Tampereen ammattikorkeakoulu

Testiautomaatioskriptien tuottaminen QAutoflow-työkalulla

Arttu Knuutila

OPINNÄYTETYÖ
Toukokuu 2019

Tieto- ja viestintätekniikka
Ohjelmistotekniikka

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tieto- ja viestintätekniikka
Ohjelmistotekniikka

KNUUTILA ARTTU
Testiautomaatioskriptien tuottaminen QAutoflow-työkalulla

Opinnäytetyö 31 sivua
Toukokuu 2019

Opinnäytetyössä tutkittiin QAutoflow-työkalun soveltuvuutta testiautomaatioskriptien tekemiseen. QAutoflow tallentaa käyttäjän selaimessa tekemät toiminnot ja generoi niiden pohjalta Robot Framework-testiautomaatioskriptin. Tätä menetelmää verrattiin manuaaliseen testitapausten tekemiseen QAutorobot-työkalun avulla. Tarkastelussa oli menetelmien erot ajankäytössä sekä testiskriptien laadussa. Menetelmiä vertailtiin tekemällä testitapauksia kahdelle eri web-sivustolle.

Työssä käytiin läpi tärkeimmät työvaiheet testiautomaatioskriptin luomiseksi kummallakin käytettävistä menetelmistä. Työvaiheiden esittelyn lisäksi tutustuttiin lyhyesti itse testitapauksen rakenteeseen sekä käytettyihin teknologioihin. Lopuksi arvioitiin menetelmien vahvuudet sekä havaitut ongelmatapaukset.

Tuloksien perusteella todettiin QAutoflow:n soveltuvan testiautomaatioskriptien tuottamiseen, mutta sivustosta ja sen testattavuudesta riippuen manuaalisen työn osuus voi vaihdella. Työkalu mahdollisti paikoin merkittävänkin säästön ajankäytön osalta, mutta se ei sovellu ainoaksi tavaksi tuottaa testitapauksia testattaessa hyvin dynaamisia ja monimutkaisia sivustoja. Parhaaksi menetelmäksi osoittautui sekä QAutoflow:n että QAutorobotin yhteiskäyttö, jolloin suoraviivaiset tapaukset voidaan käsitellä nopeasti QAutoflow:lla, mutta haasteellisemmat osiot testitapauksissa kannattaa toteuttaa manuaalisella lähestymistavalla.

Asiasanat: testiautomaatio, web-testaus, robot framework, selenium

ABSTRACT

Tampere University of Applied Sciences
ICT Engineering
Software Engineering

KNUUTILA ARTTU
Creating test automation scripts with QAutoflow tool

Bachelor's thesis 31 pages
May 2019

The purpose of this study was to investigate QAutoflow tool feasibility for creating test automation scripts. QAutoflow records user actions of browser and generates a Robot Framework test automation script based on saved data. This method was compared with manual test case creation with QAutorobot. Differences between methods were studied in terms of time consumption and quality of generated test scripts. Methods were compared by creating test cases for two different web pages.

Relevant steps regarding the creation of test automation scripts were demonstrated with both methods. The structure of the test case was introduced as well as the technologies used in test creation. Finally advantages and difficulties with each test creation method were evaluated.

The results showed that QAutoflow can be used to create test automation scripts under certain conditions. The tool allowed saving considerable amount of time but it shouldn’t be used as the only solution to create test automation scripts especially when web page under testing is more complicated. Combining both methods seemed to be the best solution. Simple test cases can be created quickly with QAutoflow whereas more challenging scenarios could be handled by manual test creation with QAutorobot.

Key words: test automation, web testing, robot framework, selenium

SISÄLLYS

LYHENTEET JA TERMIT

framework	Ohjelmistokehys, kokoelma komponentteja nopeuttamaan uusien ohjelmien toteuttamista
XPath	XML Path Language
id	identifier, web-elementin uniikki tunniste
IoT	Internet of Things

1 JOHDANTO

Web-sovellusten automaatiotestien tekeminen on perinteisesti vaatinut melko paljon manuaalista työtä sekä ohjelmoinnin osaamista. Opinnäytetyössä tutkitaan, voidaanko automaatiotestejä tuottaa onnistuneesti nauhoittamalla käyttäjän syötteitä verkkosivulle. Tietojen keräämiseen ja testien generointiin käytetään QAutomate Oy:n työkaluja QAutoflow sekä QAutorobot. QAutoflow-työkalulla tuotettuja automaatioskriptejä verrataan manuaalisesti tehtyihin automaatiotestiskripteihin toiminnan, laadun sekä ajankäytön osalta.

Opinnäytetyössä tutustutaan web-käyttöliittymien testaamisen keskeisiin käsitteisiin ja teknologioihin. Lisäksi työssä esitellään automaatioskriptien tuottamisessa käytettävät työkalut sekä käytettävät työmenetelmät. Lopuksi esitellään saavutetut lopputulokset ja pohditaan uuden menetelmän soveltuvuutta käytäntöön.

2 AUTOMAATIOTESTAUS

Ohjelmiston automaatiotestauksella tarkoitetaan jonkin ohjelman toiminnan testaamista testaukseen suunnitelluilla ohjelmistoilla. Nämä työkalut suorittavat testit sekä vertaavat saatuja tuloksia oletettuihin arvoihin.

Testauksen automatisointi kannattaa, mikäli samat testauksen vaiheet toistuvat useasti. Manuaalinen testaaminen on myös usein varsin työlästä ja aikaa vievää. Lisäksi jotkut toiminnot saattavat olla vaikeita testata manuaalisesti. Ohjelmallisella testaamisella varmistetaan myös testiympäristön sekä itse testien säilyminen samanlaisena suorituskerrasta toiseen (Software Testing Help).

2.1 Regressiotestaus

Regressiotestauksella pyritään varmistamaan, että ohjelmiston muuttuessa mikään jo olemassa olevista toiminnoista ei rikkoutunut muutoksessa. Regressiotestauksen taajuus voi vaihdella projektista toiseen. Testejä voidaan suorittaa jokaisen koodimuutoksen jälkeen tai esimerkiksi päivän päätteeksi. Muutosten yhteydessä myös testit jäävät helposti vanhentuneiksi, jolloin vaaditaan päivityksiä testeihin. Testien korjaamiseen kuluvaa aikaa voidaan minimoida järkevällä suunnittelulla sekä itse testien että testattavan ohjelmiston osalta (Test IO).

2.2 Käyttöliittymätestaus

Graafinen käyttöliittymä on sovelluksen osa, joka näkyy käyttäjälle. Tässä työssä keskitytään selaimen web-käyttöliittymien testaamiseen. Käyttöliittymätestaus painottuu web-elementtien, esimerkiksi painikkeet ja tekstikentät, toiminnan testaamiseen sekä käyttäjälle kussakin tilanteessa esitetyn datan oikeellisuuteen. Lisäksi on tärkeää varmistaa ohjelman toiminta erilaisissa ympäristöissä (Software Testing Class).

Yhdistelmiä eri selaimista, selainversioista, käyttöjärjestelmistä sekä päätelaitteista erikokoisilla näytöillä on lukematon määrä. Automatisoidulla käyttöliittymien

regressiotestauksella voidaan kattaa tehokkaasti halutut edellä mainituista yhdistelmistä.

