

Janne Vierimaa

MERIKONTTITIE TOJEN RAJAPINTAHAKU TIMBER BY PINJA -TOIMINNAN- OHJAUSJÄRJESTELMÄÄN

MERIKONTTITIETOJEN RAJAPINTAHAKU TIMBER BY PINJA -TOIMINNAN- OHJAUSJÄRJESTELMÄÄN

Janne Vierimaa
Opinnäytetyö
Kevät 2023
Tietotekniikan tutkinto-ohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu

Tietotekniikan tutkinto-ohjelma, ohjelmistokehityksen suuntautumisvaihtoehto

Tekijä: Janne Vierimaa

Opinnäytetyön nimi: Merikonttitietojen rajapintahaku Timber by Pinja -toiminnanohjausjärjestelmään

Työn ohjaajat: Pekka Alaluukas & Elvis Punnek

Työn valmistumislukukausi ja -vuosi: Kevät 2023

Sivumäärä: 31 + 1 liite

Opinnäytetyö tehtiin Pinja Group Oy:lle, joka kehittää sahoille ja puunjalostuslaitoksille räätälöityä Timber by Pinja -toiminnanohjausjärjestelmää.

Työn aiheena oli luoda toiminnallisuus, jolla satamaoperaattorilta saadaan haettua konttierittelytiedot heille lähetetyistä puutavarakuormista. Toiminnallisuuden avulla Timber by Pinja -järjestelmän asiakkaat saavat satamaan kuormissa lähettämistään puutavarayksiköistä automaattisesti konttierittelytiedot satamassa toteutuneen yksiköitten kontituksen mukaan. Toteutunutta konttierittelyä verrataan asiakkaan alkuperäiseen konttisuunnitelmaan ja päivitetään konttien ja yksiköitten tiedot Timber by Pinja -järjestelmän asiakastietokantaan.

Uusi toiminnallisuus käyttää SFTP-tiedonsiirtoprotokollaa konttitietojen hakuun satamaoperaattorin palvelimelta, josta data saadaan Pinjan palveluväylän ja .NET-sovelluskehityksen tarjoaman Windows Communication Foundation -palvelun välityksellä käsiteltäväksi ja saatetaan vaadittuun formaattiin Timber by Pinjan tietokantatallennusta varten. Koko projekti toteutettiin .NET-sovelluskehityksen puitteissa ja koodattiin pääosin C# -ohjelmointikielellä.

Projekti saatiin viimeisteltyä ja toimitettua asiakkaalle joulukuussa 2022. Kokonaisuutena projekti onnistui hyvin, vaikka aikataulu hieman venyikin alkuperäisestä suunnitelmasta. Projekti auttoi ymmärtämään Timber by Pinja -toiminnanohjausjärjestelmän toimintaa entistä paremmin ja myös syvensi ymmärrystä järjestelmän asiakkaitten liiketoiminnoista ja prosesseista. Opinnäytetyöprojektin pohjalta pystytään konttitietojen hakua laajentamaan Timber by Pinjassa koskemaan myös muita satamaoperaattoreita.

Asiasanat: toiminnanohjausjärjestelmät, järjestelmäintegraatio, ohjelmistointegraatio, C#, .NET-sovelluskehitys, SFTP, palveluväylä, ESB

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Information Technology, Option of Software Development

Author: Janne Vierimaa

Title of thesis: Fetching shipping container data via interface to Timber by Pinja ERP

Supervisors: Pekka Alaluukas & Elvis Punnek

Term and year when the thesis was submitted: Spring 2023

Number of pages: 31 + 1 appendix

The thesis was done for Pinja Group Oy, which develops Timber by Pinja, a customized ERP system for sawmills and wood processing facilities.

The topic of the project was to create an interface and functionality that fetches shipping container data from port operator to Timber by Pinja. With the help of this new functionality, Timber by Pinja customers automatically get the container data from the sent processed timber units to the system after the port operator has done the containerization of the units.

The container data is fetched via SFTP-protocol from the port operator's server and with Pinja's enterprise service bus it is being further directed to Windows Communication Foundation -service provided by .NET framework. Data is then processed to format that is used in Timber by Pinja and saved to customer Database.

Project was completed and delivered to customer in December 2022. Overall, the project was successful, even though the schedule was slightly extended from the original plan. Based on the thesis project, it will be possible to expand the container data fetching to include other port operators.

Keywords: enterprise resource planning software, software integration, C#, .NET framework, SFTP, enterprise service bus, ESB

SISÄLLYS

1	JOHDANTO	7
2	JÄRJESTELMÄINTEGRAATIO JA SEN RATKAISUMALLIT	8
2.1	Toteutustavat.....	8
2.1.1	Point-to-point-integraatio.....	8
2.1.2	Hub-and-spoke-integraatio.....	9
2.1.3	Palveluväylä.....	10
2.2	Projektissa käytetty integraatiotapa	11
3	PILVIPALVELUT.....	12
3.1	Windows Communication Foundation	13
3.2	Internet Information Services.....	13
3.3	Käyttö projektissa	13
4	SSH-PROTOKOLLA JA TIEDONSIIRTO	14
4.1	SFTP	14
4.2	Todentaminen	14
4.3	Käyttö projektissa	14
5	.NET-SOVELLUSKEHYS	15
5.1	Common Language Runtime.....	15
5.2	Framework Class Library.....	15
5.3	Kehitysympäristö	16
5.4	Käyttö projektissa	16
6	C#-OLIO-OHJELMOINTI	17
6.1	Tyypitys	17
6.2	Nimiavaruudet	17
6.3	Periytyminen.....	18
6.4	LINQ.....	19
6.5	Käyttö projektissa	20
7	TIETOKANTA	22
7.1	Relaatiomalli	22
7.2	SQL	22
7.3	Tietokannat projektissa.....	23
8	TOTEUTUS	24

8.1	Määrittely.....	24
8.2	Tekninen malli	24
8.2.1	Konttitietojen haku	25
8.2.2	Oliomuotoisen datan lähetys.....	25
8.2.3	Tietokantaan tallennus	26
8.3	Toteutus ja testaaminen	26
8.4	Toiminta kokonaisuutena.....	26
9	TULOKSET JA POHDINTA	28
	LÄHTEET.....	30
	LIITTEET	32

1 JOHDANTO

Opinnäytetyöprojekti toteutettiin Pinja Group Oy:lle loppuvuoden 2022 aikana ja dokumentoitiin opinnäytetyöraporttiin alkuvuonna 2023. Olen työskennellyt Pinjan palveluksessa täysipäiväisesti opiskelun ohella hieman alle kaksi vuotta. Tähän ajanjaksoon mahtuvat mukaan opintoihin kuuluvat työharjoittelut ja yrityslähtöiset projektit.

