettevõte, mis on suunatud kiirele kasvule
TFC	*Team Foundation Server*, tööriist, mis võimaldab täielikku meeskondlikku arendusprojekti haldamist
UI	*User Interface*, kasutajaliides
Unit testid	Programmeerimiskoodi väikeste osade testimine
XML	Extensible Markup Language, rakenduste leiendatav märgistuskeel

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Iga rakenduse kvaliteeti ja nõutavust turul kirjeldavad alati mitu omadust. Olenevalt valdkonnast võivad need erineda, kuid arendajad püüavad iga rakenduse teha kasutajale maksimaalselt mugavaks ja hoida neid võimalike vigade eest. Selleks, et seda saavutada, ei piisa ainult tähelepanelikust koodi kirjutamisest ja kõrgest haridusest. Selleks on mõeldud testimine, mis peab algama juba koodi kirjutamise ajal.

Selle bakalaureusetöö eesmärk on leida sobiv tööriist ühe iduettevõtte jaoks, mis aitaks neil edukalt automatiseerida UI (*User Interface*) teste ilma suure rahakuluta. Seejuures peaks valitud tööriist olema ka piisavalt efektiivne ja usaldusväärne. Praegu testitakse rakendust manuaalselt, aga kui ettevõtte poole hakkavad pöörduma suuremad kliendid, muutub rakenduse haldamine keerulisemaks ja suureneb ka tööjõu vajadus. UI testimise automatiseerimine lahendaks tekkivaid probleeme, mis on seotud kliendibaasi laienemise ja ülesannete hulga suurenemisega.

Töö eesmärgi saavutamiseks ja ettevõtte vajaduste täitmiseks on vaja analüüsida UI testide automatiseerimise vahendeid ja leida sobivaim, arvestades nende eeliseid ja puuduseid.

Vaadeldavas iduettevõttes korraldati ka küsitlus seal töötavate testijate hulgas, kes on otsitava testimistööriista tulevased otsekasutajad (Lisa 1). Küsitluse eesmärk oli välja selgitada töövahendile esitatavad nõuded tuginedes varasemale testijate kogemusele.

Bakalaureusetöö koosneb kolmest osast. Esimeses osas antakse sissejuhatust teemasse, millel põhineb antud bakalaureusetöö. Teises osas analüüsitakse erinevaid rakenduste UI testimise automatiseerimise vahendeid ning tuuakse välja nende tugevad ja nõrgad küljed, tuginedes suuremal määral testitava rakenduse omadustele. Kolmandas osas koondatakse rakenduse, ettevõtte ja testijate nõuded otsitavale vahendile ja võrreldakse analüüsitavaid lahendusi.

2 UI testimise tutvustus

Rakenduse arendamise protsessis üks olulisemaid etappe on testimine, mis peab toimuma paralleelselt iga arendamise sammuga. Olenevalt testimise eesmärgist ja testitavast valdkonnast neid samuti eristatakse. On olemas näiteks Unit testid, UI testimine ja teised. Antud töös vaadeldakse UI testimist.

UI testimise eesmärk on kontrollida süsteemi lõppkasutaja vaatenurgast, sest UI on alati esimene, mida kasutaja näeb, ja just see loob tema esmamulje kogu organisatsioonist. Seega selleks, et meelitada potentsiaalset klienti, on vaja tagada kvaliteetne ja usaldusväärne kasutajaliides, mida aitavad saavutada arvukad UI testid.

UI test koosneb tavaliselt kolmest üldisest etapist:

1. kasutaja interaktsioonide jäljendamine (viidetele vajutamine, teksti sisestamine, kursori liikumine jne);
2. lehekülje elementide kättesaamine;
3. testide täitmise kontroll.

Kõik need etapid aitavad tuvastada vigu ja elimineerida neid võimalikult kiiresti. Väikeste projektide korral piisab manuaalsest testimisest, sest testitavaid protsesse ei ole nii palju, et selleks tööjõudu ja raha kulutades võiks ettevõte kahjumisse jääda. Suuremate ja kiiresti kasvavate iduettevõtete korral nõuab manuaalne testimine päris suuri ressursse ja kulusid ning mida suurem on organisatsioon, seda keerulisem on tagada heatasemeline ja ladus testimine.

2.1 GUI testimine

GUI (*Graphical User Interface*) on kasutajaliidese alaliik, mis puudutab ainult graafilisi elemente [1], mida kasutaja saab näha või lugeda. GUI testimine kontrollib kasutajaliidese ühtsust ja mugavust kasutaja jaoks. Selleks on vaja kindlaks teha projekti eesmärk, sihtturu spetsiifika, piirkond, kus seda kasutama hakatakse, tehnilised ja teised

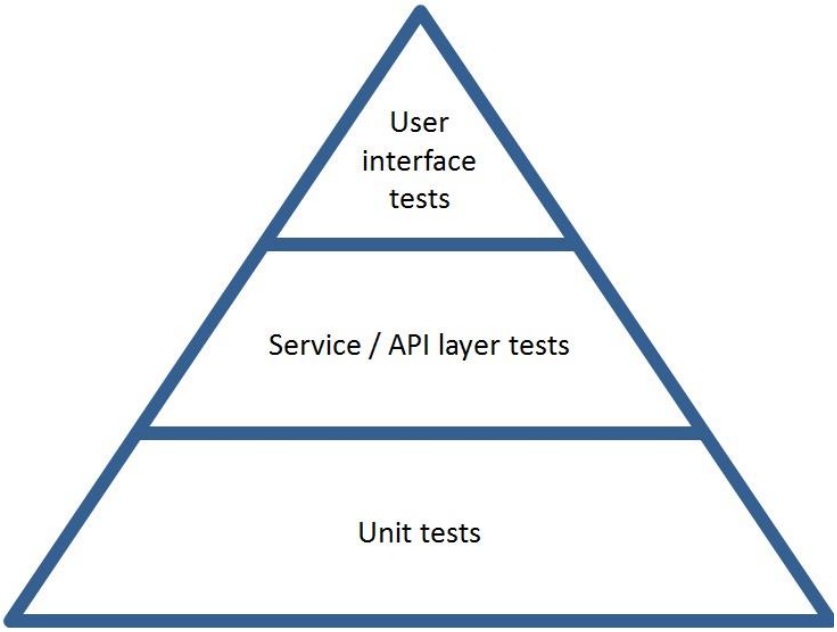
olulised aspektid, mis võiksid lõpptulemust mõjutada. Alltoodud loendis on nimetatud osa testjuhtumitest, mida võib kontrollida GUI testimisega [2]:

- sama font ja suurus kogu rakenduses;
- värvid ja üldine stiil on kooskõlas organisatsiooni spetsiifikaga;
- kõik ikoonid on ühes stiilis;
- kasutajaliides ei sisalda visuaalseid vastuolusid;
- navigatsioon on intuitiivne;
- väljade suurus vastab nõutavale informatsioonile;
- veateated on korrektsed.

Eelnevalt nimetatud omadusi on efektiivsemalt testida projekti arendamise algetappis, kui tegeldakse veel kasutajaliidese prototüübiga [3]. Testida on võimalik kas manuaalselt või automatiseerimistarkvara kaudu. Testija kontrollib sammhaaval, kas kõik graafilised elemendid on kooskõlas dokumentatsioonis kirjeldatud nõuetega või mitte. Niisamuti nagu teiste projekti osade testimises, peab testija tulemuste põhjal koostama dokumentatsiooni ja edastama selle edasiseks parandamiseks teistele meeskonnaliikmetele.

2.2 UI testimise automatiseerimine

Manuaalse testimise käigus on väga suur tõenäosus teha näpuvigu ning jätta mõned süsteemi vead ja puudused vahele. Projekti arenemise käigus saabub kindlasti hetk, kui töötavad testijad on juba ülekoormatud suure testide hulga tõttu ja uute testijate palkamine ei ole enam rahaliselt võimalik. Sellise olukorra vältimiseks võib teste automatiseerida, ootamata, et see muutuks hädavajaduseks. Joonisel 1 on esitatud testimise automatiseerimise püramiid, kus alt üles kahaneb vajalikke testide hulk ning kasvab nende täitmisaeg, lõppkasutaja asjakohasus ja samas ka haldamise maksumus [4]. Unit testid on püramiidi põhiosas, järgmine aste on API (*Application Programming Interface*) kihi testimine ja UI testid on püramiidi tipus, mis tähendab, et neid on tavaliselt kõige vähem, nende haldamine on kõige kulukam ja nad on kõige aeglasemad. Vaatamata sellele, et enamik neist omadustest ei ole eelised, on UI testimise automatiseerimisel asendamatu osa parima kvaliteedi tagamisel lõppkasutajale.



Joonis 1. Testimise automatiseerimise püramiid [4].

Üks UI testide automatiseerimise põhieeliseid on testide koostamine mitte ainult koodipõhiliselt. Näiteks *record-playback* on väga edukas viis testida rakenduse kasutajaliidest testijal, kes isegi ei tunne programmeerimist [5]. Kõik, mida on vaja teha, on teostada testitav protsess ainult üks kord ja kasutatav tööriist salvestab selle käigu edasiseks kuvamiseks juba ilma inimese osaluseta.

Veebirakenduste UI testide automatiseerimiseks on mitu lahendust, mille hulgast ettevõte võib valida olenevalt oma rakenduse platvormist ja suurusest. Automatiseerimiseks võib valida koodipõhiseid testimisraamistikke, töölaua testimise tööriistu või pilvepõhiseid testimisplatvorme [6]. Järgnevates peatükkides vaadeldakse iga tüüpi vahendeid.

