Jali Kauppinen

IoT-portaalin testien automatisointi

Opinnäytetyö
Syksy 2018
SeAMK Tekniikka
Automaatiotekniikka (AMK, Insinööri)

SeAMK
SEINÄJOEN AMMATTIKORKEAKOULU
SEINÄJOKI UNIVERSITY OF APPLIED SCIENCES

SEINÄJOEN AMMATTIKORKEAKOULU

Opinnäytetyön tiivistelmä

Koulutusyksikkö: Seamk Tekniikka

Tutkinto-ohjelma: Automaatiotekniikka

Suuntautumisvaihtoehto: Koneautomaatio

Tekijä: Jali Kauppinen

Työn nimi: IoT-portaalin testien automatisointi

Ohjaaja: Marko Hietamäki

Vuosi: 2018 Sivumäärä: 43

Tämän opinnäytetyön tarkoituksena oli selvittää, voidaanko Epec Oy:n IoT-portaalin testaaminen automatisoida. Selvitystyö tehtiin Epec Oy:n tuotekehitykseen, jossa automaattitestejä tarvittiin palveluun tulevien päivitysten toiminnan varmentamiseksi ja palvelun toimintavarmuuden ylläpitämiseen.

Selvitystyössä tutustuttiin muun muassa testausautomaatioon ja Epec Oy:n IoT-palveluihin. Tämän lisäksi työssä käytiin läpi käytettyjä työkaluja kuten ohjelmointiympäristöt, sekä Epec Oy:n tarjoama konfiguraatiotyökalu ohjausyksiköille.

Työn tavoitteena oli löytää tapa sekä luoda ohjelmistot IoT-portaalin testaamiseen. Työssä hyödynnettiin Epec Oy:n työntekijöiden aiempaa kokemusta testausautomaatiosta, sekä tekniikkoihin liittyvää aineistoa verkosta sekä kirjallisuudesta. Työssä käsiteltiin muun muassa CODESYS-ohjelmointia, Robot Frameworkilla testien luomista, sekä Wapice Oy:n tarjoaman ohjelmointirajapinnan hyödyntämistä.

Selvitystyön lopputuloksena luotiin Robot Frameworkiin valmiita testejä, joilla voidaan testata palvelun toimivuutta, sekä sovellus, jolla voidaan lisätä simuloituja laitteita palveluun ohjelmointirajapinnan kautta.

Avainsanat: Epec, automaattitestit, IoT, Robot Framework, ohjelmointirajapinta, tuotekehitys

SEINÄJOKI UNIVERSITY OF APPLIED SCIENCES

Thesis abstract

Faculty: School of Technology

Degree programme: Automation Technology

Specialization: Machine Automation

Author: Jali Kauppinen

Title of thesis: Automated tests for an IoT-portal

Supervisor: Marko Hietamäki

Year: 2018 Number of pages: 43

The purpose of this thesis was to discover a possibility to automate the testing of the IoT-portal of Epec Oy. The study was made for the research and development department of Epec Oy, which uses automated tests for various testing purposes. The automated tests would be used in case of an IoT-portal update and to ensure that the service is functioning correctly.

The theoretical part of this thesis introduced the methods and software that were used during the project. Also, the configuration tool for the control units of Epec Oy and such programming environments as Visual Studio and CODESYS, were presented. The theoretical part also introduced general information on test automation, whereas regression testing and load where studied in more detail.

The target of this thesis was to find a way to create a software for the automatic testing of the IoT-service. During the study, the employees of Epec Oy helped with the project and information was collected from the internet and literature. The thesis work also reviewed CODESYS programming, creating tests with Robot Framework and the use of the programming interface of Wapice Oy.

The final result of the study was a set of ready-made test cases on the usability of the IoT-portal and an application for the IoT-service, which can be used to add simulated devices to the IoT-portal through the programming interface.

Keywords: Epec, automated tests, IoT, Robot Framework, programming interface, research and development

SISÄLTÖ

Kuvaluettelo

KÄYTETYT TERMIT JA LYHENTEET

API	Application Programming Interface, ohjelmointirajapinta.
Dashboard	GlobE-palvelussa oleva rajapinta, jota käyttäen voidaan visualisoida arvoja.
GlobE	Epec Oy:n tarjoama palvelu, jolla voidaan tutkia koneenohjausjärjestelmästä lähetettyä tietoa verkossa.
GatE	Epec Oy:n tarjoama palvelu, jolla mahdollistetaan etäyhteys ohjausyksiköihin.
GUI	Graphical User Interface, graafinen käyttöliittymä.
HTTPS	Hyper Text Transfer Protocol Secure, protokolla, jota käytetään suojattuun tiedonsiirtoon webissä.
IDE	Integrated Development Environment, integroitu ohjelmointiympäristö.
IoT	Internet of Things, esineiden internet.
PWM	Pulse width modulation, pulssinleveysmodulaatio.
RIDE	Käyttöliittymä Robot Frameworkin käyttämiseen.

1 JOHDANTO

1.1 Työn tausta

Epec Oy:ssä on siirrytty käyttämään yhä enemmän testausautomaatiota niin laitteiden sovelluskoodissa kuin muihin palveluihin liittyen. Epec Oy on julkaissut IoT-palvelun GlobE, jonka testausta ei ole vielä automatisoitu, mikä käyttää sovellustestaajien aikaa testaamiseen. Testaus voidaan mahdollisesti automatisoida, jolloin päästäisiin eroon julkaisun yhteydessä tehtävistä manuaalisista testeistä. Näiden testien avulla varmistetaan vanhojen toimintojen toimivuus. Tämän lisäksi ilmeni myös halu suorittaa testejä, kuinka palvelu toimii suuremman rasituksen alla simuloimalla sinne laitteita GlobE-palvelun API-rajapinnan kautta ja lähettämällä näillä dataa palvelimelle.

1.2 Työn tavoite ja rajaus

Työn tavoitteena on tutkia, voidaanko Epec Oy:n IoT-palvelun GlobE käytettävyystestejä automatisoida. Lisäksi tutkitaan, voidaanko GlobE-palvelun ohjelmointirajapintaa hyödyntää palvelua rasittavia testejä tehdessä. Työ rajattiin koskemaan IoT-palveluista ainoastaan GlobE-palvelua ja näin ollen tutkimuksesta rajattiin ulos GatE-palvelun testaaminen.

1.3 Työn rakenne

Luvussa 2 käydään läpi työhön käytettyjen ohjelmistojen teoriaa, sekä tutkitaan niiden tarjoamia mahdollisuuksia. Näiden lisäksi käydään myös läpi automaatiotestaamisen perusperiaatteita sekä syvennytään työssä käytettyihin metodeihin.

Luvussa 3 kerrotaan tarpeesta, minkä vuoksi tutkimusta suoritettiin. Tämän lisäksi käsitellään asioita, kuten vaatimuksia, joita tutkimus sai aloituspalaverissa sekä työn edetessä. Lisäksi luvussa kerrotaan sovitusta aikataulusta, joka työlle määräytyi.

Luvussa 4 käsitellään syitä, minkä vuoksi tietyt ohjelmointiympäristöt sekä ohjelmointikielet valikoituivat käytettäviksi työssä. Luvussa kerrotaan myös asioista työn aloittamiseen liittyen, kuten esimerkiksi IoT-portaaliin liittyvistä alustuksista.

Luvussa 5 käydään läpi Epec Oy:n etäohjauslaitteen CODESYS-ohjelman luomiseen liittyviä asioita, sekä tarkastellaan, mitä konfigurointitiedostoja etäohjauslaite vaatii, jotta yhteys IoT-portaaliin voidaan toteuttaa.

Luvussa 6 käsitellään Robot Frameworkilla suoritettavia regressiotestejä ja nähdään kaavio testien toiminnasta, kun kaikki testikohdat suoritetaan. Luku käsittelee lopuksi Robot Frameworkin käytössä havaittuja ongelmia.

Luvussa 7 kerrotaan C#-kielellä luodusta kuormitustestistä, sen graafisesta käyttöliittymästä ja rakenteesta. Lopuksi kerrotaan, mitä ongelmia kohdattiin, kun kuormitustestin sovellusta ohjelmoitiin.

Luvussa 8 esitetään ajatuksia projektin mahdollisista kehitystarpeista ja tavoista, joilla ohjelmasta saataisiin käyttäjäystävällisempi ja hyödyllisempi tarpeen esittäneelle yritykselle. Lisäksi esitetään mahdollinen korvaava vaihtoehto, jolla suorittaa kuormitustestit.

Luvussa 9 esitetään yhteenveto projektista, kerrotaan projektin onnistumisesta sekä muista huomioonotettavista asioista.

1.4 Epec Oy

Epec Oy on seinäjokinen teknologia-alan yritys, joka on erikoistunut sulautettuihin ohjausjärjestelmiin. Alun perin Epec Oy on perustettu vuonna 1978 E-P Elektroniikka nimellä Veikko Rintamäen toimesta. Ensimmäiset 15 vuotta Epec Oy toimitti markkinoille erinäisiä elektroniikkatuotteita, kuten esimerkiksi suuria ulkoilmanäyttöjä ja tietokoneita. 1990-luvun puolivälissä Epec Oy teki strategisen päätöksen siirtyä ohjausyksikköbisnekseen, erikoistuen lujatekoisiin, liikkuvissa työkoneissa käytettäviin laitteisiin. Vuonna 2004 Epec Oy siirtyi metsäkoneyhtiö Ponsse Oyj:n omistukseen ja toimii sen tytäryhtiönä tänäkin päivänä. (Epec [Viitattu 6.9.2018].)

Epec Oy:ssä valmistetaan ohjausyksiköitä, jotka on suunniteltu erityisen vaativiin olosuhteisiin. Ohjausyksiköiden lisäksi Epec Oy:n tuotevalikoimaan kuuluu myös näyttölaitteita ja etäohjausyksiköitä. Epec Oy:n tuotteissa on tärinäkestävyys aina 100 G:hen asti, sekä IP67-luokitus, poisluettuna näyttötuotteet, joista 7-tuumaisessa versiossa on IP65-luokitus ja 12-tuumaisessa versiossa IP66-luokitus. Epec Oy on ensimmäinen suomalainen yritys, joka on julkaissut SIL2-sertifioidun turvaohjausyksikön. (Epec [Viitattu 6.9.2018].)

Fyysisten laitteiden lisäksi Epec Oy suorittaa myös palvelumyyntiä, kuten esimerkiksi projektipalveluita asiakkaille, jotka haluavat ”avaimet käteen” -ratkaisuja. Tässä palvelumuodossa toteutetaan asiakkaiden vaatimusten sekä tarpeiden puitteissa sovelluksia Epec Oy:n ohjausyksiköillä toimiviin koneenohjausjärjestelmiin. Epec Oy:llä on myös teknisen tuen tiimi, joka on sitoutunut auttamaan Epec Oy:n asiakkaita niin ohjelmoinnillisissa ongelmissa kuin Epec Oy:n tuotteisiin liittyvissä kysymyksissä. Lisäksi tekninen tuki tarjoaa asiakkailleen koulutuksia, jotka voidaan räätälöidä asiakkaiden tarpeiden mukaan. Koulutus voidaan järjestää niin ohjausyksiköiden ohjelmoinnista, tuotteista tai muihin palveluihin liittyen. (Epec [Viitattu 6.9.2018].)

