Karelia-ammattikorkeakoulu
Tradenomi (AMK) Tietojenkäsittely

Laskutuksen automatisointi leipuri-kondiittorin ja catering-palveluiden tilausten hallinnassa

OCR- ja RPA-teknologioiden soveltaminen käytännössä

Nikolett Berkiné Varga

Opinnäytetyö, helmikuu 2026

www.karelia.fi

Karelia AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Helmikuu 2026
Tradenomi (AMK) tietojenkäsittely koulutus

Tikkarinne 9
80200 JOENSUU
+358 13 260 600

Tekijä(t)
Nikolett Berkiné Varga

Nimeke
Laskutuksen automatisointi leipuri-kondiittorin ja catering-palveluiden tilausten hallinnassa

Tiivistelmä

Opinnäytetyössä tehtävänä oli kehittää automatisoitu laskutusjärjestelmä pienille ja keskisuurille elintarvikealan yrityksille, kuten leipomoille, konditorioille ja catering-palveluille. Työn tavoitteena oli tehostaa tilausten hallintaa ja laskutusta vähentämällä manuaalista työtä, virheiden riskiä ja laskutuksen viivettä. Lisäksi tavoitteena oli luoda helposti ylläpidettävä ja laajennettava ratkaisu, joka tukee elintarvikealan digitalisaatiota ja asiakaspalvelun kehittämistä.

Työ toteutettiin toiminnallisena kehittämisprojektina, jossa rakennettiin ohjelmistorobotiikkaan (UiPath) ja optiseen merkintunnistukseen (OCR) perustuva laskutusautomaatio. Järjestelmä lukee kalenterimerkinnät, muuntaa ne laskuiksi ja lähettää laskut asiakkaille PDF-muodossa sähköpostitse. Ennen lähettämistä käyttäjä vahvistaa laskun sisällön, ja laskut arkistoidaan automaattisesti pilvipalveluun. Ratkaisu testattiin käytännön toimintaympäristössä hyödyntäen todellisia tilausdokumentteja.

Työn tuloksena syntyi toimiva ja testattu automaatioratkaisu, joka voidaan integroida olemassa oleviin työkaluihin, kuten Google Calendar, ja mukauttaa erilaisiin tilaustiedonhallinnan tarpeisiin. Tulokset osoittivat, että ratkaisu vähentää manuaalista työtä ja parantaa tiedon luotettavuutta.

Kieli
suomi

Sivuja 59
Liitteet 0
Liitesivumäärä 0

Asiasanat
ohjelmistorobotiikka (RPA), prosessien automatisointi, UiPath, OCR-teknologia, Orchestrator, automaattinen laskutus, elintarvikeala

THESIS
February 2026
Degree Programme in Business Information Technology

Tikkarinne 9
80200 JOENSUU
FINLAND
+ 358 13 260 600

Author (s)
Nikolett Berkiné Varga

Title
Automation of Billing in Bakery, Pastry, and Catering Order Management

Abstract

The aim of this thesis was to develop an automated invoicing system for small and medium-sized food industry businesses, such as bakeries, patisseries, and catering services. The goal was to streamline order management and invoicing by reducing manual work, minimizing errors, and improving the efficiency of billing and documentation processes.

The work was implemented as a practical development project using Robotic Process Automation (UiPath) and Optical Character Recognition (OCR) technologies. The system reads calendar entries, converts them into invoices, and sends the invoices to customers as PDF files via email. Before sending, the user confirms the accuracy of the information, and all invoices are automatically archived in a cloud service. The solution was tested in a real operational environment using authentic order data.

As a result, a functional and tested automation solution was created. It can be integrated with existing tools such as Google Calendar and adapted to various order management processes. The results show that the automation significantly reduces manual work, improves data accuracy, and supports the digital transformation of the food service industry by enhancing efficiency and reliability.

Language
Finnish

Pages 59
Appendices 0
Pages of Appendices 0

Keywords
Robotic Process Automation (RPA), Process automation, UiPath, Optical Character Recognition (OCR), Document digitalization, Orchestrator, Automated invoicing, Food industry

Käsitteet

Attended automation	Automaatiotyyppi, jossa käyttäjä käynnistää robotin manuaalisesti ja seuraa suoritusta (UiPath Documentation 2025a).
DataTable	UiPathin tietorakenne, jota käytetään datan tallentamiseen ja käsittelyyn taulukkomuodossa.
Digitize Document	Aktiviteetti, joka muuntaa kuvan tai PDF-tiedoston analysoitavaksi digitaaliseksi dokumentiksi (UiPath Documentation 2025b).
Excel Process Scope	Aktiviteettialue, jonka sisällä voidaan suorittaa Exceliin liittyviä automaatiotoimintoja (UiPath Documentation 2025c).
Google Workspace HTTP Request	Aktiviteetti, jolla voidaan lähettää HTTP-pyyntöjä Google-palveluihin, kuten Gmail, Calendar tai Drive (UiPath Documentation 2025d).
Google Workspace Scope	UiPathin komponentti, joka mahdollistaa yhteyden Google-palveluihin, kuten Calendar ja Drive (UiPath Documentation 2025e).
Invoke Code	Aktiviteetti, jonka avulla voidaan suorittaa C#-koodia osana automaatiota (UiPath Documentation 2025f).
Invoke Workflow File	Aktiviteetti, jolla voidaan kutsua ja suorittaa toinen .xaml-työnkulku osana automaatiota (UiPath Documentation 2025g).
Machine Key	Yksilöllinen avain, jolla UiPath-robotti yhdistetään Orchestratoriin turvallisesti.
Optical Character Recognition	Teknologia, joka tunnistaa ja muuntaa kuvan sisältämä tekstin digitaaliseksi tiedoksi (Google Cloud 2024; Microsoft Azure AI Vision 2024).
Robotic Process Automation	Ohjelmistorobotiikka; teknologia, jolla automatisoidaan toistuvia, sääntöpohjaisia työtehtäviä. (UiPath Documentation 2025h).
Trigger	Aikataulu- tai ehtopohjainen toiminto, joka käynnistää automaation Orchestratorissa (UiPath Documentation 2025i).

UiPath Assistant	Käyttöliittymä, jonka avulla käyttäjä voi suorittaa ja seurata automaatioita paikallisesti (UiPath Documentation 2025j).
UiPath Orchestrator	Pilvipohjainen hallintaympäristö, jonka kautta automatisointeja hallitaan, ajastetaan ja valvotaan keskitetysti (UiPath Documentation 2025k).
UiPath Studio	Ohjelmistorobotiikan kehitysalusta, jolla työnkulkuja voidaan rakentaa visuaalisesti ilman ohjelmointia (UiPath Documentation 2025l).
Unattended automation	Täysin itsenäinen automaatio, joka suoritetaan ilman käyttäjän osallistumista (UiPath Documentation 2025a).
Use Gmail	Aktiviteetti, jolla voidaan lähettää ja vastaanottaa sähköposteja Gmail-palvelun kautta (UiPath Documentation 2025m).
Workflow	Työnkulku, joka sisältää sarjan automatisoituja tehtäviä.

Sisältö

1 Johdanto

Pienet ja keskisuuret elintarvikealan yritykset, kuten leipomot, konditoriat ja catering-palvelut, hallinnoivat usein asiakastilauksia manuaalisesti esimerkiksi puhelimitse tai paperikalentereiden avulla. Tämä tekee tilausten hallinnasta ja laskutuksesta työlästä ja altistaa prosessin inhimillisille virheille. Samalla alan toimijat kohtaavat kasvavia vaatimuksia tehokkuudesta, nopeudesta ja asiakaspalvelun laadusta. Manuaalisten prosessien yleisyys on tyypillistä monille suomalaisille pk-yrityksille, vaikka digitaaliset ratkaisut ovat yleistymässä (VTT 2020).

Tässä opinnäytetyössä kehitetään automatisoitu laskutusratkaisu, joka tehostaa tilausten hallintaa hyödyntäen ohjelmistorobotiikkaa (Robotic Process Automation, RPA) ja optista merkintunnistusta (Optical Character Recognition, OCR). Ratkaisu mahdollistaa sen, että kalenteriin kirjatut tilaukset muunnetaan automaattisesti laskuiksi, jotka tarkistetaan ja lähetetään asiakkaille sähköpostitse PDF-muodossa. Lopuksi laskut arkistoidaan automaattisesti pilvipalveluun.

Opinnäytetyön tavoitteena on kehittää toimiva automaatioratkaisu, joka vähentää manuaalista työtä, pienentää virheiden riskiä ja tukee elintarvikealan pk-yritysten digitalisaatiota. Työ perustuu käytännön toteutukseen ja testaamiseen aidossa toimintaympäristössä, ja siinä hyödynnetään UiPathin ohjelmistorobotiikka-alustaa ja sen Document Understanding -ominaisuuksia.

Tekijällä on yli seitsemän vuoden työkokemus leipomo- ja ravintola-alalta, mikä tuo syvällistä ymmärrystä prosessien erityispiirteistä ja tarpeista. Tämä käytännön kokemus yhdistettynä tietojenkäsittelyn ja digitaalisen liiketoiminnan opintoihin luo vahvan pohjan kehitystyölle.

2 Opinnäytetyön tietoperusta

2.1 Ohjelmistorobotiikka liiketoimintaprosessien tukena

Ohjelmistorobotiikka (Robotic Process Automation, RPA) tarkoittaa teknologiaa, jonka avulla tietokoneohjelma eli “robotti” voi suorittaa ihmisen tavoin toistuvia ja sääntöpohjaisia tehtäviä digitaalisissa järjestelmissä. Robotti jäljittelee käyttäjän toimia käyttöliittymässä, kuten tiedon kopioimista, syöttämistä ja siirtämistä eri sovellusten välillä. RPA ei siis ole fyysinen robotti, vaan ohjelmisto, joka toimii olemassa olevien järjestelmien päällä ilman tarvetta muuttaa niiden rakennetta (Lacity & Willcocks 2016).

Ohjelmistorobotiikka eroaa perinteisestä automaatiosta siinä, että se ei vaadi syvää järjestelmäintegraatiota tai ohjelmointia. Sen sijaan työnkulut määritellään visuaalisesti, jolloin myös liiketoiminnan asiantuntijat voivat osallistua prosessien kehittämiseen. Tyypillisiä RPA:n käyttökohteita ovat esimerkiksi tilausten käsittely, laskutus, asiakastietojen päivitys ja raportointi.

Liiketoimintaprosessien näkökulmasta ohjelmistorobotiikka tuo merkittäviä hyötyjä erityisesti silloin, kun käsitellään suuria määriä dataa tai toistuvia tehtäviä. Hyödyt liittyvät muun muassa prosessien nopeutumiseen, virheiden vähenemiseen ja resurssien vapautumiseen vaativampiin tehtäviin (Tripathi 2018). Pk-yrityksissä ohjelmistorobotiikka ja muut digitaaliset ratkaisut voivat tukea toiminnan tehostamista ilman raskaita järjestelmäuudistuksia, mikä on erityisen tärkeää rajallisten resurssien toimintaympäristössä (VTT 2020).

Opinnäytetyössä käytetty RPA-ratkaisu toteutettiin UiPath-alustan avulla. UiPath on yksi markkinoiden johtavista ohjelmistorobotiikka-alustoista, ja se mahdollistaa sekä yksinkertaisten että monimutkaisempien työnkulkujen rakentamisen (UiPath Documentation 2025h). UiPath Studio-ympäristössä hyödynnetään visuaalista työnkulkusuunnittelua, mikä tekee automaatioiden kehittämisestä saavutettavaa myös ilman ohjelmointitaustaa (UiPath Documentation 2025l). Näin RPA-prosessit voidaan suunnitella, testata ja ottaa

käyttöön tehokkaasti myös pienissä organisaatioissa ilman laajoja teknisiä resursseja.

Ohjelmistorobotiikan suoritusmalleja on pääasiassa kahta tyyppiä: attended ja unattended. Attended-robotti toimii käyttäjän ohjauksessa ja suorittaa automaatioita tämän työasemalla, kun taas unattended-robotti toimii taustalla itsenäisesti ilman käyttäjän väliintuloa. Unattended-robotti voidaan ajastaa tai käynnistää automaattisesti esimerkiksi palvelimella, ja se suorittaa prosessit täysin itsenäisesti. Tämä ero on keskeinen erityisesti silloin, kun automatisointi pyritään viemään tuotantokäyttöön, kuten tässä työssä Orchestratorin avulla on toteutettu (UiPath Documentation 2025k).

Tässä työssä ohjelmistorobotiikkaa käytetään kalenterimerkintöjen lukemiseen, tietojen käsittelyyn ja laskujen luomiseen automatisoidusti. Lisäksi robotti tarkistaa laskun tiedot ja pyytää käyttäjältä vahvistuksen ennen laskujen lähettämistä. Näin automatisointi tukee sekä tehokkuutta että tietojen oikeellisuutta.

2.2 OCR-teknologia ja dokumenttien digitalisointi

Optinen merkintunnistus (Optical Character Recognition, OCR) on teknologia, joka muuntaa skannatun tai kuvamuotoisen tekstin koneellisesti luettavaan muotoon. OCR-järjestelmä tunnistaa kirjainten ja numeroiden muotoja kuvasta ja muuttaa ne digitaaliseen tekstiksi, jota voidaan käsitellä tietokoneohjelmilla. Teknologian avulla voidaan automatisoida esimerkiksi lomakkeiden, kuittien ja laskujen tietojen tulkinta (Gill et al. 2025).

OCR-prosessin perusperiaatteet koostuvat kuvankäsittelystä, merkkien tunnistuksesta ja tuloksen jälkikäsittelystä. Ensin kuvan laatua parannetaan (esim. kontrastin säätö, kohinan poisto), minkä jälkeen algoritmit tunnistavat yksittäiset kirjaimet tai numerot. Lopuksi teksti siistitään ja muunnetaan luettavaan muotoon. Modernit OCR-ratkaisut hyödyntävät tekoälyä ja koneoppimista, mikä mahdollistaa myös käsinkirjoitetun tekstin tulkinnan vaihtelevista lähteistä (Google Cloud 2024; Microsoft Azure AI Vision 2024).

UiPathin Document Understanding -alusta yhdistää OCR-teknologian ja tekoälypohjaisen tiedon tulkinnan. Se hyödyntää useita eri OCR-moottoreita, kuten Tesseract OCR, Google Cloud OCR ja Microsoft Read API, jotka voidaan valita tilanteen mukaan. Alustan avulla voidaan tunnistaa tekstikenttiä, taulukoita ja metatietoja sekä opettaa robotti tunnistamaan toistuvia dokumenttirakenteita (UiPath Documentation 2025n).

Tässä opinnäytetyössä OCR:ää käytetään osana tilausten digitalisointia: painetut tai skannatut tilauslomakkeet muunnetaan automaattisesti tekstimuotoon, jonka perusteella robotin on mahdollista muodostaa kalenterimerkintä ja lopulta lasku. Näin paperipohjainen tieto voidaan yhdistää saumattomasti digitaaliseen työnkulkuun.

OCR-teknologian hyödyntäminen elintarvikealan yrityksissä mahdollistaa merkittävän ajansäästön, virheiden vähenemisen ja tiedon tallentamisen yhteen järjestelmään. Erityisesti pk-yrityksille tämä tarjoaa keinon yhdistää perinteiset toimintatavat ja digitaaliset prosessit ilman suuria järjestelmäinvestointeja.

2.3 Laskutusprosessit ja niiden haasteet pk-yrityksissä

Pk-yrityksille laskutusprosessi on keskeinen osa liiketoiminnan kassavirtaa, mutta se on usein manuaalinen ja altis virheille. Tyypillisiä haasteita ovat tietojen manuaalinen siirto eri järjestelmien välillä, epäyhtenäiset laskumallit, inhimilliset virheet sekä viivästykset laskujen lähettämisessä (Lahti & Salminen 2014, luku 3).

Automatisoitu laskutus voi merkittävästi vähentää näitä ongelmia, esimerkiksi kalenterimerkintöjen pohjalta tapahtuva automaattinen laskutus poistaa tarpeen syöttää tietoja manuaalisesti. Lisäksi automatisoidut tarkistusvaiheet varmistavat laskun tietojen oikeellisuuden ennen lähettämistä, mikä parantaa asiakastyytyväisyyttä ja vähentää virheiden korjaamiseen kuluva aika.

Pk-yrityksissä, joilla ei ole käytössä suuria ERP-järjestelmiä, automatisoidut mutta joustavat ratkaisut voivat tarjoa merkittäviä etuja ilman suuria investointeja tai IT-infrastruktuuria.

2.4 Elintarvikealan erityispiirteet ja digitalisaatio

Elintarvikealan pk-yritykset, kuten leipomot, konditoriat ja catering-palvelut, toimivat usein nopeatempoisessa ja asiakaskohtaisesti räätälöidyssä ympäristössä. Tilaukset saattavat tulla eri kanavien kautta – puhelimitse, sähköpostitse tai kasvotusten – ja ne kirjataan usein manuaalisesti paperikalenteriin tai yksinkertaisiin taulukkopohjiin.

Vaikka digitalisaatio on viime vuosina edennyt myös elintarvikealalla, sen hyödyntäminen vaihtelee merkittävästi yritysten koon ja resurssien mukaan. Tutkimusten mukaan monet pienet yritykset hyödyntävät edelleen manuaalisia prosesseja tilaus- ja laskutushallinnassa, mikä lisää virheiden ja viivästysten riskiä (VTT 2020).

Ohjelmistorobotiikka, pilvipalvelut ja mobiilisovellukset tarjoavat kuitenkin mahdollisuuksia tehostaa alan toimintaa ja parantaa asiakaspalvelua. Automatisoidut ratkaisut voivat esimerkiksi vähentää tilausten käsittelyaikaa ja vapauttaa henkilöstöresursseja muihin tehtäviin. Näin yritys voi parantaa palvelunsa laatua ja vastata asiakkaiden kasvaviin odotuksiin nopeudesta ja joustavuudesta.

Tässä työssä kehitetty ratkaisu vastaa elintarvikealan käytännön haasteisiin: se yhdistää manuaalisen tilaustiedon automaattiseen käsittelyyn, mikä mahdollistaa alan yrityksille asteittaisen siirtymisen kohti digitaalisempia toimintatapoja ilman, että koko liiketoimintaprosessia tarvitsee muuttaa kerralla.

3 Kehittämistyön menetelmät ja toteutus

3.1 Kehittämistyössä käytetyt menetelmät

Tässä työssä kehitetty ratkaisu vastaa elintarvikealan käytännön haasteisiin: se yhdistää manuaalisen tilaustiedon automaattiseen käsittelyyn, mikä mahdollistaa alan yrityksille asteittaisen siirtymisen kohti digitaalisempia toimintatapoja ilman, että koko liiketoimintaprosessia tarvitsee muuttaa kerralla.

Tämä opinnäytetyö on toiminnallinen kehittämistyö, jossa tavoitteena on ratkaista käytännönläheinen ongelma suunnittelemalla ja toteuttamalla automatisoitu laskutusratkaisu elintarvikealan pk-yrityksille. Toiminnallisen työn tuotoksena syntyy konkreettinen ja testattu järjestelmä, jota voidaan hyödyntää suoraan yritystoiminnassa.

Työssä käytettiin kehittämistutkimuksellinen lähestymistapaa, jossa painopiste on olemassa olevan ongelman ratkaisemisessa kehittämällä ja testaamalla uutta toimintamallia tai järjestelmää. Kehitystyö eteni iteratiivisesti, eli ratkaisu suunniteltiin, rakennettiin ja testattiin vaiheittain, saadun palautteen ja havaintojen perusteella. Tämä mahdollisti joustavan etenemisen ja käytännön toimivuuden varmistamisen.

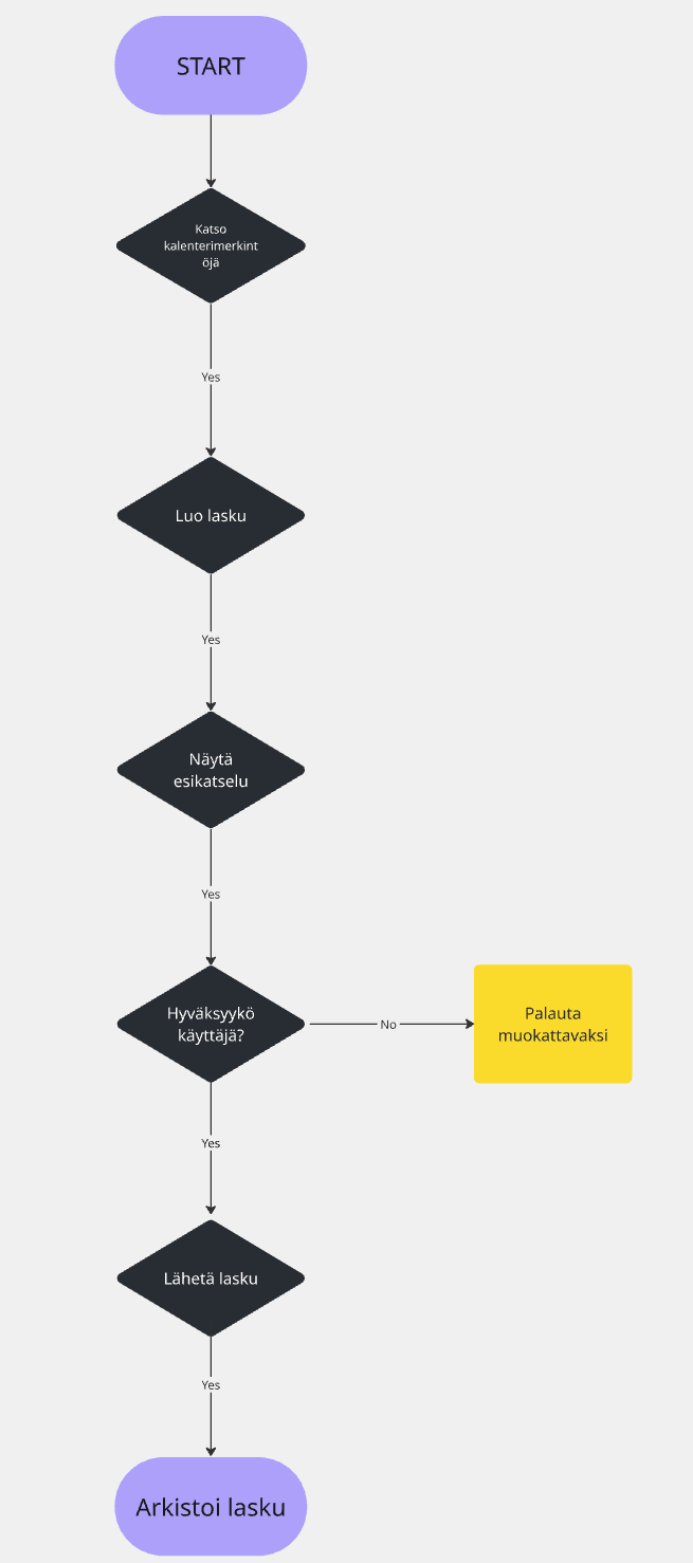
Teknisessa toteutuksessa käytettiin UiPath-ohjelmistorobotiikka-alusta, joka mahdollistaa liiketoimintaprosessien automatisoinnin. Kalenterimerkintöjen käsittelyyn käytettiin OCR-teknologiaa, erityisesti UiPathin Document Understanding -ominaisuutta, jolla paperisten tai kuvamuotoisten tietojen digitointi on mahdollista. Laskut luotiin dynaamisella mallilla ja lähetettiin sähköpostitse PDF-muodossa, ja ne arkistoitiin pilvipohjaisiin järjestelmiin. Lisäksi järjestelmä sisälsi tarkistusvaiheen, jossa käyttäjä vahvisti laskun oikeellisuuden ennen lähettämistä.