3 Vahendite analüüs

UI testimise automatiseerimise vahendi analüüsimiseks on vaja kõigepealt määrata olulisemad kriteeriumid, mis on vajalikud tulevase automatiseeritult testitava idutoote spetsiifika ja kasutatavate tehnoloogiate seisukohal. Seega on vaja analüüsida tööriistu mitte tervikuna, vaid pöörates tähelepanu just nendele omadustele, mis on testitava rakendusega kooskõlas. On oluline, et testitav rakendus on veebipõhine ning ühildub kõikide brauserite viimaste versioonidega. Samuti on teada, et rakendus on loodud Angular 6 raamistiku ja Azure platvormi kasutusel ning on kirjutatud Typescript programmeerimiskeeles. Tarkvara võimaldab luua infolaudu, kus saaks kuvada kõiki äriettevõttele analüüsimiseks vajalikke majanduslikke andmeid. Loodud infolaud sisaldab tabeleid, maatrikseid, kohandatud maatriksite aruandeid, diagramme ja KPI (*Key Performance Indicators*) vidinaid.

Järgnevalt vaadeldakse levinumaid kasutajaliidese testimise automatiseerimise vahendeid, mida on korduvalt mainitud selleteemalistes artiklites ning mis on aastate jooksul võitnud ja õigustanud kasutajate usalduse.

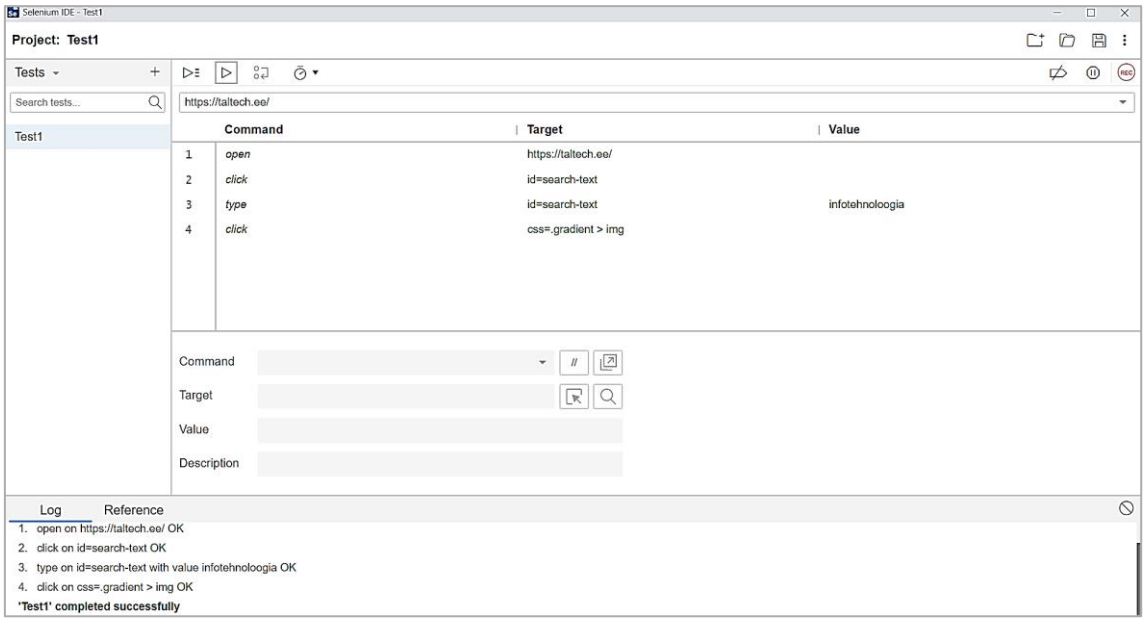
3.1 Selenium

Selenium on *open-source* töörist brauserite automatiseerimitud haldamiseks, mis omakorda kujutab endast põhiideed veebirakenduste testimise automatiseerimises [7]. Selenium pakub oma töö haldamiseks mitu varianti, näiteks Selenium IDE ja Selenium WebDriver.

Selenium IDE on lahendus väikeste test-stsenaariumite arendamiseks ja positsioneerib ennast vahendina, millest võib alustada tutvumist UI testimisega ja Seleniumi toodetega [8]. Test-stsenaariumite loomiseks on *record-playback* meetod ainult läbi Mozilla Firefox või Google Chrome brauseri rakenduse. Kasutaja ülesanne on sooritada testitav tegevus ning seejärel salvestab rakendus kõik alamtegevused (klikkimine, trükkimine) käskudena ja genereerib testi, mida hiljem võib käivitatada ka ilma inimese osaluseta. Teiste platvormide testimiseks on vaja alla laadida lisatooteid, nagu Selenium SIDE Runner, mis võimaldab seda käsurea kaudu.

Selenium IDE plussiks on see, et kasutajaliides on väga lihtne ja arusaadav isegi inimesele, kes pole kunagi varem testimisvahenditega kokku puutunud. Joonisel 2 on näidatud Selenium IDE testimiskeskkond, kus on esitatud edukalt käivitatud test, mis kontrollib otsinguriba kasutamist lõppkasutaja vaatenurgast. Test koosneb neljast sammust ehk neljast käsust:

- samm 1: *open* – avatakse veebilehekülg https://taltech.ee/;
- samm 2: *click* – vajutatakse otsinguribale;
- samm 3: *type* – sisestatakse tekst „infotehnoloogia“;
- samm 4: *click* – vajutatakse otsingunupule.

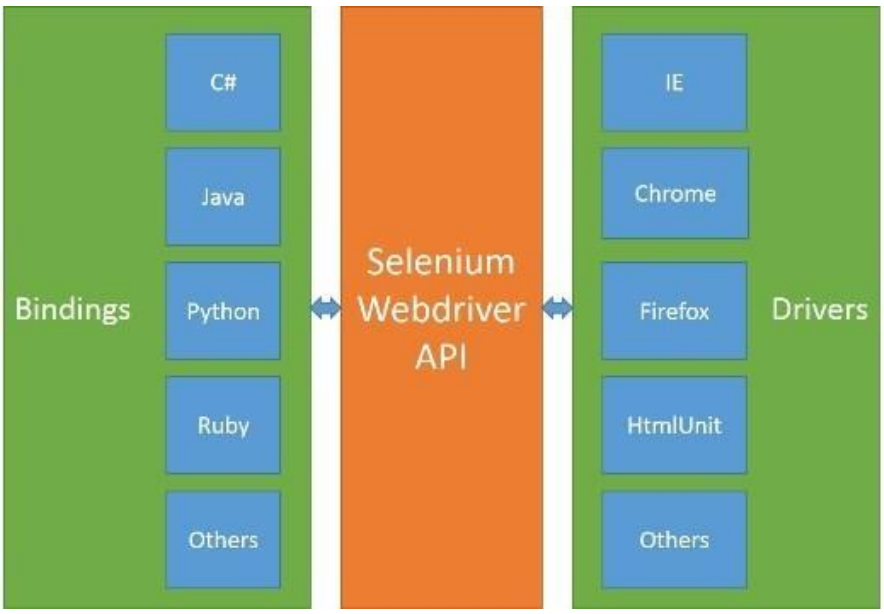


Joonis 2. Selenium IDE testimiskeskkond.

Kõik käsud genereerib programm. Mingi tegevuse käsitsi lisamisel on kõik olemasolevad Seleniumi käsud nähtavad ilmuvas loendis, mis aitab testijal vigu vältida. Käsku käsitsi sisestades tuleb olla ikka tähelepanelik, sest Selenium IDE on tundlik tühikute suhtes, nii et kui käsu ees või lõpus on liigne tühik, ei suuda Selenium seda käsku tuvastada [9]. Samuti võib suureks miinuseks pidada raporteerimise puudumist uuemates Selenium IDE versioonides.

Keerulisemate ja mahukamate testide automatiseerimiseks Seleniumil on loodud Selenium WebDriver, millel on palju eeliseid. Üks neist on mitme programmeerimiskeele toetamine, tänu millele võib testija valida testide loomiseks eelistatuima [10]. Sarnane olukord on ka brauserite toetamisega – teste võib käivitada kõigis tuntumates

veebibrauserites, nagu Google Chrome, Mozilla Firefox, Safari, Opera, Internet Explorer ja teised. Selenium WebDriveriga saab testida ainult veebipõhiseid rakendusi, mis selle analüüsi raames on plussiks. Testide kiiruse tagab otsesuhtlemine brauseritega ilma vahendajateta. Joonisel 3 on esitatud Selenium WebDriveri arhitektuur, mis demonstreerib programmiliidese otsesuhtlemist brauseritega [11]. Arhitektuur koosneb kolmest põhikomponendist: programmeerimiskeele sidumisest, Selenium WebDriver API-st ja brauserite draiveritest.



Joonis 3. Selenium WebDriveri arhitektuur [11].

Lisaks teistele Selenium WebDriveri võimalustele testida saab mitte ainult lokaalsel masinal. See on võimalik ka kaugmasinal Selenium Serveri kaudu [12]. Testide tulemuste jälgimiseks ja analüüsimiseks võib kasutada kuvatõmmiseid, mida teeb Selenium WebDriver testide salvestamise käigus [13]. See on päris mugav viis, aga selle kõrval võib kasutada ka teisi raamistikke, mis võimaldavad testitulemuste põhjalikumaid raporteid. Kõige tõhusamad selleks on JUnit ja selle analoog TestNG.

