Opinnäytetyö (AMK)

Tieto- ja viestintätekniikka

2018

Teemu Lehtonen

AJANVARAUSPALVELUN AUTOMAATIOTESTAUS

– Suunnittelu ja toteutus

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tieto- ja viestintätekniikka

2018 | 15 sivua

Teemu Lehtonen

AJANVARAUSPALVELUN AUTOMAATIOTESTAUS

Automaatiotestaus tarkoittaa sitä, että käsin tehtävät testit ohjelmoidaan suorettavan automaattisesti. Tämän seurauksena testit voidaan suorittaa nopeammin ja kustannustehokkaammin. Opinnäytetyön tavoite oli luoda automaatiotestausohjelma toimeksiantajayrityksen tuottamaan ajanvarauspalveluun. Ohjelman tarkoituksena oli helpottaa palvelun testausta.

Työn tekemisessä käytettäviä ohjelmia olivat Robot Framework, Selenium ja Python. Aineistoina toimi testauksesta ja automaatiotestauksesta kirjoitettu kirjallisuus sekä toimeksiantajayrityksen sisäinen tieto ja osaaminen. Testiohjelmaa tehtiin yrityksen ja erehdyksen kautta.

Lopputuloksena oli toimiva automaatiotestausohjelma, joka otettiin käyttöön päivittäisiin testiajoihin. Keskeinen tieto oli vastaus kysymykseen, milloin testaus kannattaa automatisoida. Automaattitestausta ei kannata esimerkiksi tehdä, jos ohjelma jota testataan, muuttuu jatkuvasti.

Työn tuloksena ajanvarauspalvelun testaus nopeutui huomattavasti, kun tulevaisuudessa yksinkertaista testausta ei tarvitse suorittaa käsin. Ohjelman kehittämistä jatketaan opinnäytetyön jälkeenkin lisäämällä enemmän testitapauksia ja koodin parantamisella.

ASIASANAT:

automaatiotestaus, testiautomaatio, testaus

BACHELOR´S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information and Communications Technology

2018 | 15 pages

Teemu Lehtonen

APPOINTMENT SERVICE'S AUTOMATION TESTING

Automation testing means that manual tests are programmed to run automatically. As a result, tests can be performed faster and more cost-effectively. The aim of this Bachelor's thesis was to create an automation testing program for the appointment service provided by the commissioning company. The purpose of the program was to ease the testing of the service.

The software used to make this testing program were Robot Framework, Selenium and Python. The material used to create this program was literature written from testing and automated testing. In addition, the knowledge of the commissioning company was used. The test program was created through trial and error.

The result was a functioning automation testing program that was introduced for daily test runs. The key knowledge was the answer to the question of when testing should be automated. For example, it is not worth testing for automated testing if the program being tested is constantly changing.

Carried out in this thesis, the testing of appointment service will be greatly accelerated when in the future there is no need for manual testing. The development of the program will continue after the thesis, by adding more test cases and code enhancement.

KEYWORDS:

automation testing, test automation, testing

SISÄLTÖ

KUVAT

1 JOHDANTO

Opinnäytetyön toimeksianto oli suunnitella ja toteuttaa automaatiotestausohjelma ajanvaraus- ja palveluohjain -nimiseen palveluun. Toimeksianto tuli palvelun tarjoavalta yritykseltä, ja työn ideana oli, että helpotetaan palvelun testausta ja saadaan tästä kustannustehokkaampaa. Automaatiotestausohjelman tekemisessä käytettiin apuna siitä kirjoitettua kirjallisuutta ja toimeksiantajayrityksen sisäistä tietoa. Lopputuloksena odotetaan toimivaa automaatiotestausohjelmaa.

Opinnäytetyön alussa käsitellään lyhyesti testausta yleisellä tasolla. Tämän jälkeen perehdytään automaatiotestaukseen. Työssä käsitellään muutama automaatiotestauksessa käytettyä ohjelmaa, joita myös käytettiin työtä tehdessä.

Tämän jälkeen opinnäytetyössä esitellään automaatiotestausohjelman suunnittelua, jossa käytetään esimerkkinä työssä tehtyä ohjelmaa. Työssä esitellään muutamia ongelmia, joita ilmeni ohjelman teon aikana. Kerrotaan myös lyhyesti, miten automaatiotestaus ohjelmaa toteutetaan. Tämän jälkeen esitellään osaa opinnäytetyössä tehdyn ohjelman rakennetta. Lopuksi kerrotaan ohjelman jatkokehityssuunnitelma.