Pinjan Forest-liiketoimintayksikkö kehittää metsäteollisuuden ohjelmistoja. Itse toimin osana sahoille ja puunjalostuslaitoksille räätälöityä Timber by Pinja -toiminnanohjausjärjestelmän kehitystiimiä. Omat vastualueeni pitävät sisällään pääasiallisesti Timber by Pinjan ohjelmistorajapintojen kehitys- ja ylläpitotehtäviä.

Opinnäytetyöprojektiin aiheena oli toteuttaa Timber by Pinja -asiakkaalle rajapinta ja toiminnallisuus, jolla eräältä satamaoperaattorilta saadaan haettua automaattisesti konttieroittelytiedot heille lähetystä puutavarakuormista. Haetut tiedot sisältävät puutavarayksikköjen kontituksessa käytettyjen merikonttien numerot, sinetinumerot sekä tiedot konttien sisältämisestä puutavarayksiköistä. Toiminnallisuus voidaan tarvittaessa ottaa käyttöön myös muille Timber by Pinja -asiakkaille. Automaattinen konttitietojen haku säästää aikaa ja vaivaa, ja lisäksi tietoa käsin syötettäessä tapahtuvien virheiden mahdollisuus poistuu. Tarve toiminnallisuudelle syntyi keskusteluissa asiakkaan edustajan kanssa.

XML-muodossa olevien konttitietojen hakuun satamaoperaattorin palvelimelta käytetään salattua SFTP-tiedonsiirtoprotokollaa, jonka avulla data saadaan Pinjan palveluväylän ja .NET-sovelluskehityksen tarjoaman Windows Communication Foundation -palvelun välityksellä käsiteltäväksi vääditteeseen formaattiin Timber by Pinjan tietokantatallennusta varten. Tietokantaan tallennetaan konttien tiedot sekä tieto niiden sisältämisestä puutavarayksiköistä. Koko opinnäytetyöprojekti toteutettiin .NET-sovelluskehityksen puitteissa ja koodattiin pääosin C# -ohjelmointikielellä. Opinnäytetyössä kuvataan käytettyä integraatiotapaa, sen käytännön toteutusta sekä toteutukseen käytettyjä teknologioita.

2 JÄRJESTELMÄINTEGRAATIO JA SEN RATKAISUMALLIT

Järjestelmäintegraatio tarkoittaa toimintaa, jolla organisaation tai organisaation ulkopuolisiin eri järjestelmiin hajautunut tieto saadaan yhdistettyä ja muunnettua sopivaksi sitä tarvitsevalle tietojärjestelmälle. Integraatiolla siis mahdollistetaan erilaisten tietojärjestelmien keskustelu keskenään. Tarve integraatiolle syntyy, kun olemassa oleva, mutta eri järjestelmiin varastoitu tieto tarvitaan yksittäiseen järjestelmään tai prosessiin. Sen sijaan että tieto siirrettäisiin järjestelmästä toiseen käsin, ohjelmistojen välisellä integraatiolla voidaan ratkaista tiedonsiirron ongelmia ja poistaa järjestelmien välisiä informaatioiloja. Onnistuneen integraation avulla manuaalisen työn määrä vähenee ja prosessit sekä mahdollisuus päätöksentekoon nopeutuvat. (Itewiki 2022; Wikipedia 2021a.)

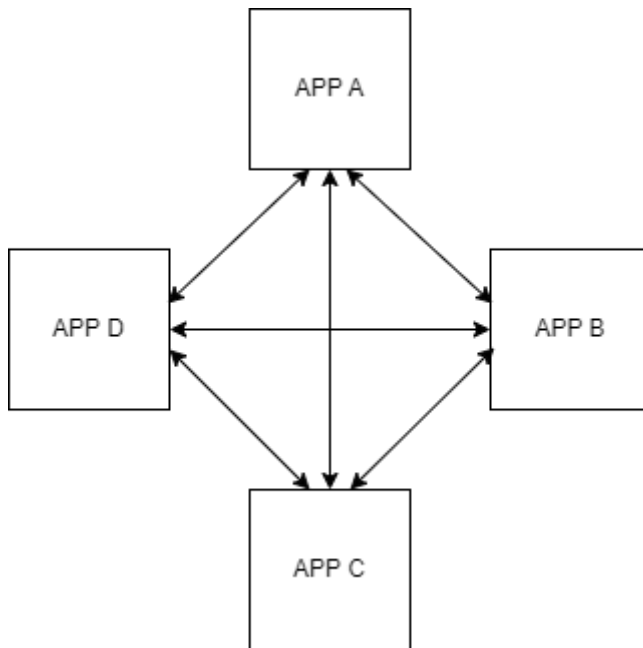
2.1 Toteutustavat

Integraatiot ja tiedonsiirto toteutetaan ohjelmointirajapintojen avulla ja ne toimivat ikään kuin ovina tietojärjestelmiin. Arkkitehtuurisesti ja teknologisesti integraation toteutukseen voidaan käyttää useita eri toteutusmalleja. Mallien soveltuvuus käyttöön riippuu kyseisestä integraatiotarpeesta ja aikataulusta, organisaation omista integraatiotyökaluista, prosesseista sekä halusta kehittää niitä. Toteutustavan valinnan tulisi kuitenkin perustua ensisijaisesti nimenomaan organisaation liiketoiminnan tarpeisiin; ratkaisu, joka tukee liiketoiminnan tavoitteita, on paras valinta. (Itewiki 2022; Wikipedia 2021.)

2.1.1 Point-to-point-integraatio

Point-to-point-integraatio on kahden järjestelmän välinen suora integraatio, jossa tieto viedään suoraan järjestelmästä toiseen. Jokaisen integroitavan ohjelmisto- tai sovellusparin väliin tehdään erillinen liitäntä. Tämä liitäntä sisältää kaikki kahden järjestelmän väliseen tiedon kääntämiseen, integraatioon ja viestintään liittyvät säännöt. (Itewiki 2022.)

Kuva 1 havainnollistaa Point-to-point-integraatiota usean eri sovelluksen välillä.

