Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

NADJA GRÖNFORS

SharePoint integraatio KUP-toiminnanohjausjärjestelmään

TIETOJENKÄSITTELYN TUTKINTO-OHJELMA
2025

TIIVISTELMÄ

Grönfors, Nadja: SharePoint integraatio KUP-toiminnanohjausjärjestelmään
Opinnäytetyö, AMK
Tietojenkäsittely
Tammikuu 2025
Sivumäärä: 29

Tämän opinnäytetyön tarkoituksena oli toteuttaa integraatio KP-ServicePartner Oy:n KUP-toiminnanohjausjärjestelmän ja Microsoft SharePointin välille. Integraation avulla automatisoitiin uusien työtilausten avaaminen toiminnanohjausjärjestelmään SharePointissa luotujen tarjousten perusteella. Toteutus tehtiin hyödyntäen Python-ohjelmointikieltä, Django-ohjelmistokehystä ja Microsoft Graph API-rajapintaa.

Ohjelma hakee voitetuista tarjouksista tarvittavat metatiedot, tarkistaa asiakkaan olemassaolon ja lisää tarvittaessa uuden asiakkaan järjestelmään. Se luo uuden työn, palauttaa työnumeron SharePointiin ja lähettää sähköposti-ilmoituksen työstä vastaavalle henkilölle.

Projektin tuloksena syntyi toimiva integraatio, joka automatisoi työn avaamisen toiminnanohjausjärjestelmään. Tämä parantaa yrityksen toimintaprosesseja ja vähentää manuaalisen työn määrää. Työn aikana opittiin paljon uusista teknologioista ja niiden hyödyntämisestä käytännön sovelluksissa. Jatkossa integraatiota voidaan laajentaa ja kehittää edelleen, jotta se palvelee yrityksen tarpeita entistä paremmin.

Avainsanat: Microsoft, Graph API, Django, SharePoint, Python, OpenSSL, toiminnanohjausjärjestelmä

ABSTRACT

Grönfors, Nadja: SharePoint Integration to the KUP Enterprise Resource Planning System
Bachelor's thesis
Business Information Systems
January 2025
Number of pages: 29

The purpose of this thesis was to implement an integration between KP-ServicePartner Oy's KUP enterprise resource planning (ERP) system and Microsoft SharePoint. The integration aimed to automate the creation of new workorders in the ERP system based on offers created in SharePoint. The implementation utilized the Python programming language, the Django framework and the Microsoft Graph API.

The program retrieves the necessary metadata from approved offers, checks if the customer already exists and adds a new customer to the system if necessary. It creates a new workorder, returns the workorder number to SharePoint and sends an email notification to the person responsible.

The project resulted in a functional integration that automates the creation of workorders in the ERP system. This improves the company's operational processes and reduces the amount of manual work required. Significant knowledge of new technologies and its usage in practical applications was gained during the project. In the future, integration can be expanded and further developed to better serve the company's needs.

Keywords: Microsoft, Graph API, Django, SharePoint, Python, OpenSSL, ERP

ALKUSANAT

Haluan esittää lämpimät kiitokseni KP-ServicePartner Oy:lle tästä mahdollisuudesta toteuttaa tämä opinnäytetyö. Erityisesti haluan kiittää Akia ja Petraa luottamuksesta, tuesta, ohjauksesta ja kannustuksesta projektin aikana.

Haluan myös osoittaa erityiset kiitokset Vesalle, joka jaksoi kerta toisensa jälkeen tukea, opastaa ja auttaa haastavissa tilanteissa. Hänen asiantuntemuksensa ja kärsivällisyytensä olivat ratkaisevassa roolissa työn onnistumisessa.

Teidän avuliaisuutenne ja yhteistyönne ovat olleet korvaamaton apu työn edistymisessä. On ollut etuoikeus työskennellä ympäristössä, jossa oppimista ja kehitystä arvostetaan.

Kiitos, että annoitte minulle mahdollisuuden oppia ja kehittyä osana KP-ServicePartnerin toimintaa.

SISÄLLYS

1 JOHDANTO 7
2 NYKYINEN JÄRJESTELMÄYMPÄRISTÖ 7
2.1 KUP-toiminnanohjausjärjestelmä 7
2.2 Microsoft 9
2.2.1 Entra ID 9
2.2.2 SharePoint 10
3 TEKNOLOGIAT JA TYÖKALUT 14
3.1 Microsoft Graph API 14
3.2 Python 15
3.3 Django 16
3.4 OpenSSL 17
3.5 Sertifikaatilla tunnistautuminen 18
4 RAJAPINNAN SUUNNITTELU 19
4.1 Tarjoustyökalu 19
4.2 Toteutuksen kuvaus 19
5 RAJAPINNAN TOTEUTUS 21
5.1 Sovelluksen rekisteröinti ja autentikointi 21
5.2 SharePointin tietojen määritys ja endpointit 23
5.3 Tietojen prosessointi 24
5.4 Sähköpostin lähetys 26
5.5 Linkin lisäys työn lisätiedot-välilehdelle 28
5.6 Virheidenkäsittely 30
6 TULOKSET JA POHDINTA 30
LÄHTEET 33

SYMBOLI- JA LYHENNELUETTELO

ORM	Object-Relational Mapping, ohjelmointitekniikka, joka helpottaa tietokannan ja olio-ohjelmoinnin välistä integraatiota muuntamalla tietokannan taulut olioiden muotoon
API	Application Programming Interface, ohjelmointirajapinta, jolla mahdollistetaan vuorovaikutus eri ohjelmistojen ja palveluiden välillä
RESTful	REST (Representational State Transfer) arkkitehtuurin periaatteiden mukaisesti rakennettu verkkopalvelu

1 JOHDANTO

Tämän opinnäytetyön tarkoituksena oli toteuttaa integraatio Microsoft SharePointin ja KP-ServicePartnerilla käytössä olevan KUP-toiminnanohjausjärjestelmän välille. Yrityksen tarjoukset tehdään SharePoint-sivustolle, jossa niille annetaan metatietona projektin tekemistä varten tarvittavia tietoja. Näitä tietoja ovat mm. asiakkaan tiedot ja viite, työlaji ja myyjän tiedot. Näiden tietojen avulla tehdään automaattisesti uusi työ toiminnanohjausjärjestelmään. Toiminnanohjausjärjestelmä palauttaa SharePointille työnumeron, joka lisätään omaan metatietokenttään.

KP-ServicePartner Oy on vuonna 2003 perustettu perheyritys, joka erikoistuu teollisuus- ja siltanostureihin ja teollisuuden kunnossapitoon. Henkilöstöä yrityksellä on noin 100 neljällätoista eri paikkakunnalla. Pääkonttori on Siilinjärvellä. (KP-ServicePartner, n.d.)

Tähän opinnäytetyöhön on käytetty ChatGPT- ja Copilot-tekoälyjä alustavan sisällysluettelon luomiseen, termien suomentamiseen sekä lauserakenteiden parantamiseen. Kaikki lähteet on etsitty itse, eikä tekoälyä ole käytetty lähteiden hankintaan.

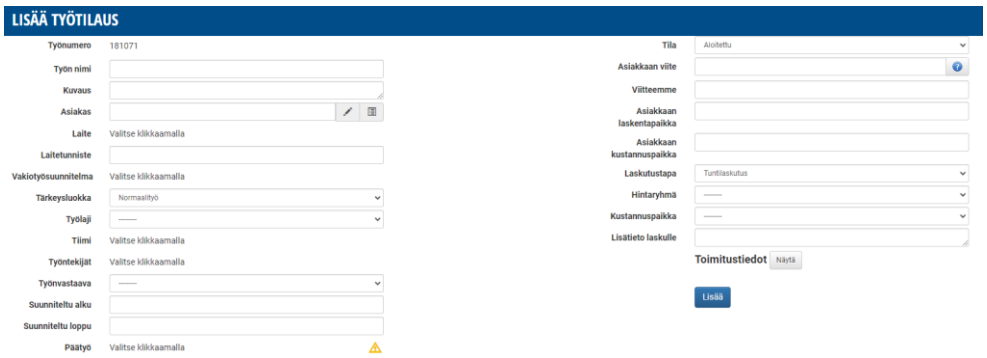
2 NYKYINEN JÄRJESTELMÄYMPÄRISTÖ

2.1 KUP-toiminnanohjausjärjestelmä

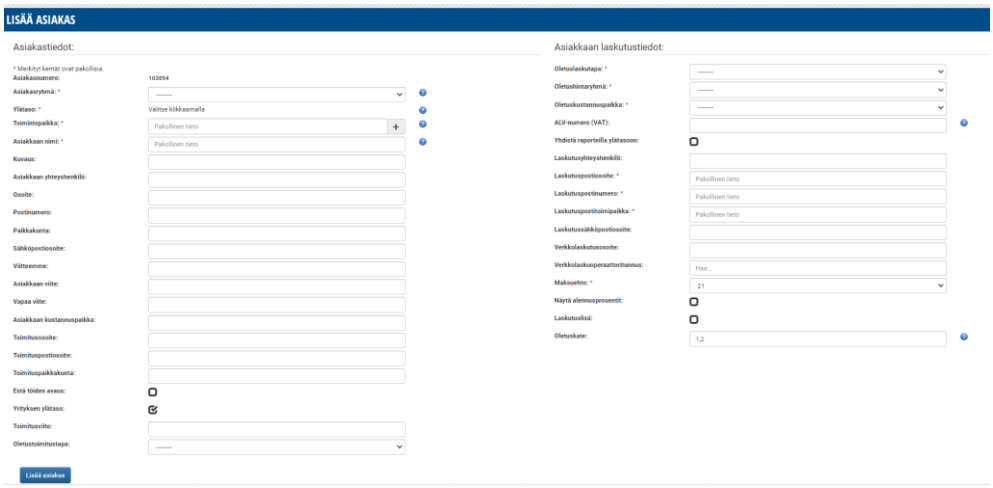
KUP on KP-ServicePartnerin itse omaan käyttöön tehty toiminnanohjausjärjestelmä, joka on julkaistu 2010. Se on toteutettu Pythonilla sitä käyttävään Django-ohjelmistokehykseen. Järjestelmää käytetään selaimella. Python ja

Django valikoituivat toteutukseen, koska taustatyön perusteella se vaikutti yhdistelmänä helpolta, koska Django sisältää niin monia helpottavia toimintoja, kuten ORM ja automaattisesti generoituva admin-sivu, joka helpotti alkuvaiheen käyttöä. Admin-sivu antaa käyttäjälle käyttöliittymän, jonka avulla tietokannan tietoja pääsee helposti katsomaan, lisäämään, poistamaan ja muokkaamaan. Hyviin puoliin lukeutui myös se, että sama ohjelmointikieli toimii front- ja backendissä. (Vuorinen, 2024)

KUP on toiminnoiltaan todella laaja, sen kautta hallitaan muun muassa asiakkuuksia, työtilauksia ja -tunteja. Tällä hetkellä jokainen uusi työ avataan ja asiakkuus perustetaan järjestelmään käsin. Sekä työn avaamiselle (Kuva 1) että asiakkuuksien perustamiselle (Kuva 2) löytyy oma lomake, johon tiedot täytetään. Sekä työn että asiakkaan lisääminen on työlästä käsin tehtynä.