Viime vuosina Epec Oy:ssä on panostettu myös IoT-palveluihin. Tuotevalikoimaan kuuluu tällä hetkellä GlobE sekä GatE. GlobE on Wapicen IoT-Ticket-palveluun pohjautuva alusta, josta on mahdollista seurata työkoneen sinne lähettämää dataa, kuten esimerkiksi paikkatietoa, käyttötunteja sekä toimilaitteiden käyttöastetta. GlobE on Epec Oy:n laitteille räätälöity ratkaisu IoT-palveluiden helppoon käyttöönottoon. GatE-palvelulla voidaan ottaa etäyhteys laitteeseen. Näin ollen päästään käsiksi ohjausyksiköillä olevaan ohjelmakoodiin, sekä voidaan etänä siirtää tiedostoja, joita on GatE-palvelua tukevalla etäohjauslaitteella, esimerkiksi loki väyläliikenteestä. (Epec [Viitattu 6.9.2018].)

Kuva 1. Ponsse Scorpion King (Ponsse [Viitattu 27.9.2018].)

2 AUTOMAATTITESTAUS JA KÄYTETYT TYÖKALUT

2.1 Testauksen automatisointi

Testausta automatisoidaan muun muassa ajan ja rahan säästämiseksi sekä mahdollisesti myös laadun takaamiseksi. Testiautomaatiolla voidaan suorittaa ihmiselle puuduttavia testejä ilman suurempaa vaivaa ja näin ollen voidaan välttää huolimattomuudesta johtuvia virheitä. (Broekman & Notenboom 2002, 217.)

Yleisiä käyttökohteita ovat esimerkiksi:

- Testit jotka pitää suorittaa monta kertaa
- Yksinkertaiset testit joita tehdään monta kertaa, mutta muuttujien arvoja muutetaan
- Monimutkaiset ja virhealttiit testit.

Muutokset, joita tapahtuu järjestelmässä, suunnittelussa sekä vaatimuksissa ovat yleisiä ja myös riski testien toimivuudelle. Näiden asioiden seurauksia voivat olla muun muassa muutoksille tehtävät ylimääräiset testit tai muutokset olemassa oleviin testeihin. Lisäksi voidaan joutua tarkastelemaan eri asioita kuin alkuperäisissä testeissä, testeissä käytettyjen muuttujia voidaan joutua määrittelemään uudestaan sekä testien toiminnallisuksiin voi tulla muutoksia. (Broekman & Notenboom 2002, 218.)

2.2 Regressiotestaus

Regressiotestauksessa testataan järjestelmän toimintaa kokonaisuutena, esimerkiksi suuren päivityksen jälkeen toiminnallisuuksien toimivuutta testataan osittain tai kokonaisuutena. Tämän lisäksi testataan myös, ettei järjestelmään tehdyt päivitykset tai lisäykset ole vaikuttaneet jo olemassa olevien toimintojen toimivuuteen. Regressiotestausta tehdään, kun vaatimukset päivittyvät ja koodiin tehdään muutoksia,

ohjelmaan lisätään uusia toiminallisuuksia ja kun korjataan virheitä tai parannetaan suorituskykyä. (Anderson 2011.)

Ohjelman ylläpitoon kuuluu, että siihen tehdään parannuksia, virheiden korjausta, optimointia, sekä olemassa olevien ominaisuuksien poistamista. Näiden toimintojen seurauksena järjestelmä saattaa toimia odottamattomalla tavalla ja se on yksi pääsyistä, minkä vuoksi regressiotestausta tehdään. (Anderson 2011.)

Regressiotestauksen haasteina on, että testisetit kasvavat suuriksi ja näin vaikeiksi ylläpitää. Suurien järjestelmien kanssa tämä tarkoittaa, ettei yrityksellä ole välttämättä resursseja suorittaa regressiotestausta koko järjestelmälle, jolloin suoritetaan osittaista regressiotestausta. Testejä minimoidessa tulee huomioida, että se vaikuttaa myös testien kattavuuteen. (Anderson 2011.)

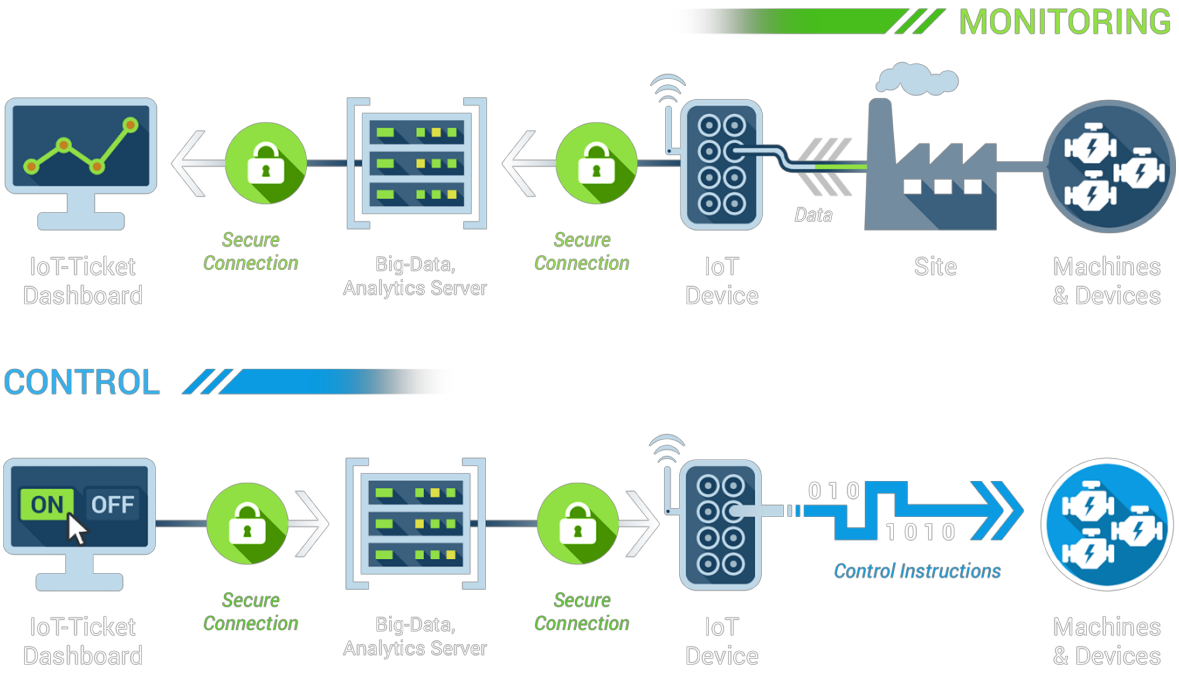
2.3 Kuormitustestaus

Kuormitusta voidaan testata esimerkiksi tekemällä äärimmäisiä rasitustestejä (stressitestaus), jotka pyrkivät hajottamaan järjestelmän, ja joilla tutkitaan järjestelmän toimintaa erittäin suuren paineen alla. Vaihtoehtoisesti testejä voidaan suorittaa käyttämällä taakkaa, joka vastaa oletettua käyttöä, ja tutkia kuinka järjestelmän odotettu suorituskyky toteutuu. Kun testataan odotettua käyttöä, on syytä ottaa huomioon mahdolliset piikit käytössä, esimerkiksi tietyt kellonajat. Näin testistä saadaan kattavampi ja voidaan kasvattaa järjestelmän uskottavuutta suuren dataliikenteen alla, sekä voidaan tunnistaa pullonkaulat ajoissa. (Menascé 2002.)

Kuormitustestauksella voidaan paikantaa järjestelmän ongelmakohtia, ennen kuin sitä toimitetaan markkinoille. Näin saadaan selville esimerkiksi aika, mikä kuluu eri toimien suorittamiseen, sekä tietoa järjestelmän komponenttien kestosta rasituksen alla. Lisäksi saadaan selville, minkälaisia viiveitä tulee asiakkaan ja palvelimen välille suuren käytön alla, mahdollisia ohjelmistosuunnittelun ongelmia, sekä palvelimien konfiguraatio-ongelmat. (Menascé 2002.)

2.4 IoT-Ticket

IoT-Ticket on Wapicen tarjoama alusta esineiden internetiin. Alustan avulla voidaan internetin kautta saada dataa laitteista, joihin ei muuten päästä käsiksi. Aluksi toimilaitteilta toimitetaan dataa etähallintalaitteeseen, josta toimitetaan datapaketti palvelimelle. Palvelimelta data saadaan näkyviin IoT-Ticket-portaalissa, jossa data voidaan visualisoinneilla muuttaa helppolukuisemmaksi. Järjestelmästä saadulla datalla mahdollistetaan esimerkiksi huoltojen ennakointi, parametrien säätäminen etänä järjestelmän asentamisen jälkeen sekä järjestelmän käyttöasteen seuraaminen. (Wapice [Viitattu 16.9.2018].)



Kuva 2. IoT-Ticketin käyttöperiaate (Wapice [Viitattu 16.9.2018].)

2.4.1 Dashboard

IoT-Ticketin Dashboard on selainpohjainen rajapinta loppukäyttäjille. Dashboardissa käyttäjä voi tarkastaa laitteen tilan, tutkia laadittuja raportteja, sekä saada tietoa laitteen suorituskyvystä ja käyttöasteesta. (Wapice [Viitattu 16.9.2018].)

Dashboardin välilehtien avulla käyttäjä voi yksilöidä näkymät eri tarpeisiin, kuten parametritaulukon, tulojen ja lähtöjen tarkastelun, karttanäkymän tai muuttujien tilan mukaan. Tämän lisäksi voidaan myös luoda näkymiä kaikille yrityksen hallinnassa

oleville koneille. Näin nähdään yhdellä välilehdellä yleiskatsaus laitteiden tilasta. (Wapice [Viitattu 16.9.2018].)

2.4.2 Ohjelmointirajapinta

Ohjelmointirajapinta (API) on välittäjä, joka mahdollistaa kahden erillisen sovelluksen yhteydenpidon. Kun toimeksiantajana toimiva sovellus kysyy palvelimelta tietoa, se tapahtuu API-rajapinnan kautta. Sieltä lähtee pyyntö palvelimelle, joka käsittelee pyynnön ja lähettää halutun tiedon toimeksiantajalle API-rajapintaa hyödyntäen. (MuleSoft [Viitattu 17.9.2018].)

IoT-Ticketissä on ohjelmointirajapinta, jota kutsutaan HTTPS-protokollan kautta. Sen avulla voidaan esimerkiksi luoda laite ja hallita siihen liittyviä attribuutteja, sekä kirjoittaa ja lukea dataa. API-rajapintaa voidaan tarkastella myös kirjautumalla sen puolelle sisään tunnuksilla, joilla on sen käyttöön oikeus. Ohjelmointirajapintaa voidaan käyttää usealla eri kielellä kuten esimerkiksi Javalla, Swiftillä sekä C#-kielellä. (Wapice 10.3.2017.)

2.5 CODESYS

CODESYS on teollisen automaation ohjelmointiin suunniteltu ohjelmisto, joka perustuu CODESYS Development Systemiin, joka on IEC 61131-3 -standardin mukainen ohjelmointityökalu. CODESYS-järjestelmää käytetään maailmanlaajuisesti useissa eri ohjelmoitavissa logiikoissa. (CODESYS [Viitattu 16.9.2018].)

