Tampereen ammattikorkeakoulu

Odoon integrointi taloushallintojärjestelmä Visma Fivaldiin

Jani Niittymäki

OPINNÄYTETYÖ
Huhtikuu 2022

Tieto- ja viestintätekniikka
Ohjelmistotekniikka

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tieto- ja viestintätekniikka
Ohjelmistotekniikka

NIITTYMÄKI, JANI:
Odoon integrointi taloushallintojärjestelmä Visma Fivaldiin

Opinnäytetyö 33 sivua, joista liitteitä 0 sivua
Huhtikuu 2022

Collapick Company Oy:n teollisuuteen optimoitu toiminnanohjausjärjestelmä Tempo laajentaa avoimen lähdekoodin toiminnanohjausjärjestelmä Odoota lisäämällä siihen käyttöä tehostavia ominaisuuksia, kuten integraatioita eri järjestelmiin, automaatioita, erilaisia helppokäyttöliittymiä sekä visualisointeja. Opinnäytetyön tarkoituksena oli toteuttaa integraatioita Odoosta Visma Fivaldiin.

Laajoilla ohjelmistointegraatioilla saadaan tarjottua valmiimpia kokonaisuuksia asiakkaille, koska useilla yrityksillä on käytössään jokin erillinen taloushallintojärjestelmä. Lisäksi Odooseen siirtymisestä tulee pienempi prosessi, koska ei tarvitse välttämättä vaihtaa olemassa olevaa järjestelmää tai opetella uutta. Integraatioiden toteuttamiseen käytettiin hyödyksi Fivaldin tarjoamaa Swagger-dokumentaatiota, josta nähtiin rajapinnan palauttama data sekä luontikutsuissa vaaditut kentät. Opinnäytetyön toteuttaminen lähti liikkeelle Fivaldin rajapinnan tarjoamien toimintojen selvittämisellä rajapintadokumentaatiota hyödyntäen, minkä jälkeen määriteltiin integraatioon toteutettavat toiminnot.

Opinnäytetyössä esitellään Odoota yleisellä tasolla, Odoo-moduulin luontia sekä esitellään Fivaldi-integraation toteutusta ja toimintaa. Opinnäytetyön tuloksena syntyi erillinen Fivaldi-moduuli, jossa ominaisuuksina ovat ostolaskujen tuonti sekä myyntitilausten tuonti ja vienti.

Asiasanat: integraatio, taloushallintojärjestelmä, odoo, collapick company, visma fivaldi

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Information and communication technology
Software development

NIITTYMÄKI, JANI:
Integrating Odoo to Financial Management System Visma Fivaldi

Bachelor's thesis 33 pages, appendices 0 pages
April 2022

Collapick Company Oy’s industry optimized enterprise resource planning system Tempo extends the open-source ERP Odoo by adding features that improve usage, such as integrations to different systems, automations, ease of use interfaces and visualizations. The purpose of this thesis was to examine and implement integrations from Odoo to Visma Fivaldi.

Broad software integrations allow companies to offer more complete packages to customers, as many companies have a separate financial management system in place already. In addition, the transition to Odoo becomes a smaller process because it is not necessary to change the existing system. The integrations were developed by the help of Swagger documentation provided by Fivaldi, which displayed the data returned in different endpoints as well as the required data for data creation endpoints. In the initial stage of this study, the available features in Fivaldi-API were examined using the provided documentation, after which the features for the integration were defined.

In this thesis, Odoo is presented on a general level, information about creating a Odoo-module is provided and implementation and operation of the Fivaldi-integration is presented. As a result of the thesis a separate Fivaldi module was created with features to import purchase invoices and import and export sales orders.

Key words: integration, finance management software, odoo, collapick company, visma fivaldi

SISÄLLYS

KÄSITTEET, LYHENTEET JA TERMIT

ERP	Enterprice resource planning, toiminnanohjausjärjestelmä
Finvoice	Suomessa standardoitu XML-muotoinen e-laskutusformaatti
LF	Line feed, uuden rivin erotin (Unicode U+000A)
MAC	Message Authentication Code, rajapinta-autentikointiin käytetty koodi
MRP	Material Requirements Planning, materiaalitarvelaskenta
Myyntilasku	Lasku, jonka tuotteen tai palvelun myyjä lähettää asiakkaalleen
Ostolasku	Lasku, joka maksetaan tuotteen tai palvelun ostamisesta
Visma Fivaldi	Norjalaisen Visma-konsernin taloushallintojärjestelmä
Python	Tulkattava ja skriptattava ohjelmointikieli
Postman	Ohjelmisto, jota voidaan käyttää rajapintojen testaamiseen
XML	Extensible Markup Language, merkintäkielien standardi
JSON	Javascript Object Notation, tekstipohjainen kieli datan esittämiseen

1 JOHDANTO

Tässä opinnäytetyössä on tarkoitus tehdä integraatioita Odoosta ulkoiseen taloushallintojärjestelmään, Visma Fivaldiin. Motivaationa tämän järjestelmän valintaan integraation kohteeksi oli kyselyt sekä tarpeet asiakkailta. Visma Fivaldin ja Odoon välille toteutetaan ostolaskujen tuonti sekä myyntitilausten tuonti ja vienti. Visma Fivaldi ei käytä standardoitua Finvoice-xml formaattia rajapinnan laskutietojen välityksessä, vaan omaa json-muotoista formaattia (Visma Fivaldi, n.d.a). Tämä json-muotoinen formaatti parsitaan tarvittavaan muotoon, jotta Fivaldin tiedot saadaan näkyviin Odoon käyttöliittymiin.

Toteutettavan integraation avulla saadaan vähennettyä manuaalista työtä asiakkailta varsinkin myyntitilausten osalta, koska integraation ansiosta ei tarvitse käydä itse lisäämässä Odoossa luotuja myyntitilauksia Fivaldiin. Lisäksi saapuvat ostolaskut saadaan linkitettyä järkevästi Odoossa tehdyille ostoille, jolloin ostojen kokonaiskuvan näkee yhdestä järjestelmästä.

Opinnäytetyö toteutetaan Collapick Company Oy:lle osaksi heidän Odoo-kokonaisuuttaan tarjoamaan lisäarvoa asiakkaille. Collapick Company Oy on Tampereella ja Joensuussa toimiva ohjelmistoyritys, joka työllistää tällä hetkellä 12 henkilöä. Collapickin pääasiakaskunta koostuu teollisuusyrityksistä ja päätuotteena Collapick tarjoaakin teollisuuteen suunnattua toiminnanohjausjärjestelmä Tempoa, joka laajentaa Odoo-ERP järjestelmän ominaisuuksia. Tämän lisäksi Collapick tarjoaa sovelluskehitystä sekä konsultointia. Collapickin tavoite on vähentää manuaalista työtä ja lisätä työn tuottavuutta, tehokkuutta sekä luoda säästöjä asiakkaille (Collapick Company, n.d).

Opinnäytetyön aikana toteutetaan koodiosuus ja varmistetaan sen toimivuus. Raportissa käydään läpi käytettyjä teknologioita sekä Odoon arkkitehtuuria ja rakennetta. Lisäksi kerrotaan tehdystä koodaustyöstä, Fivaldi-integraation toiminnasta sekä datan oikeaan muotoon muuntamisesta. Lopuksi pohditaan kokonaisuutta ja mahdollisia ilmenneitä ongelmia.

2 ODOO-ERP

Odoo on vuonna 2005 julkaistu belgialainen avoimen lähdekoodin toiminnanohjausjärjestelmä. Odoo tunnettiin aikaisemmin nimillä TinyEPR sekä OpenERP, mutta vuodesta 2014 alkaen ohjelmisto sekä yritys on ollut Odoo-nimellä. (Odoo, n.d.e). Nykyään Odoota kehittää maailmanlaajuinen lähes kahden tuhannen työntekijän Odoo S.A (Odoo, n.d.a).

2.1 Odoo ekosysteemi

Odoo sisältää perinteiset toiminnanohjausjärjestelmän ominaisuudet, kuten hallintatoiminnot tuotantoon, jakeluun, varastoon, laskutukseen ja kirjanpitoon (Odoo, n.d.b). Lisäksi Odoo pohjautuu suuresti yksityishenkilöiden ja yritysten tekemiin moduuleihin, joita Odoon sovelluskaupasta on ladattavissa tuhansittain (Odoo, n.d.h). Tämä on yksi Odoon suuri etu verrattuna muihin ERP-järjestelmiin – toiminnallisuuden laajentaminen ja kehittäminen ei ole järjestelmän kehittäjästä välttämättä kiinni, vaan jokainen voi laajentaa omaa Odoo ympäristöään kolmannen osapuolen moduuleilla halutessaan. Hyvänä vertauskuvana voisi olla WordPress, joka kasvoi suureksi palveluntarjoajaksi varsinkin suuren lisäosavalikoiman ja muokattavuutensa ansioista (Plesk, 2018).