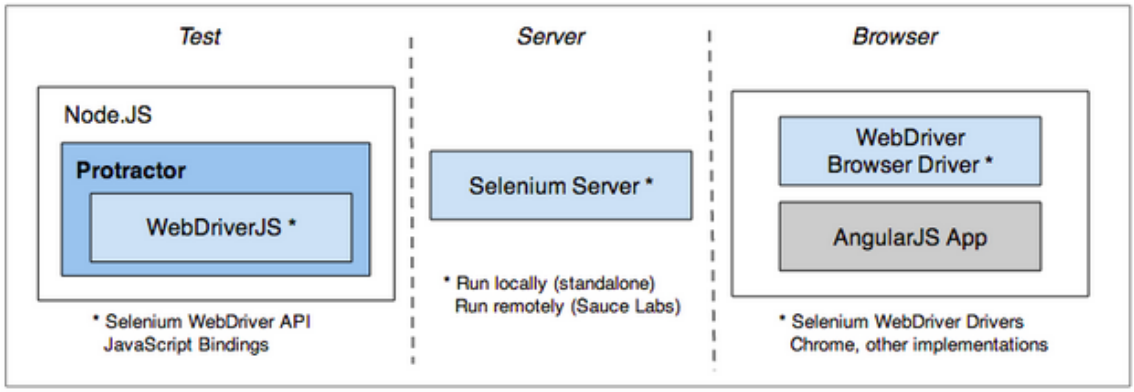
Seleniumi teste on võimalik käivitada mitmel viisil. Kui esmatähtis on testide kiirus, siis on mõistlik need käivitada *headless* režiimis. See meetod põhineb sellel, et brauseris ei laadita üles kasutajaliidese komponente, mis ennetab resursside liigset raiskamist. Kui tegemist on näiteks veebibrauseris surfamisega, siis kasutajaliidese elemente ei saa vahele jätta ja kasutusele tuleb nähtava kasutajaliidese režiim, sest selle komponentide kaudu teostatakse testitavaid tegevusi [14].

Kuigi Selenium WebDriver on üsna edukas lahendus veebirakenduste automatiseerimiseks, on tal ka puudusi. Raskusi võib tekitada näiteks see, et sellel tööriistal ei ole enda arenduskeskkonda, mis tähendab, et Selenium WebDriveriga töötamiseks tuleb kasutada teist keskkonda, näiteks Eclipse või Intellij IDEA. Samuti, kuna Selenium on avatud lähtekoodiga, ei taga see usaldusväärset tehnilist tuge [15]. Kui Selenium tarkvaral ilmneb mingi rike, mõjutab tehnilise toe puudumine paratamatult testitava rakenduse tööd.

Suuremate projektide testimiseks ei oma Selenium IDE piisavalt testimisvõimalusi ja laia funktsionaalsust ning sobib rohkem testide prototüüpide loomiseks. Selenium WebDriver seevastu on juba mitu aastat turu üks parimaid ja levinumaid tööriistu [16]. Selle funktsionaalsed võimalused sobivad vaadeldava tarkvara UI testimise automatiseerimiseks palju rohkem, nii et edaspidi vahendite võrdlemisel mõeldakse Seleniumi nime all Selenium WebDriverit.

3.2 Protractor

Protractor on raamistik Angular'i ja AngularJS'i põhiliste rakenduste E2E (*End-To-End*) testimiseks. See on üks parimaid raamistikke, kui rääkida Angular'i põhilistest rakendustest, sest ta toetab kogu spetsiifikat, mis on suunatud just sellele platvormile. Protractor töötab Selenium WebDriveri peal ja põhimõtteliselt laiendab selle võimalusi. Selline ülesehitus nõuab testijalt Protractoriga töötamiseks üldist arusaama Seleniumist. Testimise põhimõte on kasutaja tegevuste imiteerimine ja tänu WebDriveri kasutamisele toimib ka Protractor vastastikku brauseritega nende draiverite kaudu ehk otseselt [17]. Joonisel 4 on esitatud Protractori arhitektuur, mis koosneb kolmest põhiosast. Esimene osa on Test, mille kujundab testija kirjutatud testikood, kasutades Protractorit koos Selenium WebDriveriga. Server on vahendaja kirjutatud testi ja brauseri vahel. Selle ülesanne on edastada testis määratud käsud brauseri draiverile ja pärast nende täitmist tagastada nende tulemused. Kolmanda osa moodustavadki brauserite draiverid [18].



Joonis 4. Protractori arhitektuur [18].

Kuna peaaegu iga programmeerimiskeel või platvorm kasutab erinevaid meetodeid elementide identifitseerimiseks, võib testimise käigus olla raske saada kätte õiget elementi. Seepärast toetab Protractor just Angular'i spetsiifiliste lokaatorite kasutamist, mis kiirendab ja lihtsustab elementide äratundmist. Samuti täidab Protractor iga järgmist testsammu poolikuid ülesandeid ootamata, mis vabastab liigsete funktsioonide kirjutamisest ja seejärel ei ole ka kood üle kuhjatud [17]. See omadus vähendab ka testide käivitusaega. Kiiremaks ja produktiivsemaks testimiseks on Protractoris võimalus testida paralleelselt mitmes brauseris ja mitmes ühe brauseri eksemplaris, mida võib pidada suureks eeliseks.

Protractor, nagu Seleniumgi on API, mis tähendab, et see ei ole testimiskeskkond ja testide kirjutamiseks saab valida eelistatuima keskkonda. Testimisraamistikuks on vaikimisi valitud Jasmine raamistik [19]. Jasmine testimisraamistik on avaliku lähtekoodiga vahend, mis põhineb BDD (*Behavior-Driven Development*) meetodil ja on mõeldud JavaScripti koodi testimiseks [20]. See eeldab vähemalt Java programmeerimiskeele tundmist, sest ilma koodita selle vahendi abil testida ei saa. Protractori dokumentatsioon eeldab ka, et testija ei ole algaja, sest info on üsna kasin.

Raporteerimisvõimalus on realiseeritud vastavate sisseehitatud pakettide kaudu, mida vajadusel tuleb alla laadida. Raport sisaldab põhinfo testitulemuste kohta: vea kuvatõmmiseid, käivitusaega, edukate testide kirjeldust ning ebaõnnestunud teste ja informatsiooni võimalike ohtude kohta. Kõik on väga kergesti loetav ja ei vaja analüüsimiseks programmeerimisoskust.

3.3 TestComplete

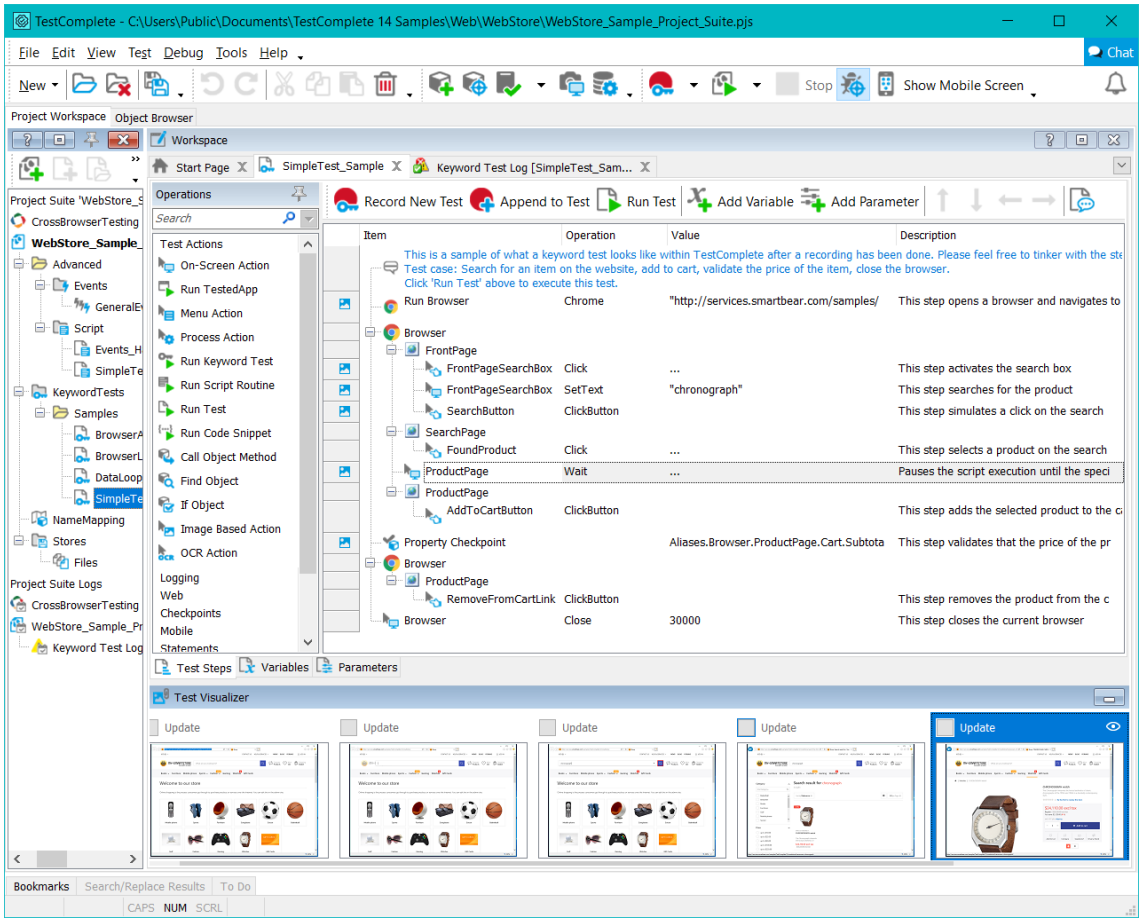
TestComplete kujutab endast võimsat UI testimise automatiseerimise tööriista, mis võimaldab kiiret ja lihtsat testide kirjutamist erinevate platvormide jaoks [21]. See vahend omab laia võimaluste kogumit ja funktsionaalsust, mille ülesanne on tagada stabiilne ja kvaliteetne testide haldamine.