3.2 Suunnittelu ja sen muutokset

Suunnitteluvaiheessa määriteltiin opinnäytetyön päätavoite: automatisoida laskutusprosessi hyödyntämällä UiPath-ohjelmistorobotiikkaa ja OCR-teknologiaa. Työvaiheet hahmoteltiin ensin Miro-alustalla, jossa visualisoitiin koko prosessi tilauksen kirjauksesta laskun arkistointiin. Työn eri vaiheet jäsennettiin omiksi workflowiksi, jotka muodostivat yhdessä yhtenäisen kokonaisuuden. Suunnitteluvaiheessa määritellyt työnkulut ja niiden väliset suhteet on esitetty kuvassa 1.

Kuva 1. Workflowt (ReadCalendar → ArchiveInvoice).

Prosessi kattoi kalenteritietojen haun, laskun luonnin, tietojen validoinnin, laskun lähetyksen ja lopuksi arkistoinnin. Pääprosessin eteneminen kuvattiin erillisellä vuokaaviolla, jonka avulla varmistettiin prosessin looginen eteneminen tilauksen käsittelystä laskun arkistointiin. Automatisoidun laskuprosessin pääkulku on kuvattu kuvassa 2.



Kuva 2. Prosessin pääkulku (flowchart).

Lisäksi suunnittelussa määritettiin tarvittavat UiPath-paketit ja Google Workspace -integraatiot, jotta eri osaprosessit voitiin yhdistää saumattomasti. Tavoitteena oli varmistaa, että prosessin jokainen vaihe on eriytetty omaksi workflowkseen. Tämä paransi ratkaisun ylläpidettävyyttä ja mahdollisti vaiheittaisen testauksen. Tavoitteena oli rakentaa selkeä ja helposti hallittava prosessi, jota voidaan tarvittaessa laajentaa tulevaisuudessa esimerkiksi uusilla automaatiotoiminnoilla. Kuva 3 kokoaa käytetyt UiPath-paketit ja niiden rooli automaatiossa.

Tarvittavat UiPath pakettit
UiPath.WebAPI.Activities (ok)
UiPath.Web.Activities
UiPath.System.Activities (ok)
UiPath.Mail.Activities (ok)
UiPath.Excel.Activities (ok)
UiPath.IntelligentOCR.Activities (ok)
UiPath.DocumentUnderstanding.ML.Activities (ok)
UiPath.OCR.Activities (ok)
UiPath.PDF.Activities (ok)
UiPath.Word.Activities
UiPath.Form.Activities (ok)
UiPath.GSuite.Activities
UiPath.GoogleWorkspace.Activities

Kuva 3. Tarvittavat UiPath-paketit.

Alkuperäinen prosessisuunnitelma (kuva 4) sisälsi erillisen *ExtractOrderData*-vaiheen sekä OCR:n liittämisen suoraan *ReadCalendar*-workflow'hun. Ajatuksena oli, että järjestelmä poimisi sekä sähköiset tilaukset Google Calendarista että paperilla olevat tilaukset OCR:n avulla, ja veisi nämä tiedot rinnakkain laskupohjaan.

Kuva 4. Alkuperäinen Workflow suunnitelma.

Työn edetessä suunnitelmaa muutettiin kahdesta syystä. Ensinnäkin havaittiin, että *ReadCalendar*-workflow oli helpompi ja luotettavampi toteuttaa siten, että se hakee kalenteritiedot ja pilkkoo niistä suoraan tilausdatan. Tällä tavalla erillinen *ExtractOrderData*-vaihe voitiin poistaa kokonaan, mikä yksinkertaisti prosessia ja teki siitä selkeämmän ylläpitää.

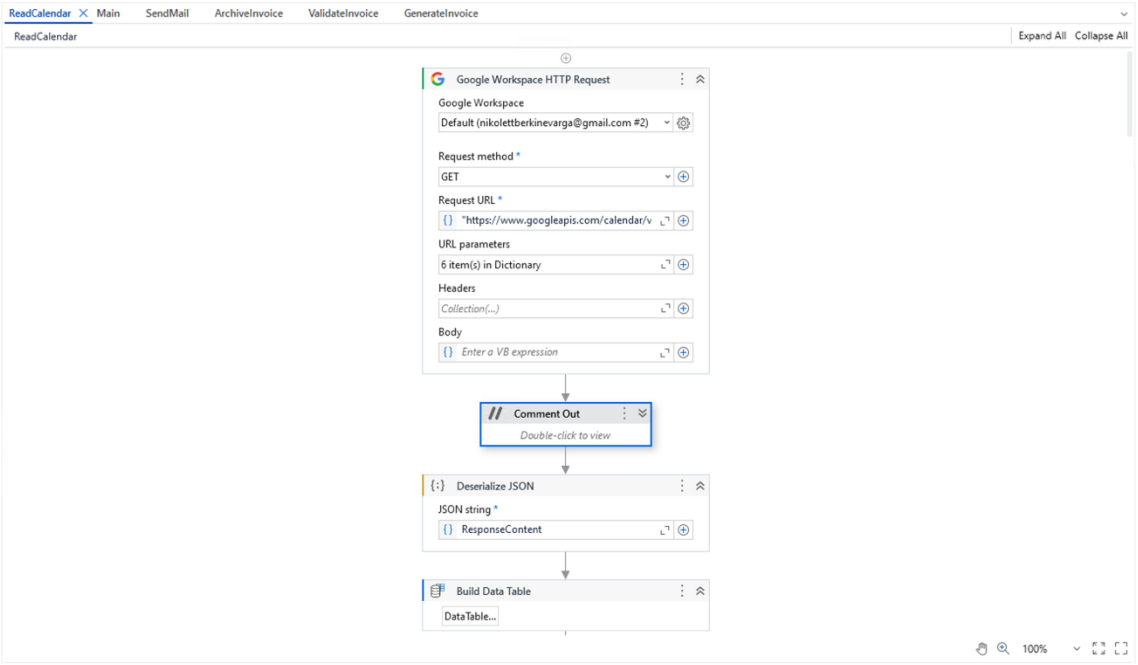
Toiseksi OCR:n toteutusta tarkennettiin käytännön testien perusteella. Alkuperäisessä mallissa OCR:n oli tarkoitus lukea tilaukset paperilta ja siirtää tiedot suoraan laskupohjaan yhdessä kalenteridatan kanssa. Testausten aikana kuitenkin huomattiin, että tehokkaampaa oli tehdä OCR:stä oma erillinen workflow, joka vie paperilta luetut tilaukset ensin Google Kalenteriin. Näin kaikki tilaukset — riippumatta siitä, onko ne kirjattu käsin paperille vai digitaalisesti — tallentuvat samaan lähteeseen, josta automaatio voi käsitellä ne yhtenäisellä logiikalla.

Tämä ratkaisu varmisti, että tieto pysyy ajantasaisena ja että *ReadCalendar*-vaihe säilyy kevyenä ja läpinäkyvänä. Lopullinen prosessisuunnitelma (kuva 1) on näin ollen yksinkertaisempi, paremmin ylläpidettävä ja vastaa käytännön tarpeisiin huomattavasti alkuperäistä rakennetta paremmin.

3.3 Workflow: ReadCalendar

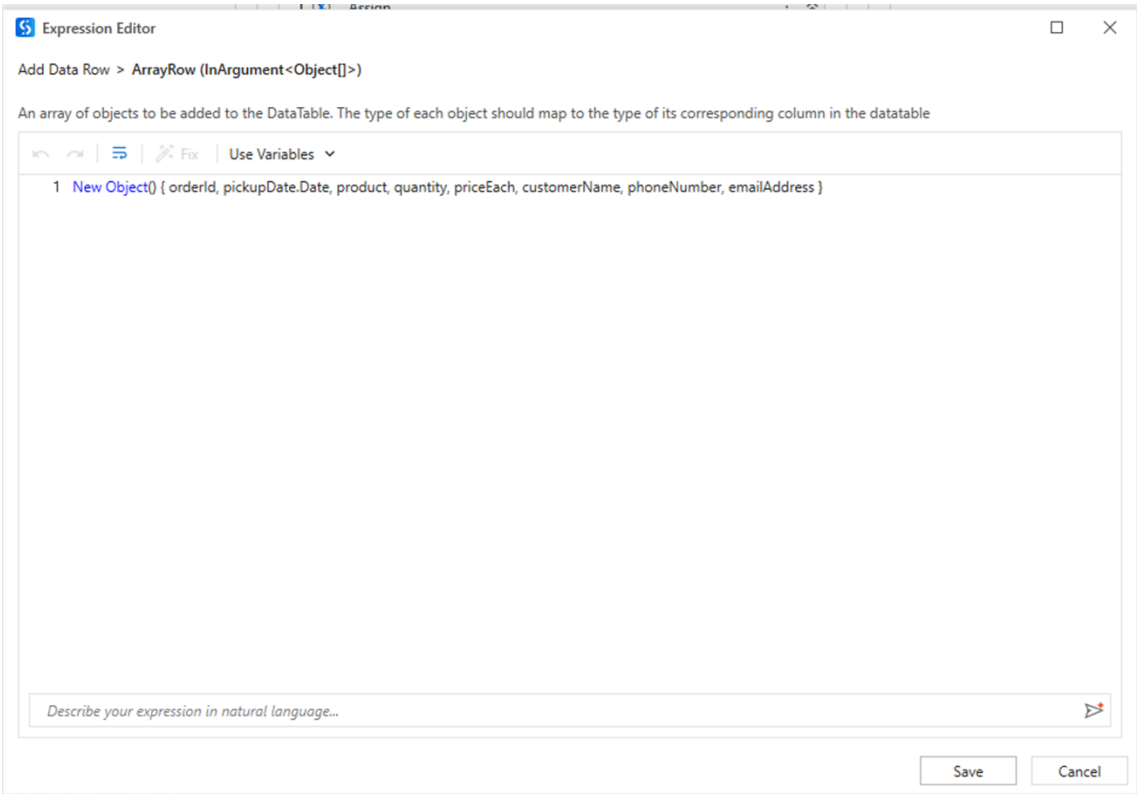
ReadCalendar-workflow’n tarkoituksena oli hakea asiakastilaukset kalenteristä ja muuntaa ne laskutusta varten strukturoituun muotoon. Ratkaisun avulla manuaaliset kalenterimerkinnät voitiin muuttaa automaattisesti käsiteltäväksi dataksi, jota käytettiin laskujen luonnin pohjana. Workflow hakee kalenteritapahtumat Google Calendarista Google Workspace HTTP Request -

rajapinnan avulla, muuntaa vastauksen JSON-muotoon ja purkaa siitä olennaiset tiedot, kuten asiakkaan nimen, yhteystiedot, tilauksen sisällön ja hinnan (kuva 5).



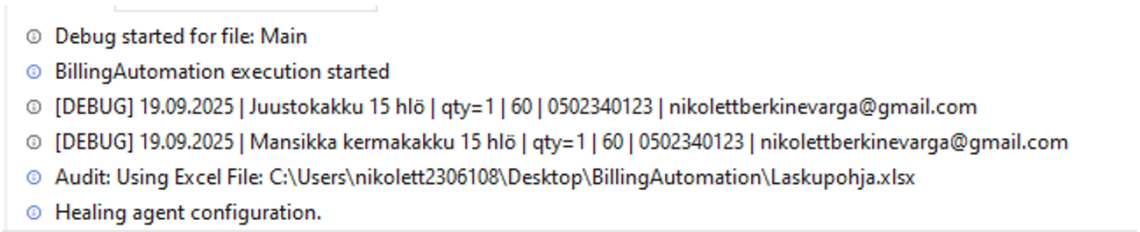
Kuva 5. ReadCalendar workflow Google Workspace HTTP Request.

Alkuvaiheessa prosessi kohtasi useita kehityshaasteita. Esimerkiksi DataTable oli suunniteltu siten, että yksi tilaus sisälsi vain yhden tuotteen, mikä aiheutti ongelmia useamman rivin tilauksissa. Tämä ratkaistiin muuttamalla DataTablen rakennetta ja lisäämällä logiikka, joka pilkkoo tilauksen rivit tuotteittain (kuva 6). Lisäksi oli määritettävä, mitkä kalenterimerkinnät käsitellään tilauksina ja mitkä ohitetaan. Näiden sääntöjen tarkentaminen tehtiin käytännön testauksen avulla.



Kuva 6. Add Data Row -ilmaisu: DataTablen sarakkeisiin kirjoitettavat kentät järjestyksessä (orderId, pickupDate.Date, product, quantity, priceEach, customerName, phoneNumber, emailAddress).

Lopullisessa muodossaan ReadCalendar-workflow mahdollistaa sen, että kaikki asiakastilaukset kerätään automaattisesti kalenterista ja muunnetaan yhtenäiseen DataTable-rakenteeseen (kuva 7). Tämä vähensi manuaalista työtä, paransi tiedon luotettavuutta ja loi pohjan seuraaville prosessivaiheille: laskun luomiselle, validoinnille ja lähettämiselle.

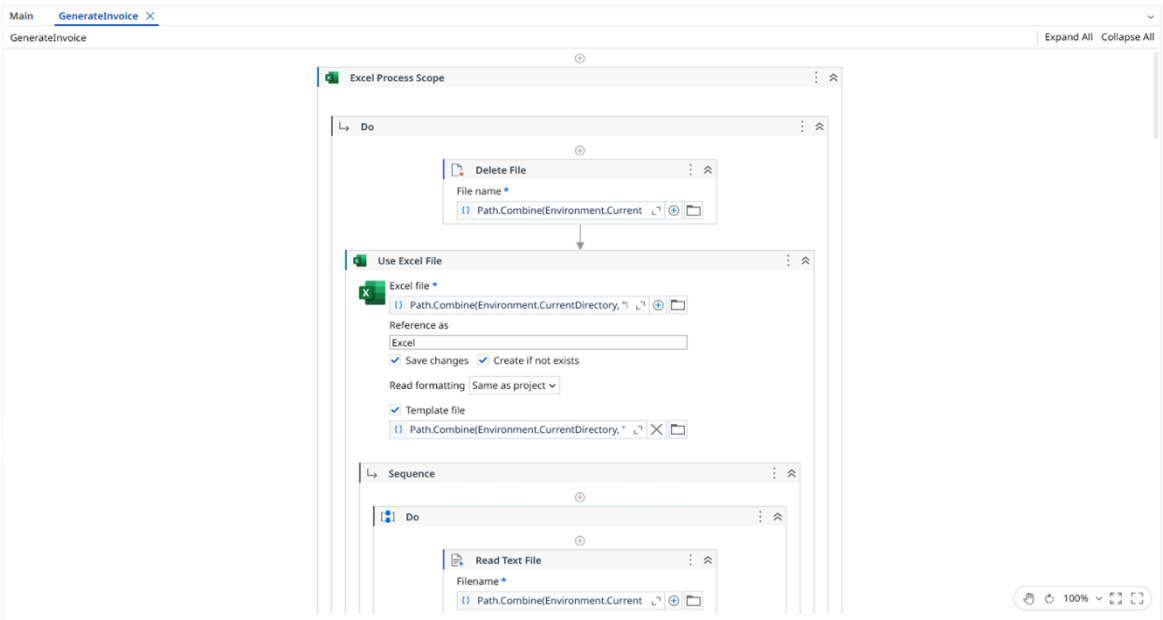


Kuva 7. ReadCalendar-workflow debug-tulos: esimerkki poimitusta tilausdatasta.

3.4 Workflow: GenerateInvoice

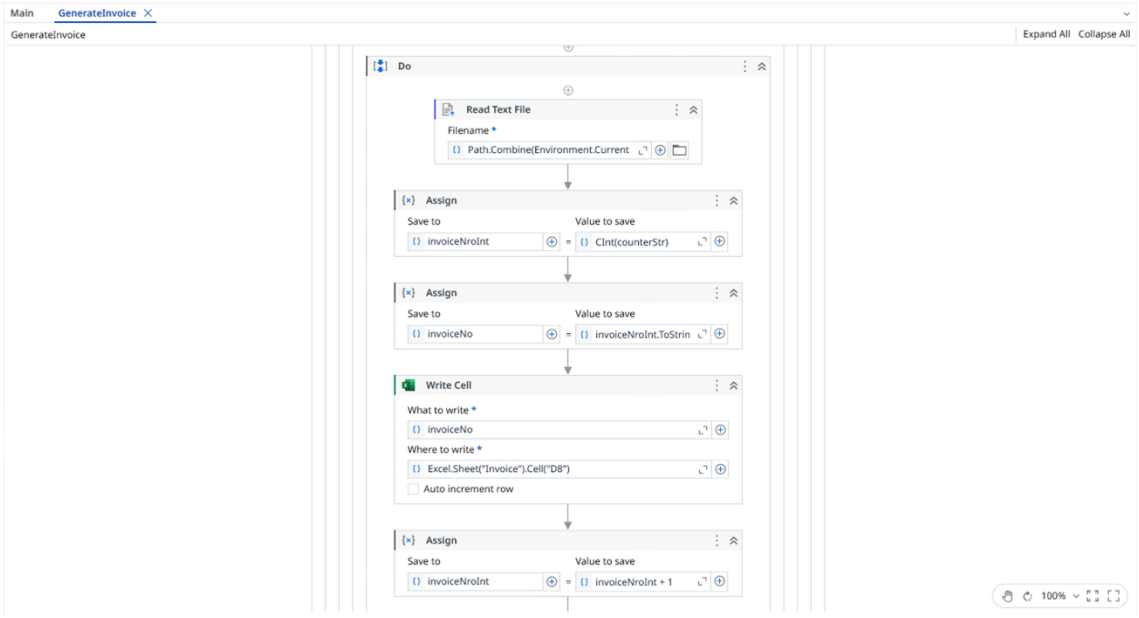
GenerateInvoice-workflow'n tehtävänä oli luoda lasku *ReadCalendar*-workflow'n tuottamasta DataTable-datasta käyttämällä muokattua Excel-laskupohjaa (template). Tämän vaiheen tavoitteena oli automatisoida laskun muodostus siten, että se luo laskun jokaisesta tilauksesta, lisää siihen asiakkaan tiedot ja tallentaa laskun sekä XLSX- että PDF-muodossa myöhempiä vaiheita varten.

Prosessi alkaa *Excel Process Scope* -rakenteella, jossa avataan laskupohja templatena. Ennen ajoa workflow tyhjentää mahdolliset aiempien suoritusten väliaikaistiedostot, jotta uusi lasku muodostuu aina puhtaaseen pohjaan (kuva 8).



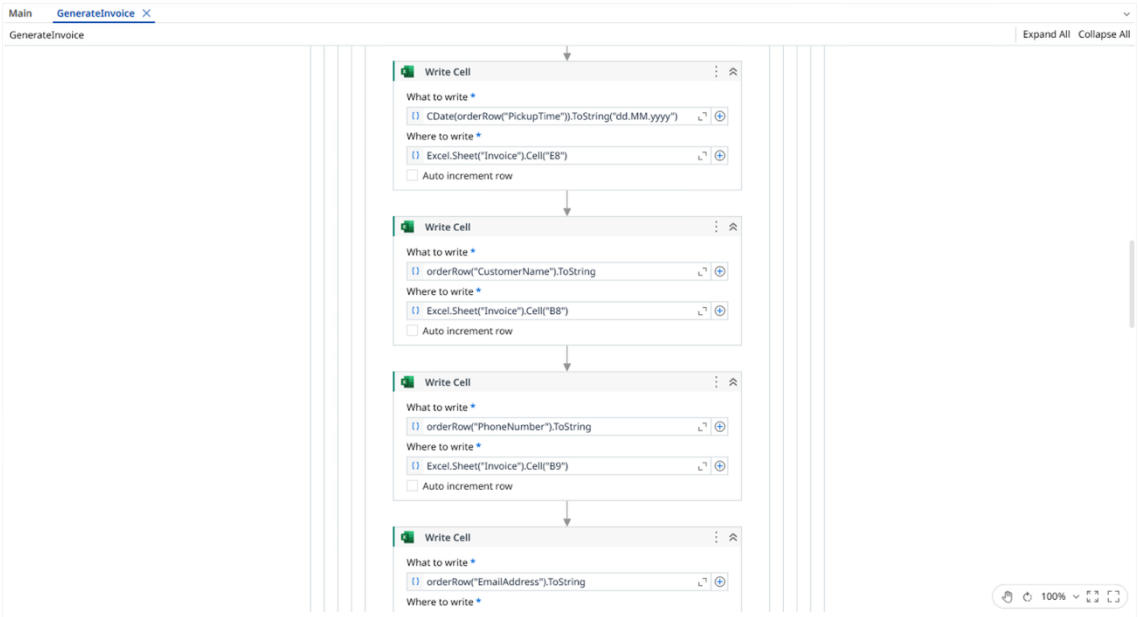
Kuva 8. GenerateInvoice: Excel Process Scope ja laskupohjan avaaminen.

Laskunumeron käsittely automatisoitiin hyödyntämällä erillistä *counter.txt*-tiedostoa, joka sisältää viimeksi käytetyn laskunumeron. Workflow lukee arvon tiedostosta, muuntaa sen kokonaisluvuksi, kasvattaa sitä yhdellä ja kirjoittaa päivitetyn numeron laskupohjaan (soluun D8). Samalla uusi laskunumero tallennetaan takaisin laskuritiedostoon seuraavaa laskua varten (kuva 9).



Kuva 9. GenerateInvoice: laskunumeron automaattinen käsittely.

