OULUN AMMATTIKORKEAKOULU

Samuli Hölttä

OHJELMISTOROBOTIIKAN KOULUTUSALUSTA

OHJELMISTOROBOTIIKAN KOULUTUSALUSTA

Samuli Hölttä
Opinnäytetyö
Syksy 2021
Tietotekniikan tutkinto-ohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tieto- ja viestintätekniikka, ohjelmistokehittäjä

Tekijä(t): Samuli Hölttä
Opinnäytetyön nimi suomeksi: Ohjelmistorobotiikan (RPA) koulutusalusta
Opinnäytetyön nimi englanniksi: Robotic Processing Automation (RPA) Training Platform
Työn ohjaaja(t): Pekka Alaluukas
Työn valmistumislukukausi ja -vuosi: Syksy 2021
Sivumäärä: 44

Opinnäytetyön aiheena on ohjelmistorobotiikan ja robottien kehitystä käsittelevän koulutusalustan kehitys. Työn toimeksiantajana toimii Bittium Wireless Oy, jossa toimin testiautomaatiokehittäjän tehtävissä.

Työn tavoitteena on saada aikaan kattava koulutuskokonaisuus, jonka avulla Bittiumin uudet työntekijät perehdytetään ohjelmistorobotiikkaan erikoistuneessa projektitiimissä työskentelyyn ja ohjelmistorobottien kehitykseen.

Koulutusalusta koottiin kolmen osakokonaisuuden ympärille, jotka muodostavat yhdessä virtaviivaisen kokonaisuuden ohjelmistorobottien kehityksestä ja sen prosesseista. Osakokonaisuuksien tehtävät suunniteltiin yhteistyössä robotiikka-projektitiimin jäsenten kanssa, jonka jälkeen tehtävät suoritettiin itse ja dokumentoitiin.

Koulutusalusta konkreettisesti sijoittuu organisaatiowikiohjelmisto Confluenceen sekä projektinhallintatyökalu Jiraan. Confluence-sivusto toimii koulutettavan oppaana koulutuksen aikana ja Jira sisältää itse koulutuksen tehtävät.

Opinnäytetyön toteutukseen käytettävän ajan rajallisuuden vuoksi koulutusalusta tällaisenaan ei ole vielä lopullinen versio, vaan sitä tullaan jatkokehittämään opinnäytetyön jälkeen. Opinnäytetyön tavoitteisiin kuitenkin päästiin ja aikaan saatiin kattava koulutuskokonaisuus, jota on helppo jatkokehittää lisää ja viimeistellä lopulliseen käyttöönottoon.

Asiasanat: RPA, ohjelmistorobotiikka, ohjelmistorobotti

ABSTRACT

Oulu University of Applied Sciences
Information Technology, Software Development

Author(s): Samuli Hölttä
Title of thesis: Robotic Processing Automation (RPA) Training Platform
Supervisor(s): Pekka Alaluukas
Term and year when the thesis was submitted: Fall 2021
Pages: 44

The topics of the thesis are a training platform regarding robotic processing automation, and the software robots -development. The client of this project is Bittium Wireless Oy, where I work as a test automation developer.

The project's goal is to produce a comprehensive set of trainings, which is used to introduce Bittium's new employees to develop software robots, as well as work in a project group specialized in software robotics.

The training platform were gathered around three subassemblies, which together formed a streamlined entirety regarding the software robotics' development and its processes. The tasks of the subassemblies were designed together with the members of the software robotics -team, after which the tasks were ran and documented individually.

The training platform is located in a corporate wiki software called Confluence and in a project management tool called Jira. The Confluence website serves as a guide for the trainees during the training, and Jira serves as a management tool for the training tasks.

Because of the limitlessness of the timeframe used on the thesis, the training platform documented here is not the final version; instead, it will be developed further after the thesis is finished. However, the goals of the thesis were met, and as a result, a comprehensive training set was developed. The set can easily be expanded upon, thus making finishing it simple for its final deployment.

Keywords: RPA, Robotic Processing Automation, software robot

SISÄLLYS

1 JOHDANTO

Ohjelmistorobotiikka on yleistyvä teknologia, jonka tarkoituksena on automatisoida rutiininomaisia toimintoja yritysten liiketoimintaprosesseissa ja tehostaa työnteon tehokkuutta ja kannattavuutta. Ohjelmistorobotiikka vapauttaa työntekijöitä puuduttavista työtehtävistä ja antaa työntekijöille mahdollisuuden keskittyä mielekkäämpiin ja yrityksen liiketoiminnan kannalta tuotteliaampiin työtehtäviin. Pitkällä aikatähtäimellä hyvin suunniteltuna ohjelmistorobotiikan avulla yritys voi saavuttaa huomattaviakin säästöjä. (Staria 2019.)

Ohjelmistorobotiikassa ei käytetä fyysisisiä robotteja, vaan robotit toimivat ohjelmallisesti jäljitellen ihmisen tekemiä näppäilyjä ja hiiren napsautuksia. Ohjelmistorobotille osoitettujen tehtävien tulisi luonteeltaan olla sellaisia, jotka eivät vaadi inhimillistä päättelykykyä tai vuorovaikutustaitoja. Tästä johtuen ohjelmistorobotit ovat parhaimmillaan yksinkertaisten ja rutiininomaisten tehtävien automatisoinnissa.

Opinnäytetyön tarkoituksena oli toteuttaa toimeksiantajalle ohjelmistorobotiikkaan liittyvä koulutusalusta, joka käsittelee ohjelmistorobotiikkaan erikoistuneen projektitiimin käyttämiä työkaluja, teknologioita ja yleisesti ottaen projektitiimin työkäytäntöjä ja toimintamalleja. Koulutusalustaa on tarkoitus käyttää uusien työntekijän perehdyttämiseen. Koulutusalusta koostuu kolmesta osakokonaisuudesta, jotka käyvät alusta loppuun läpi prosessin, miten ohjelmistorobotteja kehitetään. Jokaiseen osakokonaisuuteen kuuluu eritasoisia tehtäviä, jotka olen tehnyt myös itse opinnäytetyön aikana. Esittelen opinnäytetyössä koulutusalustan osakokonaisuudet, niiden tehtävät ja miten ratkaisin itse tehtävät.

Opinnäytetyön toimeksiantajana oli Oulussa sijaitseva informaatioteknologia-alan yritys Bittium Wireless Oy. Yrityksen toimintaan kuuluvat turvalliset viestintä- ja liitettävyysratkaisut sekä tuotekehitys taktiseen kommunikaatioon ja biosignaalien mittaamiseen ja monitorointiin. Bittium tarjoaa omien tuotteiden lisäksi myös erilaisia tuote- ja järjestelmäratkaisuja sekä suunnittelupalveluja. (Bittium 2021.)

2 OHJELMISTOROBOTIIKKA

2.1 Määritelmä

Ohjelmistorobotiikka eli RPA (Robotic Processing Automation) tarkoittaa teknologiaa, jonka tarkoituksena on automatisoida rutiininomaisia ja normaalisti ihmisen manuaalisesti tekemiä työtehtäviä erilaisissa liiketoimintaprosesseissa ohjelmallisesti. Kyseessä ei siis oli fyysinen robotti, joita käytetään esimerkiksi autoteollisuudessa, vaan koodattu ohjelmistorobotti (Torikka 2019). Manuaaliset työvaiheet, joiden tekeminen ei vaadi erityistä päättelykykyä tai ammattiosaamista, ovat tyypillisiä tehtäviä roboteille ohjelmistorobotiikassa. Ohjelmistorobottien avulla työntekijät saadaan vapautettua puuduttavista ja aikaa vievistä tehtävistä, mikä taas antaa mahdollisuuden työntekijälle tehdä hyödyllisempiä ja vaativampia työtehtäviä. Se taas parantaa tehokkuutta ja vähentää kustannuksia.

Robotit voivat toimia täysin itsenäisesti ilman erillistä ohjausta tai vaihtoehtoisesti robotti voi toimia yhdessä ihmisen kanssa, jolloin ihminen pystyy ohjaamaan robotin tekemiä toimintoja yksityiskohtaisemmin (Staria 2020). Ohjelmistorobotit toimivat pinta- ja käyttöliittymätasolla jäljittelemällä työntekijöiden näppäilyjä ja hiiren napsautuksia ja suorittamalla tehtävän samalla tavalla kuin työntekijät, esimerkiksi kirjautumalla sovelluksiin, syöttämällä tietoja, tekemällä laskutoimituksia ja suorittamalla uloskirjautumisia. Robotit käyttävät siis hyväkseen samoja tietojärjestelmien käyttöliittymiä ja rajapintoja kuin ihmiset.

Ohjelmistorobotiikka on kasvattanut suosiotaan viime vuosien aikana yritysten liiketoimintaprosessien tehostajana. Ohjelmistorobotiikan toteutuksiin tarkoitettuja työkaluja ja ohjelmistoja on saatavilla jo useita, ja niiden avulla on mahdollista saavuttaa merkittäviä hyötyjä yritysten liiketoiminnassa. Gartnerin ennustuksen mukaan vuonna 2021 ohjelmistorobotiikan markkinaosuus tulee kasvamaan 19,5 % 1,9 miljardiin dollariin verrattuna vuoden 2020 1,57 miljardiin dollariin. Ennustuksen mukaan markkinaosuus yltää kaksinumeroisiin lukuihin vuoteen 2024 mennessä. (Tucci 2021.) Suosio ohjelmistorobotiikkaa kohtaan ei tule yllätyksenä, sillä alan analyytikoiden mukaan yrityksen sijoittaman pääoman tuotto RPA-toteutuksissa voi olla kaksin- tai jopa kolminkertainen.

2.2 Sovelluskohteita

Ohjelmistorobotiikassa käytettävien ohjelmistorobottien ja niiden tarjoamien ominaisuuksien avulla suoritettavien automatisoitujen prosessien käyttökohteeksi soveltuu melkein mikä tahansa tietokoneella tehtävä työ. Ohjelmistorobotit käyttävät prosesseissaan samoja tietojärjestelmiä sekä käyttöliittymiä kuin ihmiset. Näiden järjestelmien ja käyttöliittymien käyttö eri alan töissä on nykyään arkipäivää, joten roboteille soveltuvia automaation käyttökohteita löytyy valtava määrä lähes jokaiselta alalta. Ohjelmistorobotille oikeiden tehtävien pohdinnassa hyvänä pääsääntönä on se, että prosessin kaikki askeleet tulisi kyetä määrittelemään tarkasti etukäteen sekä kaikki mahdolliset tapahtumat ja lopputulokset matkan varrella tulisi kyetä ennakoimaan. (Asatiani & Penttinen 2016, 68.)

Ohjelmistorobotiikkasovellukset sisältävät monia tehokkaita työkaluja apuna automatisointitoteutuksen kehittämiseen. Näiden ohjelmistojen avulla voidaan suunnitella ja rakentaa laajoja automaatiokokonaisuuksia. Ohjelmistojen sisäänrakennettujen perustoimintojen lisäksi voidaan niihin kehittää omia komponentteja, joiden avulla valmiita toiminnallisuuksia voidaan jatkokehittää tai luoda kokonaan uusia. Koska ohjelmistorobotit käyttävät prosesseissaan samaa käyttöliittymää kuin ihmiset, robottien toiminta perustuu esimerkiksi hiirellä tehtäviin painalluksiin ja näppäimistöstä saatuihin syötteisiin. Robotti voi siis käytännössä esimerkiksi avata dokumentin, lukea dokumentista tietoja, tallentaa ne ja lähettää tiedot eteenpäin jatkokäsittelyä varten. Joissain tapauksissa ohjelmistorobotti voi tarjota ainoan tavan tehdä järjestelmäintegraatio kahden yhteensopimattoman järjestelmän välillä. (Hämäläinen 2019, 11.)

Ohjelmistorobotille osoitetut tehtävät tulisi olla toiminnaltaan sellaisia, että tehtävien ratkaiseminen ja päätöksien tekeminen eivät vaadi monimutkaista harkinnanvaraisuutta, vaan päätöksien tekoon tarvittavat säännöt ovat etukäteen tunnistettu ja opetettu ohjelmistorobotille etukäteen (Staria 2020). Tästä syystä ohjelmistorobotteja käytetään lähinnä yksinkertaisien ja rutiininomaisien prosessien automatisointiin. Tällaisia työtehtäviä löytyy tyypillisesti organisaatioiden sisäisistä ydinprosesseista ja tukitoiminnoista. Ohjelmistorobotiikka on todettu toimi-

van automatisointiratkaisuna silloin, kun tavoitteina on automatisoida tuotannossa olevia prosesseja ja pyrkimyksenä säilyttää käytössä olevat tietojärjestelmät. (Kääriäinen ym. 2018, 8.)