Aastate jooksul on TestComplete näidanud ennast usaldusväärse ja tõhusa tööriistana, mida tõendab suur hulk positiivseid tagasisidesid ning kõrged positsioonid automatiseerimisvahendite erinevates edetabelites. TestComplete oli esimene, kes pakus objektide visuaalseks tuvastamiseks innovatiivse lahenduse, mis põhineb tehisintellektil. Praktiliselt see toimub automaatselt, kui mingi UI tekstiobjekt ei ole leitav lokaatori abil. Sellisel juhul tuvastab AI (*Artificial Intelligence*) selle OCR (*Optical Character Recognition*) meetodil, olenemata akna suurusest, resolutsioonist, kirjatüübist või keelest [22]. See uuendus kõrvaldab olulisel määral ebaedukad teststsenaariumid ja suurendab testide tõhusust. Selle tööriista veel üks, mitte vähem oluline eelis on DDT (*Data-Driven Testing*) toetamine. DDT võimaldab lihtsamaid ja vähem koormatud teste. Iga andmete muudatuse korral ei pea muutma testikoodi ja kõike uuesti käivitama, mis tavaliselt võtab testijatel palju aega. Veel väärtuslikumaks teeb selle meetodi see, et andmed võivad olla tõmmatud võõrastest allikatest, nagu Excel arvutustabelid, CSV (Comma-Separated Values) failid või andmebaasi tabelid [23]. Testimine vahenditega, kus selline võimalus puudub, tekitab testijale sageli palju raskusi andmete kättesaamisel. Eeliseks on ka see, et on võimalik veebipõhine testimine kõikides tuntuimates brauserites ja vajadusel on olemas ka mobiil- ja töölaua rakenduste testimise võimalus, nii et testitavate platvormide valik ei ole sellel tööriistal piiratud [6].

Raporteerimine TestComplete vahendis on küllaltki põhjalik. Raporti edukamaks analüüsimiseks võib testitulemused jaotada spetsiifiliste omaduste järgi, näidatakse testi läbiviimise aega, salvestatakse kõik logid, pildid ja videod ning kõike seda saab ka jagada projektitiimiga kiiremaks probleemide lahendamiseks [24]. Testitulemuste vaatamiseks ei pea tiimiliikme masinas isegi olema paigaldatud TestComplete tööriist. Plussiks meeskondlikuks testide arendamiseks on ka TFC (*Team Foundation Server*) ühenduse võimalus.

TestComplete arenduskeskkond toetab ka Seleniumi teste. Neid võib edukalt integreerida testimisprotsessi, ühendades Seleniumi ja TestComplete'i võimalused ja saada ühise testitulemuste raporti [25].

Esmamulje sellest vahendist on üsna positiivne, sest kasutajaliides tundub väga mugav, aga professionaalsemaks kasutamiseks tuleb seda muidugi põhjalikult uurida, sest TestComplete'i funktsionaalsus on väga suur. Joonisel 5 on tootja pakutav näidistest, mis lisab toote ostukorvi ning kontrollib hinna nähtavust ja korrektsust.



Joonis 5. Näidistest TestComplete'i testimiskeskkonnas.

Külastades TestComplete'i veebilehekülge, jääb mulje, et see on UI testide automatiseerimiseks ideaalne tööriist, sest kõik omadused on kirjeldatud nii hästi, et ei tule ühtegi mõtet, et seal on midagi puudu või et kasutamise käigus võib esineda mingi probleem. Kuid tööriista kasutamisel reaalses elus on suur tõenäosus avastada mitmeid puudusi. Oluliseks puuduseks võib pidada seda, et TestComplete on kättesaadav ainult Windows-operatsioonisüsteemil, aga mõned ettevõtted töötavad Linux- või Mac-operatsioonisüsteemil, mis tähendab, et neile ei ole TestComplete kättesaadav. Samuti kasutajate tagasisidet uurides võib välja tuua ka teisi puudusi. Näiteks paljud

märgivad, et testide käivitamine, kus taga on mahukas kood, võtab päris palju aega ja mõnikord võib programm üldse „hanguda“ või kokku joosta [26]. Selline olukord hävitab testimisprotsessi ja nõuab lisaresursse testide taaskäivitamiseks või tekkinud probleemi lahendamiseks.

Kuna TestComplete’is on päris palju võimalusi, tuleb selle tõhusamaks kasutamiseks testijaid koolitada. Sellist võimalust pakub ka tootja lisatasu eest, aga testija võib ka ise uurida testimiskeskkonda TestComplete’i dokumentatsiooni abil, mis on siiski palju keerulisem ning nõuab rohkem teadmisi ja aega.

Tuleb mainida ka, et see tarkvara on võrdlemisi kallis. Osta võib juba valmis komplekti, mis sisaldab kõigi platvormide testimisvõimalust ja ühte lisa, mis võimaldab piiramatut testimist mitmel platvormil paralleelselt, ning teisi lisasid võib soovi korral juurde osta. TestComplete’i tarkvara võib osta ka personaliseeritud konfiguratsiooniga, lisades ainult seda, mis on tarvis. Litsentsi tüüp on samuti valikuline ja on pakutud kaks varianti [27]:

- *Node-locked* litsents, mida võib kasutada ainult ühel masinal;
- *Floating* litsents, mida võib kasutada mitu kasutajat mitmel masinal paralleelselt.

TestComplete ühe ja sama konfiguratsiooniga *Floating* litsentsiga maksab tunduvalt rohkem kui *Node-locked* variandiga, aga kui testijate töö toimub eri masinatel, siis *Node-locked* variant ei sobi ja ettevõttel tuleb ohverdada eelarve.

3.4 Screenster

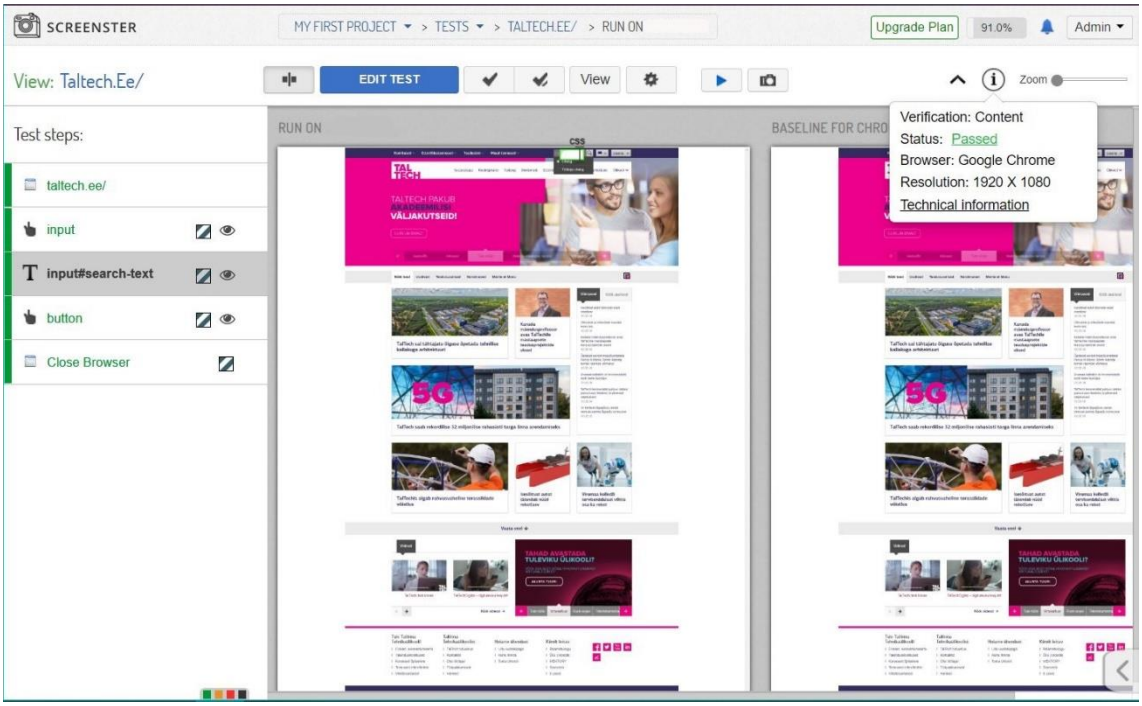
Screenster on pilvepõhine kasutajaliidese testimise platvorm, mis sarnaselt Seleniumil põhinevatele raamistikele imiteerib kasutaja tegevusi veebirakendustes või brauserites [6]. Vaatamata sellele, et samamoodi nagu Protractor, on Screenster ehitatud Seleniumi põhjal, võimaldab see testida ka ilma programmeerimisoskusteta, kuid laiem funktsionaalsus nõuab neid oskusi ja Seleniumi tundmist.