Kuva 1. KUP-toiminnanohjausjärjestelmän työtilauksen lisäämisen lomake (KUP, 2024a)



Kuva 2. Lomake asiakkaan perustamista varten KUPissa (KUP, 2024b)

Jokaiselle avatulle työlle generoituu yksilöity numerosarja, työnumero, jonka avulla töitä on helppo etsiä toiminnanohjausjärjestelmästä. Työllä seurataan työn kulkua, tehtyjä tunteja ja käytettyjä materiaaleja, joita hyödynnetään laskutuksessa ja palkanmaksussa.

2.2 Microsoft

2.2.1 Entra ID

Entra ID on Microsoftin pilvipohjainen identiteettien- ja pääsynhallintapalvelu (identity and access management, IAM), jonka avulla käyttäjille mahdollistetaan erilaisia resursseja, kuten muun muassa Microsoft 365-palvelut. Entra ID mahdollistaa myös sovelluskehittäjille ulkoisten sovellusten kehittämiseen tarvittavien todennusten luomisen ja hallinnoimisen. (What is Microsoft Entra ID?, 2024)

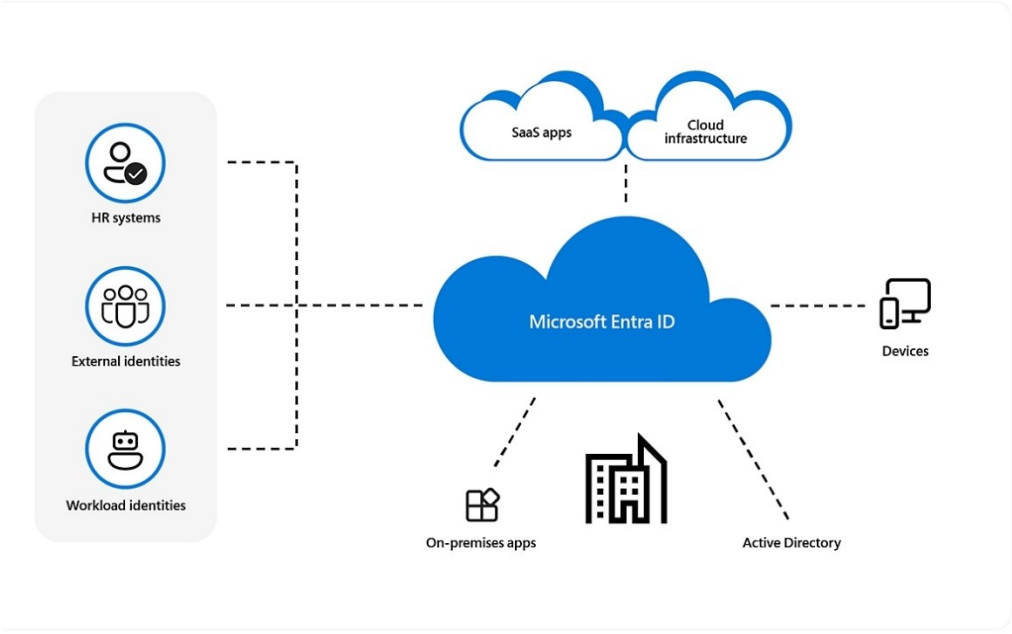
IAM-palveluilla varmistetaan, että organisaation eri resursseja päästään hallinnoimaan monipuolisesti ja valvotaan, että oikeat käyttäjät pääsevät oikeisiin resursseihin oikeaan aikaan ja oikeista syistä. Palvelun keskeisiä osa-alueita ovat:

1. Identiteetin ja käyttäjätilien hallinta: Prosessi, jossa luodaan, tallennetaan ja hallitaan identiteetti- ja käyttäjätietoja.
2. Autentikointi: Käyttäjiä, laitteita tai ohjelmistokomponentteja voidaan autentikoida. Yksittäisille käyttäjille voidaan lisätä monivaiheinen tunnistautuminen (MFA) turvallisuuden takaamiseksi.
3. Käyttäjien valtuutus: Valtuutus varmistaa, että käyttäjälle myönnetään täsmälleen se käyttö- ja pääsytaso, johon hän on oikeutettu. Käyttäjiä voidaan myös ryhmitellä tai määritellä rooleihin, jotta suurille käyttäjäryhmille voidaan myöntää samat oikeudet.
4. Pääsynvalvonta: Prosessi, jossa määritellään, kenellä tai millä on pääsy mihin resursseihin. Tämä sisältää käyttäjäroolien ja käyttöoikeuksien

määrittämisen sekä todennus- ja valtuutusmekanismien perustamisen. Pääsynvalvonta säätelee pääsyä järjestelmiin ja tietoihin.

5. Toiminnan seuranta: Raporttien luominen alustan toimista, kuten kirjautumisajasta, käytetyistä järjestelmistä ja todennustyypeistä, joilla varmistetaan vaatimustenmukaisuus ja arvioidaan turvallisuusriskejä. Tietoa saadaan ympäristön turvallisuudesta ja käyttötavoista. (What is identity and access management (IAM)?, 2024)

Kuten kuvassa 3 voi nähdä, Entra ID mahdollistaa organisaatiolle tavan hallinnoida ja turvata työntekijöiden, kumppaneiden ja asiakkaiden käyttäjätiedot, jotta eri sovellusten ja palveluiden käyttö olisi mahdollista. Entra ID tarjoaa kattavan identiteettiratkaisun, joka voidaan yhdistää vanhoista paikallisista sovelluksista ohjelmisto palveluna (SaaS) sovelluksiin, tarjoten saumattoman käyttäjäkokemuksen lisäksi paremman näkyvyyden ja hallinnan. (Microsoft, 2024a)



Kuva 3. Entra ID:n hallinnointimahdollisuudet. (Microsoft, 2024a)

2.2.2 SharePoint

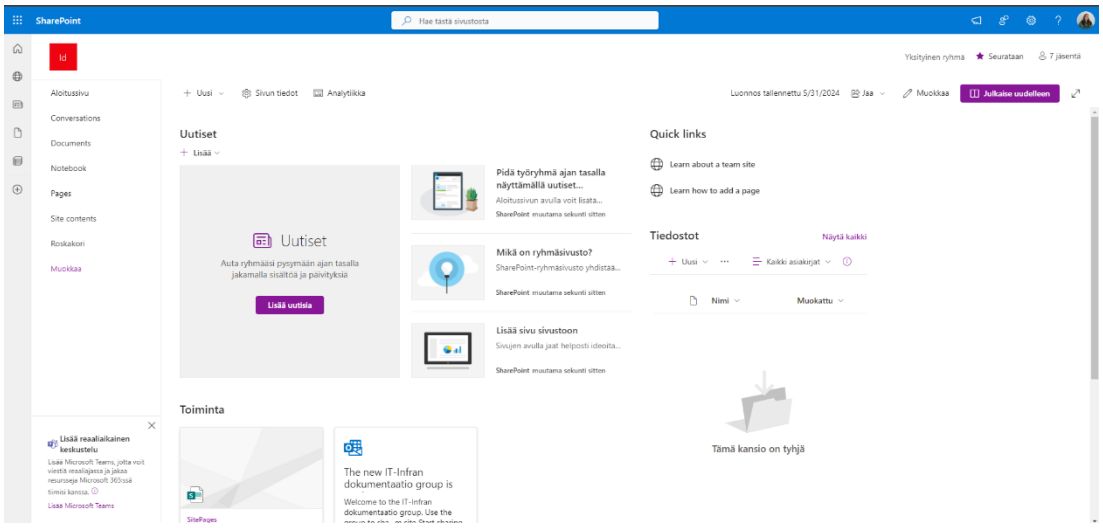
SharePoint on Microsoftin pilvipohjainen palvelu, joka soveltuu monenlaiseen käyttöön. Sen avulla voidaan luoda intranet-sivustoja, tehdä ryhmätyötä ja

jakaa ja hallita tietoa. SharePoint mahdollistaa myös erilaisten toiminnallisuuksien ja automaatioiden toteuttamisen. Sharepoint toimii siis joko valmiina ratkaisuna sellaisenaan tai kehitysympäristönä, jolla voi vastata erilaisiin yritysten tarpeisiin. (Roine & Anttila, 2015, s. 7)

SharePointissa on monia hyödyllisiä ominaisuuksia, kuten dokumenttien yhteinen reaaliaikainen muokkausmahdollisuus, hyvä hakuominaisuus ja helppo sisällönhallinta. Se tukee myös integraatiota muihin Microsoftin tuotteisiin, kuten Teamsiin, sekä työnkulkujen automaatiota, jolla voidaan automatisoida tehtäviä ja näin säästää aikaa. (Im, 2024)

SharePointilla voidaan tarkoittaa myös SharePoint Server ja SharePoint Designer 2013 palveluita. SharePoint Serverillä tarkoitetaan paikallisesti yrityksen omilla palvelimilla pyöritettävää SharePointia. Se voi olla hyvä ratkaisu suuremmille yrityksille, jotka haluavat hallinnoida koko ympäristöä ja olla itse vastuussa sen turvallisuudesta. SharePoint Designer 2013 on jo vanhentunut, vuonna 2013 julkaistu ohjelma, jolla tehtiin mm. työkulkuja. (Microsoft, 2024b)