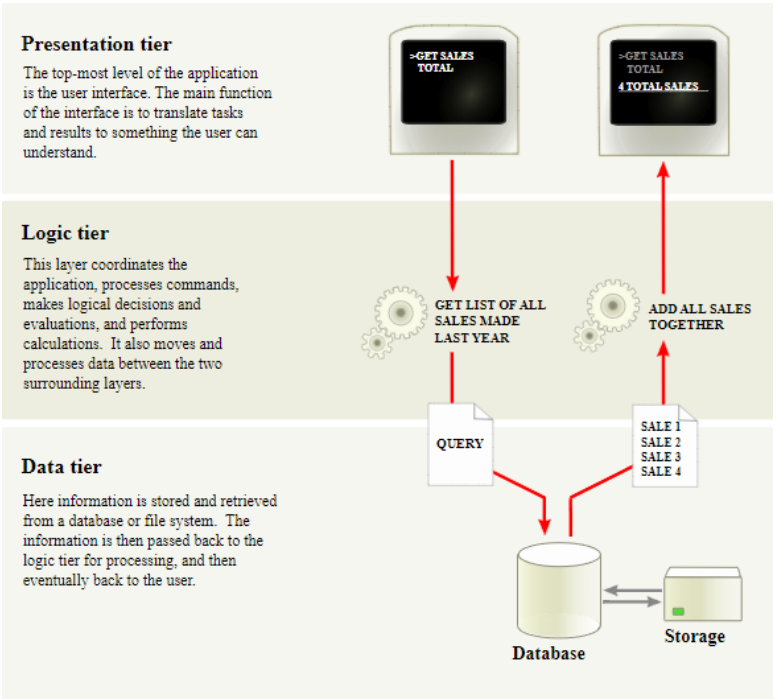
Odoosta on kaksi versiota: Enterprise, joka vaatii erillisen maksullisen lisenssin ja Community, joka on maksuton ja täysin avointa lähdekoodia. Näistä Community on suppeampi ja tarjoaakin vain Odoon ydintoiminnallisuudet. Enterprise-versio tuo lisäominaisuuksia esimerkiksi kirjanpito-, MRP- sekä varastomoduuleihin ja uusina ominaisuuksina esimerkiksi kenttäpalvelu-, laadunvalvontamoduulin sekä mobiilisovelluksen (Odoo, n.d.b). Yrityksille Odoon hankkimiseen on vaihtoehtoina tehdä se joko itsepalveluna, Odoon ”Success Pack” palvelun avulla, jolloin yrityksen Odoo-tukea tarjoaa Odoon palkkalistoilla oleva työntekijä. Lisäksi vaihtoehtona on suuremmille, yli 50-työntekijän yrityksille Odoon hankkiminen kumppanin kautta, koska kumppanit voivat tarjota yksilöllisempää palvelua, tukea sekä kehitystä (Odoo, n.d.f). Suomessa Odoolla on 7 kumppania, joista Collapick on yksi (Odoo, n.d.d).

2.2 Tekniikka

Odoon ohjelmointikielenä on pääasiassa Python. Javascriptia voidaan hyödyntää pienten käyttöliittymälisäosien ja -mukautuksien tekemiseen. Käyttöliittymät tehdään kuitenkin yleensä XML-tiedostojen avulla (Odoo, n.d.c). Lisäksi erilaisten raporttien tekemiseen on mahdollista käyttää Odoon QWeb-template moottoria. QWeb-raporteissa käytetään myös XML-merkintäkieltä. (Odoo, n.d.g)

2.2.1 Arkkitehtuuri

Odoo noudattaa kolmiosaista MVC-arkkitehtuuria, johon kuuluu esittelytaso, logiikkataso sekä tietokantataso. (KUVA 1). Esittelytason koodi koostuu pääasiassa Odoon XML-datatiedostojen avulla tehdystä käyttöliittymästä, logiikkataso Python luokista ja kontrollereista ja tietokantataso PostgreSQL-tietokannasta. Logiikkatason ja datatason välinen viestintä tapahtuu ORM-tason välityksellä eli Python luokista tehdään tietokantataulut suoraan ja kyselyt tapahtuvat ORM-funktioiden avulla (Odoo, n.d.c). Tämän ansiosta raaka-SQL kyselyitä ei ole välttämättä tarve käyttää Odoo kehityksessä. Uusien tietokantamallien tekeminen nopeutuu huomattavasti, kun Odoo tekee tarvittavat mallien kentät sekä relaatiot automaattisesti Python-tiedoston kentistä.



KUVA 1. MVC-arkkitehtuuri (Odoo, n.d.c)

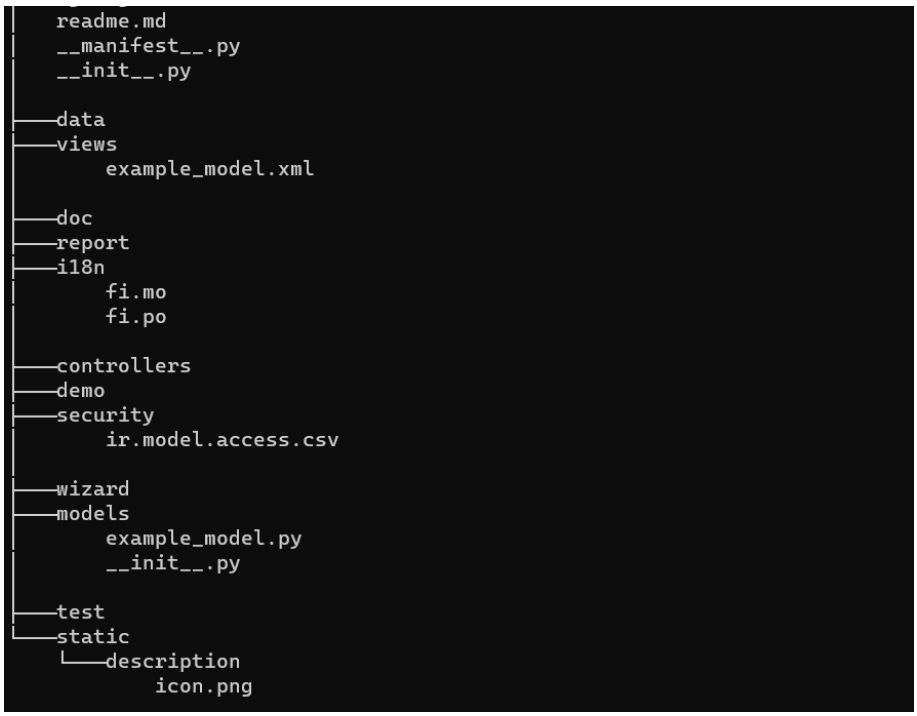
2.2.2 Moduulien luonti

Odoo moduulin rakenne koostuu karkeasti XML-käyttöliittymätiedostoista, jotka ovat views kansiossa sekä Python model-tiedostoista, jotka ovat models-kansiossa (KUVA 2). Näiden lisäksi pakollisia moduulin rakenteeseen kuuluvia asioita ovat __manifest__.py, sekä __init__.py tiedostot.

__manifest__.py tiedostolla määritellään mm. Odoon App Storea varten tiedot ja sillä tehdään tarvittavat tiedostoimportit sekä moduuliriippuvuudet. __init__.py tiedostoilla tehdään tarvittavat Python model-tiedosto importit. Muita kansiorakenteessa (KUVA 2) listattuja kansioita ja tiedostoja ei tarvitse olla, jotta moduuli toimii, vaan niitä käytetään vain tarpeen vaatiessa. Näitä on listattu alla olevassa taulukossa (TAULUKKO 1).

TAULUKKO 1. Selitykset moduulin kansiorakenteesta (Speedy Sense, 2019)

Kansio	**Tarkoitus**
data	Sisältää kovakoodatun XML-tiedostodatan.
views	Sisältää XML-käyttöliittymätiedostot
doc	Sisältää moduulin dokumentaatiotiedostot
report	Sisältää raporttinäkymien tiedostot
i18n	Sisältää käännöstiedostot
controllers	Sisältää http-endpoint tiedostot
demo	Vastaa data-kansiota, mutta käytetään vain, jos demo-data asetus on käytössä
security	Sisältää moduulien näkymätiedostojen käyttöoikeus csv-tiedostot
wizard	Sisältää popup-ikkuna näkymien tiedostot
models	Sisältää Python-model kooditiedostot
test	Sisältää Python-testit
static	Sisältää erilaisia kuvatiedostoja tai muuta staattista sisältöä käyttöliittymää varten



KUVA 2. Odoo moduulin esimerkkikansiorakenne

2.2.3 Esimerkkimoduuli

Yksinkertaisimmillaan toiminnallisuutta lisäävän moduulin saa luotua perimällä jotain Odoon luokkaa. Tämän esimerkin tapauksessa peritään res_company-luokkaa, jota Odoossa käytetään yrityksen tietojen tallentamiseen. Models kansioon luodaan Python-luokka, joka perii edellä mainittua Odoon moduulia ja lisää yhden uuden kentän sille (KUVA 3). Moduulin asennuksen yhteydessä odoo lisää res_company luokalle tiedostossa lisätyn char-kentän company_founded_year, (KUVA 3, rivi 10). Kenttä saadaan käyttöliittymään muokattavaksi lisäämällä uuden XML-tiedoston views kansioon (KUVA 4). Kuvan XML-tiedosto laajentaa Odoon yritysasetus -näkymää ja lisää siihen uuden välilehden (KUVA 4, rivi 13) ja tälle välilehdelle tekstikentän (KUVA 4, rivi 17) sekä napin (KUVA 4, rivi 18).

