Tampereen ammattikorkeakoulu

Ohjelmistorobotin toteutus Robocorp-alustalla

Opinnäytetyö

Atte Kivimäki

OPINNÄYTETYÖ
Marraskuu 2024

Tietotekniikan tutkinto-ohjelma
Tietoliikennetekniikka ja tietoverkot

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietotekniikan tutkinto-ohjelma
Tietoliikennetekniikka ja tietoverkot

KIVIMÄKI, ATTE:
Ohjelmistorobotin toteutus Robocorp-alustalla

Opinnäytetyö 29 sivua, joista liitteitä 2 sivua
Marraskuu 2024

Ohjelmistorobotiikka on viime vuosina noussut keskeiseen rooliin monien organisaatioiden sisäisten prosessien päivittämisessä nykyaikaan. Robocorp on yksi uusimmista alustoista, joka tarjoaa avoimen lähdekoodin työkaluja Python-pohjaisen ohjelmistorobotiikan kehittämiseen. Robocorp myös mahdollistaa monipuolisten automaatioratkaisujen luomisen sekä skaalautuvuuden yrityskäytössä.

Opinnäytetyön tavoitteena oli opastaa käyttäjä toteuttamaan ohjelmistorobotti käyttäen Robocorp-alustaa. Työvaiheet on esitetty sekä tekstin että havainnollistavien kuvien avulla helposti ymmärrettävässä muodossa. Opinnäytetyössä toteutettiin automaatio, joka hakee ajankohtaiset tiedot 50 arvokkaimmasta kryptovaluutasta ja koostaa tiedot Excel-taulukkoon lähetettäväksi sähköpostitse valitulle vastaanottajalle.

Opinnäytetyö toimii esimerkkinä siitä, että kuka tahansa pystyy vähäisilläkin ohjelmointitaidoilla toteuttamaan toimivan RPA-ratkaisun. Työssä myös annetaan ohjeita ratkaisuiden toteuttamiseen ja pohtia mahdollisten poikkeustilanteiden käsittelyä.

Asiasanat: rpa, robocorp, python, ohjelmistorobotiikka

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Programme in ICT Engineering
Telecommunications and Networks

KIVIMÄKI, ATTE:
Robotic Process Automation Using Robocorp-Framework

Bachelor's thesis 29 pages, appendices 2 pages
November 2024

In recent years, robotic process automation has become a central component in modernizing the internal processes for many organisations. Robocorp is one of the latest platforms that offers open-source tools for Python-based RPA development, enabling the creation of versatile automation solutions that are scalable for business use.

This thesis is aimed to assist the reader in creating a software robot with the Robocorp platform. Every stage is presented in an easily understandable format, with text and illustrative images. In this work, an automation was developed that retrieves up to date data on the 50 most valuable cryptocurrencies and compiles the information into an Excel spreadsheet, and then sends an email to a designated recipient with the file attached.

This work serves as an example to show that anyone with minimal programming skills can implement a functional RPA solution. Furthermore, the thesis seeks to offer guidance on developing these solutions and addressing possible exceptions.

Key words: rpa, robocorp, python, robotic process automation

SISÄLLYS

ERITYISSANASTO

API	Application Programming Interface eli ohjemointirajapinta, joka mahdollistaa tiedonsiirron eri ohjelmistojen välillä.
Elementti	Sovelluksen käyttöliittymän tietty osa, kuten painike, taulukko tai valikko.
Exception	Poikkeus eli ohjelman ajon aikana tapahtuva virhetilanne.
HITL	Human-in-the-loop, ihmisen syötettä tarvitaan automaation ajon aikana.
IDE	Integrated Development Environment eli ohjelmointiympäristö.
Kirjasto	Kokoelma funktioita, luokkia ja muita ohjelmointirutiineja.
Python	Ohjelmointikieli.
RPA	Robotic Process Automation eli ohjelmistorobotiikka.
Template	Ennalta määritetty malli koodin toistamiseen tehokkaasti.
UI	User Interface eli käyttöliittymä mahdollistaa vuorovaikutuksen käyttäjän ja sovelluksen välillä.
XPath	XML Path Language eli XML kyselykieli.

1 JOHDANTO

Ohjelmistorobotiikan eli RPA:n käyttö on aloitettu jo 1990-luvulla, mutta viimevuosina käyttö on lisääntynyt huomattavasti eri toimialoilla. Ohjelmistorobotiikan tarkoitus on tehostaa työtoimintaa ja vähentää itseään toistavaa rutiinityötä. Tämä mahdollistaa organisaatioille paremman käytössä olevien resurssien hyödyntämisen. Teknologian saavutettavuuden ja kehittymisen kautta heräsi kiinnostus ohjelmoida oma ohjelmistorobotti.

Ohjelmistorobotiikalla saavutetaan suurimmat hyödyt, kun sitä sovelletaan työtehtäviin, jotka ovat toistuvia ja sääntöihin perustuvia. Tyypillisimpiä automatisoitavia työtehtäviä ovat tietojen syöttäminen ja tietojen hakeminen kohdejärjestelmästä. Molemmissa työtehtävissä esiintyy toistuvuutta ja sääntöihin perustumista. Tämän takia, kyseiset työtehtävät ovat sopivia kohteita ohjelmistorobotiikan käytölle.

Tyypillisesti ohjelmistorobotiikan ratkaisuja kehitetään graafisen käyttöliittymän kautta, jossa ei tarvitse itse ohjelmoida toteutettua robottia. Robocorp on kuitenkin tuonut markkinoille kilpailijoistaan poikkeavan ratkaisun, jossa robotti tuotetaan pelkästään koodin avulla.

Opinnäytetyön tavoitteena on opastaa lukija ohjelmoimaan Robocorp-alustalla automaatio. Opinnäytetyössä toteutetun robotin tarkoituksena on kerätä 50 markkina-arvoltaan suurimman kryptovaluutan tiedot ja palauttaa niistä Excel-tiedosto käyttäjälle.

2 OHJELMISTOROBOTIIKKA

Ohjelmistorobotiikka on käännös englannin kielen sanoista Robotic Process Automation, joka voidaan lyhentää termiksi RPA. RPA:n yhteydessä luotuja automaatioita kutsutaan usein joko automaatioiksi tai roboteiksi.

2.1. Mitä on RPA?

RPA on teknologia, joka käyttää ohjelmistorobotteja automatisoimaan liiketoimintaprosesseja kokonaan tai osittain digitaalisissa käyttöliittymissä. Täysin automatisoidut robotit voivat suorittaa toistuvia ja säännönmukaisia tehtäviä digitaalisissa käyttöliittymissä matkimalla ihmisen syötteittä ilman ihmisen osallistumista. On myös mahdollista toteuttaa osittain automatisoitu työnkulku, jossa ihminen suorittaa päätöksentekoa vaativan vaiheen manuaalisesti. Tämän jälkeen robotti suorittaa työn loppuun. Osittaisia automaatioita kutsutaan nimellä Human in the Loop (HITL).

RPA soveltuu erityisen hyvin tehtäviin, jotka ovat säännönmukaisia, toistuvia ja vaativat tietojen siirtämistä tai käsittelyä eri järjestelmissä. Esimerkiksi ihmiskäyttäjä kirjoittaa saadusta sähköpostista tilauksen tiedot tilausjärjestelmään, josta saadaan tilausnumero ja se kirjoitetaan raportointitiedostoon talteen. Robotin kehittäminen työnkulkuun aloitetaan luomalla robotille käyttöoikeudet tarvittaviin järjestelmiin. Tämän jälkeen ohjelmoidaan tarvittavien tietojen haku sähköpostista ja niiden kirjoittaminen tilausjärjestelmään. Syötettyään tiedot robotti kirjoittaa tilausnumeron talteen raportointitiedostoon.

Ohjelmistorobotiikassa on yleistymässä ohjelmointirajapintojen eli API:n käyttö tarvittavien tietojen haussa sekä täytössä. Ohjelmointirajapintoja hyväksi käyttäen voidaan luoda POST- tai GET-pyyntöjä rajapintaan. Näiden avulla saadaan nopeutettua automaatiota ja kasvatettua resilienssiä, koska tarve graafiselle käyttöliittymälle vähenee automaatiossa.