2.3 Hyödyt ja mahdollisuudet

Ohjelmistorobotiikan edut voivat olla parhaimmillaan hyvin laaja-alaisia ja kauaskantoisia. Ohjelmistorobotiikalla voidaan parantaa yrityksen toimintatehokkuutta, tehostaa nykyisiä toimintamalleja ilman raskaita muutoksia jo olemassa oleviin ohjelmistoihin sekä pienentää tuotantokustannuksia (Salminen 2018). Robotit suorittavat prosessit virheettömästi, ympärivuorokautisesti sekä selkeästi ihmistä nopeammin myös suurien massadatojen kanssa. Tästä syystä ohjelmistorobotiikan kyky suorittaa tehtäviä suurissa määrissä on mittavimpia etuja, mitä ohjelmistorobotiikka tuo. Useat tapaustutkimukset yrityksistä, jotka automatisoivat prosessejaan ohjelmistorobotiikan avulla osoittavat, että prosessien automatisoiduttua niiden määrä vähintään kaksinkertaistui (Osman 2019, 70).

Robotti ottaa huomioon vain ja ainoastaan sille osoitetut tehtävät ja suorittaa prosessin joka kerta samalla tavalla. Toisin kuin ihmisten tekemiä virheitä, robottien virheitä voidaan seurata ja tutkia. Ohjelmistorobotiikassa robottien suorituskykyä seurataan ja kaikki robotin tekemät toimenpiteet prosessin aikana kirjautuvat automaattisesti järjestelmiin, joista virheiden syitä ja seuraamuksia on helppo tarkastella (Kääriäinen ym. 2018, 55).

Monen yrityksen tärkeimpänä syynä ohjelmistorobotiikan toteutuksen hankkimiselle on henkilöstötyytyväisyys (Staria 2019). Ohjelmistorobottien käyttäminen puuduttavissa ja toistuvissa tehtävissä vapauttaa työntekijöille aikaa mielekkäämpiin, vaativampiin sekä yritykselle strategisesti arvokkaampiin tehtäviin, jotka vaativat inhimillistä päättelykykyä ja vuorovaikutustaitoa. Lisäksi ohjelmistorobotiikan avulla yritys voi luopua halutessaan ulkopuolisen työvoiman käytöstä, jos yritys on niihin aiemmin turvautunut (Fujitsu 2018).

Ohjelmistorobotiikan toteutukset ovat tyypillisesti edullisia ja niiden käyttöönoton on todettu olevan usein helppoa. Ohjelmistorobottien käyttöönotto ja mallintami-

nen tapahtuu nopeasti verrattuna suurempiin yritysjärjestelmiin ja muihin automaatiomuotoihin (Syed ym. 2019). Robottien käyttöönottoprosessi voi kestää lyhyimmillään kahdesta neljään viikkoon, kun taas suurempien yritysjärjestelmien käyttöönotto- tai muutosprosessit voivat kestää useita vuosia (Asatiani & Penttinen 2016, 70). Ohjelmistorobottien toiminnallisuutta ja mallintamista on helppo suunnitella ja muokata, sillä se ei vaadi käyttäjältä erityisiä ohjelmointitaitoja.

2.4 Haitat ja uhat

Lukuisista hyvistä puolista huolimatta on ohjelmistorobotiikalla myös huonoja puolia. Ohjelmistorobotiikka on teknologiana uusi ja lisäksi voi aiheuttaa häiriöitä työprosesseissa ja kulttuurissa (Torikka 2019). Tekniset häiriöt, turvallisuusongelmat sekä virheellinen muutos- ja käyttöönottoprosessi voivat heikentää kannattavuutta ja vaikuttaa työntekijöiden tehokkuuteen ja liiketoiminnan kulkuun (Capgemini 2020).

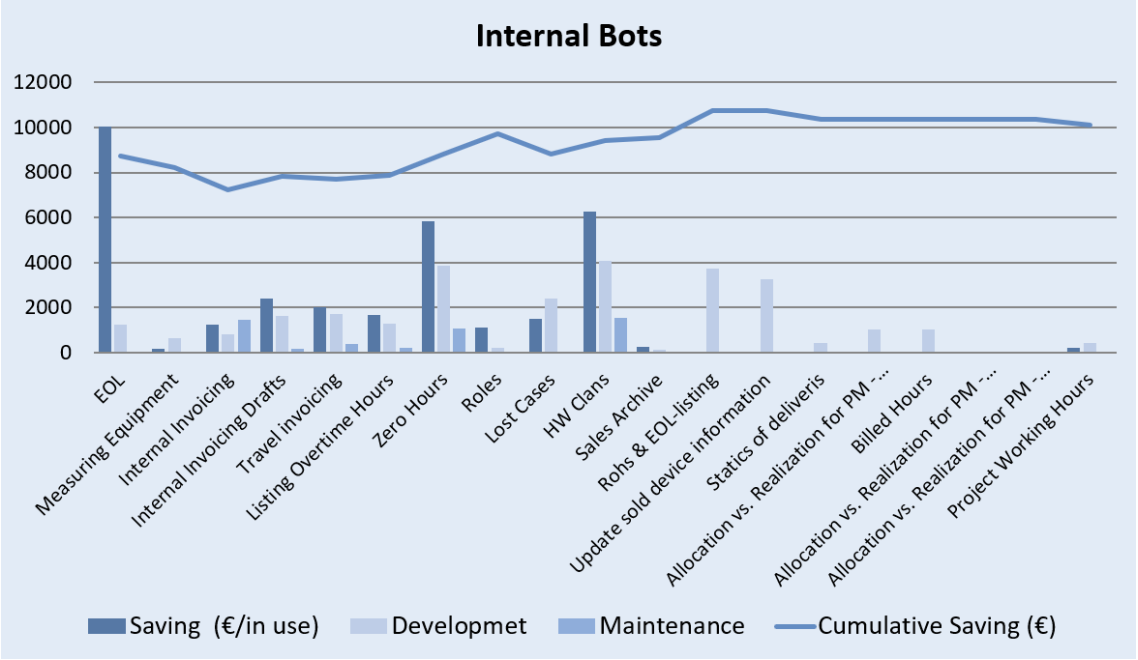
Koska RPA jäljittelee sitä, miten ihmiset suorittavat erilaisia liiketoimintaprosessiin liittyviä tehtäviä, on tärkeää, että RPA-toteutuksia ohjaavat aiheen asiantuntijat ja työntekijät, jotka ymmärtävät prosessin vaiheet ja joilla on mielipiteitä siitä, miten sitä voitaisiin parantaa automaation kanssa. Ohjelmistorobotiikan toteutuksia tarjoava yritys EY raportoi, että jopa 30-50 % alkuperäisistä RPA-hankkeista epäonnistuu juuri tästä syystä (Tucci 2021). Yritykset aliarvioivat usein RPA-robottien ylläpitoon tarvittavan IT-asiantuntemuksen ja infrastruktuurin. Useimmat yrityskäyttäjät voivat automatisoida yksinkertaisen prosessin itse RPA-toimittajien drag & drop -menulla, mutta täysimittaisen RPA-käyttöönoton toteuttaminen ja ylläpito vaatii ylläpito- sekä IT-valvontaa.

Vaikka ohjelmistorobotiikan mahdollistamat mielekkäämmät työtehtävät parantavat henkilöstötyytyväisyyttä ja ohjelmistorobottien käyttöönotosta saatu palaute on ollut pääsääntöisesti positiivista, on tutkimusten mukaan yleistä, että ohjelmistorobottien käyttöönotto aiheuttaa työntekijöiden keskuudessa vastarintaa (Lacity & Willocks 2015, 4). Yleisenä huolena on, että ohjelmistorobotiikka ja automatisoidut prosessit korvaavat työntekijöitä ja johtavat irtisanomisiin (Salminen 2018). Työntekijä voi tällöin kokea itsensä kilpailemassa työpaikasta robottia vastaa.

Tämä voi aiheuttaa jännitteitä johdon ja heidän työntekijöidensä välille sekä vaikuttaa työntekijöiden moraaliin. Tästä syystä RPA-prosessien esittelyt ja käyttöönotot tulisi käsitellä hienovaraisesti ja niistä on tiedotettava oikein (Asatiani & Penttinen 2016, 72).

3 RPA BITTIUMILLA

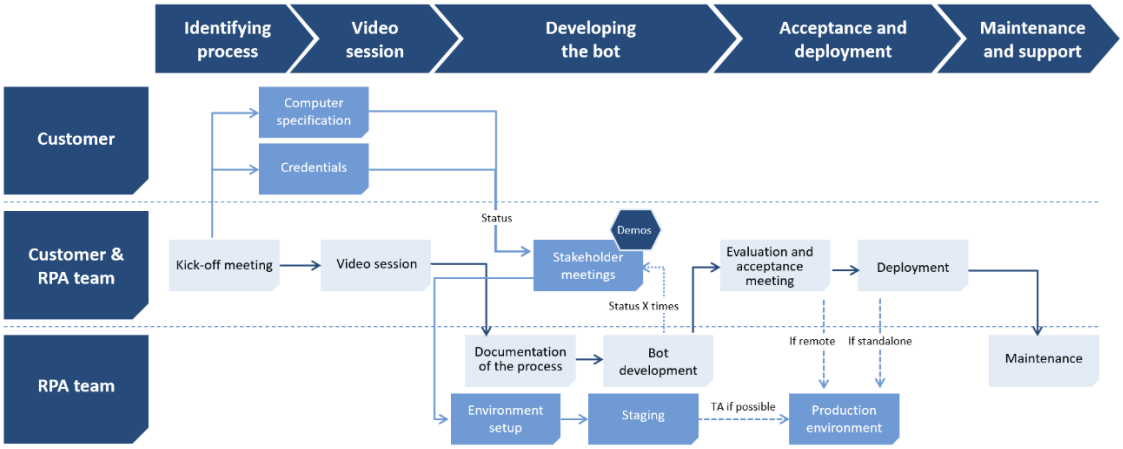
Teknisten suunnittelupalvelujen lisäksi Bittium tarjoaa ohjelmistorobotiikkaratkaisuja, jotka perustuvat Bittiumin omaan suojattuun alustaan. Bittiumilla on laaja kokemus ohjelmistokehityksessä korkeatasoisen testiautomaation tutkimus- ja tuotekehityksen parissa. Tätä taustaa on käytetty hyödyksi Bittiumin omassa RPA-projektissa nimeltään Bitnator. Bittiumilla käytiin läpi kolme erilaista kaupallista RPA-työkalua, mutta päätökseen oman RPA-työkalun, Bitnatorin, kehitykseen päädyttiin vaikeiden hinnastomallien sekä Bittiumille tärkeiden ominaisuuksien puuttumisen vuoksi. Bitnator on käytössä myös Bittiumin omissa sisäisissä prosesseissa ja tällä hetkellä käytössä on noin 70 eri robottia eri prosessien parissa (Kuva 1).



KUVA 1. Bittiumin sisäisissä prosesseissa käytössä olevia robotteja ja niistä aiheutuvat kulut ja säästöt (Bittium 2020.)

Asiakastapauksissa robotit luodaan tiiviissä yhteistyössä asiakkaan kanssa, esimerkiksi tallentamalla robottien toimintaa videolle. Robotti luodaan automatisoitavan prosessin ja sen tarpeiden perusteella ja asennetaan toimintavalmiuteen Bittiumin RPA-työkalun avulla. Työkalu on skaalautuva sekä helposti integroita-

vissa olemassa oleviin järjestelmiin. Tarvittaessa automatisoidun prosessin parannus- sekä jatkokehitystoimet voidaan tehdä rinnakkain robottikehityksen kanssa. (Kuva 2.)



KUVA 2. Toimintamalli, miten RPA-kehitys toimii asiakassuhteissa (Bittium 2020.)

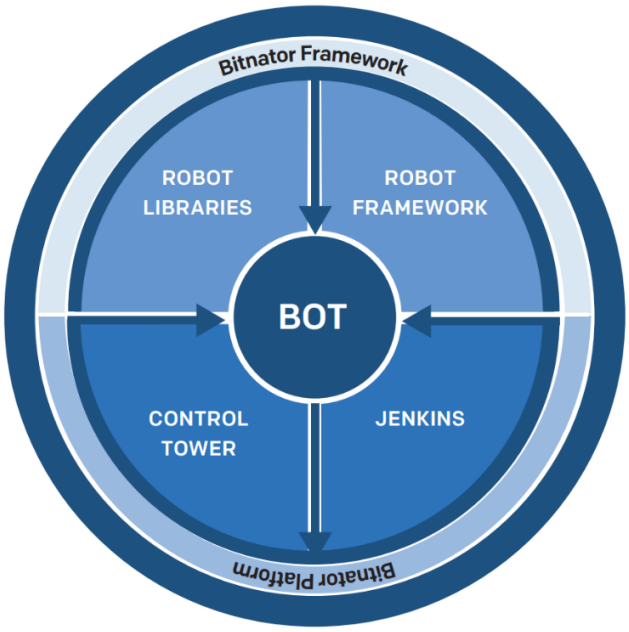
4 BITNATOR

Bitnator-työkalu on suunniteltu automatisoimaan GUI-pohjaisia prosesseja, joissa perinteinen ohjelmointirajapintapohjainen automaatio on joko liian kallista tai ei ole ollenkaan mahdollista. Bitnator tarjoaa helpon tavan hallita sekä käyttää RPA-robotteja.