2.3 Jatkuva integraatio

Nimensä mukaisesti jatkuvan integraation avulla pyritään validoimaan ja testaamaan muutokset ohjelmassa pienissä osissa sitä mukaa kun muutoksia tehdään. Mahdolliset virheet havaitaan ajoissa ja niiden korjaaminen on helpompaa. Tällöin vältetään ongelmat, joita lähes aina esiintyy yritettäessä integroida isoa komponenttia toiseen komponenttiin (Cloudbees CodeShip).

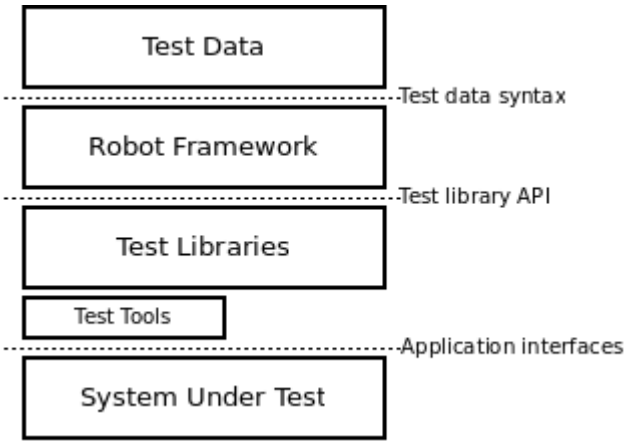
Jatkuva integraatio toteutetaan yleisimmin jonkun automaatiopalvelimen avulla. Koodimuutoksen jälkeen palvelin noutaa ohjelman lähdekoodin versionhallinnasta ja suorittaa valitut testit. Testitulokset voidaan ilmoittaa kehittäjille esimerkiksi sähköpostilla ja mahdollisiin virheisiin voidaan puuttua välittömästi.

3 TEKNOLOGIAT

Työssä käytetyt testaustyökalut QAutoflow ja QAutorobot käyttävät testien suorittamiseen avoimen lähdekoodin teknologioita Robot Framework ja Selenium.

3.1 Robot Framework-testiautomaatio framework

Robot Framework on avoimen lähdekoodin testiautomaatio framework, jota käytetään etenkin hyväksymistestaukseen ja yleisesti testiautomaatioon. Robot Framework käyttää avainsana - pohjaista syntaksia, jota on helppo lukea ja ymmärtää ilman vankkaa ohjelmointiosaamista. Ensimmäinen versio kehitettiin Nokia Networks:lla ja nykyisen kehityksestä vastaa Robot Framework Foundation. Robot Framework on yleisesti käytössä ja etenkin Suomessa voidaan puhua de facto työkalusta testiautomaation osa-alueella. Robot Framework on toteutettu Pythonilla ja sitä voidaan laajentaa Java– ja Python– kirjastoilla (Robot Framework).



KUVIO 1. Korkean tason arkkitehtuuri

Kuvion (kuvio 1) mukaisesti Robot Framework:n rakenne on modulaarinen. Lisäksi se pyrkii olemaan alusta- ja sovellusriippumaton. Tämä mahdollistaa kuhunkin tarpeeseen parhaiten soveltuvan konfiguraation luomisen lisäämällä tarvittavia kirjastoja sekä testidataa. Robot Framework soveltuukin hyvin monenlaisiin käyttökohteisiin aina käyttöliittymätestauksesta alemman tason rajapintojen testaamiseen.

Yksi tärkeimmistä ominaisuuksista on selkeä testitulosten raportointi. Robot Framework luo testiajon päätteeksi selkeät raporttitiedostot sekä HTML- että XML-muodossa. HTML-raportti havainnollistaa selkeästi, missä kohtaa mahdollinen virhe tapahtui. XML-muotoista raporttia käytetään enimmäkseen erillisten työkalujen, esimerkiksi automaatiopalvelin Jenkins:n, kanssa.

3.2 Selenium 2-framework

Selenium 2, monesti käytetään myös nimeä Selenium WebDriver, on työkalu selainpohjaisten testien automatisointiin. Selenium WebDriver mahdollistaa sekä itse selaimen hallinnan että vuorovaikutuksen web-sivujen kanssa. WebDriver tukee yleisimpiä selaimia, joista tässä työssä on käytössä Google Chrome. Myös tuettuja ohjelmointikieliä on useita, joista työssä käytetään Pythonia.

Seleniumin avulla voidaan suorittaa ohjelmallisesti selaimen hallintaan liittyviä toimintoja, esimerkiksi käynnistäminen, sulkeminen tai uuden välilehden avaaminen. Tärkein ominaisuus liittyy ohjelmalliseen vuorovaikutukseen web-sivujen kanssa. Koodilla voidaan etsiä haluttuja web-elementtejä ja kohdistaa näihin elementteihin halutut toiminnot. Toiminto voi olla vaikka painikkeen painaminen tai tekstin syöttäminen tekstikenttään. Usein on tärkeätä varmistaa, että tehty toimenpide tuotti halutun lopputuloksen. Tällöin voidaan tarkastella jonkin elementin arvoa toimenpiteen jälkeen ja verrata saatua arvoa oletettuun arvoon (SeleniumHQ).

Selenium WebDriver pyrkii realistiseen toimintojen suorittamiseen. Komennot pyritään suorittamaan kuten oikea käyttäjä suorittaisi kyseisen toiminnon. Mikäli esimerkiksi jokin tekstikenttä on poistettu käytöstä, ei myöskään WebDriver syötä siihen arvoa. Kyseinen toiminto olisi ohjelmallisesti mahdollinen, mutta suoritus on estetty realistisemman testauksen takia.

Tapoja, joilla Selenium löytää elementin, kutsutaan lokaattoreiksi. Ne voidaan jaotella sivuston rakenteesta riippuviksi sekä siitä riippumattomiksi. Rakenteesta riippuva lokaattori on esimerkiksi XPath, kun taas id ja link text ovat rakenteesta

riippumattomia. Testauksessa tulisi käyttää mahdollisuuksien mukaan rakenteesta riippumattomia lokaattoreita, koska tällöin itse testi sietää paremmin muutoksia hajoamatta (ProTech).

4 TYÖVÄLINEET

Testiautomaatioskriptien toteuttamiseen käytettiin kahta eri menetelmää QAutoflow– ja QAutorobot-työkaluja hyödyntäen. Oleellinen osa valmista automaatioskriptiä ovat QAutoLibrary-kirjaston tarjoamat funktiot, jotka tarjoavat monipuolisia toimintoja työskentelyä helpottamaan.

4.1 QAutoflow-työkalu

QAutoflow on työkalu web-käyttöliittymätestien luomiseen ja visualisointiin. Työkalu tallentaa käyttäjän selaimella tekemät toiminnot. Kerätty data tallentuu tiedostoon, jonka perusteella voidaan generoida toimiva Robot Framework testitapaus. Työkalun tavoitteena on nopeuttaa testitapausten luomista vähentämällä manuaalista työtä ja helpottaa testaukseen osallistumalla minimoimalla tarvittava ohjelmointi- ja testausosaaminen (QAutoflow).