2.2. Ohjelmistorobotiikan hyödyt

RPA robotit suorittavat työtehtävänsä itsenäisesti ja johdonmukaisesti ilman riskiä inhimillisistä virheistä. Tämä parantaa suoritettavan työn laatua, sekä estää virheistä aiheutuneita korjauskustannuksia. Opitut rutiininomaiset työtehtävät, kuten tietojen syöttäminen kohdejärjestelmään ovat alttiita ihmisen tekemille virheille (A. Harley 2017).

Robotit toimivat annettujen sääntöjen ja ohjeiden mukaisesti, jolloin ei ole riskiä ihmisten tuottamista virheistä ja samalla tietojen yhdenmukaisuus paranee. Samalla työtehtävissä tehdyistä toimista jää lokiin tiedot mitä automaatio on tehnyt ja se parantaa läpinäkyvyyttä organisaation eri osien välillä. Tämä on etenkin tärkeä asia tarkasti säännellyillä aloilla kuten terveydenhuollossa ja finanssisektorilla, joissa virheistä voi aiheutua vakavia seurauksia ja sakkoja.

Ohjelmistorobotiikka tehostaa organisaatioiden toimintaa merkittävästi, kun toistuvat ja sääntöihin perustuvat työtehtävät automatisoidaan. Esimerkiksi taloushallinnossa ohjelmistorobotit voivat hoitaa kirjanpidon rutiinitehtäviä, jolloin kirjanpitäjät voivat keskittyä strategiseen analyysiin ja päätöksentekoon. Tämä parantaa organisaation kilpailukykyä ja antaa yrityksille mahdollisuuden skaalata toimintaansa ilman rutiininomaisiin työtehtäviin liittyvää lisätyövoiman tarvetta.

Vähentämällä työtehtäviä, jotka pystytään automatisoimaan RPA:n avulla, voidaan parantaa työntekijöiden psyykkistä hyvinvointia ja vähentää työstä johtuvaa stressiä. Kun rutiinitehtävät jäävät pois jää aikaa uusille uramahdollisuuksille ja urakehitykselle esimerkiksi prosessien hallinnassa ja RPA:n ylläpidossa. Työntekijöiltä kuitenkin oletetaan sopeutumista uusiin monipuolisempiin työtehtäviin (I. Kit 2023).

3 ROBOCORP

Tässä kappaleessa perehdytään opinnäytetyössä käytettäviin Robocorpin työkaluihin, kuten Control Roomiin ja Vaultiin.

3.1. Robocorp Automation Stack

Robocorp on vuonna 2019 perustettu avoimen lähdekoodin työkaluja ja pilvialustaa liiketoimintaprosessien automatisointiin kehittävä startup-yritys (A. Hypén 2020). Robocorp Automation Stackiin kuuluva Robocorp Code on ohjelmointikehys ja -kirjasto, joka tarjoaa Python-pohjaisen kehitysympäristön ohjelmistorobotiikan automaatioille. Sen avulla voidaan rakentaa, hallita ja ohjata ohjelmistorobotteja, jotka suorittavat automaatiotehtäviä kuten tietojenkäsittelyä ja järjestelmien integrointia.

Robocorp Coden käyttöä ohjelmistorobotiikassa helpottaa muihin kilpailijoihin, kuten Blue Prismiin ja UiPathiin, verrattuna helposti saatavilla olevat tuhannet Python-kirjastot. Näiden avulla voidaan suorittaa muilla työkaluilla itse tehtävät logiikat vain importoimalla tarvittava kirjasto.

Mikäli tavoitteena on saada kokemusta automaatioiden kehittämisestä, onnistuu se helposti Robocorpin tuottamilla sertifikaattikursseilla. Kursseilla käytetty Developer-tason sopimus, joka on maksuton, tarjoaa suuren osan Robocorpin palveluista. Kyseiseen tasoon kuuluu esimerkiksi 240 minuuttia ajoaikaa Robocorp Control Roomin kautta, mutta ohjelmointiympäristön eli IDEn kautta voidaan automaatiota ajattaa loputtomiin (Sema4.ai 2024).

3.2. Ohjelmointi Robocorp Codella

Robocorp Code on Visual Studio Code-lisäosa, joka mahdollistaa ohjelmoinnin käyttäen Pythonia tai Robot Frameworkia. Näistä jälkimmäinen on testausautomaatioon tarkoitettu Python pohjainen sovelluskehys. Robocorp ilmoitti 15.2.2024 lopettavansa tuen Robot Frameworkille, kuitenkin tarjoten tietoturvapäivitykset ja virheenpäivitykset 15.2.2025 saakka, jolloin virallinen tuki päättyy.

Uusissa automaatioissa on suositeltavaa ohjelmoida automaatioita pelkästään käyttäen Pythonia ja välttää Robot Frameworkkia. (Sema4.ai 2024.)

Ohjelmointi Robocorp Codella hyödyntäen Pythonia tarjoaa loistavat mahdollisuudet kehittyä RPA:n sekä testi automaatioiden kehityksen osa-alueilla. Pythonin suuren käyttäjämäärän ansiosta ongelmatilanteisiin löytyy paljon vaihtoehtoisia ratkaisuja ja ohjeita, joiden avulla ohjelmointi on sulavampaa ja tehokkaampaa kilpailijoiden graafisia käyttöliittymiä hyödyntäviin ratkaisuihin verrattuna. Pythonin hyvänä puolena voidaan myös pitää selkeää syntaksia, jonka takia voidaan keskittyä luomaan automaation logiikkaa ja saada pienemmässä ajassa kehitettyä toimiva automaatio. Python ei ole kuitenkaan nopein ohjelmointikieli, koska ohjelmistorobotiikassa rajoittavana tekijänä on usein robotin käyttämä graafinen käyttöliittymä. (J. Anderson 2024)

3.3. Robocorp Control Room

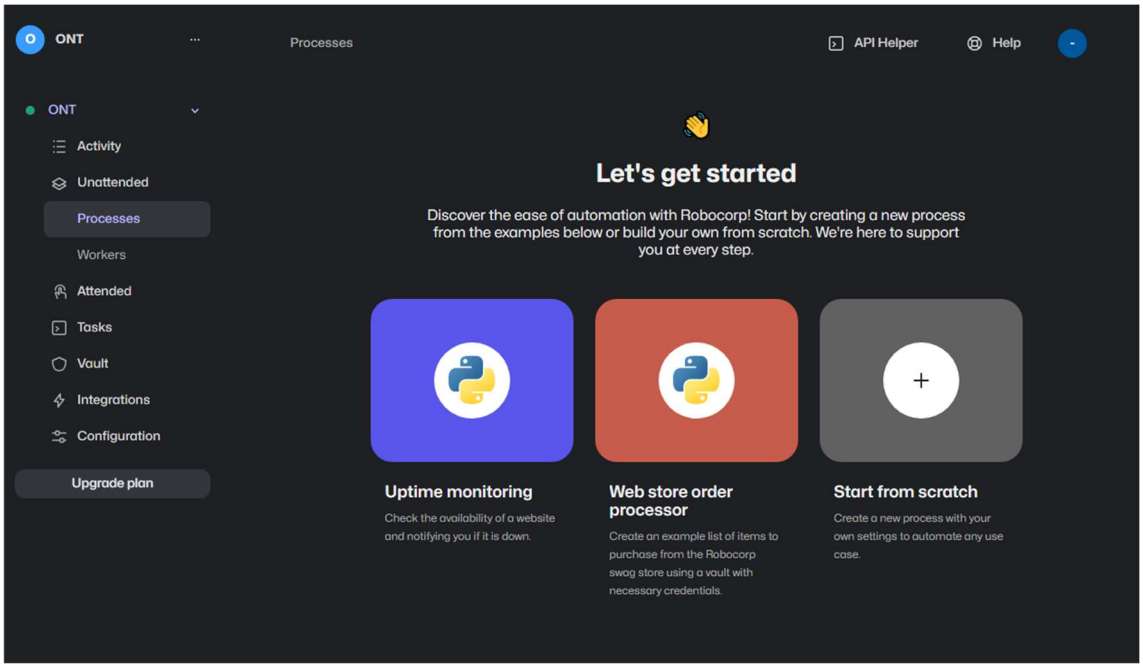
Robocorp Control Room on kätevä tapa jakaa, hallita ja seurata automaatioita reaaliajassa yhdestä keskitetystä komentokeskuksesta. Keskeinen ajatus on tarjota turvallinen, luetettava ja helposti skaalautuva käyttöliittymä automaatioiden orkestrointiin. Control Room on kokonaisvaltainen orkestrointiratkaisu, jonka avulla ei tarvitse ylläpitää ja isännöidä omia palvelimia, tietokantoja tai täysin automaattisia robotteja (Sema4.ai 2022). Tässä kappaleessa esitellään opinnäytetyössä tarvittavat toiminnallisuudet Control Roomista.