Screenster testib kogu kasutajaliidest ning toetab selleks võimalikult mugavaid viise ja funktsioone. Koodivabaks testimiseks on väga levinud *record-playback* meetod. Selles režiimis on Screenster väga tundlik dünaamiliste objektide suhtes. Testi käigus tehakse kuvatõmmiseid edasiseks võrdlemiseks ja kui mingi väike objekt, kas reklaam või

artikkel erineb esimese käivituse pildil olevast, annab Screenster hoiatuse. Testi käivitamiseks on samuti mitu viisi [28]:

- sisu tõestamine – kontrollib teksti vastavust;
- visuaalne võrdlemine – võrdleb kasutajaliidest pikslite tasandil;
- ainult navigatsioon – kontrollib ja käivitab ainult testitavaid tegevusi ilma lisaelementideta.

Testikiirus on väga sõltuv interneti kiirusest ja arvuti jõudlusest, nii et kui kasutatav internet või masin on ülekoormatud, võib kasutajaliides „hanguda“ ja käivitatavaid teste kinni hoida. Aga kuna Screenster on pilvepõhine, võib teste käivitada ja hoida ka pilves, kust nad on kergesti kättesaadavad edasiseks testimiseks. Joonisel 6 on näidatud Screensteri kasutajaliides, kus on näidiseks sooritatud test, mille ülesanne on kontrollida teksti sisestamist otsinguribasse ja otsingunupule vajutamist.



Joonis 6. Screensteri testimiskeskkond.

Screenter pakub oma kasutajatele väga põhjalikku dokumentatsiooni. Seal on esitatud ka hulk kasulikke näiteid, mis aitavad uuel kasutajal testimiskeskkonnaga tutvuda ja kohe testimist alusada. Probleemide korral võib pöörduda tugiteenuse poole, aga seda võimalust ei ole kõikide litsentside korral.

Litsents on tasuline, välja arvatud väga piiratud variant, mis on pigem nagu demo versioon. Tasuliste litsentside variante, millega võib tööle hakata, on kokku kolm ja nende võimalused suurenevad hinna järgi. Litsentsi tüübist sõltub käivitavate testide arv kuus pilves ja masinal, suurim kasutajate arv, maksimaalne käivitusaeg, tugiteenused, Jira toetus ja pideva testikäivituse võimalus [29].

Kahjuks on Screensteril veel omadusi, mida võib pidada miinusteks. Näiteks on testide käivitamine võimalik ainult Google Chrome, Internet Explorer ja Mozilla Firefox brauseris ja seetõttu veebirakendusi, mis on saadaval ka teistes brauserites Screensteriga testida ei saa.

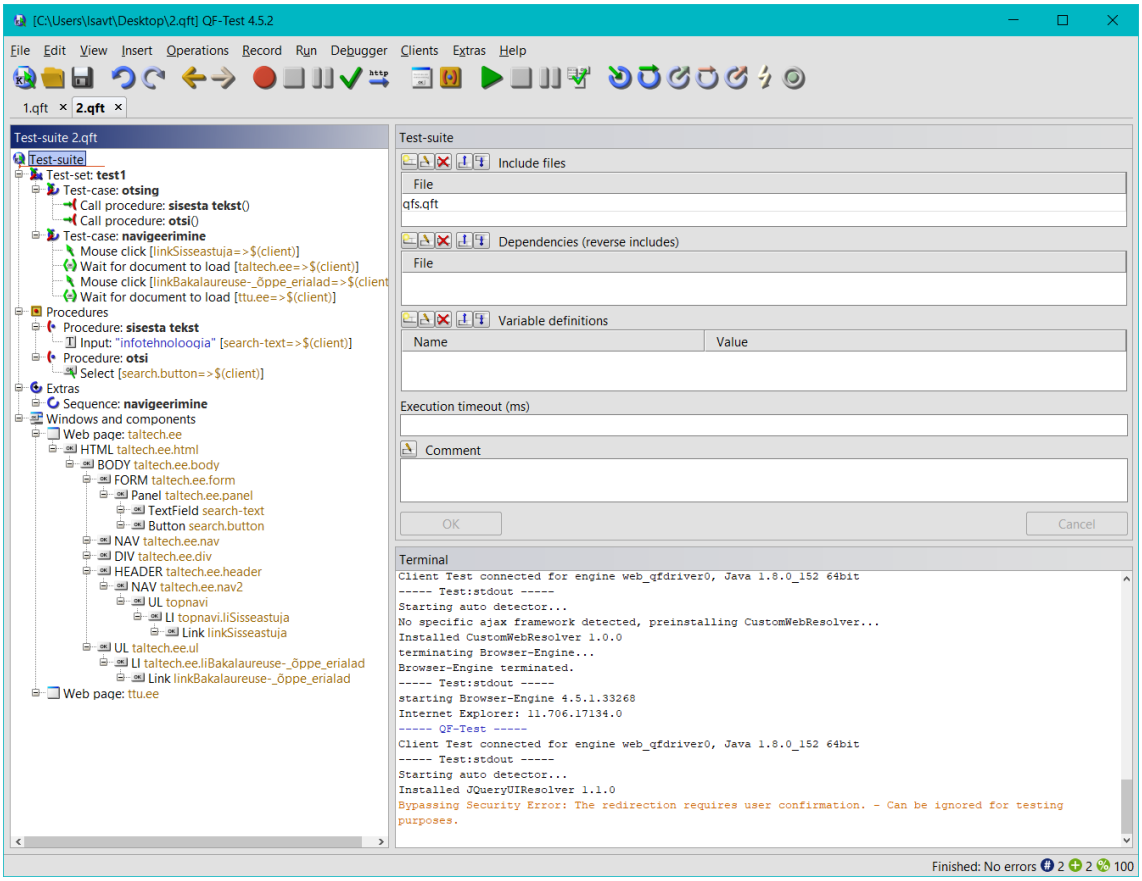
Raporteerimine Screensteri kaudu põhineb kuvatõmmiste tegemisel [30], mis on küll mugav, aga põhjalikumate projektide korral võib olla ebapiisav.

3.5 QF-Test

QF-Test on saksa GUI testimise tööriist, mis võimaldab automatiseerida teste nii programmeerimisoskusega kui ka ilma selleta [31]. QF-Test eristub teistest antud lõputöös vaadeldavatest vahenditest. See vahend ei ilmu otsingutulemuses esimesel lehel ega ole väga levinud tööriist, aga iduettevõte, mis on antud bakalaureusetöös vaadeldava rakenduse omanik, on QF-Testiga juba varem kokku puutunud. Seetõttu see on kaasatud analüüsitavate vahendite nimekirja.

QF-Test tööriist tegeleb Java- ja veebipõhiste rakenduste testimisega. Seda saab kasutada erinevates operatsioonisüsteemides, nagu Windows, Linux ja Mac, ning ka erinevates brauserites [32]. Üks oluline aspekt selle töö raames on ka Angular'i toetamine QF-Testi poolt, mis laseb ilma raskuseta testida nõutavat veebirakendust [31]. Nagu mainitud, võib automatiseeritavaid teste koostada ilma programmeerimisoskusteta. Testide koostamiseks on vaja lihtsalt valida vajalik protseduur loodud protseduuride alaosast ja tõmmata see vastava testijuhtumi alaosasse ning sama protseduuri võib kasutada mitmes testjuhtumis. Protseduure luuakse samuti ilma programmeerimiskeelteta üsna detailset pakutavat vormi täites. Kui käsitsi protseduuride lisamine on keeruline, võib testi loomiseks kasutada ka *record-playback* meetodit. Vajaduse korral võib integreerida ka Seleniumi teste [32].

Testimiskeskkond ei tundu keeruline, vaid hoopiski väga intuitiivne. Nagu on näha joonisel 7, on kõik põhitegevusteks vajalikud nupud ülemises ribas ja kui miski on arusaamatu, on paremklikiga saadaval nupp „*What's this?*", mis viitab dokumentatsioonile, kus testimiskeskkonna iga väli ja element on väga arusaadavalt ära seletatud. Testikomplektide ülesehitus on esitatud puustruktuurina, mis teeb vajalike failide vahel navigeerimise palju mugavamaks ja kiiremaks.