2 TESTAUS

Testaus tarkoittaa lyhyesti siitä, että tuotetta tai ohjelmistoa testataan, jotta sen käyttö olisi sujuvampaa ja vältyttäisiin virheiltä ja häiriöiltä, kun tuote otetaan tuotantokäyttöön.

Jyväskylän yliopiston Systemaattiset Oppimisratkaisut -hankkeessa (2011–2015) kuvataan ohjelmistotestausta seuraavasti: ”IEEE:n (1990) määritelmän mukaan ohjelmistotestaus on aktiviteetti, jossa määritellyillä ehdoilla suoritetun ohjelman tai sen osan tuloksia tarkastellaan tai tallennetaan, ja sen toimintaa arvioidaan.”

2.1 Testauksen vaiheet

Testaus voidaan jakaa kolmeen vaiheeseen, yksikkö-, integrointi ja järjestelmätestaus. Lähteenä vaiheiden määrittelemisessä käytetään Helsingin yliopiston Ohjelmistotuotantoprojektin julkaisun testaussuunnitelmaa OhtuTie (2004).

Yksikkötestaus

Yksikkötestaus tarkoittaa, että testataan yhtä ohjelman osaa. Tässä varmistetaan, että yksikkö toimii suunnitellulla tavalla. Jokainen ohjelmoija suorittaa yksikkötestauksen itse ohjelmoimalleen komponentille (OhtuTie 2004, 2).

Integrointitestaus

”Integrointitestauksessa testataan erillisten komponenttien välistä vuorovaikutusta sekä kommunikointia ja toiminnallisuutta niiden välillä. Integroitavien yksiköiden yksikkötestaus täytyy olla hyväksyttävästi suoritettu ennen tämän vaiheen aloittamista”. (OhtuTie 2004, 3.)

Järjestelmätestaus

Järjestelmätestauksessa testataan koko ohjelmistoa. Tässä vaiheessa pyritään testaamaan ohjelmistoa sille määriteltyjen vaatimusten suhteen (OhtuTie 2004, 3).

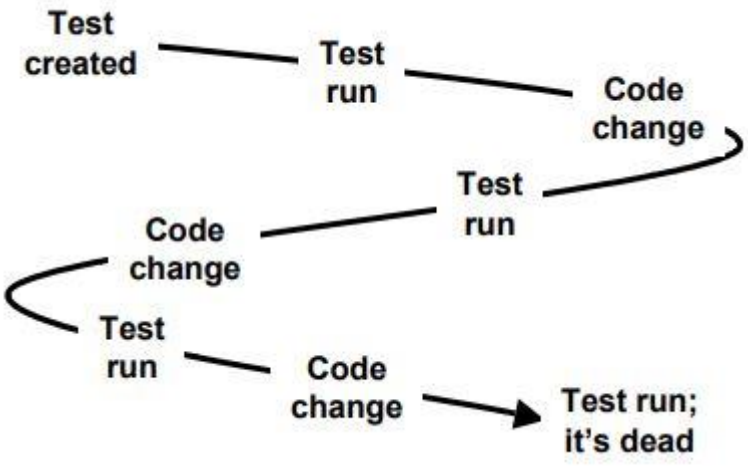
2.2 Automaatiotestaus

Automaatiotestaus on nimensä mukaisesti sitä, että testaus automatisoidaan. Tämä tarkoittaa sitä, että käsin tehtävät testit ohjelmoidaan suorettavan automaattisesti. Tämän seurauksena testit voidaan suorittaa nopeammin ja kustannustehokkaammin. Kustannustehokkuus tulee juuri nopeudesta ja siitä että tällöin käytettävä aika voidaan suunnata tehokkaammin merkityksellisempään työhön. Testejä pystytään myös suorittamaan ilman, että niiden etenemistä tarvitsee valvoa esimerkiksi yöllä.

2.2.1 Testien automatisoinnin hyödyt

Automaatiotestaus ei kuitenkaan aina ole parempi ratkaisu kuin manuaalisesti tehty testaus. Brian Marick (2000) kirjoittaa artikkelissaan, että aina pitäisi analysoida, onko halvempaa tehdä testi automaatiotestauksella vai manuaalisesti. Hän kertoo esimerkissään, että testiä ei ole järkevää automatisoida, jos testataan ohjelmaa, jonka koodi muuttuu paljon. Tämä johtuu siitä, että silloin tarvitsee myös automaatiotestausohjelman koodia päivittää ja korjata.