SharePoint rakentuu sivustoista (sites). Sivustoja on kolmea erilaista, ryhmä-, viestintä- ja keskussivustoja. Ryhmäsivustot ovat tarkoitettu tiedon säilyttämiseen, jakamiseen, ja ryhmätyöskentelyyn (kuva 4). Viestintäsivustot ovat informaation, kuten uutisten, raporttien ja statuspäivitysten jakamiseen. Keskussivustot ovat tarkoitettu sivustojen organistointiin esimerkiksi aihealueittain, projektien tai osastojen perusteella. Keskussivustoja voi luoda vain SharePoint-järjestelmävalvojana. (Microsoft, 2024c)



Kuva 4. Ryhmä-sivusto, johon ei ole vielä tehty sisältöä, näkyvissä myös erilaisia mahdollisia osioita, mitä sivustolle voidaan rakentaa. (Oma SharePoint-sivusto, 2024)

Sivustoihin voidaan tehdä sivuja (pages), kuten siirtymäsivuja ja kohdesivuja. Sivut ovat verkkosivuja, joilla on yksi tarkoitus, esimerkiksi toimia ohjeena tai raportointityökaluna. Sivulle voidaan lisätä verkko-osia, kuten tekstiä, sovelluksia ja uutisvirtoja eri lähteistä. Kaikkia verkko-osia saa mukautettua ja aseteltua sivulle haluamallaan tavalla. Sivuja voidaan jakaa tai tallentaa pohjiksi tulevaa käyttöä varten. (Microsoft, 2024d; Microsoft, 2024e)

Luettelot (lists) ovat kokoelmia tiedoista, joka takaa juostavan tavan järjestellä tietoja. Microsoftilla on valmiina useita valmiita malleja, jotka helpottavat luettelon tekoa, mutta omien mukautettujen luetteloiden luominen on myös mahdollista. Monipuoliset ja joustavat luettelot tarjoavat monia sisäänrakennettuja ominaisuuksia, joiden avulla voi tallentaa, jakaa ja käsitellä tietoa. Luetteloihin voi lisätä monenlaisia sarakkeita ja luettelon kohteisiin saa liitettyä tiedostoja tarjoamaan lisätietoa. Erilaisilla näkymillä voi järjestää, lajitella ja suodattaa tietoja monella tavoin. Luetteloiden välille voi myös luoda suhteita, joka auttaa säilyttämään tietojen eheyden, esimerkiksi, jos tieto poistetaan yhdestä luettelosta, se poistuu myös toisesta. (Microsoft, 2024f)

Tiedostokirjasto (document library) on turvallinen paikka tallentaa tiedostoja ja ryhmätyöskennellä niiden parissa. Tiedostot ovat helposti löydettävissä ja

saatavilla kaikilla laitteilla. Jokaisen uuden sivuston yhteydessä luodaan automaattisesti sivustolle tiedostokirjasto ja uusia kirjastoja voidaan lisätä tarvittaessa. Jokaiselle tiedostokirjastolle voidaan antaa omat oikeudet, eli tarvittaessa voidaan rajata joihinkin kirjastoihin pääsyä. Tiedostokirjaston näkymä on luettelo tiedostoista, kansioista ja keskeisistä metatiedoista, kuten kuka tiedoston on luonut tai muokannut sitä viimeksi. Näitä tietoja voidaan käyttää tiedostojen järjestämiseen ja löytämisen helpottamiseen. (Microsoft, 2024g)

Asiakirjasarja (document set) järjestää useita toisiinsa liittyviä asiakirjoja yhdeksi näkymäksi, jossa niitä voidaan hallita ja käsitellä yhtenä kokonaisuutena. Asiakirjasarjalla on erityisominaisuuksia, jotka helpottavat niiden luomista ja hallintaa. Niille voidaan luoda mukautettu aloitussivu, johon voidaan määrittää näkymään halutut tiedot sekä sarjalle lisätyt asiakirjat. Asiakirjasarjalle voidaan määrittää sallitut sisältötyypit, joita voidaan käyttää sarjassa sekä oletussisältö, joka luodaan ja sisällytetään automaattisesti jokaiseen uuteen sarjaan, esimerkiksi joukko tiettyjä asiakirjoja tai kansiorakenteita. Sarjalle voidaan myös määrittää jaetut metatiedot ja metatietosarakkeet, joita halutaan käytettävän. Oletusasiakirjat varustetaan automaattisesti metatiedoilla, kun asiakirjasarja luodaan. Asiakirjasarjojen kanssa voidaan käyttää vakio- tai mukautettuja työnkulkuja ja koko sarja etenee työnkulun vaiheiden läpi kuin se olisi yksi tiedosto. (Microsoft, 2024h)

SharePointiin työnkulkuja voidaan luoda luetteloille ja kirjastoille Power Automaten avulla (Microsoft, 2024i). Power Automate on verkkopohjainen työnkulkujen hallintapalvelu, jonka avulla voidaan automatisoida toimintoja sovellusten ja palveluiden välillä (Microsoft, n.d.).

3 TEKNOLOGIAT JA TYÖKALUT

3.1 Microsoft Graph API

Microsoft Graph API julkaistiin vuoden 2015 lopussa (Foley, 2017). Se syntyi asiakkaiden tarpeesta, kun aikaisemmin jokaiselle pilvipohjaiselle Microsoftin palvelulle oli oma API, joista jokaisella oli omat kutsut, erilaiset vastaukset ja autentikoinnit. Näiden palveluiden opetteluun ja integroimiseen kului asiakkailta paljon resursseja. Microsoftin tuotteiden yhteen liitetyn luonteen takia ajatus kaikkien resurssien yhdistämisestä vaikutti parhaalta ratkaisulta ongelmaan. Näin syntyi Graph API. (Vasudevan, 2017)

Microsoft Graph API on RESTful verkkorajapinta, joka tarjoaa yhtenäisen ohjelmoinnillisen mallin. Microsoft Graph APIssa on yksi päätepiste (endpoint), http://graph.microsoft.com, jonka avulla voi käyttää Microsoftin eri palveluita. Näihin palveluihin kuuluu:

- M365-palvelut, kuten esimerkiksi SharePoint, Outlook ja Teams
- Enterprise Mobility + Security-palvelut, kuten Entra ID
- Windows-palvelut, kuten laitteet ja ilmoitukset. (Use the Microsoft Graph API, 2023; Overview of Microsoft Graph, 2024)

Graph APIn hyviin puoliin kuuluu se, että se on tietoturvallinen, sillä käyttääksesi sitä, on Entra ID:iin luotava sovellus, jolle annetaan oikeudet päästä resursseihin. Oikeudet tulisi aina määrittää tarpeiden mukaan. Sovellus tarvitsee käyttöä varten myös tunnisteen (token). (Celle, 2021)

Toinen hyvä puoli on yksi päätepiste. Jokaiseen kutsuun määritellään lisäksi käytettävä versio, 1.0 tai beta, sekä resurssi, johon halutaan päästä ja mahdolliset kyselyparametrit. Vastaus on JSON-muotoinen, ja sitä voidaan käyttää useimmissa ohjelmointikielissä. (Celle, 2021). Graph mahdollistaa eri HTTP-metodit, eli GET, POST, PATCH, PUT ja DELETE. (Use the Microsoft Graph API, 2023)

Graph API on myös monikäyttöinen, sillä useissa ohjelmointikielissä on mahdollista käyttää REST-arkkitehtuuria (REpresentational State Transfer). Microsoft tarjoaa myös useita koodiesimerkkejä ja SDK:ita (Software development kit). Graphia voi myös käyttää PowerAppsin ja Power Automaten kanssa tapauksissa, joissa liittimet eivät riitä. (Celle, 2021)

Graph Explorer on sovelluskehittäjille tarkoitettu verkkopohjainen työkalu kutsujen rakentamiseen ja testaamiseen. Siinä voi halutessaan käyttää oletuksena olevia tietoja tai kirjautumalla sisään omia tietoja. Graph Explorerin avulla voi kokeilla eri HTTP-metodien kutsuja ja muun muassa nähdä mitä valtuuksia tarvitset kutsuja suorittaaksesi. (Tools for interacting with Microsoft Graph, 2023)

3.2 Python

Python on monipuolinen ohjelmointikieli, joka tukee useita lähestymistapoja ohjelmointiin, kuten olio-ohjelmointia ja funktionaalista ohjelmointia. Se on myös yksi maailman suosituimmista kielistä. Pythonin suosio perustuu helppoon ja käyttäjäystävälliseen syntaksiin, monipuolisiin käyttömahdollisuuksiin, monipuoliseen valikoimaan kirjastoja ja ohjelmistokehyksiä, isoon yhteisöön, monipuolisiin käyttöohjeisiin sekä päivityksien ja parannuksien tiheyteen. (RedSwitches, 2024; McKeown, 2019)

Toisin kuin monet muut ohjelmointikielet, Python muistuttaa kirjoitusasultaan paljolti englantia, minkä takia sitä on helppo ymmärtää, ylläpitää ja kehittää jatkossa. Python on todella monipuolinen käyttömahdollisuuksien osalta, sitä voidaan käyttää muun muassa verkko- ja sovelluskehitykseen, data-analytiikkaan, koneoppimiseen ja tekoälyyn. Pythonissa on käytössä monipuolinen valikoima kirjastoja ja ohjelmistokehyksiä helpottamaan ja yksinkertaistamaan ohjelmointia. Kirjastot ovat valmiita koodeja, joita muut ohjelmoijat ovat tehneet nopeuttamaan ohjelmointia jatkossa. Ohjelmistokehykset, kuten Django tai Flask, helpottavat ja nopeuttavat verkkokehitystä ja takaavat turvallisuuden,

skaalautuvuuden ja ylläpidon. Pythonilla on myös iso yhteisö takana, jonka vuoksi ongelmatilanteisiin on helppo löytää ratkaisu. Kieli on ollut olemassa jo yli kolmekymmentä vuotta, joten erilaisia kursseja, tutoriaaleja ja ohjeita löytyy lukuisia. Pythonia myös kehitetään jatkuvasti ja havaittuja vikoja korjataan. (Malik, 2019; RedSwitches, 2024)

Pythonissa on myös omat ongelmansa. Sitä on kritisoitu useasti siitä, että se on hidas verrattuna joihinkin muihin kieliin, mutta tämä ei kuitenkaan yleensä näy loppukäyttäjälle. Toinen huono puoli on se, ettei se taivu kovin hyvin mobiilikehitykseen, koska Python-sovellukset yleensä vievät paljon muistia ja suoritinaikaa. Pythonissa on myös suunnittelurajoituksia, koska se on dynaamisesti tyypitetty. Tämä tarkoittaa sitä, että muuttujien tyypit arvioidaan ajon aikana, joka saattaa johtaa ajonaikaisiin virheisiin. Tämän takia Python-sovellukset vaativat enemmän testausaikaa. (Malik, 2019; McKeown, 2019)

