Mete Güneysel

Myyntilaskujen vienti toiminnanohjausjärjestelmästä verkkolaskuoperaattorille

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikka

Insinöörityö

17.7.2022

Tiivistelmä

Tekijä:	Mete Güneysel
Otsikko:	Myyntilaskujen vienti toiminnanohjausjärjestelmästä verkkolaskuoperaattorille
Sivumäärä:	45 sivua
Aika:	17.7.2022
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Tieto- ja viestintätekniikka
Ammatillinen pääaine:	Monimuotokoulutus
Ohjaajat:	Yliopettaja Janne Salonen

Tämän insinöörityön tilasi Systencess Oy, joka tarvitsi integraation toiminnanohjausjärjestelmä Microsoft Dynamics 365 Business Centralin sekä verkkolaskuoperaattori Maventan välille, myyntilaskujen laskutusta varten.

Integraatiossa keskityttiin myyntilaskujen vientiin EU:ssa yleistyvän PEPPOL-verkoston kautta. Insinöörityössä pyrittiin luomaan automaatio, joka hoitaisi myyntilaskujen laskutukseen liittyvän työnkulun itsenäisesti kerran päivässä. Automaation toteutuksessa käytetyt integraatiotyökalut oli päätetty etukäteen Systencess Oy:n toimesta.

Toteutuksessa ohjelmoitiin Business Centraliin AL-ohjelmointikielellä tietokantakysely, jonka avulla Microsoftin Power Automatessa luotu automaatio pystyi hakemaan ERP-järjestelmästä tarvittavat myyntilaskujen laskutiedot. Automaatio lähetti laskutiedot Microsoft Azuren pilvipalveluun ohjelmoidulle Java-ohjelmalle, joka rakensi laskutiedoista XSLT-muunnoksen tuloksena standardinmukaisen Peppol BIS 3.0 verkkolaskun.

Työn lopputuloksena saatiin luotua toimiva automaatio, joka haki ajastetusti toiminnanohjausjärjestelmään syntyneet myyntilaskut ja toimitti ne verkkolaskuoperaattorille. Tämän integraation avulla Systencess Oy pystyy tulevaisuudessa tarjoamaan nykyisille sekä uusille asiakkaailleen laskutusmahdollisuuden PEPPOL-verkoston kautta, tarjoamassaan toiminnanohjausjärjestelmä Business Centralissa.

Avainsanat: Microsoft Business Central, Power Automate, XSLT, Azure, Java, Client Application Language (C/AL), verkkolasku, XML, PEPPOL

Abstract

Author:	Mete Güneysel
Title:	Exporting sales invoices from ERP system to e-invoice operator
Number of Pages:	45 pages
Date:	17 July 2022
Degree:	Bachelor of Engineering
Degree Programme:	Information and Communication Technology
Professional Major:	Multiform
Supervisors:	Janne Salonen, Senior Teacher

This engineering work was commissioned by Systencess Oy, who needed an integration between the ERP system Microsoft Dynamics 365 Business Central and the e-invoicing operator Maventa to invoice system generated sales invoices.

The integration focused on the export of sales invoices, through the growing PEPPOL network in the EU. The aim of the project was to create automation that would manage the workflow related to invoicing independently once a day. The integration tools used to implement the automation were decided in advance by Systencess Oy.

In the implementation, a database query was programmed into Business Central using the AL programming language, which enabled the automation created in Microsoft's Power Automate to retrieve the necessary billing information for sales invoices from the ERP system. Automation sent the invoice data to a Java program programmed in the Microsoft Azure cloud service, which created a standard Peppol BIS 3.0 e-invoice from the invoice data as a result of the XSLT transformation.

The result of the work was the creation of automation, which retrieved the sales invoices generated in the ERP system and delivered them to the e-invoicing operator. With this integration, Systencess Oy will be able to offer its current and future clients the possibility of invoicing through the PEPPOL network with Business Central.

Keywords: Microsoft Business Central, Power Automate, XSLT, Azure, Java, Client Application Language (C/AL), e-invoice, XML, PEPPOL

Sisällys

Lyhenteet

AL: *Application Language*. Microsoft Business Centralin kehittämisessä käytetty ohjelmointikieli.

BIS: *Business Interoperability Specification*. Muodolliset vaatimukset PEPPOL-verkostossa siirretyille sähköisille dokumenteille.

ERP: *Enterprise resource planning*. Toiminnanohjausjärjestelmä, jonka avulla yritys pystyy hallinnoimaan monia eri liiketoimintojen osiota keskitetysti.

HTML: *HyperText Markup Language*. Web-sivujen standardoitu kuvauskieli.

JSON: *Javascript Object Notation*. Tiedostotomuotostandardi, jota käytetään tiedonvälityksessä.

OVT: *Organisaatioiden välinen tiedonsiirto*. Mm. verkkolaskutuksessa käytetty tekniikka, jonka avulla siirretään tietoa järjestelmästä toiseen.

PEPPOL: *Pan European Public eProcurement On-Line.* Eurooppalainen verkosto standardimuotoisten sähköisten dokumenttien välittämistä varten.

XML: *Extensible Markup Language*. Merkintäkielistandardi. Standardin formaattia käytetään tiedon rakenteellistamisessa tai tiedon välittämisessä järjestelmien välillä.

XSLT: *Extensible Stylesheet Language Transformations*. Mm. XML tai JSON-tiedostomuotojen käsittelyssä käytetty kirjasto.

1 Johdanto

Tämän insinöörityön tavoitteena oli suunnitella ja kehittää integraatio, Systencess Oy:n tarjoaman toiminnanohjausjärjestelmä Business Centralin sekä verkkolaskuoperaattori Maventan välille. Integraation avulla, Systencess Oy pystyisi tarjoamaan asiakkailleen Peppol BIS 3 verkkolaskustandardin mukaisia laskuja, tarjoamansa toiminnanohjausjärjestelmän yhteydessä.

Integraation toteutuksessa kehitettiin automaatio, joka suoritti ajastetusti kerran päivässä sille määritetyt työnkulut. Automaatio haki laskutustiedot HTTP-kyselyiden avulla toiminnanohjausjärjestelmästä sekä lähetti kyselyistä saadut tiedot eteenpäin Azureen ohjelmoidulle pilvifunktiolle. Javalla ohjelmoitu pilvifunktio rakensi vastaanotetuista laskutustiedoista Peppol BIS 3 -standardin mukaisen verkkolaskun ja lähetti sen onnistuneesti verkkolaskuoperaattorille.

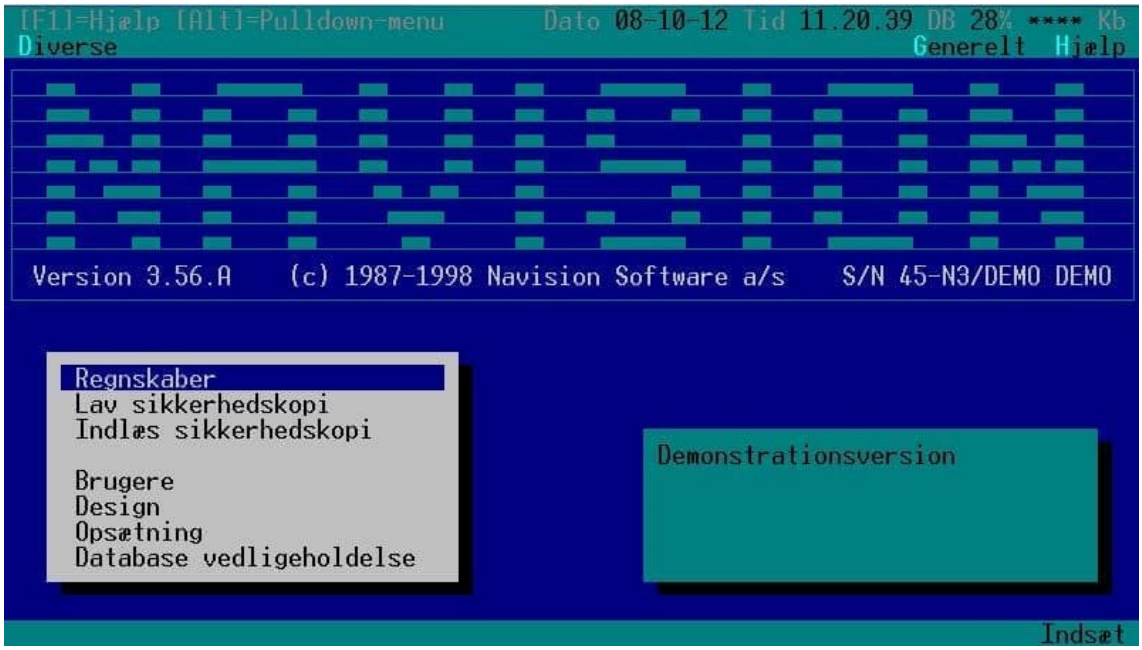
Tässä työssä kuvaillaan pääpiirteittäin integraatiossa käytettyjä työkaluja sekä niiden muodostamaa kokonaisuutta, mutta koska opinnäytetyön tuotoksena tehty automaatio on keskeinen osa tilaajayrityksen liiketoimintaa ja liikesalaisuutta, ei tässä työssä kaikilta osin avata yksityiskohtaisesti automatisoinnin eri vaiheita.

2 Ympäristö

2.1 Microsoft Dynamics 365 Business Central

Toiminnanohjausjärjestelmä Business Centralin kehitys alkoi 1980-luvulla, kun tanskalainen ohjelmistoyritys PC&C A/S (Personal Computing and Consulting) kehitti kirjanpitosovellus Pc Plus:n, josta löytyi aluksi vain kirjanpidon perustoiminnallisuudet [1]. Pc Plus oli aluksi MS-DOS-pohjainen ohjelma ja sitä pystyi käyttämään vain yksi käyttäjä, mutta ajan kuluessa ohjelmistoa kehitettiin ja myöhemmin sitä pystyi käyttämään yhä useampi käyttäjä samanaikaisesti [2].

Vuonna 1987 Pc Plus vaihtoi nimekseen Navigator ja se laajentui pelkästä kirjanpito-ohjelmasta monipuolisemmaksi ohjelmistoksi, johon liitettiin enemmän liiketoimintojen osastoja, mikä muodosti ohjelmistosta ajan mittaan enemmän ERP-järjestelmää muistuttavan kokonaisuuden [2]. Suuri kehitysaskel sovelluksessa tapahtui vuonna 1990, kun sovellus julkaistiin nimellä Navision (kuva 1). Tämän version mukana tuli Navisionin sovelluskehityksessä käytetty ohjelmointikieli Application Language. [3.] AL:n avulla järjestelmää pystyi kustomoimaan aikaisempaa tehokkaammin ja hyvinkin edistyksellisesti, asiakkaan tarpeisiin sopivaksi.



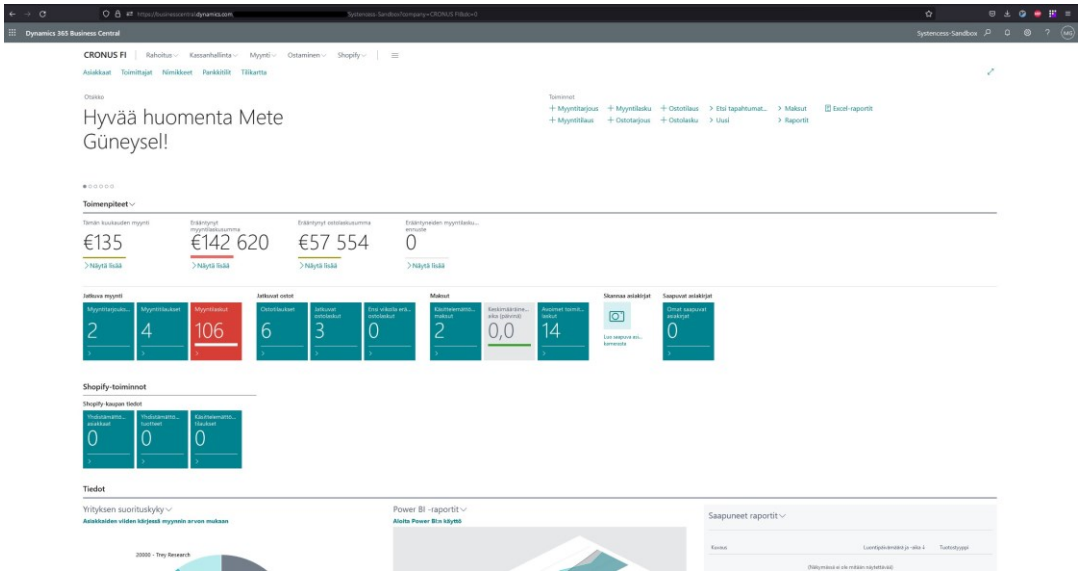
Kuva 1. Navision-sovelluksen demoversio tanskankielisellä käyttöliittymällä [4].

Vuonna 2002 Microsoft osti Navisionin kehittäjäyhtiön 1,48 miljardilla eurolla [5], mistä seurasi vuonna 2004 sovelluksen uudelleennimeäminen Microsoft Business Solutions Navision 4.0:ksi seuraavan versiojulkaisun yhteydessä [6]. Microsoft jatkoi silloisen Navisionin tuotekehitystä vielä monen vuoden ajan, rakentaen siitä täysmittaisen toiminnanohjausjärjestelmän [2].

Vuodesta 2004 lähtien, sovellukselle vakiintui vuosien ajan nimeksi Microsoft Dynamics Nav. Microsoft Dynamics Nav 2018:sta tuli sovelluksen viimeinen

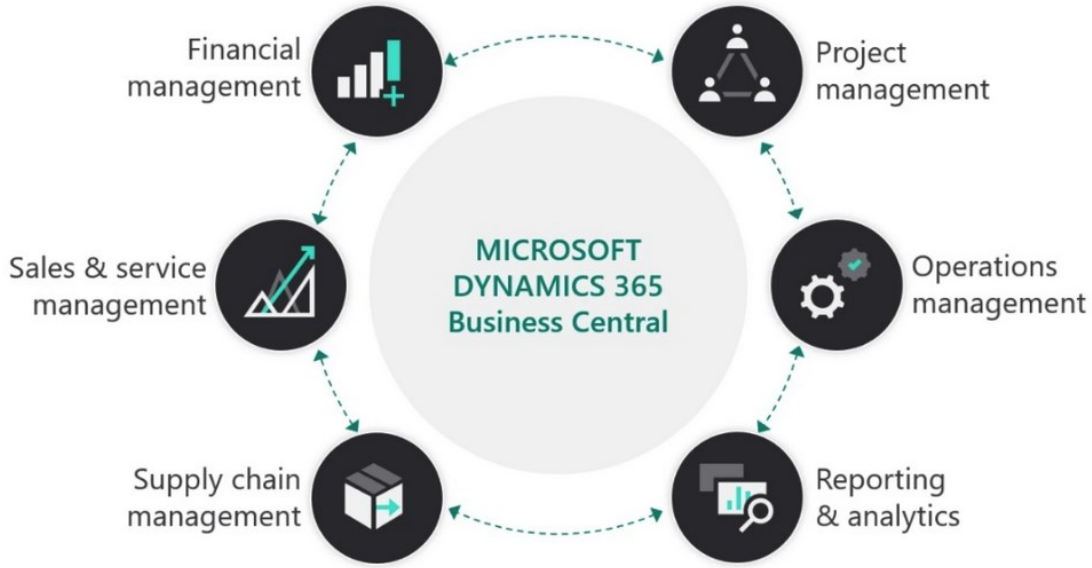
julkaistu versio, jonka jälkeen sovellus liitettiin osaksi Microsoft Dynamics 365 tuoteperhettä ja se nimettiin Microsoft Dynamics Business Centraliksi. Microsoft Dynamics Nav 2018:n tuki on määritetty loppuvan viimeistään 11.1.2028. [7.]