QAutoflow rakentuu Google Chrome-selainlaajennuksesta sekä Pythonilla toteutetusta backend:stä. JavaScriptillä toteutettu selainlaajennus kerää käyttäjän selaimessa tekemiä toimintoja ja pyytää kunkin toiminnon jälkeen käyttäjää dokumentoimaan kyseisen askeleen. Näin saadaan kerättyä kuvaavat nimet funktioille myöhempää skriptien generointia varten. Nauhoituksen lopuksi kerätyt ja dokumentoidut askeleet tallennetaan tiedostoon. Tämän tiedoston pohjalta voidaan generoida Robot Framework-testitapaus käyttämällä QAutorobot-työkalua.

Työkalu mahdollistaa lisäksi Robot Framework-avainsanojen suorittamisen osana testivaiheiden nauhoittamista. Näiden avainsanojen avulla voidaan toteuttaa monimutkaisempia toimintoja tai säästää aikaa suorittamalla sarja aikaisempia askeleita. Usein toistuva vaihe, esimerkiksi sisäänkirjautuminen, voidaan hoitaa yhdellä avainsanalla, jolloin itse testitapauksen nauhoitus päästään aloittamaan tarvitsematta suorittaa samaa vaihetta aina uudelleen.

4.2 QAutorobot-työkalu

QAutorobot on wxPython-pohjainen sovellus web-käyttöliittymätestien tekemiseen ja ylläpitoon. Sovellus tarjoaa graafisen käyttöliittymän selainnäkymien mallintamiseen ja haluttujen elementtien paikallistamiseen. QAutoflow:lla nauhoitettu data voidaan muuntaa QAutorobot:lla Robot Framework testitapaukseksi. QAutorobot toistaa testin vaiheet tiedostoon tallennetun datan perusteella ja muodostaa automaattisesti tarvittavat Robot Framework-avainsanat sekä itse testitapauksen (QAutorobot).

Tässä työssä QAutorobotia käytetään QAutoflow:lla muodostettujen testitapausten generoinnin lisäksi referenssinä manuaalisten testiskriptien luomiseen.

4.3 QAutoLibrary-testauskirjasto

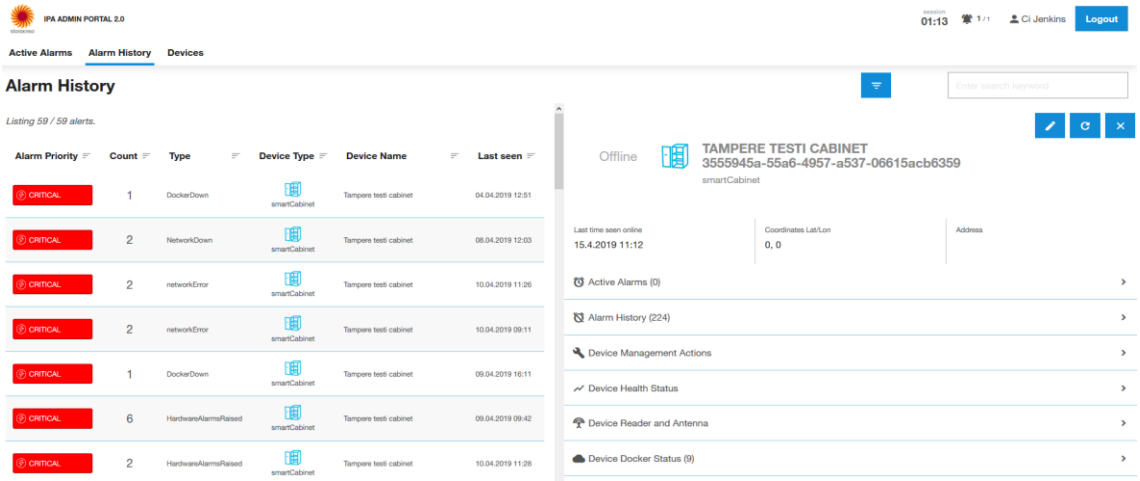
QAutoLibrary on avoimen lähdekoodin Robot Framework-kirjasto. Pythonilla toteutettu kirjasto laajentaa ja monipuolistaa Seleniumin tarjoamaa rajapintaa. QAutorobotin generoimat testiskriptit käyttävät QAutoLibrary:n funktioita. Suurimmat erot pelkkään Seleniumiin verrattuna ovat QAutorobotin tukema sivumallipohjainen avainsanojen jaottelu, mahdollisuus web-elementtien käsittelyyn Python-objekteina, tuki testisuorituksen videoimiselle sekä parannellut avainsanat. Avainsanat perustuvat Seleniumin omiin avainsanoihin, mutta QAutoLibraryn toteutuksessa avainsanat sisältävät synkronointivaiheita elementin läsnäolon varmistamiseksi sekä tukevat web-elementtien lisäksi QAutoElement-nimistä Python-objektia. Käyttäjä voi myös halutessaan kirjoittaa omia Python-funktioitaan laajentamaan olemassa olevaa kirjastoa (QAutoLibrary).

5 TYÖN KULKU JA YMPÄRISTÖ

Työmenetelmiä vertailtaessa toteutettiin samat testitapaukset molemmilla menetelmillä. Testitapauksiksi valittiin suoraviivaisia ja relevantteja käyttötilanteita kahden eri sivuston osalta.

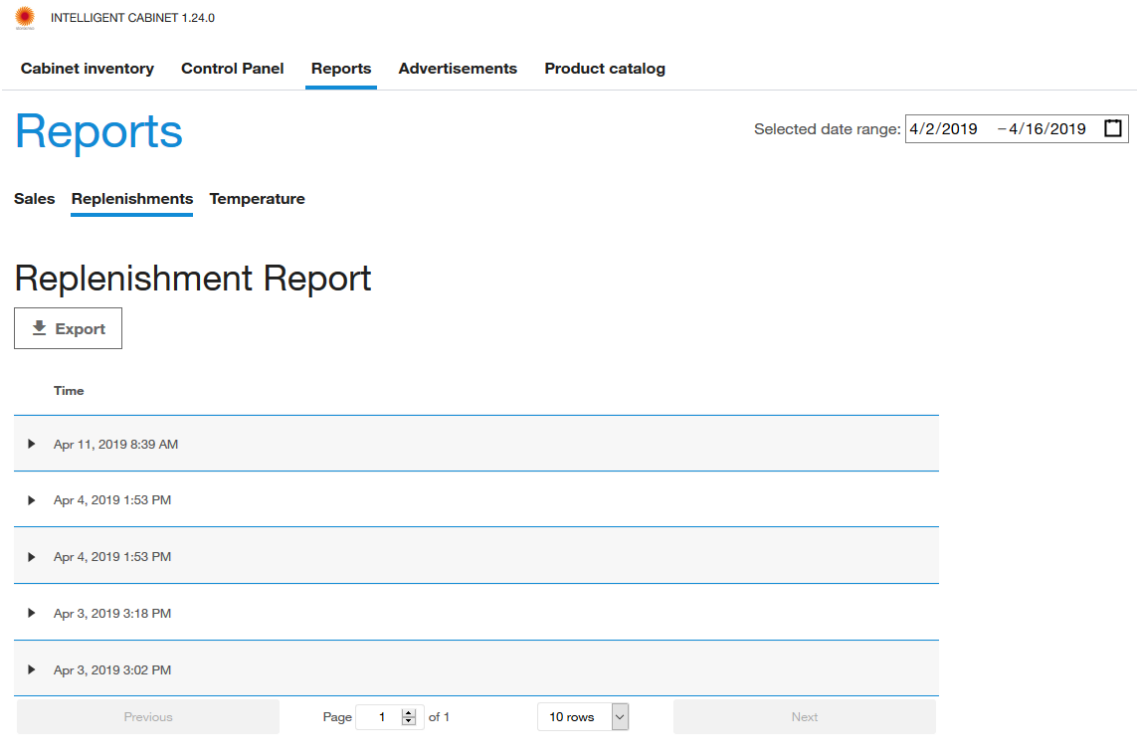
5.1 Testattavat sivustot

Työssä testattavana oli kaksi Stora Enso Intelligent Packaging:n verkkosivua, Admin Portal sekä Ecommerce. Admin Portal on verkkopalvelu IoT-laitteiden monitorointiin. Se mahdollistaa laitteiden hälytysten seurannan ja kuittaamisen sekä laitteiston tilan, esimerkiksi muistinkäytön tai suorittimen lämpötilan, havainnoinnin. Kuviossa (kuvio 2) on esitetty näkymä valitun laitteen hälytyshistoriasta.