3.3 Django

Django on korkean tason avoimen lähdekoodin Python-ohjelmistokehys (framework) verkkojärjestelmien toteuttamiseen. Djangon perustana on nopea kehittäminen mahdollisimman vähällä koodilla noudattaen DRY (don’t repeat yourself) eli älä toista itseäsi-periaatetta. Muita perusperiaatteita on selkeys, johdonmukaisuus ja se, ettei kehyksen eri kerrokset tiedä toisistaan, ellei se ole välttämätöntä. (Django Software Foundation, 2023)

Djangon perustuu MVT (Model-View-Template) arkkitehtuuriin. Mallit (model) käsittelevät tietokannan logiikkaa ja rakennetta. Näkymät (view) ovat tarkoitettu sovelluksen logiikalle ja toiminnallisuuksille. Mallipohjat (template) sisältävät käyttäjälle näkyvän sovelluksen ulkoasun ja rakenteen. Jokaisella osa-alueella on omat vastuunsa koodissa, mikä takaa hyvin järjestetyn ja ylläpidettävän rakenteen, varsinkin sovelluksissa, jotka ovat riippuvaisia suurista tietomääristä. (Gentile III, 2023)

Jokainen tietokantataulu on edustettuna Djangossa malliluokkana. Luokan attribuutit edustavat tietokantataulujen sarakkeita tai kenttiä ja jokaisen mallin yksittäinen instanssi edustaa kyseistä riviä tietokantataulussa. Näkymät toimivat välittäjinä mallien ja mallipohjien välissä. Näkymä vastaanottaa pyynnöt, käsittelee ne ja palauttaa tietyn vastauksen. Mallipohjat ovat verkkosovelluksen asettelu, ne luovat HTML-sivuja, jotka voivat käyttää dynaamista sisältöä käyttämällä malleista haettuja tietoja näkymien kautta. (Gentile III, 2023)

Objekti-relaatiomallinnus (Object-Relational Mapping, ORM) on ohjelmointitekniikka, jota käytetään tietokantatietojen tietojen muuntamisessa objekteiksi, joita voidaan käyttää olio-ohjelmointikielissä (Gentile III, 2023). Djangon ORM:n avulla tietokannan tiedot ovat Python-olioina. ORM kääntää Python-koodin SQL-kyselyiksi taustalla, mikä tekee tietokantatoiminnoista intuitiivisempaa ja vähemmän virhealtista. ORMissa on käytössä CRUD-operaatioiden (create, read, update, delete) lisäksi erilaisia filttereitä kyselyiden avuksi. Tarvittaessa Djangoon voidaan tehdä myös kyselyitä SQL-kielellä. (Codeafi, 2024)

3.4 OpenSSL

OpenSSL on avoimen lähdekoodin ohjelmistokirjasto, joka toteuttaa TLS (Transport Layer Security) ja SSL (Secure Sockets Layer) protokollia. TLS ja SSL ovat kryptografisia protokollia, jotka mahdollistavat turvallisen viestinnän tietoverkossa, kuten internetissä. OpenSSL tarjoaa laajan paketin salaukseen, salauksen purkuun ja kryptografisiin toimintoihin. Alun perin 1998 kehitetty ohjelmisto on noussut standardiksi SSL/TLS-protokollien toteutuksessa. Se tukee laajaa valikoimaa kryptografisia algoritmeja, mikä tekee siitä monipuolisen ja mukautettavan erilaisiin tietoturvavaatimuksiin. (Career Technology Cyber Security India Pvt. Ltd., 2023)

OpenSSL sisältää paljon hyödyllisiä ominaisuuksia, kuten turvallisten yhteyksien muodostamisen internetissä käyttämällä SSL/TLS-protokollia ja digitaalisten varmenteiden luomisen, allekirjoittamisen ja hallinnan. OpenSSL

sisältää laajan valikoiman kryptografisia algoritmeja salauksiin, salausten purkuun, digitaalisiin allekirjoituksiin, hajautukseen ja avainten luomiseen. Se mahdollistaa myös digitaalisten varmenteiden aitouden ja eheyden tarkistamisen, eli sillä voidaan tarkistaa, että varmenne on allekirjoitettu luotettavalla taholla. OpenSSL tarjoaa myös työkaluja avainten ja varmenteiden formaatin muuttamiseen. Sen lisäksi OpenSSL sisältää työkaluja SSL/TLS-yhteyksien testaamiseen ja vianmääritykseen, salausohjelmien tarkastamiseen, varmenteiden tietojen tutkimiseen ja verkon tietoturvaongelmien ratkaisemiseen. OpenSSL onkin laajasti käytössä erilaisissa sovelluksissa ja järjestelmissä, jotka vaativat turvallista viestintää, kryptografisia toimintoja ja varmenteiden hallintaa. (Sahu, 2013)

3.5 Sertifikaatilla tunnistautuminen

Autentikointi eli tunnistus on prosessi, jolla varmistetaan, että tunnistettava taho on todellakin se, joka väittää olevansa. Sertifikaattiin perustuva tunnistus vahvistetaan käyttämällä digitaalista varmennetta. Digitaalinen varmenne on kuin sähköinen passi, jota käytetään todistamaan taho vahvistamalla yksityinen avain. Tämä varmenne sisältää tunnistetiedot, julkisen avaimen tiedot sekä yksityisen avaimen. (Ping Identity, n.d.)

Jotta sertifikaatti toimisi oikein, yksityisen avaimen tiedot tulee vastata varmenteen julkista avainta. Yksityiset avaimet ovat aina ainutlaatuisia, ja ne käyttävät julkisen avaimen salausta tahon varmistamiseksi. Julkinen avain vahvistetaan sen suhteella yksityiseen avaimeen. Sertifikaatti takaa suuremman turvallisuuden autentikoimiseen, koska julkisen avaimen tiedot voidaan purkaa vain siihen linkitetyllä yksityisellä avaimella, ja jokainen yksityinen avain on ainutlaatuinen. (Ping Identity, n.d.)

4 RAJAPINNAN SUUNNITTELU

Toimeksiantajalla oli jo projektin alusta asti selkeä käsitys siitä, mitä ohjelman tulisi tehdä ja miten se tulisi toteuttaa. Koska toiminnanohjausjärjestelmä on Django-pohjainen, oli loogista toteuttaa ohjelma Pythonilla. Suunnitellessa ei ollut vielä varmaa, miten tiedot saisi parhaiten haettua SharePointista, mutta nopeasti selvisi, että Microsoft suosii Graph APIa ja kehittää sitä jatkuvasti, joten oli järkevintä pitkällä tähtäimellä toteuttaa ohjelma sillä. Tällä varmistettiin, ettei ohjelmaa tarvitse olla muutaman vuoden sisään kirjoittamassa uusiksi.

4.1 Tarjoustyökalu

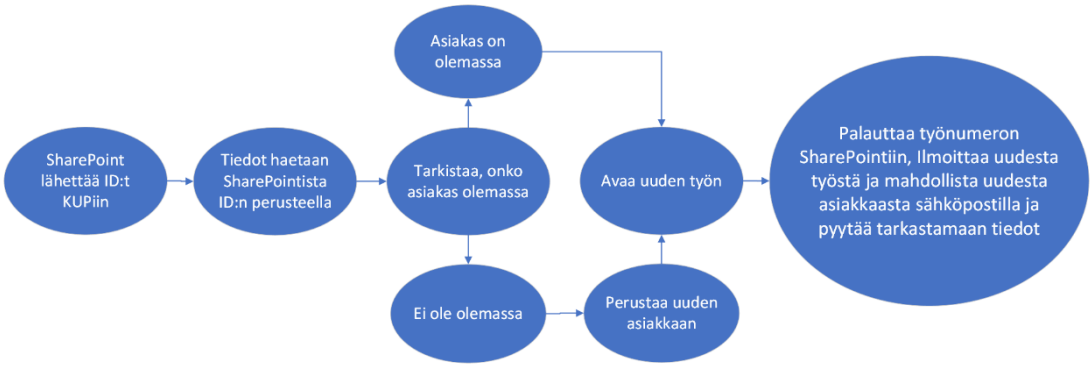
Tarjoustyökalu on SharePoint-pohjainen myynnin työkalu, jolla vastataan tulleisiin tarjouspyyntöihin ja tehdään tarjouksia. Tarjoustyökalu luo automaattisesti tarjousnumeron, tarjouksen pohjan ja täyttää asiakkaan ja myyjän tiedot tarjoukselle.

Kaikki tarjoukset arkistoidaan Arkisto-tiedostokirjastoon, joka on Tarjoustyökalu-sivustolla. Jokaisesta hyväksytystä tarjouksesta avataan uusi työ KUPiin. Isommista tarjouksista tehdään Projektit-sivustolle oma projektikansiorakenne, johon tulee kaikki projektiin liittyvät tiedot. Pienemmistä myynneistä, kuten esimerkiksi varaosista, ei tarvitse tehdä erillistä projektia. SharePointiin tallennetaan kaikki tiedostot, mitkä liittyvät töihin, joita seurataan KUPin avulla.

4.2 Toteutuksen kuvaus

SharePointista tuodaan ID-tiedot KUPin tietokantaan Microsoftin Power Automatella SharePointiin tehdyn automaation avulla. Kun tarjoustyökalusta aloitetaan tarjouksen arkistointi tai projektin teko, lähettää se samalla KUPin rajapinnalle tiedon Arkistoon ja Projekti-sivustoon tulevasta asiakirjasarjan ID:stä. Asiakirjasarjan ID:n perusteella haetaan siihen liittyvät metatiedot.

Kuvion 1 mukaisesti metatiedot haetaan SharePointista ja metatiedoista saadun asiakkaan nimen avulla tehdään tarkastus, onko asiakasta olemassa entuudestaan. Jos asiakasta ei löydy, perustetaan se toiminnanohjausjärjestelmään. Uuden asiakkaan perustamista varten metatiedoista tarvitaan asiakkaan nimi sekä asiakkaan yhteyshenkilö.



Kuvio 1. Ohjelman toiminnan vaiheet.