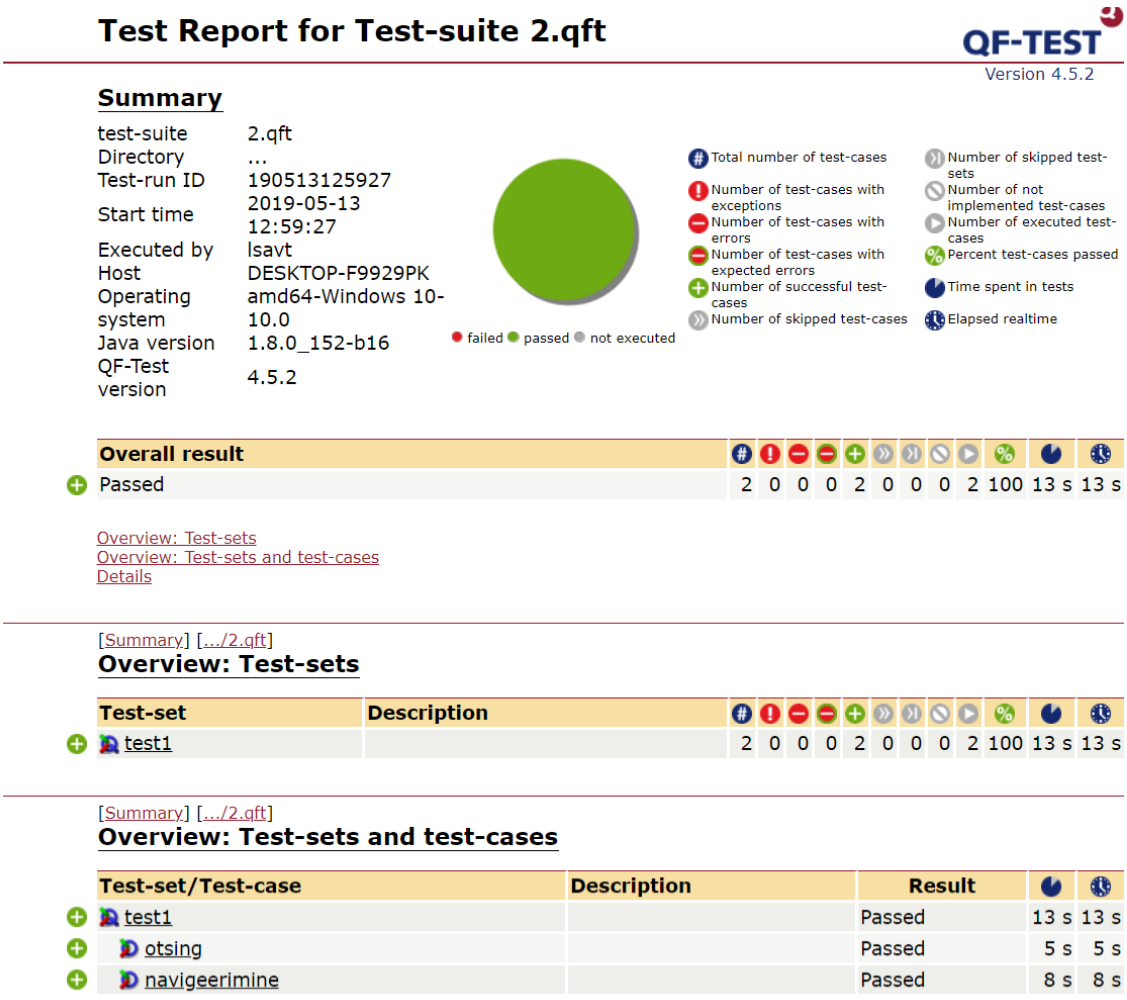


Joonis 7. QF-Testi testimiskeskkond.

Suureks plussiks, mis võib raha säästa, on ka see, et dokumentatsioon tervikuna on väga põhjalik ja kirjutatud väga mõistetavalt, nii et QF-Test tööriista omandamine iseõppimise teel on täiesti võimalik, kuid tuleb arvestada, et tuleb pühendada üsna palju aega testijate õppimisele. Näitlikustamiseks on kodulehel ka mitu ametlikku õppevideot.

Vaadeldavas testimiskeskkonnas on sisseehitatud raporteerimisvõimalus. QF-Test võimaldab testitulemusi läbi vaadata puukujuliselt, nagu on üles ehitatud ka testid ise. Logis on esitatud kõik täidetud sammud, protseduurid ja nende parameetrid, vea korral salvestatakse automaatselt ekraanipilt ja märgistatakse koht, kus ja miks see viga testi katkestas. Testitulemustel põhineva raporti võib salvestada HTML-, XML- või JUnit

formaadis, olenevalt vajadustest ja nõuetest, ning arvata sinna ainult need parameetrid, mida on analüüsimiseks vaja [33]. Joonisel 8 on osaliselt esitatud näidistesti (vt Joonis 7) raport HTML-formaadis. Salvestatud raportis on toodud üldistatud andmed testi käivituse ja tulemuste kohta ning vajaduse korral võib pöörduda kasutatud tähiste selgituste juurde lihtsama ja kiirema raporti lugemiseks mitte ainult testijatele, vaid ka ärianalüütikutele. Detailsemaid andmeid kahjuks raportis ei esitata.



Joonis 8. QF-Testis koostatud testiraport HTML formaadis.

QF-Testi ostmisel pakutakse erinevaid litsentse, millest võib valida ettevõttele sobivaima. Kõigepealt tuleb valida *Development* ja *Runtime* litsentside vahel. Nende erinevus on selles, et *Development* litsents sisaldab terve funktsionaalsuse kasutamist testikomplektide arendamiseks ja salvestamiseks, aga *Runtime* litsents ei võimalda testikomplekte salvestada [34]. QF-Testi ükski müügil olev litsents ei ole seotud vaid ühe konkreetse masina ja kasutajaga ning ei pea ostma kõike GUI tehnoloogiaid korraga ja

maksma ebavajalike asjade eest, vaid võib valida ühe tehnoloogia, millega testijad tegelikult töötama hakkavad, ja lisatasu eest võib vajaduse korral osta ka teisi [35].

4 Tööriista valimine

UI testimise automatiseerimise tööriista valimisel on ettevõtte eesmärk rahuldada enda kui terviku ja testijate vajadusi. Samuti see peab olema võimeline testima veebirakendust, mille testimiseks teda hakatakse kasutama. Selleks, et seda teostada, on vaja välja selgitada kõikide osapoolte vajadused ja seejuures oli abiks ka selles ettevõttes töötavate UI testijate küsitlus (Lisa 1). Kuna tegemist on iduettevõttega, on UI testijaid seal ainult kaks, aga nende arvamuste ja kogemuse põhjal saab koostada nõuded otsitavale vahendile, sest just nemad hakkavad selle vahendiga töötama.

Vahendi valimisel on väga olulised ka rakenduse omadused, sest igal rakendusel on eriomadused, mis kitsendavad töötamiseks sobivate tööriistade valikut. Iduettevõttel ei ole veel suurt finantsbaasi, aga toode on suunatud kiirele kasvule ja klientide hulga laienemisele, nii et testide hulk hakkab samuti kiiresti kasvama. Seepärast on automatiseerimine vaja rakendada võimalikult kiiresti ja ilma liigsete kuludeta. Oluline on mainida ka, et rakenduse arendamisel on kasutatud Angular 6 platvormi ja rakendus iseenesest on veebipõhine. Kogutud informatsiooni põhjal võib koostada nõuete loendi otsitava vahendi jaoks.

Rakenduse nõuded:

- kokkusobivus Angular'i pohiliste rakendustega;
- veebirakenduste testitavus.

Ettevõtte nõuded:

- võimalikult vähe kulusid UI testide automatiseerimiseks;
- valitud tarkvara kiire kasutuselevõtt;
- kasumi suurenemine tulevikus;
- rahulolevad kliendid.

Ettevõtte nõudeid täidetakse ainult sel juhul, kui ka testijate nõuded on enamasti täidetud. UI testijate küsitluse alusel on eraldatud nõuded automatiseerimisvahendile, mis on omakorda jaotatud funktsionaalseteks ja mittefunktsionaalseteks nõueteks. Rakenduse

nõuded kuuluvad funktsionaalsete nõuete hulka, aga ettevõtte nõudeid antud kontekstis ei jagata, sest ettevõte ei ole otsitava vahendi otsekasutaja. Funktsionaalsed nõuded kirjeldavad, mida peab otsitav tarkvara tegema, ja mittefunktsionaalsed nõuded kirjeldavad, kuidas neid funktsioone täitakse [36].

Testija nõuded:

- funktsionaalsed nõuded:
    - veebirakenduste testimine;
    - testitava rakenduse komponentide äratundmine;
    - testsammude salvestamine;
    - tekstiandmete kontroll;
    - suurte andmehulkade töötlemine;
    - testide täiustamine testide jooksmise ajal;
    - debugimine;
    - sisseehitatud raporteerimine.
- mittefunktsionaalsed nõuded:
    - mugav ja intuitiivne navigeerimine;
    - kiiresti omandatav;
    - testide „hangumise“ ja kokku jooksmise võimalus on nulli lähedane;
    - tugiteenused on kiiresti kättesaadavad iga päev kogu ööpäeva jooksul;
    - tööriist on saadaval mitmel operatsioonisüsteemil;
    - teste võib koostada ilma koodita;
    - testide hulga suurenemisel ei muutu käivitusaeg liiga kauaks;
    - teste võib käivitada mitmel masinal;
    - teste võib käivitada paralleelselt;
    - hästi loetavad raportid.

Tabelisse 1 on koondatud andmed, mis kirjeldavad kõikide asjaomaste poolte nõudeid otsitavale tööriistale. Tabeli eesmärk on kõrvutada tööriistade omadusi ja uurida, milline tööriist vastab ettevõtte nõuetele.

Tabel 1. Analüüsitavate vahendite omaduste võrdlus.