Bitnator Platform (Kuva 3) koostuu Control Towerista ja Jenkinsistä. Jenkins on avoimen lähdekoodin käännösautomaation palvelinohjelmisto, joka tarjoaa yksinkertaisen tavan pystyttää jatkuvan integraation (Continuous Integration) ympäristö lähes kaikille ohjelmistokielille sekä lähdekoodivarastoille. Lisäksi sillä voidaan automatisoida muita erilaisia rutiininomaisia kehitystehtäviä. (Heller 2020.) Bitnator-projektissa Jenkinsillä on keskeinen rooli, sillä sen avulla suoritetaan kaikki bottien etäkäytön keskeiset toiminnot, muun muassa master-slave-yhteys bottikoneisiin, skriptien ajo etänä bottikoneilla, bottiajojen aikatauluttamiset, versionhallinnan sekä käyttäjienhallinta.

Control Tower (CT) on oman käyttöliittymän sisältävä työpöytäsovellus Windows-laitteelle, joka mahdollistaa bottien ajon, ajojen tulosten tarkastelun sekä ajojen aikatauluttamisen. Control Towerilla hallitaan pääasiassa Jenkinsissä olevia botteja, mutta paikallisesti asennettujen bottien hallinta on myös mahdollista. Botit implementoidaan Bitnator Frameworkin avulla.

Bitnator Framework (Kuva 3) käyttää avoimen lähdekoodin ohjelmistoja, kuten Robot Frameworkia sekä Python-kirjastoja robottien kehitykseen. Tämä mahdollistaa nopean ja joustavan koodipohjaisen robottikehityksen. Koodipohjaisen menettelytavan hyväksikäyttö robottien kehityksessä tekee ratkaisusta tehokkaamman esimerkiksi suurten tietomäärien käsittelyssä. Robot Framework tarjoaa helpon lähestymistavan robottien kehittämiseen, sekä se on myös helppo oppia ja kehitys Robot Frameworkilla on nopeaa.



KUVA 3. Bitnator-työkalun arkkitehtuuri (Bittium 2020.)

5 RPA-KOULUTUS

Työn tavoitteena oli toteuttaa Bittiumin sisäinen RPA-koulutusalusta, jonka kautta Bittiumille tulevat uudet työntekijät on helppo perehdyttää ja kouluttaa alustassa käsiteltäviin aiheisiin. Koulutusalusta koostuu kolmesta aihealueesta, jotka liittyvät aiemmassa luvussa 4 käsiteltyyn Bittiumin RPA-työkaluun Bitnatoriin eli jatkuvan integraation työkalusta Jenkinsistä, bottienhallintatyökalusta Control Towerista sekä bottien kehitysalustasta Robot Frameworkista. Jokainen aihealue sisältää eritasoisia aihealueeseen liittyviä tehtäviä. Koulutusalustan tehtävänhallinnassa käytetään projektinhallintatyökalu Jiraa.

Koulutusalustaa on tarkoitus käyttää työntekijälähtöisesti, eli työntekijälle voidaan osoittaa tietty osakokonaisuus koulutuksesta koko koulutuksen sijaan. Jos työntekijä esimerkiksi tulee työtehtävissään kehittämään ainoastaan botteja, hän käy todennäköisesti vain bottien kehitystä käsittelevän koulutuksen.

Tehtävänäni oli yhdessä Bittiumin RPA-projektitiimin jäsenten kanssa määritellä aihealueisiin tehtävät ja niiden arviointikriteerit, jonka jälkeen suoritan itse tehtäväkokonaisuuden. Lisäksi tehtävänäni oli koota organisaatiowikiohjelmisto Confluenceen sivusto, joka tulisi olemaan koulutettavilla apuna tehtävien tekemisessä ja jonka tarkoitus on ohjata koulutettava tehtävissä liikkeelle.

Confluence-sivusto käsittelee tehtävien esivalmistelut, kuten esimerkiksi tarvittavien työkalujen ja kirjastojen asennukset, sekä käy läpi tehtävien osakokonaisuuteen liittyviä asioita. Sivustolla pidin lisäksi kirjaa aihealueittain tehtäviin käytetystä ajasta sekä toin esille tehtävien aikana nousseita huomioita, joita voidaan käyttää apuna alustan jatkokehityksessä.

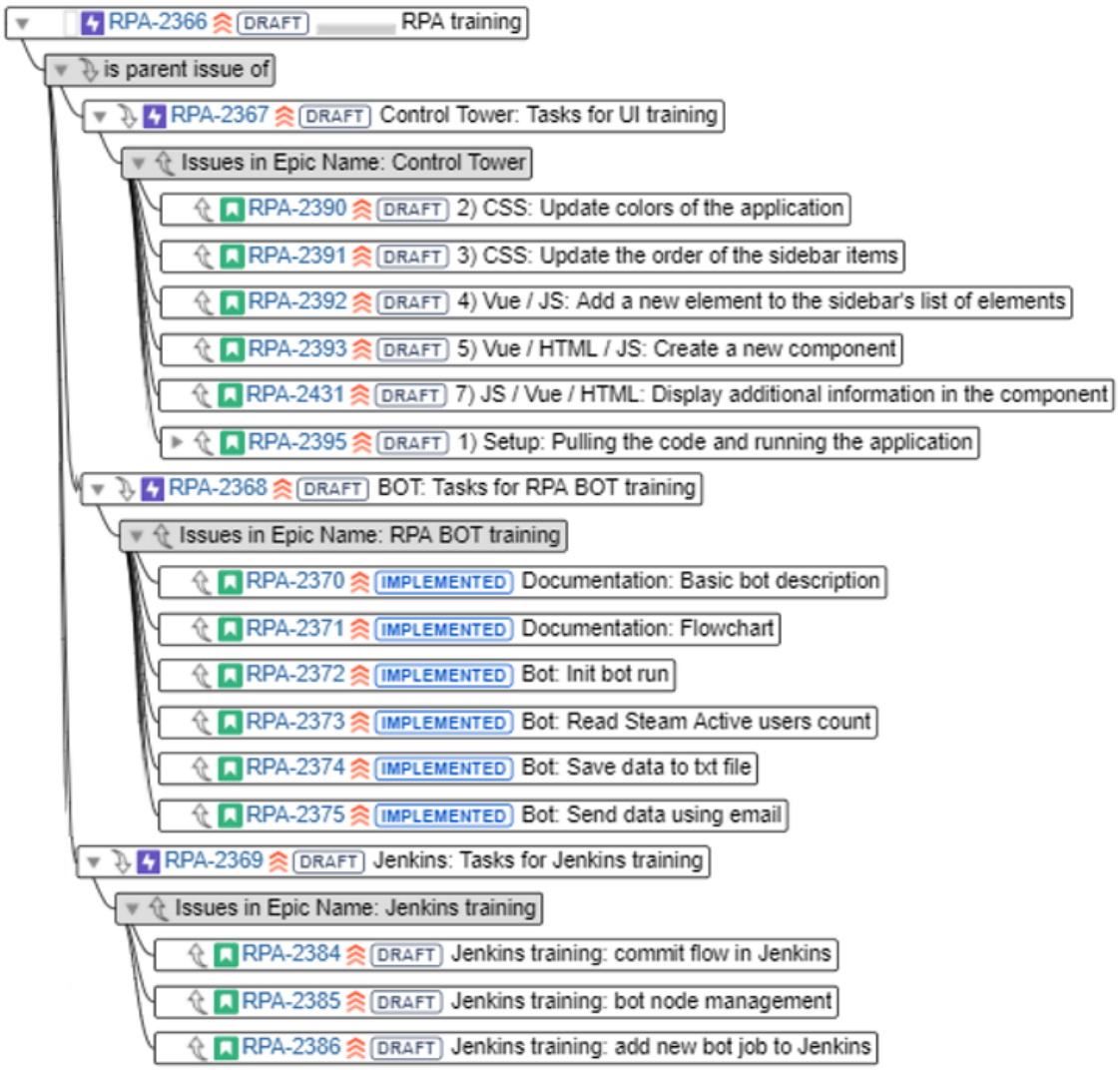
5.1 Koulutuksen tehtävät

Koulutuksen tehtävät jaettiin kolmeen eri osakokonaisuuteen, jotka liittyvät Bittiumin RPA-työkalun Bitnatorin toimintaan ja ohjelmistoon. Osakokonaisuuksien aiheet määräytyivät hyvin pitkälti aiemmassa luvussa kuvatun Bitnatorin rakenteen (Kuva 3) Bitnator Frameworkin ja Bitnator Platformin pohjalle. Bitnator Platformiin kuuluvat Jenkins ja Control Tower ovat kaksi omaa osakokonaisuutta ja

Bitnator Frameworkiin kuuluvat Robot Framework sekä siihen liittyvät kirjastot muodostavat yhden osakokonaisuuden robottien kehityksestä.

Jira - tehtävien hallinta

Tehtävien määrittely koulutuksessa tapahtuu projektihallintatyökalu Jiran avulla. Koulutettavalle työntekijälle luodaan koulutuksesta Jiraan epic (Kuva 4), jonka alta löytyy ala-epicit osakokonaisuuksista. Osakokonaisuuksien epiceille on asetettu storyja, jotka toimivat tässä tapauksessa osakokonaisuuksien tehtävinä. Tehtävissä edettäessä seurataan Jiran Workflow-menettelytapaa, jossa tehtävien status kuvastaa tehtävän etenemisen vaiheita.



Kuva 4 Jira-puu RPA-koulutuksesta

Gerrit - koodin katselmointi

Koulutuksessa tapahtuvaa ohjelmistokehitystä seurataan koodin katselmointityökalu Gerritillä. Koulutettava luo ennen tehtävien tekoa joka koulutuksesta oman Git-repositorion, jossa koulutuksen ohjelmointitehtävien versionhallinta tapahtuu. Ohjelmointia tapahtuu pääasiassa vain Robot Framework- ja Control Tower -koulutuksissa, joten Jenkins-koulutuksessa omalle Git-repositoriolle ei ole tarvetta.

Pääasiassa yksi commit koulutukselle luotuun repositorioon vastaa yhtä tehtävää. Commitin jälkeen joku RPA-tiimin jäsenistä tarkistaa commitissa mukana olevan koodin ja joko hyväksyy sen tai antaa korjausehdotuksia. Jos koodi vaatii korjauksia, ei uudelle commitille ole tarvetta, vaan commitin päivitys onnistuu helposti amend-komennolla. Kun commit on hyväksytty, se voidaan mergetä Gitin master-haaraan, josta commit on käytettävissä muun muassa botin ajoon.

5.2 Jenkins-koulutus

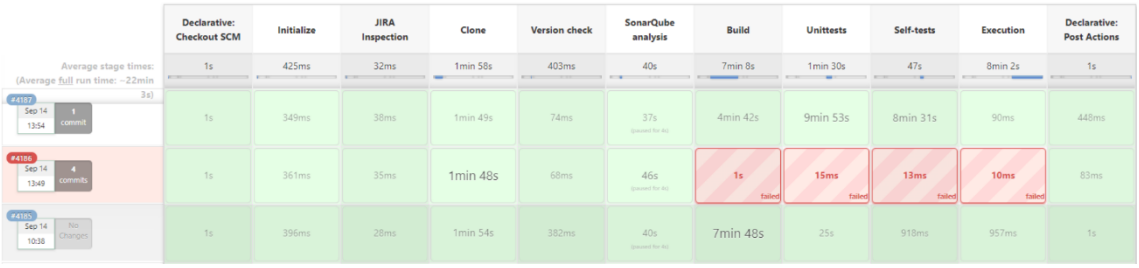
Jenkins-koulutuksen tarkoituksena on esitellä työntekijälle Bitnator-projektissa käytettävä bottien etähallintatyökalu Jenkins. Koulutuksessa käydään läpi Bitnator-projektissa käytettävät Jenkinsin perustoiminnot commitien käsittelystä ja niiden testaamisesta, bot-nodejen hallinnasta ja uusien bot-jobien luonnista. Koulutus ei sisällä paljoa käytännön tehtävää, vaan keskittyy enemmän tutustuttamaan koulutettava Jenkinsiin kolmella eri tehtäväkokonaisuudella ja confluence-sivuille tekemälläni dokumentaatiolla.