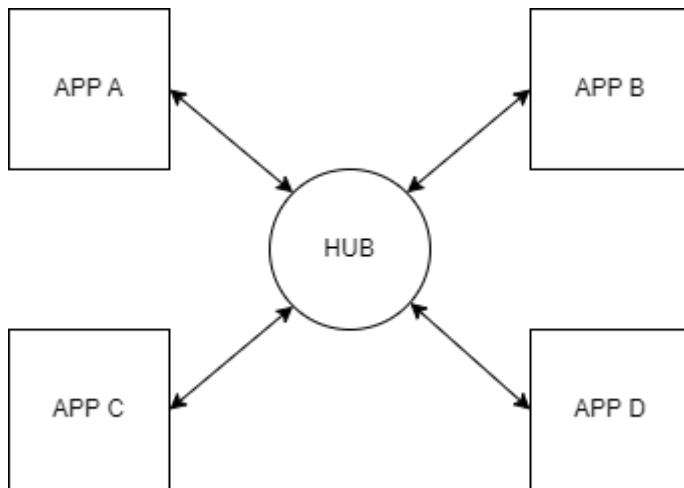


KUVA 1. Point-to-point-integraatio usean eri sovelluksen välillä

Point-to-point-integraation etuna voidaan pitää kustannustehokkuutta sekä helppoa toteutettavuutta lyhyellä aikavälillä. Hyvin usein kuitenkin ajan myötä tällaisten integraatioiden ylläpitäminen ja hallinta muodostuu erittäin vaikeaksi ja kalliiksi, koska eri integraatiot muodostavat ”integraatiospagettiverkoston”. Liiketoimintatarpeiden vaatimat muutokset tähän ”spagettiin” ovat työläitä ja virheherkkiä. Lisäksi kun organisaatiossa otetaan käyttöön uusia järjestelmiä, integraatioiden määrä kasvaa eksponentiaalisesti. Integraatiomallia pidetäänkin riskialttiina ja riskien mahdollisuus on suuri, koska integraatiot on toteutettu toisistaan irrallisina ja usein myös eri teknologioilla. (Itewiki 2022; Solita 2018.)

2.1.2 Hub-and-spoke-integraatio

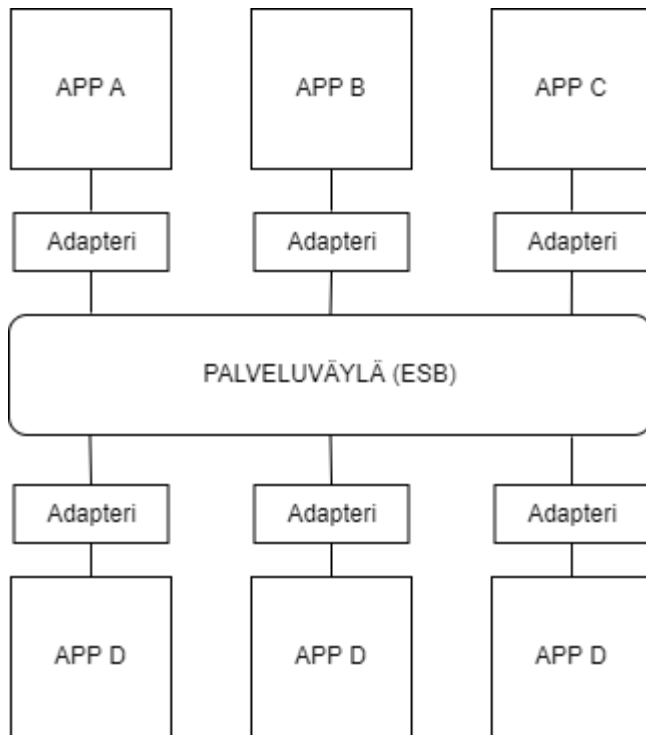
Hub-and-spoke-mallissa (kuva 2) integroitavien järjestelmien välissä toimii keskitettynä integraatioalustana viestinvälittäjä eli hubi, joka huolehtii viestien vastaanottamisen, muuttamisen ja lähettämisen kohdennettuna oikeaan järjestelmään. Näin mahdollistetaan useiden tietovirtojen hallinta ja valvonta keskitetysti. Keskitetyn integraatioalustan avulla on usein mahdollisuus ajastaa tiedonsiirtoa. Se voidaan myös saada heräämään tiedoston saapumiseen tai tietokantamuutokseen tai käynnistää esimerkiksi suoralla REST/JSON-kutsulla. Keskitetty alusta tarjoaa myös yhdenmukaisen toteutustavan integraatioille ja uusien järjestelmien lisääminen ja integroiminen on helpompaa hubin kautta. (Itewiki 2022; Solita 2018; Wikipedia 2021.)



KUVA 2. Hub-and-spoke-integraatio.

2.1.3 Palveluväylä

Palveluväylä (kuva 3) eli ESB (Enterprise Service Bus) on kehitetty poistamaan hub and spoke -integraation heikkouksia, jossa integraatio toimii yhden hubin varassa ja hubista voi muodostua tiedonvälityksen pullonkaula. Tiedonsiirto palveluväylällä tapahtuu keskitetyn väylän (Bus) kautta, mutta yhden välittäjän taakka jaettu arkkitehtuuriverkoston erillisiksi toiminnoiksi. Palveluväylä pystyy ottamaan vastaan eri teknologioilla lähetettyä tietoa, kääntämään ja välittämään sitä eteenpäin. Väylä sisältää yleensä myös niin sanotun jonojärjestelmän, johon sanomat voidaan tallentaa. Tällä varmistetaan se, että vaikka sanoma ei ole vielä saavuttanut sitä järjestelmää, johon se lopulta on menossa, voidaan kuitenkin luottaa siihen, että se on tallessa. Lähetystä voidaan tarvittaessa yrittää uudelleen palveluväylän sisäisesti. Jonotuksen lisäksi palveluväylässä voi olla myös tuki publish-subscribe-tyyppiseen viestinvälitykseen. Siinä tiettyä dataa ja sanomia itselleen haluavat järjestelmät asetetaan kuuntelijoiksi eri aiheiden viesteille tai sisällöille, jotka palveluväylä osaa reitittää vaaditulla tavalla. (HiQ Finland Oy 2022; IteWiki 2022; Solita 2018)



Kuva 3. Palveluväylä.