3.3.1 Processes (unattended automation)

Processes-näkymässä prosessi tarkoittaa automatisoitavaa työnkulkua, joka koostuu stepeistä eli pienemmistä automaation osista. Yleinen tapa hyödyntää steppejä on jakaa automaatio kehitysvaiheessa tuottaja-kuluttaja-malliin eli Producer-Consumer-malliin. Mallissa Producer-vaihe kerää tarvittavat tiedot ja järjestää ne työjonolle erillisiksi toimeksiannoiksi, josta Consumer-vaihe käsittelee toimeksiannot yksi kerrallaan. Riippuen automaation käyttötarkoituksesta voi olla tarve lisätä myös Reporter-vaihe, joka muodostaa onnistuneesti ja virheellisesti käsitellyistä tapauksista yhteenvedon.

Tästä näkymästä pystytään orkestroimaan automaatioiden toimintaa asettamalla parametreja automaatioiden ajoille, esimerkiksi ajon alkamisajankohta ja päättymisajankohta. On myös mahdollista asettaa automaation ajon tapahtuvan yhdellä tai useammalla resurssilla saman aikaisesti.

Control Roomin tyhjä Processes-näkymä uudelle ONT-nimiselle organisaatiolle on näkyvillä kuvan 1 keskiosassa. Näkymään navigoidaan kuvan vasemmalla puolella olevasta navigointipuusta painamalla Unattended-valikosta Processes-painiketta. Kuvassa näkyvällä välilehdellä näkyisi organisaatiolle saatavilla olevat automaatiot, mutta yhtään automaatioita ei ole tuotu Control Roomiin uudelle organisaatiolle, joten sivu on tyhjä.



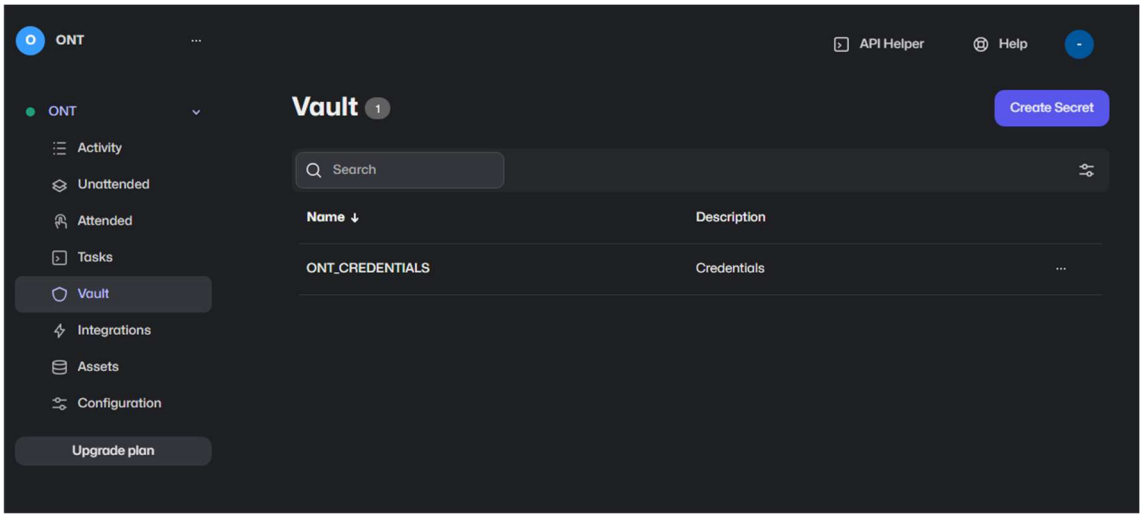
KUVA 1. Robocorp Control Roomin tyhjä Processes-näkymä uudelle ONT-organisaatiolle.

3.3.2 Control Room Vault

Control Room Vaultissa on mahdollista säilyttää tunnistetietoja tai muita arkaluontoisia tietoja, jotka ovat välttämättömiä automaatioiden sujuvalle toiminnalle. Säilytettävien tietojen tietoturvasta huolehditaan useiden tekniikoiden avulla luot-

tamuksen varmistamiseksi. Kaikki tiedot salataan AES-256 – algoritmilla Galois/Counter Mode (GCM)-tilassa. Jokainen säilytettävä tieto salataan yksilöllisellä avaimella, joka on ainutlaatuinen. Luotu ainutlaatuinen avain salataan vielä pääavaimella, tämän jälkeen salattu tieto ja yksilöllinen avain säilytetään tietokannassa (Sema4.ai 2021).

Salattavia tietoja lisättäessä navigoidaan Vault-näkymään painamalla kuvassa 2 vasemmalla puolella näkyvästä navigointipuusta Vault-painiketta. Vault-näkymä löytyy kuvan keskiosasta, jossa näkyy yksi tallennettu salattu tieto nimellä ”ONT_CREDENTIALS” ja kuvauksella ”Credentials”. Olemassa olevaa tietoa pystytään muokkaamaan tai poistamaan painamalla rivin oikealla puolella olevaa kolmea pistettä. Tarvittaessa uuden salatun tiedon pystyy luomaan painamalla Create Secret-painiketta.



KUVA 2. Control Room Vault-näkymä.

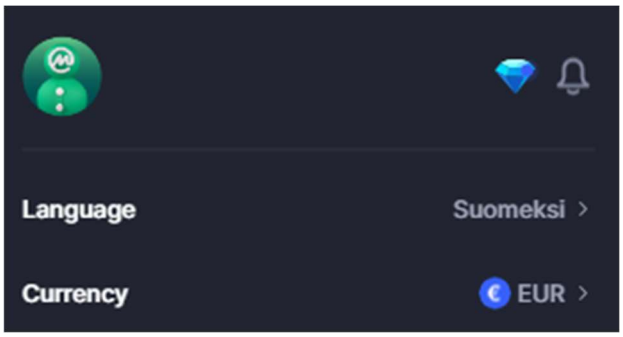
Tunnuksia pystytään hakemaan automaation ajon aikana käyttämällä Robocorp. Vault-kirjaston *get_secret*-metodia. Kun metodia kutsutaan, name-syöttöparametriin määritetään halutun tiedon nimisarakkeen arvo Vaultissa. Automaation ajon aikana on mahdollista muokata vain olemassa olevia tietoja käyttäen *set_secret*-metodia, joka korvaa kaikki arvot syöttöparametriin asetetulla tiedolla (rpaframework.org 2024).

4 AUTOMAATION TOTEUTUS

RPA-ratkaisujen toteutus voidaan jakaa projektin koosta riippumatta kolmeen vaiheeseen. Ensin suunnitellaan ja dokumentoidaan tulevan automaation työnkulku. Seuraavaksi tuotettua dokumenttia käyttäen ohjelmoidaan automaation logiikka ja validoidaan valmiin automaation toiminta asiakkaan kanssa. Viimeisenä vaiheena on automaation siirtäminen tuotantoympäristöön ja tuotanto ajojen käynnistäminen.

4.1. Suunniteltu työnkulku

Tässä opinnäytetyössä toteutettu automaatio navigoi osoitteeseen www.coinmarketcap.com ja kirjautuu Vaultista löytyvillä tunnuksilla sivustolle. Onnistuneen kirjautumisen jälkeen painetaan sivuston oikeasta yläkulmasta asetukset painiketta, avautuu kuvassa 3 näkyvä valikko. Tarkastetaan, että valittu kieli on suomi ja valuutta on euro, jos näin ei ole vaihdetaan kieli ja valuutta oikeiksi.