Tällä hetkellä Business Centralin käyttö onnistuu muun muassa verkkoselaimen kautta (kuva 2).



Kuva 2. Kuvakaappaus Business Centralin käyttöliittymästä Firefox-selaimesta 1.6.2022.

Business Central tarjoaa nykypäivänä kattavasti eri liiketoimintojen osa-alueita keskitettynä yhteen järjestelmään (kuva 3), jonka avulla mm. resurssien hallinta, analysointi ja raportointi on tehty mahdollisimman helpoksi. Osa Microsoftin omistamista O365-tuotteista, kuten Excel, Word, Outlook ja Sharepoint ovat valmiiksi integroituna järjestelmän toimintoihin. [8.]



Kuva 3. Business Centralista löytyvät liiketoimintojen eri osa-alueet [9].

2.2 Verkkolaskutus

2.2.1 PEPPOL

Vuonna 2008 käynnistyneessä PEPPOL-hankkeessa oli tavoitteena standardisoida maiden rajojen ylittävissä sähköisissä hankinnoissa käytettyjä maksutapojen tekniikoita Euroopan maissa. Hanke oli alun perin Euroopan Unionin suunnittelema sekä Euroopan komission ja PEPPOL-konsortion jäsenten rahoittama projekti. [10.]

PEPPOL-konsortio koostui alun perin 17 jäsenestä. Jäsenet olivat enimmäkseen julkisia, sähköisten hankintojen virastoja 11 maasta

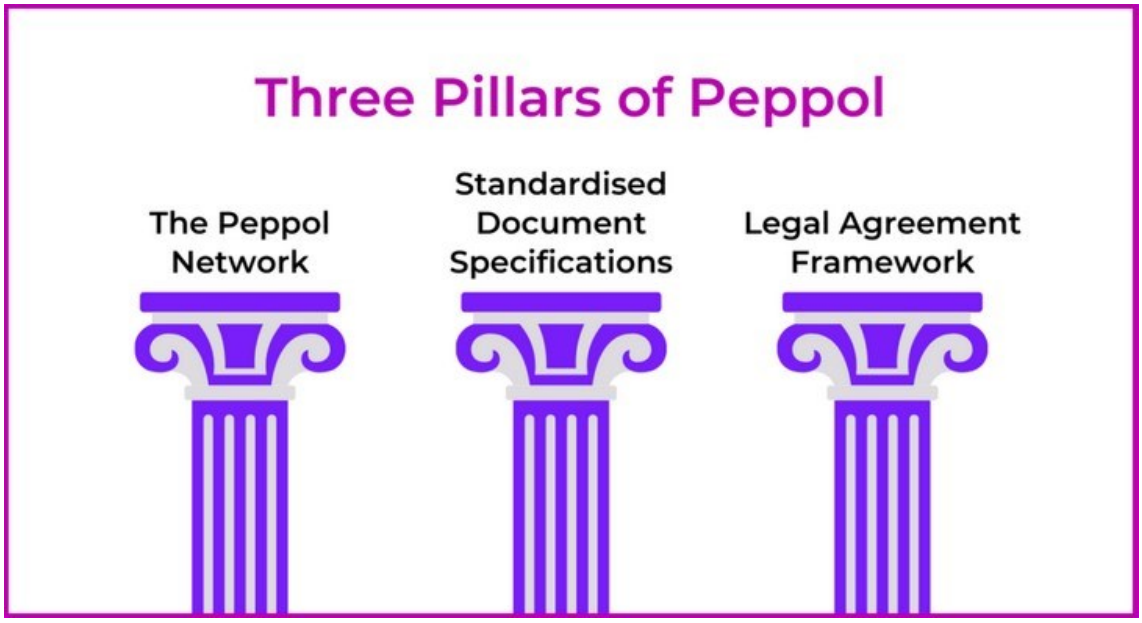
- Itävalta, Tanska, Suomi, Ranska, Saksa, Kreikka, Italia, Norja, Portugali, Ruotsi ja Iso-Britannia. [10.]

Elokuun 2012 lopussa PEPPOL-projekti saatiin päätökseen, jonka jälkeen projektin kehitystyö, ylläpito ja vastuualueet siirtyivät samana vuonna perustetulle OpenPeppol-organisaatiolle [10]. OpenPeppol määritellään voittoa tavoittelemattomaksi organisaatioksi, joka on määrätty toimimaan Belgian

AISBL:n lainalaisuudessa. OpenPeppol koostuu julkisen puolen sektorista sekä yksityisistä jäsenistä. [11.]

PEPPOL koostuu kolmesta pääkomponentista (kuva 4) [12]:

- PEPPOL-verkostosta
- Sähköisten dokumenttien spesifikaatiosta (Peppol BIS)
- Oikeudellisesta kehyksestä, joka määrittelee verkoston hallinnan

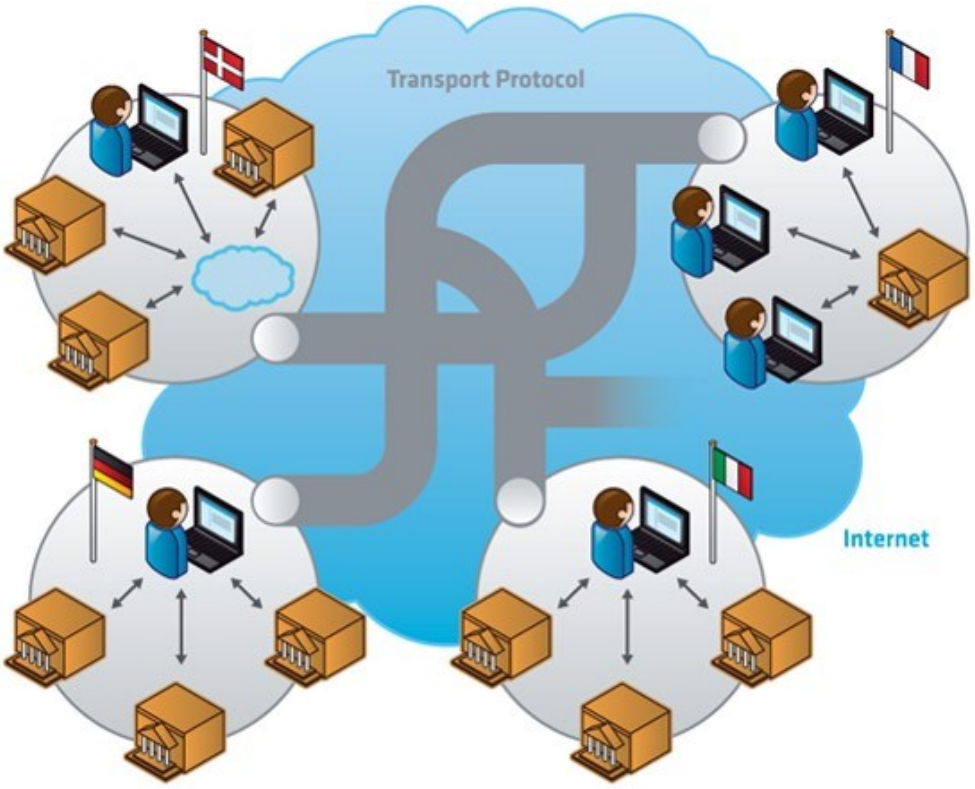


Kuva 4. PEPPOL-projektin kolme pääkomponenttia [13].

Vaikka PEPPOL oli alun perin suunnattu EU:n julkisyhteisöjen ja yritysten välille, niin vuonna 2022 OpenPeppol jäseniä löytyi jo 41 maasta, joista 32 maata oli Euroopan alueelta ja 9 Euroopan ulkopuolelta. Euroopan ulkopuoliset maat olivat

- Australia, Kanada, Kiina, Intia, Japani, Meksiko, Uusi Seelanti, Singapore ja Yhdysvallat. [14.]

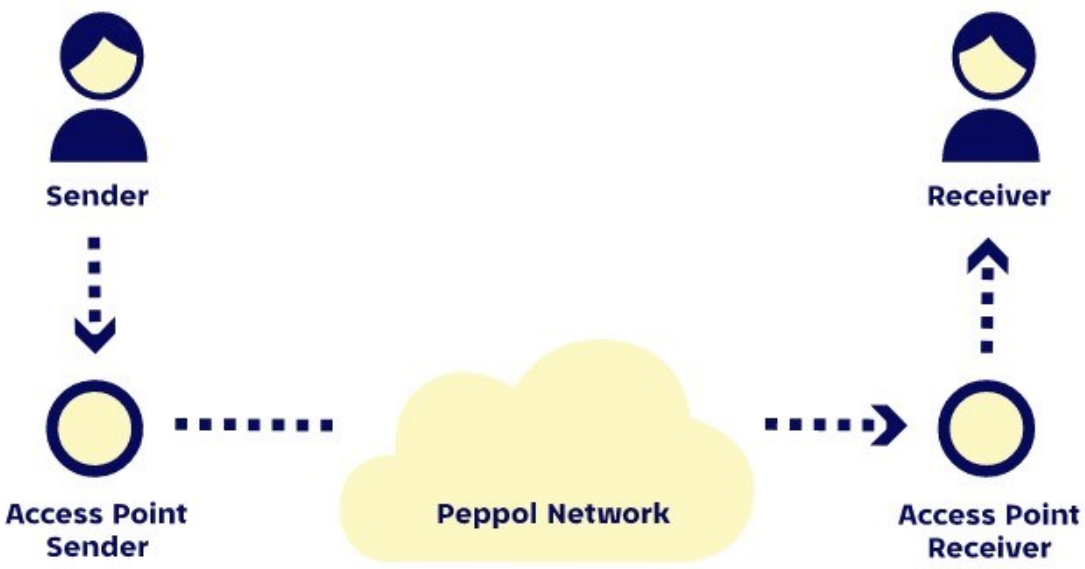
PEPPOL:ssa sähköisten dokumenttien siirto toimii niin kutsuttujen tukiasemien (Access Point) kautta (kuva 5). Tukiasema on tyypillisesti palvelu, joka on ostettavissa palveluntarjoajalta, kuten esimerkiksi verkkolaskuoperaattorilta. [15.]



Kuva 5. Mallinnus PEPPOL-verkoston toiminnasta eri maiden välillä [15].

Kun yritys lähettää standardinmukaisen laskun, palveluntarjoaja vastaanottaa laskun ja lähettää sen eteenpäin vastaanottajan palveluntarjoajalle, josta lasku siirtyy vastaanottajalle (kuva 6). Tätä prosessia kutsutaan 4-kulmaiseksi malliksi (4 corner model), koska se koostuu 4 osapuolesta. [16.] Osapuolet ovat

- lähettäjä
- lähettäjän tukiasema
- vastaanottaja
- vastaanottajan tukiasema. [16.]



Kuva 6. Sähköisen dokumentin siirrossa käytetty 4-kulmainen malli [16].

2.2.2 Sähköinen laskutus Euroopan Unionin jäsenmaissa

16.4.2014 Euroopan Unioni äänesti direktiivin 2014/55/EU puolesta, jonka mukaan kaikkien EU:n jäsenmaiden on sovellettava direktiivin edellyttämiä lakeja omassa maassaan ja laitettava ne käytäntöön [17]. Direktiivissä määritellään, että julkisen sektorin pitää pystyä vastaanottamaan hankintoihinsa liittyviä laskuja sähköisessä muodossa, myöhemmin määritellyn EN 16931-standardin mukaisesti [18].

Direktiivin tarkoituksena oli vähentää verkkolaskutuksiin liittyviä, jotka syntyivät eri maiden lakisääteisten vaatimusten sekä teknisten standardien rinnakkaisuudesta. Direktiivin ohjeistukset oli pantava käytäntöön viimeistään 27.11.2018. Jäsenmaa pystyi hakemaan siihen lykkäystä, mutta enintään 30 kuukauden päähän. [17.]

Tämän lakimuutoksen myötä alettiin jatkokehittää moniin maihin vakiintuneita laskustandardeja, jotta ne vastaisivat uutta EN 16931-standardissa määritettyä formaattia. Yksi näistä laskustandardeista oli PEPPOL-hankkeen Peppol BIS,

jonka 12.12.2018 julkaistu versio 3.0, täytti EN 16931-standardin mukaiset vaatimukset [19].

Taulukossa 1 näkyy Peppol BIS 3.0:n käyttöönottosuunnitelma, jossa vanha versio BIS2 määritettiin poistumaan käytöstä 15.5.2020 mennessä, uuden BIS3 version tullessa käyttöön.

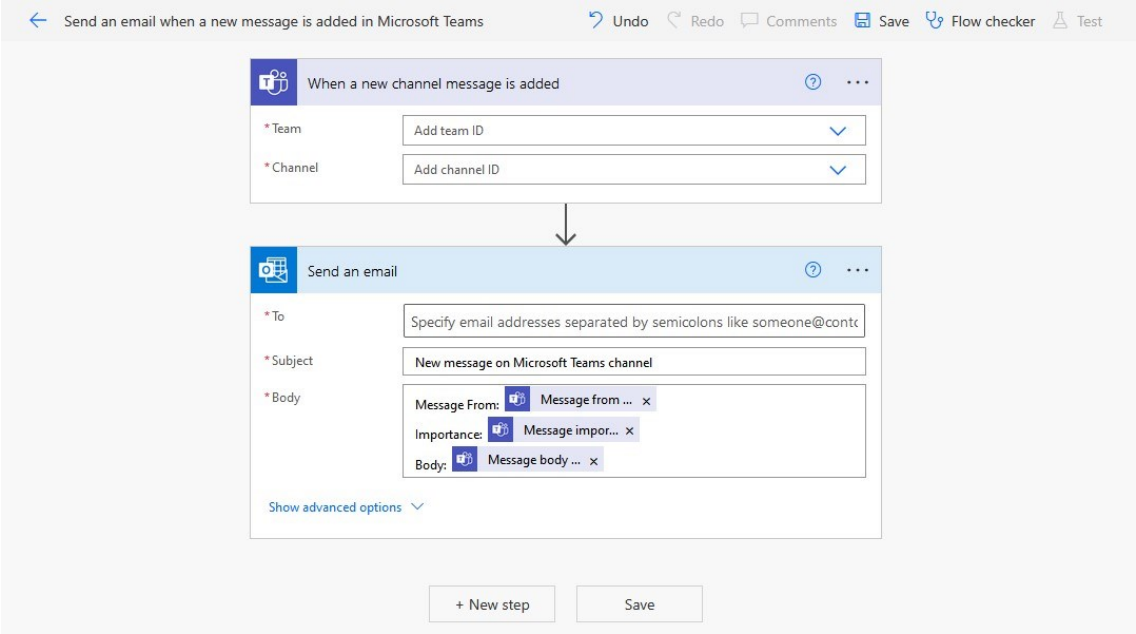
Taulukko 1. Peppol BIS versio 3.0:n käyttöönottosuunnitelma [19].

	Käyttöönotto alkaa	**Siirtymävaihe**	**Käyttöönotto valmis**
Peppol BIS versio	**12.12.2018**	**29.5.2019**	**15.5.2020**
BIS2	Pakollinen	Valittavissa	Lopetettu
BIS3	Valittavissa	Pakollinen	Pakollinen

2.3 Microsoft Power Automate

Microsoft Power Automate on Microsoftin kehittämä pilvipohjainen tehtävien automatisointityökalu, jonka avulla käyttäjä pystyy yhdistämään erilaisten sovellusten ja palvelujen työnkulkuja [20].

Power Automatea kutsutaan ns. no-code/ low-code palveluksi, joka käytännössä tarkoittaa, että yksinkertaisen työnkulkujen luominen automaatioon ei vaadi aikaisempaa ohjelmointiosaamista. Käyttöliittymä on graafinen ja sen avulla käyttäjä pystyy luomaan työnkulkuja vetämällä ja pudottamalla haluttuja prosesseja ketjuiksi toistensa perään (kuva 7). [21.]