Jokaisessa asiakirjasarjassa on metatietoja, joiden avulla työ voidaan toiminnanohjausjärjestelmään avata. Näitä ovat työn nimi, kuvaus, asiakas, työlaji, asiakkaan viite, työnvastaava ja tarjouksen numero. Osa tiedoista on kuitenkin sellaisia, joita pitää käydä jälkikäteen lisäämässä avatulle työlle, kuten laitetiedot, tekijätiedot, päivämäärätiedot ja tarkemmat laskutustiedot. Osa pakollisista kentistä on myös sellaisia, mihin täytyy asettaa jokin tieto oletukseksi ja ne tarkastavan osapuolen täytyy käydä vaihtamassa oikeiksi, tällaisia kenttiä ovat muun muassa tila, laskutustapa, hintaryhmä ja kustannuspaikka.

Kun työ on avattu toiminnanohjausjärjestelmään, palauttaa se työn numeron SharePointiin Arkisto-tiedostokirjastoon ja Projektit-sivustolle, jos työstä on tehty projekti. Samalla lähetetään työnvastaavalle sähköpostilla tieto, että uusi työ, ja mahdollisesti uusi asiakas, on lisätty KUPiin. Kun työtä seurataan KUPissa ja työhön liittyviin tiedostoihin pääsee helposti käsiksi toiminnanohjausjärjestelmän kautta, helpottaa se koko yrityksen toimintaa.

5 RAJAPINNAN TOTEUTUS

5.1 Sovelluksen rekisteröinti ja autentikointi

Entra ID:ssä voidaan rekisteröidä uusi sovellus New Applicationilla. Tätä työtä varten käytettiin jo valmiina löytyvää sovellusta, jonka eräs edellinen työntekijä oli tehnyt. Sovellukselle annettiin oikeuksiksi Microsoft Graph, application permissionina Sites.ReadWrite.All, mikä sallii sovelluksen lukea ja kirjoittaa kaikkiin SharePoint-sivustoihin kyseisellä vuokraajalla (tenant). Sovelluksesta saadaan tiedot sovelluksen ID:stä (client id) ja vuokraajan ID:stä (tenant id), joita tarvitaan sovelluksen tunnistamiseen ja ohjaamiseen oikeisiin tietoihin. Jokaisella saman vuokraajan sovelluksella on aina sama vuokraajan ID.

Sovelluksien tunnistamiseen voi käyttää joko client secretia (asiakassalasana) tai sertifikaattia. Tunnistamisella todennetaan, että sovelluksella on oikeus saada access token eli käyttöoikeustunnus. Käyttöoikeustunnus on tietoturvallinen koodi, jolla voidaan todentaa ja saada pääsy resursseihin tai palveluihin. Microsoftin myöntämät tokenit ovat yleensä tunnin voimassa, jonka aikana kirjautuminen tulee tapahtua. Tunnuksen saa haettua suoraan Entra ID:stä tai sen voi myös saada ohjelmallisesti ja automaattisesti, kun tunnistukseen käytettävät tiedot ovat oikein.

Microsoftin luoma client secret on 40 merkin mittainen satunnaisesti generoitu merkkijono, joka toimii salasanana sovellukselle. Client secretin ongelmana on, että se on käytännössä kuitenkin vain salasana, joka saattaa olla selvitettävissä. Sertifikaatit sen sijaan ovat luotettavampia ja vaikeammin selvitettävissä. Työllä käytettiin itseallekirjoitettua sertifikaattia. Sen etuja ovat kustannustehokkuus ja se, että se on helppo ja nopea saada, varmenneviranomaisen allekirjoittamat sertifikaatit maksavat ja varmennusprosessi on monimutkainen. Haittana kuitenkin on, että se ei ole yhtä turvallinen ja luotettava kuin varmenneviranomaisen allekirjoittama sertifikaatti. (Venafi, 2024.) Päädyimme kuitenkin itseallekirjoitettuun sertifikaattiin, koska ohjelma pyörii turvallisen ja rajatun palvelimen ja Microsoftin turvallisen pilvipalvelun välillä.

Sertifikaatti tehtiin OpenSSL:llä käyttämällä ohjelman 1 komentoa. Komento luo uuden 2048-bittisen RSA-avaimen ja itseallekirjoitetun SSL/TLS-varmenteen, joka on voimassa 10 vuotta. Avain tallennetaan key.pem-tiedostoon ja itse sertifikaatti cert.pem-tiedostoon.

```
openssl req -newkey rsa:2048 -new -nodes -x509 -days 3650
-keyout key.pem -out cert.pem
```

Ohjelma 1. Komento sertifikaatin muodostamiseksi. (OpenSSL, 2024)

Autentikoimiseen tarvittavia tietoja ovat authority, scopes ja tunnistautumistavan mukaan joko client secret tai sertifikaatin avain ja sormenjälki (thumbprint), jonka saa lisäämällä sertifikaatin Entraan. Authority sisältää URL-osoitteen, jota käytetään autentikointiin. Osoitteena käytettiin https://login.microsoftonline.com/{tenant_id}, joka ohjaa kirjautumaan Microsoftin palveluun tenantin tunnisteella. Scopes määrittää sovelluksen resurssit ja toiminnot, kuten ["https://graph.microsoft.com/.default"], jossa .default antaa valtuudet Entra ID:ssä määriteltyihin resursseihin. Näin koodiin ei tarvitse käydä joka kerta päivittämässä valtuuksia, jos niitä lisätään tai poistetaan sovellukselta. Tunnistautumistiedot ovat koodissa dictionaryna, joka sisältää sertifikaatin tiedot, yksityisen avaimen ja sertifikaatin sormenjäljen. Näitä tietoja tarvitaan access tokenin hakuun.

Access token haettiin MSAL-kirjastoa (Microsoft Authentication Library) käyttäen (kuva 5). MSAL-kirjastosta käytettiin ConfidentialClientApplication-luokkaa, joka vaatii autentikointiin tarvittavia tietoja toimiakseen. Luokan avulla sovellus tunnistautuu Microsoftille. Luokasta löytyy acquire_token_for_client-metodi, joka hakee access tokenin aktiiviselle luottamukselliselle sovellukselle. Metodi hakee uuden access tokenin vain, jos se ei löydä olemassa olevaa tokenia välimuistista, mikä karsii tarpeen erilliselle refresh tokenin haulle.

```
# Luodaan MSAL Client
app = msal.ConfidentialClientApplication(
    client_id=client_id,
    authority=authority,
    client_credential=cert
)

# Haetaan Access token
result = app.acquire_token_for_client(scopes=scopes)

if "access_token" in result:
    access_token = result['access_token']
    print("Access token acquired!")
else:
    print("error acquiring token: {result.get('error_description')}")
    exit()
```

Kuva 5. MSAL Clientin luonti ja access tokenin hankinta.

5.2 SharePointin tietojen määritys ja endpointit

Jokaisella SharePoint-sivustolla, sivulla ja sisällöllä on oma ID. Tässä työssä käytettiin Tarjoustenhallinta-sivuston Arkisto-tiedostokirjaston ja Projektit-sivuston ID-tietoja sekä niihin liittyvien luetteloiden ID-tietoja. Nämä tallennettiin dictionaryyn Tarjoustenhallinta-luokassa, jotta tiedot ovat helposti päivitettävissä ja käytettävissä kaikissa metodeissa.

ID-tietojen selvitys oli työlästä. Arkiston ja Projektit-sivuston ID:t saatiin SharePointin administa. Luetteloiden ID:t löydettiin Graph Explorerilla GET-kutsulla osoitteesta https://graph.microsoft.com/v1.0/sites/site-id/lists. Tuloksena saatiin kaikkien sivuston luetteloiden tiedot, mukaan lukien ID-tiedot.

Endpointin määrittämisessä käytettiin apuna Graph Exploreria. Graph Explorerilla voidaan valita haluttu HTTP-metodi ja endpoint sekä testata sen toimivuutta. Työtä varten suoritettiin testejä pelkästään GET-kutsulla, jotta käytössä olevassa Tarjoustyökalussa oleva tieto ei muutu. Tarvittaessa Graph Explorerilla voi käyttää myös muita kutsuja.

Tätä työtä varten tehtiin kaksi eri metodia. Yhden hakemaan tiedot SharePointista ja toisen palauttamaan tietoa sinne. Aluksi tarkoitus oli hakea kaikki metatiedot, mutta ongelmaksi muodostui, että tarjouksen käsittelijä tuli henkilöobjektin ID-numerona. Ongelma ratkaistiin rajaamalla haku vain tarvittaviin tietoihin. Kuvan 6 mukainen haku tuotti kaikki tarvittavat metatiedot: tarjouksen nimi, kuvaus, käsittelijä, asiakkaan viite, nimi, tarjouksen numero, työlaji ja yhteyshenkilö.

```
def get_part_of_metadata(site_id,list_id,item_id): # Hakee osan itemin metatiedoista
    endpoint = f'https://graph.microsoft.com/v1.0/sites/{site_id}/lists/{list_id}/items/{item_id}?
    expand=fields(select=Title,DocumentSetDescription,Tarjouksen_x0020_k_x00e4_sittelij_x00e4_,
    Asiakkaan_x0020_viite,Asiakkaan_x0020_nimi1,Tarjouksen_x0020_numero,Ty_x00f6_laji_x0020__x0028_KUP_x0029_,
    Asiakkaan_x0020_yhteyshenkil_x00f6_)'
    response = requests.get(endpoint,headers=headers)

    if response.status_code == 200:
        return response.json()
    else:
        return {'error': 'Request failed with status code ' + str(response.status_code)}
```

Kuva 6. Metatietojen hakutapa.

Kuva 7 metodi palauttaa tietoa takaisin halutulle kohteelle. Palautettavien tietojen täytyy olla JSON-muodossa ja tiedoille täytyy olla SharePointissa valmiina paikat, mihin ne asetetaan. Tässä tapauksessa SharePointin Arkisto-tiedostokirjaston ja Projektit-sivuston asiakirjasarjojen metatiedoista löytyi kenttä KUP-työnumerolle.

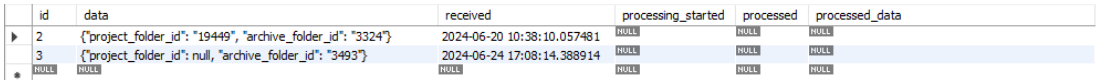
```
def patch_to_sharepoint(site_id, list_id, item_id, data): # Palauttaa dataa itemille
    endpoint = f'https://graph.microsoft.com/v1.0/sites/{site_id}/lists/{list_id}/items/{item_id}/'
    response = requests.patch(endpoint, headers=headers, json=data)

    if response.status_code == 200:
        return True
    else:
        return False
```

Kuva 7. Tietojen tallennus SharePointiin.