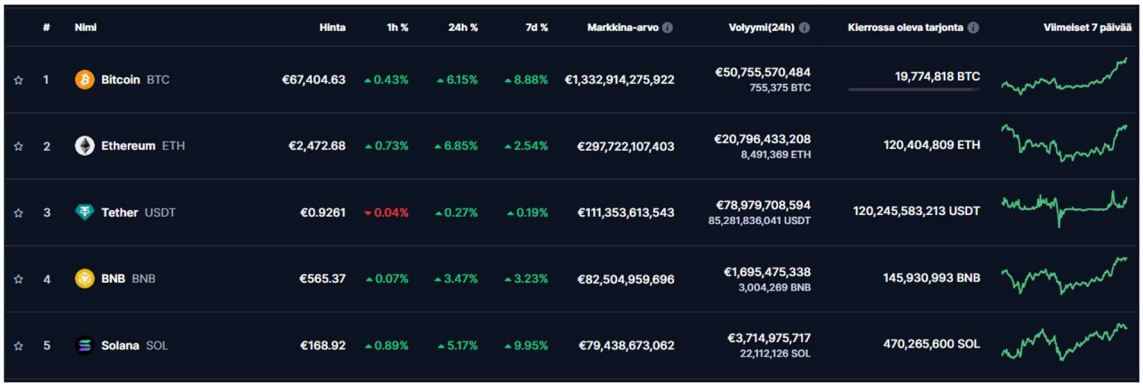


KUVA 3. Kieli ja valuutta valinnat coinmarketcap.com sivustolla.

Palataan aloitusnäyttöön painamalla vasemmassa yläreunassa sijaitsevaa logoa. Seuraavana tehtävänä on tarkistaa kielen ja valuutan vaihdon onnistuminen, jotta automaation lopputuote on halutulla valuutalla. Onnistuneen vaihdon jälkeen jatketaan seuraavaan vaiheeseen. Tietojen vaihtamisen epäonnistuessa, yritetään vaihtoa kolmesti uudelleen.

Seuraavaksi valitaan kuvassa 4 näkyvillä olevan taulukon alapuolelta alasvetovalikosta näkyville 50 riviä. Muutoksen tultua voimaan valitaan taulukon järjestys laskevaksi markkina-arvo sarakkeen mukaan. Taulukon päivittymisen jälkeen

luetaan taulukko automaation muistiin ja poimitaan sarakkeet Nimi, Lyhenne, Hinta ja Markkina-arvo.



#	Nimi	Hinta	1h %	24h %	7d %	Markkina-arvo	Volyymi(24h)	Kierrossa oleva tarjonta	Viimeiset 7 päivää
1	Bitcoin BTC	€67,404.63	0.43%	6.15%	8.88%	€1,332,914,275,922	€50,755,570,484 755,375 BTC	19,774,818 BTC	
2	Ethereum ETH	€2,472.68	0.73%	6.85%	2.54%	€297,722,107,403	€20,796,433,208 8,491,369 ETH	120,404,809 ETH	
3	Tether USDT	€0.9261	0.04%	0.27%	0.19%	€111,353,613,543	€78,979,708,594 85,281,836,041 USDT	120,245,583,213 USDT	
4	BNB BNB	€565.37	0.07%	3.47%	3.23%	€82,504,959,696	€1,695,475,338 3,004,269 BNB	145,930,993 BNB	
5	Solana SOL	€168.92	0.89%	5.17%	9.95%	€79,438,673,062	€3,714,975,717 22,112,126 SOL	470,265,600 SOL	

KUVA 4. Kryptovaluuttahinnat markkina-arvon mukaan coinmarketcap.com sivustolla taulukko muodossa.

Luodaan poimitusta datasta taulukon mukainen Excel-taulukko (ks. Liite 1. Onnistuneen ajon Excel-taulukko). Taulukon luomisen jälkeen lähetetään se sähköpostiviestin liitteenä määritettyyn sähköpostiosoitteeseen. Sähköpostin lähettämisen jälkeen automaatio on suorittanut sille määrätyn työnkulun ja sammuttaa käynnissä olevat sovellukset.

4.1.1 Poikkeustilanteiden käsittely

Automaation kohdatessa vakavan poikkeustilanteen, esimerkiksi järjestelmän kaatumisen, yrittää automaatio uudelleen koko käsittelyä. Mikäli poikkeustilanteena on esimerkiksi napin painallus, joka ei toimi ensimmäisellä yrittämällä, voidaan kokeilla uudelleen painaa nappia.

Tilanteessa, jossa ei uudelleen yrityksistä huolimatta onnistuta suorittamaan suunnitelman mukaista työnkulkua, tulee koodiin määritettyyn sähköpostiosoitteeseen saapua ilmoitus epäonnistuneesta työnkulusta.

4.2. Automaation kehitys

Työssä käytettävä Visual Studio Code eli VS Code on Microsoftin kehittämä integroitu kehitysympäristö eli IDE. Se on kevyt mutta tehokas lähdekoodieditori, joka tukee monia ohjelmointikieliä, kuten Java, C++ ja Python. Useat sen ominaisuudet ovat rakennettu yleisten kehitystehtävien tarpeiden ympärille (M. Heller 2022).

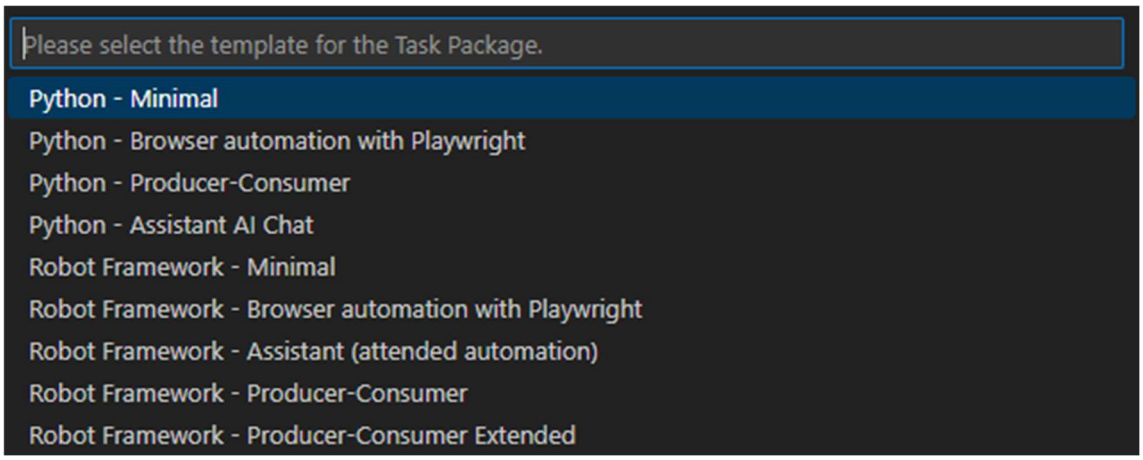
Kaikki funktionaalisuus selaimeen on toteutettu käyttäen Selenium-kirjastoa, joka on avoimen lähdekoodin työkalu, jolla pystytään automatisoimaan selain sovellusten toimintaa. Se toimii kaikilla selaimilla ja suurimmilla käyttöjärjestelmillä, sekä sen skirptit voidaan kirjoittaa useilla eri ohjelmointikielillä, kuten Java ja Python.

4.2.1 Ympäristön perustaminen

Kehitystä ei kannata aloittaa täysin tyhjältä pöydältä vaan on suositeltavaa aloittaa Robocorpin omasta robottimallista eli robot templatesta. Mallin saa luotua avaamalla VS Coden komentopaletin painamalla (”CTRL” + ”Shift” + ”P”) näppäimiä ja valitsemalla kuvassa 5 näkyvillä olevan vaihtoehdon Create Task Package (Robot).

KUVA 5. Robottimallin luominen.

Valinnan jälkeen aukeaa uusi lista, jossa on erilaisiin käyttötarkoituksiin soveltuvia Python ja Robot Framework malleja. Tässä työssä käytetään valmista Python – Minimal-mallia, joka on valittuna kuvassa 6.



KUVA 6. Saatavilla olevat robottimallit lueteltuna.

Näiden vaiheiden jälkeen on luotuna toimiva RPA ympäristö, jossa kehitystä voidaan alkaa toteuttaa.