Kuvassa 1 on lyhyesti kuvattu automaatiotestin elinkaari. Kun mietitään, kannattaako testi automatisoida vai ei, tulee arvioida, montako koodimuutosta testiohjelma kestää. Jos se ei kestä montaa koodimuutosta, automaattitestiohjelman pitää olla erittäin pätevä löytämään virheitä (Brian Marcik 2000).



Kuva 1. Brian Marickin käyttämä esimerkki testin elinkaaresta (Brian Marcik 2000).

2.2.2 Automaatiotestauksessa käytettäviä työkaluja

Automaatiotestaukseen on markkinoilla olemassa monia eri työkaluja. Työkalu valitaan testattavan ohjelman mukaan.

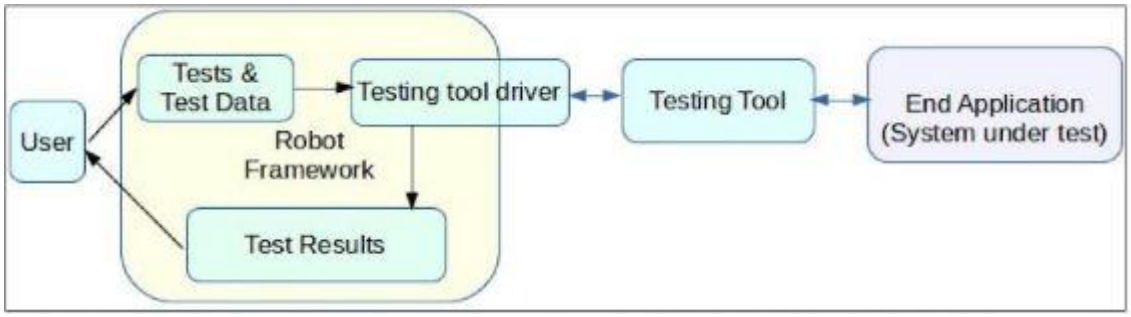
Selenium on avoimeen lähdekoodiin perustuva automaatiotestaussovellus. Selenium tukee myös useimpia nettiselaimia (Salunke 2014).

Seleniumia käytetään nettiselaimessa tehtävään automaatiotestaukseen. Seleniumin yksi suurimmista vahvuuksista on, että se on erittäin joustava automaatiotyökalu (Selenium 2018).

Robot Framework on geneerinen automaatiotestaukseen käytetty viitekehys (engl. framework) hyväksymistestaukseen ja hyväksymistestivetoiseen ohjelmistokehitykseen (Robot Framework 2018).

Viimeinen laatikko kuvassa 2 (End Application) on ohjelma, johon suoritetaan testit. Testing Tool on ohjelma, joka suorittaa itse testin. Robot Framework -laatikossa olevat yksiköt ovat osia, jotka toimivat Robot Frameworkin sisällä. Lopuksi vasemmalla on käyttäjä, joka on laittanut testin aluilleen ja jolle palautuu lopuksi testitulokset (Bisht 2013).

Kuvan 2 tarkoituksena on havainnollistaa Robot Frameworkin toimintaa. Kuvassa nähdään, että Robot Framework sisältää monta osaa, jotka keskustelevat keskenään.



Kuva 2. Robot Framework –ekosysteemi (Bisht 2013).

Atom on avoimeen lähdekoodiin perustuva tekstieditori, jonka on kehittänyt GitHub (Atom 2018).

Python on avoimeen lähdekoodiin perustuvat ohjelmointikieli (Python 2018).

3 AUTOMAATIOTESTAUSOHJELMAN SUUNNITTELU

3.1 Työn aloitus

Opinnäytetyön toimeksiantona on toteuttaa automaatiotestausohjelma ajanvaraus palvelun testiympäristöön. Palveluohjain on selaimessa käytettävä ohjelma, jolla kansalainen voi varata itselleen ajan terveydenhuollon palveluun. Testiympäristöön on luotu kuvitteellinen palvelu testausta varten. Testiympäristö eroaa tuotantoympäristöstä siten, että testiympäristöön tuodaan uudet ominaisuudet testattavaksi. Kun uudet ominaisuudet on todettu toimiviksi, voidaan ominaisuudet siirtää tuotantoympäristöön.