Kuva 7. Kuvakaappaus Power Automaten käyttöliittymästä. Esimerkissä on automatisoitu työnkulku, joka lähettää käyttäjälle sähköpostiviestin, kun seurattu Teams-kanava vastaanottaa viestin.

2.4 Microsoft Azure

2.4.1 Palvelut

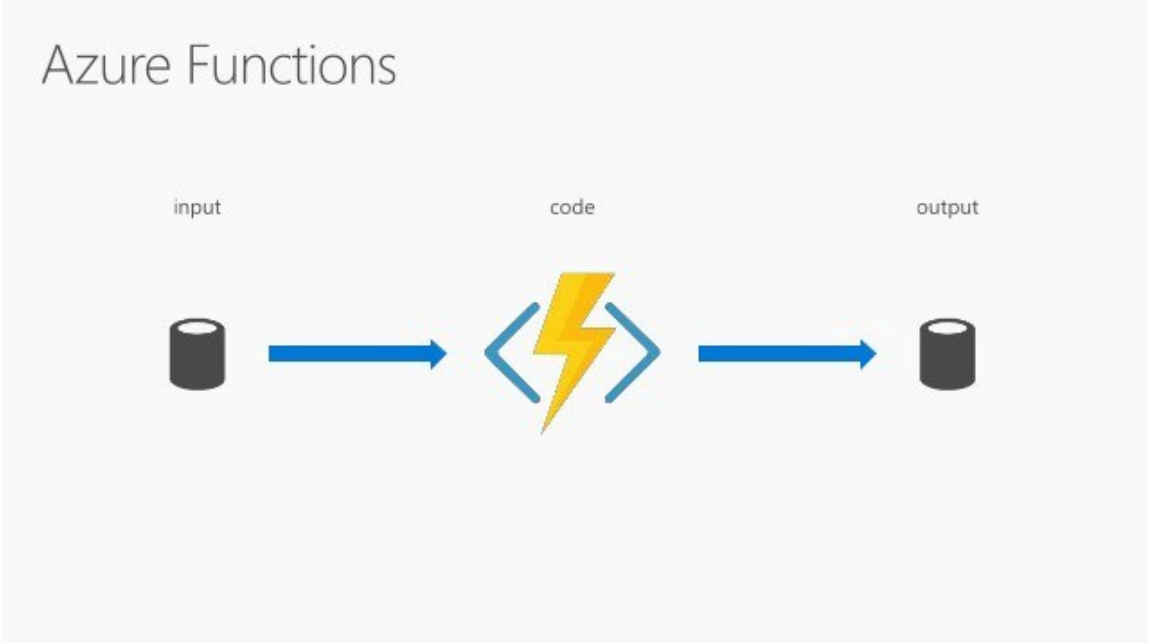
Microsoft Azure on Microsoftin omistama julkinen pilvipalvelualusta, joka tarjoaa kattavan valikoiman eri tyyppisiä palveluita monenlaisiin käyttötarpeisiin. Azuren tarjoamiin palveluihin kuuluvat mm.

- tekoäly + koneoppiminen
- analytiikka
- kehitysympäristö
- tunnistautuminen
- integraatiot
- IoT
- laskentateho
- tietokannat
- pilvitila

- virtuaalikoneet. [22.]

2.4.2 Pilvifunktio

Pilvifunktiolla tarkoitetaan ohjelmaa, joka toimii pilvipalvelimella. Azuren tapauksessa käyttäjä voi kirjoittaa ohjelmansa haluamallaan ohjelmointikielellä ja ladata sen Azuren palvelimelle. Tämän jälkeen ohjelmaa voidaan käyttää esimerkiksi tämän insinöörityön tapaan, eli lähettämällä HTTP-kutsu palvelimelle, johon pilvifunktio on konfiguroitu toimimaan. Tämä kutsu käynnistää ja suorittaa kyseisen ohjelman toiminnot, kuvassa 8 mallinnetulla tavalla.



Kuva 8. Yksinkertaistettu kuva pilvifunktion toiminnasta. Pilvifunktio käynnistyy vastaanottamalla HTTP-kutsun ja suorittaa funktioon ohjelmoidut toiminnot. [23.]

Pilvifunktioiden helppokäyttöisyys säästää myös palvelun kehitystyössä paljon aikaa, kun ei tarvitse miettiä esimerkiksi palvelininfrastruktuurin konfiguroimista, vaan pystytään keskittymään suoraan palvelun kehittämiseen. [23.]

Azure pilvifunktioita voi kirjoittaa ja ladata Azuren palvelimelle kuudella eri ohjelmointikielellä. Kielien toiminnallisuudet ja niiden käyttömahdollisuudet riippuvat

pilvifunktion asetustiedostossa määritetystä ajonaikaisesta versiosta. [24.] Taulukosta 2 näkyvät kaikki Azuren tukemat ohjelmointikielet.

Taulukko 2. Tuetut ohjelmointikielet Azuren pilvifunktiossa, riippuen käyttäjän asettamasta ajonaikaisesta versiosta. Sulkumerkeillä ympäröidyt eivät toistaiseksi ole Azuressa tuettuja ohjelmointikieliä, mutta niiden odotetaan saavuttavan täysi tuki tulevaisuudessa. [24.]

Kieli	**1.x**	**2.x**	**3.x**	**4.x**
C#	.NET Framework 4.8	.NET Core 2.1	.NET Core & .NET 5.0	.NET 6.0 & (.NET Framework 4.8)
Javascript	Node.js 6	Node.js 10 & 8	Node.js 10,12 & 14	Node.js 14 & 16
F#	.NET Framework 4.8	.NET Core 2.1	.NET Core 3.1	.NET 6.0
Java	N/A	Java 8	Java 8 & 11	Java 8 & 11
Powershell	N/A	PowerShell Core 6	PowerShell 7.0 & Core 6	PowerShell 7.0, (7.2)
Python	N/A	Python 3.6 & 3.7	Python 3.6, 3.7, 3.8 & 3.9	Python 3.7, 3.8 & 3.9

2.5 XML

Kun tietoa halutaan siirtää vastaanottajalta toiselle, on sen oltava mahdollisimman selkeästi strukturoitua ja kuvattua, jotta tiedon lähettäjä ja sen vastaanottaja ovat yhteisymmärryksessä sisällön merkityksestä. Vuonna 1998 julkaistun XML:n perusmuoto on standardoitu, joten se mahdollistaa tiedon esittämisen ja strukturoinnin järjestelmien ja alustojen välillä [25].

XML esittää tietoa elementtien avulla, joita kutsutaan myös tageiksi. XML:n syntaksi muistuttaa kovasti HTML:n syntaksia, mutta standardeilla on eroja. HTML:n ja XML:n eroina esimerkiksi ovat

- XML keskittyy pääosin tiedon siirtoon, kun taas HTML keskittyy tiedon esittämiseen
- XML:n tagit voidaan vapaasti määritellä itse tietosisällön mukaan, mutta HTML:n tagit ovat ennaltamääritetyt [26]

XML formaattia käytetään mm. PEPPOL-verkoston sähköisten dokumenttien sisältämän tiedon strukturoinnissa sekä määrittelyssä (kuva 9). Peppol BIS määrittelee mitä tietoa dokumentin pitää vähintäänkin sisältää ja minkälaisessa rakenteellisessa järjestyksessä, onnistuneen tiedonsiirron takaamiseksi [27].

```
<cac:AccountingCustomerParty>
    <cac:Party>
        <cbc:EndpointID schemeID="0037">003721291126</cbc:EndpointID>
        <cac:PartyIdentification>
            <cbc:ID schemeID="0037">50000</cbc:ID>
        </cac:PartyIdentification>
        <cac:PartyName>
            <cbc:Name>Relecloud</cbc:Name>
        </cac:PartyName>
        <cac:PostalAddress>
            <cbc:StreetName>Occam Court, 123</cbc:StreetName>
            <cbc:AdditionalStreetName>Surrey</cbc:AdditionalStreetName>
            <cbc:CityName>Helsinki</cbc:CityName>
            <cbc:PostalZone>02170</cbc:PostalZone>
            <cac:Country>
                <cbc:IdentificationCode>FI</cbc:IdentificationCode>
            </cac:Country>
        </cac:PostalAddress>
        <cac:PartyTaxScheme>
            <cbc:CompanyID>FI2829101</cbc:CompanyID>
            <cac:TaxScheme>
                <cbc:ID>VAT</cbc:ID>
            </cac:TaxScheme>
        </cac:PartyTaxScheme>
        <cac:PartyLegalEntity>
            <cbc:RegistrationName>Relecloud</cbc:RegistrationName>
            <cbc:CompanyID>282910-1</cbc:CompanyID>
        </cac:PartyLegalEntity>
        <cac:Contact>
            <cbc:Name>Veli Tiainen</cbc:Name>
            <cbc:ElectronicMail>veli.tiainen@contoso.com</cbc:ElectronicMail>
        </cac:Contact>
    </cac:Party>
</cac:AccountingCustomerParty>
```

Kuva 9. Kuvakaappaus standardinmukaisesta Peppol BIS 3.0 XML-dokumentista.

3 Integraatiosuunnitelma

3.1 Kehitysmenetelmä

Tässä työssä oli mahdollisuus vapaasti valita käytettävä kehitysmenetelmä. Eri kehitysmenetelmiä on ohjelmoinnin alalla monia ja tunnetuimpiin niistä kuuluvat mm. ketterä kehitys (Agile) ja vesiputousmalli (Waterfall) [28]. Projektin kehitysmenetelmäksi valikoitui vesiputousmalli, jota sovellettiin hieman sopivammaksi tätä työtä varten. Se koostuu viidestä eri vaiheesta

- ohjelmistovaatimusten määrittely
- suunnittelu
- ohjelmointi
- testaus
- käyttöönotto.

Tämä kehitysmenetelmä oli työn suunnitteluvaiheessa luontevinta, projektin maaliin saamisen kannalta.

3.2 Kehitystyökalut

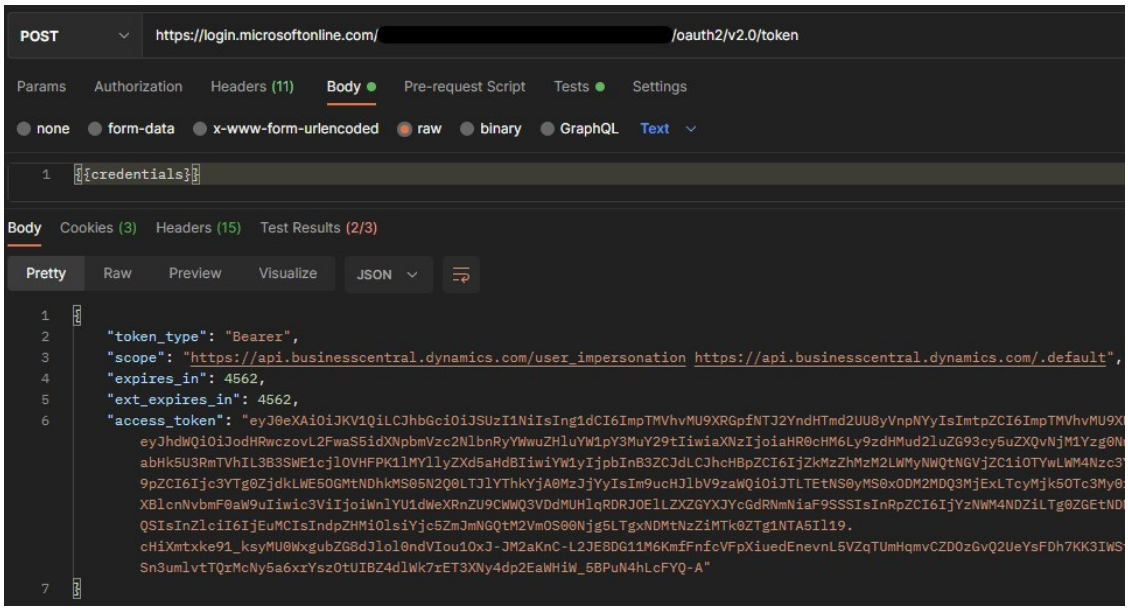
3.2.1 Postman

Postman on ohjelmontirajapinta sovellusliittymien kehitystyöhön. Se mahdollistaa HTTP-kyselyiden rakentamisen sekä kyselyiden vastausten mukana tulevan tietosisällön lukemisen. [29.]

Kun esimerkiksi automaattinen työnkulku hakee toiminnanohjausjärjestelmästä myyntilaskujen tiedot, tapahtuu tietojen hakeminen automaatiossa lähettämällä HTTP-kutsuja toiminnanohjausjärjestelmään. Kutsuissa kerrotaan järjestelmälle mitä tietoja halutaan, minkä jälkeen järjestelmä hakee tiedot ja palauttaa ne lähettäjälle HTTP-vastauksen mukana JSON-formaatissa.

Jokaiseen järjestelmän suorittamaan tietokantakyselyn kohdistamaan tauluun tarvitaan kyselyn suorittajalta suora tai epäsuora lukuoikeus [30]. Tästä syystä myyntilaskukyselyn mukaan pitää liittää myös tunnistautumiseen vaadittavat käyttäjätiedot, joista järjestelmä selvittää kyselyn suorittamiseen tarvittavat lukuoikeudet.

Graafisen käyttöliittymänsä ansiosta Postman helpottaa tämän kaiken hallinnoimista kehitystyön aikana(kuva 10). Postmaniin pystyy tallentamaan mm. web-osoitteita, tunnistautumisessa vaadittuja käyttäjätietoja, HTTP-kutsuissa vaadittuja headereita sekä tarvittaessa luomaan testejä. Postmanin avulla pystytään myös tarkastelemaan HTTP-kutsun vastauksen mukana saatua tietoa tai siihen liittyviä virheilmoituksia. [31.]



Kuva 10. Kuvakaappaus Postmanin käyttöliittymästä.

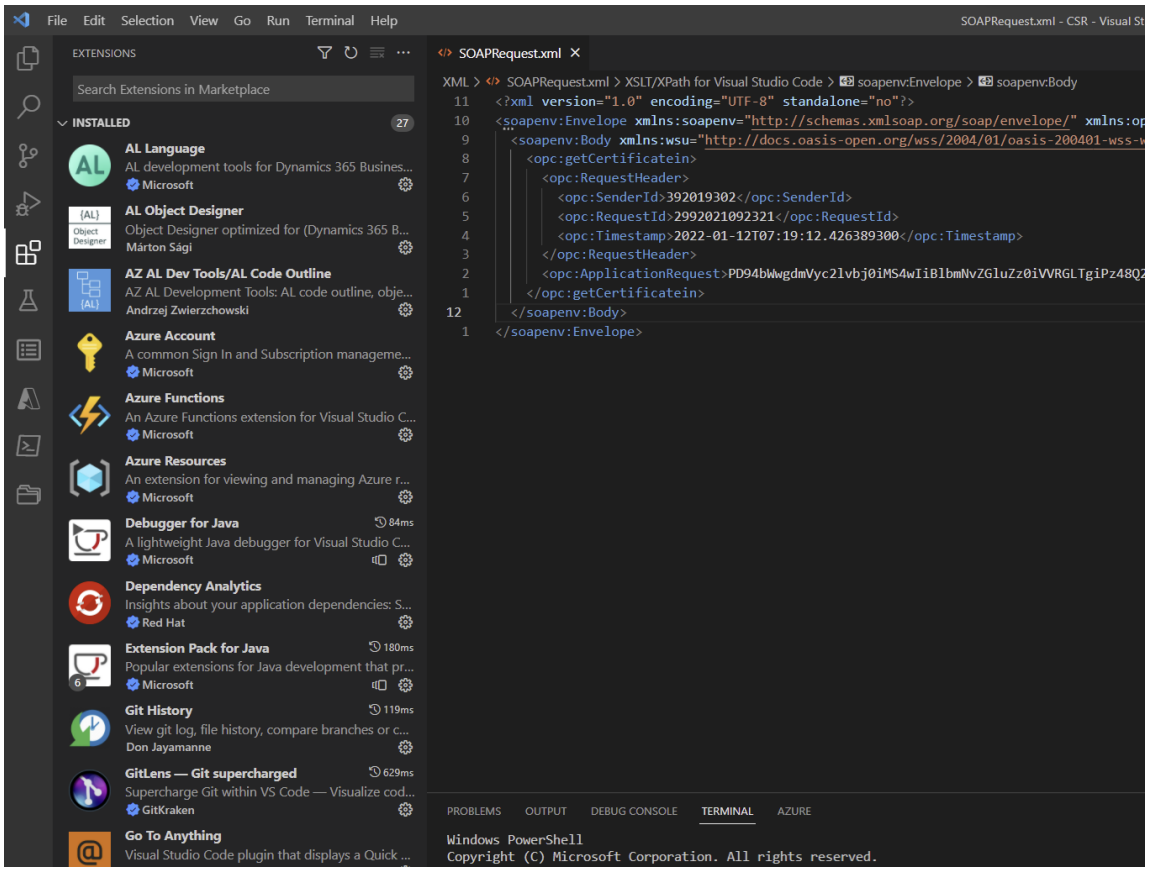
3.2.2 Visual Studio Code

Koodin editoimisessa ja kehitysympäristön integraatiossa käytettiin Microsoftin kehittämää Visual Studio Codea (VS Code). VS Code on ilmainen tekstieditori ja siitä löytyvät vahvat valmiudet Azuren pilvifunktiokehitykseen sekä Business Centralin laajennuskehitykseen.