4.2.2 Funktioiden ohjelmointi

Ennen automaation logiikan kehitystä, tunnistettiin suunnitteluvaiheessa havaitut tarpeelliset elementit. Elementtien tunnistus suoritettiin käyttämällä XML Path Language- eli XPath-lausekkeita. Työssä tarvittavien elementtien tunnisteet järjestetään käyttöliittymän eriävien ikkunoiden mukaan omiin luokkiin eli classeihin. Esimerkiksi käyttöliittymän kirjautumisikkuna ja päänäkymä ovat jaettu LOG_IN_PAGE- ja MAIN_PAGE-luokkiin. Kuvassa 7 on muuttujaan tallennettu sivuston pääsivulta sisäänkirjautumissivulle johtavan painikkeen sijainti XPath-lausekkeena ilmaistuna. Näiden jakaminen erillisiin luokkiin auttaa pitämään koodin luettavampana sekä helpottaa mahdollisien lisäosien kehitystä. Automaatioissa voi olla tarve käyttää samaa elementtiä useassa eri kohdassa työnkulkua. Tällöin organisoidut ja selkeästi nimetyt tunnistetut elementit ovat helposti implementoitavissa.

```
class MAIN_PAGE:
    log_in_page_button = "//div[contains(@class,'Search_mobile-icon')]/following-sibling::div[2]"
```

KUVA 7. Esimerkki luokkien ja XPath-lausekkeiden käytöstä työssä.

Seuraavassa vaiheessa kehitystä lisättiin toiminto selaimen käynnistykseen. Kuvassa 7 on esillä selaimen avaamiseen käytetty funktio *start_browser*. Funktiota

kutsutaan prosessin alussa, sekä määrittelemättömissä virhetilanteissa, joista ei pystytä palautumaan ilman uudelleen käynnistämistä. Automaation tarvitsee myös kirjautua sisään aina selaimen käynnistyessä uudelleen, joten funktioon on lisätty kirjautumisikkunan avaaminen. Funktioon on myös implementoitu sisäistä poikkeuskäsittelyä, joka mahdollistaa kolme yritystä saada selain avattua onnistuneesti.

```
def start_browser():
    for attempt in range(3):
        try:
            browser.open_browser(url=URL.CMC_URL, browser='chrome')
            click_element(MAIN_PAGE.log_in_page_button)
            return
        except:
            browser.close_browser()
            time.sleep(5)
    raise Exception('Selaimen avaamisessa tapahtui virhe.')
```

KUVA 7. Selaimen käynnistys funktio poikkeuskäsittelyllä.

Kuvan 7 *start_browser*-funktiossa kutsutaan kuvassa 8 näkyvää lokaalisti määritettyä *click_element*-funktiota, jonka tarkoituksena on varmistaa elementin olevan valmis vastaanottamaan klikkauksen. Selenium-kirjastosta löytyvä *click_element_when_clickable*-metodi omaa saman funktionaalisuuden. Kehitysvaiheessa huomattiin metodin aiheuttavan poikkeustilanteita, jotka ovat ratkaistu alla olevalla funktiolla. Funktiossa tarkistetaan syöttöparametriksi annetun elementin näkyvyys 5 sekunnin kuluessa tarkistuksen alusta, jos elementti on kohdejärjestelmässä näkyvissä, yritetään painaa sitä. Automaatiota kehittäessä saattaa syntyä edellytys muuttaa elementin näkyvyyden tarkastukseen käytettävää aikaa, joka voidaan suorittaa syöttämällä numeerinen arvo *timeout_seconds*-syöttöparametriin.

```python
def click_element(element_locator:str, timeout_seconds: int = 5):
    start_time = time.time()
    while time.time() - start_time < timeout_seconds:
        if browser.is_element_visible(element_locator):
            try:
                browser.click_element(element_locator)
                return
            except:
                pass
        time.sleep(0.1)
    raise TimeoutError(
        f'Elementtiä {element_locator} ei löytynyt annettuna odotusaikana: {timeout_seconds}
    )
```

KUVA 8. Dynaamisesti elementin näkyvyyttä tarkastava ja painava funktio.

Automaatiolle toteutetut toiminnallisuudet mahdollistavat selaimen käynnistämisen sekä navigoinnin sivustolla määritettyjä elementtejä painamalla. Käytettyyn verkkosivun sisäänkirjautumiseen vaaditaan sähköpostiosoitteen ja salasanan syöttäminen. Tähän tarpeeseen on olemassa valmis metodi Selenium-kirjastosta nimeltään *input_text*. Kuitenkin ohjelmoitaessa automaatiota halutaan olla varmoja syöttämien tietojen oikeellisuudesta, joten luodaan kuvan 9 *input_text_to_field*-funktio. Funktiota kutsuttaessa syöttöparametriksi määritetään tekstikentän elementin tunniste ja syötettävä teksti, jonka jälkeen tarkistetaan onko elementti näkyvissä. Tekstikentän ollessa näkyvissä kirjoitetaan *input_text*-syöttöparametriin määritetty teksti käyttäen *input_text*-metodia, jonka jälkeen luetaan syötetty teksti elementistä. Luetun ja syötetyn tekstin arvojen ollessa identtiset poistutaan funktiosta.

```python
def input_text_to_field(element_locator, input_text):
    for attempt in range(3):
        if browser.is_element_visible(element_locator):
            try:
                browser.input_text(element_locator, input_text)
                read_text=browser.get_element_attribute(element_locator, attribute='value')
                if read_text == input_text:
                    return
            except:
                pass
        time.sleep(1)
    raise Exception(f'Kirjoitettu tieto: {input_text}, luettu tieto {read_text}')
```

KUVA 9. Tekstin syöttäminen tekstikenttään ja tekstin oikeellisuuden varmistaminen.

Seuraavaksi ohjelmoitava toiminto valitsee oikeat valuutan ja kielen vaihtoehdot. Nämä asetukset vaihdetaan samassa sovelluksen ikkunassa, jonka takia toiminnot ovat yhdistetty yhteen funktioon. Kuvassa 10 tarkistetaan, onko valuutaksi valittu euro ja kieleksi suomi. Jos valinnat eivät ole oikein, ne vaihdetaan avaamalla valikko ja valitsemalla oikea vaihtoehto.

```
def choose_currency_and_language():
        try:
            if not browser.is_element_visible(SETTINGS.euro_chosen):
                click_element(SETTINGS.change_currency)
                click_element(SETTINGS.euro_choice)
            if not browser.is_element_visible(SETTINGS.finnish_chosen):
                click_element(SETTINGS.change_language)
                click_element(SETTINGS.finnish_choice)
            return
        except Exception as exception:
            raise Exception(f"Virhe kielen ja valuutan vaihdossa: {exception}")
```

KUVA 10. Kielen ja valuutan vaihto kohdejärjestelmässä.

Kielen vaihdon jälkeen automaatio palaa sovelluksen päänäkymään, jossa kryptovaluuttojen tietoja sisältävän taulukon näkyvillä olevien rivien määrä on oletuksena 100 kappaletta. Tavoitteeksi oli kuitenkin asetettu kerätä vain viidenkymmenen markkina-arvoltaan arvokkaimman kryptovaluutan tiedot. Kuvan 11 *show_50_rows*-funktiossa tarkistetaan onko rivien määrän vaihtamiseen käytetty painike näkyvillä. Onnistuneen tarkistuksen jälkeen avataan rivimäärän valitsemiseen käyttettävä valikko ja valitaan 50 riviä vaihtoehto.

Jos painiketta ei ole näkyvillä on mahdollista, että kielen vaihdon jälkeen sovellus ei oletetusti palannut päänäyttöön. Tällöin suljetaan asetukset-ikkuna ja yritetään rivimäärän valintaa uudelleen. Kaikkiin poikkeustilanteisiin ei pystytä varautumaan RPA-toteutuksissa, mutta kyseinen poikkeustilanteen estäminen oli helposti toteutettavissa.

```
def show_50_rows():
    for attempt in range(3):
        if browser.is_element_visible(MAIN_PAGE.show_rows_button):
            try:
                if browser.get_text(MAIN_PAGE.show_rows_amount) != "50":
                    click_element(MAIN_PAGE.show_rows_button)
                    click_element(MAIN_PAGE.choose_rows_per_page_50)
                    return
            except:
                try:
                    click_element(SETTINGS.close_settings_tab)
                except:
                    pass
        time.sleep(1)
    raise Exception("Ei voitu valita 50 riviä näkyviin. {exception}")
```

KUVA 11. Rivimäärän vaihtaminen.