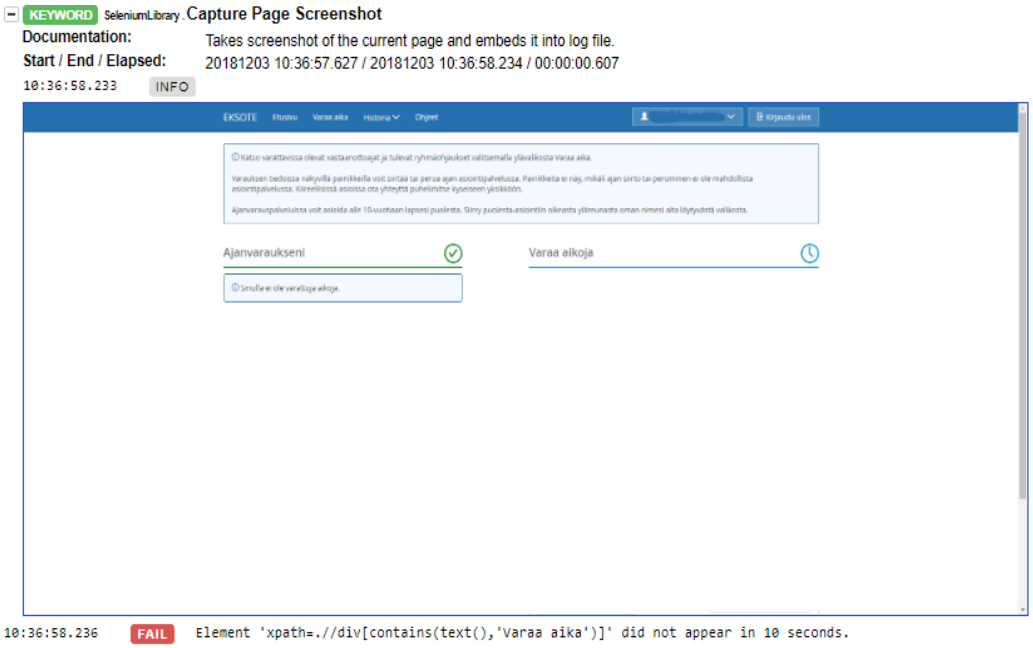
Työn alussa kuvataan testitapaus, joka halutaan testiautomatisoida. Testitapaus on kansalaisen ajanvarauksen tekeminen. Testi jaetaan kolmeen osaan. Ensimmäisessä vaiheessa sisään kirjaudutaan kansalaisen puolelle ja varataan aika. Toisessa vaiheessa siirretään varausaika toiseen ajankohtaan. Kolmannessa vaiheessa suoritetaan ajanvarauksen peruminen. Työ aloitettiin tekemällä Excel-taulukko, ja siihen hahmoteltiin vaiheet kohta kohdalta. Käytiin testitapauksia manuaalisesti läpi ja laitettiin taulukkoon ylös kaikki huomioitavat asiat.

Seuraavaksi asennettiin kaikki tarvittavat työkalut, joita tultaisiin tarvitsemaan työtä tehdessä. Nämä olivat Robot Framework, Selenium, Atom, Python ja Chrome-selaimen uusimmat ajurit. Työn toimeksiantajalla oli valmiina tehty automaatiotestauksen taustat, joten päästiin suoraan tekemään ohjelmaa.

3.2 Selaimessa tapahtuva automaatiotestaus

Automaatiotestausta tehdessä on hyvä huomioida paljon yksityiskohtia. Selaimessa tehtävässä automaatiotestauksessa on suositeltavaa tarkistaa mahdollisimman paljon elementtejä. Näillä tarkistuksilla varmistutaan, että automaatiotestausohjelma paikannin (engl. locator) on juuri siinä kohtaa kuin halutaan. Esimerkiksi kun ohjelmalla siirrytään selaimella uudelle sivulle, selaimessa tehdään tarkistus, että uudelta sivulta löytyy tietty elementti. Jos elementti löytyy, voidaan testiä jatkaa tai jos elementtiä ei löydy, testi loppuu. Jos elementtiä ei löydy, tämä tarkoittaa, että ohjelma ei ole oikeassa paikassa tai sivulta löytyy bugi.