Field- sekä button-kentille on määritelty name-parametri, jonka pitää viitata kyseisen python luokan kenttiin. Moduuli ei asennu, jos kenttäviitteitä ei löydy. Napin tapauksessa name-kentässä on määritelty onclick-metodi, ja field-kentälle muokattava tietokantakenttä.

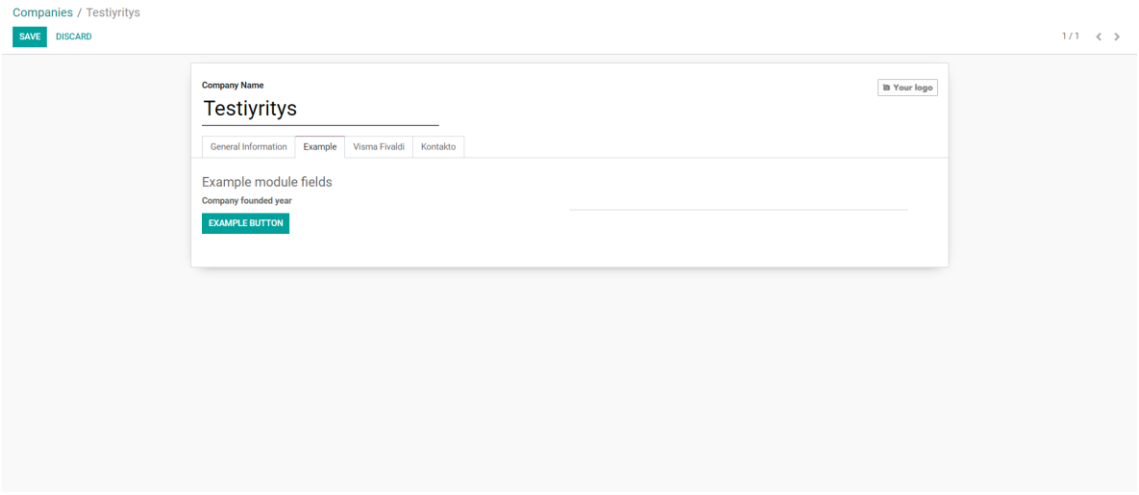
```python
from odoo import api, fields, models, _
from odoo.exceptions import ValidationError
import logging
_logger = logging.getLogger(__name__)

class ResCompany(models.Model):
    # Peritään Odoon res.company luokka
    _inherit = 'res.company'
    # Odoo luo tästä kentän tietokantaan automaattisesti - ORM mallin hyöty
    company_founded_year = fields.Char(string='Company founded year')


    # Funktio, jota kutsutaan käyttöliittymästä nappia painamalla
    def action_example_button_click(self):
        message = "No year given"
        if self.company_founded_year:
            message= "Company founded year: "+self.company_founded_year

        # Palautetaan toiminto, joka näyttää ilmoituksen käyttäjälle. Odoon toiminnallisuutta.
        return {
            'type': 'ir.actions.client',
            'tag': 'display_notification',
            'params': {
                'title': "Button click",
                'message':message,
            }
        }

```

KUVA 3. res_company.py tiedosto esimerkkimoduulille

```xml
<?xml version="1.0" encoding="utf-8"?>
<odoo>
    <!-- Vaaditut kentät perintää varten -->
    <record id="view_company_form_inherit_example_module" model="ir.ui.view">
        <!-- Näillä kentillä määritellään perittävä luokka sekä karkea sijainti, mihin halutaan lisätä -->
        <field name="name">view.company.form.inherit.example.module</field>
        <field name="model">res.company</field>
        <field name="inherit_id" ref="base.view_company_form"/>
        <field name="arch" type="xml">
        <!-- xpath lauseen avulla määritellään paikka lisättävälle elementille -->
            <xpath expr="//page[@name='general_info']" position="after">
            <!-- Luo "Example" nimisen valikon, jossa tekstikenttä sekä nappi -->
                <page string="Example" name="example_module_page">
                    <group string="Example module fields" name="example_module_fields">
                        <!-- "name" kentän pitää viitata res.company luokan muuttujiin/funktioihin,
                            jos mitään ei löydy name-kentällä, niin asennus kaatuu -->
                        <field name="company_founded_year" />
                        <button name="action_example_button_click" class="oe_highlight" string="Example button" type="object"/>
                    </group>
                </page>
            </xpath>

        </field>

    </record>

</odoo>
```

KUVA 4. res_company.xml tiedosto esimerkkimoduulilla

Käytännössä tällä koodimäärällä sekä __manifest.py__ tiedoston kenttien, kuten nimen, kuvauksen sekä riippuvuussuhteiden määrittämisellä saadaan Odoon sovelluskauppaan näkyviin moduuli, jonka voi asentaa. Asennuksen jälkeen XML-koodissa luotu käyttöliittymälisäys ilmestyy yrityksen asetussivulle (KUVA 5).



KUVA 5. Odoo yrityksien asetussivu, luotu valikko

3 VISMA FIVALDI -INTEGRAATIO

Visma Fivaldi on Norjalaisen Visma-konsernin tuottama taloushallintojärjestelmä. Fivaldi tarjoaa taloushallinto-ohjelmiston tärkeimmät ominaisuudet, kuten laskutuksen, tilauskäsittelyn, varastonhallinnan, raportoinnin sekä muita lisäominaisuuksia (Visma, n.d). Käytössä oleva REST-rajapinta ei tarjoa kaikkia Fivaldin ominaisuuksia käyttöön, vaan sen kautta saa haettua ja luotua ostolaskut, kirjanpitokirjaukset sekä myyntitilaukset. Myyntitilauksia varten on auki tuoterekisteri sekä asiakasrekisteri, joista voidaan hakea sekä luoda tarvittaessa kirjauksia. Lisäksi rajapinta tarjoaa erilaista kiinteistöhallintaan liittyvää dataa (Visma Fivaldi, n.d.a). Integraatioiden toteuttamiseen hyödynnettiin Fivaldin tarjoamaa Swagger UI -dokumentaatiota, jonka etuja ovat tekstipohjaiseen dokumentaatioon verrattuna mm. visuaalisuus sekä automaattisesti backend-koodin pohjalta päivittyvä rajapintakuvaus (Swagger, 2021).

Tämän integraation kannalta kuitenkin oleellisimmat ovat ostolasku- sekä myyntitilausrajapinta ja sitä varten tarvittavat asiakas- ja tuoterekisteri. Lisäksi käytössä on ostolaskun liitetiedostojen noutamista varten arkistorajapinta. Yleisesti myyntitilaukset ovat kokonaisuuksia, jotka sisältävät johonkin kauppaan liittyvät tuotteet tai palvelut. Myyntitilauksesta luodaan myyntilasku, joka lähetetään asiakkaalle maksettavaksi. Ostolaskut ovat laskuja, jotka ovat muodostuneet jostain yrityksen ostamasta tuotteesta tai palvelusta.

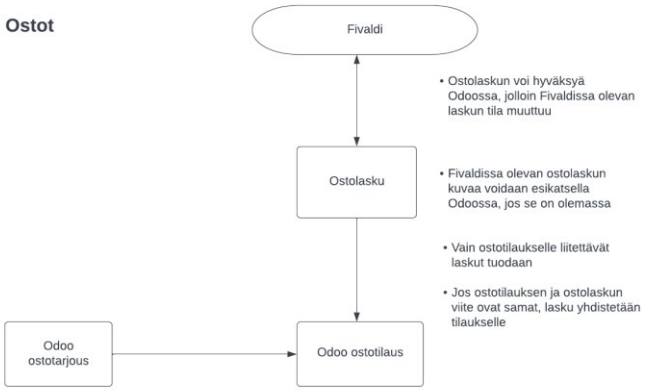
Moduulin toiminta on kuvattu alla olevassa kuvassa (KUVA 6). Lyhyesti kuvattuna ominaisuudet ovat seuraavat:

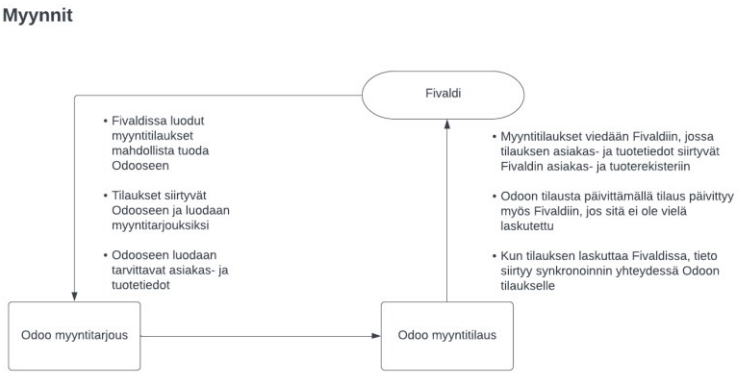
Ostot

- Ostolaskun voi hyväksyä Odoossa, jolloin Fivaldissa olevan laskun tila muuttuu
- Fivaldissa olevan ostolaskun kuvaa voidaan esikatsella Odoossa, jos se on olemassa
- Vain ostotilaukselle liitettävät laskut tuodaan
    - Jos ostotilauksen viite ja ostolaskun viite ovat samat, lasku yhdistetään ostotilaukselle