Koodin seuraavassa osuudessa käsitellään kryptovaluutta tietoja sisältävän taulukon datan lukemista ja suodattamista. Kuvassa 12 määritetään HEADERS-muuttujaan haluttujen taulukon sarakkeiden nimet lista-datatyyppiin ja luetaan taulukko *get_element_attribute*-metodia käyttäen. Seuraavaksi käytetään hyödyksi Pandas-kirjaston *read_html*-metodia, joka muuttaa HTML-muodossa olevan taulukon Dataframe-tietorakenteeksi.

```
def html_to_excel():
    HEADERS = ["Nimi", "Symboli", "Hinta", "Markkina-arvo","Kierrossa oleva tarjonta"]
    table_html = browser.get_element_attribute(MAIN_PAGE.crypto_table,"outerHTML")
    tables = pd.read_html(table_html)
    df = tables[0]
```

KUVA 12. HTML-taulukon lukeminen Dataframeen-osio *html_to_excel*-funktiossa.

Koodin seuraavassa vaiheessa selaimesta luetulle taulokko-datalle suoritettaan siivousta, jotta tuotettu Excel-taulukko on mahdollisimman helppolukuinen. Datasta poistetaan ylimääräiset valuutta-merkit ja etsitään Kierrossa oleva tarjonta-sarakkeesta arvon lopusta kryptovaluutan lyhenne. Taulukossa 1. rivillä kaksi on luettu taulukko data ennen siivousta ja rivillä kolme data on siivotussa muodossa.

TAULUKKO 1. Esimerkki datasta ennen ja jälkeen siivouksen.

Rivi	Nimi	Symboli	Hinta	Markkina-arvo

1	BitcoinBTC		€64,684.84	€1.28T€1,279,205,815,730
2	Bitcoin	BTC	64,684.84	1,279,205,815,730

Kuvassa 13 on päivitetty *html_to_excel*-funktio, johon on lisätty myös Excel-taulukon muodostaminen Pandas-kirjaston *to_excel*-metodia käyttäen. Ennen Excel-taulukon luontia tarkistetaan, sisältääkö Dataframe-tietorakenne 50 riviä ja poistetaan mahdollisesti löytyvä taulukko.

```
def html_to_excel():
    HEADERS = ["Nimi", "Lyhenne", "Hinta", "Markkina-arvo"]
    table_html = browser.get_element_attribute(MAIN_PAGE.crypto_table,"outerHTML")
    tables = pd.read_html(table_html)
    df = tables[0]
    df['Lyhenne'] = df['Kierrossa oleva tarjonta'].str.extract(r'(\b[A-Z]+\b)')
    df['Nimi'] = df.apply(lambda row: str(row['Nimi']).removesuffix(str(row['lyhenne'])), axis=1)
    df['Hinta'] = df['Hinta'].str.replace('€', '')
    df['Markkina-arvo'] = df['Markkina-arvo'].str.lstrip('€').str.split('€').str[1]
    df = df[HEADERS]
    if df.shape[0] == 50:
        output_dir = os.path.join(os.getenv("ROBOT_ROOT", ""), "output")
        file_path = os.path.join(output_dir, "output.xlsx")
        if os.path.exists(file_path):
            os.remove(file_path)
        df.to_excel(file_path, index=False)
    else:
        raise Exception("Dataframe ei sisällä 50 riviä.")
    return df
```

KUVA 13. Päivitettu funktio datan lukemiseen ja Excel-taulukon tuottamiseen.

Ennen päälogiikan rakentamista koodiin lisättiin sähköpostin lähettämiseen tarkoitettu funktio, joka ottaa syöttöparametriksi tiedostopolun Excel-tiedostoon. Kuvassa 14 olevassa funktiossa tarkistetaan onko tiedosto olemassa ja ilmaantuuko sähköpostin lähetyksessä virhetilanne.

```python
def send_email_to_recipent(file_path):
    secrets = vault.get_secret(EMAIL_CREDENTIALS)
    smtp_server = "smtp.gmail.com"
    smtp_port = 587
    sender_email = secrets["USERNAME"]
    password = secrets["PASSWORD"]

    # Luodaan sähköposti
    msg = MIMEMultipart()
    msg['From'] = sender_email
    msg['To'] = RECIPENT_EMAIL
    msg['Subject'] = "Kryptovaluutta taulukko."
    body = "Robotti on suorittanut työnkulun."
    msg.attach(MIMEText(body, 'plain'))

    # Lisätään tiedosto
    if os.path.isfile(file_path):
        attachment = MIMEBase('application', 'octet-stream')
        with open(file_path, 'rb') as attachment_file:
            attachment.set_payload(attachment_file.read())
        encoders.encode_base64(attachment)
        attachment.add_header(
            'Content-Disposition',
            f'attachment; filename={os.path.basename(file_path)}'
        )
        msg.attach(attachment)
        try: # Lähetetään sähköposti
            server = smtplib.SMTP(smtp_server, smtp_port)
            server.starttls()
            server.login(sender_email, password)
            server.send_message(msg)
            return
        except Exception as e:
            raise Exception(f"Failed to send email: {e}")
    else:
        raise Exception(f"File not found: {file_path}")
```

KUVA 14. Sähköpostin lähettäminen.

4.2.3 Pääfunktion toteutus

Robocorpilla tehdyissä automaatioissa pääfunktion eli taskin tarkoitus on koota kaikki erikseen toteutetut toiminnot ja ohjata niiden suorittamista järjestyksessä. Tämä vähentää RPA-prosessien monimutkaisuutta ja tekee niiden hallinnasta joustavampaa. Hyvin suunniteltu päälogiikka helpottaa prosessin ylläpitoa, sillä sitä voi päivittää tai laajentaa muuttamalla yksittäisiä funktioita ilman, että koko prosessia tarvitsee suunnitella uudelleen.

Edellisessä kappaleessa luodut funktiot on toteutettu ja testattu erillisinä osina ja tässä kappaleessa ne yhdistetään pääfunktioksi. Vaikka yksikkötestausta on suoritettu ohjelmoiduille funktioille, täytyy kuitenkin varmistaa myös pääfunktion toiminta erillisenä testauksena. Kuvassa 15 on toteutettu automaation pääfunktio *consumer*, johon on lisätty poikkeuskäsittely logiikkaa. Poikkeustilanteissa automaatio sammuttaa selaimen ja yrittää koko työnkulkua alusta.

```python
@task
def Consumer():
    for attempt in range(3):
        try:
            start_browser()
            sign_in()
            choose_currency_and_language()
            show_50_rows()
            file_path = html_to_excel()
            send_email_to_recipent(file_path)
            browser.close_all_browsers()
            break
        except Exception as exception:
            browser.close_all_browsers()
            raise Exception(f"Työnkulun suoritus epäonnistui: {exception}")
```

KUVA 15. Automaation päälogiikka.

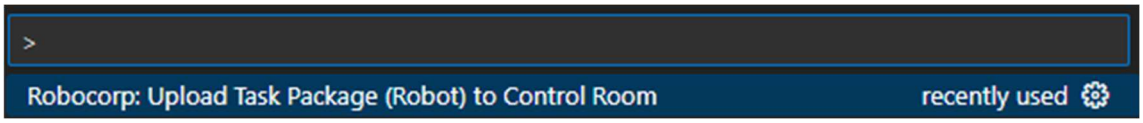
4.3. Control Roomin käyttö

Tässä vaiheessa työtä automaatio toimii suunnitelman mukaan ja sitä pystytään ajamaan lokaalisti. Kehitysvaiheessa on usein riittävää, että automaatiota pystytään ajamaan vain lokaalisti, mutta yleensä tuotannossa olevien automaatioiden täytyy pystyä ajamaan skaalautuvassa ympäristössä. Tämän takia seuraava vaihe on viedä automaatio Control Roomiin ja käynnistää ajo sieltä.

Tämän vaiheen toteutuksessa taskin vieminen vaatii, että lokaali Robocorp-ympäristö on linkitettynä halutun organisaation Control Roomiin. Jotta ympäristö saadaan linkitettyä, avataan VS Coden-komentopaletti, josta valitaan kuvassa 16 näkyvä vaihtoehto. Valinnan jälkeen painetaan ”Enter”-näppäintä, jolloin Control Room-ikkuna selaimeen, josta voidaan luoda avain, jonka avulla ympäristö linkitetään.

KUVA 16. Ympäristön linkkaaminen Control Roomiin.