5.3 Tietojen prosessointi

KUP tallentaa SharePointin lähettämät ID-tiedot tietokantaansa. Data-sarakkeeseen tallennetaan Arkisto-tiedostokirjaston asiakirjasarjan ID:n lisäksi Projektit-sivuston asiakirjasarjan ID, jos tarjouksesta on tehty projekti. Jos projektia ei ole, SharePoint lähettää ”null” tyhjän sijaan varmistaakseen tiedon

puuttuvan tarkoituksella. Tietoihin tulee myös aikaleima vastaanottoajankohdasta. Kuva 8 havainnollistaa esimerkkejä sekä projektin sisältävistä että ilman projektia olevista tarjouksista.



id	data	received	processing_started	processed	processed_data
2	{"project_folder_id": "19449", "archive_folder_id": "3324"}	2024-06-20 10:38:10.057481	NULL	NULL	NULL
3	{"project_folder_id": null, "archive_folder_id": "3493"}	2024-06-24 17:08:14.388914	NULL	NULL	NULL
NULL	NULL	NULL	NULL	NULL	NULL

Kuva 8. KUPin tietokantanäkymä SharePoint-tiedoista. (KP-ServicePartner Oy, 2024a)

For-silmukkaa käytetään käymään läpi kaikki tietokannan tiedot, joissa processed-kentän arvo on ”null”. Jokaisesta objektista tallennetaan muuttujille objektin ID sekä arkiston ja projektin asiakirjasarjojen ID-tiedot. Tämän lisäksi asetetaan aikaleima processing_started-kenttään, jotta käsittelyn aloitusajankohta voidaan määrittää.

Prosessointi hoidettiin kahdella metodilla, toinen arkiston tietoja varten ja toinen projektia varten. Metodeissa haettiin tarvittavat tiedot GET-metodilla ja palautuneet tiedot tallennettiin muuttujiin jatkokäsittelyä varten. Tarjouksen käsittelijää lukuun ottamatta tiedot olivat suoraan käytettävissä sellaisessa muodossa, kun ne saadaan. Käsittelijä täytyi jakaa split-metodilla kahtia, ennen kuin vastaavuutta voitiin hakea tietokannasta (ohjelma 2). Hakuun käytettiin iexact-parametria, jotta mahdolliset kirjainkokojen vaihtelut eivät vaikuttaisi tulokseen.

```
split_name = tarjouksen_kasittelija.split()
supervisor_obj = hlot.objects.get(firstname__ie-
xact=split_name[0], lastname__iexact=split_name[1])
```

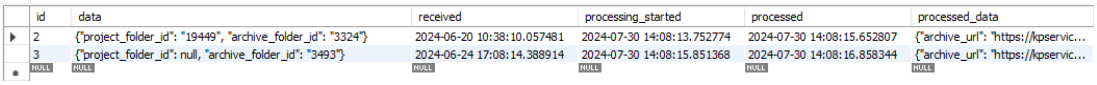
Ohjelma 2. Käsittelijän nimen haku get-metodilla ja iexact-parametrillä.

Asiakkaan haku tehtiin Djangon get_or_create-metodilla käytäen iexact-parametria, joka ensin hakee asiakasta nimellä, ja jos sellaista ei löydy, perustaa se uuden asiakkaan. Uutta asiakasta varten määriteltiin joitakin pakolliseksi merkittyjä tietoja.

Uusi työ avattiin käyttämällä Djangon objektin create-metodia. Uuteen työhön käytettiin SharePointista saatuja tietoja, kuten nimeä, kuvausta, asiakkaan viitettä ja käsittelijän nimeä. Asiakkaan tieto haettiin edellisestä asiakkaan hakumetodista. Työlle asetettiin myös tietokannan objektin ID, jotta työt voidaan tarvittaessa yhdistää tietokannan tietoihin. Jokaisella uudella työllä on yksilöllinen työnumero.

Prosessointiin käytetyt metodit lähettävät PATCH-metodilla yksilöllisen työnumeron SharePointiin. Tämän lisäksi metodit palauttavat avatun työn objektin, arkiston ja projektin URL-osoitteet sekä tiedon, perustettiinko uusi asiakas.

Kun molemmat metodit on suoritettu, silmukka päivittää tietokantaan processed_data-kenttään arkiston ja projektin URL-osoitteet sekä processed-kenttään prosessoinnin suoritusajan. Jos tarjouksesta ei ole tehty projektia, processed_data-kenttään tallennetaan vain arkiston asiakirjasarjan URL-osoite ja projektista tieto ”null”. Tämä osoittaa, että kenttä on tarkoituksella tyhjä eikä virheellinen. Kuvasta 9 käy ilmi, että prosessoinnin eri vaiheisiin on lisätty aikaleimat ja processed_data-kenttään on kertynyt tietoa.



	id	data	received	processing_started	processed	processed_data
▶	2	{"project_folder_id": "19449", "archive_folder_id": "3324"}	2024-06-20 10:38:10.057481	2024-07-30 14:08:13.752774	2024-07-30 14:08:15.652807	{"archive_url": "https://kpservic...
	3	{"project_folder_id": null, "archive_folder_id": "3493"}	2024-06-24 17:08:14.388914	2024-07-30 14:08:15.851368	2024-07-30 14:08:16.858344	{"archive_url": "https://kpservic...
*	NULL	NULL	NULL	NULL	NULL	NULL

Kuva 9. Näkymä tietokannasta prosessoinnin jälkeen. (KP-ServicePartner Oy, 2024b)

5.4 Sähköpostin lähetys

Seuraavana vaiheena oli lähettää sähköposti-ilmoitus käsittelijälle. Sähköpostin lähettämiseen oli olemassa oleva koodi, jota hyödynnettiin tässä projektissa. Metodi, jolla sähköposti lähetetään, saa parametrina avatun työobjektin, tiedot asiakkaan perustamisesta sekä Arkisto-tiedostokirjaston ja Projektit-sivuston asiakirjasarjojen URL-osoitteet.

Sähköpostiin sisältyvät aina työn nimi, työn numero ja arkiston URL-osoite. Mikäli uutta asiakasta ei ole perustettu ja tarjouksesta ei ole muodostettu

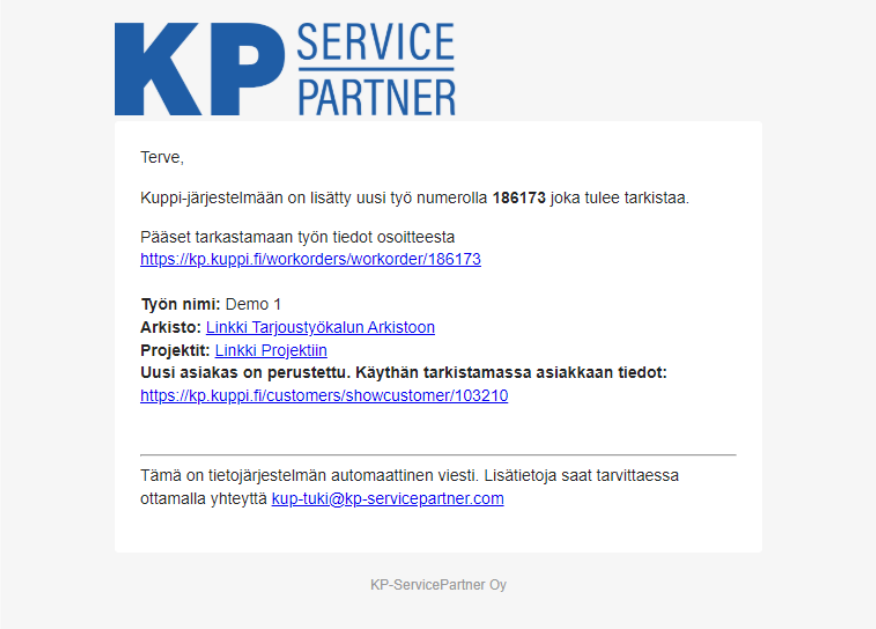
projektia, tiedot välitetään eteenpäin tyhjänä. Mikäli uusi asiakas on perustettu ja työstä on luotu projekti, nämä tiedot sisällytetään lisäksi sähköpostiin. Tiedot viedään HTML-tiedostoon, jossa sähköpostin varsinainen ulkoasu muotoillaan.

HTML-tiedostossa tarkistetaan, onko Projektin URL-tietoa olemassa. Jos tieto löytyy, lisätään linkki sähköpostiin (kuva 10). Vastaava tarkistus suoritetaan asiakkaan perustamisen yhteydessä; jos "created"-tieto on "True", lisätään tieto asiakkaan perustamisesta sähköpostiin. Sähköpostipohja oli jo valmiiksi olemassa.

```
{% extends 'email/base.html' %}
{% block content %}
<p>Kuppi-järjestelmään on lisätty uusi työ numerolla <b>{{wonum}}</b> joka tulee tarkistaa.</p>
<p>
Pääset tarkastamaan työn tiedot osoitteesta <a href="{{baseurl}}/workorders/workorder/{{wonum}}">{{baseurl}}/workorders/workorder/{{wonum}}</a><br>
<br>
    <b>Työn nimi:</b> {{ title }}<br>
    <b>Arkisto:</b> <a href="{{ archive_url }}">Linkki Tarjoustyökalun Arkistoon</a><br>
    {% if project_url != None %}
    <b>Projektit:</b> <a href="{{ project_url }}">Linkki Projektiin</a><br>
    {% endif %}
    {% if created == True %}
    <b>Uusi asiakas on perustettu. Käythän tarkistamassa asiakkaan tiedot:</b> <a href="{{baseurl}}/customers/showcustomer/{{customer}}">{{baseurl}}/customers/showcustomer/{{customer}}</a><br>
    {% endif %}
</p>
{% endblock %}
```