2.2 Projektissa käytetty integraatiotapa

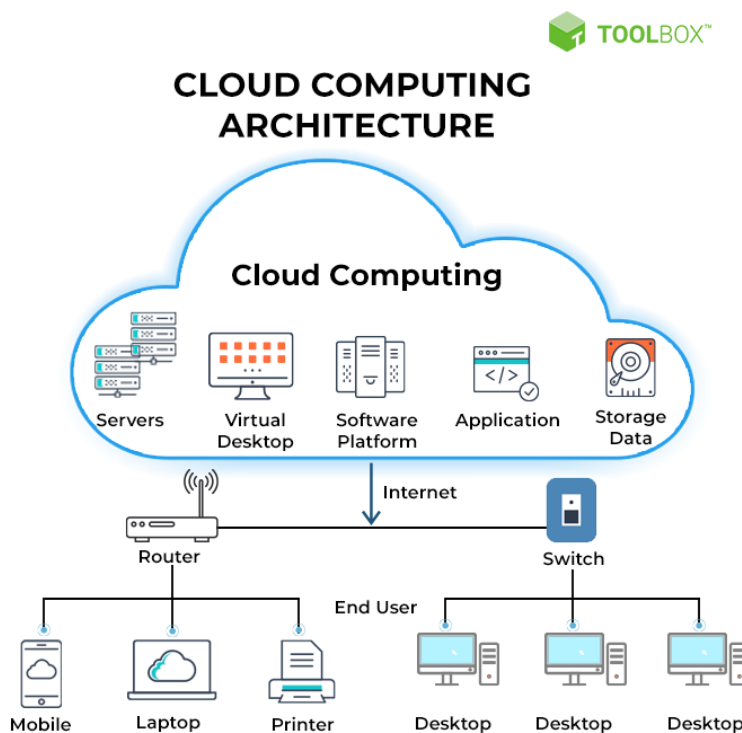
Projektiin valikoitui käyttöön olemassa oleva Pinjan palveluväylä. Tähän vaikuttaneita syitä oli useita. Projektia tullaan jatkossa laajentamaan koskemaan konttisanomien hakua useilta eri sata-maoperaattoreilta. Paluusanomina haettavien konttierittelytietojen lisäksi myös puutavarakuormien satamaan lähetykseen liittyvät sanomat ohjataan tulevaisuudessa kulkemaan tämän palveluväylän kautta. Integraation hallinta ja ylläpito keskitetysti palveluväylän välityksellä on huomattavasti hel-pompaa edellä kuvattujen mallien mukaisesti. Integraation varsinaista käytännön toteutusta on ku-vattu tarkemmin seuraavissa luvuissa ja liitteessä 1.

3 PILVIPALVELUT

Pilvipalvelu tarkoittaa tietoteknisten palvelujen, kuten sovellusten, tallennuskapasiteetin tai laskentatehon, toimittamista tyypillisesti internetin välityksellä. Pilvipalvelut voidaan jakaa pääluokkiin, jotka ovat

- SaaS (Software as a Service) eli ohjelmiston jakelumalli, jossa palvelun tarjoaja ylläpitää sovellusohjelmistoa palvelimillaan ja tarjoaa palvelua internetin välityksellä
- IaaS (Infrastructure as a Service) eli palvelimien ja palvelinsalien ulkoistaminen
- PaaS (Platform as a Service) eli palvelualustan ulkoistaminen, jossa palveluntuottaja tarjoaa sovellusaloja palveluna. (Wikipedia 2022f.)

Kuvassa 4 on havainnollistettu pilvipalvelujen tyypillistä arkkitehtuurimallia ja palvelujen toimittamista internetin välityksellä loppukäyttäjille.



KUVA 4. Pilvipalvelujen arkkitehtuurimalli (Spiceworks 2022.)

Pilvipalvelut tuovat mukanaan kustannustehokkuuden, joustavuuden ja nopeuden palveluiden toteutuksessa. Lisäksi etuna on, että pilvessä sijaitsevaan palveluun ja siihen liittyvään dataan pääsee käsiksi millä tahansa laitteella, jossa on internet-yhteys. (Wikipedia 2022f.)

3.1 Windows Communication Foundation

Windows Communication Foundation eli WCF on osa Microsoftin .NET-sovelluskehystä ja sitä käytetään pääasiassa palvelukeskeistä arkkitehtuuria hyödyntävien, toisiinsa yhdistettävien sovellusten rakentamiseen. WCF mahdollistaa datan lähettämisen asynkronisesti eri päätepisteiden välillä. Palveluja voidaan julkaista Windows-sovelluksina tai -palveluina ja myös Internet Information Servicesin välityksellä. WCF on suunniteltu tarjoamaan helposti hallittavissa oleva lähestymistapa verkkopalvelujen luomiseen ja ylläpitoon. (Microsoft 2021; Wikipedia 2022g.)

3.2 Internet Information Services

Internet Information Services eli IIS on Microsoftin kehittämä palvelinohjelmistokokonaisuus, jota käytetään Windows-palvelimissa. Yleisimmin IIS:iä käytetään ASP.NET-verkkosovellusten, staattisten verkkosivujen tai WCF-palvelujen isännöintiin. Sitä voidaan käyttää myös FTP-palvelimena tai laajentaa isännöimään muilla alustoilla rakennettuja verkkosovelluksia. IIS on saatavilla eri Windows-versioille ja se on laajasti mukautettavissa erilaisia käyttötarkoituksia varten. (Stackify 2022; Wikipedia 2022h.)

3.3 Käyttö projektissa

Projektissa käytettävä palveluväylä sijaitsee Pinjan pilvipalvelussa ja toistaiseksi se on toteutettu useana erillisenä WCF-palveluna, jotka toimivat IIS:n päällä. Palveluväylän toteutus mahdollistaa myös muiden teknologioiden käytön palvelujen toteuttamiseen. Projektissa kontitustietojen hakeamiseen käytettävä rajapinta toteutettiin näitä olemassa olevia WCF-palveluja hyödyntämällä. Palveluväylässä sijaitseva palvelun kutsuja kutsuu ajastetusti Pinjan palvelua, joka hakee konttiterittelytiedot ulkoiselta palvelimelta ja lähettää ne eteenpäin asiakkaan palvelimelle asennetun palvelun päätepisteeseen. Asiakkaan palvelu huolehtii datan tallennuksen asiakastietokantaan. Toteutusta on kuvattu tarkemmin luvussa 8.

4 SSH-PROTOKOLLA JA TIEDONSIIRTO

Secure Shell eli SSH on salattuun tietoliikenteeseen tarkoitettu verkkoprotokolla, joka mahdollistaa verkkopalvelujen käytön turvallisesti salaamattoman verkon yli. SSH-sovellukset perustuvat niin sanottuun asiakas-palvelinarkkitehtuuriin ja SSH:n yleisin käyttötapa onkin SSH-asiakasohjelman avulla muodostettu etäyhteys SSH-palvelimeen. SSH-etäyhteyden avulla voidaan toista tietokonetta käyttää merkkipohjaisen konsolin välityksellä. SSH:n avulla on myös mahdollista suojata FTP-, HTTP- tai muuta liikennettä. (Wikipedia 2022a; Wikipedia 2022b.)

4.1 SFTP

SSH File Transfer Protocol (tai Secure File Transfer Protocol) eli SFTP on SSH:n mukana oleva erillinen tiedonsiirtoon tarkoitettu protokolla. Se mahdollistaa erilaiset tiedosto-operaatiot salatun SSH-yhteyden yli. SFTP:ssä siirrettävä data suojataan salauksella, eikä esimerkiksi tekstitiedostoja siirretä selväkielisinä. SFTP ei kuitenkaan ole SSH:n yli käytetty FTP, vaan erikseen suunniteltu protokolla. (Linux.fi 2016; Wikipedia 2022c.)