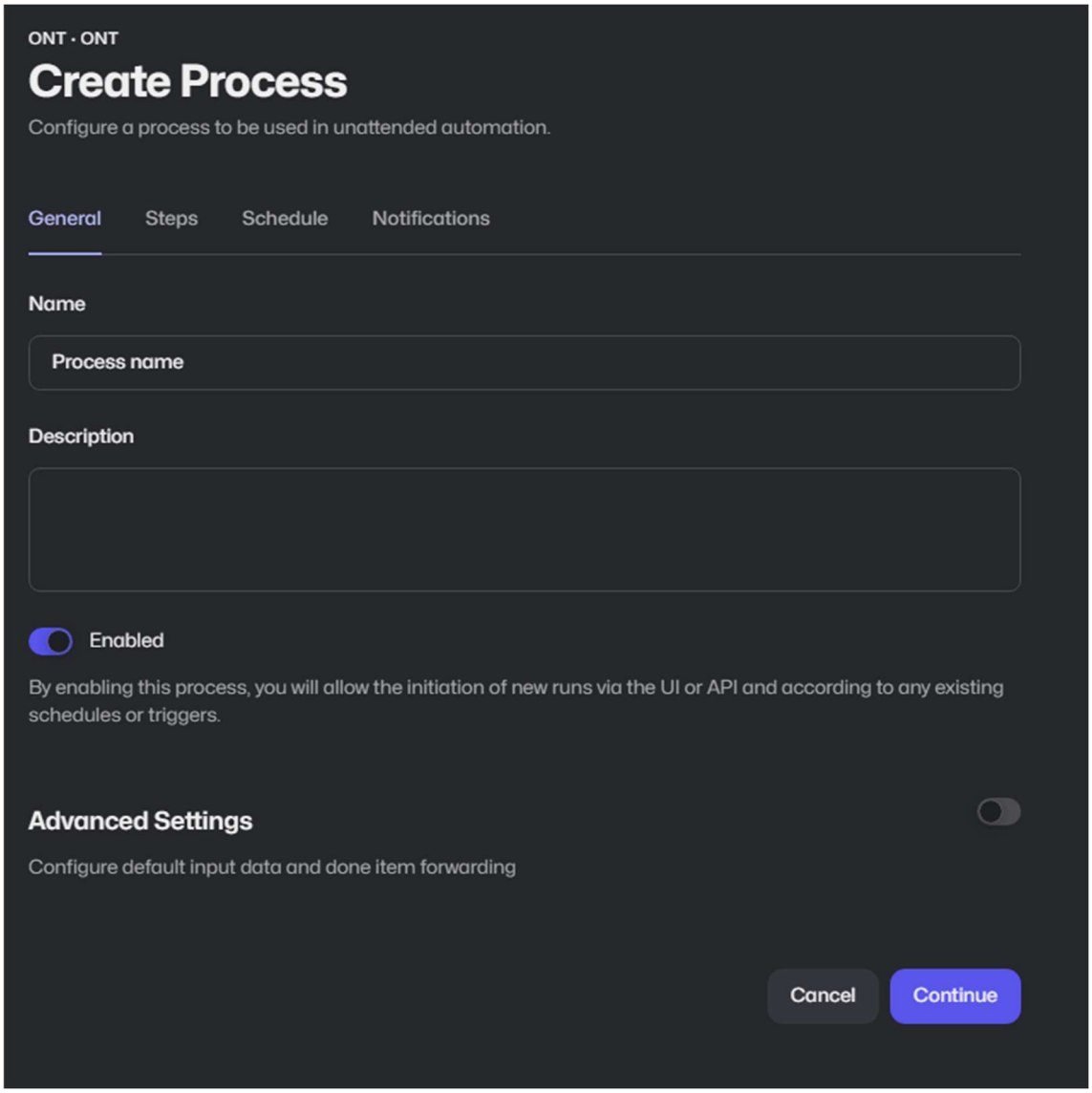
Onnistuneen linkityksen jälkeen avataan uudelleen komentopaletti ja valitaan kuvan 17 vaihtoehto. Valitaan avautuvasta valikosta vaihtoehto ”Create new robot” ja annetaan taskille nimi. Tässä työssä taskin nimeksi annettiin ONT_Robotti.



KUVA 17. Taskin vieminen Control Roomiin

Näiden vaiheiden jälkeen Control Roomiin viedystä taskista voidaan luoda ajettava prosessi. Navigoidaan Control Roomin Unattended Processes- välilehdelle ja valitaan vaihtoehto ”Start from scratch”. Valinnan jälkeen näytölle avautuu kuvan 18 näkymä, jossa asetetaan uudelle prosessille parametreja neljällä eri välilehdellä.

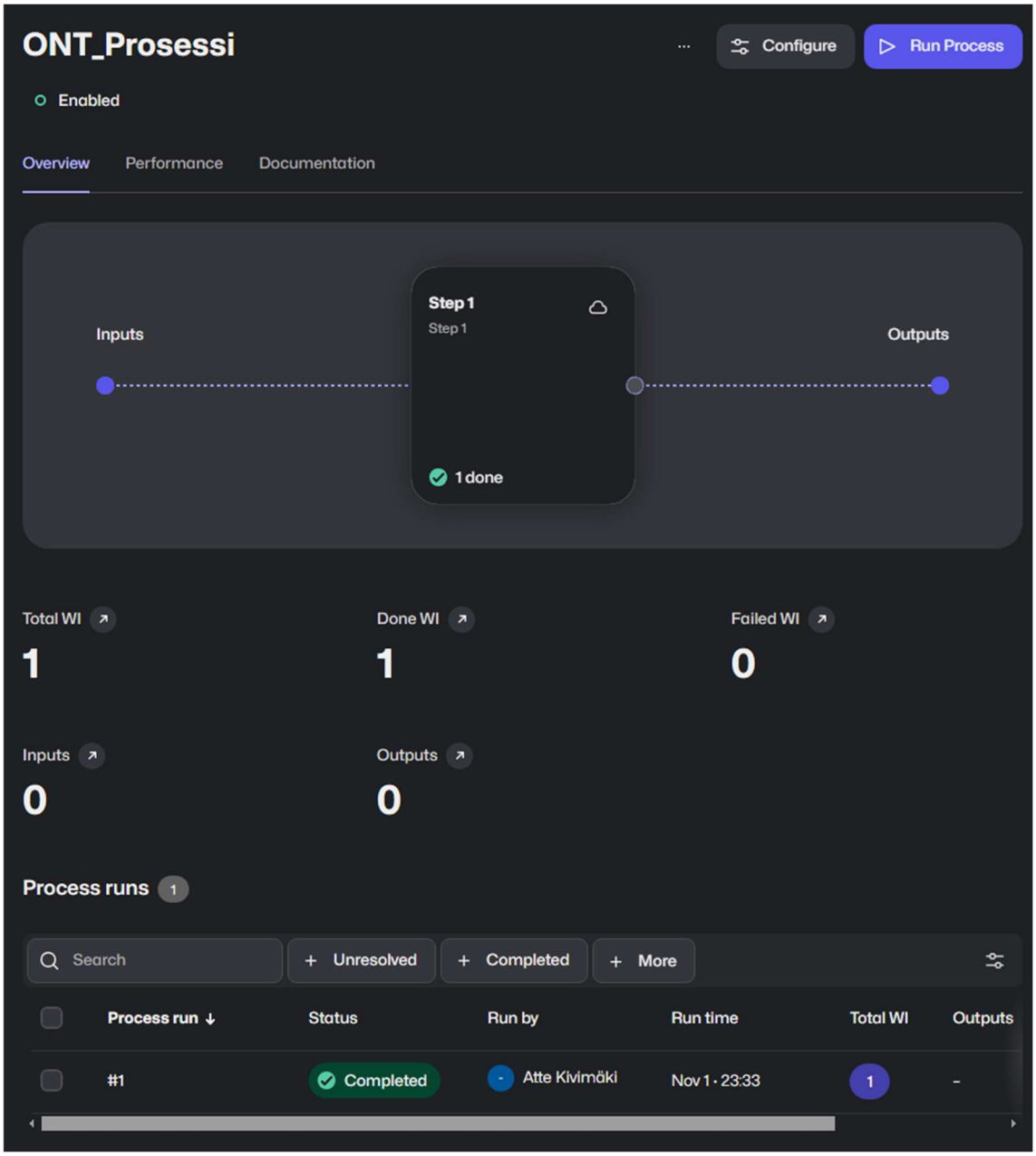
- General-välilehdellä valitaan prosessin nimi, kuvaus ja saako prosessia ajaa UI- tai API-käskyn avulla. On mahdollista asettaa edistyneitä asetuksia, kuten määrittää mukautettuja Work Itemeitä, jotka prosessi ottaa käsittelyyn käynnistyessään.
- Steps-välilehdellä voidaan ketjuttaa useita taskeja ajettavaksi toisensa jälkeen, sekä valita kuinka monella resurssilla prosessi suoritetaan.
- Schedule-välilehdeltä pystytään aikatauluttamaan prosessin aloitus aika- tai sähköpostiherätteen perusteella. On myös mahdollista valita prosessin ajon käynnistyvän vain, jos käsittelemättömiä Work Itemeitä löytyy.
- Notifications-välilehdeltä määritetään halutaanko ilmoitus prosessin ajon onnistumisesta tai kaatumisesta. Ilmoitukset voidaan lähettää sähköpostin ja/tai API:n kautta.