Tehtävä 1: Commit-flow Jenkinsissä

Ensimmäisessä tehtävässä tutustutaan Jenkinsin käyttöön apuvälineenä projektin versionhallinnassa sekä testaamisessa. Bitnatorin Jenkins-palvelin muodostaa jatkuvan integraation putken, jonka läpi kaikki projektiin liittyvät ohjelmistomuutokset eli commitit kulkevat. Bitnator-projektissa putki muodostuu Jenkinsiin luoduista putken osista, joita kutsutaan bot-jobeiksi. Nämä putken osaset vastaavat ohjelmiston rakentamisesta luotettavalla ja toistettavalla tavalla sekä rakennetun ohjelmiston etenemisestä erilaisten testaus- ja käyttöönottovaiheiden läpi.

Putki on jaettu kolmeen eri kehitysvaiheeseen: commit-, staging- sekä production-vaiheeseen. Jokaiselle vaiheelle on luotu nimensä mukainen bot-job, joka tuo versionhallintajärjestelmä Gitistä automaattisesti kyseisessä vaiheessa olevan botin uusimman version ja ajaa sille bot-jobissa määriteltyjä toimintoja ja testejä. Näitä bot-jobeja kutsutaan nimillä Commit-RPA, Staging-RPA sekä Production-RPA.

Commit-vaiheessa olevat botit ovat kehitysvaiheessa. Kuten edellä mainittiin, vaiheille luodut omat bot-jobit toimivat automaattisesti. Commit-vaiheessa olevien bottien kohdalla tämä tarkoittaa sitä, että bottien kehityksessä Gitiin tehdyt commitit menevät automaattisesti commit-vaiheelle luodun bot-jobin Commit-RPA:n läpi. Jenkinsin käyttöliittymän avulla commiteille suoritettuja toimintoja, testejä ja niiden lokitietoja voi seurata (Kuva 5).

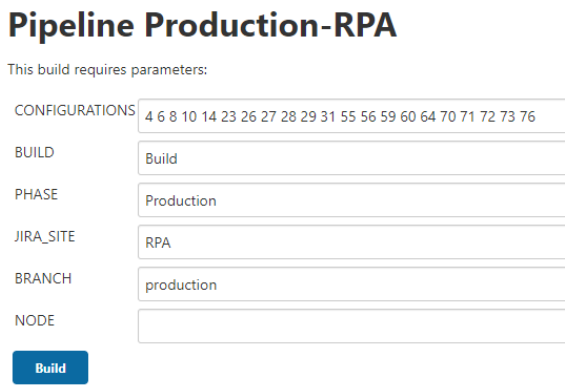


KUVA 5. Suorituslokit Commit-RPA-jobista Jenkinsissä

Vaihe bottien kehityksessä, jossa botit committeineen ovat valmiita mergeä varten Gitissä, kutsutaan Staging-vaiheeksi. Toisin sanoen aina, kun botti ja kehityksen aikana kertyneet commitit mergetään Gitissä, se menee Staging-RPA-jobin läpi. Staging-RPA-job pitää sisällään lähes samat vaiheet kuin Commit-RPA-job sillä erolla, että lopuksi staging-vaiheen läpi käynyt merge päivitetään staging-haaraan ja SonarQube-analyysi sekä version check (Kuva 6) jäävät pois.

KUVA 6. Suorituslokit Staging-RPA-jobista

Production-RPA-job eroaa aiemmista siten, että se ajetaan manuaalisesti (Kuva 7). Jobia ajettaessa sille syötetään parametrit, jotka sisältävät muun muassa ajoon mukaan tulevat botit. Production-ajo suoritetaan pääsääntöisesti aina sprinttien lopussa kahden viikon välein ja ajoon otetaan mukaan kaikki Jenkin-sissä olevat botit. Ajon jälkeen botit ja niihin tehdyt muutokset päätyvät tuotan-toon ja muutoksista generoidaan julkaisutiedot.

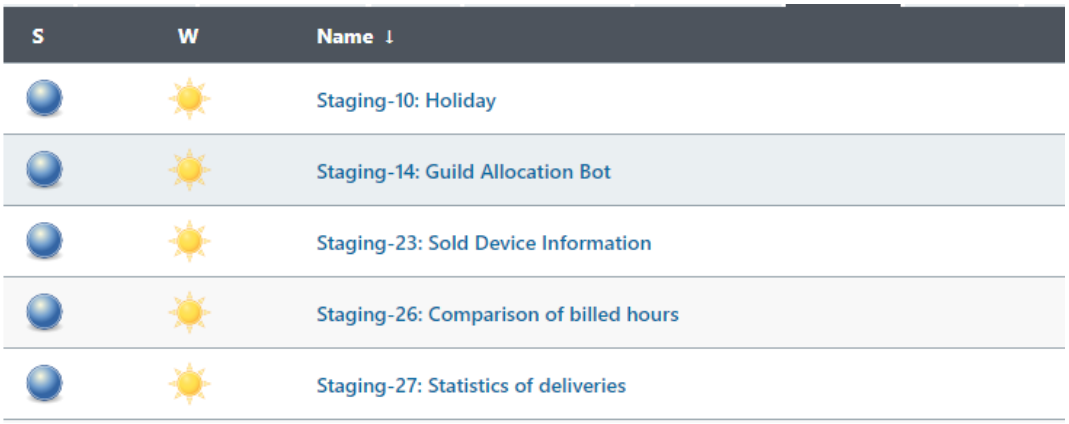


KUVA 7. Käyttöliittymä Production-RPA-jobin ajamiseen parametreineen

Tehtävä 2: Bot-jobien hallinta

Bitnator-projektin versionhallinnassa käytettyjen Commit-, Staging- ja Produc-tion-RPA-jobien lisäksi Jenkins-ympäristöön voidaan luoda boteille omia yksittäi-siä bot-jobeja (Kuva 8). Näiden avulla botteja voidaan ajaa etänä bot-nodejen laitteilla ja seurata Jenkinsistä käsin testien kulkua ja tuloksia.

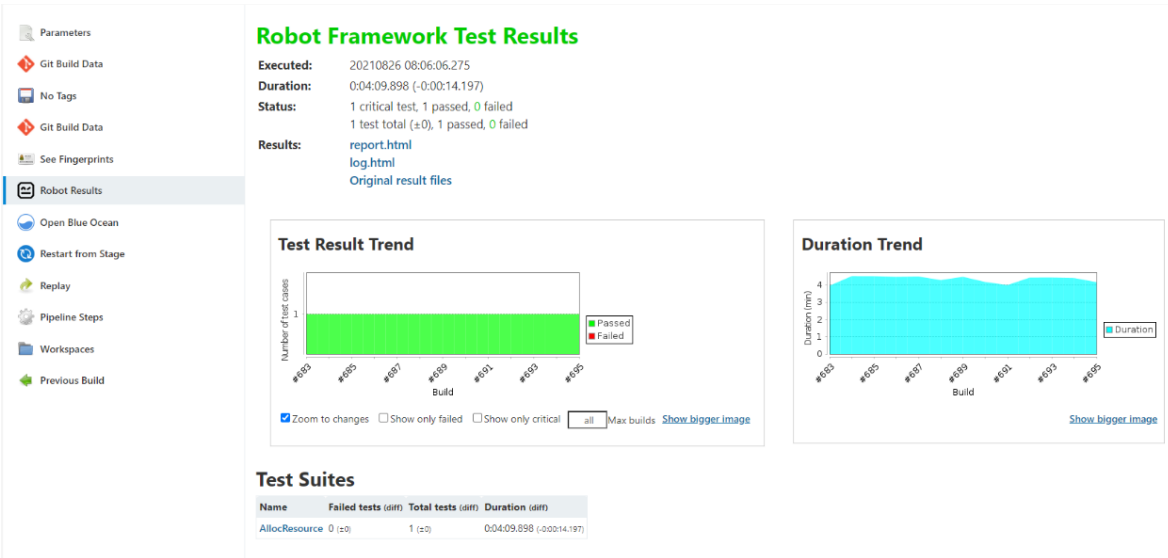
Bot-jobit ja niiden luonti seuraa samaa versionhallintaperiaatetta edellisessä teh-tävässä sivuttujen automaattisesti toimivien bot-jobien kanssa. Bot-job luodaan vastaamaan botin sen hetkistä kehitysvaihetta. Jokaisen botin kehitysvaiheesta tulee luoda bot-job Jenkinsiin, eli aloitettaessa uuden botin kehitys luodaan com-mit-vaiheen bot-job, mergen jälkeen luodaan staging-vaiheen bot-job, jonka jäl-keen luodaan production-vaiheen bot-job.



KUVA 8. Lista Staging-vaiheessa olevista bot-jobeista

Jenkinsissä bot-jobien luomisen konfiguraatioasetuksissa on käytössä erinäisiä string-parametreja, joihin esimerkiksi botin kehitysvaihe konfiguroidaan. Nimeämisessä käytetään botin nimen edessä vaiheen tunnistetta väliviivalla erotettuna. Esimerkiksi jos botin nimi olisi “training”, tulisi staging-vaiheen bot-jobin nimeksi “Staging-training” (Kuva 8).

Bot-jobia ajamalla bottien käyttämä Robot Framework generoi testausalustansa pohjalta Jenkinsiin testituloksia. Testitulokset pitävät sisällään graafeja ajamistaan testeistä sekä Robot Frameworkin alkuperäiset HTML result-tiedostot (Kuva 9).



KUVA 9. Testituloksien käyttöliittymä Jenkinsissä

Tehtävä 3: Bot-nodejen hallinta

Kolmannessa tehtävässä perehdytään Jenkinsin nodeihin ja niiden käyttöön Bitnator-projektissa. Nodet ovat laitteita osana Jenkins-ympäristöä, joita käytetään ajamaan esimerkiksi pipelinejä ja projekteja käyttäjänsä Jenkins-ympäristössä. Bitnator-projektissa nodeja käytetään boteille luotujen omien bot-jobien ja aiemmassa kappaleessa käsiteltyjen versionhallintaan liittyvien bot-jobien ja niihin kuuluvien testien ajoon.

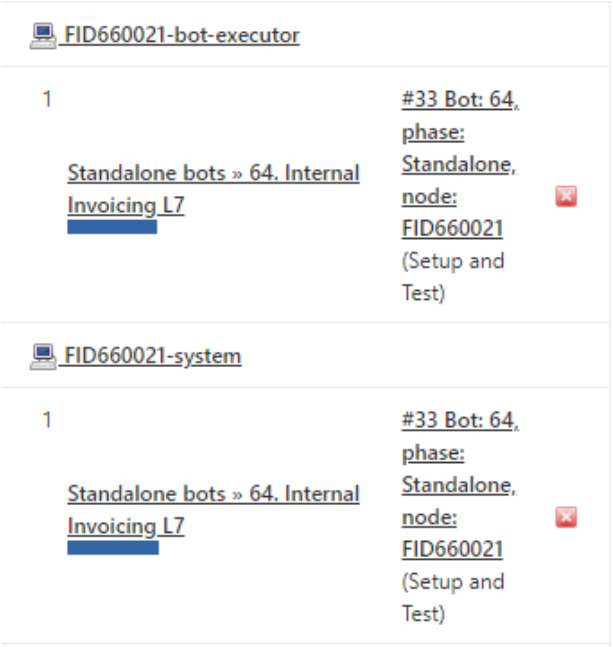
Bitnator-projektissa nodet ovat luokiteltu kolmeen erilaiseen nodetyyppiin: system-nodeihin, bot-executor-nodeihin ja linux-nodeihin. Linux-nodet ovat virtuaaliympäristössä toimivia Linux-käyttöjärjestelmällä varustettuja laitteita, joita käytetään vain lähinnä yleisien osien ajoon, jotka vaativat Linux-käyttöjärjestelmän toimiakseen.

System-nodet ovat bottikäyttöön osoitetuilla fyysisillä Windows-koneilla system-tunnuksella pyöriviä Jenkins-prosesseja. Prosessit käynnistyvät Windows-servicenä, joten ne ovat aina päällä. System-nodeille on asetettu enemmän oikeuksia, joten niitä käytetään myös tarvittaessa käskyihin, jotka näitä korkeamman tason oikeuksia vaativat.

Bot-executor-nodet system-nodejen tapaan ovat myös Jenkins-prosessin omaavia fyysisiä Windows-koneita. Bot-executor-nodet ovat niitä, jotka suorittavat itse bottiajon, joten tätä varten bot-executor-nodet tarvitsevat myös aktiivisen UI-session. Nodet käynnistyvät bottikäyttäjän kirjauduttua koneelle automaattisesti testin alussa ja sammuvat testin lopussa, kun bottikäyttäjä kirjautuu ulos koneelta.