4.2 Todentaminen

SSH-protokolla käyttää autentikointiin eli todentamiseen joko käyttäjätunnusta ja salasanaa tai niin sanottua avainparia. SSH-avain vaatii kaksi osaa: käyttäjän yksityisavaimen ja julkisen avaimen, joka on tallennettu palvelimelle. Kirjautumisyrityksen tapahtuessa palvelin tarkistaa, että yksityinen ja julkinen avain vastaavat toisiaan, ja sallii kirjautumisen vain siinä tapauksessa. (Tavu.io 2022.)

4.3 Käyttö projektissa

Projektissa rakennetussa toiminnallisuudessa SFTP-protokollaa käytetään konttierittelysanomien hakuun ja tiedonsiirtoon satamaoperaattorin palvelimelta Pinjan palveluväylään. Autentikointiin käytetään käyttäjätunnusta ja salasanaa. SFTP-protokollan käyttö tapahtuu Pinjan palveluväylässä toimivan palvelun käyttämän avoimen lähdekoodin Renci SSH.NET -kirjaston avulla, jota kuvataan tarkemmin luvussa 6.

5 .NET-SOVELLUSKEHYS

.NET (dot NET) on Microsoftin kehittämä sovelluskehys, joka tarjoaa ympäristön monenlaisten sovellusten kehittämiseen ja tukee useita eri ohjelmointikieliä. .NET-sovelluskehysten puitteissa voidaan rakentaa Windows-työpöytäsovelluksia, web-sovelluksia ja mobiilisovelluksia ja se soveltuu myös pelikehityskäyttöön. Se koostuu kahdesta pääosasta: Common Language Runtime ja .NET Framework Class Library. (Wikipedia 2022i.)

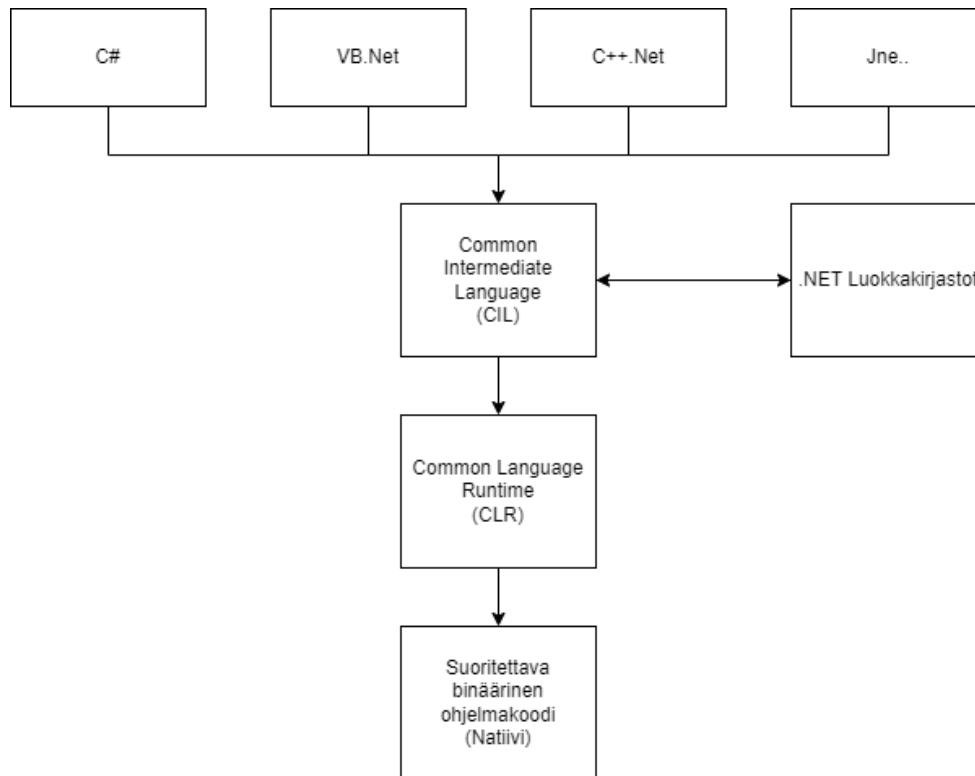
5.1 Common Language Runtime

Common Language Runtime (CLR) on ajonaikainen ympäristö sovellusten suorittamiseen. Se vastaa sovelluksen suorituksen hallinnasta, muistin hallinnasta, poikkeuksien käsittelystä ja muista tehtävistä, jotka liittyvät sovelluksen suorittamiseen. CLR tukee useita eri ohjelmointikieliä, kuten C#, VB.NET, F# ja C++/CLI. Ohjelmointikielten lähdekoodi käännetään ensin tavukoodiksi, jota Microsoft kutsuu välikieleksi (Common Intermediate Language, CIL.) Ajonaikainen ympäristö kääntää välikielisen koodin ohjelman suorituksen aikana kohdeympäristön konekielelle. (Microsoft 2022a; Wikipedia 2022i.)

5.2 Framework Class Library

Framework Class Library (FCL) on laaja kokoelma luokkakirjastoja, jossa on valmiita toimintoja moniin eri tarpeisiin. Se sisältää luokkia esimerkiksi tiedostojen käsittelyyn, verkon ja tietokantojen hallintaan, sekä grafiikan käsittelyyn. Sisäisesti FCL:n luokat on jaettu useihin eri "paketteihin", jotka on ryhmitelty eri tehtävien mukaan. (Techopedia 2022; Wikipedia 2022i.)

Kuva 5 kuvaa .NET-sovelluskehysten toimintaperiaatetta kaavion muodossa.



Kuva 5. .NET-sovelluskehityksen toimintaperiaate (Wikipedia 2022i.)

5.3 Kehitysympäristö

Visual Studio on Microsoftin kehittämä integroitu kehitysympäristö (IDE), jota tyypillisesti käytetään .NET-sovelluksien kehitykseen. Visual Studio tarjoaa kehittäjille monipuoliset työkalut ja ominaisuudet, kuten koodieditorin, debuggausominaisuudet ja versionhallinnan. Visual Studio sisältää myös useita laajennuksia ja lisäosia, kuten erilaisia malleja projektien aloittamiseen, komponentteja ja kirjastoja, jotka helpottavat sovellusten kehittämistä ja käyttöönottoa. Vaikka .NET onkin alun perin suunniteltu Windows-alustalle, myöhemmin kehitetty avoimen lähdekoodin .NET Core mahdollistaa sovellusten kehittämisen Visual Studiolla Windowsin lisäksi myös Linux- ja macOS-alustoille. (Microsoft 2022b; Wikipedia 2022j.)