Robot Frameworkissa on ominaisuus, että kun tapahtuu virhe se ottaa kuvankaappauksen kohdasta, jossa virhe tapahtui. Tämä helpottaa, kun selvitetään, että miksi virhe tuli. Kuvankaappauksesta näkee suoraan, onko ohjelma oikealla sivulla selaimessa ja mahdollisesti suoraan mikä on ollut virheenä. Kuvasta 3 näkee, että ohjelma on oikealla sivulla ja, että tarkastettava elementtikin löytyy sivulta. Virhe on tapahtunut elementin määrittelemisessä. Elementin pitäisi olla kuvan 3 tapauksessa xpath=.//a[contains(text),'Varaa aika']. Tämä selviää, kun elementin tarkastaa sivun lähdekoodista. Ohjelmaa tehdessä näitä tuli paljon.



Kuva 3. Robot Frameworkin ottama kuvankaappaus.

Työtä tehdessä huomattiin, että jotain virheitä ei tullut, kun suoritettiin vain tietty osa koko sarjasta. Nämä virheet nousivat esiin vasta, kun ajettiin kaikki testitapaukset peräkkäin. Esimerkiksi, kun otettiin ajanvarausta tehdessä päivämäärä, kellonaika ja asiointipiste talteen muuttujiin, jotta pystytään varmistumaan, että ajanvarauksesta ei muutu nämä tiedot, kun palataan takaisin palveluohjaimen etusivulle ajanvarauksen teon jälkeen. Virhe nousi esille testin toisessa osassa, jossa tehdyn ajanvarauksen aikaa siirrettiin uuteen ajankohtaan. Ohjelma otti uuden päivämäärän, kellon ajan ja asiointipisteen muuttujiin, mutta ei jostain syystä osannut käyttää uusia arvoja vertailussa.

Varmaa syytä tähän virheeseen ei löydetty. Kun virhettä katsottiin lokista, näkyi, että ohjelma oli tallentanut uudet arvot muuttujiin. Virhe saatiin ohitettua ottamalla siirretyn ajanvarauksen päivämäärän, kellonajan ja asiointipisteen arvot uusiin muuttujiin.

3.3 Automaatiotestausohjelman toteuttaminen

Seleniumissa käytetään elementtien paikantamiseen XPath-ilmaisua. XPathia käyttäen kuvan 4 maalatulla rivillä merkitty elementti kohdennetaan näin: Wait Until Page Contains Element xpath=.//a[@data-pid='23']. Jos elementistä löytyy id, sitä voi hyödyntää varmistamaan, että ohjelman paikannin on varmasti oikeassa elementissä. Id on elementeissä aina yksilöllinen. Esimerkkikomennolla ohjelma on kohdennettu tuohon elementtiin. Jos halutaan varmistua, että tuo elementti löytyy sivulta, niin tämä komento riittää. Komentoa voidaan myös jatkaa muun muassa komennolla Click Element, jolloin ohjelma painaa tätä kyseistä elementtiä. Click Element -komentoa ei voi käyttää ennen kuin ohjelma on kohdennettu elementtiin.

Kuva 4. Osa googlen etusivun lähdekoodia (Google 2018).

3.4 Automaatiotestausohjelman rakenne

Kuvassa 5 on työssä tehdyn automaatiotestausohjelman ensimmäisen osan rakenne. Siinä näkyy avainsanat (engl. keywords), joita käytettiin. Avainsanojen sisältö on siirretty eri sivulle atomissa selkeyden takia.

```
Avph kansalaisen ajanvaraus: Kansalainen tekee ja tarkastaa ajanvarauksen 1/3
    [Tags]    QA  avph
    Ohjaa suoraan kansalaisen AVPH etusivulle
    Tarkasta kansalaisen alue AVPH sivulla
    Siirry avph Avoimiin palveluihin
    Valitse palvelu: Automaatiotestaus

    Tarkistetaan oleminen ajanvarauksen valinta sivulla
    Valitaan asiointipiste
    Valitaan vastaanottaja
    Varmistetaan kalenteri komponentti näkyy
    Tarkistus onko kalenterikuukaudella vapaita aikoja
    Valitse varauksen aika päivälle

    Tarkista Varauksen vahvistaminen popup
    Otetaan päivämäärä, kello ja asiointipiste talteen muuttujiin
    Kirjoitetaan lisätiedot ja lisätään aikaleima
    Vahvista varaus

    Varmista oleminen Ajanvaraus paatapahtuma sivulla
    Etsi teksti Ajanvaraukset varattu onnistuneesti
    Tarkasta ajanvarauksen tiedot
    Katsotaan ajanvarauksen valmistautumisohjeet
    Palataan takaisin avph etusivulle

    Tarkasta että tehdyn ajanvarauksen tila Odottaa käyntiä
    Katso yksityikohtaiset varauksen tiedot
    Tarkistetaan että ollaan varauksen tiedot sivulla
    Tarkasta ajanvarauksen tiedot

    Kirjaudu ulos AVPH palvelusta
```