KUVIO 2. Admin Portal

Ecommerce on palvelu itsepalvelukaapin operoimiseen ja hallintaan. Sovellus mahdollistaa kaapin ajan tasaisen sisällön tarkastelun sekä tuoteluettelon muokkaamisen. Lisäksi myynti– ja täydennysraportit ovat ladattavissa Microsoft Excel-muodossa. Kuviossa (kuvio 3) on esitetty sivuston varastontäydennysnäkymä. Erillisiä tapahtumia voidaan tarkastella tuotetasolla.



KUVIO 3. Ecommerce

5.2 Testausmetodit

Menetelmiä vertailtaessa QAutoflow:lla tehtiin testitapauksia sekä Admin Portal- että Ecommerce-sivustoille. Vertailukohdaksi Admin Portal:n osalta tehtiin samat testitapaukset myös manuaalisesti käyttäen QAutorobotia.

5.3 Testitapaukset

Admin Portalin osalta tehtiin kolme eri käyttötapausta koskevaa testitapausta, joilla testattiin kyseisen käyttöliittymän keskeiset toiminnallisuudet. Testitapaukset käsittelivät aktiivisiä hälytyksiä, hälytyshistoriaa sekä laitteen ominaisuuksien muokkaamista.

Älykaapin hallintapalvelua testattiin kahden toiminnallisuuden osalta. Testitapauksissa testattiin älykaapin tilan verifiointia, täydennystapahtuman tarkastamista täydennysraportista sekä myytävän tuotteen tietojen editointia.

5.3.1 Admin Portal – Aktiiviset hälytykset

Testitapauksen kuvaus on esitetty alla olevassa listassa.

- Kirjaudu sisään Admin Portal-sivustolle
- Valitse hiirellä näkymä Active Alarms
- Suodata hakutuloksia valitsemalla ajanjaksoksi viimeinen kuukausi
- Järjestä hakutulokset hälytysten lukumäärän mukaan
- Valitse ensimmäinen laite, jolloin laitteen tiedot avautuvat
- Valitse Device Health Status
- Verifioi levynkäytön arvo
- Kirjaudu ulos

5.3.2 Admin Portal – Hälytyshistoria

Testitapauksen kuvaus on esitetty alla olevassa listassa.

- Kirjaudu sisään Admin Portal-sivustolle
- Valitse näkymä Alarm History
- Suodata hakutuloksia valitsemalla ajanjaksoksi viimeinen viikko
- Järjestä hakutulokset viimeisimmän havaitun hälytyksen mukaisesti
- Valitse laite, jolla on tuorein hälytys
- Valitse laitteen aktiiviset hälytykset
- Verifioi hälytyksen lähde
- Kirjaudu ulos

5.3.3 Admin Portal – laitteen tietojen editointi

Testitapausten kuvaus on esitetty alla olevassa listassa.

- Kirjaudu sisään Admin Portal-sivustolle

- Valitse näkymä Devices
- Valitse ja verifioi haluttu laite
- Valitse Edit Device Information
- Muuta laitteen kenttiä: Description, Country, Latitude ja Longitude
- Tallenna muutetut tiedot
- Verifioi muuttuneen tiedot
- Kirjaudu ulos

5.3.4 Ecommerce – Kaapin tilan tarkistaminen

Testitapausten kuvaus on esitetty alla olevassa listassa.

- Kirjaudu sisään älykaapin hallintakäyttöliittymään
- Valitse haluttu älykaappi
- Valitse Control Panel - näkymä
- Verifioi älykaapin tila
- Kirjaudu ulos

5.3.5 Ecommerce – Täydennysraportin tarkastaminen

Testitapausten kuvaus on esitetty alla olevassa listassa.

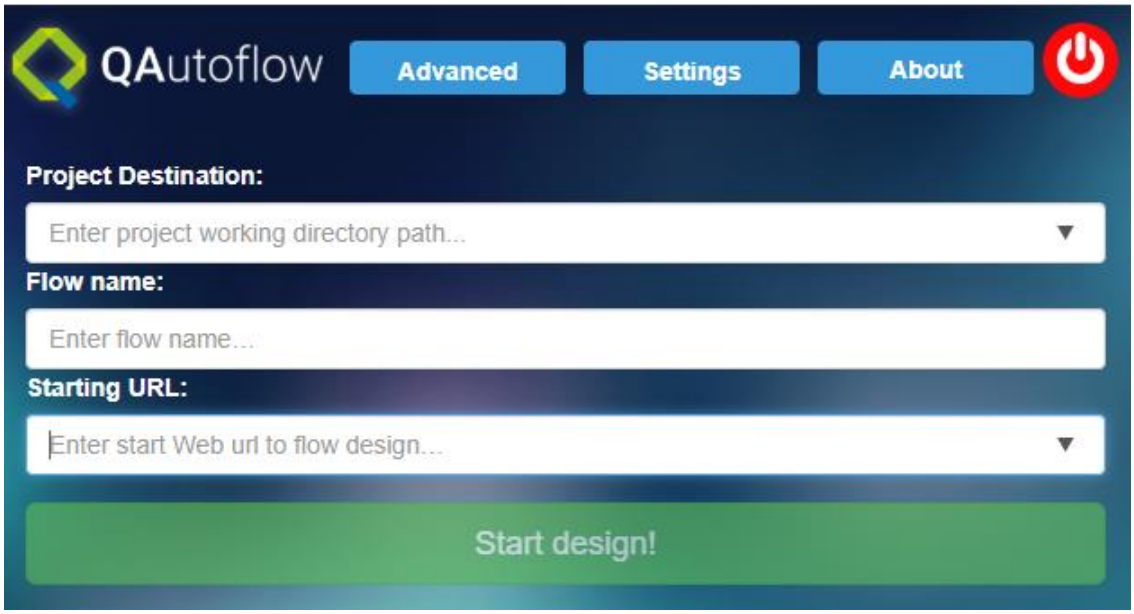
- Kirjaudu sisään älykaapin hallintakäyttöliittymään
- Valitse haluttu älykaappi
- Valitse Reports – näkymä
- Valitse Replenishments
- Valitse päivämääräalue
- Valitse viimeisin täydennystapahtuma
- Verifioi täydennystapahtuman tuote
- Kirjaudu ulos

6 PUOLIAUTOMAATTINEN TESTITAPAUS QAUTOFLOW:LLA

Esitellään yhden testitapauksen teko QAutoflow:n avulla. Testiaskelten määrittäminen tapahtuu siten, että ensin tehdään haluttu toimenpide, esimerkiksi jonkin painikkeen painaminen. Tämän jälkeen dokumentoidaan kyseinen toimenpide. Annettuja kuvauksia käytetään lopullisessa Robot Framework – testitapauksessa avainsanojen niminä, joten on pyrittävä antamaan kullekin vaiheelle kuvaava ja uniikki nimi.