Myynnit

- Myyntitilaukset viedään Fivaldiin, jossa tilauksen asiakas- ja tuotetiedot siirtyvät tilauksen tietojen lisäksi Fivaldin asiakas- ja tuoterekisteriin
- Odoon tilausta päivittämällä tilaus päivittyy myös Fivaldiin, jos sitä ei ole vielä laskutettu
- Kun tilauksen laskuttaa Fivaldissa, tieto siirtyy synkronoinnin yhteydessä Odoon tilaukselle
- Fivaldissa luodut tilaukset mahdollista tuoda Odooseen ja niistä luodaan myyntitarjoukset
- Odooseen luodaan tarvittavat asiakas- ja tuotetiedot
    - Asiakas- ja tuotetiedot voidaan päivittää Fivaldiin myös erillisistä näkymistä





KUVA 6. Moduulin toiminta korkealla tasolla

3.1 Moduulin rakenne

Moduuli noudattaa kansiorakenteeltaan aikaisemmin määriteltyjä rakenteita (KUVA 2). Alla olevassa taulukossa (TAULUKKO 2) on määritelty tämän moduulin tiedostot ja niiden tarkoitus. Taulukossa määritellään vain luokkatiedostot sekä käyttöliittymätiedostot, koska init, manifest sekä muut vaaditut tiedostot on jo selitetty aikaisemmin.

TAULUKKO 2. Visma Fivaldi moduulin tiedostot

Tiedosto	**Tarkoitus**
models/account_move.py	Perii Odoon account_move luokan ja lisää siihen lisäkenttiä Fivaldin ostolaskuja varten.
models/sale_order.py	Perii Odoon sale_order luokan ja lisää siihen lisäkenttiä Fivaldin myyntitilauksia varten.
models/res_company.py	Perii Odoon res_company luokan ja lisää siihen lisäkenttiä Fivaldi-integraatiota varten.
models/res_partner.py	Perii Odoon res_partner luokan ja lisää siihen lisäkenttiä Fivaldi-integraatiota varten.
models/fivaldi_helper.py	Sisältää useassa paikassa käytettyjä funktioita sekä autentikointilogiikan.
models/decoders/decode_fivaldi_purchase_invoice.py	Käytetään muuntamaan Fivaldin rajapinnasta tulevat ostolaskut Odoon ostolaskuiksi.
models/decoders/decode_fivaldi_sale_order.py	Käytetään muuntamaan Fivaldin rajapinnasta myyntitilaukset Odoon myyntitilauksiksi.
models/encoders/encode_fivaldi_sale_order.py	Käytetään muuntamaan Odoon myyntitilaukset Fivaldin myyntitilauksiksi ennen luonti- tai muokkausrajapintojen kutsumista.
models/helpers/fivaldi_api.py	Sisältää kaikki Fivaldi-rajapinnan tarvittavat endpointit.

models/helpers/fivaldi_purchase_sync.py	Sisältää kaiken ostolaskujen tuontiin tarvittavan logiikan.
models/helpers/fivaldi_sale_order_sync.py	Sisältää kaiken myyntitilausten tuontiin ja vientiin tarvittavan logiikan.
views/account_move.xml	Käyttöliittymätiedosto, luo näkyviin ostolaskuja varten tarvittavat synkronointinapit sekä tietokentät.
views/res_company.xml	Käyttöliittymätiedosto, luo näkyviin Fivaldi integraatioon tarvittavat tekstikentät sekä testausnapit (KUVA 7).
views/res_partner.xml	Käyttöliittymätiedosto, luo yksittäisen asiakkaan asetusnäkymiin Fivaldi-ID:n noutopainikkeen.
views/sale_order.xml	Käyttöliittymätiedosto, luo näkyviin myyntitilauksia varten tarvittavat synkronointinapit sekä tietokentät.
data/fivaldi_cron.xml	Sisältää ajastetun ajamisen asetukset.

3.2 Autentikointi

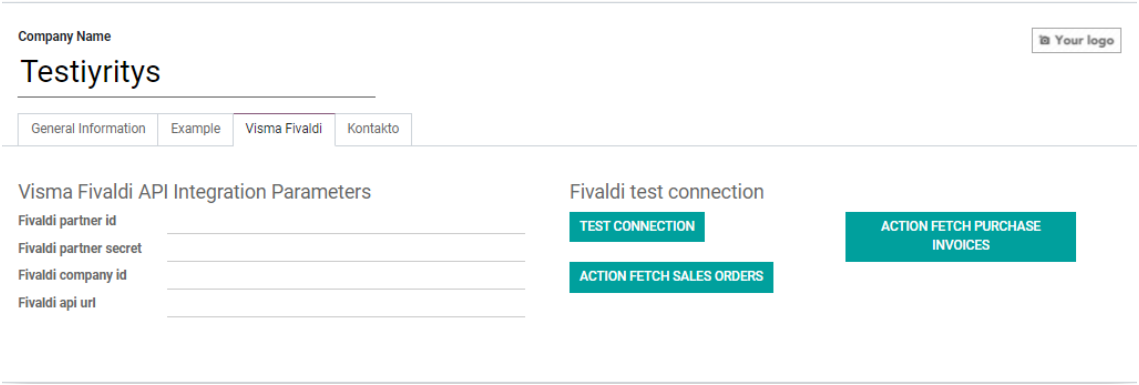
Fivaldi-rajapinnassa ei ole perinteistä access token autentikointia, vaan jokaiseen kutsuun generoidaan MAC-tunniste. Tunniste on string-muotoinen merkkijono, joka sisältää seuraavan listauksen mukaisen sisällön. Merkkijonon jokainen vaihe erotetaan newline-erottimella LF.

1. Http-metodi (GET, POST, PUT, PATCH, DELETE)
2. Kutsun body MD5 hashattuna. Jos bodya ei ole, annetaan tyhjä string
3. Kutsun content-type headerin arvo. Jos tätä headeria ei ole, annetaan tyhjä string
4. Kutsun kaikki X-Fivaldi-alkuiset headerit. Kaikissa kutsuissa vaadittuina on X-Fivaldi-Timestamp sekä X-Fivaldi-Partner, joten nämä tulee ainakin tästä kohdasta. Headerit erotellaan muodossa key:value + LF
5. Kutsun polku, alkaen ensimmäisestä /-merkistä loppuen query-string ?-merkkiin
6. Kutsun query-string, jos se on olemassa. Muuten ei anneta mitään

Tämän listan kohdista muodostunut merkkijono enkoodataan HMAC SHA256 muotoon käyttäen Fivaldista saatua kumppanisalasanaa avaimena. Saatu hash enkoodataan Base64-muotoon ja tästä saatu koodi lisätään http-kutsuun muodossa Authorization: Fivaldi LUOTU_KOODI. (Visma Fivaldi, n.d.b)

Fivaldi tarjoaa kehittäjille Postman-kokoelman, joka sisältää toimivan Javascriptillä toteutetun esimerkin rajapinnan autentikointiin. Tätä hyödyntäen on helppo muuntaa logiikka mille tahansa rajapintaa käyttävälle järjestelmälle. (Visma Fivaldi, n.d.b)

Tarvittavat autentikaatioparametrit on luotu res_company luokan asetusnäkymään xml muokkauksilla (KUVA 7). Samassa näkymässä on lisäksi toiminnot yhteyden testaamiseen. Tällä saadaan varmistettua, että kenttiin syötetyt arvot ovat oikein.



KUVA 7. Yrityksen asetusnäkymässä olevat Visma Fivaldi API tunnisteet

3.3 Myyntitilausten tuonti ja vienti

Tarkoituksena on viedä Fivaldiin Odoosta myyntitilaukset, jotta saadaan generoitua ja lähetettyä laskut niiden perusteella. Ideaalitilanne olisi, että Fivaldiin vietäisiin vain myyntilaskut ilman tilauksia, mutta Fivaldin rajapinta ei tue tätä. (Visma Fivaldi, n.d.a). Esimerkiksi Visman Netvisor tukee suoraan myyntilaskujen viemistä (Netvisor Tuki, n.d.).

Odoon myyntitilaukset koostuvat kokonaisuutena kahdesta päätaulusta: sale_order, joka sisältää myyntitilauksen päätiedot ja sale_order_line, joka sisältää myyntitilauksen tuoterivitiedot. Lisäksi tarpeellisia tauluja integraatiota varten, joihin myyntitilaus on yhteydessä ovat res_partner sekä product_product. Näistä res_partner sisältää tilauksen asiakastiedot ja product_product rivin tuotetiedon.

Fivaldi-myyntitilauksella on useita rajapinnan kautta muokattavia kenttiä, mutta tässä integraatiossa täytetään aluksi vain pakolliset kentät, joilla saadaan Odoon tilaus näkymään Fivaldissa ja Fivaldin tilaukset Odoossa. Asiakkaiden toiveiden mukaisesti on mahdollista myöhemmin jatkokehittää integraatiota lisäämällä dataa rajapintakutsuihin myös muista kentistä. Integraation mukana siirtyvät kentät on selitetty seuraavissa taulukoissa (TAULUKKO 3, TAULUKKO 4, TAULUKKO 5). Taulukoista on selitetty lisää seuraavissa kappaleissa.