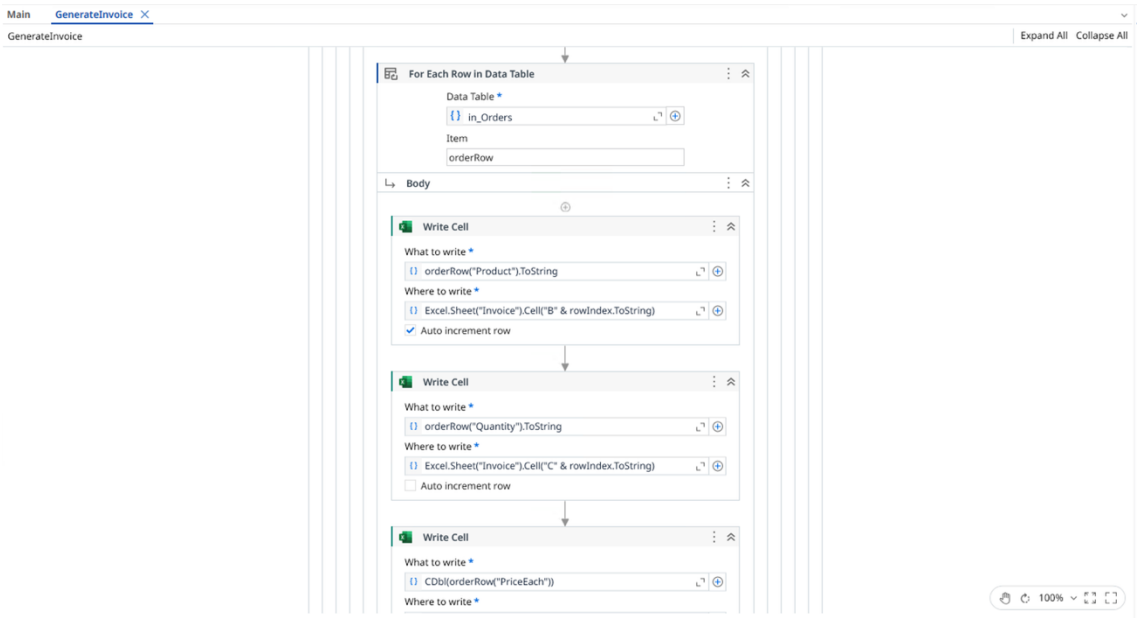
Seuraavaksi workflow täyttää laskun otsikkotiedot asiakkaan tiedoilla, jotka se lukee *orderRow*-datasta. Näitä ovat asiakkaan nimi, puhelinnumero, sähköpostiosoite sekä tilauksen päivämäärä, joka muunnetaan automaattisesti suomalaiseen muotoon. Tiedot kirjoitetaan laskupohjan soluihin B8–B10 ja E8 (kuva 10).



Kuva 10. GenerateInvoice: asiakastietojen kirjoittaminen laskupohjaan.

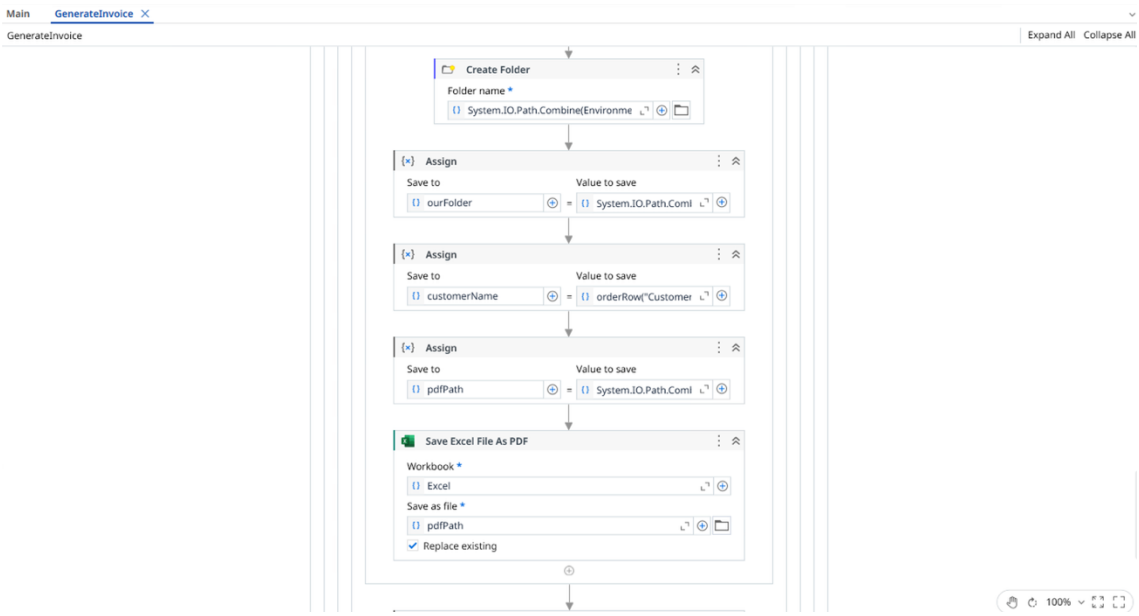
Kun asiakastiedot on lisätty, workflow täyttää tilauksen tuoterivit silmukassa. Määritetty aloitusrivi on 16, ja jokaiselle riville kirjoitetaan tuotenimi,

kappalemäärä ja yksikköhinta. Jokaisen tuotteen jälkeen rivinumero kasvaa yhdellä, jolloin useamman tuotteen tilaukset muodostuvat oikein (kuva 11).



Kuva 11. GenerateInvoice: asiakastietojen kirjoittaminen laskupohjaan.

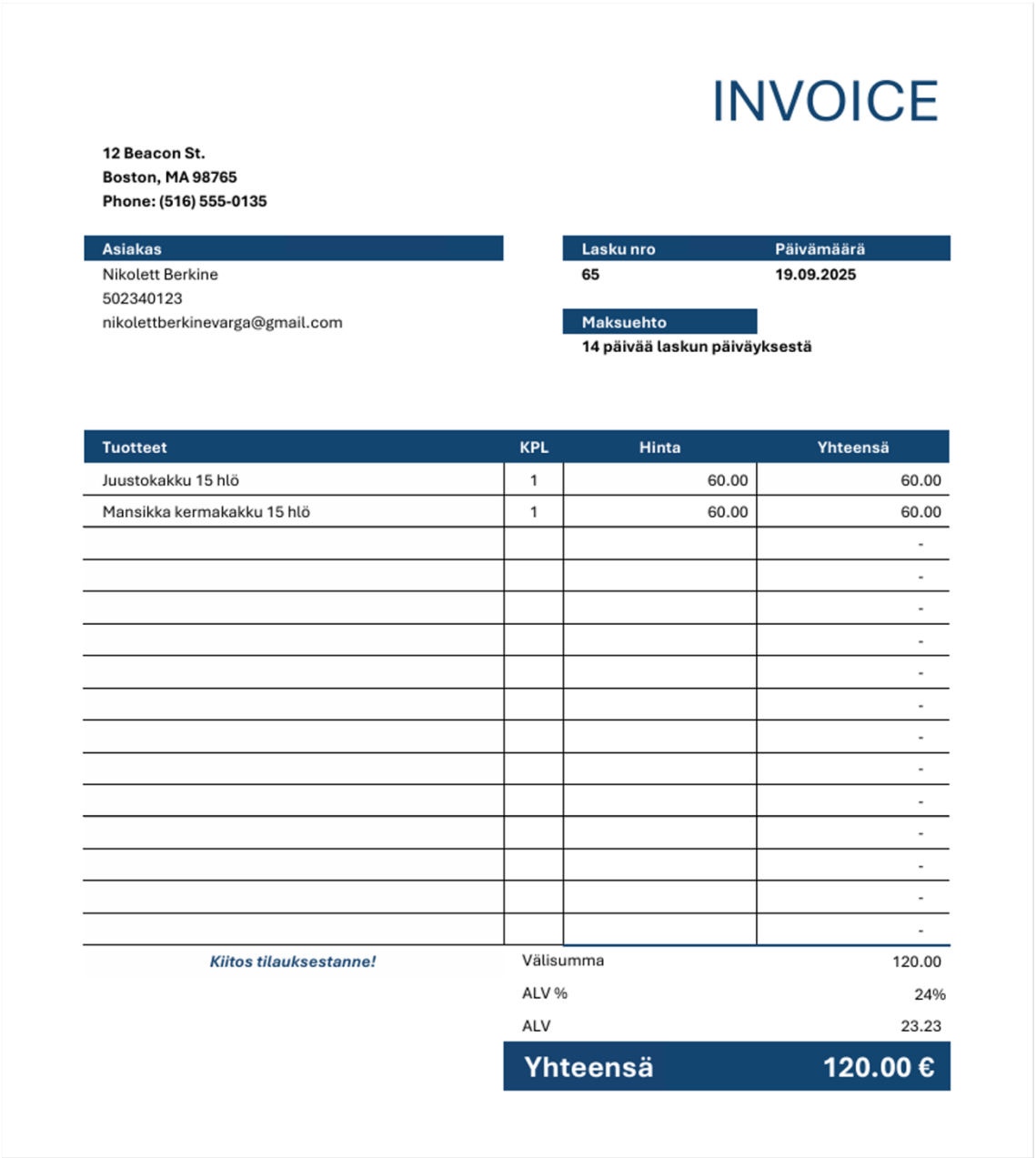
Laskun luomisen jälkeen workflow huolehtii tiedostojen tallennuksesta ja polkujen määrittelystä. Järjestelmä luo automaattisesti asiakaskohtaisen kansiorakenteen “Invoices”-kansioon, jossa laskut tallennetaan sekä Excel- että PDF-muodossa. Tiedostonimeen lisätään päivämäärä ja asiakkaan nimi, jotta vältetään ylikirjoitus (kuva 12).



Kuva 12. GenerateInvoice: tallennus ja PDF:n muodostaminen.

Kehitysvaiheessa ilmeni useita haasteita, kuten monituoterivien käsittely, tiedostopolkujen hallinta ja tietotyyppien muunnokset. Aluksi Excel-pohja täyttyi virheellisesti, kun tilaus sisälsi useita rivejä, mutta ongelma ratkaistiin indeksiohjauksella, jossa jokainen rivi kirjoitetaan yksitellen. Polkuongelmat korjattiin rakentamalla kansiorakenne dynaamisesti, ja tiedostojen nimet normalisoitiin, jotta ne säilyvät yksilöllisinä. Tyyppimuunnokset ratkaistiin varmistamalla, että hinnat ja määrät muunnettiin numeeriseen muotoon ennen laskentakaavoja.

Lopullinen GenerateInvoice-workflow tuottaa valmiin laskun automaattisesti asiakkaan tilaustietojen perusteella (kuva 13). Ratkaisu tukee useita tuoterivejä per tilaus, numeroi laskut automaattisesti ja palauttaa tiedostopolut seuraavia vaiheita varten, kuten validointiin, lähetykseen ja arkistointiin.



INVOICE

12 Beacon St.
Boston, MA 98765
Phone: (516) 555-0135

Asiakas
Nikolett Berkine
502340123
nikolettberkinevarga@gmail.com

Lasku nro	Päivämäärä
65	19.09.2025

Maksuehto
14 päivää laskun päiväyksestä

Tuotteet	KPL	Hinta	Yhteensä
Juustokakku 15 hlö	1	60.00	60.00
Mansikka kermakakku 15 hlö	1	60.00	60.00
			-
			-
			-
			-
			-
			-
			-
			-
			-
			-
			-
			-
			-

Kiitos tilauksestanne!

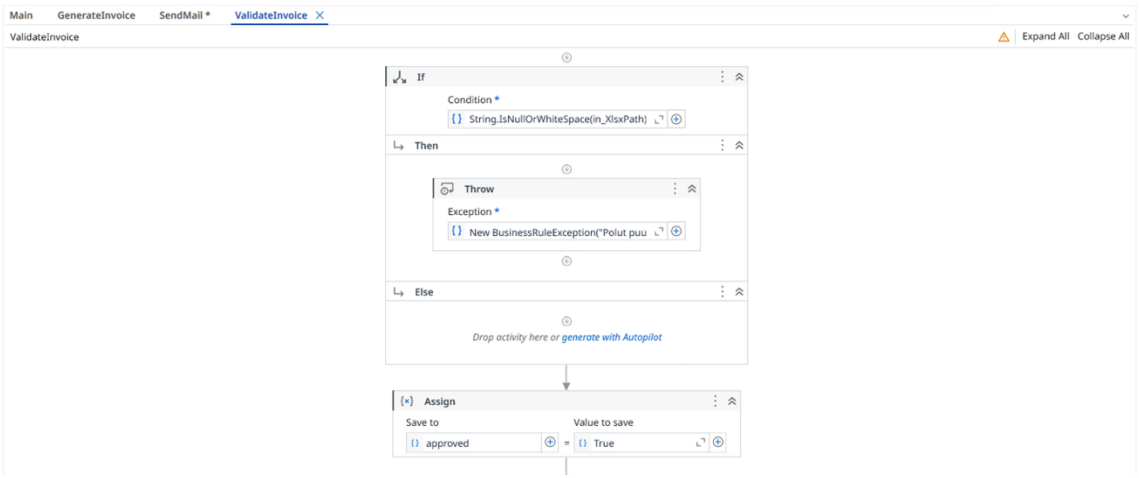
Välisumma	120.00
ALV %	24%
ALV	23.23
Yhteensä	**120.00 €**

Kuva 13. Esimerkki valmiista laskusta.

3.5 Workflow: ValidateInvoice

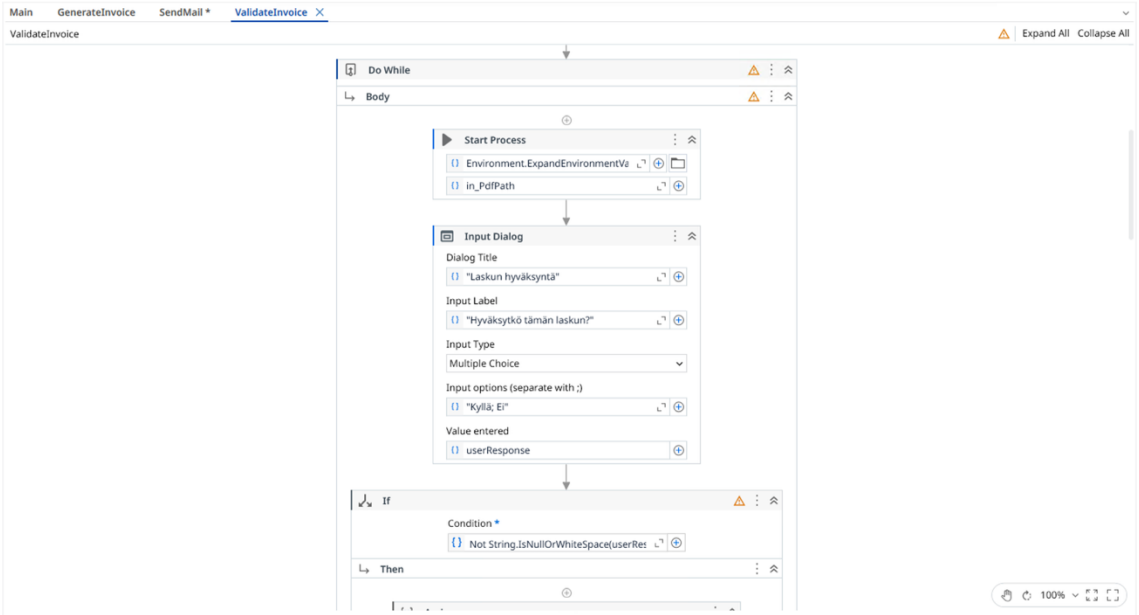
ValidateInvoice-workflow vastaa laadunvarmistuksesta ennen laskun lähettämistä asiakkaalle. Sen tarkoituksena on varmistaa, että automaattisesti luodun laskun tiedot ovat oikein ja että käyttäjä voi vielä tarkistaa tai tarvittaessa korjata ne ennen lähetystä. Tällä tavoin prosessiin lisätään manuaalinen tarkistusvaihe, joka ehkäisee virheitä ja varmistaa laskun sisällön oikeellisuuden.

Workflow käynnistyy, kun laskun tiedostopolut (*in_XlsxPath* ja *in_PdfPath*) on määritelty oikein. Jos polku puuttuu tai on virheellinen, prosessi pysähtyy ja antaa virheilmoituksen, mikä suojaa väärien laskujen käsittelyltä (kuva 14).



Kuva 14. ValidateInvoice: syötepolkujen tarkistus ja virheenkäsittely.

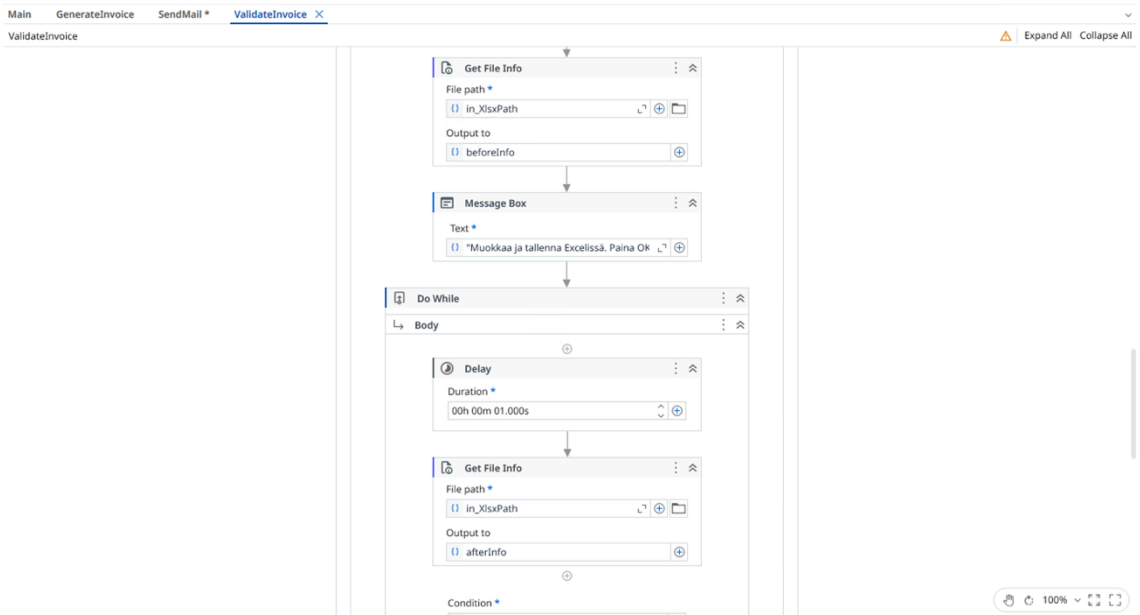
Kun tiedostopolut on vahvistettu, workflow avaa laskun käyttäjälle esikatseluun (Start Process) ja näyttää sen automaattisesti PDF-muodossa. Käyttäjältä kysytään hyväksyntä *Input Dialog*-toiminnon avulla. Näytölle avautuu kysymys: “Hyväksytkö tämän laskun?” ja vaihtoehtoina ovat *Kyllä* tai *Ei* (kuva 15).



Kuva 15. ValidateInvoice: käyttäjän hyväksyntäkysely Input Dialogilla.

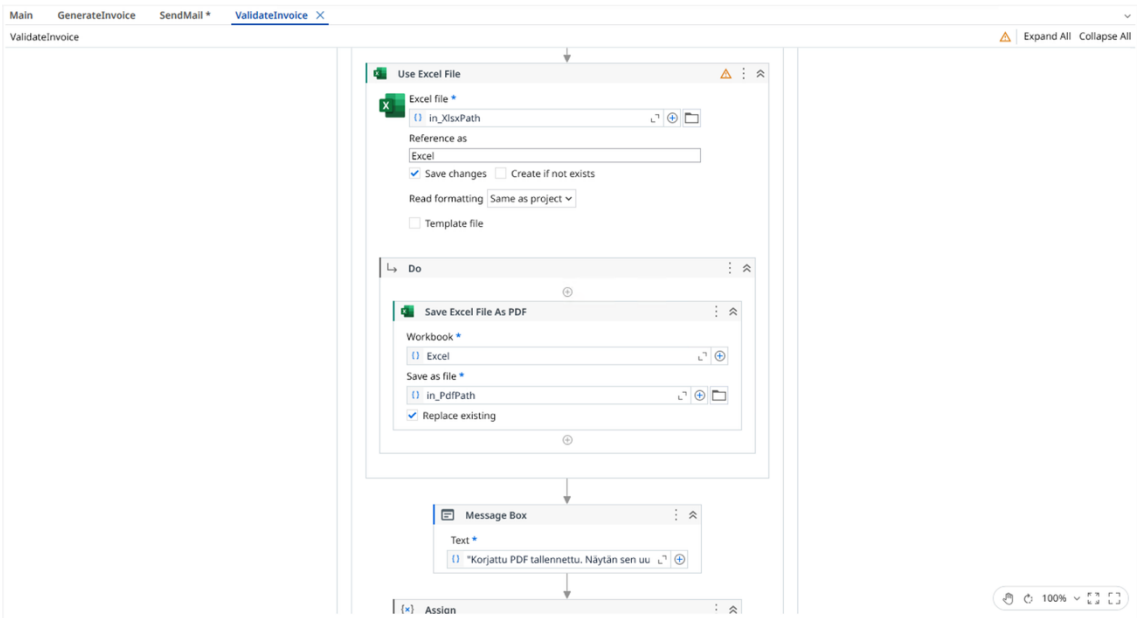
Jos käyttäjä hyväksyy laskun, arvo *approved* asetetaan todeksi ja workflow jatkaa prosessia eteenpäin. Jos käyttäjä hylkää laskun, ohjelma avaa laskun

automaattisesti Excelissä, jolloin käyttäjä voi tehdä tarvittavat korjaukset. Näytölle tulee viesti, jossa ohjeistetaan: “Muokkaa ja tallenna Excelissä, paina OK.” Tämän jälkeen workflow seuraa tiedoston muokkausaikaa ja odottaa, kunnes käyttäjä on tallentanut muutokset (kuva 16).



Kuva 16. ValidateInvoice: hylätyn laskun avaus Excelissä ja ohjeistus muokkaamiseen.

Kun korjaukset on tallennettu, workflow muodostaa laskusta automaattisesti uuden PDF-version (Save Excel File as PDF) ja näyttää sen käyttäjälle esikatseluna. Näin käyttäjä voi tarkistaa, että korjattu versio on oikea ennen prosessin jatkamista (kuva 17).



Kuva 17. ValidateInvoice: korjatun laskun uudelleentallennus PDF-muodossa.