CODESYS tukee kaikkia IEC 61131-3 -standardin määrittelemiä ohjelmointikieliä, jotka ovat Structured Text (ST), Function Block Diagram (FBD), Ladder (LD), Inssturction List (IL) sekä Sequential Function Chart (SFC). CODESYS-ohjelmistossa luotua ohjelmakoodia voidaan myös käyttää online-tilassa, joka mahdollistaa virheiden havaitsemisen aikaisessa vaiheessa, sekä helpottaa testausinsinöörin työtä koodia testattaessa. (Epec [Viitattu 16.9.2018].)

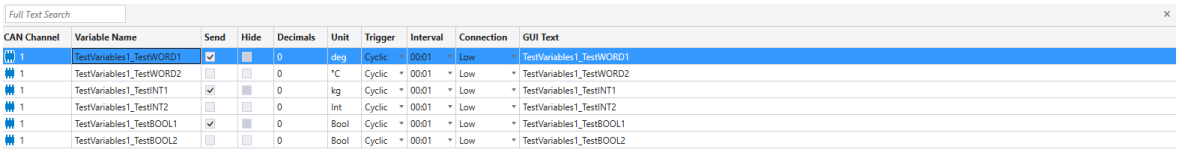
2.6 MultiTool

MultiTool on Epec Oy:n tarjoama konfigurointisovellus, joka luo laitteelle koodipohjan CODESYS-ohjelmaan. MultiToolissa pystytään tekemään asetukset väyläviesteille, joilla laitteet kommunikoivat keskenään, väylähierarkialle, tulojen ja lähtöjen tyypeille ja laitteiden parametreille. Lisäksi voidaan tehdä GlobE-konfigurointi. Kun nämä asiat voidaan tehdä siihen räätälöidyllä työkalulla, vältetään laitetta manuaalisesti konfiguroitaessa syntyvät virheet. Sovellus on saatavilla Windows-käyttöjärjestelmää käyttäville laitteille. Luodut konfiguraatiot siirretään CODESYS-projektiin, kun se luodaan MultiToolin kautta. (Epec 2018a.)

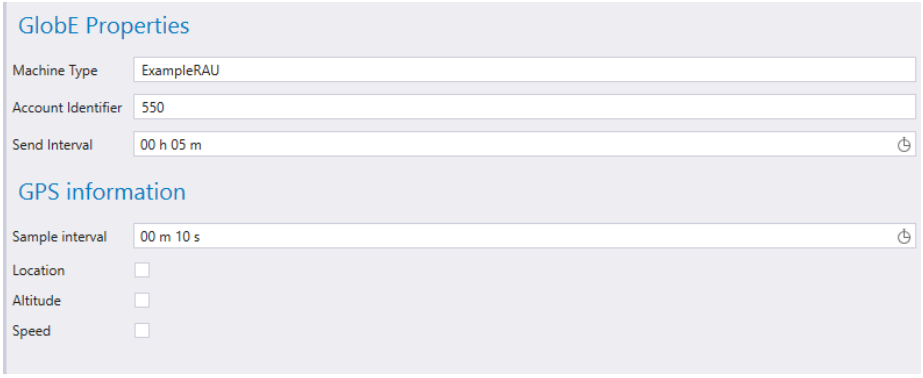
Väylää konfiguroitaessa voidaan MultiToolissa määritellä esimerkiksi tunniste, jolla väyläviestejä lähetetään ja vastaanotetaan. Lisäksi voidaan myös määrittää väyläviestit jotka lähetetään ja vastaanotetaan, lähetysintervalli sekä asema väylähierarkiassa. Laitteen asema määrittää, antaako kyseenomainen laite muille käskyn siirtyä toiminnalliseen tilaan vai vastaanotetaanko käynnistymiskäsky joltain muulta laitteelta. (Epec 2018a.)

Epec Oy:n ohjausyksiköissä on tulo- ja lähtöpinnejä, joita voidaan konfiguroida eri tiloihin. Esimerkiksi tuloista osa voidaan asettaa pulssi- tai analogiatuloksi, digitaali- tai analogiatuloksi sekä digitaali- tai pulssituloksi. Lähtöpinnejä voidaan asettaa digitaali- tai PWM-lähdöiksi. Lisäksi ohjausyksiköillä on myös referenssijännitelähtöjä, joita ei tarvitse erikseen konfiguroida. (Epec 2018a.)

GlobE-palveluun voidaan tehdä konfiguraatiot esimerkiksi muuttujista, joiden tila halutaan lähettää, yrityksestä, jonka alla laite näkyy portaalissa, sekä nimi, jolla laite näytetään. Lähetettäviin muuttujiin voidaan määrittää, kuinka usein muuttujan arvo otetaan talteen, millä nimellä se näytetään portaalissa, sekä kuinka usein arvot kollektiiviseesti lähetetään portaaliin. Näiden lisäksi voidaan myös määrittää GPS-asetuksia, kuten esimerkiksi lähetysintervalli. (Epec 2018a.)



Kuva 3. GlobE-portaaliin menevät muuttujat MultiToolissa



Kuva 4. Etäohjauslaitteen GlobE-konfigurointiin liittyvät ominaisuudet

2.7 Robot Framework

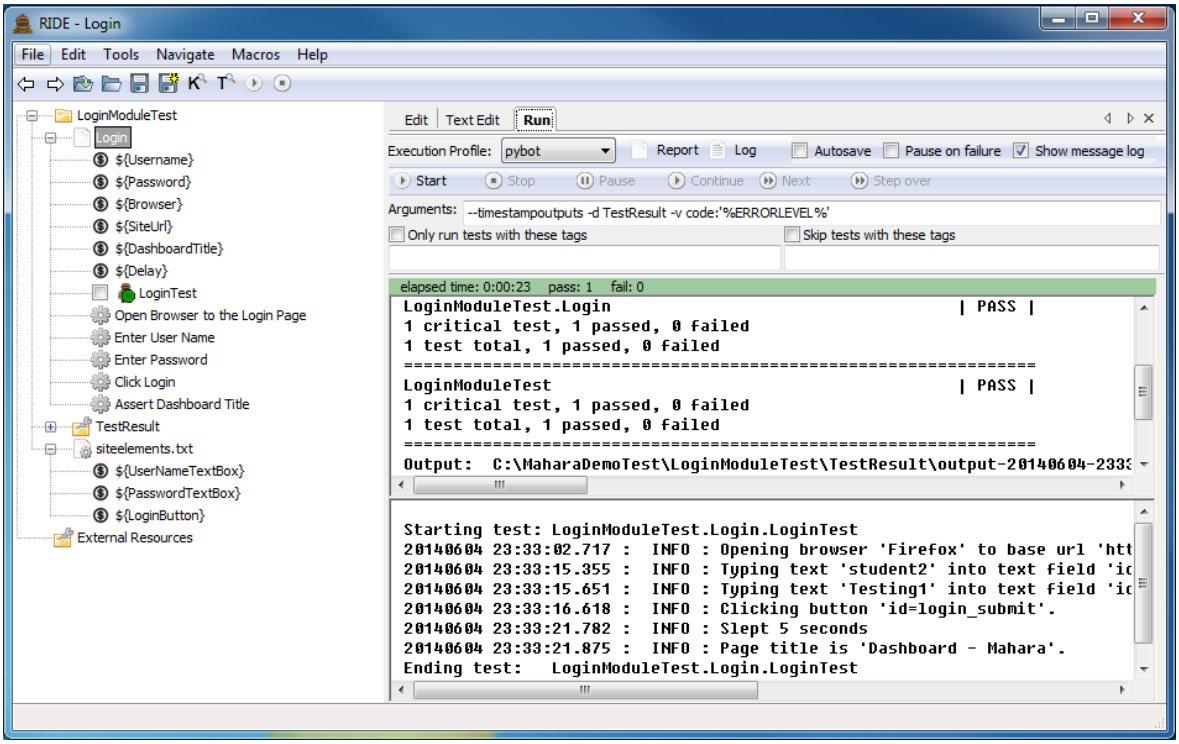
Robot Framework on yleiskäyttöinen, avoimen lähdekoodin kehys testiautomaation luomiseen, se on laajennettavissa kirjastoilla. Robot Framework on alkujaan Python-pohjainen, mutta sitä voidaan myös ohjelmoida Javalla. Robot Frameworkia käytettäessä käyttäjä antaa avainsanoja, joita käytetään implementoitujen kirjastojen funktioissa, minkä seurauksena Robot Frameworkin peruskäyttö ei vaadi laajaa ohjelmointikokemusta. Se on alkujaan luotu Nokia Networksin käyttöön, mutta on nykyään Robot Frameworks Foundationin sponsoroima. (Robot Framework [Viitattu 31.8.2018].)

2.7.1 Ohjelmakirjastot

Robot Frameworkiin on saatavilla useita eri ohjelmakirjastoja esimerkiksi verkkosivujen, tietokantojen tai mobiiliapplikaatioiden testaamiseen. Robot Frameworkissa on myös lukuisia sisäisiä kirjastoja perustoiminnallisuuksien suorittamiseen. (Robot Framework [Viitattu 1.9.2018].)

Selenium-kirjasto on verkkotestauskirjasto Robot Frameworkiin. Kirjasto sisältää Selenium-työkalun sisäisesti ja se tukee Python 2.7-versiota, sekä Python 3.4-versiota tai uudempaa. Tällä hetkellä Selenium-työkalu eikä -kirjasto tue IronPythonia. (Github [Viitattu 23.9.2018].)

Selenium-kirjaston sisältämät avainsanat ovat alhaista tasoa ja tarvitsevat siksi usein ulkopuolisen argumentin toimiakseen. On siis suositeltavaa käyttää Robot Frameworkin korkeamman tason avainsanoja, jotka sisältävät Selenium-kirjaston avainsanoja. Kirjaston avainsanoilla pystyy toteuttamassa muun muassa tekstin syöttöä tekstikenttään verkkosivulla, sivun otsikon tarkastelua, napin valintaa sekä selaimen avauksen. (Github [Viitattu 23.9.2018].)



Kuva 5. Testien suorittamista RIDE-käyttöliittymässä (Seleniummaster [Viitattu 27.9.2018].)

2.7.2 Loki testeistä

Robot Framework luo testeistä lokin, josta voidaan seurata muun muassa testien onnistumista, testeihin kulunutta aikaa eriteltynä testin yksittäisiin avainsanoihin,

sekä testien sisältö avainsanakohtaisesti. Jos testi epäonnistuu, Selenium ottaa tilanteesta kuvankaappauksen, josta voidaan myöhemmin tarkastella, mikä aiheutti testin epäonnistumisen. (Robot Framework [Viitattu 17.9.2018].)

2.8 C#

C# on Microsoftin sisällä kehitetty kieli, jonka ensimmäinen laajempi jakelu tapahtui heinäkuussa 2000. Kielen tärkeimmät kehittäjät ovat Anders Hejlsberg, Scott Wiltamuth ja Peter Golde. C# on olio-orientoitunut kieli, mikä tähtää yksinkertaisuuteen, moderniuteen sekä yksinkertaisuuteen. C# muistuttaa syntaksiltaan C- ja C++-kieliä. Vaikkakin C#-kielellä ohjelmoidut ohjelmat käyttävät vain vähän prosessorin laskentatehoa ja muistia. Kieltä ei luotu kilpailemaan C-kielen kanssa tehokkuudessa. (Ecma 5.12.2017.)