Nõuded	**Selenium**	**Protractor**	**TestComplete**	**Screenster**	**QF-Test**
Hind, €/litsents aastas	tasuta	tasuta	4000–7700€	500–800€	1000–2000€
Kokkusobivus Angular'i pohiliste rakendustega	+	+	+	+	+
Veebiraken-duste testimine	+	+	+	+	+
Komponentide äratundmine	koodiga, kasutades lokaatoreid	Angular'i spetsiifiline lokaatorite kasutamine	keerukas	väga mugav	väga mugav
Testsammude salvestatamine	kasutatava arendus-keskkonna kaudu	+	+	+	+
Tekstiandmete kontroll	+	+	AI tehnoloogiaga lisatasu eest	sisu tõestamise režiimis	+
Suurte andmehulkade töötlemine	+	+	jõudlus väheneb märgatavalt	jõudlus väheneb märgatavalt	võib tekkida vajadus manuaalselt segavate failide kogumi filtreeri-miseks
Testide täiustamine testide jooksmise ajal	-	+	+	-	-
Debugimine	kasutatava arendus-keskkonna kaudu	+	+	+	+
Mugav ja intuitiivne navigeerimine	-	-	+	+	+

Nõuded	Selenium	Protractor	TestComplete	Screenster	QF-Test
Kiiresti omandatav	-	-	-	+/-	+
Testide „hangumine“ või kokku jooksmine	harva	harva	aeg-ajalt võib juhtuda	aeg-ajalt võib juhtuda	uuemates versioo-nides väga harva
Tugiteenus	abi foorumitest	abi foorumitest ja korduma kippuvad küsimustest	kiire ja kvaliteetne	kallimate litsentsidega	kiire ja kvaliteetne
Toetavad operatsiooni-süsteemid	Windows, Linux, Solaris	kõik	Windows	kõik	Windows, Linux, Mac
Programmeeri-misoskuste vajadus	+	+	+/-	+/-	+/-
Testide kiirus	+	+	lühikesed testid on kiired	+	+
Testide käivitamine mitmel masinal	+	Selenium Serveri kaudu	kaugtöölaua ühendust kasutades	-	Daemon režiimi installi-misel
Sisseehitatud raporteerimine	-	+	+	nõrk	+

4.1 Vahendite võrdlus

Analüüsitavate vahendite omaduste koondtabelis on lihtne jälgida vastavust etteantud nõuetele (vt Tabel 1). Selgema arusaama kujundamiseks vaadeldakse iga nõuet eraldi, mis võimaldab leida kõige tugevama tööriista igas konkreetses aspektis.

- **Hind.** Kõige kallim variant on TestComplete ning Screenster koos QF-Testiga on palju madalamas hinnavehemikus. Ülejäänud kaks vahendit on tasuta, aga see ei tähenda, et nad oleksid selles kriteeriumis parimad, sest avaliku lähtekoodiga vahendid nõuavad palju lisakulusid.

- **Kokkusobivus Angular'i pohiliste rakendustega.** Seda nõuet täidavad kõik vaadeldavad vahendid.
- **Veebirakenduste testimine.** Kõik vaadeldavat vahendid omavad sellist võimalust.
- **Komponentide äratundmine.** Hästi saavad sellega hakkama QF-Test ja Screenster oma mugavuse pärast. Kuid Protractoril on siin rohkem eeliseid, sest seal komponentide äratundmine toimub Angular'il põhinevate rakenduste kõige mugavamal ja efektiivsemal viisil. Selenium ja TestComplete vahendites on see võimalus realiseeritud kõige kehvemini.
- **Testsammude salvestatamine.** Kõikides vahendites on see nõue täidetud, aga Seleniumis sõltub see kasutatavast arenduskeskkonnast.
- **Tekstiandmete kontroll.** Kõikides vahendites on see võimalus edukalt realiseeritud. TestComplete kasutab selleks kõige tõhusamat ja uuemat viisi, aga see on võimalik ainult kõige kallima litsentsi ostmisel.
- **Suurte andmehulkade töötlemine.** Koodipõhised vahendid Selenium ja Protractor töötavad mahuka andmehulgaga üsna edukalt. QF-Testi puhul võib tekkida olukord, kus on vaja manuaalset sekkumist, mida tahaks pigem vältida. TestComplete ja Screenster sellega edukalt hakkama ei saa.
- **Testide täiustamine testide jooksmise ajal.** Selline võimalus on realiseeritud ainult Protractoris ja TestComplete'is.
- **Debugimine.** Kõik vaadeldavad vahendid võimaldavad debugimist.
- **Mugav ja intuitiivne navigeerimine.** Selle all mõistetakse navigeerimist vahendi kasutajaliideses, mida omavad kõik vahendid peale Seleniumi ja Protractori, kuna need on raamistikud, mitte keskkonnad. Kõige mugavam tundub QF-Testi kasutajaliides (vt Joonis 7), sest nii testifailid kui ka vajalikud funktsioonid tööribal on struktureeritud ja kiiresti leitavad.
- **Kiiresti omandatav.** Seleniumi, Protractori ja osaliselt Screensteri omandamise teeb raskeks programmeerimisoskuse nõutus, sest nad nõuavad koodi kirjutamist. Protractori omandamist loetakse siin kontekstis samuti aeglaseks ja raskeks, kuna selle kasutamiseks on vaja omandada ka vajalikud programmeerimiskeeled. TestComplete'i keskkonnas on samuti võimalik koodi kirjutada, aga raskus põhineb selle laia funktsionaalsusel, mille omandamiseks tuleb kasutada palju resursse. QF-Test on kõige kiiremini ja kergemini omandatav.

- **Testide „hangumine“ või kokku jooksmine**. See omadus peab puuduma, et nõue oleks täidetud, ja kõige lähemal on sellele Selenium, Protractor ja uuemad QF-Testi versioonid. TestComplete’is ja Screensteris provotseerivad sellist olukorda suured testid ja andmehulgad.
- **Tugiteenus.** Tekkivate küsimuste ja probleemide korral aitab kõige paremini TestComplete’i, QF-Testi ja kallima litsentsiga Screensteri tugi. Kuna Selenium ja Protractor on avatud lähtekoodiga raamistikud, tuleb nende puhul lahendusi otsida pigem kasutajate foorumites.
- **Toetavad operatsioonisüsteemid.** Protractoril ja Screensteril selles küsimuses piiranguid ei ole. Selenium ja QF-Test toetavad mitte kõiki, vaid kõige levinumaid operatsioonisüsteeme. TestComplete on saadav ainult Windowsil.
- **Programmeerimisoskuste vajadus.** Seleniumis ja Protractoris testimine toimub ainult programmeerimiskeelte kasutamisega, ülejäänutes võib testida ka ilma programmeerimiseta.
- **Testide kiirus.** Kõik vahendid tagavad kiire testimise käivituse, aga TestComplete’is testide mahu suurenemisel nende käivituskiirus väheneb.
- **Testide käivitamine mitmel masinal.** Seleniumil see on kõige kergemini teostatav, kuna tal on juba sisseehitatud Selenium Server, aga teistel on vaja enne alla laadida vastavaid faile. Screensteril selline võimalus puudub.
- **Sisseehitatud raporteerimine.** Sisseehitatud raporteerimisvõimalus puudub ainult Seleniumis. Kõige põhjalikumad raportid on Protractoris, TestComplete’is ja QF-Testis.

Nüüd, kui on olemas üldine arusaam sellest, kuidas iga vahend iga konkreetset nõuet täidab, tuleb valida parim. Kahjuks ei täida ükski vahend kõiki etteantud nõudeid korraga, kuna neid on palju, aga samas ei ole kõik nõuded ka samaväärsed ehk mõni nõue on olulisem kui teine ja vastupidi. Esineb ka seda, et mingi tööriist väidetavalt omab sobivat omadust, mis tähendab, et teoreetiliselt on nõue täidetud, kuid tegelikult ei ole see täidetud piisaval määral või on omadus realiseeritud väga kehvasti.

Tabelis 2 on esitatud iga vahendi nõuetelevastavuse suhteline hinnang. Hinnangud kirjeldavad nõuete täitmist ja nende lihtsamaks tajumiseks on igale hinnangule omistatud numbriline väärtus:

- hinne „0“ – nõue ei ole täidetud ja vahendil puudub selline omadus;
- hinne „1“ – nõue on osaliselt täidetav ja omadus on kehv;
- hinne „2“ – nõue on täidetud, aga keskpäraselt;
- hinne „3“ – nõue on täielikult täidetud ja omadus töötab eeskujulikult.

Kõik nõuded on jagatud kolmeks sektsiooniks omaduste tähtsuse järgi vastavalt küsitlusest saadud informatsioonist nõuete olulisuse kohta. Punane sektsioon sisaldab kõige olulisemaid nõudeid ettevõtte testijate jaoks, kollane sektsioon sisaldab nõudeid, millede täitmine on keskmise tähtsusega ja roheline sektsioon sisaldab nõudeid, millede täitmine on soovituslik.

Tabel 2. Nõuete täidetuse hinnang.