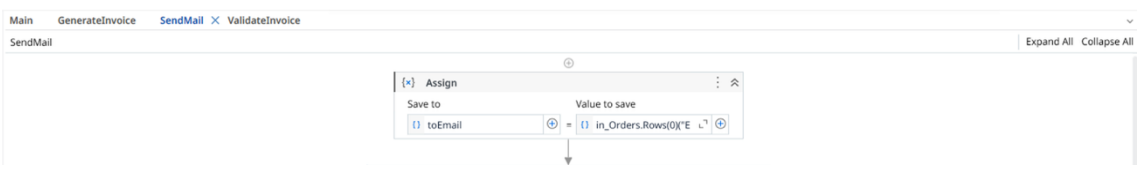
Kehitysvaiheessa suurimmat haasteet liittyivät siihen, miten workflow saatiin odottamaan käyttäjän muokkaukset ennen PDF:n uudelleentallennusta. Tämä ratkaistiin vertaamalla Excel-tiedoston muokkausaikoja ennen ja jälkeen korjauksen. Käyttöliittymä suunniteltiin yksinkertaiseksi ja selkeäksi, jotta prosessin käyttäminen ei vaadi teknistä osaamista. Tätä varten hyödynnettiin UiPathin *Input Dialog*– ja *Message Box*–aktiviteetteja, joilla viestintä ja ohjeistus saatiin käyttäjäystävälliseksi.

ValidateInvoice-workflow lisää automaatioprosessiin tärkeän laadunvarmistusvaiheen. Sen avulla jokainen lasku voidaan hyväksyä sellaisenaan tai palauttaa muokattavaksi ennen lähettämistä. Ratkaisu vähentää virheiden riskiä ja lisää loppukäyttäjän hallintaa laskuprosessin lopputuloksesta.

3.6 Workflow: SendMail

SendMail-workflow vastaa laskujen lähettämisestä asiakkaalle sähköpostitse. Sen tavoitteena on automatisoida laskujen jakelu siten, että hyväksytty ja tallennettu lasku lähtee oikealle vastaanottajalle liitetiedostona ilman manuaalista työtä. Ratkaisun avulla yrityksen laskutusprosessi nopeutuu merkittävästi, ja virheiden riski sähköpostiosoitteiden käsittelyssä pienenee.

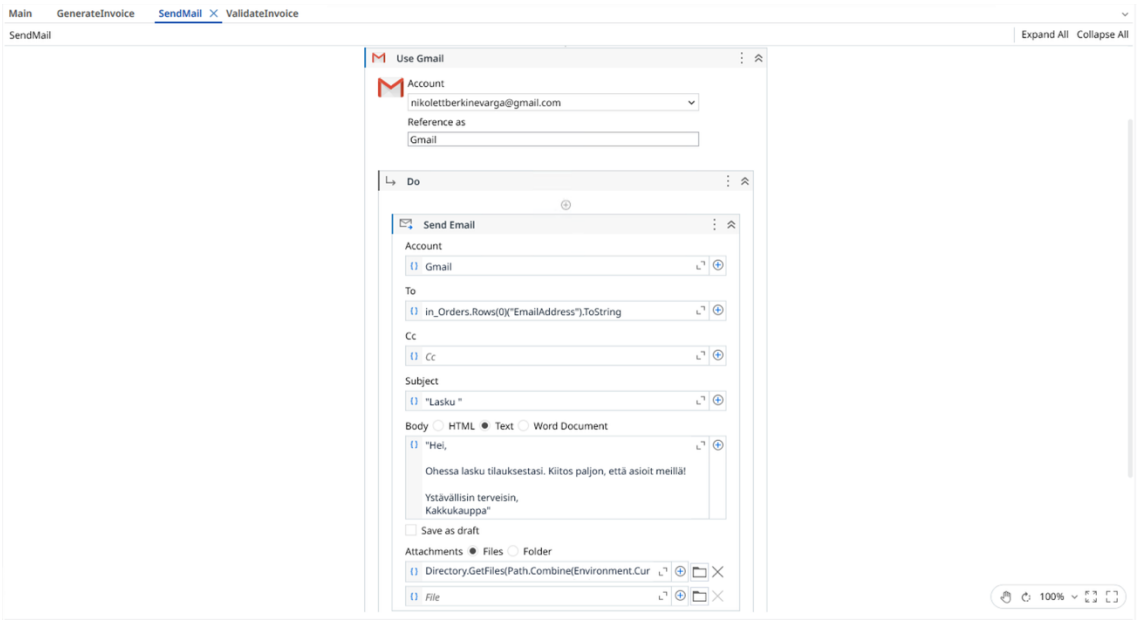
Workflow hyödyntää UiPathin *Use Gmail* –rakennetta, jonka avulla muodostetaan yhteys käyttäjän Gmail-tiliin ja sallitaan sähköpostien lähetys ohjelmallisesti. Viestin lähetys toteutetaan *Send Email* -aktiviteetilla, jossa määritetään vastaanottajan sähköpostiosoite, aihe ja viestin sisältö. Prosessi käynnistyy, kun lasku on tallennettu PDF-muotoon, ja se käyttää *DataTable*-rakenteesta haettua asiakkaan sähköpostiosoitetta (emailAddress) viestin vastaanottajana (kuva 18).



Kuva 18. SendMail: asiakkaan sähköpostiosoitteen lukeminen ja tallentaminen muuttujaan toEmail.

PDF-lasku lisätään viestin liitteeksi automaattisesti. Liitetiedoston polku määritetään aiemmassa vaiheessa luodun laskun tallennussijainnin perusteella, jolloin ohjelma hakee tiedoston polun *in_PdfPath*-muuttujasta. Viesti lähetetään automaattisesti ilman käyttäjän erillistä vahvistusta, mikä tekee prosessista täysin unattended-tyyppisen.

Prosessiin sisältyy myös varmistusvaihe, jossa ohjelma tarkistaa, että sähköpostiosoite on oikeassa muodossa ennen lähetystä. Jos osoite puuttuu tai on virheellinen, workflow ei lähetä viestiä, vaan siirtää laskun erilliseen kansioon manuaalista tarkistusta varten (kuva 19). Tämä suojaa väärien osoitteiden aiheuttamilta lähetysvirheiltä.



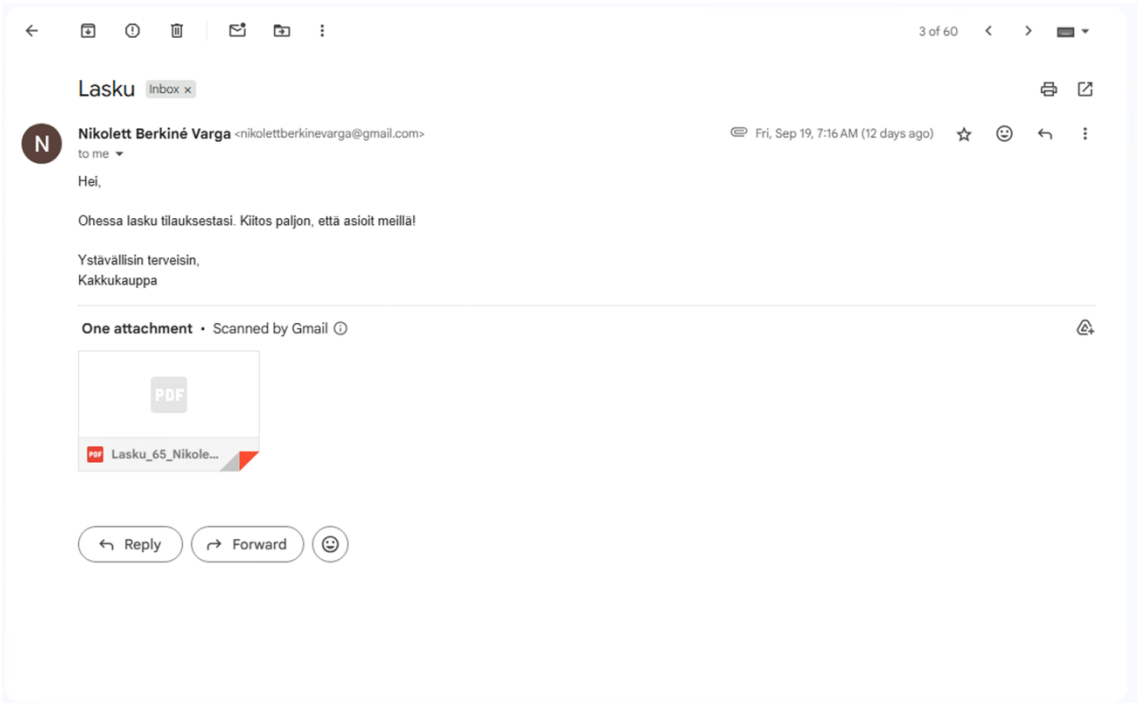
Kuva 19. SendMail: laskun lähettäminen asiakkaalle sähköpostilla Gmail-aktiviteetin avulla.

Lopuksi workflow kirjaa onnistuneen lähetyksen lokiin (*log.txt*) ja tulostaa konsoliin viestin, jossa näkyy vastaanottajan sähköpostiosoite ja tiedot lähetetystä laskusta. Tämä luo läpinäkyvyyttä prosessin etenemiseen ja helpottaa mahdollisten poikkeamien seurantaa.

Testausvaiheessa havaittiin, että Outlookin *Send Outlook Mail Message* -aktiviteetti ei toiminut luotettavasti PDF-laskujen liittämisessä, mikä johti virheisiin erityisesti tilanteissa, joissa tiedosto oli juuri luotu tai käytössä. Tämä ongelma ratkaistiin siirtymällä *Use Gmail* -rakenteeseen, joka osoittautui

vakaammaksi ja paremmin soveltuvaksi tähän käyttötapaukseen. Gmail-toteutus mahdollisti myös liitetiedostojen käsittelyn sujuvasti ilman ylimääräisiä viiveitä tai virheilmoituksia.

SendMail-workflow viimeistelee automaattisen laskutusprosessin varmistamalla, että hyväksytyt laskut toimitetaan oikeille vastaanottajille nopeasti ja virheettömästi sähköpostitse (kuva 20). Ratkaisu vähentää manuaalista sähköpostityötä ja tukee koko prosessin automatisoitua työnkulkua yhdessä arkistointivaiheen kanssa.

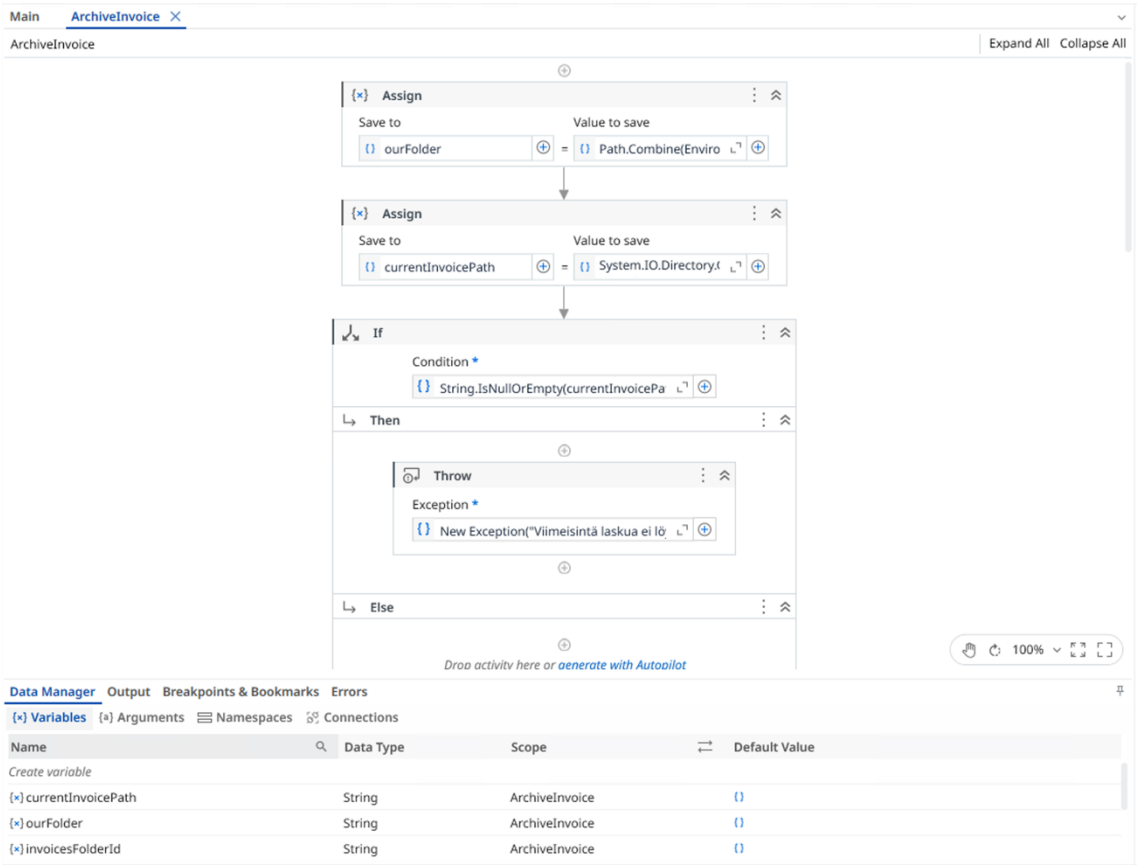


Kuva 20. Asiakkaalle saapunut sähköpostiviesti, jossa lasku toimitetaan PDF-liitteenä.

3.7 Workflow: ArchiveInvoice

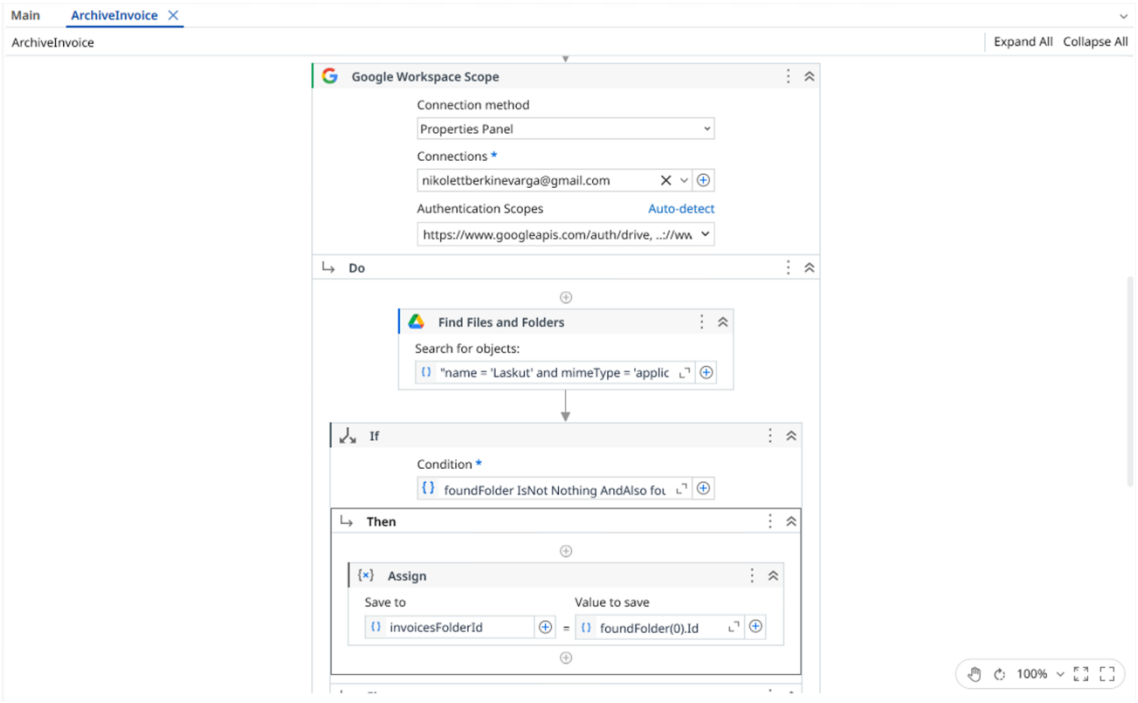
ArchiveInvoice-workflow vastaa automaattisesti luotujen laskujen tallentamisesta ja arkistoinnista Google Drive -pilvipalveluun. Sen tavoitteena on varmistaa, että jokaisesta lähetetystä laskusta jää pysyvä tallenne PDF-muodossa, ja että tiedostot sijoitetaan hallitusti oikeaan kansiorakenteeseen. Jokainen lasku tallennetaan samaan *“Laskut”*-kansioon, ja tiedoston nimi muodostetaan laskunumeron ja asiakkaan nimen perusteella, mikä helpottaa myöhempää hakua ja kirjanpitoa.

Workflow alkaa muodostamalla yhteyden käyttäjän Google Drive -tiliin *Google Workspace Scope* -rakenteen avulla (kuva 21). Tämä aktiviteetti hallinnoi autentikoinnin G Suite-palveluihin ja varmistaa, että kaikki seuraavat vaiheet suoritetaan oikeuksien puitteissa.



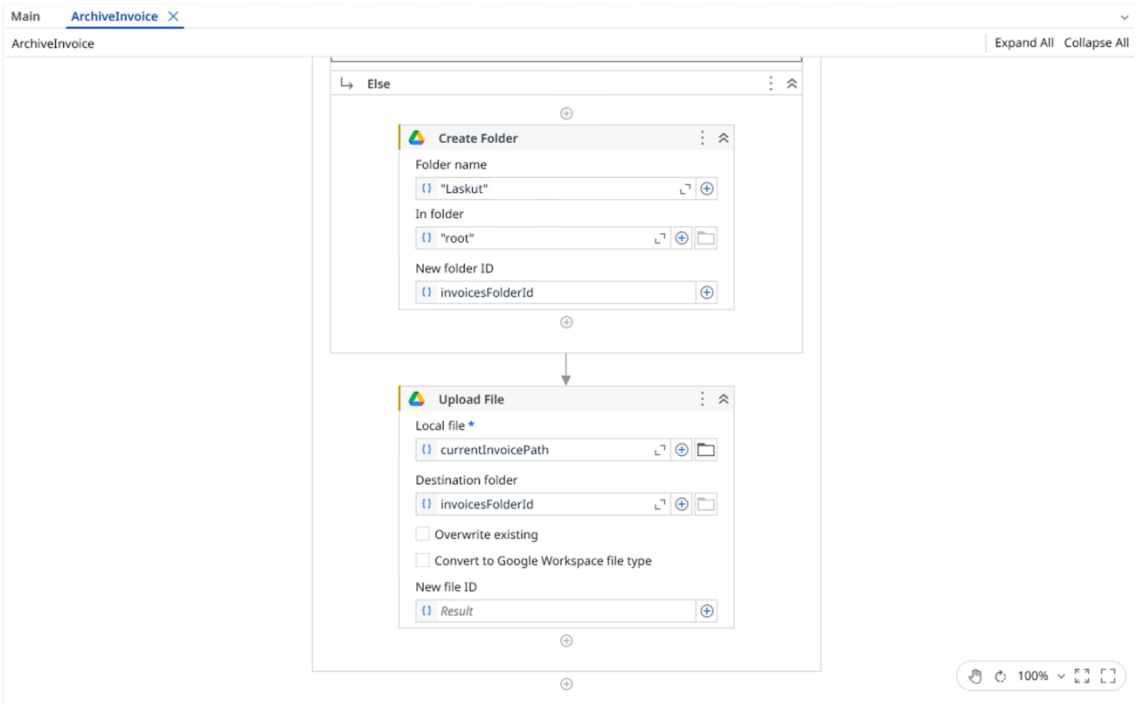
Kuva 21. ArchiveInvoice: tiedostopolkujen määrittäminen ja virheenkäsittely.

Tämän jälkeen workflow hakee *Find Files and Folders* -aktiviteetilla olemassa olevan “Laskut”-kansion. Jos kansiota ei löydy, ohjelma luo uuden kansion *Create Folder* -aktiviteetilla ja tallentaa sen tunnisteen myöhempää käyttöä varten (kuva 22).



Kuva 22. ArchiveInvoice: Google Workspace Scope -yhteyden muodostus ja kansion tarkistus.

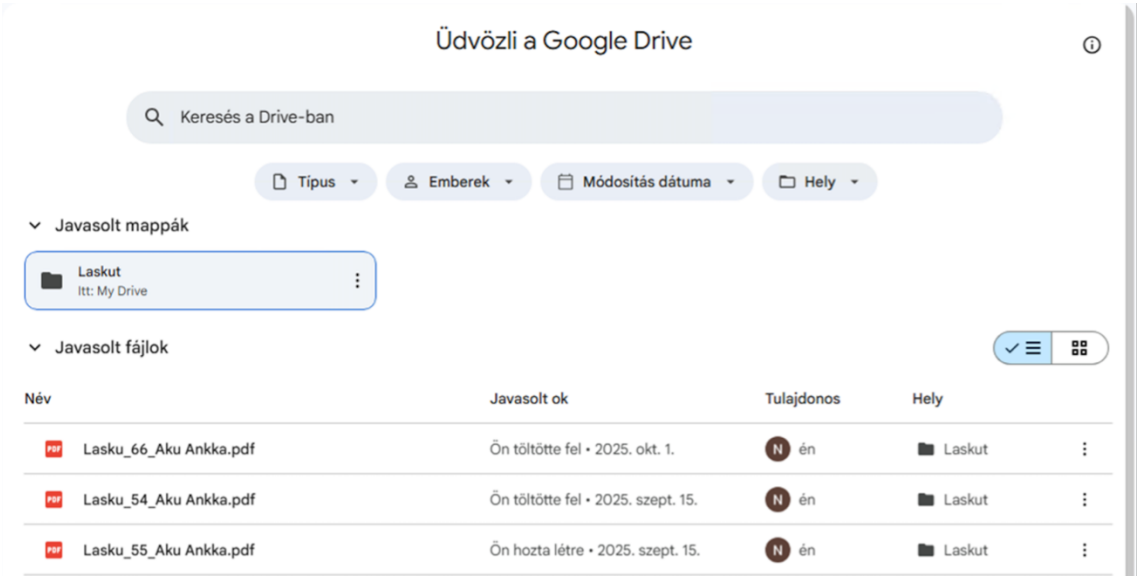
Kun kansiorakenne on vahvistettu, workflow siirtää laskun *Upload File* -aktiviteetilla. Tiedosto ladataan automaattisesti Google Driveen määritettyyn kansioon, ja jos kansiossa on jo samanniminen tiedosto, ohjelma lisää tiedostonimeen aikaleiman ylikirjoituksen estämiseksi (kuva 23).



Kuva 23. ArchiveInvoice: uuden *“Laskut”*-kansion luominen ja laskun lataus Driveen.