Kaikille Jenkins-ympäristön nodeille on määritelty ID-numero ja jokaista ID-numeroa kohti on osoitettu yksi system-node ja bot-executor-node (Kuva 10). Tämä johtuu siitä, koska nodeille osoitetuilla Windows-koneilla on vain yksi executor ja vain yhtä bot-jobia voidaan ajaa kerrallaan. Koneen ID-numero on normaalisti koneen hostname, mutta tässä tapauksessa FID-tyyli on Bittiumin tapa nimetä koneet. System-nodeja siis käytetään varaamaan itselleen bot-jobien ajoa varten bot-executor-node muodostaen näin master-slave-yhteyden. Käytännössä tämä

tarkoittaa sitä, että testien alussa testaamista varten varataan vapaana oleva system-node, joka varaa itselleen omaa ID-numeroaan vastaavan bot-executor-noden, joka taas hoitaa bot-jobin ajon.



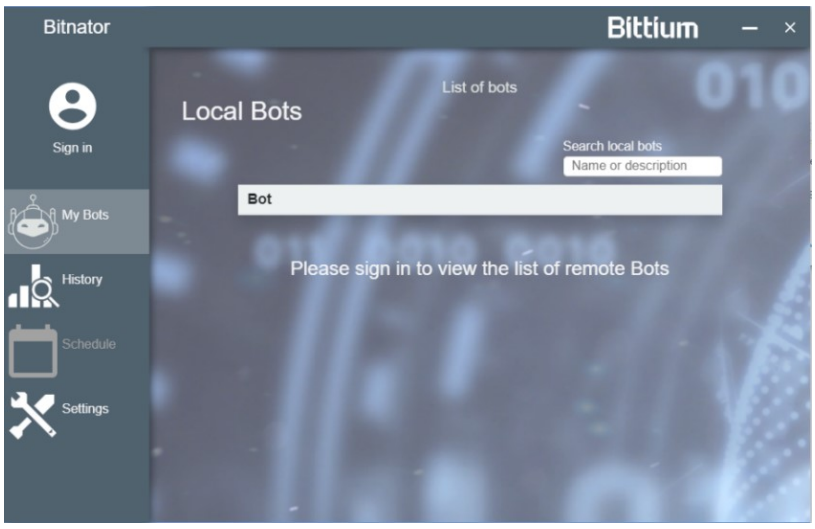
KUVA 10. System- ja bot-executor-noden välinen yhteys bottia ajaessa Jenkins-käyttöliittymässä

5.3 Control Tower -koulutus

Control Tower -koulutuksen tarkoitus on perehdyttää työntekijä Bitnator-projektin bottien hallintatyökalun Control Towerin kehittämiseen ja antaa pieni kosketus web-kehitykseen. Koulutus keskittyy Control Tower -sovelluksen frontend-puolen kehitykseen. Control Tower on Electron-ohjelmistokehystä käyttävä sovellus. Electron mahdollistaa GUI-työpöytäsovellusten kehityksen yhdistämällä Chromiumin renderointimoottorin ja Node.js-ajon. Sovellus käyttää Vue.js:ää JavaScript-kehyksenä käyttöliittymäkehityksessä, sekä muita web-applikaatiolle ominaisia ohjelmointikieliä, kuten HTML:ää ja CSS:ää. Tehtävissä käytettävään VSCode-työkaluun tulee koulutusta varten olla asennettuna ESLint-lisäosa, joka analysoi koodia ja antaa korjauskehotuksia väärin tehdystä tai muotoillusta koodista. Koulutuksessa on kuusi tehtävää, joista tulee koulutuksessa edetessä tehtävä kerrallaan haastavampia.

Tehtävä 1: Git-repositorion kloonaaminen ja sovelluksen käynnistys

Koulutus alkaa kehitysympäristön pystytyksellä ja sovelluksen Git-repositorion kloonaamisella. Sovelluksen kehitystä varten tarvitaan jokin lähdekoodieditori ja koneelle tulee olla asennettuna Node.js ja Git. Itselläni oli tehtäviä tehdessä käytössä VSCode-editori. Git-repositorion kloonaamisen jälkeen täytyy asentaa Node.js:n tarvitsemat node-paketit *"npm install"*-komennolla. Tämän jälkeen sovelluksen pitäisi olla valmis käynnistettäväksi "*npm run dev*"-komennolla (Kuva 11).



KUVA 11. Control Tower käyttöliittymä ensimmäistä kertaa käynnistettäessä

Tehtävä 2: Palkkien värien vaihtaminen

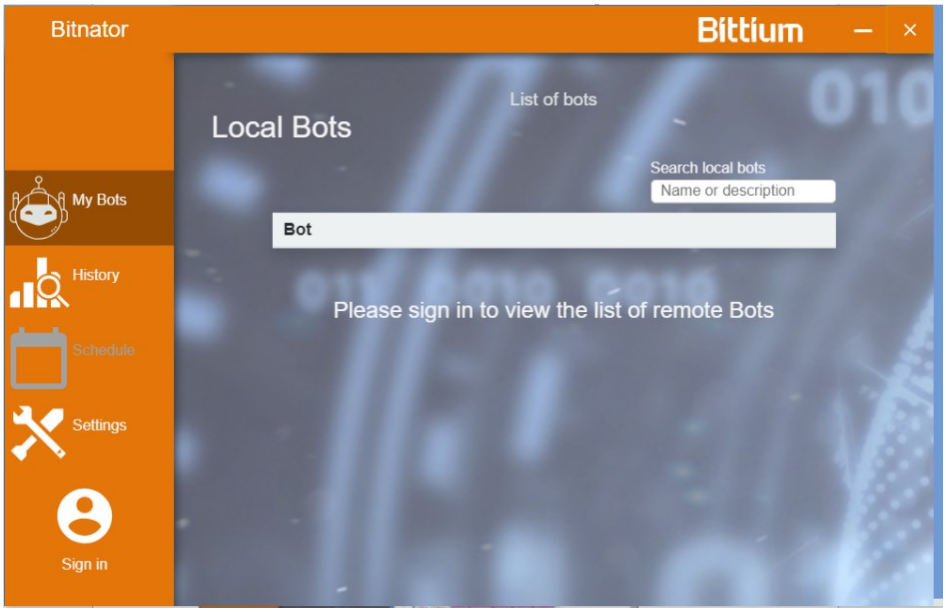
Toisen tehtävän tavoitteena on vaihtaa sovelluksen sivu- ja yläpalkin väriä sekä väriä, joka muuttuu hiirtä leijuttamalla sivu- ja yläpalkkien elementtien päällä. Värit ovat määritelty sovelluksen CSS-tiedostossa, jonka löytyessä värien vaihto on helppo tehtävä. Tehtävänannossa on määritelty värit, joita tulisi käyttää (Kuva 12).

```
:root {
    --frame-color: #E37509;
    --frame-button-hover: #964D03;
```

KUVA 12. Värit määritellään CSS-muuttujiin

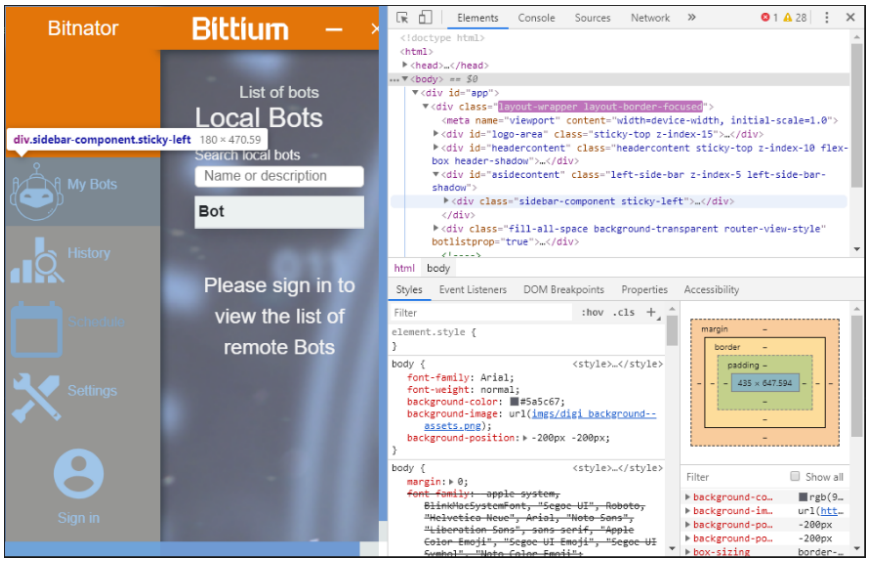
Tehtävä 3: Sivupalkin elementtien järjestyksen vaihto

Kolmannessa tehtävässä perehdytään sivupalkin toimintaan ja sen elementteihin. Tarkoitus on vaihtaa elementtien järjestystä siten, että “Sign In”-elementti siirretään listan pohjimmaiseksi ja listan elementit asettuvat palkin pohjalle (Kuva 13).



KUVA 13. Näkymä edellisen tehtävän värimuutoksien ja "Sign In"-elementti siirtämisen jälkeen

Kuten edellisessä tehtävässä, tämäkin tehtävä vaatii muutoksia HTML- ja CSS-koodiin. Tehtävänannossa ei anneta valmiiksi muokattavaa komponenttia, joten oikeaa komponenttia hakiessa koulutuettavalle tulee projektin tiedosto- ja komponenttirakenne tutuksi. Muokattavan komponentin löytää helposti käyttämällä Electronin mukana tulleen Chromiumin kehittäjätyökalua ja sen elementtihakemistoa (Kuva 14).



KUVA 14. Kehittäjätyökalun elementtihakemisto

Tehtävä 4: Uuden elementin luonti sivupalkkiin

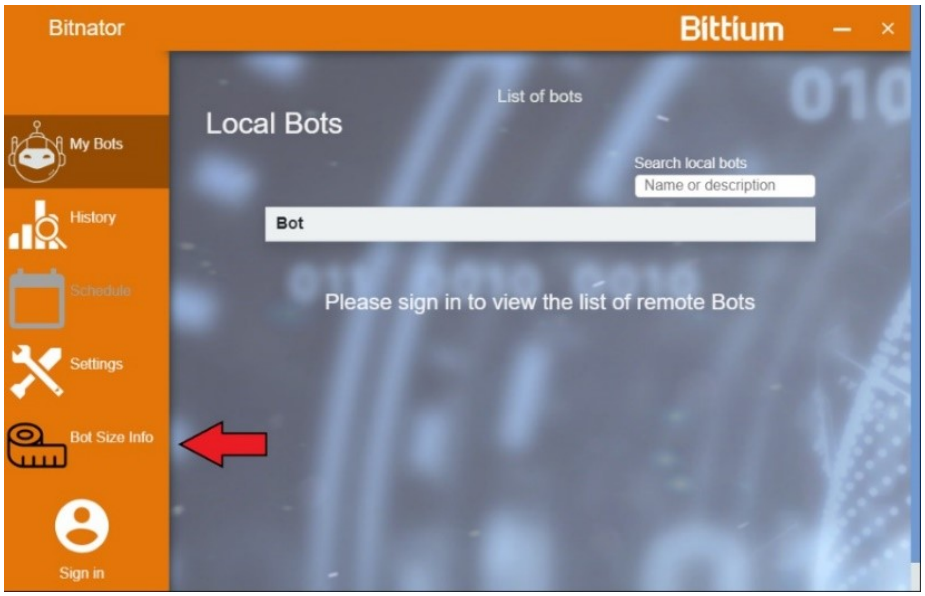
Neljännessä tehtävässä luodaan uusi elementti sivupalkkiin nimeltä “Bot Size Info”. Elementillä ei tarvitse olla vielä click-eventiä näkymän vaihtamiseksi.

Sovelluksen sivupalkin toimintoja käsittelevässä komponentissa on matriisi elementeistä. Matriisin elementeille on asetettu muuttujia, joille asetetaan elementin id, palkissa näkyvä teksti, elementin käyttämä kuva ja kaksi boolean-muuttujaa, joilla säädellään elementin näkymää sekä elementin käytettävyyttä (Kuva 15).

```
{
    id: 'botSizeInfo',
    text: 'Bot Size Info',
    imagePath: 'static/measuring-tape.png',
    selected: false,
    disabled: false,
},
```

KUVA 15. Elementti tietoineen matriisiin sijoitettuna

Vue mahdollistaa template-syntaksillaan Vue-instanssin datan sitomisen renderöityyn DOMiin. Tätä käytetään hyväksi tässäkin tehtävässä renderöidessä elementit sivupalkkiin. Matriisi loopataan Vue-templaatissa, joka renderöi elementit sivupalkkiin (Kuva 16).