TAULUKKO 3. Fivaldi myyntitilaus – odoo myyntitilaus

Fivaldi	**Odoo (sale_order)**	**Kuvaus**
id	fivaldi_order_id	Fivaldi tilauksen id
orderNumber	fivaldi_order_number	Fivaldi tilauksen numero
orderStatus	fivaldi_state	Fivaldi tilauksen tila
readyToInvoice	fivaldi_ready_to_invoice	Onko tilaus valmiina laskutukseen Fivaldissa
deliveryDate	delivery_date	Tilauksen toimituspäivä
orderDate	order_date	Tilauksen tilauspäivä
externalOrderNumber	name	Ulkoinen tilausnumero, viedään Fivaldiin Odoon tilauksesta. Muuten määrittelemätön
saleOrderRowDTOS	order_line	Myyntitilauksen tilausrivit

TAULUKKO 4. Fivaldi myyntitilausrivi – Odoo myyntitilausrivi

Fivaldi	**Odoo (sale_order_line)**	**Kuvaus**
textRow	-	Onko rivi tekstirivi vai tilausrivi, käytännössä tilauksen viennissä Fivaldiin aina false. Tuonnissa tuodaan vain rivit, joilla tämä arvo on false
productCode	product.default_code	Rivin tuotteen tuotekoodi, tilausta tuotaessa tuote haetaan tai luodaan Odooseen.
description	name	Rivin tuotteen nimi
unitPriceExcludingTax	price_unit	Rivin tuotteen veroton yksikköhinta
quantity	product_uom_qty	Rivin tuotteen määrä
-	product.product_id	Luodun tuotteen id

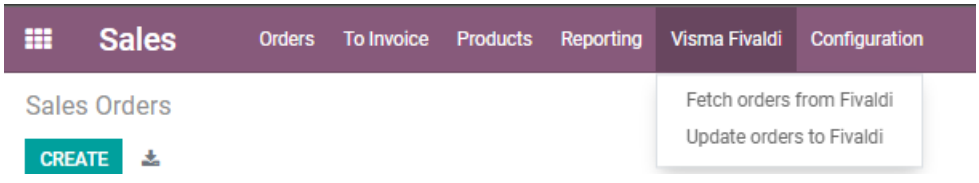
TAULUKKO 5. Fivaldi asiakas – Odoo asiakas

Fivaldi	**Odoo (res_partner)**	**Kuvaus**
customerId	fivaldi_customer_id	Fivaldi asiakkaan id
businessId	business_id	Asiakkaan y-tunnus
customerName	name	Asiakkaan nimi
email	email	Asiakkaan sähköposti
phoneNumber	phone	Asiakkaan puhelinnumero
isCustomer	-	Onko asiakas vai ei, arvo aina True
ovtIdentifier	e_invoice_ovt	E-laskutustunnus

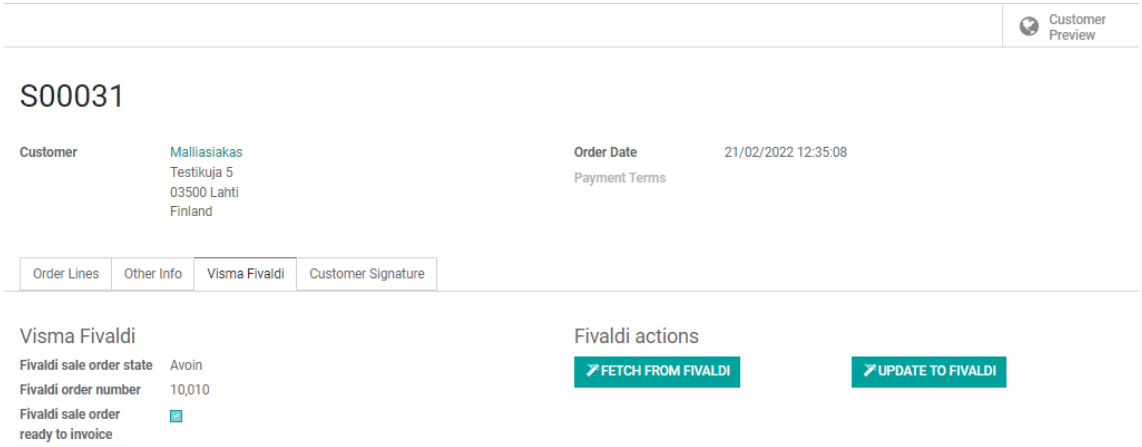
3.3.1 Myyntitilausten vienti

Myyntitilauksen vientiin on kolme vaihtoehtoa:

1. Myyntinäkymän yläpalkissa olevan Visma Fivaldi -valikon Update orders to Fivaldi -nappi, joka päivittää tai luo kaikki myyntitilaukset Fivaldiin (KUVA 8).
2. Yksittäisen myyntitilauksen sivulla oleva Update to Fivaldi -nappi (KUVA 9), joka päivittää tai luo yksittäisen myyntitilauksen Fivaldiin.
3. Automaattinen ajastus, joka ajaa x minuutin välein. Tämä ajaa kohdan 1. toiminnallisuuden. Automaattisen ajastuksen toimintaa käyty läpi tarkemmin kappaleessa 3.5.



KUVA 8. Myyntinäkymän Visma Fivaldi -valikko

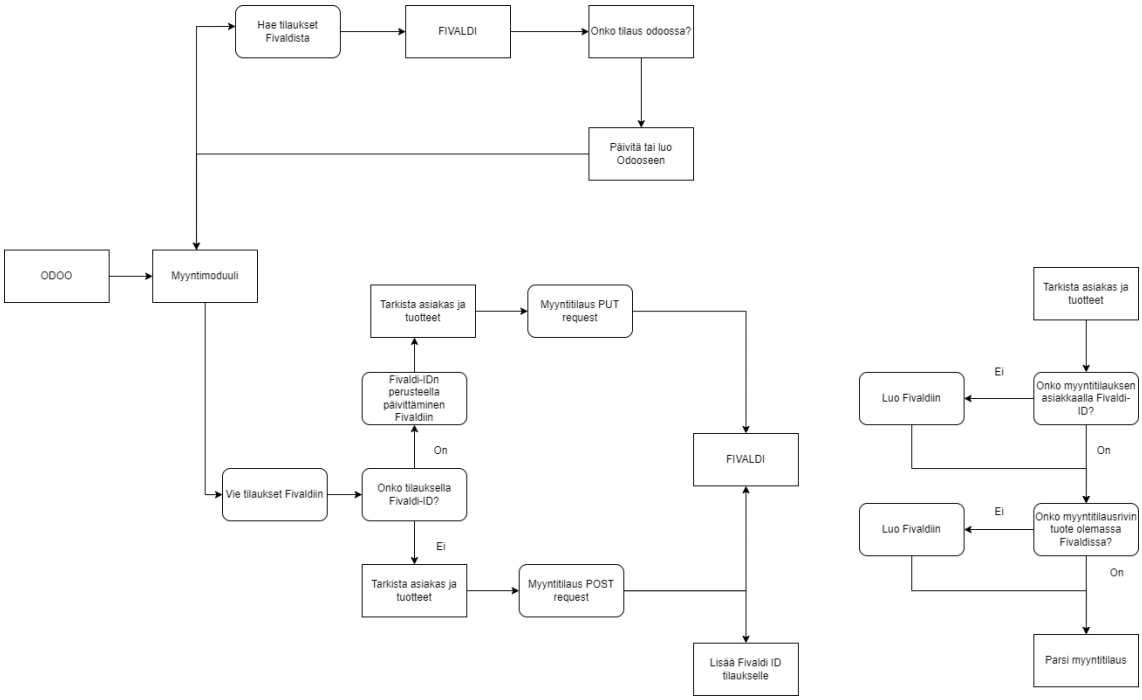


KUVA 9. Yksittäisen myyntitilauksen Fivaldi-välilehti

Käytännössä vientiflow menee alla olevan kaavion (KUVA 10) mukaisesti. Viennin yhteydessä luonti tai päivitys päätellään fivaldi_order_id kentän perusteella, koska tätä kenttää tarvitaan yksittäisten tilausten päivittämiseen. Tilausta ei ole vielä Fivaldissa, jos sillä ei ole arvoa tässä kentässä. Jos tilauksella on arvo kyseisessä kentässä, tämä kertoo, että tilaus on myös Fivaldissa. Vientiä aloitettaessa, Odoosta haetaan ne myyntitilaukset, jotka eivät ole draft-tilassa ja joilla fivaldi_state ei ole INVOICED. Tähän ehtoon menee kaikki vahvistetut uudet Odoo myyntitilaukset ja olemassa olevista ne, joita ei ole vielä laskutettu Fivaldissa. Fivaldi-laskutuksen jälkeen tilausta ei enää viedä Fivaldiin, koska tilausta ei voi enää päivittää Fivaldissa, eikä Odoossa. Laskutetut tilaukset saadaan päätteltyä Odoo tilauksen fivaldi_state kentästä, joka päivittyy INVOICED-tilaan synkronoinnin yhteydessä laskutetuille.