Kehitystyön aikana havaittiin, että Google Drive -tallennuksen hyödyntäminen helpotti laskujen pitkäaikaista säilytystä ja varmuuskopiointia. Ratkaisu osoittautui vakaaksi, ja yhteys muodostui automaattisesti *Google Workspace Scope* -rakenteen kautta jokaisen ajon yhteydessä ilman manuaalista todennusta.

Lopullinen ArchiveInvoice-workflow viimeistelee laskutusprosessin siirtämällä hyväksytyt ja lähetetyt laskut automaattisesti pysyvään pilviarkistoon (kuva 24). Ratkaisu varmistaa laskudatan säilyvyyden, eheän tallennusrakenteen ja tarjoaa yritykselle luotettavan tavan hallita kirjanpitoaineistoa digitaalisesti.



Kuva 24. Valmiit laskut arkistoituna automaattisesti Google Driven *“Laskut”*-kansioon. Ratkaisu varmistaa, että kaikki laskut tallentuvat oikeaan paikkaan ilman manuaalista työtä.

3.8 Workflow: OCR-Reader

Elintarvikealan yrityksissä asiakastilaukset vastaanotetaan usein useiden eri kanavien kautta: sähköpostilla, puhelimitse tai paperilomakkeina. Näiden tilausten käsittely tapahtuu tyypillisesti manuaalisesti, jolloin työntekijä siirtää tiedot järjestelmään tai laskutuspohjaan käsin. Tämä vie aikaa ja lisää virheiden riskiä, kuten virheellisiä summia, puuttuvia tietoja tai kaksinkertaisia kirjauksia.

Lisäksi tieto on hajallaan eri lähteissä, mikä vaikeuttaa kokonaisprosessin hallintaa ja raportointia.

Tällaisessa ympäristössä perinteinen ohjelmistorobotiikka (Robotic Process Automation, RPA) kohtaa rajoituksia. RPA toimii tehokkaasti, kun käsiteltävä tieto on rakenteellista – esimerkiksi taulukkomuotoista, tietokantaan tallennettua tai selkeissä kentissä sijaitsevaa digitaalista dataa. Paperiset ja kuvamuotoiset tilaukset sen sijaan sisältävät *epärakenteista dataa*, jota RPA ei voi tulkita suoraan, koska teksti on kuva-aineiston sisällä eikä koneen luettavissa.

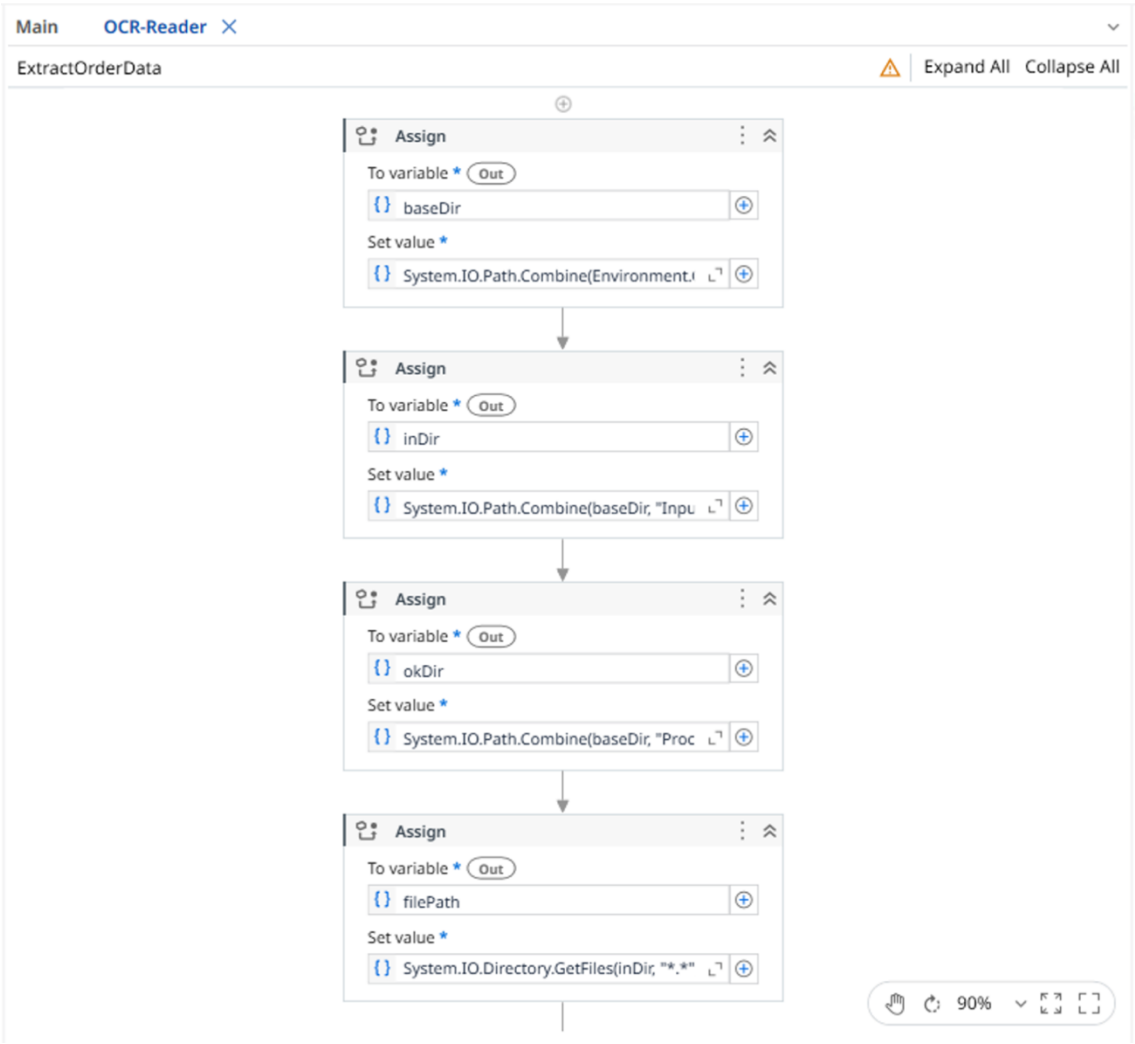
OCR-teknologia (Optical Character Recognition) ratkaisee tämän ongelman tunnistamalla ja tulkitsemalla kuvassa olevan tekstin. Sen avulla tekstisisältö voidaan muuntaa digitaaliseen, rakenteiseen muotoon, jota robotti pystyy käsittelemään automaattisesti. Käytännössä tämä tarkoittaa, että myös skannatut tai valokuvat tilauslomakkeista voidaan lukea ja käsitellä samalla automaatiolla kuin sähköiset tilaukset. OCR siis toimii siltana manuaalisen ja digitaalisen tiedonhallinnan välillä.

Tässä opinnäytetyössä OCR-Reader-workflow toimii nimenomaan tämän sillanrakentajana. Se poistaa tarpeen manuaaliselle tietojen siirrolle ja mahdollistaa sen, että kaikki tilaukset – riippumatta niiden muodosta – siirtyvät automaattisesti digitaaliseen laskutusprosessiin. Tämä tekee kokonaisuudesta yhtenäisen, virheettömämmän ja huomattavasti tehokkaamman.

OCR-Reader-workflow’n tarkoituksena oli automatisoida paperimuotoisten ja kuvamuodossa olevien tilausten tunnistaminen ja muuntaminen digitaaliseen muotoon jatkokäsittelyä varten. Workflow hyödyntää OCR-teknologiaa (Optical Character Recognition), joka lukee asiakastilaukset skannatuista tiedostoista, kuten PDF- tai kuvamuodoista, ja tulkitsee niistä olennaiset tiedot – asiakkaan nimen, yhteystiedot, tuotteet, määrät, hinnat sekä nouto- tai toimituspäivän. Tavoitteena oli vähentää manuaalista tietojen syöttämistä ja mahdollistaa tilausten siirtäminen automaattisesti laskutusprosessiin. Näin myös käsin kirjoitetut tai tulostetut tilaukset voitiin integroida osaksi digitaalista työnkulkua. OCR-Reader loi pohjan yhtenäiselle laskutusautomaatiolle, joka kattaa sekä sähköiset että paperimuotoiset tilaukset.

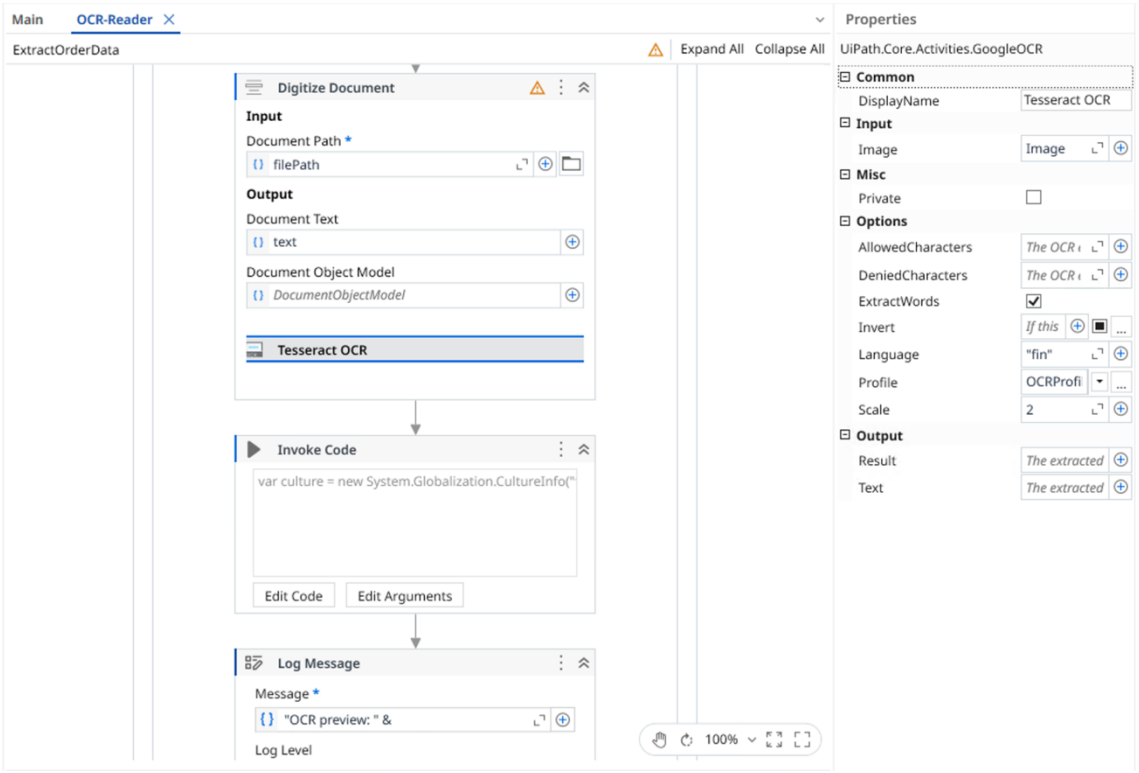
Alkuperäisen suunnitelman mukaan OCR-komponentin oli tarkoitus lukea paperimuotoinen tilaus suoraan laskupohjaan, jolloin tilauksen tiedot siirtyisivät välittömästi laskutukseen. Jo ennen varsinaisen OCR-Reader-workflow'n rakentamista arkkitehtuuria kuitenkin muutettiin parantamaan ylläpidettävyyttä, virheenkäsittelyä ja prosessin hallittavuutta. Muutoksen myötä OCR ei enää vienyt tietoja suoraan laskupohjaan, vaan kirjaa ne Google Kalenteriin tapahtumana, jolloin kaikki tilaukset – riippumatta siitä, olivatko ne sähköisiä vai skannattuja – kulkevat saman automaatioputken kautta (ReadCalendar → GenerateInvoice → ValidateInvoice → SendMail → ArchiveInvoice). Tämä ratkaisu teki järjestelmästä yhtenäisemmän ja hallittavamman sekä mahdollisti tehokkaan virheenkäsittelyn ja prosessin laajennettavuuden.

Prosessi alkaa, kun työnkulku tunnistaa *InputOrders*-kansioon lisätyn uuden kuvan tai PDF-tiedoston. Tunnistetut tiedostopolut ja käsiteltävien tilausten lähdekansiot alustetaan ennen varsinaista OCR-käsittelyä, jotta tiedostojen käsittely ja virhetilanteiden hallinta voidaan toteuttaa hallitusti (kuva 25).



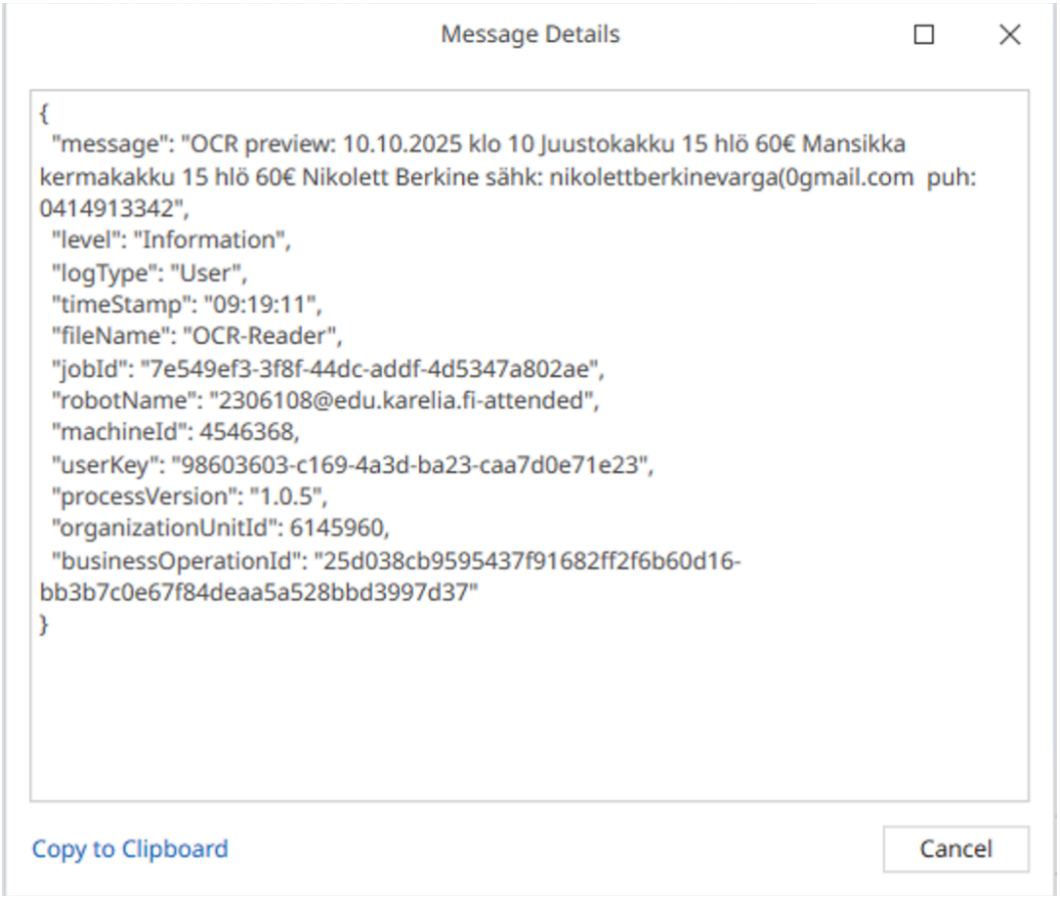
Kuva 25. Polkujen alustukset: baseDir, inDir, okDir.

Tämän jälkeen tiedosto luetaan *Digitize Document* -aktiviteetilla hyödyntäen Tesseract OCR -moottoria, jossa kieliasetuksena on suomi. OCR-tunnituksen tuloksena dokumentin sisältämä teksti muunnetaan raakatekstimuotoon ja tallennetaan muuttujaan *text*, jota käytetään pohjana tietojen jäsentämisessä (kuva 26).



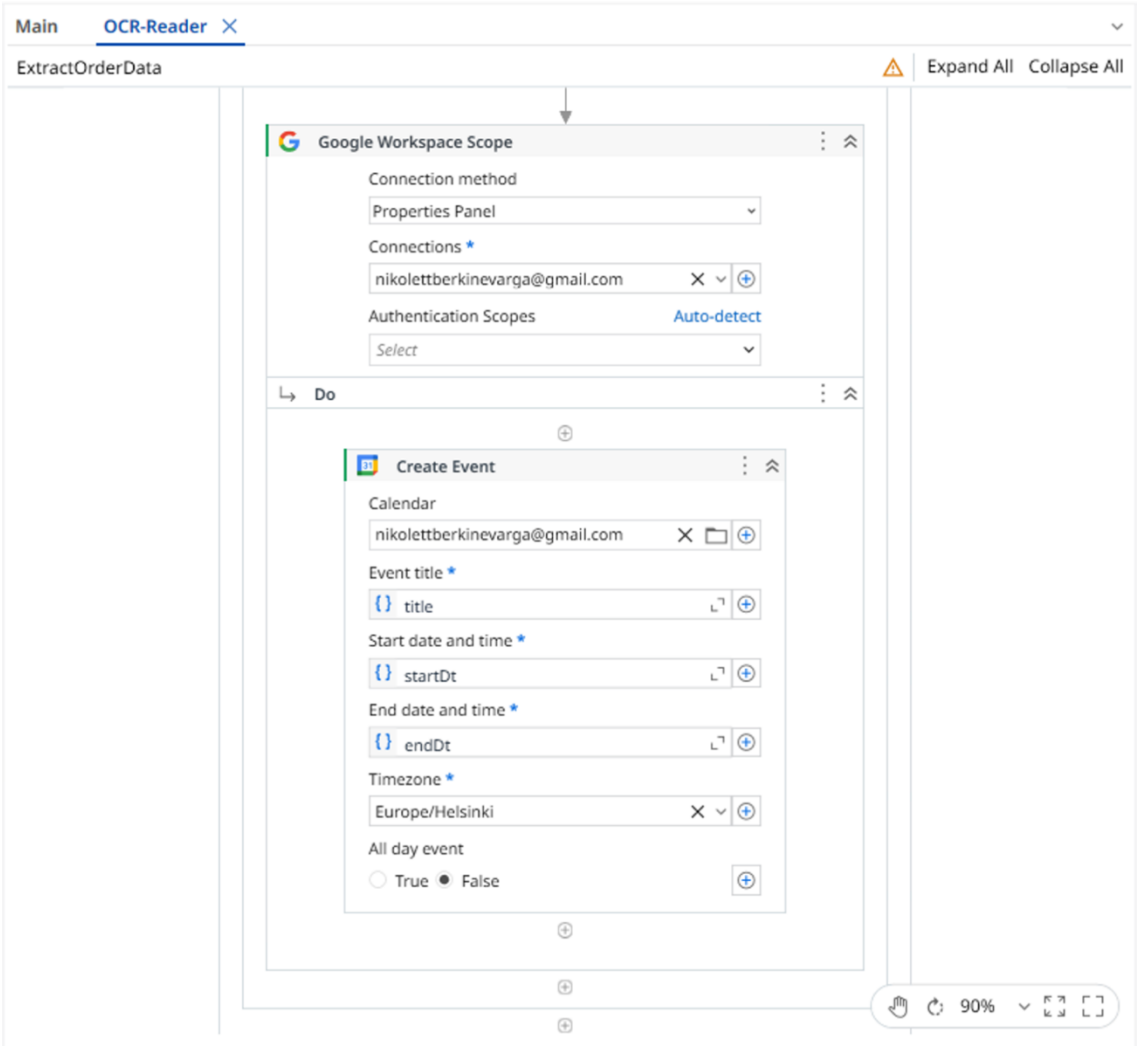
Kuva 26. Kuvassa näkyy Digitize Document ja sen alla Tesseract OCR (Input = filePath, Output = text).

Seuraavaksi tekstin jäsentäminen suoritetaan *Invoke Code* -vaiheessa C#-koodin avulla. Tässä vaiheessa raakateksti puhdistetaan ylimääräisistä merkeistä, rivinvaihdoista ja näkymättömistä välilyönneistä, minkä jälkeen koodi poimii olennaiset tiedot säännöllisten lausekkeiden (Regex) avulla. Ohjelma tunnistaa päivämäärät ja kellonajat, tuoterivit ja hinnat, sekä asiakkaan nimen, sähköpostiosoitteen ja puhelinnumeron. Esimerkiksi päivämäärä ilmaisuissa kuten “10.10.2025 klo 10” erotellaan automaattisesti päivä ja aika, ja hinnat tunnistetaan euro-symbolin perusteella. Tämän jälkeen teksti normalisoidaan suomalaiseen muotoon (fin), jotta numerot, desimaalit ja päivämäärät tallentuvat yhdenmukaisesti (kuva 27).



Kuva 27. Invoke Code -vaiheen muodostama jäsennelty OCR-tulos. Kuvassa näkyy, miten koodi on poiminut tilauksesta päivämäärän, tuotteet, hinnat sekä asiakkaan yhteystiedot ja tallentanut ne lokiin myöhempää käyttöä varten.

Kun tiedot on poimittu ja jäsennelty, workflow luo uuden tapahtuman Google Kalenteriin. Tämä tapahtuu *Google Workspace Scope* -yhteyden kautta, jossa *Create Event* -aktiviteetti tallentaa asiakkaan tilauksen kalenterimerkinnäksi. Tapahtuman otsikoksi asetetaan asiakkaan nimi tai “Tilaus”, ja kuvaukseen lisätään kaikki tunnistetut tiedot, kuten tuotteet, määrät ja yhteystiedot (kuva 28).



Kuva 28. OCR-Reader workflowin Google Workspace Scope ja sen sisällä Create Event -aktiviteetti.

Kehitysvaiheessa havaittiin useita haasteita. Käsinkirjoitetun tekstin tunnistus oli epäluotettavaa, sillä kirjainten epätasaisuus, viivastot ja valaistuksen varjot aiheuttivat virheitä. Paperin tausta vaikutti myös tulokseen: musta tai tummasävyinen tausta heikensi tunnistusta merkittävästi, ja ruudullinen paperi sekoitti rivinvaihdot ja rivijärjestyksen. Lisäksi Tesseract OCR:n tunnistustarkkuus riippui huomattavasti kontrastista ja fontista – parhaat tulokset saavutettiin tietokoneella kirjoitetusta tekstistä vaalealla taustalla.