VS Coden laajennuskaupasta (kuva 11) ladataan integraation kehittämistä helpottavat laajennukset

- Azure Tools
- Azure Account
- Azure Functions
- AL Language
- AL Object Designer
- Language Support for Java™ by Red Hat
- Maven for Java
- Test Runner for Java.

Ladatut laajennukset helpottavat ja automatisoivat Business Centralin laajennusten kehitystä AL:llä sekä Azuren pilvifunktioiden kehitystyötä Javalla.



Kuva 11. Kuvakaappaus VS Coden käyttöliittymästä. Laajennuksia voi etsiä laajennuskaupasta, vasemmassa yläreunassa olevasta hakukentästä.

3.2.3 XSLT

XSLT:n avulla pystytään muokkaamaan XML-tiedoston tietosisältöä halutunlaiseksi. Esimerkiksi myyntilaskun rakentaminen Peppol BIS 3.0 -standardin formaattiin tapahtuu tässä työssä XSLT-kielellä, pilvifunktiossa pyörivän Java koodin sisällä. XSLT-prosessori lukee käännösprosessissa sille annetun merkkijonon, eli esimerkiksi toiminnanohjausjärjestelmän palauttamat laskutustiedot sekä muunnosta varten ohjelmoidun .XSLT tyylitiedoston. Näiden kahden syötteen avulla prosessori muodostaa lopputulokseksi halutunlaisen dokumentin. [32.]

Esimerkkikoodi 1:ssä näkyy koodipätkä XSLT-tyylitiedostosta, missä ohjelmalle syötetyt yhteystiedot muunnetaan XSLT-tyylitiedoston avulla myyntilaskusta Peppol BIS 3.0 verkkolaskuformaattiin.

```
<xsl:template name="contactInfo">
     <xsl:param name="role"/>
      <xsl:param name="name"/>
      <xsl:param name="phoneNo"/>
      <xsl:param name="email"/>
      <xsl:element name="{$role}ContactName">{$name}</xsl:element>

      <xsl:element name="{$role}CommunicationDetails">
           <xsl:element name="{$role}PhoneNo">{$phoneNo}</xsl:element>
           <xsl:element name="{$role}Email">{$email}</xsl:element>
     </xsl:element>
</xsl:template>
```

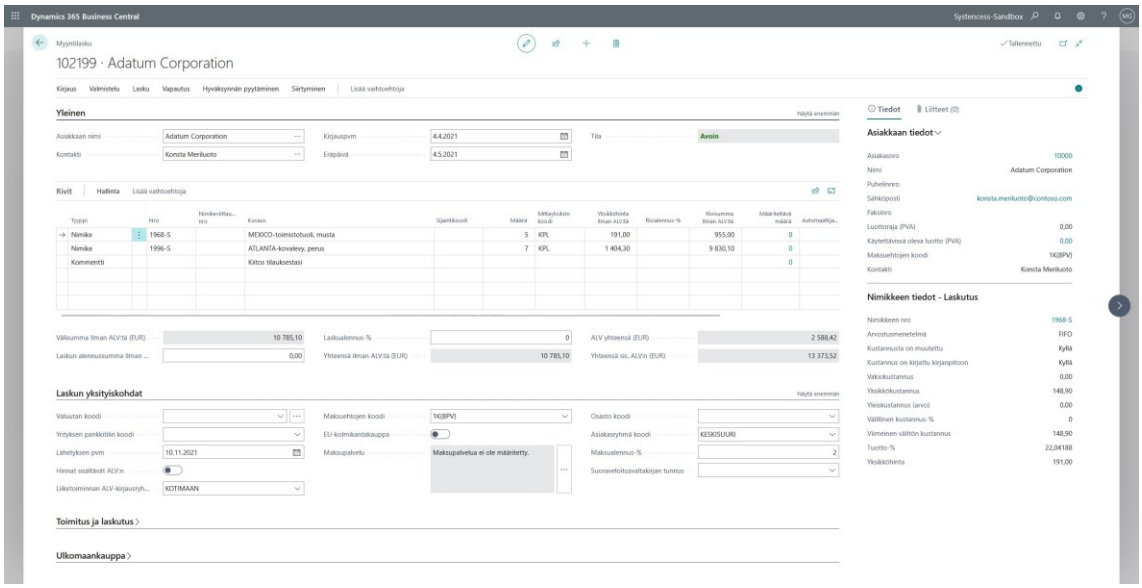
Esimerkkikoodi 1. Koodipätkä XSLT-kielellä kirjoitetusta muunnoksesta, missä ohjelma lukee sille syötetystä merkkijonosta ostajan ja myyjän tiedot ja luo niistä tagit XML tiedostoon.

Toiminnanohjausjärjestelmä palauttaa tiedot JSON-formaatissa, joten käännös JSON:sta XML:ään tehdään Azuren pilvifunktiossa Java koodissa. Käännösprosessia varten pilvifunktion projektiin on lisättävä riippuvuudeksi Saxxon kirjasto versiolla 9.9.0–1. Saxxon:sta löytyy kattavat työkalut XML-dokumenttien käsittelyyn mm. XSLT-muunnoksessa vaadittu XSLT-prosessori.

3.3 Myyntilaskun elinkaari Business Centralissa

3.3.1 Myyntilaskun luonti

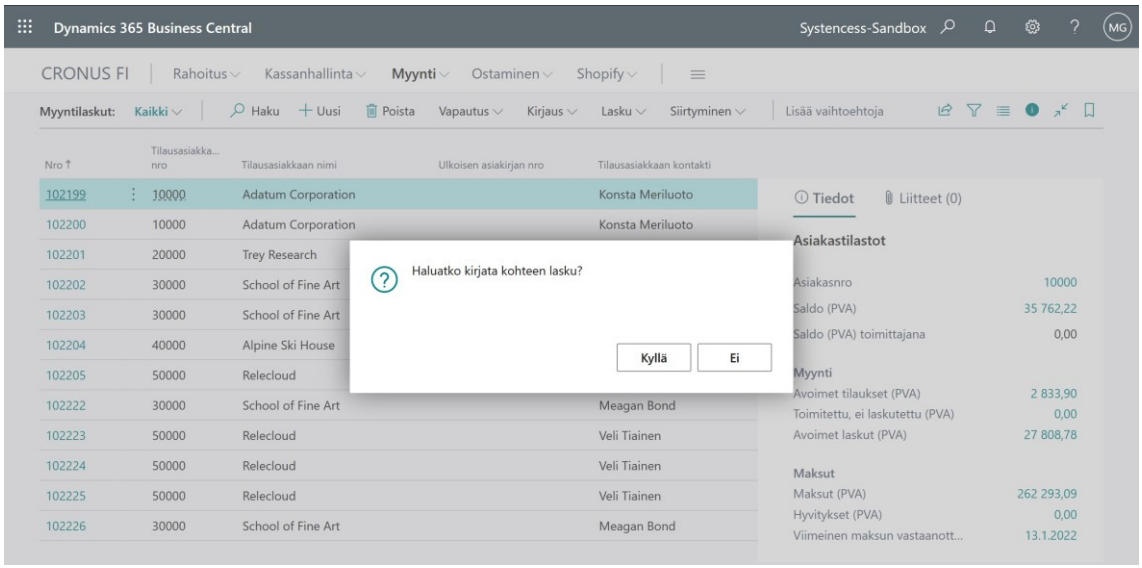
Myyntilasku luodaan Business Centralissa *Myyntilasku* näkymässä, painamalla valikkonappia ”Uusi”. Tämän jälkeen aukeaa näkymä (kuva 12), missä lisätään tarvittavat laskun tiedot lomakkeen kenttiin. Kun myyntilasku on valmis se pitää vielä vapauttaa käsiteltäväksi tulevaa laskutusta varten, joko luomisen yhteydessä tai sen jälkeen. Oletuksena laskun tila on *Avoin* ja vapauttamisen jälkeen laskun tila on *Vapautettu*.



Kuva 12. Kuvakaappaus Business Centralin myyntilaskun luomisnäkymästä.

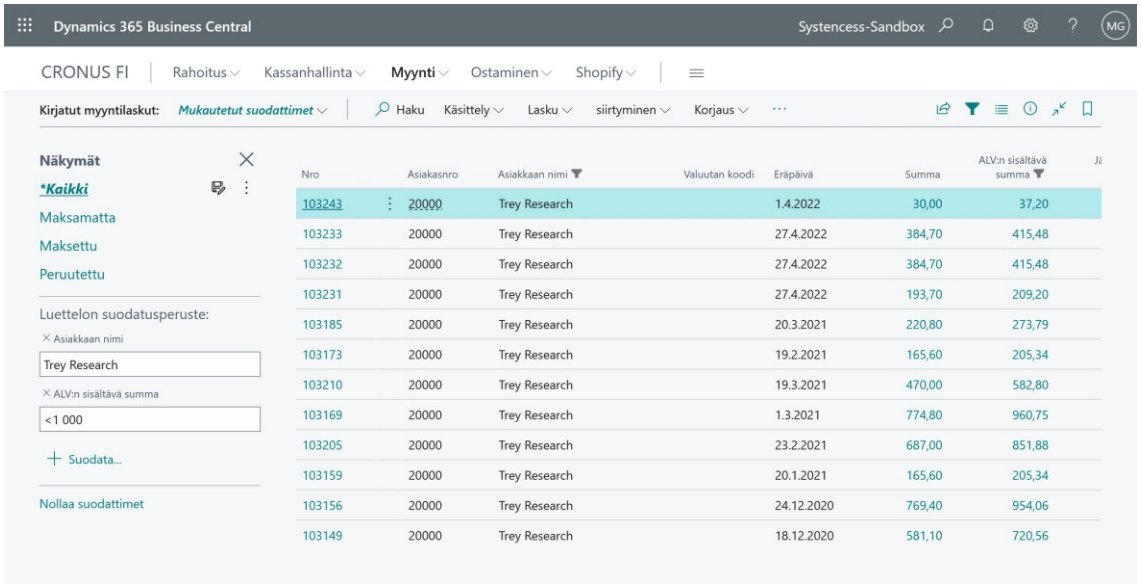
3.3.2 Myyntilaskun laskutus

Myyntilaskun laskutus Business Centralissa tapahtuu kirjaamalla myyntilasku toimitetuksi (kuva 13), jonka jälkeen lasku siirtyy järjestelmässä myyntilasku -näkymästä kirjattuihin myyntilaskuihin.



Kuva 13. Kuvakaappaus Business Centralista juuri ennen myyntilaskun kirjaamista laskutetuksi.

Kirjatut myyntilaskut näkymästä (kuva 14) pystytään jälkeenpäin etsimään jokaisen kirjatun myyntilaskun tiedot tehokkaasti, näkymästä löytyvien suodattimien avulla. Näkymästä voidaan myös tarvittaessa korjata tai peruuttaa kirjattuja myyntilaskuja.



Kuva 14. Kirjatut myyntilaskut -näkymä.

Suunnitelmana on luoda järjestelmässä oleviin myyntilaskuihin liittyvä tietokantakysely. Tämä kysely hakisi kaikkien valmiina laskutettaviksi olevien myyntilaskujen laskunumerot. Näiden laskunumeroiden avulla pystyttäisiin käyttämään järjestelmästä valmiiksi löytyvää kyselyä, joka hakisi halutut myyntilaskut tietokannasta.

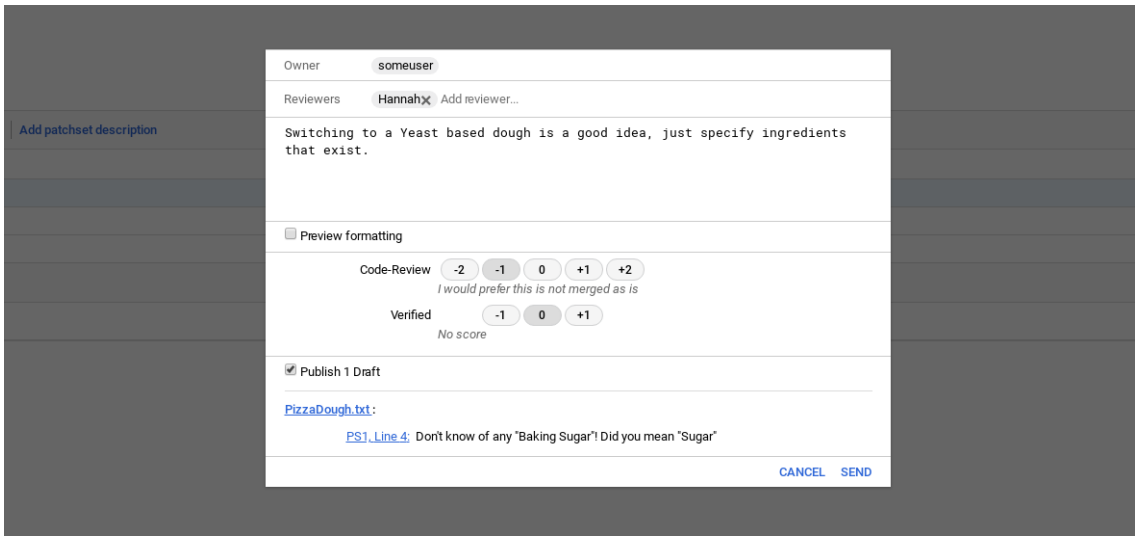
Kun myyntilaskut saataisiin onnistuneesti toimitettua verkkolaskuoperaattorille, niin sen jälkeen automaatio kävisi kirjaamassa myyntilaskut kirjattuihin myyntilaskuihin.

3.4 Versionhallinta

3.4.1 Pilvifunktion ja AL-laajennusten versionhallinta

Versionhallintaan käytetään Git-versionhallintaa. Git:n lisäksi, versionhallintaan on liitetty Googlen kehittämä [33] ilmainen verkkopohjainen työkalu Gerrit, joka on konfiguroitu Systencessin ylläpitämälle palvelimelle repositorien hallintaa varten. Gerritin työnkulkuun kuuluu, että aina kun koodiin tehdään muutoksia, muutoksen hyväksyntäviestiin (kuva 15) lisätään pääarvioija (review) ja arvioija (Cc). Arvioijat antavat arvion koodille väliltä (-2) ja (+2).