Vientiflow on uuden sekä olemassa olevan tilauksen tapauksessa pääpiirteittäin lähes sama, koska tilausrivin tuotteiden ja asiakkaan olemassaolo pitää tarkistaa Fivaldista viennin yhteydessä. Jos asiakasta tai tuotetta ei löydy, kun tilausta viedään, rajapinta palauttaa virheen ja tilausta ei saada vietyä. Tämän vuoksi integraatio luo Fivaldiin ennen tilauksen vientiä tarvittavat asiakas- ja tuotetiedot, jos niitä ei sieltä löydy. Integraatio etsii Fivaldista asiakastiedot nimen perusteella ja tuotetiedot tuotekoodin perusteella. Jos asiakkaan luonti Fivaldiin on tarpeellista, Odoo asiakkaan ja Fivaldi asiakkaan muunnos näkyy yllä olevassa taulukossa (TAULUKKO 5). Tuotteen kohdalla muunnos noudattaa myyntitilausrivin dataa (TAULUKKO 4) muuten, paitsi kenttinä viedään vain productCode, description ja hintakenttänä toimii salesPrice.

Uuden tilauksen tapauksessa flowssa on yksi lisäkutsu varmasti enemmän, koska tilauksen luonti-endpoint ei palauta luodun tilauksen id-kentän arvoa, vaan externalBatchId arvon. Tämän arvon perusteella saadaan haettua luonnin jälkeen erillisellä get-kutsulla externalBatchId perusteella Fivaldista luotu myyntitilaus, jonka orderNumber ja id saadaan yhdistettyä Odoo myyntitilaukseen.



KUVA 10. Flow myyntitilausten tuonnista ja viennistä

3.3.2 Myyntitilausten tuonti

Myyntitilausten tuontiin on samat kolme vaihtoehtoa, kuin viennilläkin. Yläpalkki, yksittäisen jo Fivaldissa olevan tilauksen sivu tai ajastettu ajo. Myyntitilausten tuonnin (KUVA 10) yhteydessä merkataan aina viimeisimmän onnistuneen haun ajanhetki res_company taulukkoon. Tämä aika lisätään myyntitilaushakukutsuun, joka suodattaa myyntitilaukset luontihetken mukaan. Tällä saadaan vain uusimmat ja ne tilaukset, joita Odooseen ei ole vielä luotu. Jos aikasuodatusta ei käytä, Fivaldin rajapinta palauttaa kaikki siellä olevat myyntitilaukset. Odoossa olevat nykyiset myyntitilaukset, joilla on fivaldi_order_id sekä fivaldi_state ei ole INVOICED haetaan erikseen id:n perusteella. Tämän tulokset sekä päivämäärällä haetut tilaukset lisätään listaan, joka ajetaan luonti ja päivitysfunktion läpi.

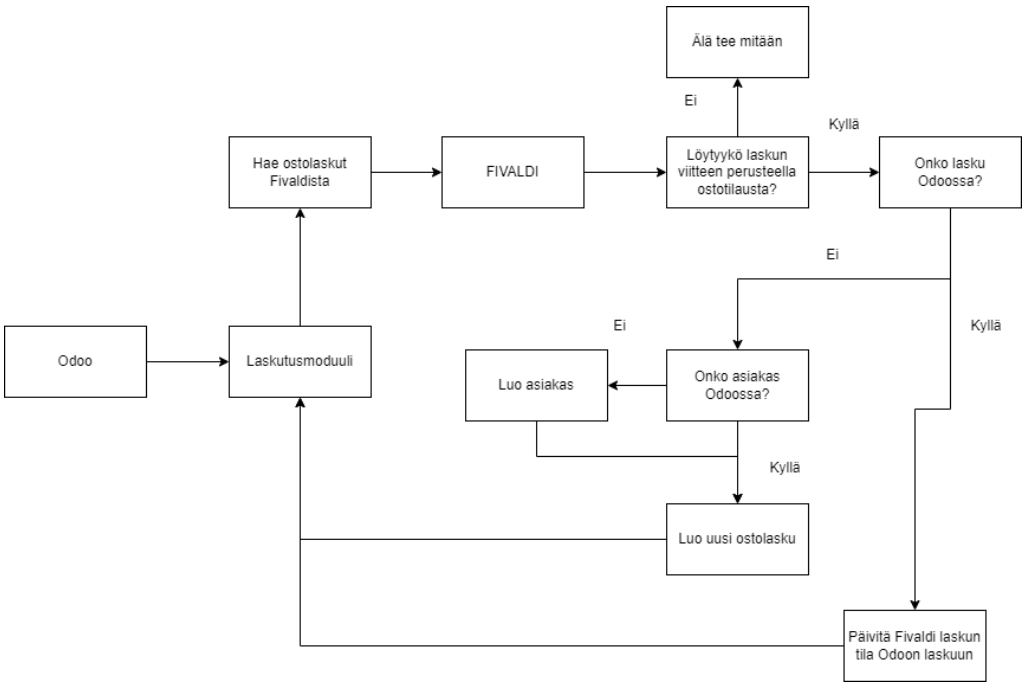
Uutta tilausta luotaessa luontiflow tarkistaa Odoon puolelta, onko asiakasta sekä myyntitilausrivin tuotetta olemassa. Tuote sekä asiakas luodaan, mikäli niitä ei löydy. Luonnin yhteydessä uudelle tilaukselle tuodaan kaikki pakolliset kentät. Olemassa olevalle tilaukselle päivitysflow muuttaa vain fivaldi_state sekä fivaldi_ready_to_invoice kenttiä. Näillä saadaan seurattua tilauksen valmistumista ja laskuttamisen valmiustilaa. Tämän valmiustilan muuttaminen onnistuu Odoosta myyntitilauksen asetuksiin luodulla kentällä, joka on myös näkyvissä yllä olevassa kuvassa (KUVA 9).

3.4 Ostolaskujen tuonti

Fivaldista tuodaan ostolaskut Odooseen, jonka kautta ne saadaan esikatseltua pdf-muotoisena sekä hyväksyttyä tarvittaessa. Lisäksi Odoon kirjanpitonäkymiin ja erilaisiin muihin tietonäkymiin saadaan kulutustietoja ostolaskujen kautta. Odooseen tuodaan kaikki ostolaskut, joille löytyy ostotilaus laskun viitteen perusteella. Viitteen avulla ostolasku saadaan linkitettyä Odoon ostotilaukseen. Ostolaskurajapinnassa on lukittuina oletuksena tietoturvasyistä kaikki luonti- ja päivitysrajapinnat, joten jos ne halutaan käyttöön, pitää ottaa yhteyttä Fivaldin tukeen.

3.4.1 Ostolaskudatan tuonti

Fivaldista ostolaskujen tuonti tapahtuu alla olevan kuvan mukaisesti (KUVA 11). Ostolaskujen tuonti aktivoidaan joko yläpalkista myyntitilausten tavoin (KUVA 8) tai ajastetulla ajolla. Onnistuneen ostolaskujen noudon yhteydessä talletetaan res_company taulukkoon viimeisin noutoaika, jota käytetään ostolaskujen noutoon rajapinnasta. Rajapintakutsuun annetaan tämä aika, jolla saadaan suodatettua myyntitilausten tavoin vain uusimmat ostolaskut. Ostolaskujen muuntaminen Odoon tietomalliin tapahtuu alla olevien taulukoiden tavoin (TAULUKKO 6, TAULUKKO 7, TAULUKKO 8).



KUVA 11. Flow ostolaskujen tuonnista Fivaldista

Ostolaskujen tuontiflow (KUVA 11) noudattaa pitkälti samaa logiikkaa, kun myyntitilausten tuonti (KUVA 10). Eroina on tuotteiden olemassaolon tarkastamisen puuttuminen. Tätä ei tehdä, koska ostolaskun luominen onnistuu ilman tuoterelaatioita. Lisäksi ennen ostolaskujen tuontia tarkastetaan, löytyykö Odoosta laskun viitteen perusteella ostotilausta. Tällöin ostolasku saadaan linkitettyä ostotilaukselle ja sen seuraaminen järkevöityy.