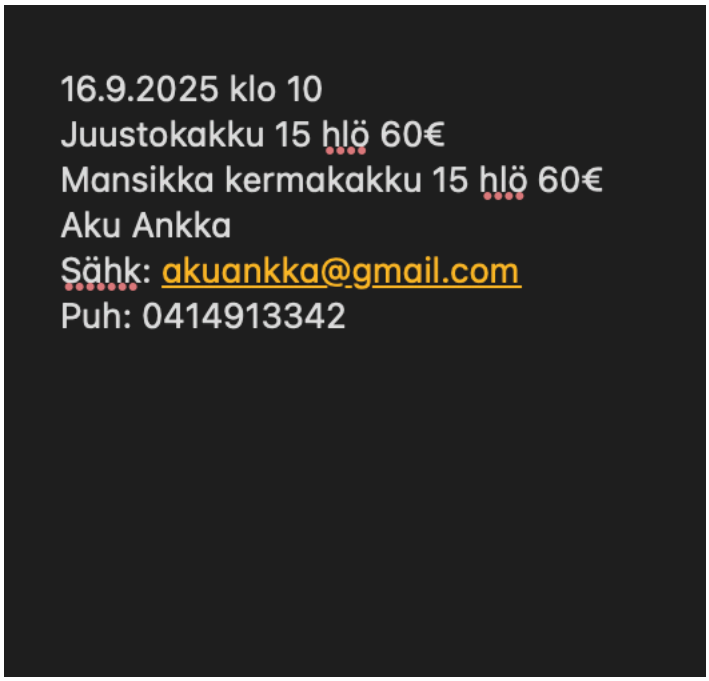
OCR:n kieliasetukset aiheuttivat alkuvaiheessa epäilyjä virheiden syistä. Aluksi ongelmien ajateltiin johtuvan siitä, ettei suomen kieltä tunnistettu oikein, mutta kun suomen OCR-paketti otettiin käyttöön, kävi ilmi, että ongelmat johtuivat pääasiassa kuvan laadusta eikä kielestä. Suomen kielen asettaminen kuitenkin paransi ääkkösten (ä, ö) ja numeroiden tulkintaa. Erikoismerkit, erityisesti “@”,

olivat sitkeimpiä virheiden lähteitä: OCR tunnisti ne usein virheellisesti symboleina kuten “(0” tai “ät”.

Näiden ongelmien ratkaisemiseksi toteutettiin useita ohjelmallisia parannuksia. Ensimmäiseksi OCR-tekstin normalisointi automatisoitiin siten, että siitä poistetaan kaikki erikoisvälilyönnit, ylimääräiset rivinvaihdot ja näkymättömät merkit. Rivien väliin lisättiin loogisia jakokohtia, kuten päivämäärän ja kellonajan väliin, jotta tekstin jäsentäminen helpottui. Lisäksi ajan ja päivämäärän tunnistus parannettiin siten, että kellonaika haetaan vain, jos tekstissä esiintyy sana “klo”, mikä poisti virheet, joissa esimerkiksi päivämäärä “17.10” tulkittiin ajaksi “17:10”. Tuoterivit tunnistettiin Regex-malleilla, jotka etsivät tuotteen nimen, henkilömäärän (esim. “15 hlö”) ja hinnan euroina, ja samalla estettiin puhelinnumeroiden virheellinen tulkinta hinnoiksi.

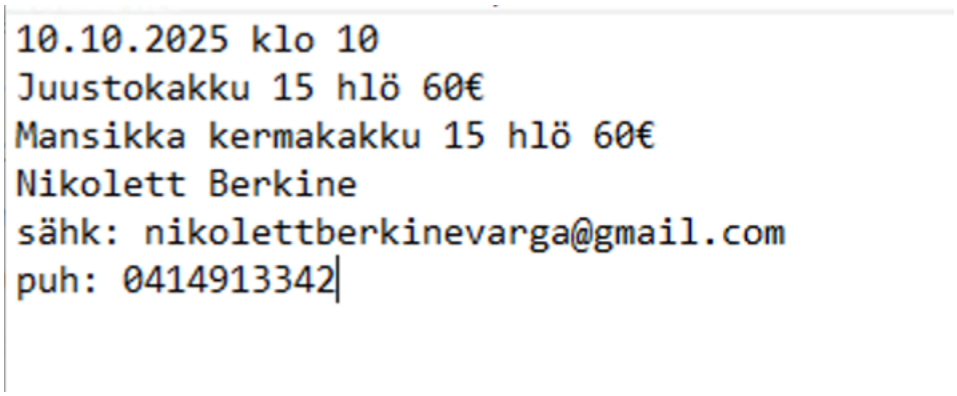
Erikoismerkkien korjaus tehtiin ohjelmallisesti korvaamalla tyypilliset virheelliset muodot kuten “ät”, “[at]” tai “(0gmail.com” oikeiksi sähköpostiosoitteiksi. OCR:n tuottaman tekstin puhdistus ja virheenkorjaus vähensivät merkittävästi virheiden määrää ja tuottivat yhtenäisen datarakenteen automaattista käsittelyä varten.

OCR-Readerin toimintaa testattiin useilla erilaisilla dokumenteilla, joissa vaihtelivat tekstin tyyppi, tausta ja kirjoitustapa. Testeissä havaittiin, että mustalla taustalla oleva teksti jäi lähes aina tunnistamatta, mikä heikensi OCR-tuloksen luotettavuutta merkittävästi (kuva 29).



Kuva 29. Tulostettu teksti mustalla taustalla.

Vaalealla taustalla oleva tulostettu teksti tunnistettiin OCR-prosessissa huomattavasti tarkemmin ja suurin osa tekstisisällöstä saatiin oikein digitaalisen muotoon (kuva 30).



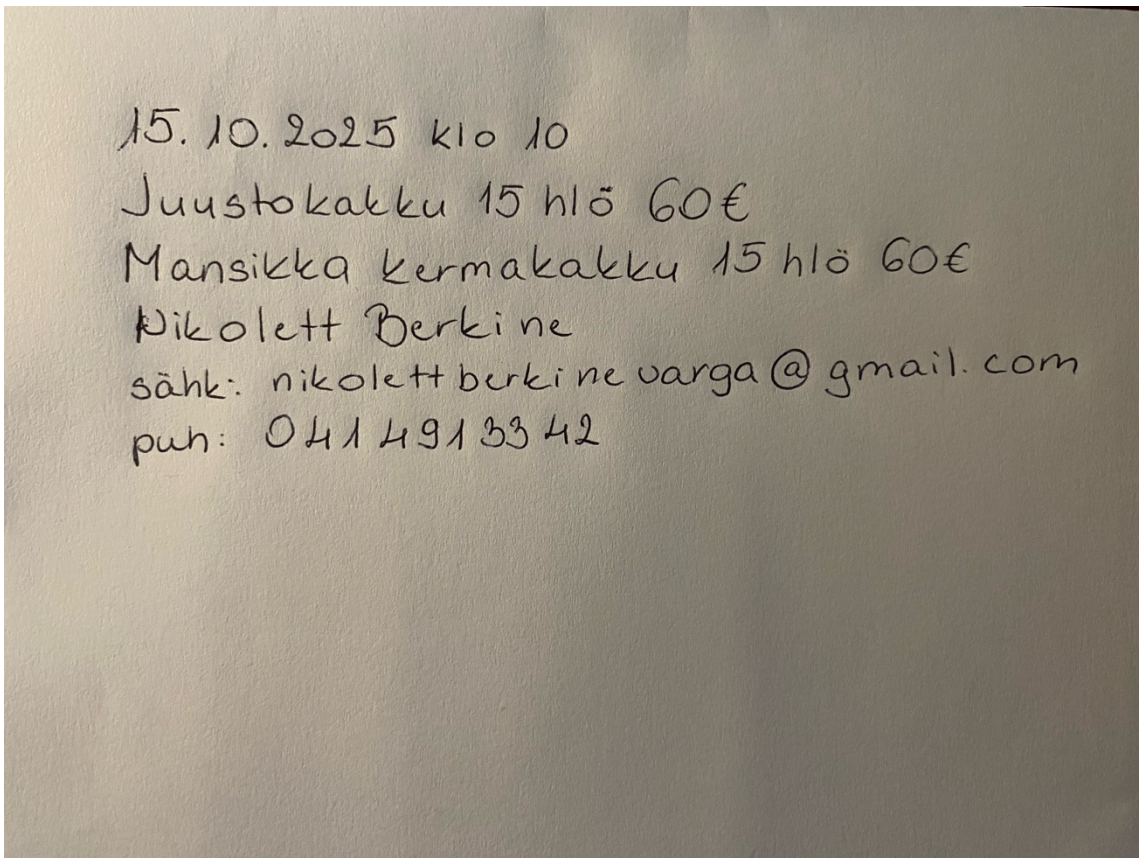
Kuva 30. Tulostettu teksti valkoisella taustalla.

Tekstin korostaminen ja fontin vahventaminen paransivat OCR-tunnistuksen tarkkuutta jonkin verran, mutta erityisesti erikoismerkkien, kuten @-merkin tunnistuksessa esiintyi edelleen virheitä (kuva 31).

17.10.2025 klo 10
Juustokakku 15 hlö 65 €
Mansikka kermakakku 15 hlö 60 €
Nikolett Berkine
Sähk: nikolettberkinevarga@gmail.com
Puh: 0414913342

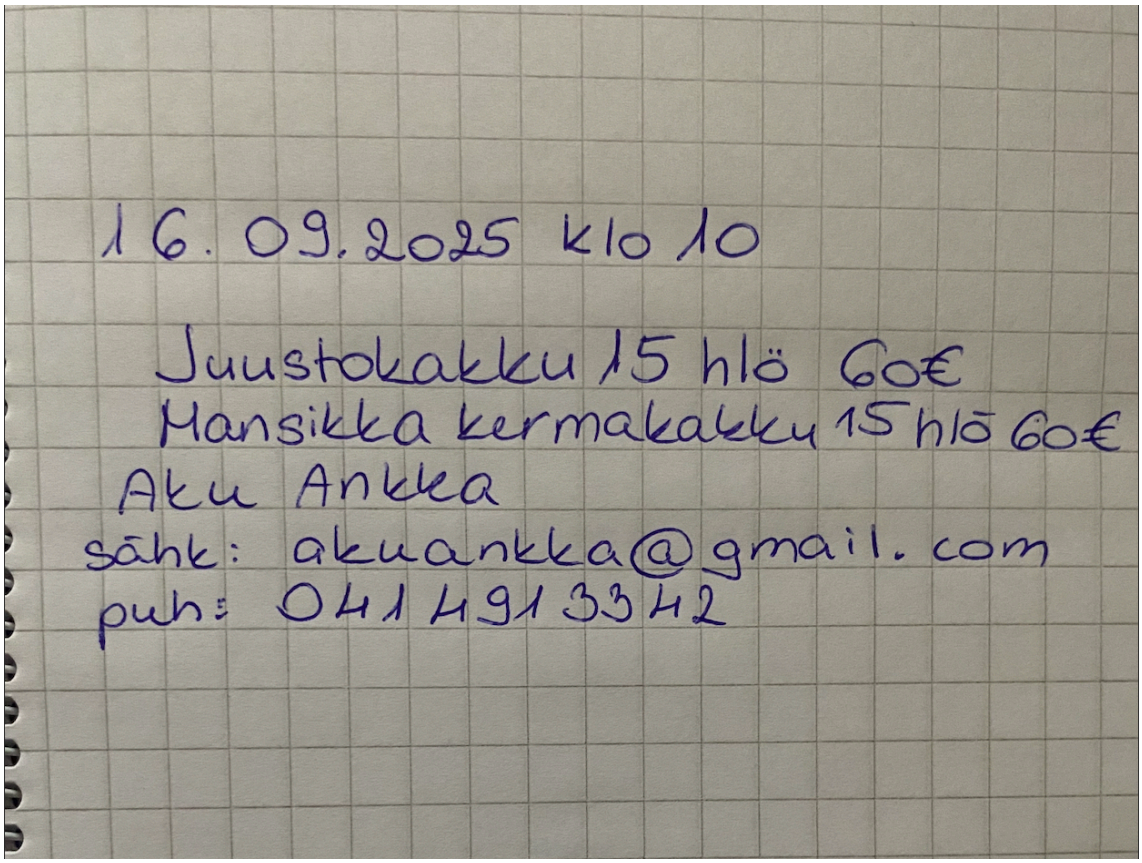
Kuva 31. Tulostettu teksti korostettuna (fontti vahvennettu).

Käsinkirjoitetun tekstin tunnistus tuotti osittaisia tuloksia, mutta virheitä esiintyi erityisesti kirjainten muodossa ja rivien jäsentelyssä, vaikka teksi oli kirjoitettu vaalealle paperille (kuva 32).



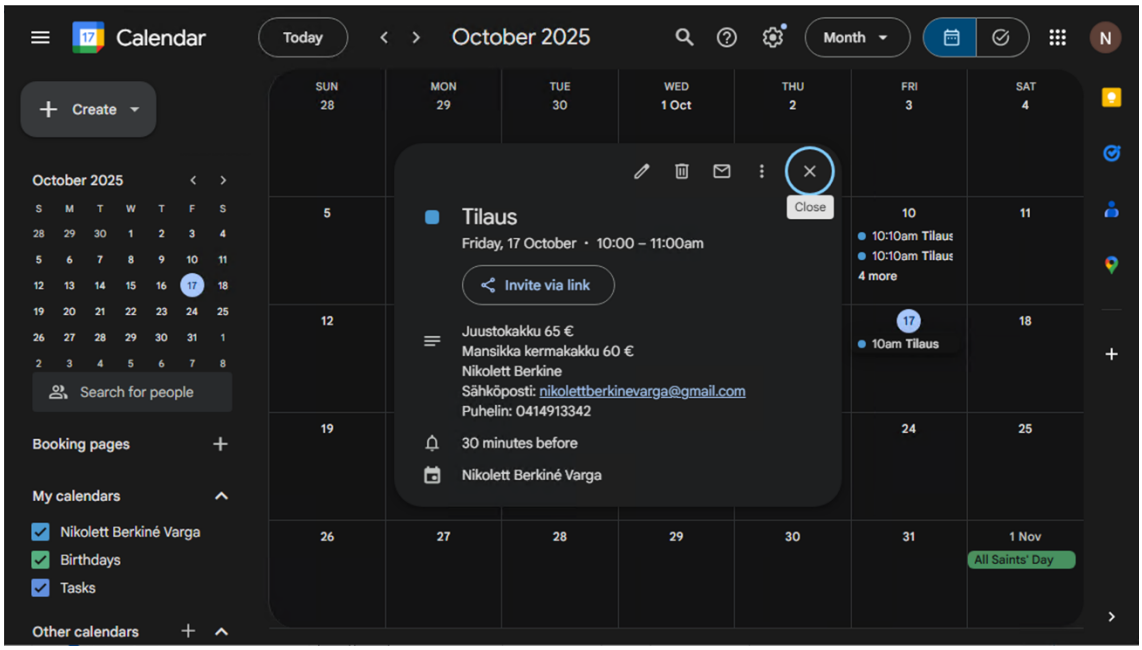
Kuva 32. Käsin kirjoitettu teksti valkoisella paperilla.

Ruutupaperille kirjoitettu käsinkirjoitettu teksti heikensi OCR-tunnistuksen tarkkuutta entisestään, sillä taustaviivat häiritsivät merkkien ja rivirakenteen tunnistamista (kuva 33).



Kuva 33. Käsin kirjoitettu teksti ruutupaperilla.

Tulosten perusteella OCR-Reader toimii luotettavasti painetun tekstin kanssa, mutta käsinkirjoitetun tai visuaalisesti monimutkaisen materiaalin kohdalla tarkkuus laskee merkittävästi. Tesseractin perusmalli ei hallitse käsialan vaihteluita ilman erillistä, koneoppimiseen perustuvaa lisämallia. Lopullisessa automaatiossa OCR-Reader luo onnistuneesta tunnistuksesta kalenterimerkinnän, jossa otsikko on standardoitu muotoon “Tilaus”, ja kuvauksessa näkyvät asiakkaan tiedot, tuotteet ja tilauksen sisältö (kuva 34).

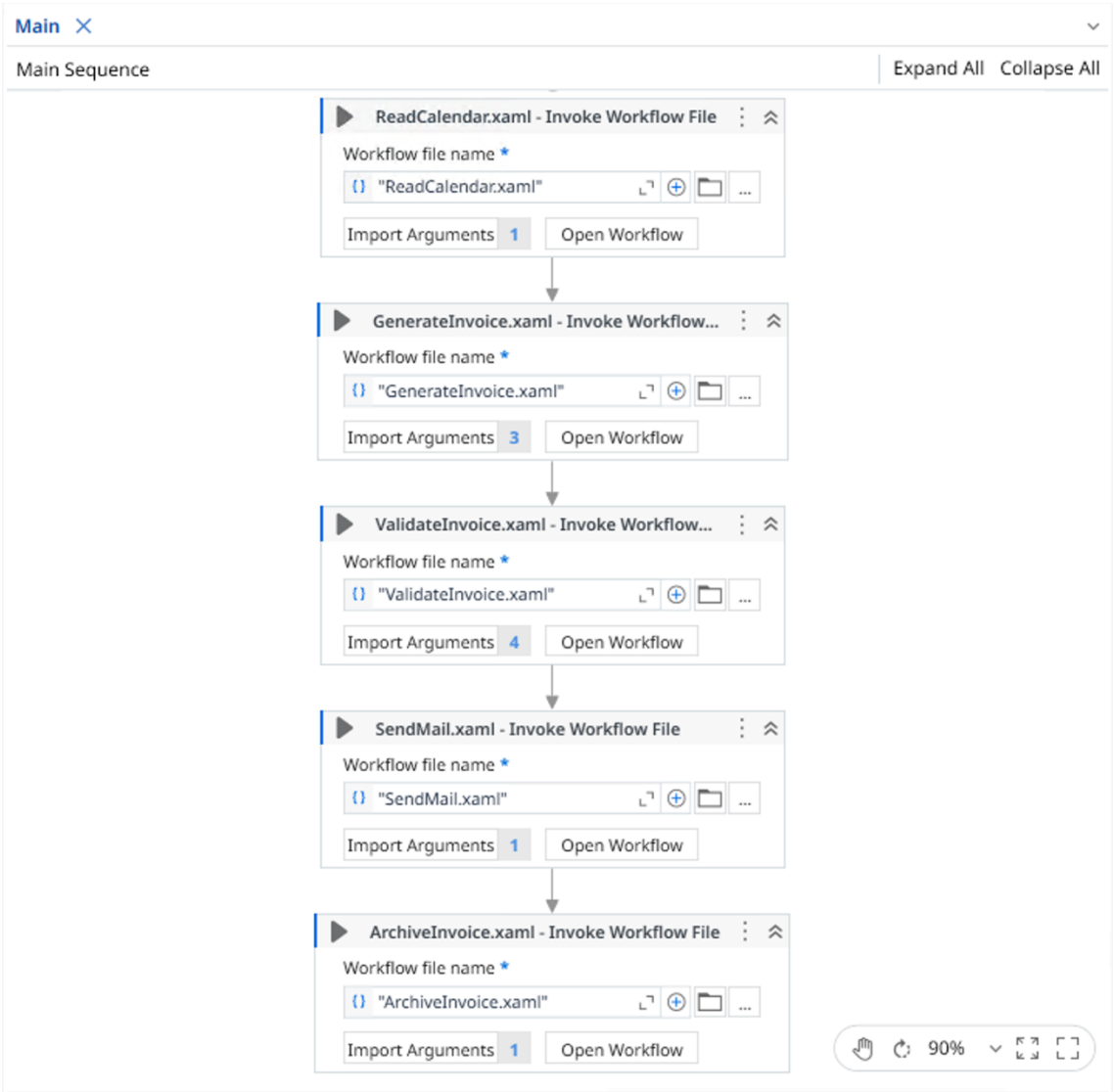


Kuva 34. OCR-Readerin tuottama kalenterimerkintä Google-kalenterissa.

Kokonaisuudessaan OCR-Reader-workflow'n kehitys merkitsi merkittävää askelta kohti täysautomaattista tilausten käsittelyä. Se mahdollisti eri lähteistä tulevan tiedon yhdistämisen yhteen prosessiputkeen, jossa myös manuaaliset, paperimuotoiset tilaukset voidaan siirtää digitaalisiksi ja automatisoituun laskutusprosessiin ilman manuaalista tiedonsyöttöä.

3.9 Workflow: Main

Main-workflow toimii automaation pääprosessina, joka yhdistää kaikki aiemmin kuvatut työnkulut yhdeksi kokonaisuudeksi. Sen tehtävänä on hallita työnkulkujen suoritusjärjestystä, välittää tiedot vaiheesta toiseen ja varmistaa, että prosessi etenee virheettömästi aina tilauksen lukemisesta laskun arkistointiin (kuva 35).



Kuva 35. Main-workflow UiPath Studio -ympäristössä.

Main-sequence rakentuu *Invoke Workflow File* -aktiviteeteista, joiden avulla kukin erillinen työnkulku suoritetaan omassa vaiheessaan. Prosessi etenee seuraavasti:

1. ReadCalendar.xaml

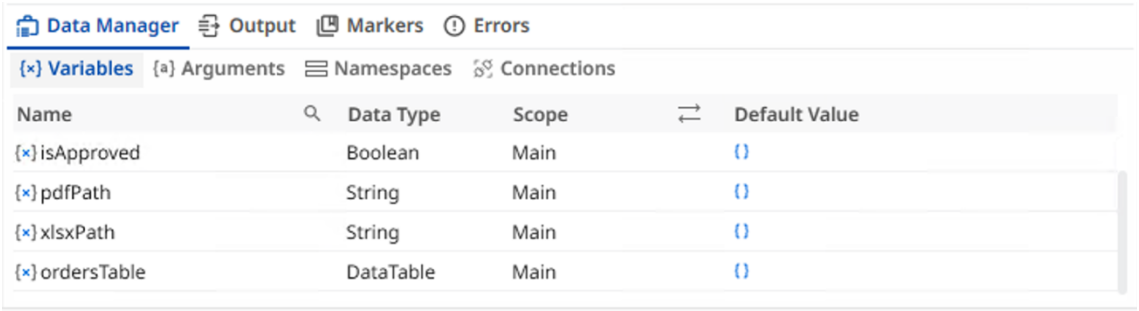
   Hakee tilausmerkinnät Google-kalenterista ja muuntaa ne taulukkomuotoon (DataTable). Tulokset tallennetaan muuttujaan ordersTable, jota käytetään laskun luonnin pohjana.
2. GenerateInvoice.xaml

   Luo laskun ordersTable-muuttujan sisällön perusteella käyttäen valmista Excel-pohjaa. Workflow palauttaa laskun tallennuspolut xlsxPath ja pdfPath, joita käytetään seuraavissa vaiheissa.
3. ValidateInvoice.xaml

Avaa automaattisesti luodun laskun käyttäjälle tarkistettavaksi.
Käyttäjän hyväksyntä tai hylkäys tallennetaan muuttujaan isApproved.
Hyväksytty lasku etenee lähetykseen, hylätty voidaan korjata ja tallentaa uudelleen.