2.9 Visual Studio

Visual Studio on Microsoftin tuottama IDE-ympäristö, jolla voidaan luoda ohjelmia mobiililaitteille, työpöytä- tai konsolisovelluksia eri käyttöjärjestelmille tai verkkosovelluksia. Visual Studiossa on laaja tuki eri ohjelmointikielille, kuten esimerkiksi C++-kielelle, C#-kielelle ja Visual Basicille, mutta myös kolmannen osapuolen ohjelmointikielille, kuten Javalle ja Pythonille. (Visual Studio [Viitattu 23.9.2018].)

Visual Studion ominaisuuksia hyödyntämällä voidaan luoda laajennuksia, pelejä ja sovelluksia millä tahansa kielellä ja sillä voidaan luoda myös graafisia käyttöliittymiä. Visual Studiota voi käyttää myös versionhallintaan, jolloin kaikki uusimmat versiot projekteista on aina henkilöstön käytettävissä. (Visual Studio [Viitattu 23.9.2018].)

2.9.1 Olio-ohjelmointi

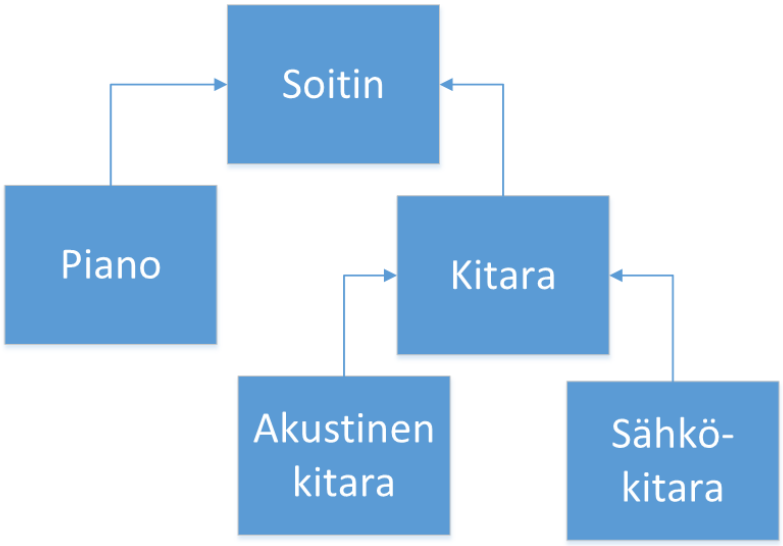
Olio-ohjelmointi viittaa ohjelmistosuunnitteluun, jossa luodaan tietoa ja funktioita sisältäviä luokkia. Luokat ovat uudelleen käytettäviä ohjelman osia, jota kutsumalla

saadaan olioita. Esimerkkinä voidaan käyttää käsiä: On olemassa oliot vasen ja oikea käsi, jotka molemmat voidaan määrittää luokasta käsi muuttamalla luokan muuttujien arvoja. Näin ollen samaa koodia ei tarvitse kirjoittaa useaan kertaan, vaan voidaan hyödyntää olemassa olevaa tietorakennetta. Tärkeimpiin periaatteisiin olio-ohjelmoinnissa kuuluvat luokka, perintä ja monikäyttöisyys. (Harle 2009.)

Luokka on määrittely, joka sisältää olion ominaisuuksia, menetelmiä ja attribuutteja. Luokka toimii laajennettavana pohjana olion luomiseen ja tuottaa alkuarvot ilmentymämuuttujille sekä rungot metodeille. Kaikki oliot, jotka on luotu samasta luokasta jakavat samat metodit, mutta sisältävät erilliset kopiot ilmentymämuuttujista. (Bruce 2002, 18.)

Tiedon periytyminen mahdollistaa ylemmän luokan tiedon käytön alemmalla luokkatasolla. Perinnässä luokka voidaan johtaa toisesta luokasta, jos kyseiset luokat jakavat samoja attribuutteja ja metodeja. Tällä mahdollistetaan poikkeuksien luonti sekä kustomoidun logiikan luonti olemassa oleville rakenteille. (Stackify [Viitattu 23.9.2018].)

Monikäyttöisyydellä tarkoitetaan tietyn operaattorin tai metodin sitomista erilaisiin toteutuksiin. Monikäyttöisyydellä parannetaan ohjelman laajennettavuutta ja voidaan lisätä uusia luokkia muokkaamatta aiempaa ohjelmakoodia ja kaikkia samaan luokkahierarkiaan kuuluvia käsitellään kuin yliluokkaa. (Harju 2005.)



Kuva 6. Esimerkkikuva olio-ohjelmoinnin periaatteesta

3 LÄHTÖTILANNE JA VAATIMUKSET

3.1 Tarve

Projektin tarve tuli ilmi kesän aikana käydyissä keskusteluissa, kun ilmeni, kuinka paljon aikaa IoT-palvelun testaaminen kuluttaa, kun se tehdään manuaalisesti. Tämän seurauksena alettiin miettiä vaihtoehtoista toimintatapaa, kuten testien automatisoimista. Epec Oy:llä oli entuudestaan osaamista testiautomaatioon, joten päätös testauksen automatisoinnista oli selkeä. Tämän jälkeen käytiin läpi, millä tavalla projekti toteutetaan ja kuka sen toteuttaisi.

3.2 Projektin aloitus

Tutkimusprojekti aloitettiin käymällä asiaa koskevan henkilöstön kanssa läpi aloituspalaveri, jossa sovittiin tutkimuksen ohjaaja, keskusteltiin käytettävistä toimintatavoista, määritettiin tutkimusta tukevat henkilöt sekä asetettiin aikatavoitteita. Aloituspalaverin jälkeen aloitettiin ohjelmistojen kartoittaminen, millä työ tultaisiin tekemään, ja millä saataisiin paras mahdollinen lopputulos.

3.3 Vaatimukset

Automaattitestien vaatimuksiksi asetettiin helppokäyttöisyys sekä mahdollisimman kattavat testausmahdollisuudet. Regressiotesteillä pitää pystyä todentamaan palvelimen perustoiminnallisuuksien toiminta järjestelmäpäivityksen jälkeen, ja suorituskykytesteillä pitää pystyä todentamaan, että palvelun käyttö pysyy edelleen miellyttävänä, kun palvelimella on enemmän laitteita, jotka lähettävät toistuvasti dataa.

4 OHJELMAN SUUNNITTELU

4.1 Ohjelmistojen valinta

Robot Frameworkin valinta ympäristöksi, jolla testausautomaatio suoritetaan, tapahtui Epec Oy:llä jo olevan asiantuntijuuden vuoksi. Epec Oy hyödyntää Robot Frameworkia muissa projekteissa, joten ammattimaisten neuvojen saaminen projektiin on yksinkertaista. Lisäksi Epec Oy:llä ei ole syytä yksittäisen projektin vuoksi harkita muita ympäristöjä ja kuluttaa resursseja sen käyttöönottoon. Projektissa pystyttiin hyödyntämään Robot Frameworkia Selenium-kirjaston kanssa, näin on mahdollista luoda automaattitestejä verkkosivulle. Selenium-kirjaston ominaisuudet ja Robot Frameworkin sisäiset kirjastot kattavat kaiken, mitä testejä suunniteltaessa tarvitaan. Lisäksi testejä on mahdollista laajentaa muilla Robot Frameworkiin saatavilla kirjastoilla.

Visual Studio valikoitui tutkimuksen suorittajan aiemman käyttökokemuksen perusteella sekä laajojen ominaisuuksien vuoksi. Tämän lisäksi Visual Studiossa on laaja tuki sekä ympäristön helppokäyttötoiminnot, mitkä tekevät ohjelman luomisesta huomattavasti helpompaa. Visual Studion laaja käyttö kansainvälisesti takaa sen, että siihen on saatavilla käyttöä helpottavaa apua runsaasti.

4.2 Ohjelmointikielen valinta

Ohjelmointikieleksi, jolla ollaan yhteydessä IoT-palvelun GlobE ohjelmointirajapintaan, valikoitu C#. Tämä valinta perustuu tutkimuksen tekijän aiempaan käyttökokemukseen, joka helpottaa työn aloittamista ja lisäksi C#-kielen laajaan käyttötukeen, mikä johtuu kielen yleisyydestä. C#-kielen valitsemisen ohjelmointikieleksi mahdollisti se, että Wapice tarjoaa IoT-Ticketille rajapinnan C#-kielelle.

4.3 Testien alustus

Robot Frameworkin automaattitestejä tehdessä huomattiin, että jos halutaan testata paremmin käyttäjäkokemusta, oli aluksi ohjelmoitava Epec Oy:n etäohjausyksikkö. Tällä etäohjausyksiköllä olisi otettava yhteys Epec Oy:n testipalvelimeen ja lähetettävä sinne arvoja. C#-ohjelmointirajapinnan kautta luodulla laitteella ei voitaisi testata esimerkiksi laitteen aktivointia, mikä on keskeinen ominaisuus kyseisessä IoT-palvelussa.

Jotteivat suoritettavat testit vaikuttaisi muihin olemassa oleviin testeihin, kuten esimerkiksi kirjaston testeihin, Epec Oy:n IoT-palvelun testipalvelimelle luotiin oma yritys testejä varten. Tähän yritykseen luotiin myös käyttäjä, jolla oli käyttöoikeudet vain kyseiseen sijaintiin, eikä testejä konfiguroitaessa ja kokeiltaessa pystytä rikkomaan muiden testien toimivuutta.

5 ETÄOHJAUSYKSIKÖN OHJELMOINTI

Ohjelmointi aloitettiin luomalla MultiTool-projekti Epec Oy:n etäohjausyksikölle, jonka jälkeen laitteelle luotiin ohjelma CODESYS-ohjelmistolla.

Kuva 7. Epec Oy:n etäohjausyksikkö 6100-21 (Epec 2018b.)

5.1.1 Laitteen konfigurointi

Etäohjauslaitteen ohjelman luominen aloitettiin alustamalla laitteelle arvoja MultiToolissa, kuten yritys, jonka alle laite tulisi sekä nimi, jolla laite näkyisi portaalissa. Lisäksi määritettiin muuttujia, jotka lähetettäisiin palvelimelle, sekä parametreja, joihin voitaisiin tehdä muutoksia palvelusta, ja joiden muutos voitaisiin näin nähdä laitteella.

5.1.2 CODESYS-ohjelma

Kun tarvittavat määritykset laitteelle oli tehty, voitiin luoda laitteelle CODESYS-projekti, johon MultiToolissa tehdyt alustukset tulevat näkyviin ja kaikki tarvittavat kirjastot ovat jo käytössä. Yksinkertaisen CODESYS-projektin luominen GlobE-palvelun käyttöön on yksinkertaista tehdä helppokäyttöisten ja selkeiden kirjastojen

vuoksi. Ohjelmakoodiin laitettiin lukugeneraattori, joka generoi mahdollisimman erilaisia arvoja. Tällä saatiin IoT-portaaliin lähetettäviä arvoja simuloitua ja vaihdettua, minkä lisäksi alustettiin kirjasto, joka suorittaa yhteydenoton GlobE-palveluun.

5.1.3 Konfigurointitiedostot

Ohjelmakoodin lisäksi on tarkistettava että laitteelta löytyy kaikki tarvittavat konfigurointitiedostot. Tiedostot, joita laitteelta on löydyttävä, ovat muun muassa konfigurointitiedosto palvelimelle. Siinä määritetään myös yhteyden tyyppi. Tämä tarkoittaa, että tiedostossa tehdään valinta, käytetäänkö yhdistämiseen etäohjauslaitteen Ethernetiä vai luodaanko yhteys modeemin yli hyödyntäen laitteeseen tulevaa SIM-korttia.