KUVA 16. Uusi elementti Control Tower -näkymässä

Tehtävä 5: Uuden Vue-komponentin luonti

Viides tehtävä on jatkoa tehtävälle neljä. Viidennessä tehtävässä edellisessä tehtävässä luodulle sivupalkkielementille luodaan toiminnallisuus. Elementtiä klikatessa näkymän tulisi vaihtua ja näkymään luodaan HTML <table>-taulukko tehtävänannossa määritellyn JSON-matriisin ja sen sisällä olevien JSON-objektien perusteella (Kuva 17). JSON-objektit pitävät sisällään arvon ”Description”, mutta tässä tehtävässä arvo jätetään HTML-taulukosta pois. Description-arvo tulee peliin seuraavassa tehtävässä.

```
const data = [
    {
        'Size category': 'Intangible',
        Size: 'a few electrons',
        Examples: 'RPA bot',
        Description: 'A bot that does well-defined and repetetive tasks for humans',
    },
    {
        'Size category': 'Tiny',
        Size: '2µm - 10 µm',
        Examples: 'Nanobot in blood stream',
        Description: 'A medical bot that can detect irregularities in a blood stream',
    },
    {
        'Size category': 'Medium',
        Size: '25 cm - 40 cm',
        Examples: 'A vaccuum bot',
        Description: 'A bot that cleans the floors. Pets either hate or love these.',
    },
    {
        'Size category': 'Medium',
        Size: '25 cm - 40 cm',
        Examples: 'A lawn mower bot',
        Description: 'A bot that mows the lawn.',
    },
    {
        'Size category': 'Large',
        Size: '50cm x 200 cm',
        Examples: 'Atlas by Boston Dynamics',
        Description: 'Who knows what they are up to?',
    },
];
```

KUVA 17. JSON-matriisi objekteineen

Jotta JSON-objekti saadaan luotua HTLM-tauluksi, JSON-matriisia täytyy loopata. Loop luo ensin HTML-taulun ja sen otsikot, jonka jälkeen rivi kerrallaan loop luo alemmat solut oikean otsikon alle (Kuva 18).

```
const table = document.createElement('table');
table.setAttribute('id', 'my-table');

let tr = table.insertRow(-1); // TABLE ROW.

for (let i = 0; i < col.length; i++) {
    const th = document.createElement('th'); // TABLE HEADER.
    th.innerHTML = col[i];
    tr.appendChild(th);
}

for (let i = 0; i < data.length; i++) {
    tr = table.insertRow(-1);
    for (let j = 0; j < col.length; j++) {
        const tabCell = tr.insertCell(-1);
        if (col[j] === 'Description') {
            continue;
        }
        tabCell.innerHTML = data[i][col[j]];
    }
}
```

KUVA 18. For-loop JSON-matriisin parsimiseksi

JSON-objektissa muuttujien nimet ovat taulun otsikoita ja muuttujien arvot listataan tauluun oman otsikkonsa alle (Kuva 19). Koska tässä tehtävässä Description-kenttä jätettiin pois, karsitaan se loop-vaiheessa pois.

Size category	Size	Examples
Intangible	a few electrons	RPA bot
Tiny	2µm - 10 µm	Nanobot in blood stream
Medium	25 cm - 40 cm	A vaccuum bot
Medium	25 cm - 40 cm	A lawn mower bot
Large	50cm x 200 cm	Atlas by Boston Dynamics

KUVA 19. Taulukko rederöitynä Control Toweriin

Tehtävä 6: Vue-komponentin laajennus

Kuudennessa tehtävässä taulun toiminnallisuuteen lisätään JSON-objektien description-muuttuja. Taulun tulisi toimia siten, että riviä klikatessa riville ja JSON-

objektille aiemmin määritelty description-arvo tulee näkyviin tauluun (Kuva 20). Lisäksi rivin tulisi muuttaa väriä, kun rivi on valittuna ja riviä on klikattu.

Size category	Size	Examples	Description
Intangible	a few electrons	RPA bot	A bot that does well-defined and repetetive tasks for humans
Tiny	2µm - 10 µm	Nanobot in blood stream	
Medium	25 cm - 40 cm	A vaccuum bot	
Medium	25 cm - 40 cm	A lawn mower bot	
Large	50cm x 200 cm	Atlas by Boston Dynamics	

KUVA 20. Ensimmäistä taulukon riviä klikattu

Tätä varten toiminnallisuus tarvitsee siis HTML-taulun riville määritellyn click-eventin (Kuva 21). Loopin sisälle määritelty click-event on asetettu HTML-taulun rivikomponentti tr:lle, jossa pystyakseli asetetaan muuttujaan ja muuttujalle annetaan for-loopin indeksin avulla oikea description-muuttujan arvo.

```
tr.onclick = function () {
    const cell = this.getElementsByTagName('td')[3];
    cell.innerHTML = data[i].Description;
};
```

KUVA 21. Taulukon riville määritelty click-event

Jotta rivin väri vaihtuu sitä valittaessa ja hiirtä sen päällä leijuttaessa, tarvitaan muutoksia CSS:ään. Tyylittelyt ovat sijoitettu Vue-komponentissa <style>-elementtiin (Kuva 22).

```
<style>
    tr:hover,
    tr:focus-within {
        background: #88b7e298;
        outline: none;
        cursor: pointer;
    }
```

KUVA 22. CSS-tyylittelyt <style>-elementissä

5.4 Robot Framework -koulutus

Robot Framework -koulutuksessa tutustutaan, miten Bitnatorissa käytettäviä botteja kehitetään. Koulutuksessa on kaikkiaan kuusi tehtävää, jotka käyvät läpi bottien kehityksen eri vaiheet Bitnator-projektissa. Koulutuksen päätteeksi koulutettavalla tulisi olla tehtävissä määriteltyjen toiminnallisuuksien mukaisesti kehitetty käyttövalmis botti.

Bottien kehityksessä käytetään Robot Frameworkia. Robot Framework on avoimen lähdekoodin työkalupakki, jota voidaan käyttää ohjelmistoprosessien testaamiseen ja automatisointiin. Robot Frameworkia käytetään laajasti testityökaluna hyväksyntätestipohjaiseen kehitykseen eri sovelluksissa ja yhä enemmän myös robottiprosessiautomaatioon. (Robocorp 2021.)

Botit ja Robot Framework käyttävät Bittiumin itse kehittelemiä kirjastoja tai johdannaisia olemassa olevista kirjastoista. Tällä tavalla Bitnator-projektissa käytettävien bottien kehitys on joustavampaa sekä botit saadaan räätälöityä parhaiten vastaamaan asiakkaiden ja Bittiumin vaatimuksia. Kirjastot asennetaan lokaalisti kehitysympäristöön Confluence-sivustolle tekemäni ohjeistuksen mukaisesti.

Tehtävä 1 ja 2: Botin dokumentaatio

Bottien kehitys alkaa dokumentaatiolla. Jokaiselle bottien uudelle käyttötapaukselle luodaan lyhyt kuvaus siitä, mitä botti tekee, miten botti toimii ja mitä varten botti luodaan. Kuvauksen lisäksi botin toiminnasta kehitellään vuokaavio, jonka pohjalta botin kehitystä aletaan toteuttamaan.

Tässä tehtävässä luodaan botille edellä mainitun mukainen kuvaus sekä botin toiminnallisuutta mukaileva vuokaavio. Kuvaus ja vuokaavio dokumentoidaan tehtävien hallinnassa käytettyyn projektinhallintatyökaluun Jiraan.

Tehtävä 3: Kehitysympäristön luonti ja botin ajo

Kolmannessa tehtävässä luodaan kehitysympäristö bottien kehittämistä varten. Kehitysympäristössä käytetään VSCodea, Gitiä sekä Pythonia. Bitnator-projektin botit käyttävät Bittiumin omaa Pythonin paketti-indeksiä kolmannen osapuolen

Python -kirjastojen säilyttämiseen sekä bottien vaatimien tiettyjen kirjastoversioiden käyttämisen sallimiseen. Paketti-indeksi voidaan määritellä yksittäisellä pip-komennolla ”--index-url”-argumentin avulla (Kuva 23).

```
$ pip install package --index-url https:
```

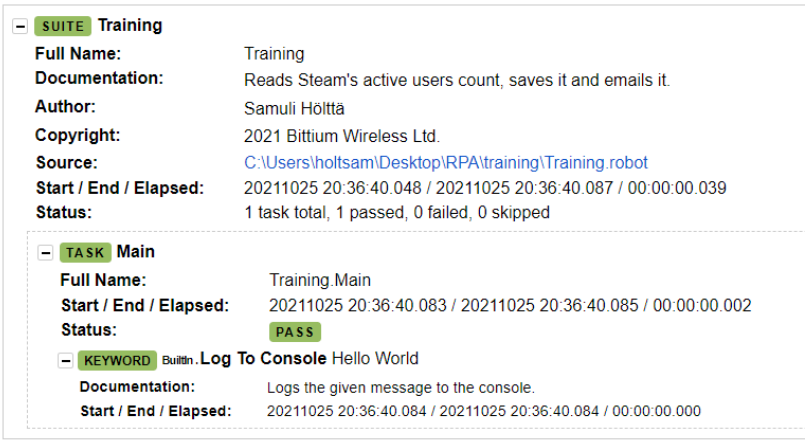
KUVA 23. Pip-komento paketti-indeksin määrittämiselle

Botille luodaan tehtävien ajaksi oma Git-repositorio. Repositorioon kopioidaan Gitistä löytyvä bottitemplaatti, joka sisältää pohjan botin kehitykselle. Templaatin kopioinnin jälkeen botin kansioon luodaan itse botin toiminnallisuutta ohjaava robot-tiedosto. Botille luodaan Pythonin virtuaalinen kehitysympäristö, jonne asennetaan kaikki botin tarvitsemat kirjastot ja jossa bottia ajetaan. Kehitysympäristön pystyttämisen jälkeen testataan sen toiminta lisäämällä konsolilokitus robot-tiedostoon (Kuva 24) ja ajamalla botti.

```
*** Settings ***
Documentation               Training bot

Metadata    Author          Samuli Hölttä
Metadata    Copyright       2021 Bittium Wireless Ltd.

*** Variables ***

*** Keywords ***

*** Task ***
Main
    Log To Console    Hello World
```

KUVA 24. "Hello World"-konsolilokitus robot-tiedostossa

Robot Framework luo botin ajosta suorituslokeja, jossa nähdään robot-tiedoston sisältö sekä Robot Frameworkin suorittamat testit ja niiden tulokset. Suorituslokit generoidaan HTML-tiedostoksi, jonka saa auki selaimella (Kuva 25).



KUVA 25. Selaimeen avattu Robot Frameworkin suoritusloki

Tehtävä 4: Datan luku selaimelta

Neljännessä tehtävässä on tarkoitus lukea selaimelta ja tehtävänannossa määritellyltä sivustolta dataa. Tässä tapauksessa haettava data on Steam-sivustolta Steamiin sillä hetkellä kirjautuneiden käyttäjien määrä. Sivustolta luettu data tulisi asettaa johonkin muuttujaan.

Tehtävässä käytetään Bittiumin kehittelemää selaimen automatisointikirjastoa, joka perustuu Robot Frameworkin tukemaan kolmannen osapuolen selainkirjastoon Seleniumiin. Tässä tehtävässä käytetään kolmea funktiota. Yhdellä avataan selain, toisella avataan haluttu sivusto ja kolmannella haetaan halutun datan sisältävä HTML-elementti. Sivuston avaavan funktion parametriksi sivuston URL-osoite. HTML-elementin etsivä funktio tarvitsee kaksi parametria: paikannintunnuksen sekä paikannintunnuksen tyypin (Kuva 26).

```
*** Variables ***
${URL}=                 https://store.steampowered.com/stats/Steam-Game-and-Player-Statistics?l=english
${LOCATOR_TYPE}=        class name
${LOCATOR}=             statsTopHi

*** Task ***
Main
    Firefox.Setup Private
    Firefox.Open URL    ${URL}
    ${Users_element}=    Firefox.Wait For Element    ${LOCATOR_TYPE}    ${LOCATOR}
```

KUVA 26. Muuttujien ja funktioiden määritys robot-tiedostossa

Paikannintunnuksen ja sen tyypin löytää selaimen kehittäjätyökalun elementtihakemiston avulla. Kun tiedetään mitä sivustolta halutaan hakea, elementtihakemistossa navigoimalla halutun datan sijaintiin saadaan paikannintunnuksen ja sen tyypin nimet. Tässä tapauksessa tyyppi on ”class name” ja ”statsTopHi” (Kuva 27).

```
<span class="statsTopHi">24,056,252</span>
```

KUVA 27. Elementtihakemistosta haettu elementti, joka pitää sisällään halutun datan