4. SendMail.xaml
   Lähettää hyväksytyn laskun asiakkaalle sähköpostitse.
   Käyttää Gmail-integraatiota ja liittää laskun PDF-muodossa viestiin.
   Lähetyksen onnistuminen kirjataan lokiin.
5. ArchiveInvoice.xaml
   Siirtää lähetetyn laskun automaattisesti Google Drive -arkistoon.
   Jos arkistokansiota ei ole olemassa, workflow luo sen automaattisesti.

Main-workflow’n muuttujat (ordersTable, xlsxPath, pdfPath, isApproved) toimivat tietojen siirtoyhteytenä eri vaiheiden välillä (kuva 36). Tämä rakenne tekee prosessista selkeän, modulaarisen ja helposti ylläpidettävän.



Kuva 36. Main-workflow’n keskeiset muuttujat UiPath Studio -ympäristössä.

Projektin testauksen aikana Main-sequence toimi vakaasti, ja kaikki työnkulut suorittuivat järjestyksessä ilman virheitä. Ainoa varoitus liittyi OCR-Readerin asetukseen ApplyOcrOnPdf = No, joka on tässä tapauksessa tarkoituksellinen: OCR-tunnistus suoritetaan vain alkuperäisille tilaustiedostoille, ei laskuprosessin PDF-tiedostoille.

Kokonaisuutena Main-workflow yhdistää automaation osat yhdeksi yhtenäiseksi laskutusprosessiksi, joka kattaa koko ketjun tilauksen lukemisesta laskun arkistointiin pilvipalvelussa.

4 Testaus

4.1 Testauksen tavoite

Testauksen tavoitteena oli varmistaa, että kehitetty laskutusautomaatioprosessi toimii virheettömästi kaikissa työnkulun vaiheissa — OCR-lukemisesta laskun arkistointiin asti. Testauksessa tarkasteltiin jokaisen workflow'n (OCR-Reader, ReadCalendar, GenerateInvoice, ValidateInvoice, SendMail ja ArchiveInvoice) toimii suunnitellusti sekä erillisenä osana että osana kokonaisuutta (Main-prosessi). Lisäksi testauksella arvioitiin, miten järjestelmä reagoi virheellisiin tai puutteellisiin syötteisiin ja miten ne voidaan tunnistaa ajoissa.

4.2 Testausympäristö

Testaus toteutettiin UiPath Studio -kehitysympäristössä ja UiPath Orchestratorin hallitsemana robotilla, joka suoritti työnkulut automaattisesti. OCR-Reader käytti Tesseract OCR -moottoria (kieliasetuksella "fin"), ja Google-palvelut integroitiin Google Workspace Scope -aktiviteettien avulla. Testidataa varten luotiin viisi erilaista tilaustiedostoa (PDF ja JPG), jotka vastasivat todellisia tilauksia. Näiden avulla simuloitiin sekä paperisia että sähköisiä tilauksia, jotta voitiin varmistaa ratkaisun toimivuus erilaisissa käyttötilanteissa.

4.3 Testauksen toteutus ja tulokset

4.3.1 Testausprosessin kuvaus

Testaus toteutettiin vaiheittain siten, että ensin tarkistettiin yksittäisten workflow'iden toimivuus ja sen jälkeen koko prosessin suoritus Main-workflow'n kautta. Jokainen työnkulku testattiin erikseen, jotta mahdolliset virheet voitiin havaita jo varhaisessa vaiheessa ja varmistaa, että osaprosessit toimivat suunnitellusti ennen niiden yhdistämistä kokonaisratkaisuksi.

Testauksessa arvioitiin erityisesti sitä, millä tavoin workflow'n toimivuus voitiin todentaa, mistä mahdolliset virheet tunnistetaan ja miten järjestelmä reagoi virheelliseen syötteeseen. Tavoitteena oli paitsi osoittaa, että prosessi tuottaa

oikeat tulokset, myös varmistaa, että se käsittelee poikkeustilanteet hallitusti ilman koko prosessin keskeytymistä.

Testaus suoritettiin käytännön tilanteissa hyödyntäen todellisia tilaus- ja laskudokumentteja, jotta prosessin luotettavuutta voitiin arvioida realistisesti. Kunkin workflow'n toiminta ja sen virheiden hallinta kuvataan tarkemmin seuraavissa alaluvuissa.

4.3.2 ReadCalendar workflow

Toimivuuden tunnistaa siitä, että kaikki Google Kalenterin tapahtumat luetaan ja muunnetaan DataTable-rakenteeksi ilman puuttuvia rivejä. Virheellinen toiminta ilmenee tyhjinä tai epätäydellisinä riveinä DataTablessa. Testauksessa virhe havaittiin, kun useampi tuoterivi yhdistyi samalle asiakkaalle. Tämä korjattiin lisäämällä rivikohtainen käsittelylogiikka, joka pilkkoo tilaukset yksittäisiksi riveiksi.
Tuloksena: kaikki tilaukset siirtyivät yhtenäisesti laskutuksen pohjaksi.

Workflow on herkkä tilanteille, joissa Google Kalenterin tapahtumien rakenteessa on poikkeavuuksia. Esimerkiksi puuttuva asiakastieto, väärä päivämäärämuoto tai tyhjä otsikkokenttä voi estää tapahtuman käsittelyn. Lisäksi kalenterin aikavyöhykkeen asetukset voivat vaikuttaa siihen, miten päivämäärät ja kellonajat tulkitaan. Virheelliset tapahtumat ohitetaan, ja ne kirjautuvat lokiin jatkokäsittelyä varten.

4.3.3 GenerateInvoice workflow

Toimiva workflow luo laskun jokaisesta tilauksesta, täydentää laskunumeron automaattisesti ja tallentaa laskun XLSX- ja PDF-muodoissa oikeaan kansioon. Virheellinen toiminta näkyy puuttuvina tiedostoina, tyhjinä soluarvoina tai virheellisinä hintalaskentoina. Testauksessa ilmeni virhe, jossa lasku jäi tallentumatta, jos laskupohja oli auki Excelissä. Tämä ratkaistiin lisäämällä viive ennen tallennusta.
Tuloksena: laskujen automaattinen muodostus ja tallennus toimivat vakaasti.

Workflow voi epäonnistua, jos laskupohjaa ei ole saatavilla, tiedostonimi sisältää virheellisiä merkkejä tai Excel on auki taustalla suorituksen aikana. Lisäksi, jos DataTable sisältää puuttuvia tai vääränmuotoisia tietoja, laskun soluarvot voivat jäädä tyhjiksi tai laskenta epäonnistua. Virhetilanteissa workflow pysäyttää prosessin ja ilmoittaa virheestä lokiin.

4.3.4 ValidateInvoice workflow

Workflow toimii oikein, kun lasku avautuu PDF-muodossa, käyttäjälle esitetään hyväksyntäkysymys ja hyväksynnän jälkeen lasku etenee lähetykseen. Virheellisen toiminnan huomaa siitä, että PDF ei avaudu tai ohjelma ei reagoi käyttäjän valintaan. Testauksessa todettiin, että käyttäjän hylkäämä lasku avautui Excelissä muokattavaksi ja muodostui oikein uudelleen tallennettaessa. Tuloksena: laadunvarmistus toimi suunnitellusti, ja käyttäjä pystyi estämään virheellisten laskujen lähetyksen.

Prosessin voi rikkoa tilanne, jossa PDF-tiedosto ei avaudu oikein tai käyttäjän hyväksyntäikkuna ei saa syötettä. Jos laskun tiedostopolku on virheellinen tai PDF-tiedosto on lukittu, ohjelma ei voi jatkaa validointia. Tällöin virhe tallentuu lokiin ja lasku siirtyy tarkistettavaksi manuaalisesti.

4.3.5 SendMail workflow

Toimiva workflow lähettää hyväksytyn laskun automaattisesti asiakkaan Gmail-osoitteeseen ja liittää siihen PDF-laskun. Virheellisen toiminnan tunnistaa virheilmoituksesta tai puuttuvasta viestistä. Testauksessa havaittiin, että virhe syntyi, jos asiakkaan sähköpostiosoite puuttui tai sisälsi virheellisen muodon. Workflow kuitenkin tunnistaa tilanteen ja siirtää laskun *ErrorEmails*-kansioon, mikä estää väärän viestin lähetyksen.
Tuloksena: sähköpostin lähetys toimi kaikissa testitapauksissa, joissa syöte oli oikea.

Sähköpostin lähetys voi epäonnistua, jos asiakkaan osoite on virheellinen, tyhjä tai jos Gmailin autentikointi ei ole voimassa. Myös liian suurikokoiset liitteet voivat estää viestin lähettämisen.

4.3.6 ArchiveInvoice workflow

Workflow toimii oikein, kun lähetetyt laskut siirtyvät automaattisesti Google Driveen tai paikalliseen “Laskut”-kansioon, ja lokitiedostoon kirjautuu onnistunut tallennus. Virhetilanteen tunnistaa siitä, että tiedostopolku puuttuu tai Drive-yhteys ei muodostu. Testissä todettiin, että tällaisissa tilanteissa ohjelma keskeyttää ajon hallitusti ja kirjaa virheen lokiin ilman tietojen menetystä.
Tuloksena: arkistointi toimi vakaasti, ja kansiorakenne säilyi hallittuna.

Arkistointivaihe voi epäonnistua, jos määritetty arkistokansio puuttuu, tiedostonimi sisältää kiellettyjä merkkejä tai yhteys Google Driveen katkeaa. Näissä tapauksissa workflow kirjaa virheen lokitiedostoon ja säilyttää laskun väliaikaisessa “PendingArchive”-kansiossa, kunnes yhteys on palautettu.

4.3.7 OCR-Reader workflow

Toimivuuden tunnistaa siitä, että painetut tilaukset luetaan onnistuneesti ja niistä syntyy automaattisesti Google Kalenteriin uusi tapahtuma, joka sisältää asiakkaan tiedot ja tilatut tuotteet. Jos workflow ei toimi oikein, kalenterimerkintää ei synny lainkaan tai se sisältää virheellisiä tietoja, kuten väärän päivämäärän, puuttuvan sähköpostiosoitteen tai virheellisen hinnan. Tyypillinen virhe syntyi käsinkirjoitetusta tai heikkolaatuisesta kuvasta, jolloin OCR-tunnistus ei löytänyt kaikkia merkkejä tai korvasi niitä symboleilla. Workflow kuitenkin tunnistaa virheen ja siirtää kyseisen tiedoston *Error*-kansioon manuaalista tarkistusta varten, jotta virheellinen data ei etene laskutusprosessiin.

Workflow toimii luotettavasti selkeän ja painetun tekstin kanssa, mutta sen toimintaa voivat heikentää useat visuaaliset ja tekniset tekijät. Heikkolaatuiset tai tummasävyiset kuvat, epätasainen valaistus, taustaviivat ja käsinkirjoitettu teksti voivat aiheuttaa virheitä OCR-tunnistuksessa. Lisäksi kuvien matala resoluutio, vino skannauskulma tai tekstin sijoittelu useaan sarakkeeseen voi johtaa siihen, että osa tiedoista jää tunnistamatta tai vääristyy. Näissä tilanteissa workflow saattaa tuoda virheellistä tai puutteellista tietoa, kuten väärän hinnan tai asiakkaan nimen, jolloin tieto ei etene automaattiseen laskutusprosessiin vaan jää tarkistettavaksi ennen seuraavaa vaihetta.

4.4 Kokonaisprosessin testaus

Kun kaikki työnkulut yhdistettiin Main-workflow'n kautta, prosessi suoritti koko laskutusautomaation alusta loppuun ilman manuaalista väliintuloa, lukuun ottamatta laskun hyväksyntää. Koko prosessin läpimenoaika yhdelle tilaukselle oli keskimäärin 45 sekuntia, ja kaikki laskut muodostuivat, lähetettiin ja arkistoitiin oikein. Prosessin toimivuus varmistettiin vertaamalla tuloksia manuaalisesti laadittuihin laskuihin. Erot olivat marginaalisia ja liittyivät vain käsinkirjoitetun OCR-tuloksen epätarkkuuksiin, kuten taulukossa 1 on esitetty.

Workflow	Tulos	Huomiot
OCR-Reader	Toimii painetulla tekstillä	Käsikirjoitus ei toimii
ReadCalendar	Toimii luotettavasti	Rivirakenne korjattu
GenerateInvoice	Automaattinen ja vakaa	PDF-muunnon onnistui
ValidateInvoice	Käyttäjäkontrolli toimii	Ei virheitä
SendMail	Viestit lähtivät oikein	Liitteet löytyvät
ArchiveInvoice	Arkistointi onnistui	Virheenkäsittely parani

Taulukko 1. Testattujen workflow-prosessien tulokset ja huomiot.

4.5 Testauksen johtopäätökset

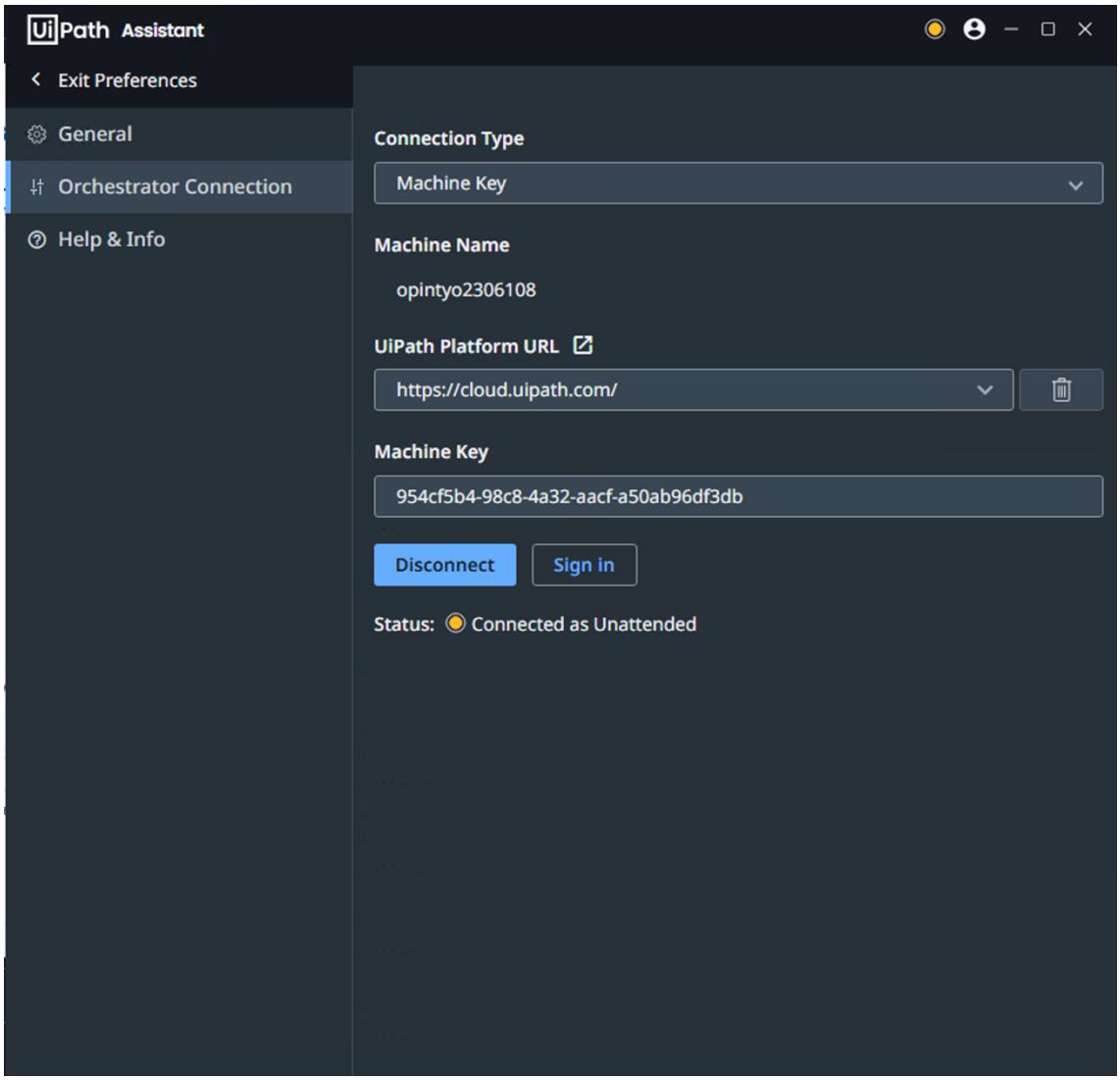
Testaustulokset osoittavat, että kehitetty UiPath-pohjainen laskutusautomaatioprosessi toimii luotettavasti ja täyttää sille asetetut tavoitteet. Prosessi on modulaarinen, virheensietoinen ja helposti ylläpidettävä. Jokaisessa vaiheessa havaittiin selkeät indikaattorit toimivalle ja virheelliselle suoritukselle, mikä helpottaa jatkokehitystä ja mahdollisten ongelmien tunnistamista. OCR-komponentin osalta haasteet liittyvät edelleen käsinkirjoitettujen tilausten tunnistukseen, mutta painetun materiaalin osalta tulokset olivat täysin toimivia.

5 Automatisoidun ajon käyttöönotto Orchestratorissa

UiPath Orchestrator on pilvipohjainen hallintaympäristö, jonka avulla automaatioprosesseja voidaan hallita, ajastaa ja suorittaa keskitetysti. Orchestrator mahdollistaa automaatioiden ajamisen ilman käyttäjän manuaalista käynnistystä, mikä tekee siitä täysin itsenäisen (“unattended”) ratkaisun. Tässä luvussa kuvataan, miten Orchestrator liitettiin opinnäytetyössä kehitettyyn laskutusautomaation ratkaisuun ja miten sen käyttöönotto mahdollistaa prosessin automaattisen suorittamisen ilman käyttäjän väliintuloa. Käytännössä Orchestrator toimii palvelimena, joka hallinnoi robottien ajamista ja valvontaa, kun taas UiPath Assistant toimii käyttäjän työasemalla paikallisena ohjauspaneelina.

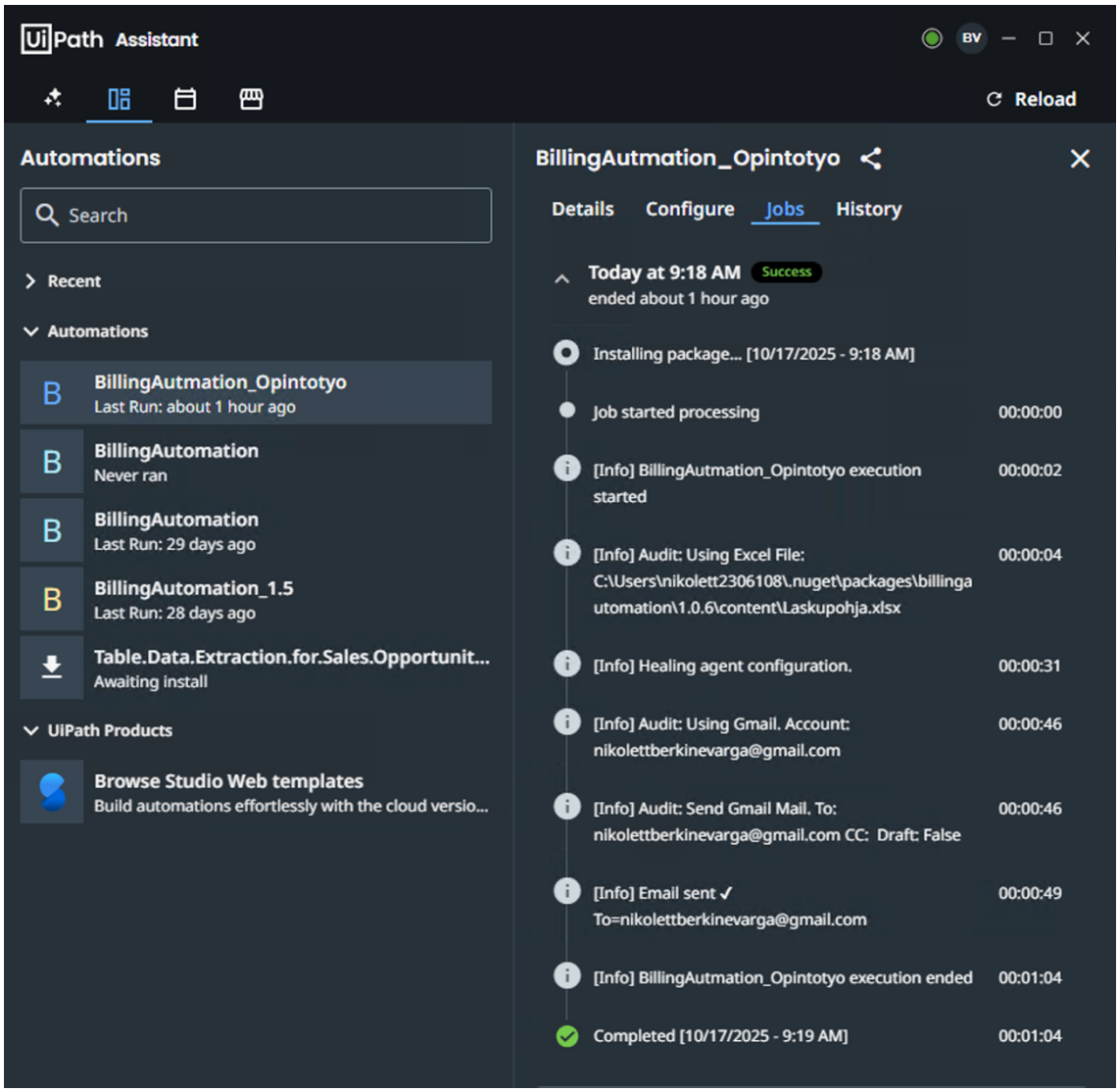
Alkuperäisessä suunnitelmassa Orchestratoria ei ollut tarkoitus ottaa osaksi toteutusta, vaan automaation oli tarkoitus toimia paikallisesti käyttäjän käynnistämänä. Kehitystyön aikana havaittiin kuitenkin, että Orchestratorin käyttöönotto parantaa prosessin hallittavuutta ja mahdollistaa sen ajastamisen täysin automaattisesti. Näin prosessi voi toimia ilman manuaalista ohjausta, mikä tekee ratkaisusta aidosti unattended-tyyppisen ja vähentää käyttäjän kuormitusta.