Nõuded	**Selenium**	**Protractor**	**TestComplete**	**Screenster**	**QF-Test**
Madal hind	1	1	1	3	2
Kokkusobivus Angular’i põhiliste rakendustega	3	3	3	3	3
Veebirakenduste testimine	3	3	3	2	3
Komponentide äratundmine	2	3	1	2	2
Suurte andmehulkade töötlemine	3	3	1	1	2
Tekstiandmete kontroll	2	2	3	2	2
Debugimine	1	2	3	3	3
Programmeerimisoskuse mittevajalikkus	0	0	1	2	3
Testide kiirus	3	3	1	2	3
Keskmine hinne	2	2,2	1,9	2,2	2,6
Kiiresti omandatav	1	1	1	2	3
Testsammude salvestatamine	1	2	2	2	2
Testide „hangumise“ ja kokku jooksmise puudumine	2	2	1	1	2
Testide täiustamine testide jooksmise ajal	0	2	2	0	0
Tugiteenus	1	1	3	2	3
Toetab mitu operatsioonisüsteemi	2	3	1	3	2

Nõuded	Selenium	Protractor	TestComplete	Screenster	QF-Test
Sisseehitatud raporteerimine	0	2	2	1	2
Keskmine hinne	1	1,9	1,7	1,6	2
Testide käivitamine mitmel masinal	3	3	2	0	2
Mugav ja intuitiivne navigeerimine	0	0	2	2	3
Keskmine hinne	1,5	1,5	2	1	2,5

Iga sektsiooni kohta igaüks tööriist omab keskmist hinnet, mis on saadud kõikide hinnangute alusel. See keskmine hinne näitab konkreetse sektsiooni nõuete täitmistaset. Kõige olulisemaid nõudeid täideti peaaegu sama hästi, kuid järgmistes sektsioonides hinnete erinevus on juba rohkem nähtav. Ükski vahenditest ei täida kõiki etteantud nõudeid täielikult, aga ikkagi saab eristada üht üldliidrit. Hindamise tulemusel on saadud, et kõige edukaim tööriist on QF-Test.

5 Kokkuvõte

Antud bakalaureusetöö põhieesmärk oli leida sobiv tööriist iduettevõttele, mis võimaldaks tõhusat ja suhteliselt odavat UI testimise automatiseerimist. Kuna iduettevõte areneb väga kiiresti ja kliendibaas suureneb, peab tööriist olema piisavalt usaldusväärne, et tagada häireteta ja kasutajasõbralik rakendus.

Automatiseeritud UI testidele üleminek nõuab kaalukaid investeeringuid ja seetõttu oli ettevõtte üks olulisemaid nõudeid litsentsi madal hind. Ka testijad samuti esitasid otsitavale tööriistale oma nõuded, mis puudutasid selle funktsionaalsust. Arvestades testitava rakenduse spetsiifikat koostati ka rakenduse nõudeid. Koondatud nõuete põhjal analüüsiti kõige levinumaid UI testide automatiseerimise vahendeid. Analüüs algas iga vahendi ülevaate ja proovimisega ning seejärel kõrvutati vaadeldavate vahendite omadused nõuetega. Analüüsi ja järgneva võrdluse põhjal valiti QF-Test, mis rahuldas enamikku etteantud nõuetest. Puudu jäi ainult üks nõe, testide täiustamise võimalus testide jooksmise ajal, aga see nõue ei olnud esmatähtis. Tulemus on suurepärane, arvestades ka seda, et sellise nõuete hulga korral on peaaegu võimatu leida täielikult rahuldavat tööriista.

Üllatuslik oli see, et QF-Testi konkurentideks selles analüüsis olid tuntumad ja kallimad tööriistad, mida on peetud väga edukateks ja tõhusateks. Seetõttu võib teha järelduse, et tuntus ja kõrge hind ei ole kvaliteedi näitajad.

Kokkuvõtteks võib kindlasti öelda, et eesmärk on saavutatud ja iduettevõtte nõuded saavad leitud lahenduse abil parimal viisil täidetud.

Kasutatud kirjandus

[1] Meador, D. (2018). Graphical User Interface (GUI). [WWW] https://www.tutorialspoint.com/graphical-user-interface-gui (04.05.2019)

[2] Karam, L. M. (2017). A Guide to GUI Testing. [WWW] https://dzone.com/articles/a-guide-to-gui-testing (04.04.2019)

[3] Тестирование UI (пользовательского интерфейса). [WWW] https://woxapp.com/ru/our-blog/testing-the-ui-user-interface/ (17.04.2019)

[4] The test automation pyramid. [WWW] https://www.ontestautomation.com/the-test-automation-pyramid/ (06.04.2019)

[5] McPeak, A. (2018). Record and Replay Testing vs. Scripting: Which is Right for You? [WWW] https://crossbrowsertesting.com/blog/test-automation/record-and-replay-testing/ (06.04.2019)

[6] 9 automation testing tools to consider for web applications in 2018. [WWW] https://screenster.io/automation-testing-tools-in-2018/#Automated_functional_testing_of_the_GUI_with_Screenster (11.05.2019)

[7] Что такое Selenium? [WWW] https://selenium2.ru/ (08.04.2019)

[8] How To Export Tests from Selenium IDE. [WWW] http://elementalselenium.com/tips/6-export-from-selenium-ide (08.04.2019)

[9] Selenium IDE. [WWW] https://selenium2.ru/docs/selenium-ide.html (08.04.2019)

[10] Selenium WebDriver. [WWW] https://www.seleniumhq.org/projects/webdriver/ (10.04.2019)

[11] Architectire of Selenium WebDriver. [WWW] http://makeseleniumeasy.com/2017/03/03/architecture-of-selenium-webdriver/ (10.04.2019)

[12] What is Selenium? [WWW] https://www.seleniumhq.org/ (10.04.2019)

[13] Gochenour, P. (2018). Getting Started with Selenium for Automated Website Testing. [WWW] https://wiki.saucelabs.com/display/DOCS/Getting+Started+with+Selenium+for+Automated+Website+Testing (12.04.2019)

[14] UI testing considerations. [WWW] https://docs.microsoft.com/en-us/azure/devops/pipelines/test/ui-testing-considerations?view=azure-devops&tabs=mstest (12.04.2019)

[15] Advantages and Disadvantages of Selenium. [WWW] https://www.gcreddy.com/2016/05/advantages-and-disadvantages-of-selenium.html (12.04.2019)

[16] Top 15 Selenium Alternatives in 2019. [WWW] https://www.guru99.com/selenium-alternatives.html (13.04.2019)

[17] Protractor. [WWW] https://www.protractortest.org/#/ (15.04.2019)

[18] Chourasia, A., Patel, S. (2018). End-to-End(E2E) testing for an AngularJS with Protractor and gulp using IBM Cloud DevOps Delivery pipeline. [WWW]

https://developer.ibm.com/in/2018/08/16/end-end-testing-angularjs-protractor-gulp-using-ibm-cloud-devops-delivery-pipeline/ (15.04.2019)

[19] Tutorial. [WWW] https://www.protractortest.org/#/tutorial (16.04.2019)

[20] Jasmine. [WWW] https://jasmine.github.io/ (16.04.2019)

[21] The Easiest-to-Use Automated UI Testing Toolwith Artificial Intelligence. [WWW] https://smartbear.com/product/testcomplete/overview/ (18.04.2019)

[22] Artificial Intelligence for Faster and Smarter Software Testing - North America Session. [Video] https://www.youtube.com/watch?v=6MLb7FqR188 (18.04.2019)

[23] Automated Test Data Generation and Data-Driven Testing. [WWW] https://smartbear.com/product/testcomplete/features/easily-create-data-driven-tests/ (20.04.2019)

[24] Automated Test Reporting & Analysis in TestComplete. [WWW] https://smartbear.com/product/testcomplete/features/test-reporting-analysis/ (12.05.2019)

[25] Selenium Test Scripts with TestComplete. [WWW] https://smartbear.com/product/testcomplete/features/selenium-testcomplete/ (20.04.2019)

[26] TestComplete Reviews. [WWW] https://www.g2.com/products/testcomplete/reviews (22.04.2019)

[27] TestComplete. [WWW] https://smartbear.com/product/testcomplete/pricing/ (11.05.2019)

[28] Running Tests: Verification Modes. [WWW] https://screenster.io/documentation/running-tests-verification-modes/ (23.04.2019)

[29] Pricing. [WWW] https://screenster.io/pricing/ (23.04.2019)

[30] Screenster Reviews. [WWW] https://www.saasworthy.com/product/screenster (12.05.2019)

[31] Comparison Protractor vs. QF-Test for Angular. [WWW] https://www.qfs.de/en/qf-test/web-testing/test-automation-protractor-vs-qf-test.html (26.04.2019)

[32] Automated GUI Test tool QF-Test. [WWW] https://www.qfs.de/en.html (05.05.2019)

[33] Comparison Selenium vs. QF-Test. [WWW] https://www.qfs.de/en/qf-test/web-testing/test-automation-selenium-vs-qf-test.html (05.05.2019)

[34] QF-Test Product Description and Licensing Models. [WWW] https://www.qfs.de/fileadmin/Webdata/pdf/en/product-description-qf-test.pdf (13.05.2019)

[35] QF-Test Pricing. [WWW] https://www.qfs.de/en/qf-test/pricing.html (05.05.2019)

[36] Tepandi, J. (2018). Tarkvara protsessid, kvaliteet ja standardid. 14-15. [WWW] https://tepandi.ee/tks-loeng.pdf (10.05.2019)

Lisa 1 – Küsimustik

1. Kuidas te hinnate oma oskusi UI testide automatiseerimises?
   a) Professionaal
   b) Keskmine või algaja
   c) On läbitud vastav koolitus
   d) Kuulsin sellest, aga ei ole kunagi kokkupuutunud
   e) Ültse ei ole kogemust
2. Kas teile on oluline, et UI testide koostamine oleks võimalik ka ilma programmeerimiskeeleta?
   a) Jah
   b) Ei
3. Mis on teie arvates peamised asjad/funktsioonid, mis peavad olema UI testide automatiseerimise tööristal? (Loendage olulisuse kahanevas järjekorras)
4. Mis teile segab UI testimise automatiseerimise tööristades? Kirjeldage peamised puudused, mis võivad takistada teie tööd. (NB! Ainult, kui te olete varem kokkupuutunud UI testimise automatiseerimisega)