Kuva 5. Automaatiotestausohjelman rakenne.

Kuvassa 6 näkyy yhden avainsanan sisältö. Kun ohjelman rakentaa avainsanoihin voidaan samaa avainsanaa hyödyntää uudelleen. Tällöin ei tarvitse kirjoittaa samaa asiaa moneen kertaan.

```
Siirry avph Avoimiin palveluihin
    Wait Until Page Contains Element   xpath=.//a[contains(text(),'Varaa aika')]    timeout=${SET_TIMEOUT}
    Click Element                      xpath=.//a[contains(text(),'Varaa aika')]
    Sleep   2
```

Kuva 6. Avainsanan sisältö.

4 TULOKSET JA JOHTOPÄÄTÖKSET

Opinnäytetyön tavoite oli luoda automaatiotestausohjelma ajanvarauspalveluun, ja kartoittaa tietoa automaatiotestauksesta. Tuloksena oli toimiva automaatiotestausohjelma, jota käytetään palveluohjaimen päivittäisissä testiajoissa. Opinnäytetyössä tehty taustatutkimus laajensi myös tietoa automaatiotestauksesta.

Opinnäytetyöstä toimeksiantajayritys sai nopeutettua palveluohjaimen testausta huomattavasti, kun jatkossa yksinkertaista testausta ei tarvitse enää suorittaa käsin. Palveluohjaimen automaatiotestauksen tekemistä jatketaan siirtymällä uusiin ja monimutkaisempiin testitapauksiin. Tullaan myös kehittämään ja parantamaan jo tehtyä ohjelmaa palveluohjaimen päivitysten yhteydessä. Ohjelmaa tullaan parantamaan siten, että käydään koodia läpi myöhemmin uudestaan, ja arvioidaan voiko jonkun osan tehdä paremmin ja kestävämmällä ratkaisulla. Ideaali tilanne on, että lopuksi käsin tehtävä testaus kohdistuu vain uusien ominaisuuksien testaamiseen.

LÄHTEET

Atom 2018. Atom Flight Manual. Viitattu 27.9.2018

https://atom.io/

Systeemiset Oppimisratkaisut (SysTech), 2011-2015, Jyväskylän yliopisto. Viitattu 23.9.2018

http://smarteducation.jyu.fi/projektit/systech

Ohjelmistotuotantoprojekti OhtuTie 2004. Helsingin yliopisto. Viitattu 23.9.2018

https://www.cs.helsinki.fi/group/otie/

Brian Marick. 2000. Reliable Software Technologies. Viitattu 24.9.2018

https://www.stickyminds.com/sites/default/files/article/file/2014/When%20Should%20a%20Test%20Be%20Automated.pdf

Selenium 2018. Documentation. Viitattu 24.9.2018

https://www.seleniumhq.org/

Robot Framework 2018. Introduction. Viitattu 27.9.2018

http://robotframework.org/

Sumit Bisht. 2013. Robot Framwork Test Automation. Viitattu 27.9.2018

https://books.google.fi/books?id=_RO8AQAAQBAJ&printsec=frontcover&hl=fi&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false

Sagar Salunke. 2014. Selenium Web Driver in Java. Viitattu 24.9.2018

https://books.google.fi/books?id=wAqQBgAAQBAJ&printsec=frontcover&dq=selenium+webdriver&hl=fi&sa=X&ved=0ahUKEwiwzpSe89PdAhVLposKHRkgCVAQ6AEIPTAD#v=onepage&q=selenium%20webdriver&f=false

Google 2018. Etusivu. Viitattu 29.10.2018

https://google.com/

Python 2018. Documentation. Viitattu 29.10.2018

https://www.python.org/