Yhteyskonfiguraation lisäksi laitteelle laitetaan MultiToolin luomat tiedostot, joista toinen on listaus muuttujista, joita lähetetään palvelimelle ja toinen listaus parametreista, joita laitteella on, ja joita halutaan näyttää portaalissa. Lisäksi valinnaisena tiedostona voidaan lähettää laitteen tapahtumiin liittyviä tiedostoja, jotka mahdollistavat tapahtumien näyttämisen portaalissa.

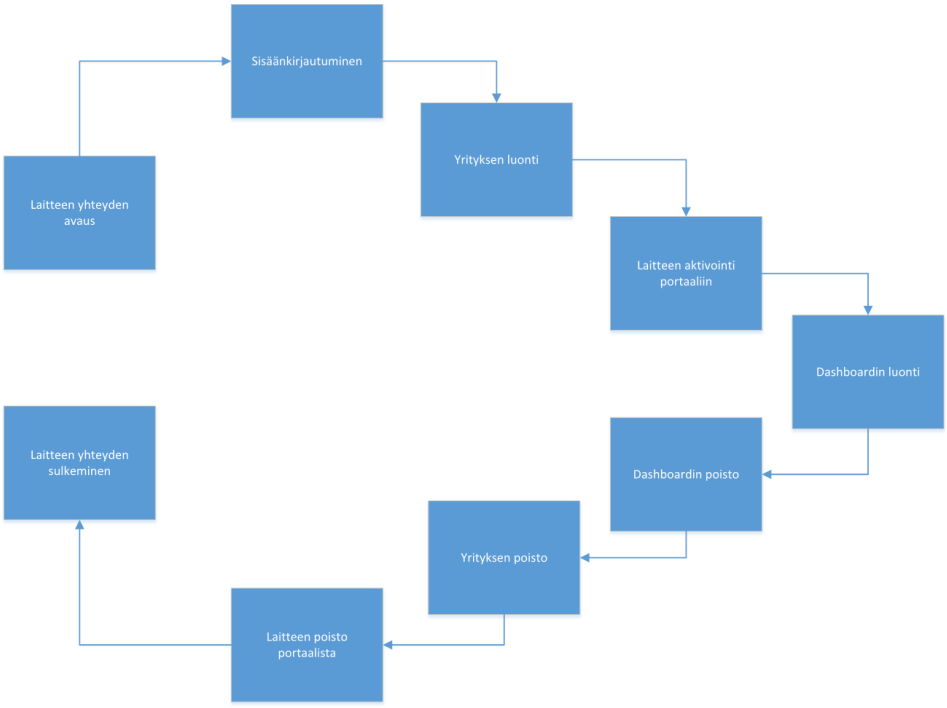
6 REGRESSIOTESTIT

6.1 Rakenne

Jokainen testi sisältää useita muuttujia, joita käyttäjä voi määritellä. Näillä muuttujilla voidaan määrittää objekteja, joita Robot Framework etsii sivustolta tai mitä tunnuksia käytetään sisäänkirjautumiseen. Testien kannalta oleellisia muuttujia ovat muun muassa:

- Kirjautumistunnukset, joissa määritetään myös salasana jolla sisään kirjautuminen ei toimi virheellistä kirjautumista testattaessa.
- Verkkosivujen osoitteet, joissa testien aikana ollaan. Näitä käytettään testeissä tarkasteluun, toimiiko portaali odotetulla tavalla.
- Verkkosivujen otsikot, käytetään kun tarkastellaan, ollaanko halutulla sivulla.
- Laitteen nimi, jonka tietoja hyödynnetään testeissä.

Jokaisen testin kohdatessa poikkeuksen tarkastelussa, kyseinen testi epäonnistuu ja tästä otetaan kuva, mikä esitetään Robot Frameworkin testilokissa. Muiden testien jälkeen oletusarvoisena lähtökohtana on, että Robot Frameworkin avaamassa selainikkunassa ollaan palvelun aloitussivulla, mihin ohjaudutaan kirjautumisen jälkeen.



Kuva 8. Regressiotestin toimintaperiaate, kun kaikki vaiheet suoritetaan

6.1.1 Sisäänkirjautumisen testaus

Sisäänkirjautumisen testillä testataan sisäänkirjautumisyritys ensiksi väärällä salasanalla ja sen jälkeen oikeilla tunnuksilla. Aluksi tarkistetaan, että kirjautumissivu on oikea ja sivuston otsikko täsmää. Seuraavaksi syötetään käyttäjänimi ja salasana niille tarkoitettuihin kenttiin ja painetaan kirjautumisnappia.

Testin testatessa kirjautumista väärillä tunnuksilla tarkastetaan, että sivun osoite muuttuu osoitteeksi, joka viittaa kirjautumisen epäonnistumiseen. Tämän lisäksi kirjautumiskenttien alle tulee punainen laatikko, joka sisältää tekstin ”Incorrect username or password”, mikä viittaa vääriin tunnuksiin. Näiden asioiden ilmetessä testi on hyväksytysti suoritettu.

Oikeilla tunnuksilla kirjautumista testattaessa tarkastellaan, että sivun osoite muuttuu portaalin etusivuksi, sekä sivun otsikko muuttuu muotoon ”Etunimi @ Epec GlobE”, Etunimen ollessa muuttuja testissä, se voidaan vaihtaa käyttäjän haluamaksi.

6.1.2 Yrityksen luonti ja poisto portaalissa

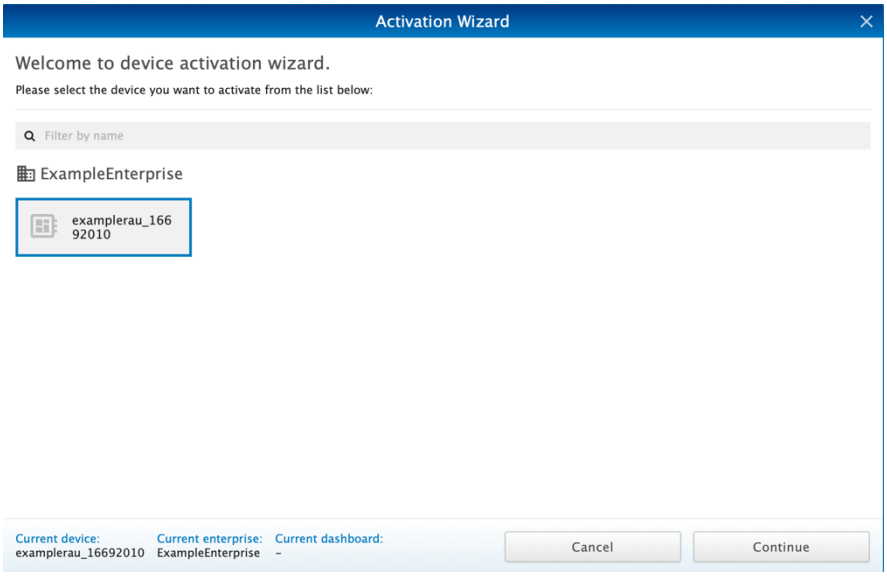
Yrityksen luonti alkaa, kun valitaan resurssiselaimesta (Resource Browser) yritys, jonka alle uusi yritys luodaan. Tämän jälkeen siirrytään hallintatilaan (Management Mode) ja tarkastetaan, että selailutilasta ollaan poistuttu. Hallintatilassa painetaan Create Enterprise -painiketta, jonka jälkeen odotetaan, että yrityksenluonti-ikkuna aukeaa. Tässä ikkunassa uudelle yritykselle annetaan nimi, ja painetaan OK-painiketta. Ikkunassa olisi myös mahdollista määrittää Dashboard-mallipohjia, mutta tämänhetkisessä versiossa testeistä sitä ei toteuteta. Kun OK-painiketta on painettu, odotetaan portaalista tulevaa ilmoitusta yrityksen onnistuneesta luonnista, jonka jälkeen siirrytään pois hallintatilasta ja testi menee suoritetuksi.

Yrityksen poisto alkaa valitsemalla resurssiselaimesta yritys, mikä aiotaan poistaa, minkä jälkeen siirrytään hallintatilaan. Hallintatilassa painetaan Delete Enterprise -painiketta, minkä jälkeen odotetaan, että yrityksen poistoon liittyvä varmennusikkuna aukeaa. Tässä ikkunassa painetaan Confirm-painiketta, minkä jälkeen odotetaan, että portaali antaa ilmoituksen onnistuneesta yrityksen poistosta. Tämän jälkeen siirrytään pois hallintatilasta ja testi menee suoritetuksi.

6.1.3 Laitteen aktivointi

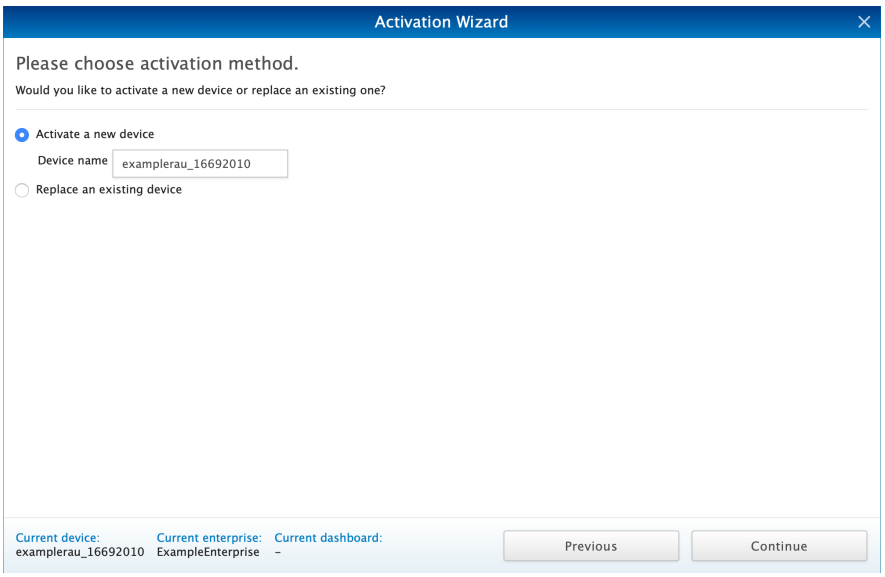
Kun testataan laitteen aktivointia, työssä käytetyn etäohjauslaitteen on alkuarvoisesti oltava jo näkyvissä portaalissa uutena laitteena. Jokaisessa vaiheessa seuraavassa aktivointiprosessin kohdassa tarkistetaan, että edellinen avoinna ollut kohta on sulkeutunut ja senhetkinen näkyvä ikkuna on se, jossa aktivoinnin tässä vaiheessa kuuluu olla.

Laitteen aktivointi -testi aloitetaan käynnistämällä portaalissa laitteiden aktivointiprosessi, jonka ensimmäisessä ikkunassa aukeaa näkymä aktivointia odottavista laitteista. Tässä ikkunassa tarkastetaan, onko laite aktivoitavien laitteiden joukossa. Laitteen löytyessä täältä, se valitaan ja siirrytään seuraavaan ikkunaan painamalla Continue-painiketta.