Unattended-ajon avulla automaatio voidaan nyt suorittaa pilvipohjaisesti ja ajastetusti ilman käyttäjän manuaalista käynnistystä. Yhteys UiPath Studio -kehitysympäristön, Orchestratorin ja paikallisen käyttöliittymän eli UiPath Assistantin välillä määritettiin Machine Key -tunnuksen avulla. Tämä yhteys mahdollistaa sen, että robotti (bot) voidaan ajaa etänä Orchestratorista käsin ilman, että käyttäjän tarvitsee avata Studioa. Yhteyden tila “Connected as Unattended” vahvisti, että automaatio on liitetty oikein Orchestratoriin ja valmis etäajoon (kuva 37). Machine Key toimii tunnisteena, jolla robotti rekisteröidään Orchestratoriin ja liitetään turvallisesti oikeaan työympäristöön. Tämä varmistaa, että yhteys toimii vain valtuutetussa ympäristössä.



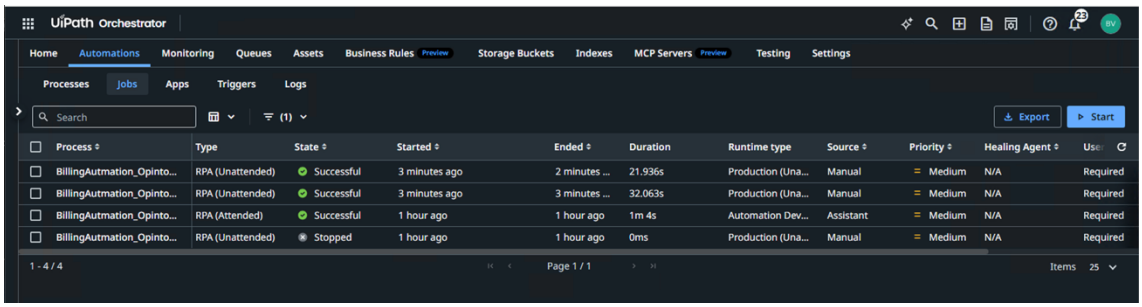
Kuva 37. UiPath Assistantin Orchestrator Connection – yhteys muodostettu onnistuneesti (Connected as Unattended).

UiPath Assistant toimii käyttöliittymänä, jonka kautta käyttäjä voi seurata, käynnistää ja hallita automaatioita paikallisesti. Opinnäytetyössä Assistantia käytettiin testauksen aikana erityisesti manuaalisissa ajoissa, joilla varmistettiin, että prosessi toimii oikein ennen Orchestratorin kautta tapahtuvaa unattended-suoritusta. Manuaalisessa ajossa käyttäjä pystyy seuraamaan työnkulun etenemistä reaaliaikaisesti ja näkemään sen tilan suoraan Assistantin näkymässä (kuva 38).



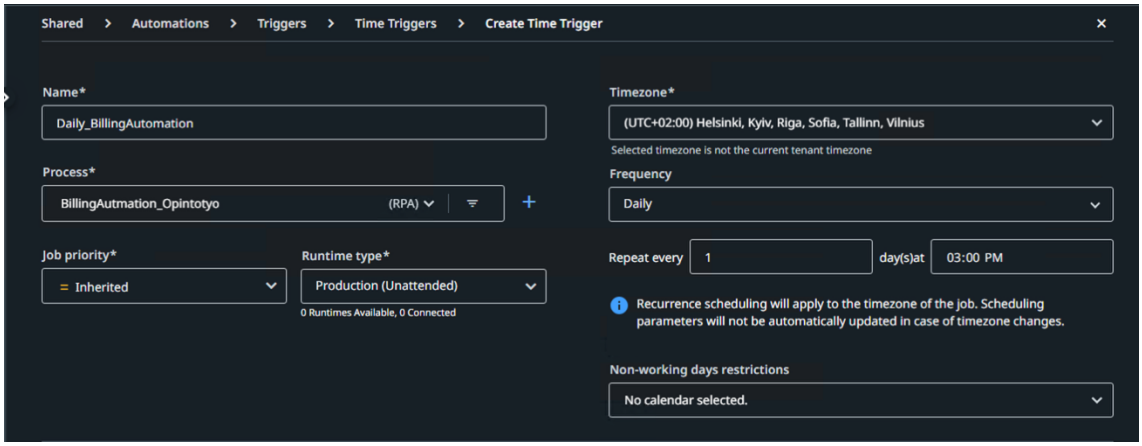
Kuva 38. UiPath Assistantin näkymä onnistuneesta manuaalisesta ajosta.

Orchestratorin *Jobs*-näkymässä näkyi useita onnistuneita ajokertoja sekä attended- että unattended-tilassa (kuva 39). Keskimääräinen suoritus kesti 20–30 sekuntia per prosessi, ja kaikki työnkulut (ReadCalendar, GenerateInvoice, ValidateInvoice, SendMail ja ArchiveInvoice) suorittuivat virheettömästi. Tulosten perusteella voidaan todeta, että automaatio on valmis tuotantokäyttöön ja kykenee toimimaan itsenäisesti ilman käyttäjän ohjausta.



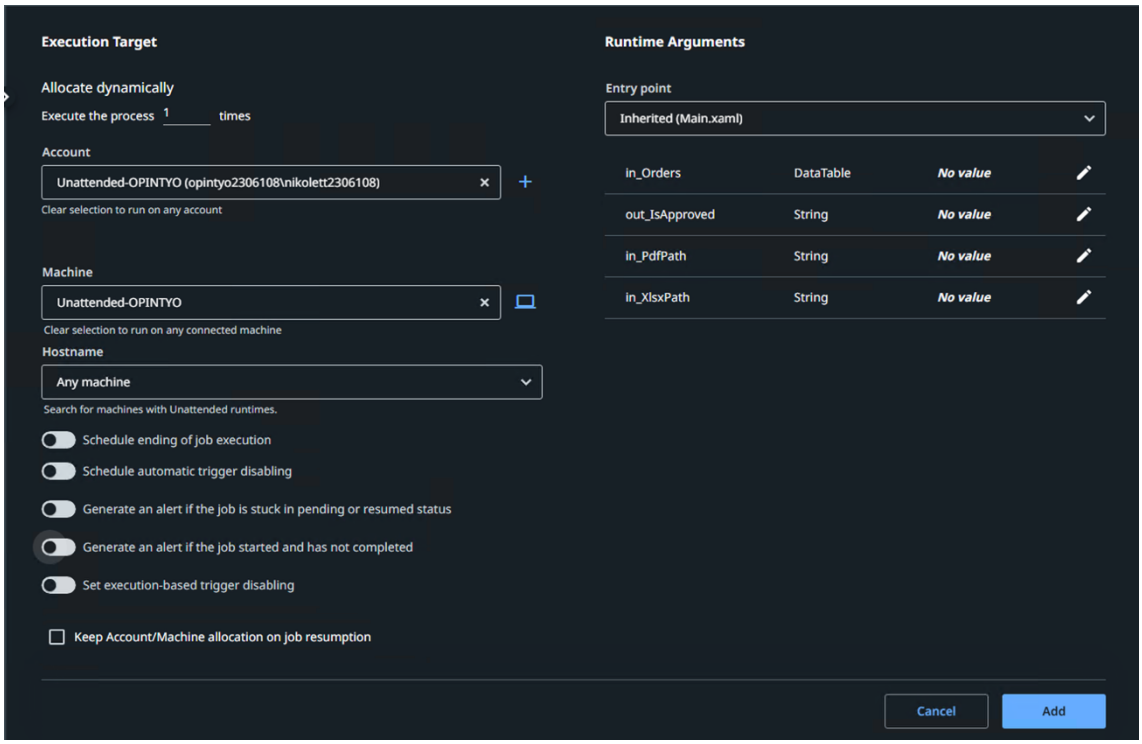
Kuva 39. Orchestratorin Jobs-näkymä, jossa näkyy onnistuneita Unattended- ja Attended-ajoja (BillingAutomation_Opintotyo -prosessi).

Jatkotestauksessa Orchestratoriin luotiin myös ajastus (*Time Trigger*), jonka avulla laskutusprosessi voidaan käynnistää automaattisesti esimerkiksi kerran päivässä (kuva 40). Ajastus mahdollistaa täysin autonomisen toiminnan, jossa järjestelmä suorittaa laskutuksen säännöllisesti ilman manuaalista valvontaa.



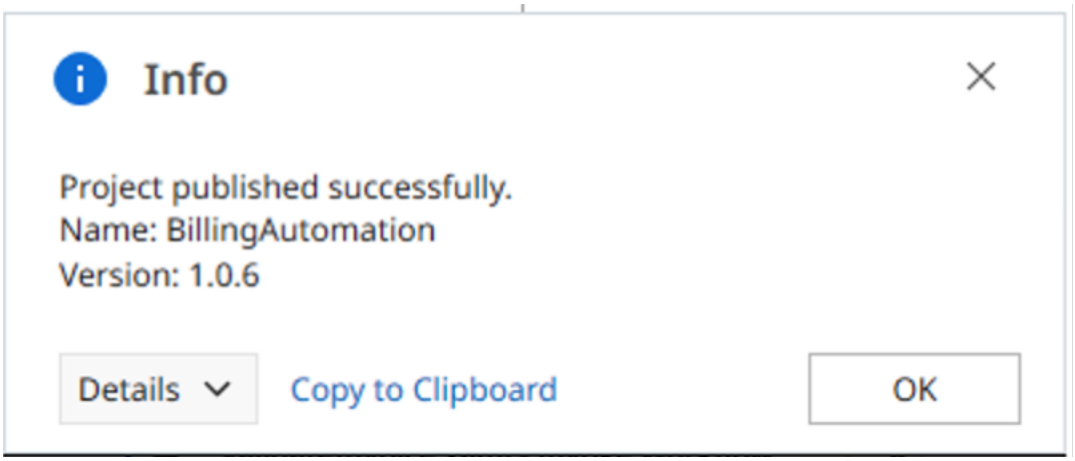
Kuva 40. Triggerin määritys päivittäistä ajastusta varten.

Kun ajastus on aktiivinen, Orchestrator käynnistää automaatioprosessin automaattisesti määritetyn aikataulun mukaisesti. Prosessin suorittamista ja sen tilaa voidaan seurata Orchestratorin Jobs-näkymässä, josta näkyvät ajon tila, kesto sekä mahdolliset virheilmoitukset (kuva 41).



Kuva 41. Automaatioprosessin suoritus ja ajon asetukset UiPath Orchestratorissa.

Lisäksi prosessi julkaistiin Orchestratorin *Packages*-näkymään, jossa se on valmiina ajettavaksi ja hallittavaksi etänä (kuva 42).



Kuva 42. BillingAutomation-prosessi julkaistu Orchestratoriin onnistuneesti.

Tulosten perusteella voidaan todeta, että automaatio on valmis tuotantokäyttöön ja kykenee toimimaan itsenäisesti ilman käyttäjän ohjausta. Orchestratorin käyttöönotto toi prosessiin merkittävän kehitysaskeleen: automaatiota voidaan nyt hallita keskitetysti, ajastaa toistuvasti ja valvoa suorituksia ilman paikallista käyttöä. On kuitenkin huomioitava, että opinnäytetyössä käytetty Orchestrator-ympäristö perustuu opiskelijalisenssiin, jota ei voi sellaisenaan hyödyntää kaupallisessa tuotantokäytössä. Mikäli ratkaisua halutaan kehittää eteenpäin yritysympäristössä, tarvitaan erilliset kaupalliset lisenssit sekä Orchestrator- että robottikäyttöön. Tämä rajoitus ei vaikuttanut opinnäytetyön testaukseen, mutta se on tärkeää huomioida jatkokehityksen ja mahdollisen tuotantokäytön kannalta.

UiPath Assistantin ja Orchestratorin yhdistelmä mahdollistaa sekä paikallisen testauksen että keskitetyn, automatisoidun ajon hallinnan, mikä tekee ratkaisusta joustavan ja helposti ylläpidettävän kokonaisuuden.

Jatkokehityksessä Orchestratorin toiminnallisuuksia voitaisiin hyödyntää laajemmin ajastusten, resurssien hallinnan ja useamman robotin yhtäaikaisen ajon tukena. Näin laskutusautomaatiosta olisi mahdollista kehittää täysin itsenäinen ja skaalautuva ratkaisu, joka palvelee yritysympäristöjen tarpeita.

6 Jatkokehitysideat

Vaikka kehitetty laskutusautomaatioprosessi toimii kokonaisuutena luotettavasti ja täyttää sille asetetut tavoitteet, työn aikana nousi esiin useita jatkokehitysmahdollisuuksia, joiden avulla ratkaisua voitaisiin laajentaa ja parantaa entisestään.

6.1 OCR-tunnistuksen parantaminen tekoälypohjaisella mallilla

Nykyinen ratkaisu hyödyntää Tesseract OCR -moottoria, joka toimii hyvin painetun tekstin kanssa, mutta ei tunnista käsinkirjoitettua tekstiä luotettavasti. Jatkossa OCR-komponenttia voisi kehittää ottamalla käyttöön tekoälypohjaisen käsinkirjoituksen tunnistusmallin, kuten Google Cloud Vision OCR:n tai Azure Form Recognizerin. Tämä mahdollistaisi myös monikielisten tilausten ja vaihtelevien dokumenttipohjien käsittelyn automaattisesti.

6.2 Datan tallennus ja analytiikka

Laskutuksen yhteydessä muodostuvaa dataa voisi jatkossa hyödyntää laajemmin liiketoiminnan kehittämisessä. Esimerkiksi laskutus- ja tilausdata voitaisiin siirtää automaattisesti tietokantaan (SQL tai Google Sheets) raportointia ja analytiikkaa varten.
Tämä mahdollistaisi reaaliaikaisen seurannan, kuten myyntimäärien, asiakaskohtaisten tilausten ja sesonkien analysoinnin.

6.3 Käyttöliittymä yrityskäyttäjille

Ratkaisu toimii tällä hetkellä UiPath Studio -ympäristössä, mutta sen voisi tulevaisuudessa liittää yksinkertaiseen web-pohjaiseen käyttöliittymään. Käyttöliittymän avulla käyttäjä voisi tarkistaa laskuja, käynnistää automaation ja seurata ajotilastoja ilman UiPath-osaamista. Tämä parantaisi käytettävyyttä ja tekisi ratkaisusta helpommin hyödynnettävän myös pienyrityksille.

6.4 Täysin automatisoitu laskutus ja maksuseuranta

Unattended-ajon onnistunut käyttöönotto avaa mahdollisuuden laajentaa prosessia edelleen. Seuraava vaihe voisi olla laskujen automaattinen lähetys, maksujen seuranta ja muistutusten hallinta ilman manuaalista tarkistusta. Integroimalla järjestelmään esimerkiksi VismaNet- tai Procountor-rajapinnan, laskut ja maksutapahtumat voitaisiin käsitellä täysin automaattisesti.

6.5 Monikäyttöinen prosessipohja muille toimialoille

Kehitetty malli voitaisiin sovittaa myös muille palvelualoille, kuten siivous-, kauneus- tai huoltoalalle, joissa laskutus perustuu kalenterimerkintöihin tai varauksiin.
Jatkokehityksessä prosessista voidaan luoda helposti muokattava “template”, jota yritykset voivat hyödyntää omissa toistuvissa prosesseissaan ilman laajaa ohjelmointityötä.

Jatkokehityksen kannalta keskeisintä on OCR-tarkkuuden parantaminen, datan hyödyntäminen ja prosessin laajentaminen täysin autonomiseksi laskutus- ja maksujärjestelmäksi. Näiden kehitysaskelten myötä ratkaisu voisi kasvaa prototyypistä täysimittaiseksi kaupalliseksi sovellukseksi pk-yritysten tarpeisiin.

7 Pohdinta

Tämän opinnäytetyön tavoitteena oli kehittää automatisoitu laskutusratkaisu elintarvikealan pienille ja keskisuurille yrityksille. Työn lähtökohtana oli vähentää manuaalista työtä ja virheitä laskutusprosessissa hyödyntämällä ohjelmistorobotiikkaa (UiPath) ja optista tekstintunnistusta (OCR). Projekti toteutettiin käytännön kehittämistyönä, ja sen aikana rakennettiin useista työnkuluista koostuva automaatio, joka hallitsee koko prosessin tilauksesta laskun arkistointiin.

Projektin lopputulos vastasi hyvin asetettuja tavoitteita. Automaatioprosessi toimi testauksen perusteella vakaasti, ja se onnistui hoitamaan laskujen luonnin, validoinnin, lähetyksen ja arkistoinnin täysin automaattisesti. OCR-Reader

mahdollisti myös paperimuotoisten tilausten digitoinnin ja siirtämisen samaan prosessiputkeen. Lopulta automaatio saatiin toimimaan Unattended-ajona Orchestratorissa, mikä teki siitä täysin itsenäisen ja tuotantovalmiin järjestelmän.

Työprosessi oli iteratiivinen ja sisälsi useita kokeilu- ja virheenvaiheita, joiden kautta ratkaisu kehittyi merkittävästi. Erityisen hyödylliseksi osoittautui vaiheittainen testaus, jossa jokainen workflow testattiin erikseen ennen niiden yhdistämistä pääprosessiksi. Tämä vähensi virheiden määrää ja teki lopputuloksesta luotettavamman. Projekti osoitti myös sen, että UiPath tarjoaa erittäin monipuoliset mahdollisuudet pk-yritysten arjen prosessien automatisointiin ilman raskaita IT-investointeja.

Työn aikana opittiin paljon ohjelmistorobotiikan, OCR-teknologian ja Orchestrator-ympäristön käytöstä. Erityisesti Unattended-ajon käyttöönotto tarjosi konkreettisen kokemuksen siitä, miten automaatio voidaan siirtää kehitysympäristöstä tuotantoon ja ajastaa toimimaan itsenäisesti.

OCR:n osalta opittiin, että tunnistuksen laatu riippuu voimakkaasti syötteen visuaalisesta laadusta, mutta ohjelmallisilla korjauksilla (tekstin siistintä, regex-haku, erikoismerkkien normalisointi) tuloksia voidaan parantaa huomattavasti.

Kokonaisuutena projekti onnistui hyvin ja täytti sille asetetut tavoitteet. Ratkaisu on käytännössä toimiva, skaalautuva ja sovellettavissa muillekin toimialoille, joissa laskutus perustuu kalenterimerkintöihin tai varauksiin. Työn tekeminen vahvisti tekijän teknistä osaamista sekä ymmärrystä automaation suunnittelusta, testaamisesta ja käyttöönotosta. Projektin kautta syntyi konkreettinen esimerkki siitä, miten digitaaliset työkalut voivat tehostaa elintarvikealan yritysten toimintaa ja vähentää manuaalista kuormitusta. Jatkossa kehitystyötä voisi laajentaa erityisesti OCR:n tarkkuuden parantamiseen, datan hyödyntämiseen raportoinnissa sekä täysin automatisoidun laskutus- ja maksuprosessin rakentamiseen. Unattended-ajon onnistuminen luo vahvan pohjan näiden jatkokehitysaskelten toteuttamiselle tulevaisuudessa.

Lähteet

Gill, R., Hooda, S., Srivastava, D. & Harnal, S. 2025. Handbook of Intelligent Automation Systems Using Computer Vision and Artificial Intelligence. Wiley-Scrivener.

Google Cloud 2024. Optical Character Recognition (OCR) overview. https://cloud.google.com/vision/docs/ocr. 22.10.2025

Hyrylä, ML. 2023. Uudistuva elintervikeala. Työ- ja elinkeinoministeriön julkaisuja 2023:5. Helsinki Työ- ja elinkeinoministeriö. https://urn.fi/URN:ISBN:978-952-327-629-1. 11.11.2025

Lacity, M. & Willcocks, L. 2016. A New Approach to Automating Servies. MIT Sloan Management Review.

Lahti, S. & Salminen, T. 2014. Digitaalinen taloushallinto. Helsinki: Alma Talent. Alma Insights -tietokanta. 12.11.2025

Microsoft Azure 2024. Azure AI Vision – Read API documentation. https://learn.microsoft.com/en-us/azure/ai-services/computer-vision/overview-ocr 22.10.2025.

Tripathi, A.M. 2018. Learning Robotic Process Automation. Birmingham: Packt Publishing.

UiPath Documentation 2025a. About Robots. https://docs.uipath.com/orchestrator/automation-cloud/latest/user-guide/about-robots 25.10.2025.

UiPath Documentation 2025b. Digitize Document Acitivty. https://docs.uipath.com/activities/other/latest/document-understanding/digitize-document 26.10.2025.

UiPath Documentation 2025c. Excel Process Scope. https://docs.uipath.com/activities/other/latest/productivity/excel-process-scope-x 26.10.2025.

UiPath Documentation 2025d. Google Workspace HTTP Request. https://docs.uipath.com/activities/other/latest/productivity/google-workspace-general-http-request-connections 26.10.2025.

UiPath Documentation 2025e. Google Workspace Scope. https://docs.uipath.com/activities/other/latest/productivity/gsuite-application-scope 26.10.2025.

UiPath Documentation 2025f. Invoke Code Activity. https://docs.uipath.com/activities/other/latest/workflow/invoke-code 26.10.2025.

UiPath Documentation 2025g. Invoke Workflow File Activity. https://docs.uipath.com/activities/other/latest/workflow/invoke-workflow-file 26.10.2025.

UiPath Documentation 2025h. What is Robotic Process Automation. https://www.uipath.com/rpa/robotic-process-automation 22.10.2025.

UiPath Documentation 2025i. Orhestrator user guide. https://docs.uipath.com/orchestrator/automation-cloud/latest/user-guide/about-triggers 07.11.2025.

UiPath Documentation 2025j. Assistant user guide. https://docs.uipath.com/assistant/standalone/2024.10/user-guide/about-uipath-assistant 07.11.2025

UiPath Documentation 2025k. Introduction to Orchestrator. https://docs.uipath.com/orchestrator/automation-cloud/latest/user-guide/introduction 25.10.2025.

UiPath Documentation 2025l. Studio User Guide. https://docs.uipath.com/studio/standalone/2022.10/user-guide/introduction 22.10.2025.

UiPath Documentation 2025m. Use Gmail. https://docs.uipath.com/activities/other/latest/productivity/gmail-application-card 26.10.2025.

UiPath Documentation 2025n. Document Understanding Modern Projects User Guide. https://docs.uipath.com/document-understanding/automation-cloud/latest/user-guide/moving-document-understanding-modern-projects-between-tenants-or-organizations 07.11.2025

VTT Technical Research Centre of Finland Ltd 2020. Digitalisation in Finnish manufacturing SME companies. Espoo: VTT. https://cris.vtt.fi/ws/portalfiles/portal/42494175/Digitalisation_in_Finnish_manufacturing_SMEs_final.pdf 22.10.2025