KUVA 18. Prosessin luominen.

Prosessille annettiin nimeksi ONT_Prosessi ja valittiin luotu ONT_Robotti-taski käsiteltäväksi prosessiksi. Aikataulutusta prosessi ei tarvitse, koska se käynnistetään tässä työssä UI:n kautta. Ennen jatkamista sähköpostiosoite lisättiin,

jotta virhe tilanteesta tulee sähköpostiheräte. Seuraavaksi selain siirtyy automaattisesti luodun prosessin hallinta sivulle, josta ajo voidaan käynnistää painamalla ”Run Process”- näppäintä. Kuvassa 19 on esillä onnistuneesta ajosta tuleva näkymä.



KUVA 19. Onnistunut Control Room ajo.

Nyt tehdylle automaatiolle on toteutettu kaikki halutut toiminnallisuudet ja testattu niiden toimivuus pilvessä, että lokaalisti. Voidaan todeta, että ohjelmoitu automaatio täyttää suunnitelmassa asetetut kriteerit.

5 POHDINTA

Opinnäytetyön keskeinen tavoite oli opastaa lukija toteuttamaan toimiva RPA-ratkaisu Robocorp-alustaa hyväksi käyttäen, sekä esitellä kuinka mahdollisesti tarvittavaa poikkeuskäsittelyä lisätään koodiin. Työn tuloksena saatiin helppolukuinen ja selkeästi seurattava ohje, jonka avulla lukija voi toteuttaa itse automaation.

Opinnäytetyössä tuotettiin sovelluksesta löytyvä taulukko onnistuneesti Excel-taulukko muotoon, joka on konkreettinen esimerkki RPA:n hyödyistä datan keräämisessä.

Työssä onnistuttiin mielestäni hyvin ja opinnäytetyötä voi pitää onnistuneena, sillä lukija voi toteuttaa ohjelmistorobotin alusta loppuun saakka luoduilla ohjeilla. Ajallisten haasteiden vuoksi robotille ei toteutettu kaikkiin mahdollisiin poikkeustilanteisiin käsittelyä, vaan yritettiin luoda robotille kyky palautua käynnistäen kohde sovellus uudestaan. Kehityskohteena työssä olisi mielestäni lisätä esimerkkejä mahdollisista poikkeustilanteista ja niiden käsittelystä.

LÄHTEET

A. Harley. 2017 Variations on Practiced Patterns Cause Mistakes. Viitattu 28.10.2024 https://www.nngroup.com/articles/practiced-patterns-mistakes/

A. Hypén. 2020 Ohjelmistoyritys Robocorpissa työkykyongelmiin reagoidaan varhain. Viittattu 28.10.2024 https://www.varma.fi/tama-on-varma/ajankohtaista/uutiset-ja-artikkelit/artikkelit/2020-q4/ohjelmistoyritys-robocorpissa-tyoky-kyongelmiin-reagoidaan-varhain/

Sema4.ai. 2024. Robocorp pricing. Viitattu 28.10.2024 https://sema4.ai/docs/automation/pricing

Sema4.ai. 2024. Embracing Python for Automation-as-code. Viitattu 28.10.2024 https://updates.sema4.ai/release/HHWEX-embracing-python-for-automation-as-code

I. Kit. 2023. The Psychological Impact of RPA on Employee Morale and Job Satisfaction. Viitattu 29.10.2024 https://thescimus.com/blog/impact-of-rpa-on-employee-morale-and-satisfaction/#:~:text=RPA%20Impact%20on%20Employees%201%20Alleviating%20Monotony%2C%20Enhancing,Growth%20...%204%20Cultural%20and%20Psychological%20Shifts%20

M. Heller. 2022. What is Visual Studio Code? Microsoft's extensible code editor. Viitattu 3.11.2024 https://www.infoworld.com/article/2335960/what-is-visual-studio-code-microsofts-extensible-code-editor.html

Geegsforgeeks.org. 2024. Selenium – Components, Features, Uses and Limitations. https://www.geeksforgeeks.org/selenium-basics-components-features-uses-and-limitations/

Sema4.ai. 2022. Control Room. Viitattu 29.10.2024 https://sema4.ai/docs/automation/control-room

Sema4.ai 2021. Technical architecture and security. Viitattu 29.10.2024 https://sema4.ai/docs/automation/control-room/technical-architecture-and-security

rpaframework.org. 2024. RPA.Robocorp.Vault. Viitattu 30.10.2024 https://rpaframework.org/libdoc/RPA_Robocorp_Vault.html

J. Anderson. 2024. Python vs C++: Selecting the Right Tool for the Job. Viitattu 1.11.2024 https://realpython.com/python-vs-cpp/#summary-python-vs-c

LIITTEET

Liite 1. Onnistuneen ajon Excel-taulukko

Nimi	Symboli	Hinta	Markkina-arvo
Bitcoin	BTC	64,684.84	1,279,205,815,730
Ethereum	ETH	2,312.47	278,438,775,434
Tether	USDT	0.9174	110,619,013,663
BNB	BNB	528.95	77,190,254,965
Solana	SOL	155.53	73,160,798,553
USDC	USDC	0.9186	31,944,951,719
XRP	XRP	0.4688	26,657,884,563
Dogecoin	DOGE	0.1485	21,771,876,012
TRON	TRX	0.1545	13,357,696,848
Toncoin	TON	4.44	11,291,294,537
Cardano	ADA	0.3147	11,011,784,364
Shiba Inu	SHIB	0.00001644	9,689,075,225
Avalanche	AVAX	22.97	9,349,738,391
Chainlink	LINK	10.49	6,576,665,261
Bitcoin Cash	BCH	328.60	6,500,431,988
Polkadot	DOT	3.63	5,487,505,658
UNUS SED LEO	LEO	5.56	5,143,516,867
Sui	SUI	1.81	5,004,473,292
Dai	DAI	0.9186	4,928,461,548
Litecoin	LTC	63.61	4,779,992,893
NEAR Protocol	NEAR	3.73	4,543,303,821
Aptos	APT	8.39	4,346,618,256
Uniswap	UNI	7.03	4,218,297,569
Pepe	PEPE	0.058392	3,530,441,970
Internet Computer	ICP	7.19	3,404,649,166
Bittensor	TAO	444.43	3,280,343,323
Artificial Superintelligence Alliance	FET	1.18	2,875,347,599
Monero	XMR	143.50	2,647,127,490
Kaspa	KAS	0.1055	2,644,019,197
Ethereum Classic	ETC	17.08	2,551,062,288
Stellar	XLM	0.08493	2,526,927,764
First Digital USD	FDUSD	0.9164	2,326,509,353
Stacks	STX	1.51	2,263,603,231
Render	RENDER	4.36	2,259,632,191
POL (ex-MATIC)	POL	0.2943	2,256,974,115
dogwifhat	WIF	2.19	2,190,072,264
OKB	OKB	35.12	2,107,143,252
Aave	AAVE	131.54	1,969,075,418
Filecoin	FIL	3.24	1,932,238,527
Arbitrum	ARB	0.4846	1,926,656,699

Optimism	OP	1.48	1,855,955,809
Mantle	MNT	0.5488	1,847,776,149
Immutable	IMX	1.10	1,809,013,511
Celestia	TIA	4.47	1,787,608,748
Cronos	CRO	0.06709	1,782,803,260
THORChain	RUNE	5.17	1,745,729,396
Injective	INJ	17.55	1,734,909,842
Fantom	FTM	0.6022	1,688,387,621
Hedera	HBAR	0.04255	1,603,392,326
VeChain	VET	0.01943	1,573,620,076