6.1 Nauhoittamisen aloittaminen

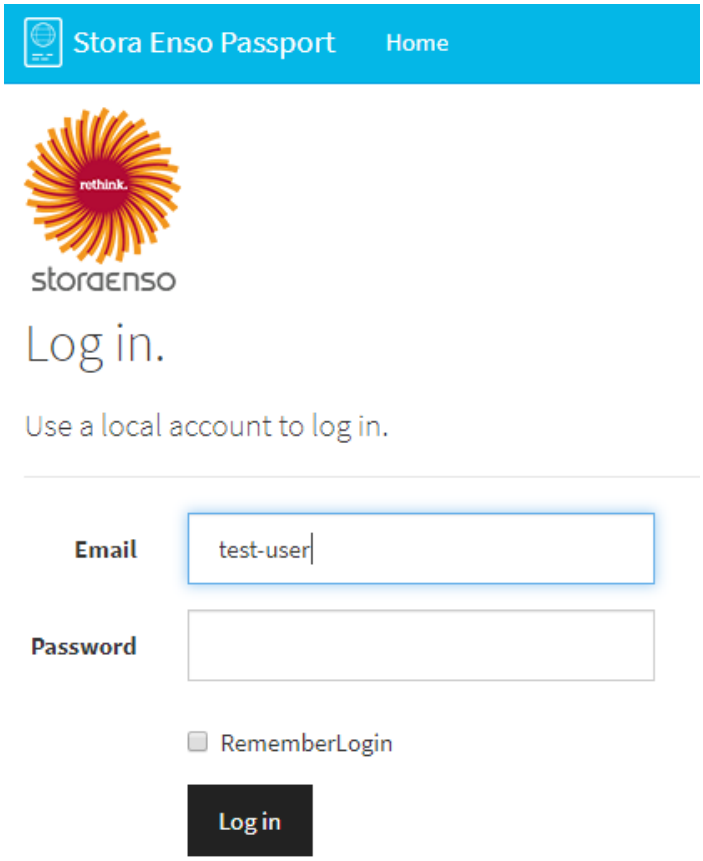
QAutoflow:n käynnistämisen jälkeen ensimmäinen vaihe on määritellä projektin asetukset. Näistä pakollisia ovat tiedostojen tallennuspolku, testitapauksen nimi sekä testattavan sivuston osoite. Pakollisten kenttien täytön jälkeen testitapauksen luonti voidaan aloittaa painamalla ”Start design!” – painiketta (kuvio 4), jolloin selain siirtyy annettuun osoitteeseen.



KUVIO 4. QAutoflow alkuasetukset

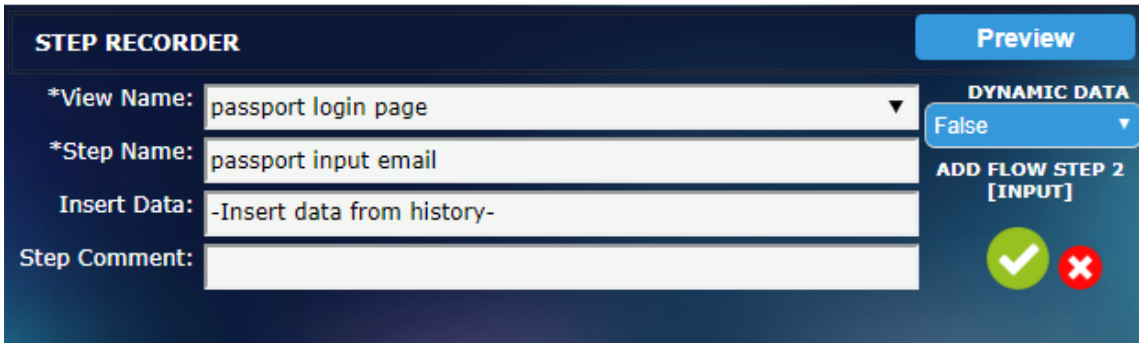
6.2 Testivaiheet

Yhdessä vaiheessa suoritetaan yksi toimenpide, joka voi olla esimerkiksi tekstikentän täyttäminen, hiiren klikkaus, tekstin verifiointi tai yhdistelmä edellä mainittuja toimenpiteitä. Kuviossa (kuvio 5) toimenpiteenä on tekstikentän täyttö.



KUVIO 5. Tekstikentän täyttö

Samalla selaimen oikeaan alakulmaan tulee näkyville dialogi (kuvio 6), joka kysyy käyttäjää dokumentoimaan tehdyn toimenpiteen. Pakollisia täytettäviä kenttiä ovat näkymän nimi sekä testivaiheen nimi. Projektikohtaisia historiatietoja kerätään paikallisesti tietokoneelle ja mikäli samaan elementtiin kohdistuva testivaihe tehdään uudelleen, ohjelma ehdottaa viimeksikäytettyjä arvoja. Tekstikentän voi täyttää kerättyjen tietojen perusteella pudotusvalikon avulla. Lopuksi vaihe tulee hyväksyä tai hylätä, jonka jälkeen voidaan siirtyä seuraavan testivaiheen nauhoitukseen. Kun kaikki halutut vaiheet on nauhoitettu, voidaan testitapaus päättää painamalla ”Stop design” – painiketta.



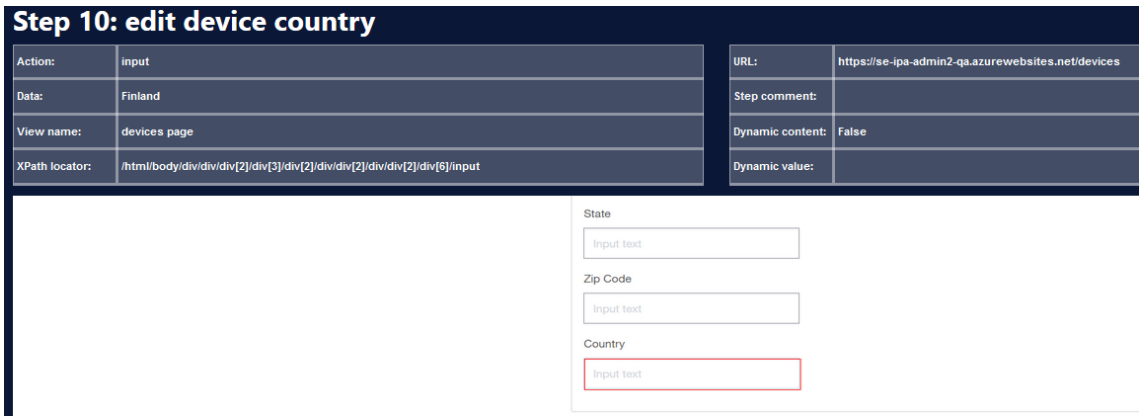
KUVIO 6. Testivaiheen dokumentointi

6.3 Raportti

Testitapauksen päättämisen jälkeen generoituu raportti, josta voi visuaalisesti tarkastella ovatko kerätyt vaiheet oikeita. Raportissa esitetään kuvion (kuvio 7) mukaisesti lista kaikista kerätyistä testivaiheista.