5.4 Käyttö projektissa

Projektin varsinainen ohjelmointiosuus toteutettiin kokonaisuudessaan .NET-sovelluskehityksen puitteissa. Kehitysympäristönä toimi Microsoftin Visual Studio 2019. Muutamia pienempiä VB.Net-kielillä toteutettuja osuuksia lukuun ottamatta pääasiallisena ohjelmointikielenä projektissa käytettiin C#-ohjelmointikieltä, josta on kerrottu tarkemmin seuraavassa luvussa.

6 C#-OLIO-OHJELMOINTI

C# (C sharp) on Microsoftin .NET-alustalle kehitetty moderni korkean tason olio-ohjelmointikieli. C# on suosittu, tehokas ja monikäyttöinen ohjelmointikieli Windows-työpöytäsovellusten, Web-sovellusten, mobiilisovellusten, pilvipalveluiden ja pelien kehitykseen. C#:n perusrakenteet ovat samankaltaiset kuin C- ja C++-kielissä, ja se onkin C-kielen jälkeläinen. Kieli sisältää monia moderneja konsepteja, kuten nimiavaruudet, pakettien hallinta, poikkeukset ja generoitavat tietotyypit. Se tarjoaa myös erittäin kattavan tukikirjaston, joka tarjoaa paljon valmiita toimintoja moniin eri tarpeisiin, kuten tiedostojen käsittely, verkon hallinta ja grafiikan käsittely. (Wikipedia 2022k.)

6.1 Tyypitys

C# on vahvasti ja staattisesti tyypitetty ohjelmointikieli, jossa kaikilla muuttujilla ja vakioilla on tietotyyppi ja se täytyy määritellä ennen käyttöä. Tämä takaa sen, että data on aina oikeassa muodossa, ja estää virheitä, jos datan tyyppi ei vastaa odotettua. C# sallii kuitenkin myös yleiskäyttöisen ”var”-tietotyypin käytön. Tällöin tietotyyppi täytyy kuitenkin pystyä yksiselitteisesti päättämään, kun muuttujaan asetetaan arvo. Todellisuudessa siis myös tämä tietotyyppi on staattinen. (Wikipedia 2022k.)

6.2 Nimiavaruudet

Nimiavaruuksia käytetään tyypillisesti ohjelmointikielissä jäsentämään ohjelmiston sisällä käytettäviä tunnisteita. Nimiavaruudet mahdollistavat saman nimen käytön eri osissa ohjelmaa ilman, että nimet sekoittuvat toisiinsa. Kuvan 6 esimerkissä periaatetta on havainnollistettu luomalla kaksi samannimistä luokkaa, jotka sijaitsevat eri nimiavaruuksissa.

C#-koodi muodostuu nimiavaruuksista, niissä olevista tyypeistä (esim. luokat) ja tyypeissä olevista jäsenistä. Tyyppien avulla voidaan luoda muuttujia, jotka sisältävät viittauksen yhteen tai useampaan olioon. (Microsoft 2023; Wikipedia 2022k.)

```

1 namespace MyCompany.DataAccess
2 {
3     class Employee
4     {
5         public string Save()
6         {
7             // Koodi, joka tallentaa tiedot tietokantaan
8             return "Data saved successfully";
9         }
10    }
11 }
12
13 namespace MyCompany.BusinessLogic
14 {
15     class Employee
16     {
17         public string CalculateBonus()
18         {
19             // Liiketoimintalogiikka, joka laskee palkkion
20             return "Bonus calculated successfully";
21         }
22     }
23 }
24
25 class Program
26 {
27     static void Main(string[] args)
28     {
29         // Käytetään tietokantatietoja käsittelevää luokkaa
30         MyCompany.DataAccess.Employee dataEmployee = new MyCompany.DataAccess.Employee();
31         Console.WriteLine(dataEmployee.Save());
32
33         // Käytetään liiketoimintalogiikkaa käsittelevää luokkaa
34         MyCompany.BusinessLogic.Employee logicEmployee = new MyCompany.BusinessLogic.Employee();
35         Console.WriteLine(logicEmployee.CalculateBonus());
36     }
37 }

```

KUVA 6. Esimerkki nimiavaruuksista ja kahdesta samannimisestä luokasta

6.3 Periytyminen

Periytyminen tarkoittaa C#-kielessä sitä, että luokka voi periä ominaisuuksia ja metodeja toisesta luokasta. Luokka- ja rajapintatyyppejä voidaan C#-kielessä organisoida perimähierarkioiksi niin, että kaikkien luokkien käyttämät jäsenet löytyvät hierarkian abstrakteimmasta luokasta ja uudet luokat lisäävät tarkentavia jäseniä tarvittaessa. Kaikkia perittävän luokan metodeja ei ole pakollista kuitenkaan periä sellaisenaan, vaan perivä luokka voi määritellä ne uudelleen "override"-avainsanan avulla. (Kuva 7.) Perivä luokka voi myös piilottaa perittävän jäsenen tai keskeyttää perittävän luokan jäsenen periytymisen. (Microsoft 2023; Wikipedia 2022k.)

```

1  abstract class Animal
2  {
3      public string Name { get; set; }
4      public abstract string MakeNoise();
5  }
6
7  class Dog : Animal
8  {
9      public override string MakeNoise()
10     {
11         return "Woof!";
12     }
13 }
14
15 class Cat : Animal
16 {
17     public override string MakeNoise()
18     {
19         return "Meow!";
20     }
21 }
22
23 class Program
24 {
25     static void Main(string[] args)
26     {
27         Dog dog = new Dog();
28         dog.Name = "Fido";
29         Console.WriteLine(dog.MakeNoise()); // Woof!
30
31         Cat cat = new Cat();
32         cat.Name = "Whiskers";
33         Console.WriteLine(cat.MakeNoise()); // Meow!
34     }
35 }

```

KUVA 7. Esimerkki periytymisestä

6.4 LINQ

LINQ (Language Integrated Query) on Microsoftin kehittämä teknologia, joka mahdollistaa SQL-tyyppisten kyselyjen kirjoittamisen C#- ja VB.NET -ohjelmointikielissä. Se on integroitu osaksi .NET-sovelluskehystä ja sisältää useita erilaisia tietolähteitä, kuten oliot, taulukot, kokoelmat, XML-dokumentit ja SQL-tietokannat. LINQ:n avulla on mahdollista kehittää luettavampaa ja ylläpidettävää koodia, ja se tekee kyselyiden kirjoittamisesta helpompaa ja tehokkaampaa. (Microsoft 2022c.)