Hyväksyttyyn koodimuutokseen vaaditaan aina vähintään yksi (+2) ja samalla, arvioinneista ei saa löytyä yhtään (-2). Jos nämä ehdot toteutuvat, koodimuutos on valmis hyväksyttäväksi ja se voidaan tallentaa versionhallintaan. [34.]



Kuva 15. Esimerkki versionhallintaan liitetyn Gerritin käyttöliittymästä, jossa tehdään uudelle koodimuutokselle hyväksymispyyntöä [34].

3.4.2 Automaation työnkulkujen versionhallinta

Kun automaattinen työnkulku luodaan Power Automaten käyttöliittymässä, se tallentuu palveluun automaattisesti, käyttäjän tallennettua työnkulun ensimmäistä kertaa. Tästä syystä työnkululle ei erikseen luoda hakemistoa

Systencessin versionhallintaan. Työnkulku olisi mahdollista ladata Power Automaten palvelusta kokonaisuudessaan JSON-tiedostona, jonka voisi tallentaa versionhallintaan, mutta tässä toteutuksessa sitä ei koettu tarpeelliseksi.

4 Integraation toteutus

4.1 AL laajennuskehitys ja laajennuksen lataus järjestelmään

VS Codessa oletuspikanäppäinkomento F1 näytti kaikki mahdolliset komennot, jotka voitiin suorittaa kyseisessä editorin näkymässä. VS Codeen asennetun AL Object Designer -laajennuksen avulla, suoritettiin komento *AL: Go!*. Tällä komennolla laajennus loi automaattisesti tarvittavat tiedostot ja kansiot Business Central:n laajennuskehitystä varten.

Käyttäjälle jäi valittavaksi

- kohdekansio, mihin laajennuksen tiedostot tallentuvat
- ajonaikainen versio
- kehityksessä käytetty ympäristö.

Tässä toteutuksessa kehitystyö tapahtui Business Centralin pilvipalvelussa ja ajonaikaiseksi versioksi valittiin 9.0. Kaikkia edellämainittuja asetuksia pystyi tarvittaessa muuttamaan vielä jälkikäteen projektikansiosta löytyvistä app.json ja launch.json tiedostoista. App.json (esimerkkikoodi 2) tiedostoon voitiin määritellä mm.

- ajonaikainen versio
- ympäristön versio
- riippuvuudet
- onko laajennuksen koodi ladattavissa järjestelmästä.

```
{
    "id": "XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX",
    "name": "ExtensionPackage",
    "publisher": "DemoUser",
    "version": "1.0.0.0",
    "brief": "",
    "description": "",
    "privacyStatement": "",
    "EULA": "",
    "help": "",
    "url": "",
    "logo": "",
    "screenshots": [],
    "dependencies": [
        {
            "id": "XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX",
            "publisher": "Microsoft",
            "name": "System Application",
            "version": "20.0.0.0"
        },
        {
            "id": "XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX",
            "publisher": "Microsoft",
            "name": "Base Application",
            "version": "20.0.0.0"
        }
    ],
    "platform": "20.0.0.0",
    "idRanges": [
        {
            "from": 55500,
            "to": 55599
        }
    ],
    "contextSensitiveHelpUrl": "https://www.systencess.fi/",
    "resourceExposurePolicy": {
        "allowDebugging": true,
        "allowDownloadingSource": false,
        "includeSourceInSymbolFile": false
    },
    "runtime": "9.0",
    "features": [
        "TranslationFile",
        "GenerateCaptions"
    ]
}
```

Esimerkkikoodi 2. App.json tiedosto, johon on määritelty laajennuksen tiedot.

Projektin toisessa konfiguraatiotiedostossa launch.json:ssa (esimerkkikoodi 3), voitiin määritellä kehitettävään laajennukseen mm.

- mihin ympäristöön laajennuspaketti ladataan
- laajennuspaketin versio

- toiminnanohjausjärjestelmässä käynnistyvä näkymä, laajennuksen latauksen jälkeen
- onko kyseessä kehitys- vai tuotantoympäristö
- tietokantatauluihin kohdistuvien muutosten päivitystyyli.

```
{
    "version": "0.1.0",
    "configurations": [
        {
            "type": "al",
            "request": "launch",
            "name": "Systencess-Sandbox",
            "tenant": "XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX",
            "environmentType": "Sandbox",
            "environmentName": "Systencess-Sandbox",
            "startupObjectId": 20,
            "startupObjectType": "Page",
            "breakOnError": true,
            "launchBrowser": true,
            "enableLongRunningSqlStatements": true,
            "enableSqlInformationDebugger": true,
            "schemaUpdateMode": "Synchronize"
        }
    ]
}
```

Esimerkkikoodi 3. Launch.json tiedosto, jossa määritetään mm. ympäristö, mihin laajennus ladataan.

4.2 Myyntilaskujen haku tietokannasta

Business Centralista ei työn tekohetkellä löytynyt valmista tietokantakyselyä, jonka avulla vapautetut myyntilaskut olisi saatu suoraan haettua tietokannasta. Kysely luotiin itse laajennuksen yhteyteen ja se ohjelmoitiin Business Centralin kehitykseen tehdyllä AL kielellä.

Jokaiselle luodulle AL -laajennustyypille oli määritettävä oma uniikki ObjectID. Tämän ObjectID:n avulla Business Central tiesi mistä laajennuksesta ja toiminnosta oli kyse. Järjestelmän yhdenmukaisen toiminnan kannalta Microsoft oli määritellyt järjestelmänsä käyttöön tietyt ObjectID alueet ja kehittäjien käyttöön omat ObjectID alueet (taulukko 3). [35.]

Taulukko 3. Business Centralin kaikki ObjectID alueet ja niiden kuvaukset sekä käyttötarkoitukset [35].

ObjectID alue	**Kuvaus**
0 - 49 999	Varattu Business Centralin perussovelluksen toiminnallisuudelle. Vain Microsoftin käytössä.
50 000 - 99 999	Kustomointiin ja laajennuskehitykseen varattu ID alue. Vapaasti käytettävissä.
100 000 - 999 999	Business Centralin lokalisointiin varattu alue. Vain Microsoftin käytössä.
1 000 000 - 69 999 999	Alue on varattu Business Centralin sovelluskaupan laajennuskehitykseen. ID tältä alueelta pitää erikseen pyytää Microsoftilta. Microsoft ei suosittele tätä uusille kehittäjille.
70 000 000 - 74 999 999	Alue on varattu Business Centralin sovelluskaupan verkkolaajennuksiin. Microsoft suosittelee tätä uusille kehittäjille.

Tietokantakyselyn luonti aloitettiin luomalla projektiin APIQuery.al tiedosto. Tiedoston alkuun kirjoitettiin minkä tyyppinen laajennus oli kyseessä. Valittavissa oli 7 eri tyyppiä

- Codeunit, MenuSuite, Page, Query, Report, Table, XmlPort. [36.]

Kyselyn tyypiksi asetettiin esimerkkikoodi 4:n mukaisesti Query ja ObjectID:ksi arvo väliltä 55 500 – 55 599. Tämä arvoväli määritettiin laajennuksen asetustiedostossa app.json (esimerkkikoodi 2), joka kertoi Business Centralille varaamaan laajennukselle ObjectID:t näiltä väleiltä.

Näiden lisäksi pakollinen kenttä oli vielä QueryType, jolle annettiin arvo *API*. Tämä arvo kertoo Business Centralille, että kysely toimi web-palveluna, joten sen avulla ei voi esittää tietoa järjestelmän käyttöliittymässä, eikä kyselyllä voida kirjoittaa tietokantaan, vaan ainoastaan lukea sieltä. Syötettyjen tietojen avulla pystyttiin jo muodostamaan esimerkkikoodi 4:n kaltainen tietokantakyselyn alku. Kyseyln muut lisäasetukset olivat vapaasti valittavissa ja niiden määrittelyihin löytyi lisää tietoa Microsoftin dokumentaatiosta.

```
query 55555 "APIV2 - Get customer info"
{
    QueryType = API;
    Caption = 'Get customer name and address';
    APIPublisher = 'DemoPublisher';
    APIGroup = 'DemoGroup';
    APIVersion = 'v2.0';
    EntityName = 'getCustomerInfo';
    EntitySetName = 'getCustomerInfo';

    elements
    {
        dataitem(Customer; Customer)
        {
            column(Name; Name)
            {

            }

            column(Address; Address)
            {

            }
        }

    }
}
```

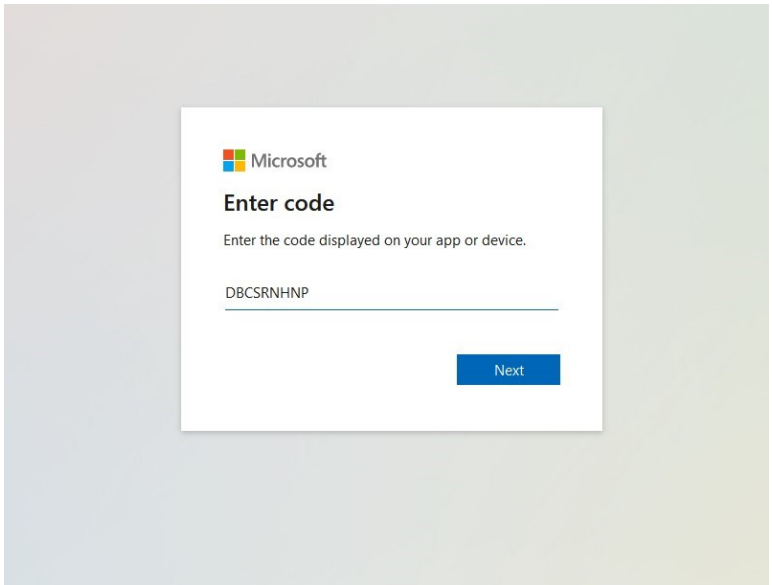
Esimerkkikoodi 4. Esimerkki yksinkertaisesta AL-kielellä tehdystä tietokantakyselystä. Kysely hakee taulusta *Customer* kaikki rivit ja palauttaa vastauksen mukana rivien sarakkeet *Name* sekä *Address*.

Kun myyntilaskujen numerot palauttava tietokantakysely oli luotu laajennukseen, oli laajennus vielä ladattava kokonaisuudessaan Business Centraliin. Laajennuksen lataaminen järjestelmään onnistui helposti VS Codeen asennetun AL Object Designer -laajennuksen avulla. Yksinkertaisin tapa ladata laajennuksia VS Coden kautta Business Centraliin oli painaa näppäinkomento Ctrl+F5, joka suoritti VS Codessa komennon *al.publishExistingExtension*. Tämä komento käynnisti tapahtumaketjun, missä ensin valittiin ympäristö, johon

laajennus haluttiin ladata. Valinnan jälkeen VS Coden alareunaan ilmestyi valikko (kuva 16), josta näkyi seuraavassa vaiheessa tarvittu tunnistautumiskoodi.

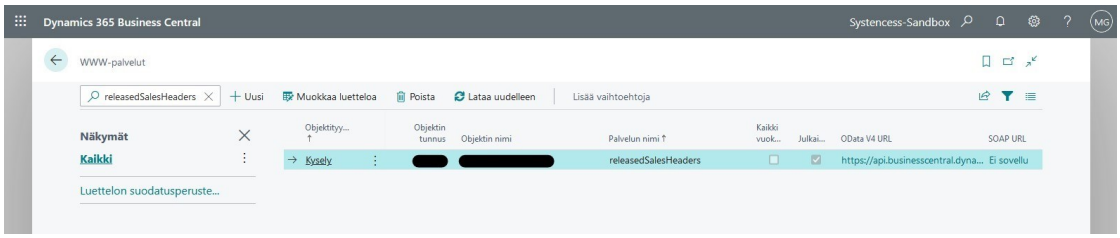
Kuva 16. Kuvakaappaus VS Codesta Ctrl+F5 komennon jälkeen, jolla käynnistetään laajennuksen lataus Business Centraliin.

Kun Copy & Open -nappia painettiin, tunnistautumiskoodi kopioitui leikepöydälle, josta se syötettiin selaimeen aukeavaan välilehden näkymään (kuva 17). Jos koodi täsmäsi selaimessa pyydettyn koodin kanssa, niin seuraavaksi kysyttiin vielä käyttäjän Microsoft-tunnuksia. Tunnuksilla tarkistettiin, että käyttäjällä oli tarvittavat oikeudet laajennusten lataamiseen järjestelmään. Tämän jälkeen laajennus oli ladattu onnistuneesti järjestelmään.



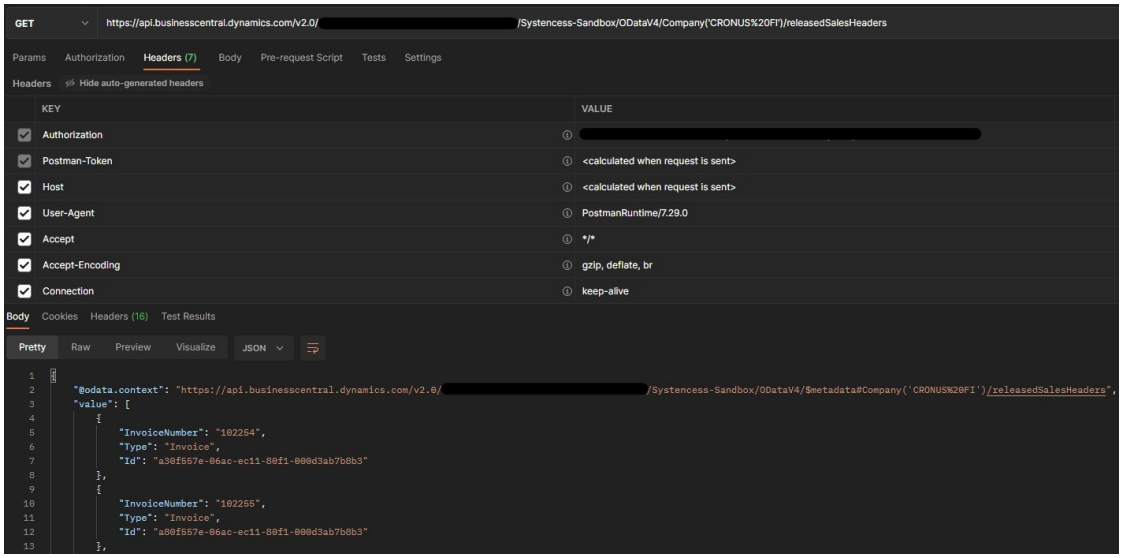
Kuva 17. Kuvakaappaus selaimeen aukeavasta välilehdestä, johon syötetään VS Codesta kopioitu tunnistautumiskoodi.

Kun laajennus oli saatu ladattua järjestelmään, se piti vielä käydä aktivoimassa järjestelmän WWW-palvelut näkymästä (kuva 18).



Kuva 18. Luotu tietokantakysely aktivoidaan järjestelmän käyttöön järjestelmän www-palvelut näkymästä.

Kun kysely oli valmis käytettäväksi aktivoinnin jälkeen, sitä pystyttiin testaamaan ja käyttämään HTTP-kutsulla (kuva 19).