Kirjaston avuksi luodaan myös oma Python-moduuli, jonne voidaan luoda funktioita eri käyttötarkoituksiin. Tässä tehtävässä olen luonut funktion, joka muuttaa kirjaston avulla saadun halutun datan sisältävän HTML-elementin tekstiksi (Kuva 28). Moduulin funktiota käytetään robot-tiedostossa ja asetetaan funktion palauttama arvo muuttujaan.

```
def get_users_count(self, element):
    ''' Reads Steam's active users count.

    Arguments:
        element: Element where the user count is stored and read from

    Returns:
        Steam's active users count value
    '''
    self.users_count = element.text
    return self.users_count
```

KUVA 28. Funktio, joka muuttaa elementin sisällön tekstiksi

Tehtävä 5: Datan tallennus tekstitiedostoon

Viidennessä tehtävässä tallennetaan edellisessä tehtävässä haettu data tekstitiedostoon. Käytössä ei ole kirjastoja, joissa olisi valmiita funktioita tekstitiedostojen luomiseen, joten apuna käytetään edellisessäkin tehtävässä käytettyä Python-moduulia. Loin moduuliin funktion, jossa parametrina toimii tallennettava data (Kuva 29).

```python
def save_data(self, data):
    ''' Saves `data` to a text file.

    Arguments:
        data: Data to write to the text file
    '''
    self.file_name = os.path.join(self.bot_temp, "users_count.txt")
    self.txt_file = open(self.file_name, "w")
    self.txt_file.write(f"Steam's active users count: {data}")
    self.txt_file.close()
```

KUVA 29. Funktio datan tallentamiseen tekstitiedostoksi

Edellisessä tehtävässä data asetettiin muuttujaan robot-tiedostossa, joten tätä muuttujaa voi käyttää moduulista tuodun funktion parametrina (Kuva 30). Funktio kirjoittaa tekstitiedoston bottikansion juuresta löytyvään temp-kansioon.

```
*** Task ***
Main
    Firefox.Setup Private
    Firefox.Open URL    ${URL}
    ${Users_element}=    Firefox.Wait For Element    ${LOCATOR_TYPE}    ${LOCATOR}
    ${Value}=    SteamData.Read Steam User Count    ${Users_element}
    SteamData.Save Data    ${Value}
```

KUVA 30. Elementistä saatu data (Value) asetetaan moduulissa luodun funktion parametriksi

Tehtävä 6: Datan lähetys sähköpostilla

Viimeisessä tehtävässä selaimesta luettu sekä tekstitiedostoon tallennettu data lähetetään sähköpostilla. Sähköpostilähettämiseen käytetään Bittiumin kehittämää SMTP-kirjastoa, joka perustuu Pythonin smtplib-kirjastoon. Sähköpostissa tulee olla tekstimuodossa neljännessä tehtävässä luettu data sekä liitteenä viidennessä tehtävässä luotu tekstitiedosto (Kuva 31).

KUVA 31. Sähköpostinäkymä botin lähettämästä viestistä

Sähköpostin viesti muodostetaan aiemmin tallennetusta tekstitiedostosta. Loin moduuliin funktion, joka avaa määritellystä polusta tekstitiedoston, lukee tekstitiedon sisällön ja palauttaa sen (Kuva 32). Funktion palauttavaa arvoa käytetään sähköpostin viestinä seuraavassa kappaleessa käsiteltävässä sähköpostin lähetysfunktiossa (Kuva 33).

```
def read_text_file(self):
    ''' Reads a text file from the temp folder.

    Returns:
        Content of the text file
    '''
    self.file_to_read = self.bot_temp + "users_count.txt"
    with open(self.file_to_read) as f:
        content = f.read()
        f.close()
    return content
```

KUVA 32. Funktio tekstitiedoston sisällön lukemiseksi

Tässä tehtävässä SMTP-kirjastoa suoraan robot-tiedostossa käyttämisen sijaan päätin käyttää kirjastoa aiemmin luodussa Python-moduulissa. Loin moduuliin funktion, joka perii SMTP-kirjastosta sähköpostin lähetysfunktion ”send_mail”. Lähetysfunktio tarvitsee parametreikseen vastaanottajan ja lähettäjän sähköpostiosoitteet, sähköpostin otsikon, itse viestin sekä sähköpostin liitteen (KUVA 32). Funktion parametrit viestiä ja liitettä lukuun ottamatta on määritelty moduulin __init__-funktiossa. Vesti tulee moduuliin luodun funktion ”email_function” parametrista ”message”, joka saadaan käyttämällä edellisen kappaleen tekstitiedoston lukufunktiota (Kuva 32). Liite saadaan __init__-funktioon määritellyn polun ”bot_temp” ja tekstitiedoston nimen muodostamasta muuttujasta ”file_to_send”.

```
def email_function(self, message):
    ''' Emails a `message` and a text file from temp file as an attachment.

    Arguments:
        message: Message to send
    '''
    self.file_to_send = self.bot_temp + "users_count.txt"
    self.smtp = SMTP(self.host, self.port)
    self.smtp.send_mail(self.sender_mail,
        self.receiver_mail,
        self.mail_subject,
        message,
        attachment = self.file_to_send
        )
```

KUVA 33. Funktio, jota käytetään sähköpostin lähettämiseen

Lisäksi moduuliin on luotu funktio, joka poistaa temp-kansioon luodun tekstitiedoston. Funktio ensin tarkistaa löytyykö tiedostoa ja sen löytyessä poistaa sen (Kuva 34).

```
def clean_temp_files(self):
    '''
    Remove temporary files.
    '''
    if os.path.isfile(self.bot_temp + "users_count.txt"):
        os.remove(self.bot_temp + "users_count.txt")
```

KUVA 34. Funktio, joka poistaa tekstitiedoston sähköpostin lähettämisen jälkeen

Tehtävissä käytetyt kirjastot, luodut moduuli ja sen funktiot yhdessä robot-tiedostossa luovat botin toiminnallisuuden. Ajettaessa, botti aukaisee siis selaimen, avaa määritellyn URL-osoitteen, hakee oikean datan sisältävän HTML-elementin, tallentaa elementistä luetun datan tekstitiedostoon, lähettää tekstitiedoston sisällön viestinä sekä tekstitiedoston liitteenä sähköpostilla ja poistaa luodun tekstitiedoston (Kuva 35).

```
*** Variables ***
${URL}=                 https://store.steampowered.com/stats/Steam-Game-and-Player-Statistics?l=english
${LOCATOR_TYPE}=        class name
${LOCATOR}=             statsTopHi

*** Task ***
Main
    Firefox.Setup Private
    Firefox.Open URL    ${URL}
    ${Users_element}=    Firefox.Wait For Element    ${LOCATOR_TYPE}    ${LOCATOR}
    ${Value}=    SteamData.Read Steam User Count    ${Users_element}
    SteamData.Save Data    ${Value}
    ${Message}=    SteamData.Read Text File
    SteamData.Email Function    ${Message}
    [Teardown]    SteamData.Clean Temp Files
```

KUVA 35. Lopullinen robot-tiedosto muuttujineen ja funktioineen

5.5 Koulutuksien arviointi

5.5.1 Arviointikriteerit

Jenkins-koulutuksessa käytännön tekemistä on vähän, joten konkreettista arvioitavaa ei juurikaan ole. Koulutuksen arviointi voisi tapahtua koulutettavan ja kou-

lutuksesta vastaavan RPA-tiimin jäsenen kesken siten, että koulutettava ja koulutusvastaava käyvät yhdessä koulutuksessa käytäviä asioita läpi. Koulutusta voisi käydä läpi pala kerrallaan koulutuksen sisältöä mukaillen: commit-flow, bot-jobien hallinta, bot-nodejen hallinta. Samalla voisi pureutua vähän syvemmin koulutusvastaavan kanssa Jenkins-putkeen ja koulutettava voi esittää kouluttajalle koulutuksen aikana nousseita kysymyksiä. Koulutuksen jälkeen koulutettavalla tulisi olla pinnallisin puolin selkeä kuva Botnator-projektin Jenkins-kokonaisuudesta.

Control Tower- ja Robot Framework -koulutuksen tehtävät ovat pitkälti ohjelmointipainoisia, joten koulutuksien arvioinnissa voitaisiin pureutua koodiin esimerkiksi tarkastelemalla koodin rakennetta, laatua ja menetelmiä miten tehtävät on ratkaistu. Arvioinnissa koodin katselmointiin voidaan käyttää Gerrit-työkalua, jolla ohjelmistoon tehdyt muutokset ovat helposti paikannettavissa.

Koodin rakenteessa huomioitavia asioita voisivat olla esimerkiksi sisennykset ja ylimääräiset rivinvaihdot, jotka ovat Gerritissä helposti havaittavissa. Koodin laadullisia määritelmiä voisivat olla esimerkiksi, että koodi tekee mitä sen kuuluukin, koodin tyyli on johdonmukainen, koodia on helppo ymmärtää sekä koodi on helposti muokattavissa.

Kuten missä tahansa, ratkaisumenetelmiä tehtäviin voi olla monia. Koulutuksien tehtävien ratkaisumenetelmien arvioinnissa voitaisiin keskittyä esimerkiksi siihen, onko toimiva ratkaisu saavutettu ohjelmassa käytettävien teknologioiden määrittelemien hyvien toimintatapojen mukaisesti. Esimerkiksi Control Towerissa käytettävä Vue.js eroaa monella tapaa tavallisesta JavaScript- sekä HTML-ohjelmoinnista, joten ratkaisun yhteensopivuus Vue.js:n kanssa voi olla yksi arvioitavista aiheista, kun taas Robot Framework -koulutuksessa voidaan keskittyä Robot Frameworkin kehityksessä käytettäviin eri käytännön toimintatapoihin.

5.5.2 Arviointimallit

Koulutuksien arvioinnissa käytetään apuna arviointimalleja, joiden avulla saadaan kartoitettua konkreettisesti koulutettavan osaaminen. Mallien avulla saadaan selville koulutettavan lähtötaso ja sitä kautta ohjattua uudet työntekijät heille

sopiviin tehtäviin. Arviointimalleina käytetään taulukkoja, jotka on määritelty koulutuskohtaisesti. Koulutettava luokitellaan arviointimallin kriteerien perusteella yhteen kolmesta taitotasosta: noviisi, junior ja senior.

Jenkins-koulutus

Konkreettisen tekemisen vähyyden vuoksi Jenkins-koulutuksessa arviointi perustuu lähinnä koulutettavan ja kouluttajan välillä käytyyn arviointikeskusteluun. Kouluttaja arvioi työntekijän osaamista keskustelemalla ja kyselemällä kysymyksiä koulutuksen käsittelemistä aiheista. Koulutuksen arvioinnissa otetaan huomioon tehtävien suorittamiseen kulunut aika, arviointikeskustelu sekä avun tarve kouluttajalta. Koulutuksen arvioinnissa käytettävä taulukko ilmenee alla olevasta kuvasta (Kuva 36).

	Taitotaso 3: Senior	**Taitotaso 2: Junior**	**Taitotaso 1: Noviisi**
Kouluttajan avun tarve	Koulutettava ei tarvitse apua tehtävissä etenemiseen lainkaan	Koulutettava ei tarvitse juurikaan apua tehtävissä etenemiseen	Koulutettava tarvitsee huomattavaa apua kouluttajalta tehtävissä etenemiseen
Arviointikeskustelut	Arviointikeskusteluissa koulutettava osaa vastata kysymyksiin ja ymmärtää koulutuksen kokonaiskuvan	Arviointikeskusteluissa koulutettava osaa vastata oaan kysymyksistä ja ymmärtää osan asioista	Arviointikeskustelussa ilmenee koulutettavalla olevan vaikeuksia asioiden ymmärtämisessä
Tehtäviin käytetty aika	Koulutettava saa suoritettua tehtävät ripeästi	Koulutettavalla tehtävien suoritukseen käytettävä aika on kohtuullinen	Koulutettavalla tehtävien suoritukseen käytettävä aika venyy

KUVA 36. Jenkins-koulutuksen arviointitaulukko

Control Tower- ja Robot Framework -koulutus

Control Tower- ja Robot Framework -koulutukset ovat arviointikriteereiltään samankaltaisia. Molemmat koulutukset painottuvat ohjelmistokehitykseen, joten arvioinnissa kiinnitetään huomiota enimmäkseen koodiin. Arvioinnissa keskitytään koodin laatuun, rakenteeseen sekä tehtävien ratkaisutapaan. Myös avun tarve ja tehtävien ratkaisuun käytetty aika otetaan huomioon arvioinnissa (Kuva 37).