6.5 Käyttö projektissa

Kuten luvussa 5 todettiin, opinnäytetyöprojekti toteutettiin pääosin C#-kielellä. Ohjelmakoodiltaan Timber by Pinja on erittäin laaja kokonaisuus, joka on jaettu useisiin eri projekteihin, jotka sisältävät useita nimiavaruuksia. Lisäksi palveluväylä ja Timber by Pinjan asiakkaitten käytössä oleva WCF-palvelu ovat erikseen omina projekteinaan. Kuten aiemmin todettiin, nimiavaruuksien käytöllä mahdollistetaan useiden samannimisten luokkien käyttö. Monet projektien luokista on myös periytetty hierarkkisesti edellä kuvatun mallin mukaisesti.

LINQ-kyselykieltä käytettiin projektissa erityisesti XML-raakadatasta deserialisoidun olion saattamisessa Timber by Pinjan konttitieto-olioksi. Lisäksi LINQ:n avulla olioista ja niiden sisältämistä listoista saatiin haettua tarvittava data asiakastietokantaan tallentamista varten. Kokonaisuudessaan LINQ:n käyttö sujuvoitti ohjelmakoodia projektissa huomattavasti. (Kuva 8.)

```
using System;
using System.Linq;
using System.Collections.Generic;

class Program
{
    static void Main(string[] args)
    {
        List<Employee> employees = new List<Employee>()
        {
            new Employee { Name = "John", Age = 20, Departments = new List<string> { "HR", "IT" } },
            new Employee { Name = "Jane", Age = 30, Departments = new List<string> { "IT" } },
            new Employee { Name = "Jim", Age = 40, Departments = new List<string> { "Marketing" } },
        };

        var ITEmployees = from employee in employees
                          where employee.Departments.Contains("IT")
                          select new { employee.Name, Departments = string.Join(", ", employee.Departments) };

    }
}

class Employee
{
    public string Name { get; set; }
    public int Age { get; set; }
    public List<string> Departments { get; set; }
}
```

KUVA 8. Esimerkki LINQ:n käytöstä oliolistan suodattamisessa: Valitaan ainoastaan ne työntekijät, joiden osastona on IT.

Datan hakemiseen SFTP-palvelimelta käytettiin SSH Renci -kirjastoa. Renci SSH -kirjasto tarjoaa C#- kieleen ja .NET – sovelluskehikseen mahdollisuuden SSH-protokollan käyttöön. Se sisältää toiminnallisuuden, joka mahdollistaa salattujen yhteyksien luomisen ja hallinnoinnin erilaisten verk-

kolaitteiden ja palvelimien välillä. Projektissa kirjastoa käytettiin koodissa luotaessa yhteys sata-
maoperaattorin SFTP-palvelimeen ja tiedostojen lataamiseen SFTP-protokollan välityksellä. (Kuva
9.)

```
1  using Renci.SshNet;
2
3  namespace RenciSshNetExample
4  {
5      class Program
6      {
7          static void Main(string[] args)
8          {
9              string host = "example.com";
10             int port = 22;
11             string username = "username";
12             string password = "password";
13             string remoteFilePath = "/path/to/remote/file.txt";
14             string localFilePath = @"C:\path\to\local\file.txt";
15
16             using (var sftp = new SftpClient(host, port, username, password))
17             {
18                 sftp.Connect();
19
20                 using (var file = File.OpenWrite(localFilePath))
21                 {
22                     sftp.DownloadFile(remoteFilePath, file);
23                 }
24
25                 sftp.Disconnect();
26             }
27         }
28     }
29 }
```

KUVA 9. Yksinkertaistettu esimerkki tiedostojen lataamisesta Renci SSH -kirjaston tarjoaman SFTP-protokollan avulla

7 TIETOKANTA

Tietokanta tarkoittaa järjestettyä tietokoneen tallentamaa tietojen ja informaation koostetta, joista eri hakumenetelmiä hyödyntämällä voidaan hakea yksittäinen tieto tai eri tietoyhdistelmiä. Tietokantojen sisältämää tietoa voidaan myös korjailla ja päivittää. Tietokannassa olevan tiedon hakemiseen, tallentamiseen sekä käsittelemiseen ja hallintaan käytetään hallintajärjestelmää eli Database Management System -ohjelmistoa. (Wikipedia 2022d.)

7.1 Relaatiomalli

Relaatiotietokannat perustuvat relaatiomalliin, jonka avulla tietokannan peruskäsitteet määritellään. Relaatiomallissa tietokantaan tallennettava data on järjestetty äärellisiksi listoiksi, jotka on ryhmitelty erillisiksi taulukoiksi eli tauluiksi. Taulut sisältävät tietoa sarakkeissa ja riveissä. Taulun jokainen sarakke on taulussa samaan arvoalueen määrittelyyn liittyvien arvojen lista ja vastaa yleensä yhtä tietoluokkaa, esimerkiksi nimeä tai osastoa. Jokainen rivi eli tietue sisältää data-arvot risteäville sarakkeille. (Kuva 10.) Eri taulujen tiedot yhdistetään toisiinsa taulun avaimella, yleisimmin ID:llä. (Wikipedia 2022e.)



Kuva 10. Esimerkki relaatiotietokannan tauluista ja niiden yhteyksistä

7.2 SQL

SQL on lyhenne sanoista Structured Query Language eli strukturoitu kyselykieli, jolla relaatiotietokantojen sisältämää dataa käsitellään ja hallitaan. SQL-kielessä käytetään erilaisia komentoja, kuten CREATE, SELECT, INSERT, UPDATE, DELETE ja DROP. Komentojen avulla muodostetuilla SQL-kyselyillä relaatiotietokantaan voidaan tehdä erilaisia hakuja, muutoksia ja lisäyksiä. SQL-kieli

sisältää myös erilaisia rakenteita, kuten JOIN ja UNION, jotka mahdollistavat tietojen hakemisen useista relaatiotietokannan tauluista yhdellä lauseella. (Wikipedia 2022e.)

Kuvan 11 esimerkki esittää SQL-lauseen, jolla tietoja haetaan useasta eri tietokannan taulusta JOIN-rakenteen avulla. WHERE-ehto asettaa haulle halutut rajoitukset.

```
SELECT products.name AS 'Product Name', suppliers.company_name AS 'Supplier
Name', orders.quantity AS 'Quantity'
FROM orders
JOIN products
ON orders.product_id = products.product_id
JOIN suppliers
ON products.supplier_id = suppliers.supplier_id
WHERE orders.order_date BETWEEN '2022-01-01' AND '2022-12-31'
```

KUVA 11. Esimerkki SQL-lauseesta

7.3 Tietokannat projektissa

Projektissa tietokantoja käytetään tallentamaan SFTP-palvelimelta haetut konttiterittelytiedot ensin palveluväylän jonotusjärjestelmän tietokannan tauluihin raakadatana, sekä oliomuotoiseksi normalisoituna datana. Edelleen jatkokäsitelty data tallennetaan Timber by Pinjan asiakastietokantaan kahteen erilliseen tauluun, jotka on linkitetty ID:n perusteella toisiinsa edellä kuvatun relaatiomallin mukaisesti. Projektin toteutuksessa Timber by Pinjan asiakastietokannan konttitietoja sisältävään tauluun myös lisättiin kaksi uutta saraketta. Tietokantojen hallintaohjelmistona projektissa käytettiin Microsoft SQL Serveria.