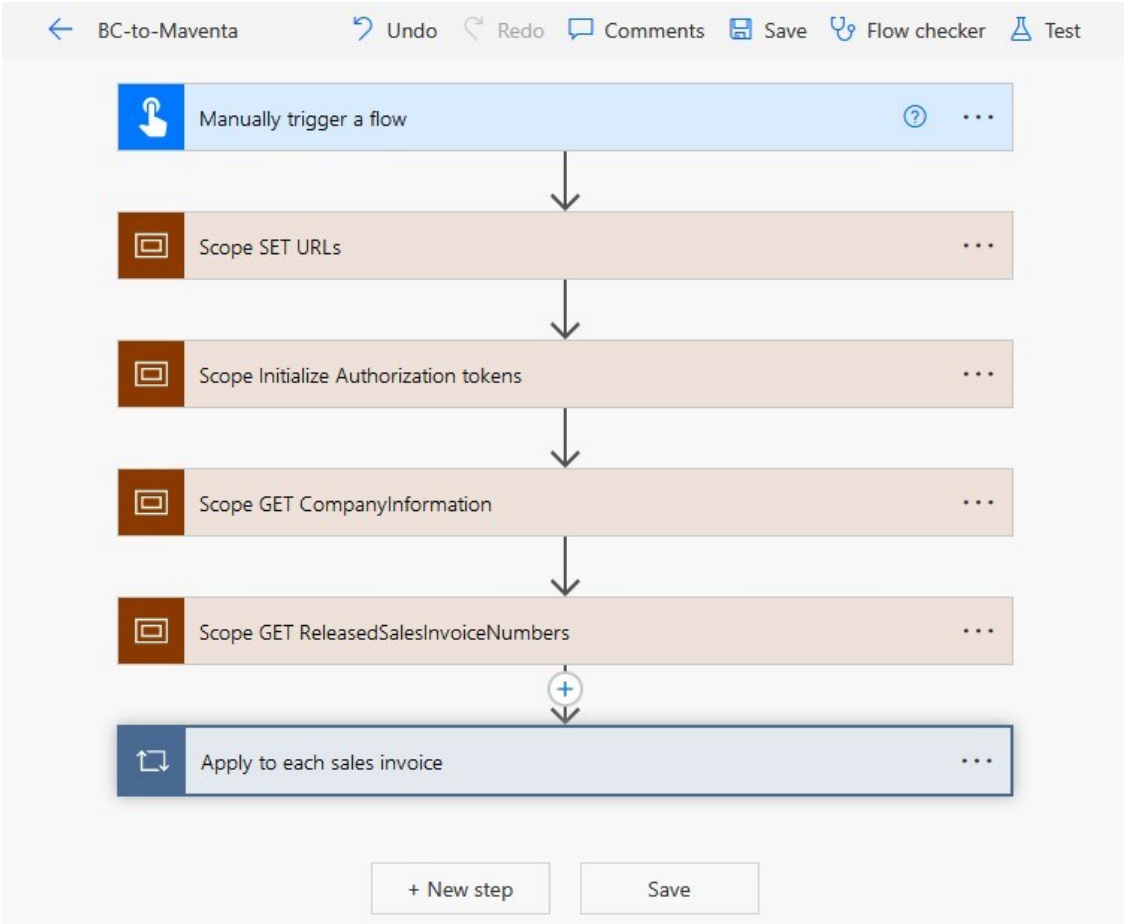
Kuva 19. Kuvakaappaus Postmanilla suoritetusta HTTP-kutsusta. Business Centraliin tehty HTTP-kysely, palauttaa vapautetut myyntilaskut ja myyntitilaukset.

4.3 Työnkulun automatisointi Power Automatella

Työnkulun automaatio jaettiin viiteen pääkategoriaan (kuva 20), jotka suoritettiin seuraavassa järjestyksessä

- automaation käyttämien URL-osoitteiden asettaminen globaaleiksi muuttujiksi
- tunnistautumisiin vaadittujen acces tokenien haku

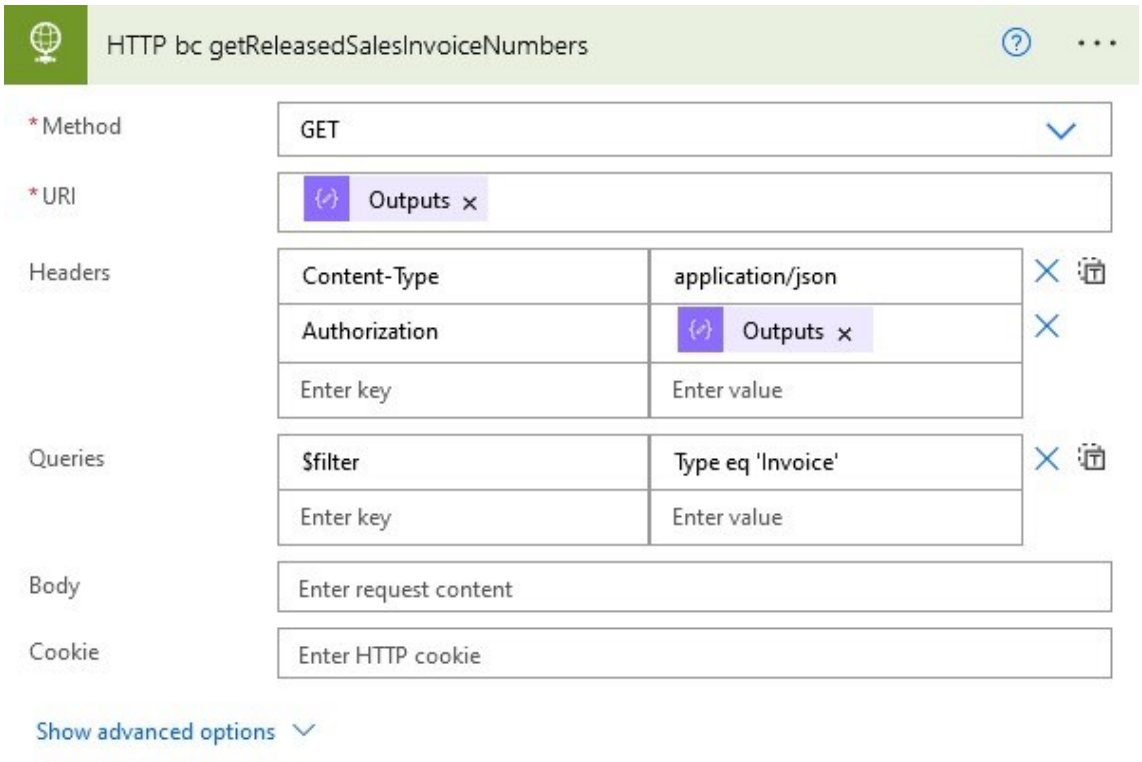
- laskuttajayrityksen tietojen haku
- vapautettujen myyntilaskujen laskunumeroiden haku
- myyntilaskujen lähetys Azuren pilvifunktiolle.



Kuva 20. Kuvakaappaus Power Automaten käyttöliittymästä, jossa on automaation suorittama työnkulku. Kuvan automaatio käynnistyy napin painalluksella, mutta se muutetaan myöhemmin toimimaan ajastetusti kerran päivässä.

Business Centraliin lisätty tietokantakysely palautti järjestelmästä kaikkien vapautettujen myyntilaskujen laskunumeroiden lisäksi vapauteuttujen myyntitilausten laskunumerot, joten tätä varten HTTP-kutsun mukaan oli liitettävä suodatinparametri, jonka avulla järjestelmä palautti vain vapautettujen myyntilaskujen laskunumerot. Suodatinparametrin lisäys tapahtui antamalla HTTP-kutsun mukaan parametri *$filter*, jolle asetettiin arvo *Type eq 'Invoice'*. Type sanalla viitattiin kyselyn palauttaman JSON:n kenttään *Type* (laskutyyppi)

ja *eq 'Invoice'* kertoi järjestelmälle, että palautuneista tuloksista haluttiin suodattaa pois kaikki muut paitsi myyntilaskut (kuva 21).



Kuva 21. Power Automaten HTTP-kutsu Business Centraliin, joka palauttaa vapautettujen myyntilaskujen laskunumerot. Web-osoite ja tunnistautumistoken ovat tallennettuina dynaamisiin *Outputs* ympäristömuuttujiin. Suodatin on lisätty kutsun URL-parametriksi.

4.4 Azure pilvifunktio

Kun pilvifunktio oli ohjelmoitu ja konfiguroitu toimimaan Azuressa, aloitettiin funktion ajaman ohjelman kirjoittaminen.

Automaation haettua myyntilaskujen tiedot Business Centralista oli seuraava vaihe lähettäessään laskuja verkkolaskuoperaattorille. Jos pilvifunktio kohtasi virhetilanteen kutsun seurauksena, pilvifunktion suoritus pysähtyi ja se palautti automaatiolle virheilmoituksen vastauksena. Ohjelmassa oli 4 eri vaihetta, jotka olivat

- saapuvan HTTP-kutsun sisällön validointi
- kutsun mukana tulevan myyntilaskun kääntäminen Peppol BIS standardin mukaiseksi XSLT-muunnoksella
- lähtevän HTTP-kutsun rakentaminen ja myyntilaskun liittäminen kutsun multipart/form-data tietosisällöksi
- sanoman lähettäminen verkkolaskuoperaattorille.

4.5 XSLT-muunnos

Pilvifunktion tekemä muunnos JSON:sta XML:ksi tapahtui käyttämällä XSLT kirjastoa. XSLT:n avulla tieto saatiin tarkasti määriteltyä ja strukturoitua Peppol BIS 3.0 standardin mukaiseksi. XSLT-muunnos tapahtui Java-ohjelmassa esimerkkikoodi 5:n mukaisesti. Ensiksi luotiin merkkijono <invoice></invoice>. Luodun merkkijonon elementin sisälle laitettiin kutsun mukana saadut laskutustiedot. Kun merkkijono oli valmis, XSLT-prosessori pystyi lukemaan tietoa sille syötetystä merkkijonosta avain-arvo periaatteella ja teki syötetyistä laskutustiedoista käännöksen, Peppol BIS 3.0 -standardin mukaiseen XML formaattiin.

```
try (final StringWriter stringWriter = new StringWriter();){

     final Source xmlSource = new StreamSource(new
StringReader("<invoice>" + json + "</invoice>"));
     final Source xsltSource = new StreamSource(new
StringReader(xslt));

     final Templates templates =
TransformerFactory.newInstance().newTemplates(xsltSource);
     final StreamResult transformedXML = new
StreamResult(stringWriter);
     final Transformer XSLTtransformer = templates.newTransformer();

     XSLTtransformer.transform(xmlSource, transformedXML);

     return stringWriter.toString();

  } catch(final Exception e) {
      throw new TransformerException(e.getMessage());
}
```

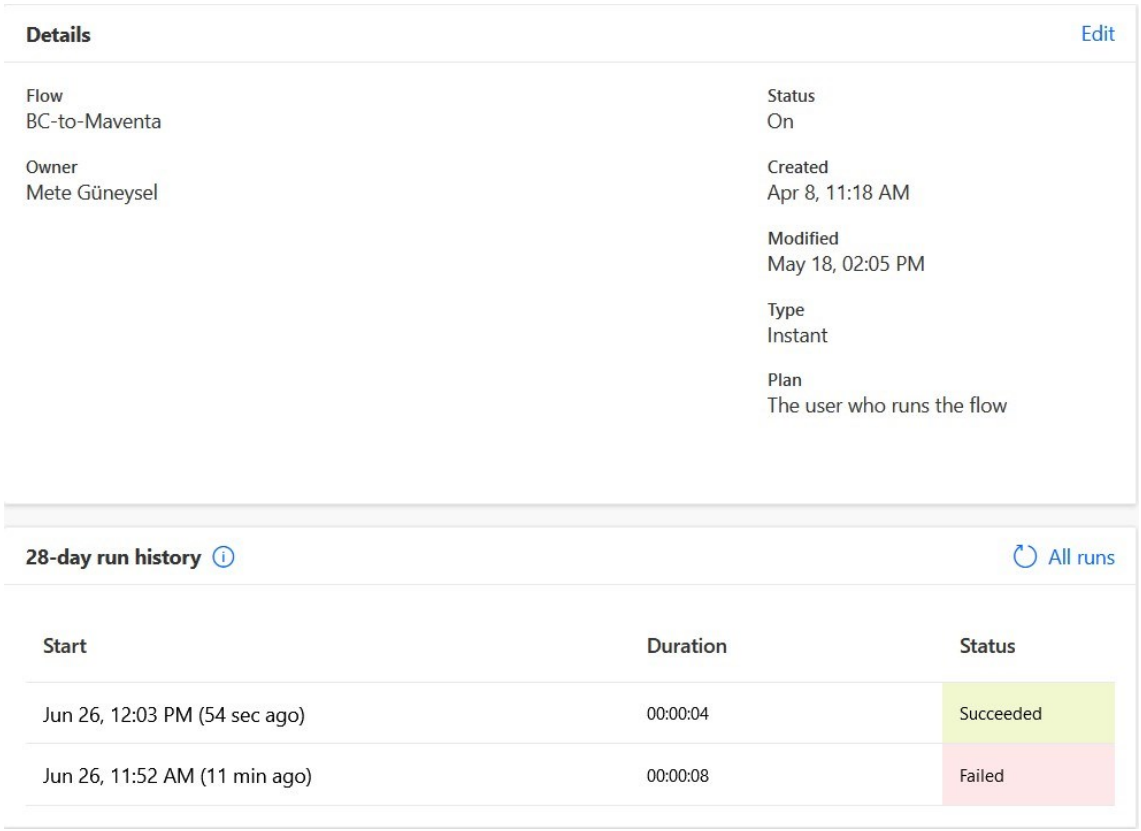
Esimerkkikoodi 5. Java ohjelmassa suoritettu XSLT-käännös. Lopputuloksena syntyy Peppol BIS 3.0 standardin mukainen myyntilasku.

4.6 Verkkolaskun validointi

Poikkeustilanteiden varalta, myyntilaskuista muodostuneet verkkolaskut validoitiin automaation kolmessa eri vaiheessa.

4.6.1 Ensimmäinen validointi

Ensimmäinen validointi tapahtui pilvifunktion suorittamassa XSLT-käännöksessä. Jos vastaanotetusta myyntilaskusta löytyi tarvittavat tiedot, lähetti ohjelma käännöksestä muodostuneen verkkolaskun verkkolaskuoperaattorille. Jos kaikkia tarvittavia tietoja ei löytynyt myyntilaskusta, käännös epäonnistui, ohjelma keskeytyi ja siitä palautui virheilmoitus automaatiolle (kuva 22).



Kuva 22. Kuvakaappaus Power Automaten käyttöliittymästä. Käyttöliittymän lokitiedoista pystytään tarkistamaan, onko kyseinen automaatiokerta onnistunut vai epäonnistunut. Kumpaakin tilannetta on mahdollista tutkia käyttöliittymässä vielä yksityiskohtaisemmin.

4.6.2 Toinen validointi

Toisen validoinnin laskulle teki verkkolaskuoperaattori Maventa omassa palvelussaan, vastaanottettuaan myyntilaskun. Palvelu vastaanotti verkkolaskuja vain tietyissä standardinmukaisissa laskuformaateissa, kuten esimerkiksi Peppol BIS 3.0 standardin XML-formaatissa.

Vaikka verkkolasku olisikin mennyt perille Maventaan, se ei automaattisesti tarkoittanut, että lasku olisi toimitettu onnistuneesti perille, sillä Maventa palautti ilmoituksen onnistuneesta laskun vastaanotosta, jos se oli muodollisesti standardinmukainen. Maventa ei tarkistanut verkkolaskun sisällön oikeellisuutta vielä tässä vaiheessa tarkasti, eli se ei tarkistanut löytyivätkö esimerkiksi myyntilaskuun asetetut vastaanottajan ovt-tunnus ja ovt-operaattori tiedoista. Jos ovt-tunnusta ei löytynyt laskusta tai se oli virheellinen, Maventa asetti laskun tilaksi *FAILED* (esimerkkikoodi 6). Tätä tilannetta varten oli tehtävä vielä kolmas validointi automaation toisessa työnkulussa.

```
{
    "id": "XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX",
    "status": "FAILED",
    "reference": "99999999999",
    "number": "102250",
    "sender": {
        "eia": null,
        "bid": "1234567-8",
        "name": "CRONUS FI",
        "country": "FI"
    },
    "recipient": {
        "eia": "003718540478",
        "bid": "1234567-1",
        "name": "Adatum Corporation",
        "country": "FI",
        "operator": "NDEAFIHH"
    },
    "created_at": "2022-03-28T07:44:46Z",
    "date": "2022-03-02",
    "date_due": "2022-04-02",
    "source_format": "PEPPOL30",
    "sum": 99.0,
    "sum_tax": 122.76,
    "currency": "EUR",
    "destination": "NOT_FOUND",
    "comment": null,
    "files": [],
    "actions": [
        {
            "type": "CREATED",
            "channel": "EINVOICE",
            "message": null,
            "happened_at": "2022-03-28T07:44:46Z"
        },
        {
            "type": "ERROR",
            "channel": null,
            "message": "SEND ERROR (REASON: BANK)",
            "happened_at": "2022-03-28T07:44:58Z"
        }
    ]
}
```


Esimerkkikoodi 6. Maventan palvelusta palautunut vastaus JSON-muodossa, josta huomataan, että myyntilaskun tila (status) on *FAILED*. Tiedoston sisällöstä voidaan päätellä, että myyntilaskun tietoihin on syötetty väärä OVT-operaattori tai OVT-tunnus, jonka takia laskua ei pystytä toimittamaan vastaanottajalle.