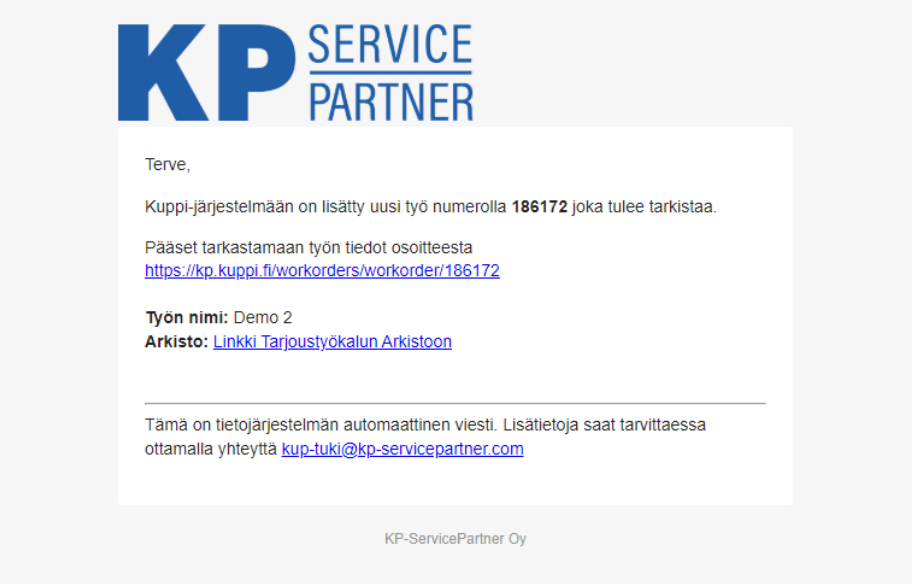
Kuva 10. Sähköpostin HTML-tiedosto

Kuvassa 11 on esimerkki sähköpostista, jossa asiakas on perustettu ja projekti on tehty. Sähköposti on lyhyt, informatiivinen ja selkeä.



Kuva 11. Esimerkki sähköpostista, jossa on perustettu uusi asiakas ja työstä on tehty projekti. (KP-ServicePartner Oy, 2025a)

Jos tarjouksessa on ollut jo olemassa oleva asiakas, ja työstä ei ole tehty projektia, sähköposti näyttää kuvan 12 kaltaiselta.



Kuva 12. Esimerkki sähköpostista, jossa asiakas on olemassa ja työstä ei ole tehty projektia. (KP-ServicePartner Oy, 2025b)

5.5 Linkin lisäys työn lisätiedot-välilehdelle

Lisätiedot-välilehteen liittyen on käytettävissä metodi, jonka avulla sinne voidaan lisätä tietoja. Kyseiseen metodiin on lisätty osa, joka näkyy kuvassa 13. Tämä osa hakee processed_data-kentän tiedot kaikista työobjekteista, joiden external_api-kentässä on arvo "sharepoint_tarjousten_hallinta". Haussa käytetään ID-tunnistetta, eli työobjektin external_api_id-kentän arvon tulee vastata tietokannan rivin ID:tä. URL-tiedot, jotka tallennetaan tietokantaan, otetaan talteen muuttujaan, jota käytetään HTML-tiedostossa.

```
if workorder_object.external_api == 'sharepoint_tarjousten_hallinta':
    sharepoint_data = SharepointIncomingWebhookData.objects.get(id=workorder_object.external_api_id)
    sharepoint_urls = sharepoint_data.processed_data
```

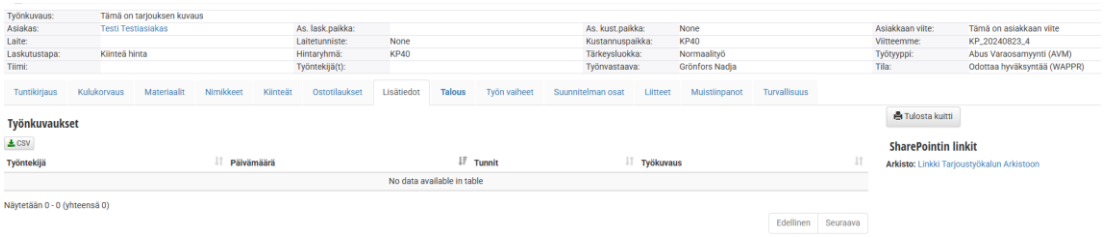
Kuva 13. Processed_data-kentän tietojen hakeminen tietokannasta.

HTML-tiedostoon lisättiin yksinkertainen koodi kuvan 14 mukaisesti. Mikäli sharepoint_urls ei ole arvossa ”None” eli se ei ole tyhjä, arkiston linkki lisätään työn Lisätiedot-välilehdelle. Tämän jälkeen tarkistetaan, onko projektin URL-tieto saatavilla, ja jos se on olemassa, se lisätään myös näkymään kyseisellä välilehdellä.

```
{% if sharepoint_urls != None %}
    <h2>SharePointin linkit</h2>
    <b>Arkisto:</b> <a href="{{ sharepoint_urls.archive_url }}">Linkki Tarjoustyökalun Arkistoon</a><br>
    {% if sharepoint_urls.project_url != None %}
        <b>Projektit:</b> <a href="{{ sharepoint_urls.project_url }}">Linkki Projektiin</a><br>
    {% endif %}
{% endif %}
```

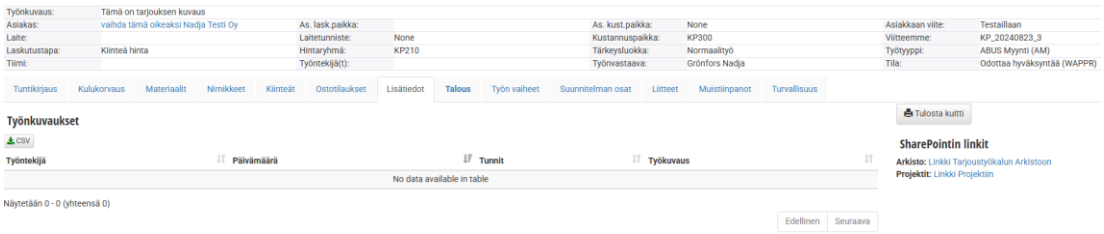
Kuva 14. HTML-tiedoston osio, joka käsittelee linkkien lisäämistä lisätiedot-välilehdelle.

Linkit näkyvät lisätiedot-välilehden oikeassa reunassa samalla tavalla kuin lähtevässä sähköpostissa. Jos projektia ei ole tehty, välilehdellä on vain arkisto-linkki kuvan 15 mukaisesti.



Kuva 15. Linkki arkistoon. (KUP, 2025a)

Jos työstä on luotu projekti, Lisätiedot-välilehdellä näkyy sekä arkiston linkki että linkki SharePointissa sijaitsevaan projektikansioon (kuva 16).



Kuva 16. Arkiston ja projektin linkit. (KUP, 2025b)

5.6 Virheidenkäsittely

Djangon transaction atomic -konseptia hyödynnetään virheidenkäsittelyssä (kuva 17). Tämä tarkoittaa, että tietokantatoimenpiteet suoritetaan atomisesti, eli kaikki toimenpiteet joko suoritetaan onnistuneesti tai mitään niistä ei suoriteta. Kyseisessä tapauksessa koodilohko käsitellään yhtenä kokonaisuutena: jos jokin lohkon osista epäonnistuu, kaikki muutokset perutaan (rollback). Jos lohko suoritetaan loppuun onnistuneesti, muutokset hyväksytään (commit). Lisäksi mahdolliset virheet kirjataan lokiin, jotta voidaan myöhemmin tarkastaa virheen syy ja ryhtyä toimenpiteisiin estääkseen sen toistumisen.

```
try:
# Käytetään atomicia, jos tapahtuu virheitä ei kannan tietoja päivitetä
with transaction.atomic():
    workorder_obj, archive_url, created = self.process_archive(archive_id,api_id) # arkiston prosessointi
    if project_id is not None:
        project_url = self.process_project(workorder_obj.wonum, project_id) # projektin prosessointi
    processed_data = {
        'archive_url' : archive_url,
        'project_url' : project_url
    }
    unprocessed_obj.processed_data = processed_data # asetetaan kantaan menevät url-tiedot
    unprocessed_obj.processed = datetime.now() # asetetaan aikaleima, kun prosessointi on lopetettu
    unprocessed_obj.save()
    self.send_email(workorder_obj, created, archive_url, project_url)
except Exception as e:
ue_logger.exception(e)
unexpected_error = True
```

Kuva 17. Transaction atomicin avulla suoritettava virheidenhallinta.

6 TULOKSET JA POHDINTA

Ohjelmaa testattiin useaan otteeseen testikantaa vasten kehitysvaiheessa. Osa uuden asiakkaan ja työn perustamiseen vaadittavista tiedoista on pakollisia, ja niitä tunnistettiin testauksen avulla. Virheilmoitusten kautta selvisi, mitkä tiedot olivat pakollisia Lopulta, kun kaikki pakolliset tiedot olivat oikein, voitiin todeta, että ohjelma toimi kuten pitäisi. Sähköpostin lähetys ja KUPin näkymien tiedettiin jo toimivan entuudestaan, ja jokaisessa testissä saatiinkin oikeanlaiset näkymät toiminnanohjausjärjestelmään sekä sähköpostiviestit.

Ohjelman suurin riski liittyy tarjoustyökalun käyttäjien mahdollisiin virheisiin, koska kaikki metatiedot täytetään manuaalisesti. Tarjoustyökaluun täytetään lomake, jonka avulla generoidaan tarjous, ja metatiedot perustuvat tämän lomakkeen tietoihin. Aluksi sähköposti-ilmoituksia tarkkailtiin sen varmistamiseksi, ettei olemassa olevia asiakkaita perusteta uudelleen toiminnanohjausjärjestelmään. Tästä syystä sähköposti-ilmoitukset lähetettiin myös tietohallinnon sähköpostiin.

Ohjelman toimintaperiaate esiteltiin tarjoustyökalun käyttäjille, jotta he ymmärtäisivät sen toiminnan sekä heidän toimiensa vaikutukset suorittamiseen. Käyttäjille korostettiin oikeinkirjoituksen tärkeydestä metatietojen kohdalla.

Projektin alussa Microsoft Graph API ja Django olivat minulle uusia teknologioita, joten käytin aikaa tutustuakseni niiden dokumentaatioihin. Tein myös erilaisia testejä ja omia projekteja, jotta oppisin käyttämään niitä. Aluksi kaikki yksityiskohdat ohjelman toiminnasta eivät olleet selkeitä, ja käytin jonkin verran aikaa selvittääkseni asioita, jotka eivät lopulta toteutuneet.