KUVIO 7. Yleiskatsaus nauhoitetusta testitapauksesta

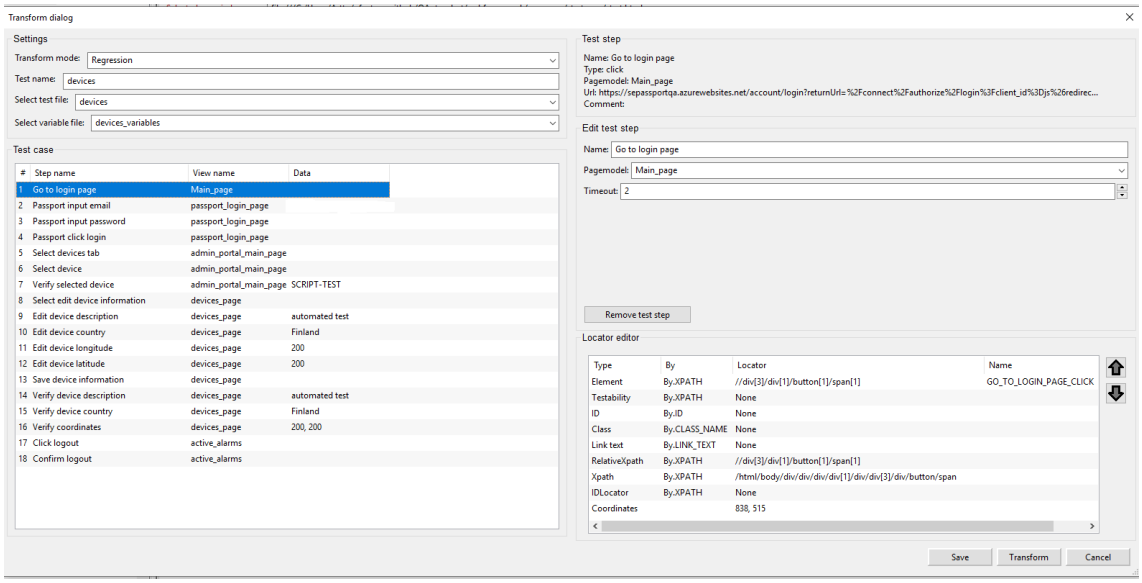
Valitsemalla tarkasteluun jokin testivaihe, näytölle aukeaa kyseisen testivaiheen tarkemmat tiedot (kuvio 8) sekä kuva, josta käy ilmi elementti johon toiminto on kohdistettu.



KUVIO 8. Testivaiheen tarkemmat tiedot

6.4 Robot Framework–testitapauksen tuottaminen

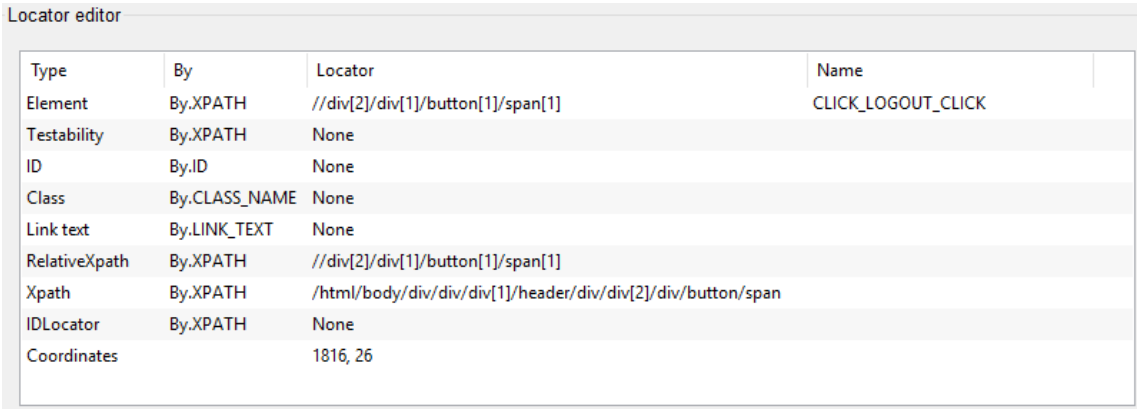
QAutoflow.n tuottama data tallennetaan json-muotoiseen tiedostoon. Tämä tiedosto voidaan lukea QAutorobot:lla ja generoida datan pohjalta Python-koodia. Kuviossa (kuvio 9) on näkymä QAutorobot:n dialogista, jossa voi muokata tallennettua dataa. Muokkaaminen on mahdollista esimerkiksi testivaiheen nimen ja datan osalta. Lisäksi kokonainen testivaihe voidaan poistaa, mikäli nauhoitusvaiheessa on tallennettu ylimääräinen testivaihe.



KUVIO 9. Tallennetun datan tarkastelu

Web-elementin lokaattorin valitseminen vaikuttaa ratkaisevasti syntyvän testitapauksen toimivuuteen, mikäli testattava sivusto muuttuu. Tallennettu data sisäl-

tää useita mahdollisia vaihtoehtoja elementin etsimiseksi ja tämän prioriteettilistan sisältöä ja järjestystä voidaan muokata. QAutorobot käy listan läpi testitapausta generoidessaan ja käyttää ensimmäistä toimivaa lokaattoria tai viimeisenä vaihtoehtona web-elementin koordinaatteja. Kuviossa (kuvio 10) esitetään mahdollisia lokaattoreita, joista tarkasteltavalla elementillä ei ole kuin Xpath:n eri variaatiot.



KUVIO 10. Lokaattorin valinta

Datan tarkistamisen jälkeen painikkeella ”Transform” aloitetaan testitapauksen automaattinen generointi. Tietojen perusteella suoritetaan samat tallennetut toiminnot uudessa selainikkunassa ja muodostetaan Robot Framework–avainsanat. Testivaiheissa käytetyt tekstisyötteet, esimerkiksi käyttäjätunnus, tallennetaan erillisiin resurssitiedostoihin.

6.5 Robot Framework-testitapaus

QAutorobot käyttää niin sanottuja sivumalleja selkeyttämään testien rakennetta. Sivumalli on Python-luokka, johon tallennetaan näkymässä käytetyt elementit

sekä funktiot. Kuviossa (kuvio 11) Stora Enso Passport-sisäänkirjautumissivua vastaava sivumalli.

```python
# -*- coding: utf-8 -*-
import QAutoRobot
from QAutoLibrary.QAutoElement import QAutoElement

from selenium.webdriver.common.by import By
from time import sleep


class Passport_login_page():
    # Pagemodel timestamp: 20190414110306
    # Pagemodel url: https://sepassportqa.azurewebsites.net/account/login?returnUrl=%2Fconnect%2Fauthorize%2Flogin%3Fclient_id%3Djs%26redirect_u
    # Pagemodel screen resolution: (1920, 1080)

    PASSPORT_CLICK_LOGIN_CLICK = QAutoElement((By.XPATH, u'//form[1]/div[5]/div[1]/button[1]'), coordinates=(488, 471), size=(76, 43))
    PASSPORT_INPUT_PASSW_INPUT = QAutoElement((By.ID, u'Password'), coordinates=(488, 366), size=(280, 43))
    PASSPORT_INPUT_EMAIL_INPUT = QAutoElement((By.ID, u'Email'), coordinates=(488, 308), size=(280, 43))


    def __init__(self):
        """
        Pagemodel initialization

        """
        pass

    def passport_click_login(self):
        QAutoRobot.click_element(self.PASSPORT_CLICK_LOGIN_CLICK)

    def passport_input_email(self, text=None):
        QAutoRobot.input_text(self.PASSPORT_INPUT_EMAIL_INPUT, text)

    def passport_input_password(self, text=None):
        QAutoRobot.input_text(self.PASSPORT_INPUT_PASSW_INPUT, text)
```

KUVIO 11. Sivumallin sisältö Python-koodina

Nauhoituksen aikana dokumentoitua näkymän nimeä käytetään sivumallin nimenä ja testivaiheiden nimiä puolestaan funktioiden niminä. QAutorobot:n editori näyttää sivumallia vastaavan kuvakaappauksen, josta selviää mistä näkymästä on kysymys ja miten tallennetut elementit sijoittuvat sivumalliin.