	Taitotaso 3: Senior	Taitotaso 2: Junior	Taitotaso 1: Noviisi
Kouluttajan avun tarve	Koulutettava ei tarvitse apua tehtävissä etenemiseen lainkaan	Koulutettava ei tarvitse juurikaan apua tehtävissä etenemiseen	Koulutettava tarvitsee huomattavaa apua kouluttajalta tehtävissä etenemiseen
Koodin laatu	Koodi toimii kuten pitäkin ja koodi on selkeää ja helposti jatkokehitettävissä	Koodi toimii, mutta koodi on epäselvä tai vaikeasti jatkokehitettävissä	Koodi ei toimi kunnolla, koodi on epäselvä ja vaikea ymmärtää
Koodin rakenne	Koodin rakenne on kiitettävää ja virheitä ei juurikaan ole	Koodin rakenne on hyvä, mutta pieniä virheitä löytyy	Koodin rakenteessa puutteita rivinvaihdoissa ja sisennyksissä ja koodi on rikkonaista.
Ratkaisumenetelmät	Ratkaisumenetelmät noudattavat käytettävien teknologioiden määriteltyjä hyviä toimintatapoja	Ratkaisumenetelmät noudattavat osittain käytettävien teknologioiden määriteltyjä hyviä toimintatapoja	Ratkaisumenetelmät eivät noudata käytettävien teknologioiden määriteltyjä hyviä toimintatapoja
Tehtäviin käytetty aika	Koulutettava saa suoritettua tehtävät ripeästi	Koulutettavalla tehtävien suoritukseen käytettävä aika on kohtuullinen	Koulutettavalla tehtävien suoritukseen käytettävä aika venyy

KUVA 37. Control Tower- ja Robot Framework -koulutuksen arviointitaulukko

6 KOULUTUKSEN JATKOKEHITYS

Koulutus tällaisenaan ei ole vielä viimeinen lopullinen versio, vaan sitä tullaan jatkokehittämään. Opinnäytetyönä rakentunut osuus antaa hyvän pohjan ja valmiuden koulutuksen jatkokehitykselle ja viimeistelylle Bittiumin ja etenkin RPA-projektitiimin tarpeiden mukaiseksi. Koulutusalustalle on jo jatkokehitysideoita ja käyn niitä lyhyesti läpi tässä kappaleessa.

Jenkins-koulutus on tällä hetkellä teoriapainotteinen, jossa käytännön tehtävää ei paljon vielä ollut. Bitnator-projektissa ja koulutuksessa käytettävää Jenkins-alustan rakennetta ja toiminnallisuutta ei olla vielä ehditty optimoimaan lopullisen koulutuskokonaisuuden vaatimalle tasolle. Jatkossa siis Jenkins-alustaa tullaan kehittämään koulutuksen vaatimien tarpeiden mukaisesti.

Control Tower -koulutus antoi hyvät lähtökohdat työpöytäsovelluksen frontend-kehitykseen, mutta tällä hetkellä Control Towerilla ei ole käytössä backendiä tai rajapintoja kommunikoidakseen esimerkiksi Jenkinsin kanssa, joten bottien ajo onnistuu vain lokaalisti asennettujen bottien kanssa. Jatkossa koulutuksen tehtävien kehitysympäristölle luodaan jonkinlainen backend ja rajapinta kommunikoimaan Jenkinsin kanssa ja näin mahdollistetaan bottien ajon myös Jenkinsistä. Control Tower -koulutukselle on myös suunniteltu jo kaksi tehtävää liittyen juuri backendiin ja rajapintoihin.

Robot Framework -koulutuksen jatkokehitys painottuu nykyisessä koulutuksessa luodun botin lisäominaisuuksien kehitykseen. Koulutukselle on jo suunniteltu kaksi uutta jatkotehtävää, joissa luodaan botille kaksi lisäominaisuutta. Toisessa tehtävässä botti laitetaan lukemaan dataa jostain rajapinnasta ja toisessa botti tallentaa lukemansa tiedon Bittiumin tietokantaan.

Lopullinen versio koulutuksesta tulee olemaan yhtenäinen kokonaisuus, jossa kaikki koulutuksen osakokonaisuudet ja niissä käsitellyt asiat linkittyvät yhteen ja antavat kokonaiskuvan RPA-bottien kehityksestä ja käytöstä. Robot Framework -koulutuksessa luodaan botti määriteltyine ominaisuuksineen, joka kulkee Jenkins-koulutuksessa käsitellyn Jenkins-putken läpi, joka taas kommunikoi Control Towerin kanssa.

7 YHTEENVETO

Ohjelmistorobotiikka on koko ajan yleistyvä teknologia, joka tulee olemaan osa monien yritysten liiketoimintaprosessien automatisointia. Ohjelmistorobotiikkaa pidetään helppona ensimmäisenä ponnahduslautana kohti automatisoituja prosesseja yrityksien liiketoiminnassa.

Bittiumilla on käytössä kymmeniä ohjelmistorobotteja erilaisten sisäisten työtehtävien automatisoimiseen ja robotteja kehitetään koko ajan lisää. Ohjelmistorobotit ovat säästäneet paljon aikaa ja vaivaa työntekijöiltä rutiininomaisten ja puuduttavien työtehtävien parissa.

Opinnäytetyön tarkoituksena oli luoda koulutusalusta uusien Bittiumin työntekijöiden kouluttamiseksi. Koulutusalusta osakokonaisuuksineen antaa hyvän kokonaiskuvan ohjelmistorobottien kehityksen prosessista sekä projektityöskentelyn luonteesta Bittiumilla.

Opinnäytetyön toteutukseen käytettävä aika on rajallinen, joten luonnollisesti koulutusalustaa ei saatu tässä ajassa saatettua loppuun ja jatkokehitettävää jäi. Opinnäytetyön tavoitteisiin kuitenkin päästiin ja aikaan saatiin kattava koulutuskokonaisuus, jota on helppo jatkokehittää lisää ja viimeistellä lopulliseen käyttöönottoon. Jokaiselle koulutuksen osakokonaisuudelle on jo suunnitteilla hyviä jatkokehitysideoita, joita tämän opinnäytetyön jälkeen aletaan toteuttamaan.

Ohjelmistorobotiikka oli aiheena minulle uusi, joten suurin osa koulutuksessa käsiteltävistä aiheista eivät olleet minulle tuttuja. Tämä toi mukanaan omia haasteita, mutta toisaalta myös konkretiaa sille, millaiset lähtökohdat kokemattomammalla työntekijällä koulutusta aloittaessa on. Tehdessäni koulutuksille luotuja tehtäviä itse koulutuksen toiminta nähtiin käytännössä. Opinnäytetyön aikana tekemäni dokumentaatio tehtäviin käytettävistä tunneista antaa kuvan, minkä tasoisia tehtävät ovat ja millaisia tehtäviä jatkossa voitaisiin kehittää.

LÄHTEET

Arvo, Rebekka 2019. Ohjelmistorobotiikan käyttö diagnostisten ohjausjärjestelmien automatisoinnissa, Case Verohallinto. Turun yliopisto. Turun kauppakorkeakoulu. Pro gradu -tutkielma. Hakupäivä 12.10.2021. https://www.utupub.fi/bitstream/handle/10024/148070/Arvo_Rebekka_Gradu.pdf?sequence=1&isAllowed=y.

Asatiani, Aleksandre & Penttinen, Esko 2016. Turning robotic process automation into commercial success – Case OpusCapita. Journal of Information Technology Teaching Cases 6 (2), 67–74. Hakupäivä 12.10.2021. Deepdyve-tietokanta. Vaatii käyttöoikeuden.

Bittium 2020. Sales Items. Sisäinen lähde.

Bittium 2021. Tietoa yhtiöstä. Hakupäivä 1.11.2021. https://www.bittium.com/bittium-lyhyesti/tietoa-ja-taloudellisia-lukuja/tietoa-yhtiosta.

Capgemini 2020. The risk of RPA implementation and how to mitigate it. Hakupäivä 8.10.2021. https://www.capgemini.com/fi-en/2020/09/the-risk-of-rpa-implementation-and-how-to-mitigate-it/.

Friberg, Karolina 2021. Ohjelmistorobotiikan hyödyt ja tulevaisuus tilitoimistossa. Hämeen ammattikorkeakoulu. Liiketalouden koulutusohjelma. Opinnäytetyö. Hakupäivä 11.10.2021. https://www.theseus.fi/bitstream/handle/10024/494097/Friberg_Karoliina.pdf?sequence=2&isAllowed=y.

Fujitsu 2018. Ohjelmistorobotti hoitaa MTV:n toistuvat rutiinityöt ja tuo isot säästöt. Hakupäivä 13.10.2021. https://www.net.fujitsu.fi/fi-FI/2018/Ohjelmistorobotti_hoitaa_MTVn_toistuvat_(9834)#.

Heller, Martin 2020. What is Jenkins? The CI server explained. Infoworld. Hakupäivä 22.10.2021. https://www.infoworld.com/article/3239666/what-is-jenkins-the-ci-server-explained.html.

Hyytinen, Eero 2019. Ohjelmistorobotiikka vapauttaa ihmisen tekemään tehtäviä, joissa hän on ylivertainen. Codemen. Hakupäivä 9.10.2021. https://www.codemen.fi/ohjelmistorobotiikka-vapauttaa-ihmisen-tekemaan-tehtavia-joissa-han-on-ylivertainen/.

Hämäläinen, Jarno 2019. Ohjelmistorobotiikka (RPA) osana asiakaspalveluprosessia. Karelia-ammattikorkeakoulu. Tietojenkäsittelyn koulutusohjelma. Opinnäytetyö. Hakupäivä 10.10.2021. https://www.theseus.fi/bitstream/handle/10024/167425/Hamalainen_Jarno.pdf?sequence=2&isAllowed=y.

Kääriäinen, Jukka, Aihkisalo, Tommi, Halen, Marco, Holmström Harald, Jurmu, Petri, Matinmikko, Tapio, Seppälä, Timo, Tihinen, Maarit & Tirronen, Justus 2018. Ohjelmistorobotiikka ja tekoäly - soveltamisen askelmerkkejä. Selvitys- ja tutkimustoiminnan julkaisusarja 65/2018. Valtioneuvoston kanslia. Hakupäivä 15.10.2021. https://julkaisut.valtioneuvosto.fi/bitstream/handle/10024/161123/65-2018-Ohjelmistorobotiikka%20ja%20tekoaly.pdf?sequence=1&isAllowed=y.

Willcocks, Leslie & Lacity, Mary 2015. Robotic processing automation at Telefonica O2. The London School of Economics and Political Science 15 (2), 1-19.

Mänty, Toni 2020. Konkurssivalvonta ohjelmistorobotiikan avulla. Oulun ammattikorkeakoulu. Tietojenkäsittelyn koulutusohjelma. Opinnäytetyö. Hakupäivä 12.10.2021. https://www.theseus.fi/bitstream/handle/10024/337719/M%c3%a4nty_Toni.pdf?sequence=2&isAllowed=y.

Osman, Cristina-Claudia 2019. Robotic Process Automation: Lessons Learned from Case Studies. Informatica Economica 23 (4), 66–75. Hakupäivä 3.10.2021. ProQuest -tietokanta. Vaatii käyttöoikeuden.

Robocorp 2021. Basic concepts of Robot Framework. Hakupäivä 27.10.2021. https://robocorp.com/docs/languages-and-frameworks/robot-framework/basics.

Salminen, Lauri 2018. Ohjelmistorobotiikka työtä tehostamassa. HAMK Unlimited. Hakupäivä 8.10.2021. https://unlimited.hamk.fi/yrittajyys-ja-liiketoiminta/ohjelmistorobotiikka-tyota-tehostamassa/.

Staria 2019. Mitä on ohjelmistorobotiikka. Hakupäivä 3.10.2021. https://staria.com/fi/blogi/mita-ohjelmistorobotiikka/.

Staria 2020. Milloin yrityksen kannattaa investoida ohjelmistorobotiikkaan. Hakupäivä 5.10.2021. https://staria.com/fi/blogi/milloin-yrityksen-kannattaa-investoida-ohjelmistorobotiikkaan/.

Syed, Rehan, Suriadi, Suriadi, Adams, Michael, Bandara, Wasana, Leemansa, Sander J.J, Oyang, Hajo A., ter Hofstede, Arthur H.M., van de Weerd, Inge, Wynn, Moe Thandar & Reijers, Hajo A. 2020. Robotic Process Automation: Contemporary themes and challenges. Computers in Industry. Volume 115. Hakupäivä 14.10.2021. https://www-sciencedirect-com.ezp.oamk.fi:2047/science/article/pii/S0166361519304609?via%3Dihub.

Torikka, Mikko 2019. 5 syytä, miksi ohjelmistorobotiikka epäonnistuu. Tivin verkkolehti 15.2.2019. Hakuipäivä 7.10.2021. https://www.tivi.fi/uutiset/5-syyta-miksi-ohjelmistorobotiikka-epaonnistuu/6f2e3e50-bafd-3e00-9f56-daae207c2439.

Tucci, Linda 2021. Ultimate Guide to RPA (Robotic Process Automation). TechTarget. Hakupäivä 4.10.2021. https://searchcio.techtarget.com/Ultimate-guide-to-RPA-robotic-process-automation.