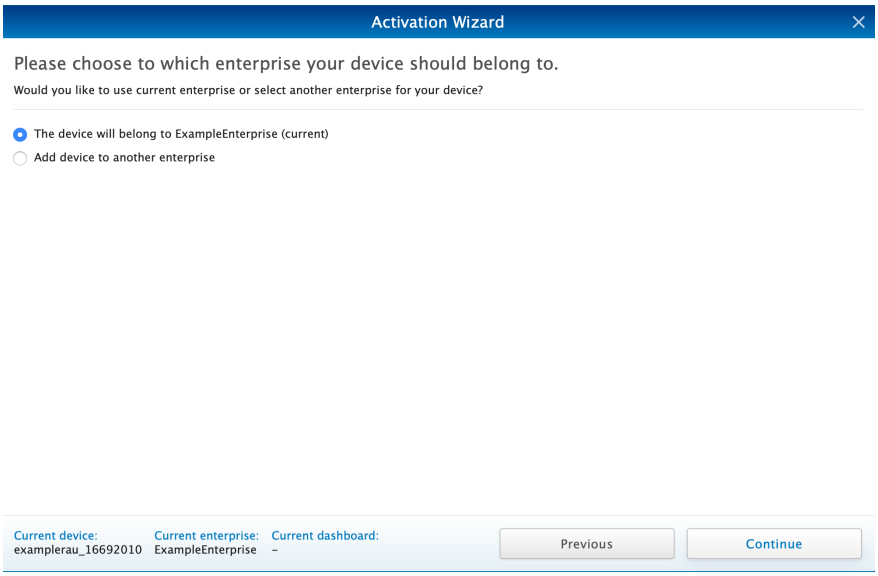
Kuva 9. Aktivointiprosessin ensimmäinen ikkuna

Seuraavassa ikkunassa aktivointiprosessissa katsotaan, että nimi, jolla laite halutaan lisätä, on oikea. Tässä ikkunassa voitaisiin laitteen nimeä halutessa vaihtaa, mutta testi ei sisällä tätä vaihtoehtoa. Vaihtoehtona on myös korvata jo olemassa oleva laite uudella aktivoimattomalla laitteella. Testissä tarkastellaan vain, että laitteen nimi täsmää parametriin määritettyyn nimeen. Jos nimet täsmäävät, siirrytään seuraavaan kohtaan painamalla Continue-painiketta.



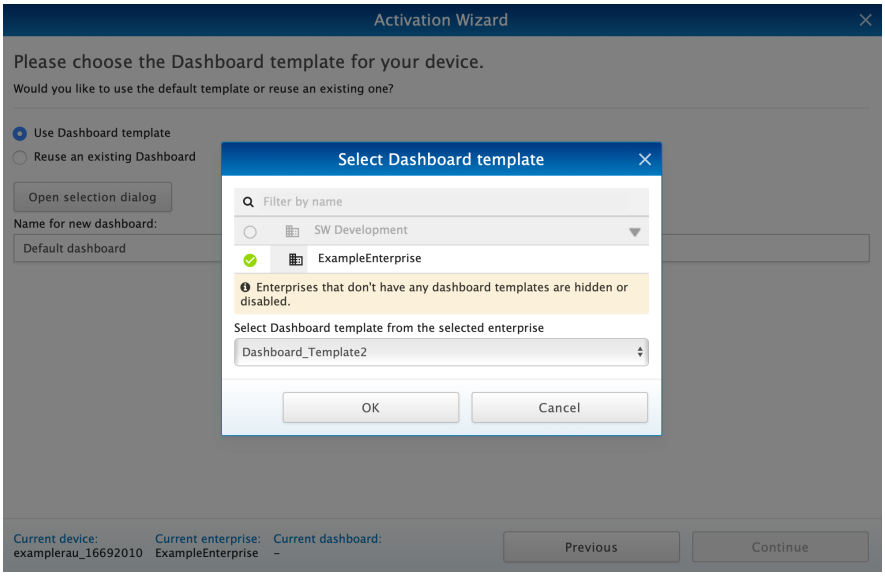
Kuva 10. Aktivointiprosessin toinen ikkuna

Seuraava ikkuna sisältää yritysvalinnan, jossa voidaan lisätä laite siihen yritykseen, jonka alle se on tullut näkyviin uutena laitteena, tai voidaan siirtää laite jonkin toisen yrityksen alle. Tässä testi tarkastaa, että aktivointi-ikkunassa näkyvä yritys täsmää yritykseen, joka on määritetty testiin parametrina, eikä siten vaihda laitetta toisen yrityksen alle. Seuraavaan ikkunaan siirrytään painamalla Continue-painiketta.



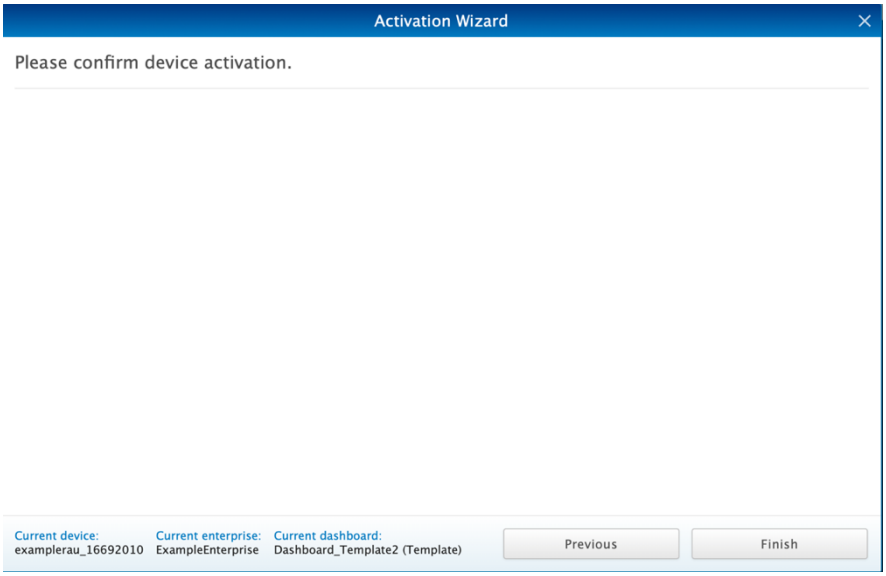
Kuva 11. Aktivointiprosessin kolmas ikkuna

Seuraavassa ikkunassa voidaan joko luoda laitteelle Dashboard jonkin yrityksen mallipohjasta tai ottaa laitteelle Dashboard jo olemassa olevalta laitteelta. Tässä testissä Dashboard otetaan yrityksestä, joten ensin tarkastetaan, että yrityksen mallipohja on valittu vaihtoehto. Tämän jälkeen avataan dialogi, josta valitaan parametrilla määritetty yritys ja tarkastetaan, että vetovalikossa on valittuna testiin määritetyn mallipohjan nimi. Tämän tarkastuksen jälkeen, mikäli valinta on oikea, dialogi suljetaan painamalla OK-painiketta. Seuraavaksi tarkastetaan, että dialogi on suljettu ja painetaan Continue-painiketta ja siirrytään seuraavaan vaiheeseen.



Kuva 12. Aktivointiprosessin neljäs ikkuna

Viimeinen ikkuna aktivointiprosessissa on tietojen tarkistamista ja varmistamista varten. Tässä kohtaa testissä tarkastetaan vielä kerran, että ikkunan alalaidassa olevat tiedot täsmäävät parametreihin, joita testiin on annettu. Kun nämä tarkastelut on tehty, painetaan Finish-painiketta, jonka jälkeen odotetaan, että portaalin oikeaan yläreunaan tulee ilmoitus aktivoinnin suorittamisesta. Kun ilmoitus on saatu, testi menee hyväksytysti läpi.



Kuva 13. Aktivointiprosessin viimeinen ikkuna

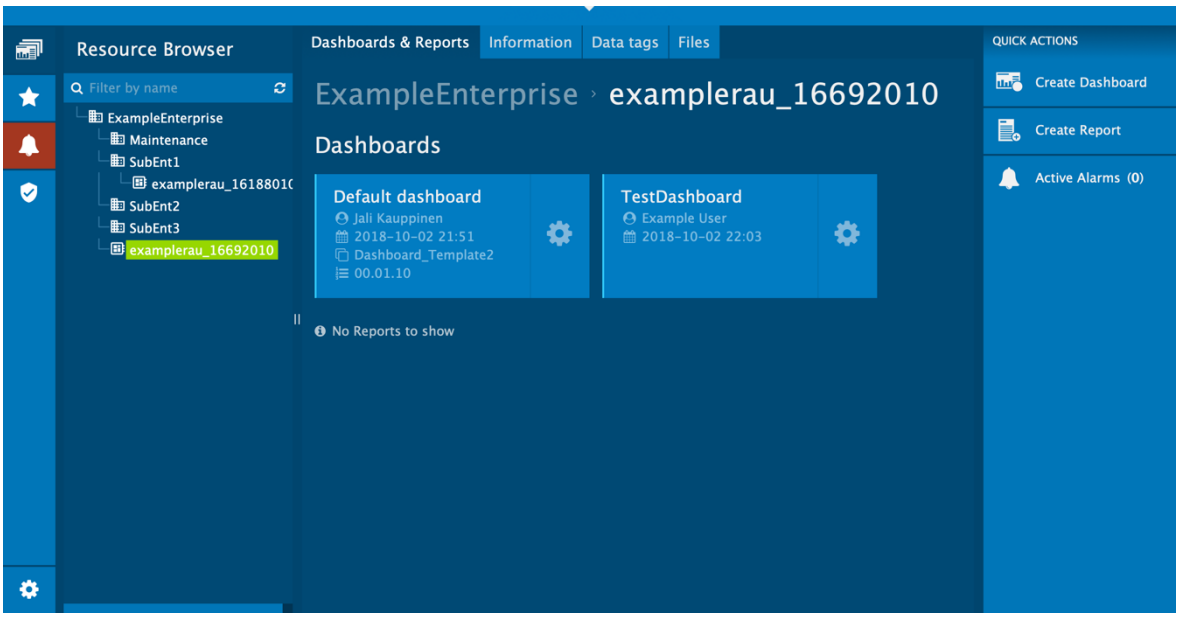
6.1.4 Dashboardin luominen ja poisto

Dashboardin luominen aloitetaan tarkastamalla, että resurssiselain sisältää halutun laitteen. Kun laite on valittu, voidaan selaustilassa valita pikatoiminnoista Dashboardin luonti Create Dashboard -painikkeella. Kun painiketta on painettu, avautuu ikkuna, jossa syötetään Dashboardille parametrilla määritetty nimi, sekä valitaan vetovalikosta layout-tyyli, jota Dashboardille halutaan käyttää. Tässä testissä valitaan tyhjä layout, joten Dashboardille ei ole valmiina mitään elementtejä.

Kun nimen syöttö ja layout-tyylin valinta on tarkastettu, painetaan Create-painiketta, jonka jälkeen luotu Dashboard aukeaa. Testi tarkastaa, että Dashboard on auennut, jonka jälkeen vedetään Progress bar -elementti tyhjälle Dashboardille. Tämän jälkeen tarkastetaan, että elementti on nähtävillä Dashboardilla, minkä jälkeen voidaan painaa Save Dashboard -painiketta. Kun tallennusikoni on poistunut ruudusta, voidaan painaa Back-painiketta, minkä kautta siirrytään pois Dashboardin muokkaustilasta. Kun tästä tilasta on poistuttu, testi merkitään onnistuneeksi.