Python-funktioita käytetään avainsanoina testitapauksessa. Testitapauksen toiminta on ymmärrettävissä selkän avainsanojen nimeämisen ansiosta, vaikka ei tuntisi avainsanojen tarkempaa sisältöä. Testitapauksen rakenne on esitetty kuviossa (kuvio 12).

```
3  Suite Setup    Test suite setup
4  Suite Teardown    Test suite teardown
5  Test Setup    Setup
6  Test Teardown    Teardown
7
8  Library    QAutoLibrary.QAutoRobot
9  Resource    ../resources/setup_and_teardown.robot
10 Resource    ../resources/Active alarms_variables.robot
11
12 *** Variables ***
13
14
15 *** Test Cases ***
16 Test Active alarms
17
18    Open application url    ${url_Active alarms}
19    Go to login page
20    Passport input email    ${passport_login_page Email}
21    Passport input password    ${passport_login_page Password}
22    Passport click login
23    Click filter devices button
24    Timeframe select last month
25    Click order by count
26    Select first device
27    Click device health status
28    Verify filesystem usage    ${active_alarms Verify filesystem usage_verification}
29    Click logout
30    Confirm logout
```

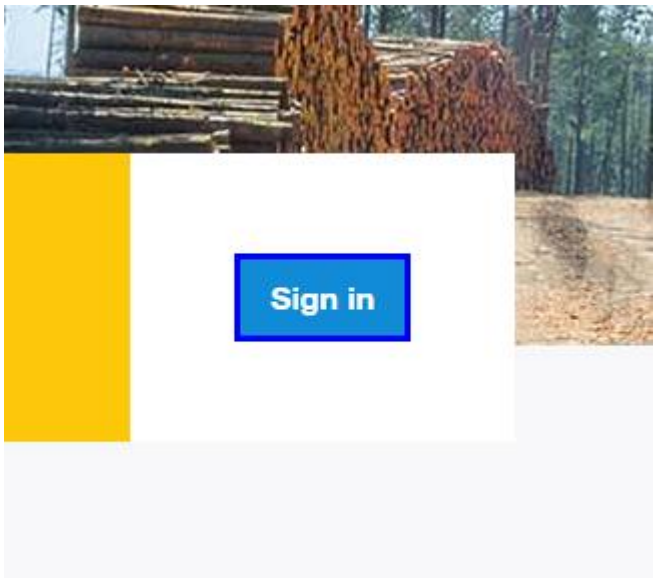
KUVIO 12. Generoitu testitapaus

7 MANUAALINEN TESTITAPAUS QAUTOROBOT:LLA

Testitapauksen manuaalinen tekeminen noudattaa kaavaa, jossa luodaan ensin web-sivun näkymästä tyhjä sivumalli, johon lisätään tarvittavat elementit sekä funktiot. Eniten aikaa kuluu elementtien lokaattoreiden määrittämiseen sekä itse funktioiden kirjoittamiseen.

7.1 Elementtien manuaalinen etsiminen

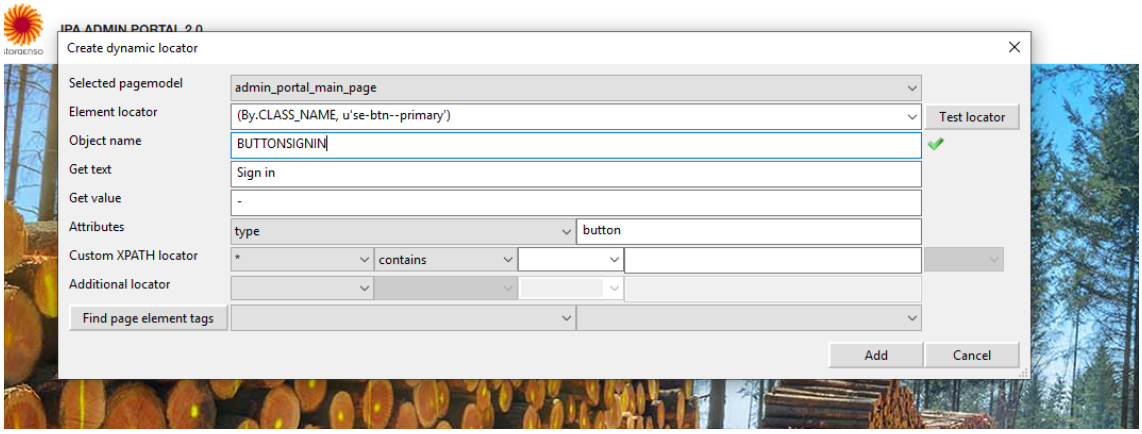
Elementin lokaattorin määrittämiseen voi käyttää esimerkiksi selaimen kehittäjätyökaluja, joilla voi tarkastella sivun rakennetta. QAutorobot mahdollistaa elementin lokaattorien määrittämisen ja niiden testaamisen. Tarkastellaan esimerkkiä, jossa etsitään kuvion (kuvio 13) mukaiselle painikkeelle lokaattori.



KUVIO 13. Valittu elementti korostettuna

Lokaattorin määrittäminen onnistuu kuvion (kuvio 14) mukaisen dialogin avulla. QAutorobot etsii automaattisesti elementille lokaattoreita, joista voi valita mieleisen pudotusvalikosta ja tallentaa elementin haluamaansa sivumalliin. Monimutkaisempien lokaattorien muodostaminen onnistuu räätälöimällä Xpath-lokaattori esimerkiksi hakemalla tietyn tyyppisten elementtien sisältämiä sanoja. Tämä toi-

mii etenkin, jos tiedetään ettei sivulla ole muita kyseisen sanan sisältäviä elementtejä. Hakuehtoja voidaan myös ketjuttaa and – ja or – operaattoreiden avulla.



KUVIO 14. Lokaattorin muodostaminen

7.2 Robot Framework testitapaus

Testitapauksen rakenne on sama verrattuna QAutoflow:lla tehtyyn testiin. Erottavana tekijänä avainsanat ja itse testitapaus on kirjoitettava käsin Python-koodina automaattisen generoinnin sijaan. Lisäksi testitapauksessa käytettävät muuttujat on luotava resurssitiedostoon.