4.6.3 Kolmas validointi

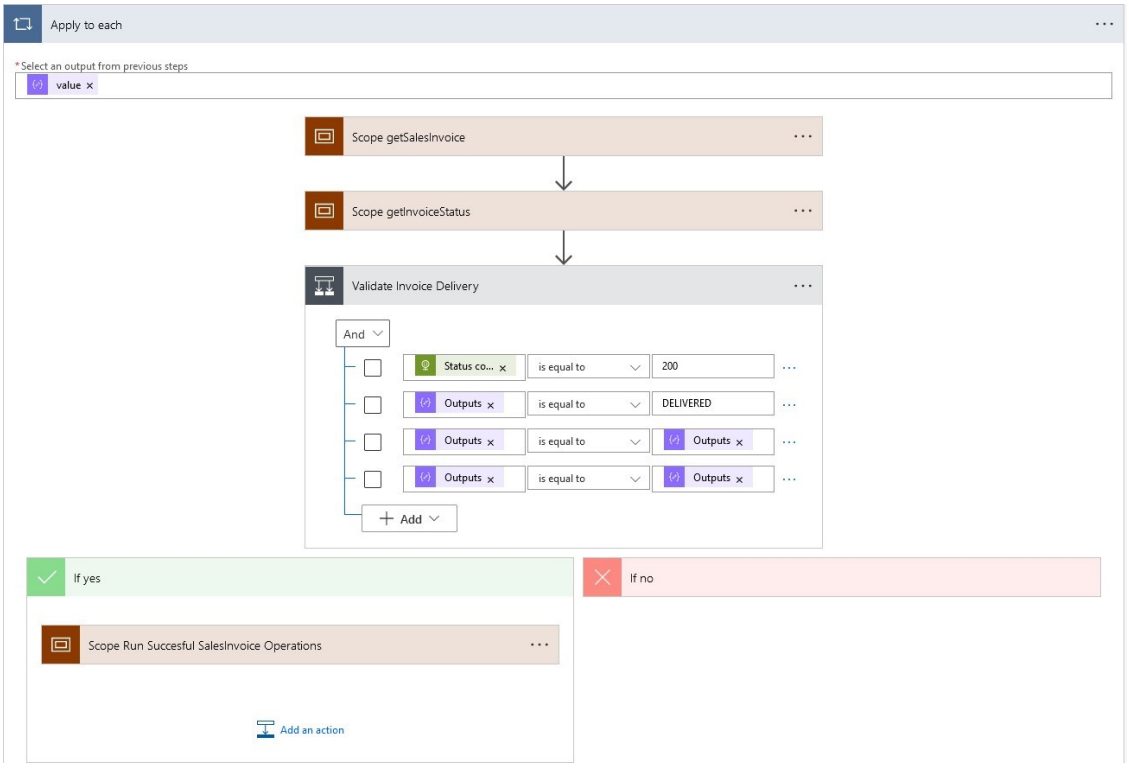
Kolmas ja viimeinen validointi tapahtui Power Automaten toisessa automatisoidussa työnkulussa, joka suoritettiin 30 minuuttia ensimmäisen työnkulun jälkeen. Tällä 30 minuutin aikavälillä varmistettiin se, että edelliset

verkkolaskut olivat varmasti toimitettu Maventaan ja validointi oli ehtinyt tapahtumaan Maventan palvelimilla.

Tämä työnkulku lähetti pilvifunktiolle myyntilaskun laskunumeron ja pilvifunktio teki laskunumeron avulla HTTP-kutsun Maventan palvelimelle. Palvelin palautti pyydetyn laskun tiedot ja tiedoista tarkastettiin, että

- palautuneen HTTP-kutsun tila oli 200
- myyntilaskun tila oli *DELIVERED*
- laskunumero täsmäsi tarkastettavan laskun kanssa
- laskun summa täsmäsi tarkastettavan laskun kanssa.

Jos kaikki tiedot täsmäsivät, työnkulku teki tarvittavat toimenpiteet ja siirsi myyntilaskun kirjattuihin myyntilaskuihin Business Centralissa (kuva 23).



Kuva 23. Kuvakaappaus Power Automaten toisesta työnkulusta, jossa varmistetaan Maventaan lähetetyn myyntilaskun onnistunut toimitus.

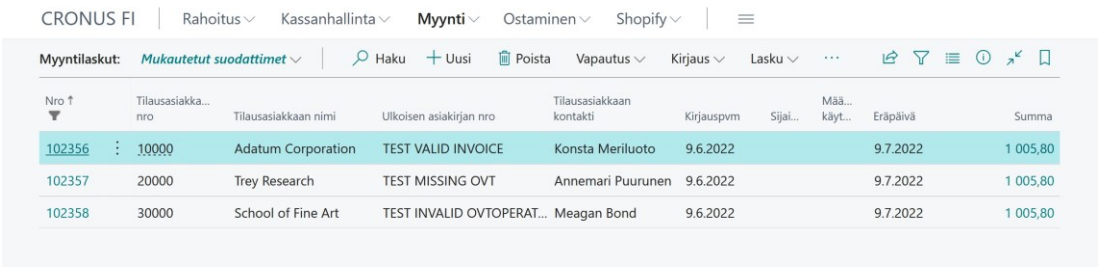
5 Integraation testaus

5.1 Myyntilaskujen luonti

Testausta varten luotiin kolme erillistä myyntilaskua (kuva 24). Myyntilasku 1:n tietoihin syötettiin kaikki onnistuneeseen verkkolaskutukseen tarvittavat tiedot. Myyntilaskun lopputulokseksi odotettiin muodostuvan validi verkkolasku, jonka Maventa hyväksyisi ja merkitsisi kehitysympäristössään toimitetuksi. Laskun lisätietokenttään laitettiin teksti ”TEST VALID INVOICE”.

Myyntilasku 2:n asiakastiedoista jätettiin tarkoituksella pois vastaanottajan ovt-tunnus, jonka odotettiin aiheuttavan virheilmoitus pilvifunktion käännösprosessissa. Tämän seurauksena laskutiedoista ei muodostuisi verkkolaskua, jonka pilvifunktio olisi voinut lähettää Maventaan eli työnkulun suoritus pysähtyisi jo ennen Maventaa. Laskun lisätietokenttään laitettiin teksti ”TEST MISSING OVT”.

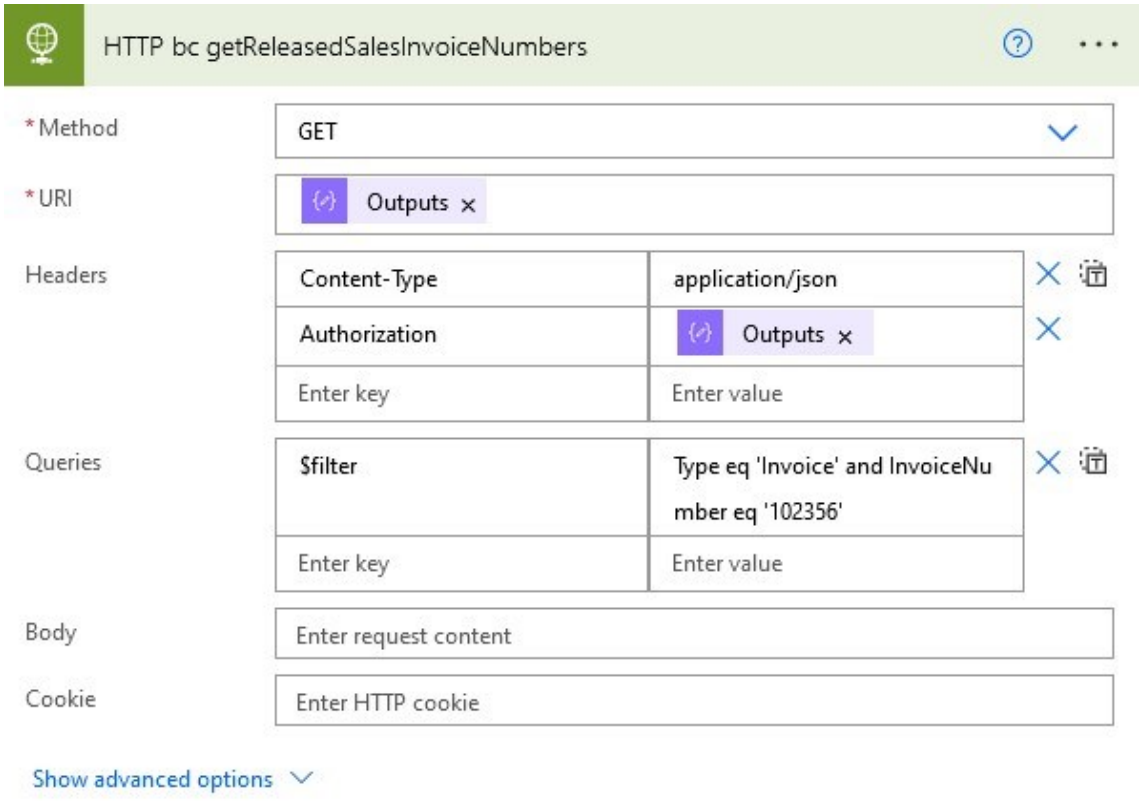
Lasku 3:n myyntilaskuun syötettiin kaikki tarvittavat tiedot, mutta laskutietojen ovt-operaattorin arvoksi laitettiin *INVALIDOPERATOR*. Tästä odotettiin saavan virheilmoitus automaation toisessa työnkulussa eli, Maventan tarkastamisprosessissa, jossa Maventa asettaisi verkkolaskun tilaksi *FAILED*. Laskun lisätietokenttään laitettiin teksti ”TEST INVALID OVTOPERATOR”.



Kuva 24. Kuvakaappaus Business Centralin *Myyntilaskut* näkymästä. Näkymä on suodatettu näyttämään vain testausta varten luodut myyntilaskut.

5.2 Myyntilaskujen vienti verkkolaskuoperaattorille

Selkeyden vuoksi, automaation työnkulun myyntilaskunumeroiden hakuun lisättiin suodatinarvo *InvoiceNumber eq '{laskunumero}'*, joka haki testilaskuista vain yhden myyntilaskun kerrallaan (kuva 25). Tällä tavalla pystyttiin tarkastelemaan jokaisen testilaskun elinkaari mahdollisimman tarkasti.



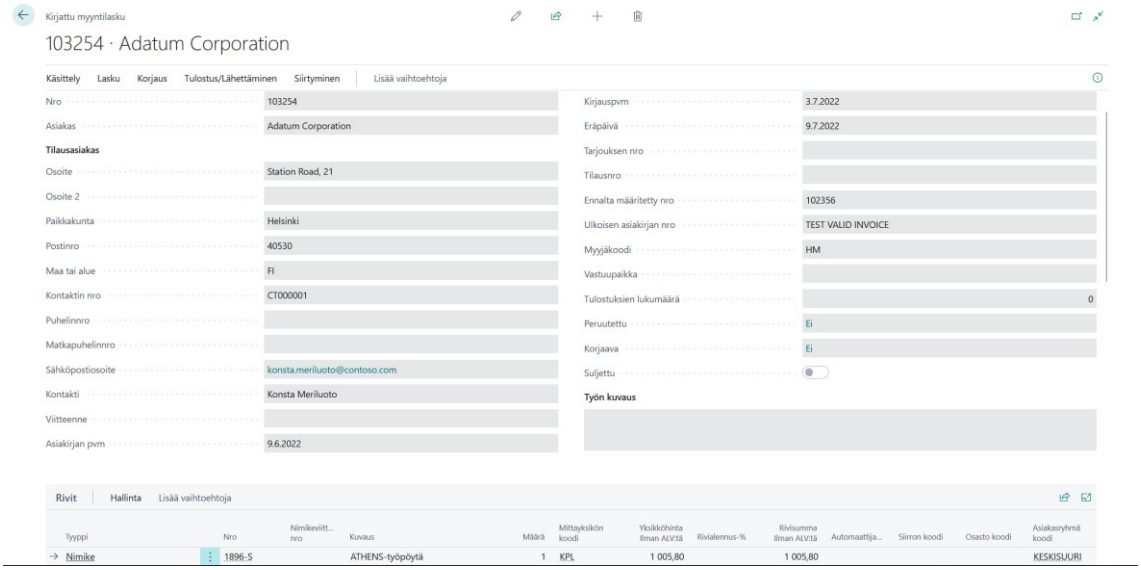
Kuva 25. Kuvakaappaus Power Automaten käyttöliittymästä. Laskunumerohakuun on lisätty filtteri, joka hakee vain yhden laskun kerrallaan.

5.2.1 Myyntilasku 1

Automaatio laitettiin hakemaan ensimmäiseksi laskunumerolla 102356 ollut myyntilasku ja viemään se Maventaan. Power Automaten käyttöliittymästä pystyttiin todentamaan, että automaation ensimmäinen työnkulku oli onnistunut (kuva 26) ja myyntilasku oli läpäissyt kaksi ensimmäistä laskuvalidointia.

Kuva 26. Kuvakaappaus Power Automaten käyttöliittymästä. Lasku 1:n lähetys verkkolaskuoperaattorille oli onnistunut.

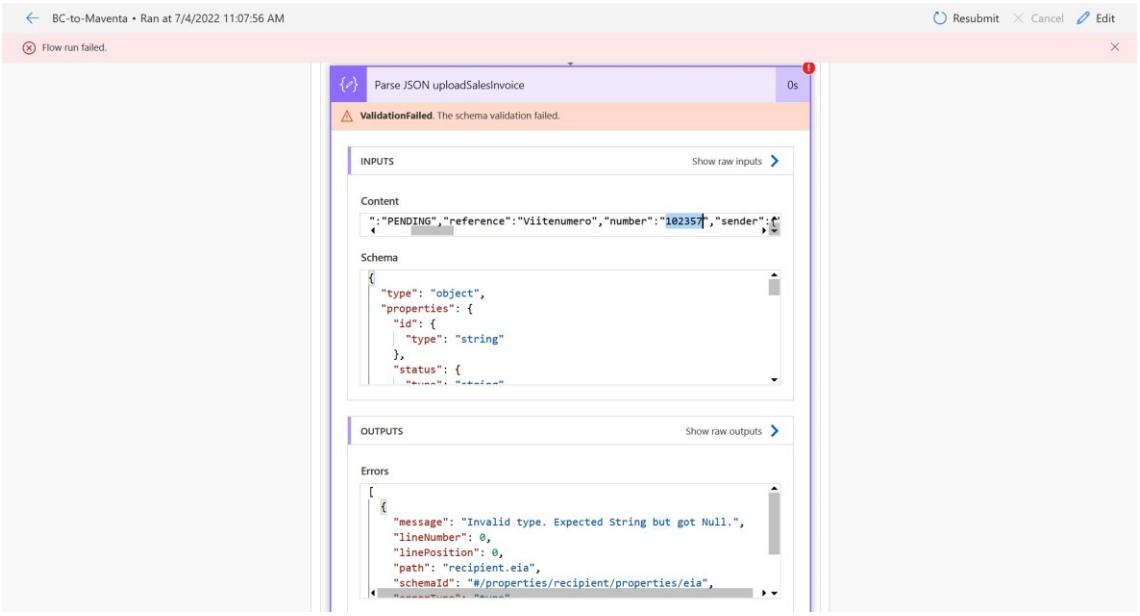
Lasku 1:n viimeinen validointi tehtiin suorittamalla automaation toinen työnkulku. Onnistunut työnkulku kävi kirjaamassa myyntilaskun Business Centraliin laskutetuksi. Tämä kirjaustapahtuma oli tarkastettavissa Business Centralin *Kirjatut myyntilaskut* näkymästä (kuva 27).