TAULUKKO 6. Odoo ostolasku – Fivaldi ostolasku

Fivaldi	**Odoo (account_move)**	**Kuvaus**
invoiceId	fivaldi_invoice_id	Fivaldi laskun id
-	move_type	Aina in_invoice ostolaskuilla
bankRefNo	ref	Laskun viitenumero
bankRefNo	payment_reference	Maksun viitenumero
originalInvoiceNo	name	Laskun nimi/numero
dueDate	invoice_date_due	Laskun eräpäivä
total	amount_total	Laskun kokonaissumma
invoiceDate	invoice_date	Laskun lähetyspäivä
TAULUKKO 7	partner_id	Laskun tietojen perusteella luotu tai haettu asiakas id
iban	partner_bank_id	Pankkitilin id haetaan iban-koodin ja partner id:n perusteella Odoosta, jos ei sitä löydy niin uusi pankkitili luodaan
-	fivaldi_last_sync	Viimeisin synkronointi, päivitetään luonnissa tai päivityksessä
invoiceStatus	fivaldi_invoice_status	Laskun tila Fivaldissa
currentStepId	fivaldi_invoice_step_id	Laskun Fivaldi vaiheen id
currentStep	fivaldi_invoice_step	Laskun Fivaldi vaiheen nimi
invoiceRows (TAULUKKO 8)	invoice_line_ids	Laskun rivit

TAULUKKO 7. Fivaldi asiakas – Odoo asiakas ostolaskudatan pohjalta

Fivaldi	**Odoo (res_partner)**	**Kuvaus**
supplierId	fivaldi_customer_id	Laskun lähettäjän id
supplierName1	name	Laskun lähettäjän nimi
supplierAddress1	street	Laskun lähettäjän osoite
supplierAddress2	street2	Laskun lähettäjän osoite
supplierPost-alCode	zip	Laskun lähettäjän postinumero
supplierCity	city	Laskun lähettäjän kaupunki
supplierCountryId	country_id	Laskun lähettäjän maakoodi

TAULUKKO 8. Fivaldi ostolaskurivi – Odoo ostolaskurivi

Fivaldi	**Odoo(account_move_line)**	**Kuvaus**
productAmount	quantity	Rivin tuotteen määrä
name	name	Rivin tuotteen nimi
total / productAmount	price_unit	Rivin tuotteen yksikköhinta
-	exclude_from_invoice_tab	Poistetaanko rivi näkyvistä laskuvälilehdeltä. Aina False
vatPercent	tax_ids	Rivin veroprosentti. Odoosta haetaan kyseisen veroprosentin ID tai luodaan uusi

Pääasiassa Odoosta on käytössä kolme taulua, account_move, joka sisältää ostolaskun tiedot, account_move_line, joka sisältää ostolaskun rivitiedot sekä res_partner, joka sisältää asiakastiedot sekä. Lisäksi tietoa on erilaisissa aputauluissa kuten res_partner_bank, account_tax, ir_attachment sekä purchase_order. Näistä res_partner_bank sisältää laskulta tuotuja pankkitietoja asiakkaalle, account_tax-taulun avulla noudetaan veroprosenttien id-tietoja, ir_attachment-tauluun tallennetaan laskun liitteet ja purchase_order tauluun tallennetaan relaatiotieto ostolaskuihin.

3.4.2 Ostolaskun vahvistaminen

Ostolaskujen vahvistaminen Fivaldissa Odoon kautta tapahtuu käyttöliittymässä (KUVA 13) näkyvän ”CONFIRM IN FIVALDI”- painikkeen kautta. Painike on näkyvissä vain vahvistamattomille laskuille. Painikkeen klikkaamisen jälkeen kutsutaan post-kutsulla approve-endpointtia, joka vahvistaa laskun nykyisen vaiheen ja siirtää sen Fivaldissa käsittelyssä eteenpäin. Ostolaskut voidaan myös vahvistaa massana niiden listanäkymästä valitsemalla halutut laskut ja klikkaamalla vahvistuspainiketta.

3.4.3 Ostolaskun liitteiden tuonti

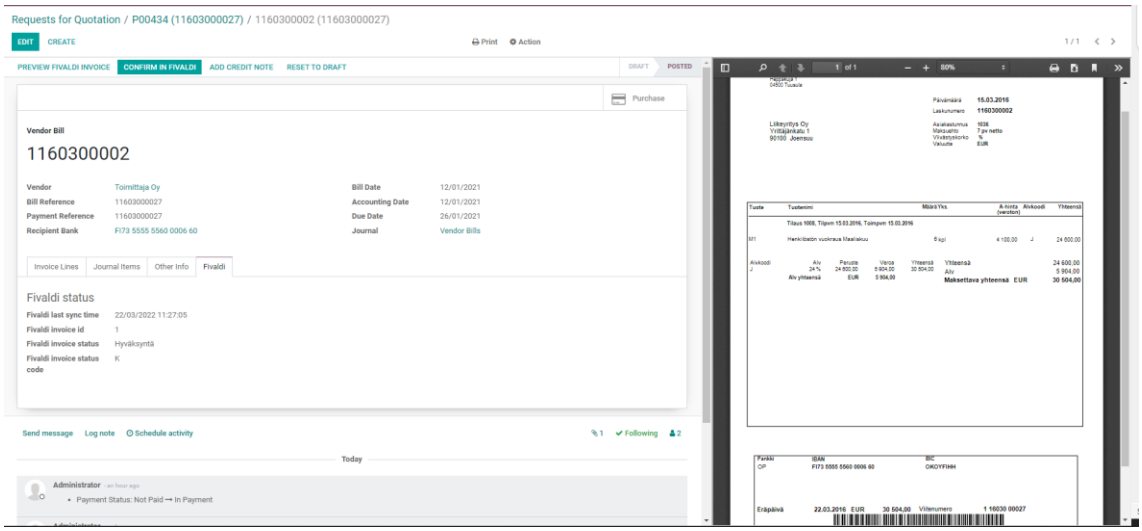
Fivaldin rajapinta tukee myös ostolaskujen liitteiden tuomista, jolloin ne haetaan erikseen arkistorajapinnan kautta. Ostolaskuhakuun pitää lisätä parametriksi ”includeAttachment” -boolean arvo, jolloin laskulle tulee liitetiedostot array-muotoisessa ”archiveFileIds” -kentässä. Liitetiedostot eivät tule palautusarvossa suoraan, vaan pelkästään niiden id:t. Liitetiedostojen latausosoitteet pitää hakea vielä id:n perusteella arkistorajapinnasta, jolloin saadaan linkki, jonka kautta tiedostoa voidaan esikatsella (Visma Fivaldi. n.d.a).

Odoossa liitetiedosto pitää luoda ir_attachment tauluun oikealle mallille, joka on tässä tapauksessa account_move, sekä liittää oikealle laskulle laskun sisäisen id:n perusteella. Haluttu liitetiedosto ladataan rajapinnasta saadusta linkistä ja muunnetaan base64 muotoon, jonka jälkeen se lisätään edellä mainittujen kenttien lisäksi riville. Lisäksi ir_attachment taululle määritellään mm. nimi, datan tyyppi, kuvaus sekä tiedoston tyyppi, jotka ovat näkyvillä alla olevassa kuvassa (KUVA 12).

```
self.env['ir.attachment'].create({
    'name': name,
    'type': 'binary',
    'res_id': res_id,
    'res_model': model,
    'datas': data,
    'mimetype': 'application/x-pdf',
    'description': _('Fivaldi %s document.') % account_move.name,
})
```

KUVA 12. Liitetiedoston luonti laskulle

Liitteiden tuomisen jälkeen liitetiedostot ilmestyvät laskun tietonäkymän sivupalkkiin näkyviin (KUVA 13). Laskun liitteiden näkyminen Odoossa yhdessä näkymässä nopeuttaa huomattavasti laskun käsittelyprosessia käyttäjän näkökulmasta, kun näkee tarvittavat tiedot yhdessä näkymässä ja tämän jälkeen laskun voi hyväksyä, kun kokee sen tarpeelliseksi. Samassa kuvassa (KUVA 13) vasemmassa yläreunassa näkyy myös laskun hyväksyntä- sekä esikatselupainike

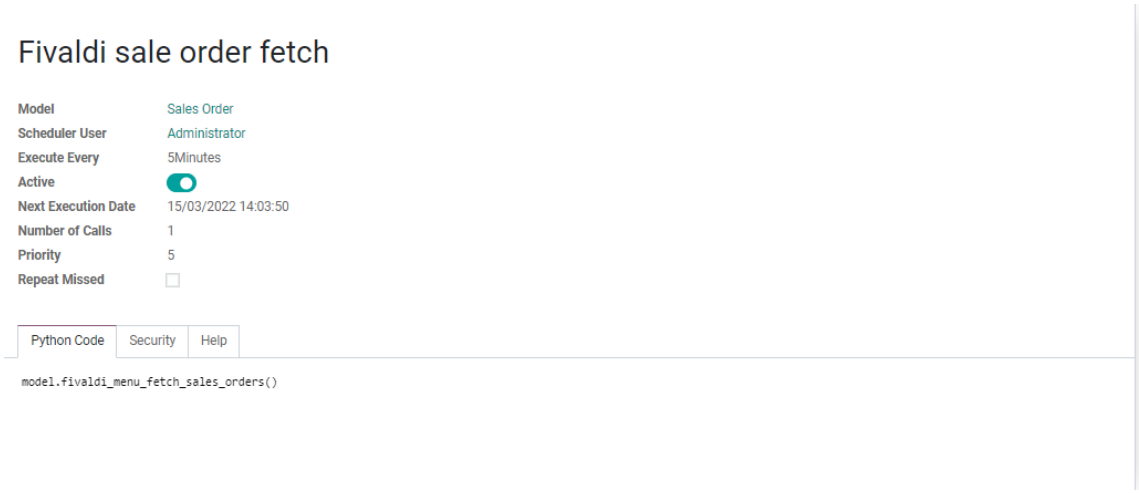


KUVA 13. Fivaldista tuotu liitetiedosto näkyvissä laskulla

3.5 Integraatioiden ajastettu ajaminen

3.5.1 Integraatioiden ajaminen käyttöliittymän kautta

Käyttöliittymästä ajastetun tehtävän luominen onnistuu asetusnäkymän kautta, settings -> technical -> scheduled actions -> create. Lomakkeella (KUVA 14) määritetään ajon asetukset ja python-koodissa kutsutaan valitusta mallista metodi, joka ajetaan. Alla olevassa kuvassa (KUVA 14) on määritetty myyntitilausten synkronointi Fivaldin kanssa, lomakkeen kentissä tärkeimmät määritellyt kentät ovat malli (Model) sekä toistoväli (Execute Every). Lisäksi lomakekenttien alla olevassa koodikentässä kirjoitetaan ajossa ajettava koodi, esimerkin tapauksessa kutsutaan Fivaldi moduulissa luotua metodia fivaldi_menu_fetch_sales_orders. Tämä on sama metodi. jota kutsutaan käyttöliittymän painikkeesta (KUVA 8)