Aluksi ei ollut selkeää, että automaatiota käytettäisiin tietojen siirtämiseen toiminnanohjausjärjestelmään. Erilaisia ratkaisuja tietojen siirtoon tuli pohdittua laajasti. Yhtenä ratkaisuna olisi ollut jatkuva Arkiston ja Projektien tietojen läpikäynti ja jos KUP-työnumero-kentässä ei olisi ollut tietoa, olisi se otettu käsittelyyn.

Projektin aikana heräsi paljon ajatuksia, miten integraatiota SharePointin ja KUPin välillä voisi hyödyntää tulevaisuudessa. Yrityksen henkilöstö käyttää pääsääntöisesti toiminnanohjausjärjestelmää kaikkeen tekemiseen ja Microsoftin ollessa vielä uusi järjestelmänä, voi SharePoint tuntua hankalalta varsinkin asentajille, jotka viettävät työpäivänsä muualla kuin tietokoneella. Työskentelyä helpottavia toimenpiteitä tuli mieleen useita, kuten saumaton pääsy mobiililaitteella toiminnanohjausjärjestelmästä SharePointiin ja mahdolliset kansionäkymät toiminnanohjausjärjestelmän puolelle.

Kaikkien asiakkaiden SharePoint-kansiot voisi olla KUPissa asiakasnäkymässä näkyvissä, jotta asiakkaille tehtyjä tarjouksia pääsisi katsomaan helposti. KUPissa on listattuna myös eri laitteet ja laitteiden ohjeet löytyvät jatkossa SharePointista. Asentajien työtä helpottaisi huomattavasti, jos laitteen ohjeet olisivatkin suoraan avattavissa laitteen näkymästä toiminnanohjausjärjestelmässä. Ohjeiden lisäksi laitteille olisi hyvä saada suoraan näkyviin laitteiden huoltoraportit ja muut tiedostot, jotka liittyvät laitteeseen. Tätä projektia voisi jatkossa kehittää niin, että KUPissa töillä näkyisi projektin tiedostot linkkien sijaan.

LÄHTEET

Career Technology Cyber Security India Pvt. Ltd. (2023). OpenSSL. Haettu 16.7.2024 osoitteesta https://medium.com/@careertechnologymira-road/openssl-75444bc7f3e6

Celle, D. (2021). Why and how to use Microsoft Graph API? Haettu 11.7.2024 osoitteesta https://www.smartview.fr/en/utiliser-microsoft-graph-api/

Cloudflare. (n.d.). What is TSL (Transport Layer Security)? Haettu 30.5.2024 osoitteesta https://www.cloudflare.com/learning/ssl/transport-layer-security-tls/

Codealfi. (2024). Unlocking the power of Django ORM: A comprehensive guide to understanding different methods. Haettu 22.7.2024 osoitteesta https://medium.com/@codealfi/unlocking-the-power-of-django-orm-a-comprehensive-guide-to-understanding-different-methods-3c2ccfd48571

Django Software Foundation. (2023). Design philosophies. Haettu 9.7.2024 osoitteesta https://docs.djangoproject.com/en/5.0/misc/design-philosophies/

Foley, M. (2017). Microsoft Graph APIs: The Glue That's Starting to Stick. Haettu 11.7.2024 osoitteesta https://redmondmag.com/articles/2017/07/01/graphapis.aspx

Gentile III, A. (2023). Basics of Django: Model-View-Template (MVT) architecture. Haettu 19.7.2024 osoitteesta https://angelogentileiii.medium.com/basics-of-django-model-view-template-mvt-architecture-8585aecffbf6

Gitlan, D. (2024). What is OpenSSL? How does OpenSSL work? Haettu 30.5.2024 osoitteesta https://www.ssldragon.com/blog/what-is-openssl/

Im, H-J. (2024). What is SharePoint? Understand Microsoft SharePoint In Plain English. Haettu 5.7.2024 osoitteesta https://www.proser-veit.com/blog/what-is-sharepoint-in-plain-english

KP-ServicePartner. (n.d.). Tietoa meistä. Haettu 30.5.2024 osoitteesta https://kp-servicepartner.com/tietoa-meista/

KP-ServicePartner Oy. (2024a). KUP-tietokanta. Pääsy rajattu vain tietyille käyttäjätunnuksille.

KP-ServicePartner Oy. (2024b). KUP-tietokanta. Pääsy rajattu vain tietyille käyttäjätunnuksille.

KP-ServicePartner Oy. (2025a). Järjestelmän automaattinen sähköposti-ilmoitus käyttäjälle.

KP-ServicePartner Oy. (2025b). Järjestelmän automaattinen sähköposti-ilmoitus käyttäjälle.

KUP. (2024a). Työtilauksenlisäyslomake. Haettu 30.5.2024 osoitteesta https://kp.kuppi.fi

KUP. (2024b). Asiakkaanlisäyslomake. Haettu 30.5.2024 osoitteesta https://kp.kuppi.fi

KUP. (2025a). Työnäkymä. Haettu 9.1.2025 osoitteesta https://kp.kuppi.fi

KUP. (2025b). Työnäkymä. Haettu 9.1.2025 osoitteesta https://kp.kuppi.fi

Malik, U. (2019). Pros and cons of Python Programming Language. Haettu 3.6.2024 osoitteesta https://learnpython.com/blog/python-programming-advantages-disadvantages/

McKeown, R. (2019). Why is Python so popular? Haettu 3.6.2024 osoitteesta https://learnpython.com/blog/why-is-python-so-popular/

Microsoft. (2024a). Azure Active Directory is now Microsoft Entra ID. Haettu 31.5.2024 osoitteesta https://www.microsoft.com/en-us/security/business/identity-access/microsoft-entra-id

Microsoft. (2024b). What is SharePoint? Haettu 10.7.2024 osoitteesta https://support.microsoft.com/en-us/office/what-is-sharepoint-97b915e6-651b-43b2-827d-fb25777f446f

Microsoft. (2024c). Sites in SharePoint. Haettu 23.7.2024 osoitteesta https://support.microsoft.com/en-us/office/Sites-in-SharePoint-545b9394-5641-4576-a10b-e25e5b6eb837

Microsoft. (2024d). Plan your SharePoint communication site. Haettu 23.7.2024 osoitteesta https://support.microsoft.com/en-us/office/plan-your-sharepoint-communication-site-35d9adfe-d5cc-462f-a63a-bae7f2529182

Microsoft. (2024e). Pages in SharePoint. Haettu 23.7.2024 osoitteesta https://support.microsoft.com/en-us/office/Pages-in-SharePoint-3833b917-a39d-43f9-85ff-f17949fc1034

Microsoft. (2024f). Introduction to lists. Haettu 23.7.2024 osoitteesta https://support.microsoft.com/en-us/office/introduction-to-lists-0a1c3ace-def0-44af-b225-cfa8d92c52d7

Microsoft. (2024g). What is a document library? Haettu 23.7.2024 osoitteesta https://support.microsoft.com/en-us/office/what-is-a-document-library-3b5976dd-65cf-4c9e-bf5a-713c10ca2872

Microsoft. (2024h). Introduction to Document Sets. Haettu 24.7.2024 osoitteesta https://support.microsoft.com/en-us/office/introduction-to-document-sets-3dbcd93e-0bed-46b7-b1ba-b31de2bcd234

Microsoft. (2024i). Create a flow for a list or library. Haettu 23.7.2024 osoitteesta https://support.microsoft.com/en-us/office/create-a-flow-for-a-list-or-library-a9c3e03b-0654-46af-a254-20252e580d01

Microsoft. (n.d.). Introducing Power Automate. Haettu 23.7.2024 osoitteesta https://learn.microsoft.com/en-us/training/modules/get-started-flows/1-introduction

Oma SharePoint-sivusto. (2024). Pääsy rajattu vain tietyille käyttäjätunnuksille.

OpenSSL. (2024). OpenSSL Documentation. Haettu 21.11.2024 osoitteesta https://docs.openssl.org/master/man1/openssl-req/

Overview of Microsoft Graph. (2024). Haettu 30.5.2024 osoitteesta https://learn.microsoft.com/en-us/graph/overview

Ping Identity. (n.d.). Certificate-based authentication. Haettu 15.7.2024 osoitteesta https://www.pingidentity.com/en/resources/identity-fundamentals/authentication/certificate-authentication.html

RedSwitches. (2024). Python Unveiled: Advantages and Disadvantages of Python Programming Language. Haettu 5.6.2024 osoitteesta https://www.redswitches.com/blog/advantages-and-disadvantages-of-python/

Roine, J., & Anttila, J. (2015). SharePoint ja Office 365: Hyvät, pahat ja rumat (Päivitetty painos 11/2015). SharePoint Hyvät, Pahat ja Rumat.

Sahu, J. (2013). What is OpenSSL. Why use OpenSSL? Haettu 17.7.2024 osoitteesta https://medium.com/@jeevansahu72290/what-is-openssl-why-use-openssl-4c65cab16530

Tools for interacting with Microsoft Graph. (2023). Haettu 4.7.2024 osoitteesta https://learn.microsoft.com/en-us/graph/use-the-api#tools-for-interacting-with-microsoft-graph

Use the Microsoft Graph API. (2023). Haettu 4.7.2024 osoitteesta https://learn.microsoft.com/en-us/graph/use-the-api

Vasudevan, K. (2017). Insights into the Success of the Microsoft Graph API with Principal API Architect, Gareth Jones. Haettu 11.7.2024 osoitteesta https://medium.com/product-dev-stories/insights-into-the-success-of-the-microsoft-graph-api-with-principal-api-architect-gareth-jones-29d05bc6a9bc

Venafi. (2024). Self-Signed Certificates: Benefits, Risks, and Options. Haettu 21.11.2024 osoitteesta https://venafi.com/machine-identity-basics/self-signed-certificates-cyber-criminals-can-quickly-turn-strength-vulnerability/

Vuorinen, A. (6.8.2024). Henkilökohtainen keskustelu KP-ServicePartner Oy:n kehityspäällikkö, Aki Vuorisen, kanssa.

What is identity and access management (IAM)? (2024). Haettu 30.5.2024 osoitteesta https://learn.microsoft.com/fi-fi/entra/fundamentals/introduction-identity-access-management

What is Microsoft Entra ID? (2024). Haettu 29.5.2024 osoitteesta https://learn.microsoft.com/fi-fi/entra/fundamentals/whatis