Kuva 27. Kuvakaappaus Business Centralin Kirjatut myyntilaskut näkymästä. Tekstikentässä *Ulkoisen asiakirjan nro* näkyy sama tekstikuvaus, kuin lasku 1:n kuvauksessa sekä tiedoista löytyy myös sama laskunumero.

5.2.2 Myyntilasku 2

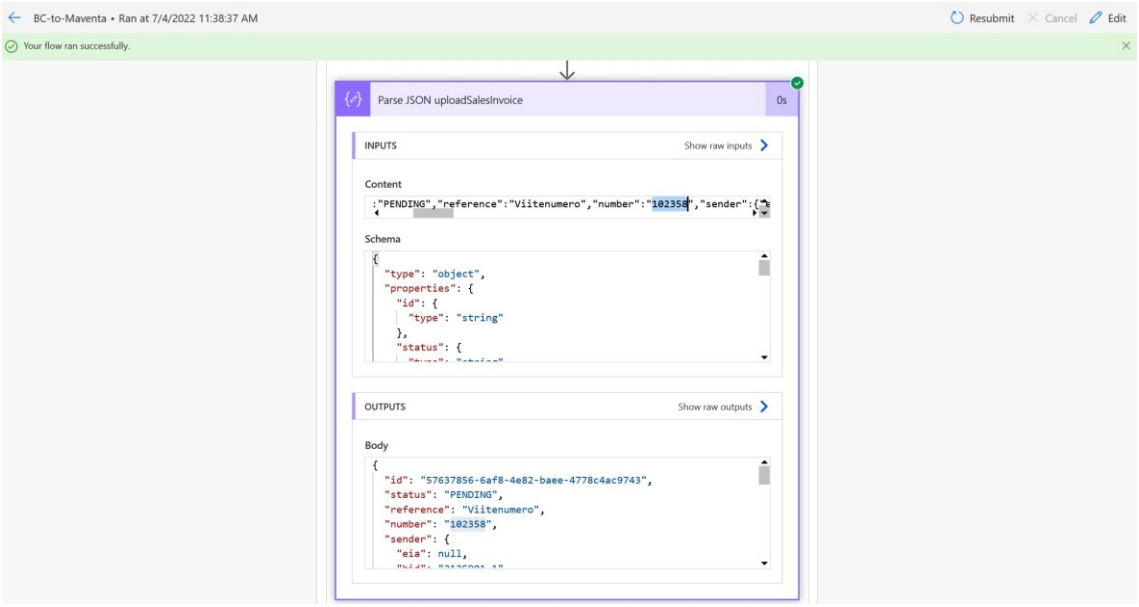
Automaatio laitettiin hakemaan seuravaksi laskunumerolla 102357 oleva myyntilasku 2, jonka vienti verkkolaskuoperaattorille odotettiin epäonnistuvan, laskutiedoista puuttuneen ovt-tunnuksen takia. Automaation odotettiin saavan virheilmoitus Maventan palvelusta, kun palveluun yritettäisiin lähettää verkkolasku, josta ei löytyisi verkkolaskustandardissa pakollisiksi määritettyjä kenttiä. Maventan validointipalvelun palauttamista tiedoista odotettiin löytyvän virheviesti, jonka automaatio tulkitsisi epäonnistuneeksi työnkuluksi (kuva 28).



Kuva 28. Kuvakaappaus Power Automaten työnkulun lokitiedoista. Alhaalla oleva virheviesti on Maventan validointipalvelusta.

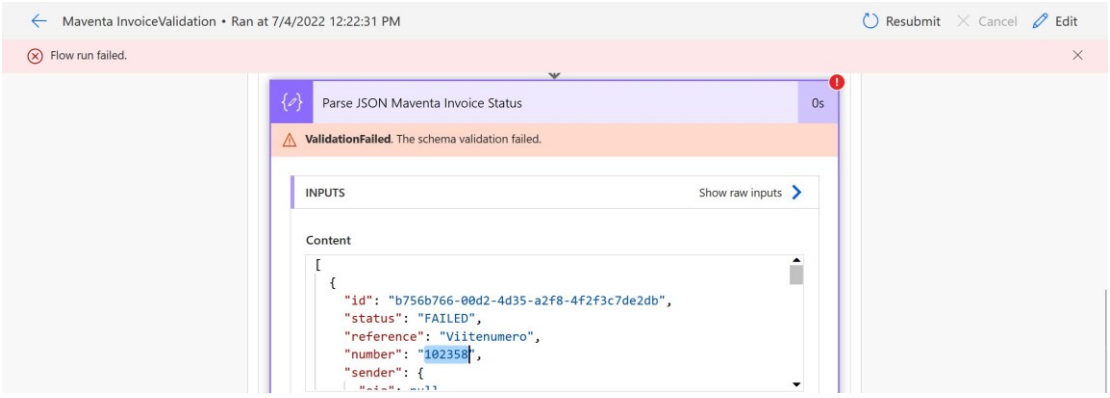
5.2.3 Myyntilasku 3

Testauksen viimeisessä osiossa automaatio haki laskunumerolla 102358 olevan myyntilasku 3:n ja lähetti sen Maventaan. Tämän laskun odotettiin menevän onnistuneesti läpi automaation ensimmäisessä työnkulussa (kuva 29).



Kuva 29. Kuvakaappaus Power Automaten työnkulun lokitiedoista.

Virheilmoitusta odotettiin vasta automaation toisessa työnkulussa, jossa Maventan validointipalvelu palauttaisi viestin epäonnistuneesta laskun toimituksesta. Tämä virheilmoitus näkyisi toisen työnkulun loppuvaiheessa, jossa tarkistettiin Maventaan lähetetyn verkkolaskun tila (kuva 30).



Kuva 30. Kuvakaappaus Power Automaten työnkulun lokitiedoista. Laskun lähetys on epäonnistunut viimeisessä validointivaiheessa puutteellisten tietojen takia.

6 Yhteenveto

Tämän työn tarkoituksena oli monipuolistaa Systencess Oy:n tarjoaman toiminnanohjausjärjestelmä Business Centralin laskutusvaihtoehtoja.

Työssä tutkittiin myös vaihtoehtoisia tapoja saada myyntilaskut vietyä toiminnanohjausjärjestelmästä asiakkaille. Business Centraliin tuli tämän työn alkuvaiheessa päivitys, joka mahdollisti integraation kolmannen osapuolen palvelun kautta. Integraation avulla Peppol BIS verkkolaskuja olisi voitu toimittaa Business Centralista suoraan asiakkaille, mutta se osoittautui taloudellisesti epäkannattavaksi nykyisiä ja tulevia asiakkaita ajatellen. Vaihtoehtoinen integraatio tuki myös vain Peppol BIS 2.1 verkkolaskustandardia, joten sen kautta ei olisi voitu lähettää uudempaa Peppol BIS 3 verkkolaskua. Jo aloitetun integraation tekeminen sai siten jatkua.

Työn aikana kehitettyyn automaation liitettiin myös mahdollisuus lähettää Finvoice 3.0 laskustandardin mukaisia verkkolaskuja. Tällä lisäyksellä pyrittiin edistämään jo ennestään kehitettyjä Systencess Oy:n integraatioväyliä ja saada ne toimimaan monipuolisesti yhteen Business Centralin kanssa.

Työn suurin haaste sijoittui projektin keskivaiheille, kun Business Centralista haetut tiedot oli muunnettava XSLT:n avulla verkkolaskustandardissa määritettyyn muotoon. XSLT-kielen syntaksin opettelu oli eniten aikaa vievä vaihe työssä. XSLT:n opetteluun käytetty aika mahdollisti kuitenkin edellämainitun Finvoice laskutusmahdollisuuden liittämisen integraatioon erittäin vaivattomasti.

Työssä tehty automaatio suunniteltiin ja ohjelmoitiin hyvin dynaamiseksi, minkä avulla mahdollistettiin sekä nykyisille että tuleville asiakkaille laskutusmahdollisuus Maventan kautta. Tehtyä automaatiota voi tulevaisuudessa vielä jatkokehittää, ohjelmoimalla ja liittämällä pilvifunktioon lisää XSLT-muunnoksia. Näiden muunnosten avulla automaatio pystyisi luomaan nykyisten Peppol BIS 3.0 ja Finvoice 3.0 verkkolaskustandardien

lisäksi myös muiden verkkolaskustandardien mukaisia verkkolaskuja ja lähettämään ne Maventan palveluun.

Lähteet

1 Das, Narottam, Evolution of Microsoft Dynamics NAV - An Infographic. 2018. Verkkoaineisto. <https://www.appseconnect.com/evolution-of-microsoft-dynamics-nav-infographic/>. Luettu 26.5.2022.

2 Carina Sørensen & Michael Nielsen. 2015. The Story of Navision and Navision. Verkkoaineisto. <https://link.springer.com/chapter/10.1007/978-3-319-17145-6_15>. Luettu 26.5.2022.

3 Covens, Dirk. 2020. 365 MYWAY. Verkkoaineisto. <https://www.365myway.com/en/blog/navision>. Luettu 26.5.2022.

4 Navision to Dynamics 365 Business Central. Verkkoainesto. Triangle. <https://www.triangle.es/en/navision-to-dynamics-365-business-central/>. Luettu 26.5.2022.

5 Microsoft Acquires Navision. 2002. Microsoft. Verkkoaineisto. <https://news.microsoft.com/2002/07/11/microsoft-acquires-navision/>. Luettu 28.5.2022.

6 Dynamics NAV Versions List. 2021. Verkkoaineisto. Navision Planet. <https://www.navisionplanet.com/dynamics-nav-versions-list/>. Luettu 1.7.2022.

7 Dynamics NAV 2018. Verkkoaineisto. Microsoft. <https://docs.microsoft.com/fi-fi/lifecycle/products/dynamics-nav-2018>. Luettu 22.6.2022.

8 What integrations allows Microsoft 365 Business Central with Office 365. Verkkoaineisto. Triangle. <https://www.triangle.es/en/what-integrations-allows-microsoft-365-business-central-with-office-365>. Luettu 23.6.2022.

9 Business Central. Verkkoaineisto. TEKenable. <https://tekenable.ie/microsoft-platform/microsoft-business-solutions-dynamics365/business-central/>. Luettu 5.6.2022

10 History of OpenPeppol. Verkkoaineisto. OpenPeppol. <https://peppol.eu/about-openpeppol/history-of-openpeppol/>. Luettu 14.6.2022.

11 What is OpenPeppol. Verkkoaineisto. OpenPeppol. <https://peppol.eu/about-openpeppol/what-is-openpeppol/>. Luettu 14.6.2022.

12 What is Peppol. Verkkoaineisto.<https://peppol.eu/what-is-peppol/>. Luettu 14.6.2022.

13 Peppol is a major European success story leveraging international best practice for digital business interoperability. Verkkoaineisto. Celtrino. <https://www.celtrino.com/peppol/>. Luettu 14.6.2022.

14 Peppol Reach and Country Profiles. Verkkoaineisto. <https://peppol.eu/what-is-peppol/peppol-country-profiles/>. Luettu 15.6.2022.

15 About OpenPeppol. Verkkoaineisto. <https://peppol.eu/about-openpeppol/>. Luettu 15.6.2022.

16 Zachariasen, Dennis. What is PEPPOL? The 6 most important things to know. Verkkoaineisto.<https://peppol.com/blog/what-is-peppol/>. Luettu 15.6.2022.

17 EUR-Lex - 32014L0055 – EN. 2014. Verkkoaineisto. EUR-Lex. <https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32014L0055>. Luettu 15.6.2022.

18 EN 16931 compliance. 2018. Verkkoaineisto. European Commission. <https://ec.europa.eu/digital-building-blocks/wikis/display/DIGITAL/EN+16931+compliance>. Luettu 15.6.2022.

19 Migration Plan. 2022. Verkkoaineisto. OpenPeppol. <https://docs.peppol.eu/poacc/upgrade-3/migration/>. Luettu 16.6.2022.

20 Microsoft documents. 2022. Verkkoaineisto. Microsoft. <https://docs.microsoft.com/fi-fi/power-automate/getting-started>. Luettu 16.6.2022.

21 Pilvityönkulun luominen Power Automatessa. 2022. Verkkoaineisto. Microsoft. <https://docs.microsoft.com/fi-fi/power-automate/get-started-logic-flow>. Luettu 16.6.2022.

22 Azure products. 2022. Verkkoaineisto. Microsoft. <https://azure.microsoft.com/en-gb/services/>. Luettu 16.6.2022.

23 Ardal, Thomas. 2017. An introduction to Azure Functions and why we migrate. 2022. Verkkoaineisto. <https://blog.elmah.io/migrating-from-windows-services-to-azure-functions/>. Luettu 17.6.2022.

24 Supported languages in Azure Functions. 2021. Verkkoaineisto. Microsoft. <https://docs.microsoft.com/en-us/azure/azure-functions/supported-languages>. Luettu 17.6.2022.

25 XML introduction. 2022. Verkkoaineisto. MDN Contributors. <https://developer.mozilla.org/en-US/docs/Web/XML/XML_introduction>. Luettu 18.6.2022.

26 Martin, Matthew. 2022. XML vs HTML | What is Difference between XML and HTML. Verkkoaineisto. <https://www.guru99.com/xml-vs-html-difference.html>. Luettu 18.6.2022.

27 Peppol BIS Billing. Verkkoaineisto. OpenPeppol. <https://docs.peppol.eu/poacc/billing/3.0/bis/>. Luettu 18.6.2022.

28 Top 4 software development methodologies. 2017. Verkkoaineisto. Synopsys Editorial Team.<https://www.synopsys.com/blogs/software-security/top-4-software-development-methodologies/>. Luettu 19.6.2022.

29 What is Postman. Verkkoaineisto. Postman. Available: <https://www.postman.com/>. Luettu. 29.5.2022.

30 Permissions Property. 2022. Verkkoaineisto.Microsoft. <https://docs.microsoft.com/en-us/dynamics365/business-central/dev-itpro/developer/properties/devenv-permissions-property>. Luettu 26.5.2022.

31 Introduction. 2021. Verkkoaineisto. Postman. <https://learning.postman.com/docs/getting-started/introduction/>. Luettu 18.6.2022.

32 XSLT processor for Java. 2003. Verkkoaineisto. <https://docs.oracle.com/cd/A97630_01/appdev.920/a96621/adx05xsj.htm>. Luettu 7.6.2022.

33 Gerrit's History. 2022. Verkkoaineisto. <https://www.gerritcodereview.com/about.html>. Luettu 20.6.2022.

34 Working with Gerrit: An example. 2022. Verkkoaineisto. <https://gerrit-review.googlesource.com/Documentation/intro-gerrit-walkthrough.html>. Luettu 20.6.2022.

35 Object Ranges in Business Central. 2022. Verkkoaineisto. <https://docs.microsoft.com/en-us/dynamics365/business-central/dev-itpro/developer/devenv-object-ranges>. Luettu 26.6.2022.

36 ObjectType Option Type. 2022. Verkkoaineisto. <https://docs.microsoft.com/en-us/dynamics365/business-central/dev-itpro/developer/methods-auto/objecttype/objecttype-option>. Luettu 30.6.2022.