KUVA 14. Odoo ajastetun tehtävän määrittely käyttöliittymässä

3.5.2 Integraatioiden ajaminen moduulin koodin kautta

Ajastetun tehtävän luominen koodin kautta on hieman työläämpi, kuin käyttöliittymän kautta, koska tehtävää varten joutuu luomaan moduuliin oikean muotoisen xml-tiedoston ja määrittelemään sen manifest.py-tiedostoon asennettavaksi tiedostoksi. Käytännössä xml-tiedosto sisältää samat kentät, jotka käyttöliittymän

lomakkeessakin on määritettävissä. Alla olevassa kuvassa (KUVA 15) on näkyvissä myyntitilauksen noutoon tehty ajastettu tehtävä.

```xml
<?xml version="1.0" encoding="utf-8"?>
<odoo>
    <data>
        <record id='ir_cron_fetch_fivaldi_sale_orders' model='ir.cron'>
            <field name='name'>Fetch fivaldi sale orders</field>
            <field name='model_id' ref='sale.model_sale_order'/>
            <field name='state'>code</field>
            <field name='code'>model.fivaldi_menu_fetch_sales_orders()</field>
            <field name='interval_number'>5</field>
            <field name='interval_type'>minutes</field>
            <field name='numbercall'>1</field>
            <field name='priority'>5</field>
            <field name="doall" eval="False"/>
        </record>
    </data>
</odoo>
```

KUVA 15. Odoo ajastetun tehtävän määrittely koodissa

4 POHDINTA

Opinäytetyön toteuttamisen aikana kertyi lisää paljon lisää tietoa Odoosta niin yleisellä tasolla, kuin teknisestä toteutuksesta. Lisäksi tutummaksi muodostui erilaiset myyntiin ja laskutukseen liittyvät asiat Odoon kirjanpito- ja myyntityökalujen läpikäymisen sekä Fivaldin kautta. Suurempia haasteita toteutuksen aikana ei ilmennyt, mutta yksi mainittava asia voisi olla Odoon XML-tiedostojen avulla käyttöliittymämuokkauksien tekeminen. Näiden XML-tiedostojen syntaksi on usein haasteellista, koska se sisältää paljon asioita, joita ei ole dokumentoitu välttämättä hyvin ja näiden asioiden löytäminen onnistuu vain selaamalla Odoon lähdekoodista vastaavia tiedostoja ja tarkastelemalla niiden toteutustapoja.

Rajapintojen kanssa joutuu työskentelemään nykyään melkein jatkuvasti, koska lähes kaikki palvelut tarjoavat jonkinlaisen rajapinnan ulospäin käytettäväksi. Rajapinnoissa yksi mielenkiintoinen seurattava asia on kuinka yhteneväisiä ne ovat sisäisesti, eli onko samankaltaisella datalla samat avaimet eri api-endpointeissa tai onko rajapinnan eri osioilla samat ominaisuudet. Esimerkiksi Fivaldin rajapinnan tapauksessa ostolaskuihin liittyvä rajapinta on huomattavasti laajempi, kuin vaikka myyntitilauksien. Yksittäisten kenttien muokkaaminen myyntitilaukselle ei ole mahdollista suoraan patch-kutsulla, vaan koko tilaus pitää lähettää joka kerta, jos jotain kenttää haluaa muokata. Ostolaskuilla käytettävissä on patch-kutsut, joilla yksittäisten kenttien muokkaaminen on mahdollista.

Erityisen positiivista Fivaldin rajapinnassa on Swagger-käyttöliittymän olemassaolo, jossa on selkeästi visualisoituna saatavilla olevat api-endpointit sekä niiden vaatima ja palauttama data. Lisäksi Swagger-dokumentaatio on aina ajan tasalla rajapinnan ominaisuuksien kanssa, koska Swagger luo dokumentaation automaattisesti backend-koodin pohjalta. Swagger-pohjaisella dokumentaatiolla on siis useita etuja verrattuna pelkkään tekstipohjaiseen dokumentaatioon, jossa api-endpointtien tiedot saattavat olla omissa formaateissa ja dokumentaation päivittäminen on kehittäjistä kiinni.

Opinnäytetyön tavoitteena oli toteuttaa integraatio, jolla on mahdollista vähentää manuaalista työtä Odoota ja Visma Fivaldia yhdessä käyttäviltä asiakkailta. Suu-

rin manuaalista työtä vähentävä tekijä on myyntitilausten sekä ostolaskujen automaattinen siirtyminen Odoon ja Fivaldin välillä. Ilman integraation mahdollistamaa automaatiota, esimerkiksi Odoossa luodut myyntitilaukset pitäisi lisätä manuaalisesti Fivaldiin eikä niiden tilatiedot siirtyisi automaattisesti Odooseen. Lisäksi Fivaldista saapuvat ostolaskut eivät yhdistyisi suoraan Odoossa tehdyille ostotilauksille, joka hidastaisi niiden käsittelyä, koska pitäisi hyppiä kahden järjestelmän välillä varmistamassa ovatko ostotilaus ja ostolasku yhteneviä.

Integraation ja opinnäytetyön tavoitteisiin mielestäni päästiin, koska alussa määritellyt ominaisuudet, eli myyntitilausten tuonti- ja vienti sekä ostolaskujen tuonti saatiin toteutettua erilliseen integraatiomoduuliin. Tuotetun moduulin ominaisuuksien avulla Odoon ja Fivaldin yhteiskäyttäjille saadaan tarjottua helpotuksia vähentämällä manuaalisia työvaiheita, joita tiedon siirtämisestä ohjelmistojen välillä aiheutuisi.

LÄHTEET

Collapick Company. N.d. Collapick Tempo – ERP teollisuuteen. Verkkosivu. Luettu 14.3.2022 Collapick Tempo - teollisuuteen optimoitu Odoo ERP

Netvisor Tuki. N.d. Resurssit – myyntilaskut, -tilaukset ja hyvityslaskut. Verkkoaineisto. Luettu 14.3.2022. Resurssit - Myyntilaskut, -tilaukset ja hyvityslaskut : Netvisor

Plesk. 2018. Why WordPress is still the most popular CMS choice for websites. Luettu 14.3.2022. Why WordPress is still the most popular CMS choice for websites - Plesk

Odoo. N.d.a. About us. Verkkosivu. Luettu 14.3.2022 About Us | Odoo

Odoo. N.d.b. Compare Odoo Editions. Verkkosivu. Luettu 2.2.2022 Odoo Enterprise vs Community | Odoo Editions Comparison

Odoo. N.d.c. Chapter 1: Architecture Overview. Verkkosivu. Luettu 2.2.2022 Chapter 1: Architecture Overview — Odoo 15.0 documentation

Odoo. N.d.d. Find a Partner. Verkkosivu. Luettu 2.2.2022. Find a Partner - Odoo

Odoo. N.d.e. The Odoo story. Verkkosivu. Luettu 14.3.2022. The Odoo story

Odoo. N.d.f. Odoo pricing. Verkkosivu. Luettu 14.3.2022. Odoo Pricing | Odoo

Odoo. N.d.g. QWeb Templates. Verkkosivu. Luettu 14.3.2022. QWeb Templates — Odoo 15.0 documentation

Odoo. N.d.h. 10000 Apps in the Odoo app store. Verkkosivu. Luettu 14.3.2022. 10,000 Apps in the Odoo App Store | Odoo

Speedy Sense. 2019. Understand Odoo Module Structure. Verkkosivu. Luettu 14.3.2022. Understand Odoo Module Structure - SpeedySense

Swagger. 2021. What is Swagger? Verkkosivu. Luettu 22.4.2022. What is Swagger

Visma. N.d. Valitse tarpeisiisi sopiva taloushallinto-ohjelma. Verkkosivu. Luettu 14.3.2022. Taloushallinto-ohjelmat | Visma Taloushallinto - Visma

Visma Fivaldi. N.d.a. Visma Fivaldi Customer API. Verkkoaineisto. Luettu 14.3.2022. Swagger UI (fivaldi.net)

Visma Fivaldi. N.d.b. Fivaldi Customer API. Verkkoaineisto. Luettu 14.3.2022 https://manuals.fivaldi.net/customer/api/auth.html