Dashboardin poisto portaalista tapahtuu valitsemalla selaustilassa laite, jolta halutaan Dashboard poistaa. Kun laite on valittu onnistuneesti, painetaan poistettavan Dashboardin nimen vieressä olevaa hammaspyörää, jonka alta avautuu valikko,

jossa painetaan Delete-painiketta. Tästä avautuu Dashboardin poiston varmistusikkuna, jossa painetaan Delete-painiketta. Tämän jälkeen odotetaan, että portaalista tulee ilmoitus, että Dashboard on poistettu onnistuneesti ja testi voidaan merkitä suoritetuksi.

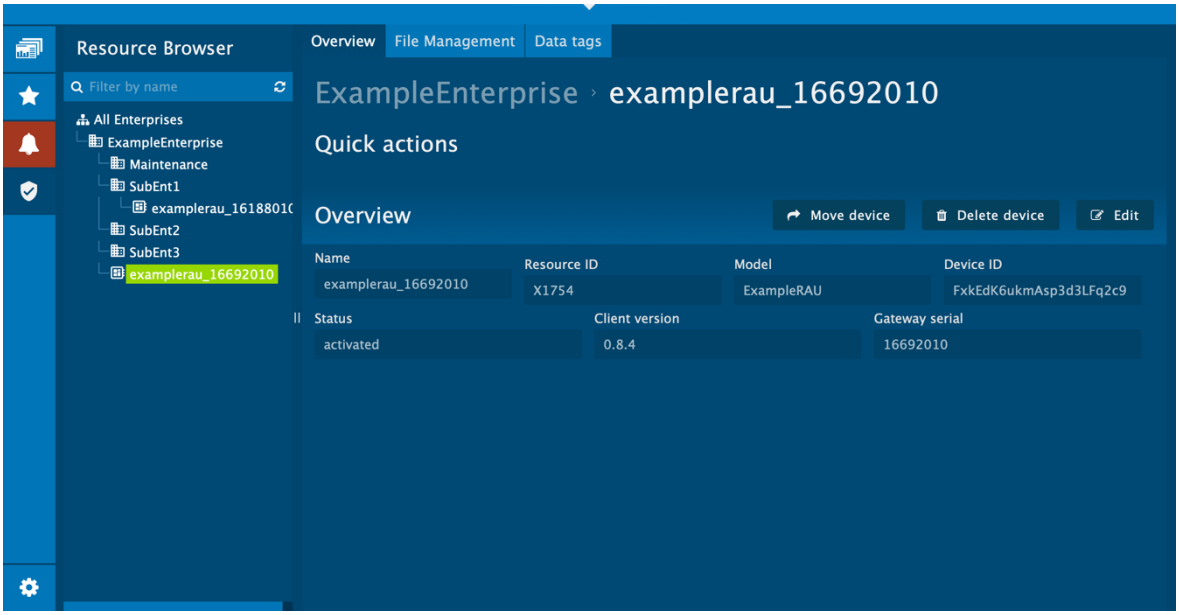


Kuva 14. Laitenäkymä selaustilassa

6.1.5 Laitteen poisto portaalista

Laitteen poistaminen alkaa tarkastamalla, että laite löydetään resurssiselaimesta, jonka jälkeen laite valitaan, ja siirrytään hallintatilaan painamalla hallintatilan painiketta. Testi tarkastaa, että tässä vaiheessa on siirrytty hallintatilaan ja poistuttu laitteiden selaustilasta. Kun tarkistus on tehty, painetaan Delete device -painiketta, jonka jälkeen tarkastetaan, että laitteen poistamiseen liittyvä varmistusikkuna aukeaa.

Varmistusikkunan auettua painetaan ikkunassa olevaa Confirm-painiketta, jonka jälkeen tarkastetaan, että portaalissa annetaan ilmoitus laitteen poistamisen onnistumisesta sekä varmistetaan resurssiselaimesta, että laitteen nimi ei ole siellä enää näkyvissä. Näiden tarkastusten jälkeen poistutaan hallintatilasta takaisin selaustilaan. Kun on tarkastettu, että hallintatila ei ole enää näkyvissä, testi menee suoritetuksi.



Kuva 15. Laitenäkymä hallintatilassa

6.2 Ongelmakohdat

Robot Frameworkin ensimmäisiä testejä tehdessä huomattiin, että aika ajoin testipalvein toimi hitaasti, mikä aiheutti testien epäonnistumista aikakatkon vuoksi. Näin ollen testeihin oli lisättävä viivettä, mutta sen aiheuttamana Robot Frameworkista tuli virheilmoitus, joka esti viiveen lisäämistä tarpeellista määrää. Tämän vuoksi testien rakenne muutettiin alkuperäisestä niin, että jokaisen testin yhteydessä ensimmäinen asia, mikä tehtiin, oli sisäänkirjautuminen. Alkuperäisissä testeissä sisäänkirjautuminen tehtiin vain kerran, jonka jälkeen suoritettiin muut testit heti sen perään sulkematta selainta. Testirakenteen muutoksen yhteydessä selain avataan jokaisen testin alussa ja suljetaan testien lopussa, mikä vaikuttaa hidastamalla suoritusaikaa.

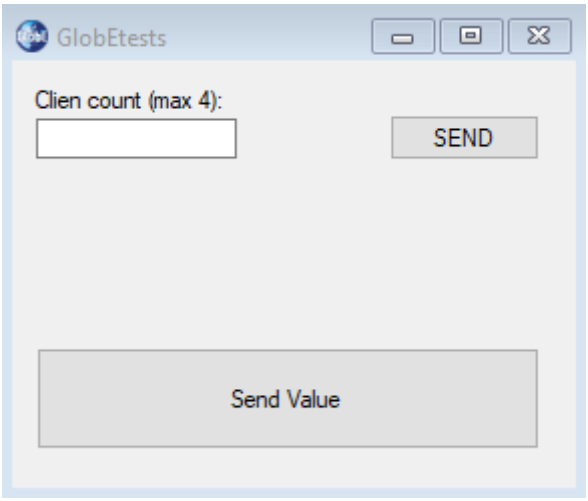
Aikaisemman rakenteen ongelmana oli myös heikompi toimintavarmuus. Yhden testin epäonnistuminen tarkoitti sitä, että myös loput testit tulivat epäonnistumaan sen jälkeen toteutuksen vuoksi. Tämä tarkoittaa siis, että selaimen avaaminen ja sulkeminen jokaisessa testissä erikseen ei välttämättä menetä aikasäästöä, koska testien virheherkkyys on suurempi, jos selainikkunaa ei suljeta testien välissä.

7 KUORMITUSTESTIT

7.1 Graafinen käyttöliittymä

Graafisen käyttöliittymän piti olla yksinkertainen ja helposti ymmärrettävä. Ensimmäisessä revisiossa oleva GUI onkin todella yksinkertainen, koska tutkimusta tehdessä katsottiin, että tarvittavat tiedot esimerkiksi palvelimelle lähetettävien muuttujien muutoksesta nähdään GlobE-portaalissa. Näin datan näkeminen sovelluksen käyttöliittymästä jää kehitysehdotukseksi seuraaviin versioihin. Esitettiin myös toive mahdollisuudesta päättää, minkälaista arvoa GlobE vastaanottaisi, mutta tämäkin jätettiin aikataulun vuoksi myöhempää kehitystä varten.

Sovelluksen käyttöliittymässä on mahdollista määrittää, kuinka montaa asiakasta (client) sovelluksen käyttäjä haluaa käyttää. Asiakasmäärä määrittää, kuinka monta laitetta GlobE:n luodaan. Kun laitteiden määrä on määritetty, voidaan painiketta painamalla luoda laitteet palvelimelle ja toista painiketta painamalla voidaan aloittaa datan lähetys kyseisille laitteille. Projektissa rajoitettiin asiakkaiden määrä neljään, koska jokaiselle asiakkaalle luotiin omat tunnuksensa. Jos kuitenkin testaustyökalu otetaan laajempaan käyttöön, on mahdollista pyytää esimerkiksi Wapicea luomaan suurempi määrä käyttäjiä, joita tässä voitaisiin hyödyntää.

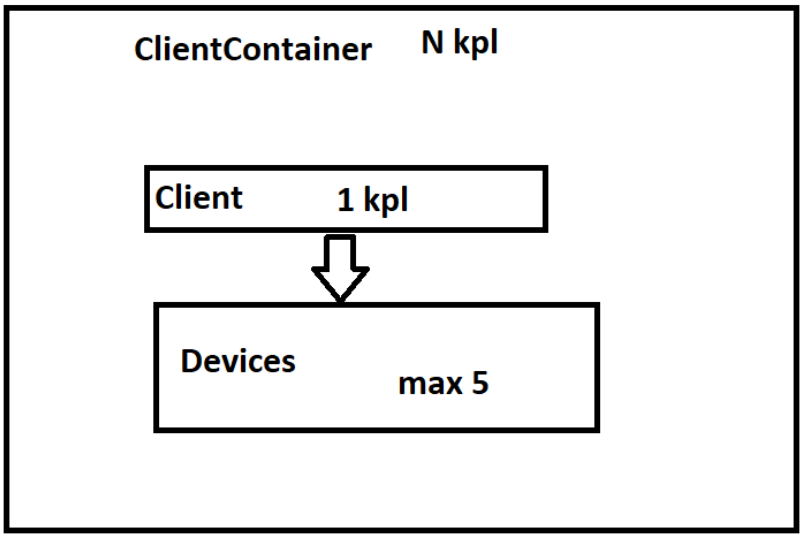


Kuva 16. Kuormitustestin graafinen käyttöliittymä

7.2 Rakenne

Kun ohjelma käynnistetään, se luo ensimmäisenä käyttöliittymän latauksen yhteydessä neljä muuttujaa, jotka sisältävät attribuutteja kuten arvo, yksikkö ja muuttujan nimi. Nämä muuttujat ovat portaaliin simuloidulta laitteelta lähteviä tietoja. Lisäksi tässä yhteydessä asetetaan myös salasana käyttäjille, missä päädyttiin käyttämään kaikilla samaa salasanaa ohjelmoinnin yksinkertaistamiseksi. Alustuksen yhteydessä asetetaan myös muita arvoja, jotka ovat ohjelman toiminnan kannalta välttämättömiä.

Alustusten ja käyttöliittymän latauksen jälkeen voidaan suorittaa asiakkaiden lisääminen. Kun käyttöliittymässä on määritetty se määrä, mikä asiakkaita halutaan, painetaan Send-painiketta. Tämän jälkeen ohjelma suorittaa asiakkaanluonnin toteuttavan olion niin monta kertaa, kun käyttöliittymään on syötetty. Jokaisen suoritettavan kierroksen välillä käyttäjän viimeisenä merkkinä olevaan numeroon merkitään senhetkisten suoritusten määrä. Näin pystytään kiertämään rajoitetta liittyen asiakaskohtaiseen laitemäärään.



Kuva 17. Asiakasolion rakenne

Kun asiakkaan kirjautumistunnukset on annettu, lisätään viisi kappaletta näennäislaitteita, jotka sisältävät attribuutteja, kuten nimi, konetyyppi ja valmistava yritys. Laitteille on myös mahdollista lisätä haluttaessa omia tekstiattribuutteja. Kun laitteet on luotu, ne ovat nähtävissä portaalissa samalla tavalla kuin fyysisetkin laitteet.