8 TULOKSET

Testitapausten perusteella suoritettiin arviointi kummankin menetelmän vahvuuksista ja heikkouksista eri osa-alueilla. Arvioinnin pääpaino oli QAutoflow:n suorituskyky suhteessa manuaaliseen testitapausten toteuttamiseen QAutorobot:lla.

8.1 Ajankäyttö

Ajankäytön arvioinnissa ei huomioida testitapauksien suunnitteluun kulunutta aikaa, koska sen oletetaan olevan sama molemmille menetelmille. Testitapauksen tekeminen QAutoflow:lla vaati keskimäärin noin kymmenen minuuttia aikaa. Manuaalisen menetelmän tapauksessa aikaa kului noin yksi tunti testitapausta kohden.

Kummankin menetelmän käyttäminen nopeutuu suhteessa testitapaukseen, kun testitapausten määrää lisätään. QAutoflow;n historiatietojen keruu nopeuttaa testivaiheen dokumentointia esitäyttämällä tekstikentät valmiiksi, jolloin saavutetaan huomattavaa nopeusetua. Nopeutuminen koskee myös manuaalista testien kirjoittamista QAutorobot:lla, mikäli avainsanat on toteutettu uudelleenkäytettävyys huomioiden.

Puhtaasti ajankäytön kannalta voidaan todeta, että QAutoflow on melko ylivoimainen tarkasteltaessa yksinkertaisia testitapauksia. QAutorobot:n manuaalisen generoinnin vaatimaan aikaan vaikuttaa selvästi testaajan ammattitaito. Kokeneempi testaaja päässee huomattavasti parempiin tuloksiin etenkin tutuissa ympäristöissä.

8.2 Testitapausten luotettavuus

Suoraviivaisia testitapauksia käsiteltäessä QAutoflow:n nauhoittamiseen perustuva menetelmä pääsi varsin toimiviin lopputuloksiin. Generoitujen testitapausten rakenne vastaa pääosin manuaalisia testitapauksia. Testivaiheiden määrän kas-

vaessa testitapauksessa, kasvaa myös itse testitapauksen mitta jokaisen testivaiheen muodostaessa oman avainsanansa. Manuaalisella menetelmällä tietyn kokonaisuuden avainsanat voidaan yhdistää yhden avainsanan alle, jolloin luettavuus ja selkeys paranee. Tämä onnistuu melko pienellä vaivalla myös QAutoflow:n generoinnin jälkeen, mutta vaatii tutustumista tuotetun testikoodin rakenteeseen.

Monimutkaisempia toimintoja testattaessa törmättiin selviin rajoitteisiin QAutoflow:n osalta. Testattavan sivuston dynaamisuus vaikeuttaa luotettavien testitapausten generointia. Kun datan määrä ja laatu ei ole ennakolta tiedossa on vaikeata luoda ohjelmallisesti koodia, joka toimii luotettavasti eri tilanteissa. Haaste koskee osin myös manuaalista menetelmää, mutta tällöin voidaan käyttää monimutkaisempia hakufunktioita yksittäisen elementin lokaattorin sijaan. Tilanne helpottuu, mikäli sivuston suunnittelussa on huomioitu testattavuus ja relevanteille elementeille on annettu asianmukaiset tunnisteet.

8.3 Ongelmakohdat

Haastavien tai huonosti skaalautuvien käyttötilanteiden lisäksi testattavilla sivuilla voidaan törmätä tapauksiin, jotka on pakko toteuttaa manuaalisesti. Osa toiminnoista on turvallisuussyistä johtuen rajattu selaimen kontekstin ulkopuolelle. Esimerkiksi kuvan lataaminen palvelimelle ei onnistu selaimesta käsin, vaan tällöin tarvitaan manuaalisesti tehtäviä funktioita, joilla on pääsy laitteen tiedostojärjestelmään.

Ongelmia aiheuttaa myös standardista poikkeavat toteutukset testattavilla sivustoilla. Jotkin sivustot estävät esimerkiksi hiiren painalluksia tai joitakin muita toimintoja tai vaihtoehtoisesti käsittelevät niitä normaalista poikkeavasti. Näissä tapauksissa toimintojen nauhoittaminen häiriintyy aiheuttaen väärin tulkittuja tai puuttuvia testivaiheita.

9 POHDINTA

Testitapausten tuottaminen QAutoflow:n avulla osoittautui toteutettujen testitapausten perusteella toimivaksi menetelmäksi tietyin reunaehdoin. QAutoflow:n suurimmat vahvuudet olivat huomattava ajansäästö yksinkertaisissa testitapauksissa sekä pienempi oppimiskynnys manuaaliseen testiskriptien kirjoittamiseen verrattuna.

QAutoflow ei kuitenkaan sovellu ainoaksi tavaksi tuottaa tarvittavat testiautomaatioskriptit, vaan manuaalisia menetelmiä tarvitaan monimutkaisempien testitapauksien tuottamiseen ja niiden ylläpitoon. QAutoflow:n käyttökelpoisuuteen vaikuttaa merkittävästi, onko testattavan sivuston suunnittelussa ja toteutuksessa huomioitu testattavuutta. Luodut testiautomaatioskriptit sietävät huonosti testattavan sivuston muutoksia, mikäli tarvittaville web-elementeille ei ole annettu kunnollisia attribuutteja niiden luotettavan paikantamisen mahdollistamiseksi. Havaittuja ongelmia voidaan minimoida käyttämällä manuaalisesti luotuja avainanoja osana testitapausten nauhoitusta.

QAutoflow:ta pystytään hyödyntämään parhaiten, jos sitä käytetään yhdessä manuaalisen menetelmän kanssa. Tällöin testiautomaatioon erikoistumattomat henkilöt voivat osallistua testitapausten suunnitteluun ja toteutukseen, mutta ammattitaitoinen testaaja vastaa testien lopullisesta tuottamisesta ja ongelmallisten testivaiheiden toteuttamisesta. QAutoflow nopeuttaa myös kokeneen testaajan työtä, joten hyöty ei rajoitu ainoastaan kokemattomampien testaajien hyödyntämiseen testauksessa.

LÄHTEET

Software Testing Help. What is automation Testing. Luettu 7.4.2019.
https://www.softwaretestinghelp.com/automation-testing-tutorial-1/

Software Testing Class. Gui testing in software testing. Luettu 7.4.2019.
https://www.softwaretestingclass.com/gui-testing-in-software-testing/

ProTech. Selenium Tutorial: Locators. Luettu 8.4.2019.
https://www.protechtraining.com/content/selenium_tutorial-locators

Test IO. What is regression testing? Luettu 8.4.2019.
https://test.io/blog/what-is-regression-testing/

CloudBees CodeShip. Continuous integration essentials. Luettu 9.4.2019.
https://cms.codeship.com/continuous-integration-essentials

Robot Framework. Robot Framework User Guide. Luettu 13.4.2019.
http://robotframework.org/robotframework/latest/RobotFrameworkUser-Guide.html

SeleniumHQ. Selenium Documentation. Luettu 13.4.2019.
https://www.seleniumhq.org/docs/index.jsp

QAutomate. QAutoflow. Luettu 14.4.2019.
https://www.qautomate.fi/qautoflow

QAutomate. QAutorobot Luettu 14.4.2019.
https://www.qautomate.fi/qautorobot

QAutomate. QAutoLibrary. Luettu 14.4.2019.
https://github.com/QAutofamily/QAutoLibrary