Kun laitteet ovat näkyvissä portaalissa, niille saadaan näkyviin alustusvaiheessa määriteltyjä muuttujia. Näiden muuttujien lähettäminen tapahtuu painamalla Send Value -painiketta käyttöliittymässä. Kun painiketta on painettu, käynnistyy ajastin, joka lähettää näitä neljää muuttujaa 20 sekunnin välein portaaliin. Arvoa muutetaan jokaisella lähetyskerralla, mikä takaa laitteelle luotuun Dashboardiin arvojen vaihtelua.

7.3 Ongelmakohdat

Ohjelmoinnin alussa kohdattiin ongelmia suunnittelussa ja ohjelman rakenteessa. Ongelmat johtuivat kokemuksen puutteesta ja aiemman suunnittelukokemuksen vähäisyydestä.

Ohjelmaa tehdessä saatiin laitteet vaivattomasti näkymään portaalissa, mutta muuttujia lähetettäessä huomattiin, että sovellus kaatuu ja antaa virheilmoitusta. Tämän

aiheuttajana olivat lähetettävässä muuttujassa olevat attribuutit, joita Epec Oy:n IoT-portaaliin ei ole tarkoitus lähettää. Vikatilanne saatiin korjattua poistamalla ylimääräiset attribuutit.

Alkuperäisen suunnitelman mukaan käyttöliittymässä ei määritetty asiakkaiden määrää, vaan laitteiden määrä, mikä perustui oletukseen, että laitteiden määrää ei olisi rajoitettu asiakaskohtaisesti. Jokaisella asiakkaalla saa Wapicen REST Clientia käytettäessä kuitenkin olla vain viisi laitetta. Tämän seurauksena ohjelman rakenne muutettiin niin, että kun asiakas luodaan, se omistaa automaattisesti 5 laitetta, eikä laitteiden määrää erikseen määritettäisi.

Ongelmana kohdattiin myös, että kun palveluun on luotu laitteita, on ne poistettava käsin luomisen jälkeen. Tämä aiheuttaa runsaasti manuaalista työtä, mikäli palvelimeen luodaan runsaasti laitteita, koska vaihtoehtona ei ole tehdä monivalintaa, vaan laitteet on poistettava yksitellen.

8 JATKOKEHITYS

Projektin myötä toteutettuihin testeihin ja ohjelmistoihin on mahdollista tehdä jatkokehitystä. Robot Frameworkilla luotuihin testeihin voidaan luoda lisää tapauksia, jolloin testit koskisivat paremmin palvelun tarjontaa. Kun palveluun tulee uusia ominaisuuksia, on niille mahdollista lisätä omat testinsä.

Sovellukseen, joka lisää laitteita palveluun, on luotava lisää käyttäjiä, mikäli sen halutaan toimivan tarkoituksenmukaisemmin. Tähän sovellukseen olisi hyvä myös löytää jokin keino, millä voitaisiin poistaa laitteet automaattisesti palvelusta ja säästettäisiin myös siinä aikaa, sekä saataisiin vähennettyä tuottamatonta työtä.

Kuormitustestaus olisi myös mahdollista suorittaa jollain muulla, esimerkiksi kaupallisella ohjelmistolla. Tähän on saatavia useita erilaisia vaihtoehtoja, kuten esimerkiksi LoadView, jolla voidaan simuloida jopa tuhansien eri käyttäjien yhteydenottoa palvelun API-rajapintaan ja näin luoda runsas määrä palvelinta rasittavia pyyntöjä. LoadView-ohjelmistolla on myös mahdollista visualisoida testien tulokset ja näin ollen tehdä testien tulosten läpikäynnistä vaivattomampaa sekä selkeämpää.

9 POHDINTAA JA YHTEENVETO

Tämän työn tarkoituksena oli selvittä voidaanko Epec Oy:n Globen testausautomatisoida sen sijaan, että testit tehtäisiin käsin. Tämän ollessa mahdollista, voitiin luoda testejä, joilla voitaisiin testata GlobEn käyttöliittymää, sekä rasittaa palvelua simuloimalla sinne suuri määrä laitteita.

Selvitystyön tuloksena, sekä aiempaan tietoon ja asiantuntijuuteen pohjautuen päätettiin valita työskentely-ympäristöksi Robot Framework ja sen Selenium-kirjasto, sekä C#-ohjelmointikieli ja Microsoftin Visual Studio. Laitteita lisäävässä ohjelmassa hyödynnettiin Wapicen tarjoamaa REST-clientia, jolla saatiin itsetehdyn sovelluksen sekä GlobEn välille yhteys. Robot Frameworkin testejä luodessa hyödynnettiin Epec Oy:llä jo olevaa tietotaitoa ja käytiin kysymässä apuja testien luomiseen muilta työntekijöiltä.

Kun ohjelmointiympäristöt olivat valittu, alettiin suunnitella testien rakennetta. Regressiotesteihin liittyen suunniteltiin, mitä asioita tulisi testata ja minkä asioiden tulisi määrittää, onko testi suoritettu onnistuneesti. Tämän yhteydessä piirrettiin muun muassa kappaleessa kuusi nähty kaavio testien suorittamisjärjestyksestä, mikäli kaikki testin osat suoritettaisiin. Kuormitustestien osalta pohdittiin asioita kuten käyttöliittymässä näkyviä tietoja sekä tutkittiin millaisia rajoituksia Wapicen tarjoamassa rajapinnassa on.

Luoduista ohjelmista saatiin valmiiksi versiot joilla sekä varmistetaan palvelun toimintojen oikeanlainen toimivuus että lisätään simuloituja laitteita palveluun. Simuloiduilla laitteilla voidaan testata rasitusta. Laitteiden lisäys on tällä hetkellä vain rajoitettu 20:een, koska käyttäjäkohtainen laitemäärä on rajoitettu viiteen ja testejä varten palveluun luotiin neljä käyttäjää. Regressiotesteihin tullaan lisäämään testejä, joilla voidaan testata palvelimessa vielä useampia ominaisuuksia.

LÄHTEET

Anderson. 2011. Regression Testing. [Verkkojulkaisu]. University of Edinburgh. [Viitattu 23.9.2018]. Saatavilla: https://www.inf.ed.ac.uk/teaching/courses/st/2011-12/Resource-folder/11_regression.pdf

Broekman, B. & Notenboom, E. 2002. Testing Embedded Software. Addison-Wesley Professional.

Bruce, K B. 2002. Foundations of Object-Oriented Languages. MIT Press.

CODESYS. Ei päiväystä. The System. [Verkkosivu]. 3S-Smart Software Solutions GmbH. [Viitattu 16.9.2018]. Saatavilla: https://www.codesys.com/the-system.html

Ecma. 5.12.2017. C# Language Specification. [Verkkojulkaisu]. Ecma International. [Viitattu 23.9.2018]. Saatavilla: http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-334.pdf

Epec. Ei päiväystä. Epec company presentation. [Verkkojulkaisu]. Epec Oy. [Viitattu 6.9.2018]. Saatavilla: Vain yrityksen sisäisessä käytössä.

Epec. Ei päiväystä. CODESYS 2.3 ja 3.5 ohjelmointiympäristöt. [Verkkosivu]. Epec Oy. [Viitattu 16.9.2018]. Saatavilla: http://www.epec.fi/fi/ohjausjarjestelmat/ohjelmointi-ja-huolto-ohjelmat/codesys-2-3-ja-3-5/

Epec. 2018a. Epec MultiTool User Manual. [Verkkojulkaisu]. Epec Oy. [Viitattu 17.9.2018]. Saatavilla Epecin Extranetistä: http://epec.planeetta.com/Public/Manuals/MultiToolWebhelp/MultiTool.html. Vaatii käyttöoikeuden.

Epec. 2018b. Epec MultiTool User Manual. [Verkkojulkaisu]. Epec Oy. [Viitattu 27.9.2018]. Saatavilla Epecin Extranetistä: http://epec.planeetta.com/Public/Technical_Documents/6100/WebHelp/6100_webhelp_man000629.htm. Vaatii käyttöoikeuden.

Github. Ei päiväystä. SeleniumLibrary. [Verkkosivu]. Selenium Community. [Viitattu 23.9.2018]. Saatavilla: https://github.com/robotframework/SeleniumLibrary/

Harju. 2005. Monimuotoisuus. [Verkkojulkaisu]. Haaga-Helia ammattikorkeakoulu. [Viitattu 23.9.2018]. Saatavilla: http://myy.haaga-helia.fi/~ict1td002/ILTATIKO/OSIO_3/Luennot/Monimuotoisuus_1.0.ppt

Harle. 2009. Object Oriented Programming. [Verkkojulkaisu]. Robert Harle. [Viitattu 23.9.2018]. Saatavilla: https://www.cl.cam.ac.uk/teaching/0910/OOProg/OOP.pdf

Menascé. 2002. Load Testing of Web Sites. [Verkkojulkaisu]. George Mason University. [Viitattu 23.9.2018]. Saatavilla: https://pdfs.semanticscholar.org/0094/bdbfb7a1364c5c32cda6b54c3d1e685b6c55.pdf

MuleSoft. Ei päiväystä. What is an API? (Application Programming Interface). [Verkkosivu]. MuleSoft Inc. [Viitattu 17.9.2018]. Saatavilla: https://www.mulesoft.com/resources/api/what-is-an-api

Ponsse. Ei päiväystä. Ponsse Scorpionking. [Verkkosivu]. Ponsse Oyj. [Viitattu 27.9.2018]. Saatavilla: https://www.ponsse.com/fi/tuotteet/harvesterit/scorpionking

Robot Framework. Ei päiväystä. INTRODUCTION. [Verkkosivu]. Robot Framework organization. [Viitattu 31.8.2018]. Saatavilla: http://robotframework.org

Robot Framework. Ei päiväystä. LIBRARIES. [Verkkosivu]. Robot Framework organization. [Viitattu 1.9.2018]. Saatavilla: http://robotframework.org/#libraries

Robot Framework. Ei päiväystä. Pass And Fail Test Log. [Verkkosivu]. Robot Framework organization. [Viitattu 17.9.2018]. Saatavilla: http://robotframework.org/robotframework/latest/images/visible_log_level.html

Stackify. 14.12.2017. OOP Concept for Beginners: What is Inheritance?. [Verkkosivu]. Stackify. [Viitattu 23.9.2018]. Saatavilla: https://stackify.com/oop-concept-inheritance/

Seleniummaster. Ei päiväystä. Page Object Model in Selenium Robot Framework Python. [Verkkosivu]. Selenium Master. [Viitattu 27.9.2018]. Saatavilla: http://seleniummaster.com/sitecontent/index.php/selenium-robot-framework-menu/selenium-robot-framework-python-menu/193-page-object-model-in-selenium-robot-framework-python

Visual Studio. Ei päiväystä. What's new in Visual Studio 2017. [Verkkosivu]. Microsoft Corp. [Viitattu 23.9.2018]. Saatavilla: https://visualstudio.microsoft.com/vs/whatsnew/

Wapice. Ei päiväystä. IoT-Ticket Platform. [Verkkosivu]. Wapice Ltd. [Viitattu 16.9.2018]. Saatavilla: https://iot-ticket.com/platform

Wapice. 10.3.2017. IoT-Ticket Platform. [Verkkojulkaisu]. Wapice Ltd. [Viitattu 17.9.2018]. Saatavilla: https://iot-ticket.com/images/Files/IoT-Ticket.com_IoT_API.pdf