8 TOTEUTUS

Tarve merikonttitietojen automaattiselle haulle syntyi keskusteluissa erään Timber by Pinja -järjestelmän asiakkaan kanssa. Lähtötilanne ennen konttitietojen automaattista hakua oli, että puutavarayksiköistä muodostetut puutavarakuormat kuljetettiin satamaan, jossa ne pakattiin merikontteihin. Kontituksen kyseisessä satamassa suorittanut satamaoperaattori toimitti kontituksessa käytettyjen merikonttien tiedot Timber by Pinja -järjestelmän asiakkaalle sähköpostilla. Asiakkaan työntekijät syöttivät käsin konttien tiedot Timber by Pinja -järjestelmään. Yhteistyössä asiakkaan ja satamaoperaattorin kanssa konttitietojen hakuun ja käsittelyyn haluttiinkin kehittää uusi, parempi toimintamalli.

8.1 Määrittely

Projektissa mukana olleelta satamaoperaattorilta saatavilla olevista kontitustiedoista tärkeimpiä olivat kontituksessa käytettyjen konttien konttinumerot ja konttien sinettnumerot. Lisäksi projektin määrittelyssä päädyttiin siihen, että näiden tietojen lisäksi Timber by Pinja -asiakkaan oma kontin tilaamiseen liittyvä viitenumero päivitetään Timber by Pinjan asiakastietokantaan erilliseen uuteen sarakkeeseen. Varsinaisten konttitietojen lisäksi myös konttien sisältämien puutavarayksikköjen linkitystä kontituksessa käytettyyn konttiin haluttiin ylläpitää. Näin ollen määrittelyvaiheessa todettiin, että Timber by Pinja -järjestelmässä tulisi olla ajantasaiset tiedot laivakuljetuksissa käytetyistä merikonteista, sekä niiden sisältämistä puutavarayksiköistä. Nämä tiedot haluttiin haettavaksi automaattisesti, jolloin tarve käsin tehtävälle työlle poistuu, sekä myös virheiden mahdollisuus eliminoiduu. Lisäksi integraation tulisi olla tarvittaessa laajennettavissa mahdollisimman helposti myös koskemaan muita satamaoperaattoreita, projektissa mukana olleen operaattorin lisäksi.

8.2 Tekninen malli

Kuten luvuissa 2 ja 3 todettiin, integraatio toteutettiin Pinjan palveluväylää käyttäen. Toistaiseksi projektissa käytetyt palvelut on toteutettu WCF-palveluina, mutta palveluväylän toteutus mahdollistaa myös muilla teknologioilla (esim. gRPC) rakennettujen palvelujen hyödyntämisen. Toteutuksesta haluttiin helposti skaalautuva ja laajennettavissa oleva ja konttitietojen hakua tullaan jatkossa laajentamaan koskemaan myös muita satamaoperaattoreita. Laajentamisen toteuttamiseksi riittää,

kun varsinainen konttitietojen haku ja vastaanotto palveluväylään toteutetaan vastapuolen (satamaoperaattorin) vaatimalla tavalla, sekä saatu raakadata normalisoidaan Timber by Pinjan konttitietoa edustavan luokan standardin mukaiseksi oliomuotoiseksi dataksi. Vastaanottavaan luokkaan ja siitä muodostettuun olioon voidaan lisätä uusia tietojäseniä tarpeen mukaan. Palveluväylä huolehtii ja reitittää oliomuotoisen datan oikealle vastaanottajalle. Kuvassa 12 on havainnollistettu rajapinnan ja palveluväylän toimintaa kokonaisuutena.

8.2.1 Konttitietojen haku

Projektissa mukana olleen satamaoperaattorin konttitietojen haku tapahtuu SFTP-protokollaa käyttäen. Palveluväylää kutsuva palvelu kutsuu tietyin aikavälein varsinaista palveluväylässä sijaitsevaa palvelua hakemaan saatavilla olevat konttitietoja sisältävät XML-muotoiset tiedostot satamaoperaattorin SFTP-palvelimelta. Haku suoritetaan jokaiselle Timber by Pinja -asiakkaalle, jolle palveluväylän tietokantaan on määritetty kyseisen toiminnon vaatimat tiedot. Kun on varmistuttu siitä, että tiedostot on saatu onnistuneesti ladattua, SFTP-palvelimella olevat tiedostot siirretään luettuja tiedostoja varten varattuun kansioon.

Ladattujen XML-tiedostojen sisältö tallennetaan palveluväylän tietokantaan ensin raakadatana, josta se deserialisoidaan edelleen oliomuotoon. Deserialisointia varten palveluväylään rakennettiin luokka, joka pohjautuu PapinetWoodX -tiedostostandardiin, mutta ei kuitenkaan sisällä kaikkea sen tietosisältöä. Deserialisoitu PapinetWoodX-tyyppinen olio muunnetaan Timber by Pinjan standardimuotoiseksi konttitieto-luokan olioksi. Konttitieto-olio serialisoidaan xml-muotoon ja tallennetaan datana tietokantaan odottamaan jatkokäsittelyä. Raakadatarivit, jotka on onnistuneesti muutettu olioiksi, merkitään käsitellyiksi. Palveluväylään rakennettu deserialisointiin ja serialisointiin liittyvä logiikka huolehtii myös virreehallinnasta, mikäli raakadatan muuntaminen oliomuotoon ei onnistu.

8.2.2 Oliomuotoisen datan lähetys

Palveluväylän tietokantaan tallennetut Timber by Pinjan standardimuotoiset konttitietoja sisältävät käsittelemättömät datarivit ja niiden xml-sisältö deserialisoidaan takaisin oliomuotoon ja lähetetään useita olioita sisältävänä listana palveluväylän avulla Timber by Pinja -asiakkaan WCF-palvelun päätepisteeseen. Palveluväylän tietokanta sisältää tiedot asiakkaista ja päätepisteistä ja logiikka

osaa reitittää oikean datan oikealle asiakkaalle. Palveluväylän logiikka huolehtii siitä, että lähetyksen epäonnistuessa sitä yritetään uudelleen. Jonotusjärjestelmän ansiosta voidaan varmistua siitä, että kaikki palveluväylän tietokannassa olevat rivit tulevat lähetetyiksi. Mikäli tietokannassa on virheellisiä rivejä, joiden lähetys ei onnistu, ne voidaan havaita ja tutkia tarvittaessa manuaalisesti.

8.2.3 Tietokantaan tallennus

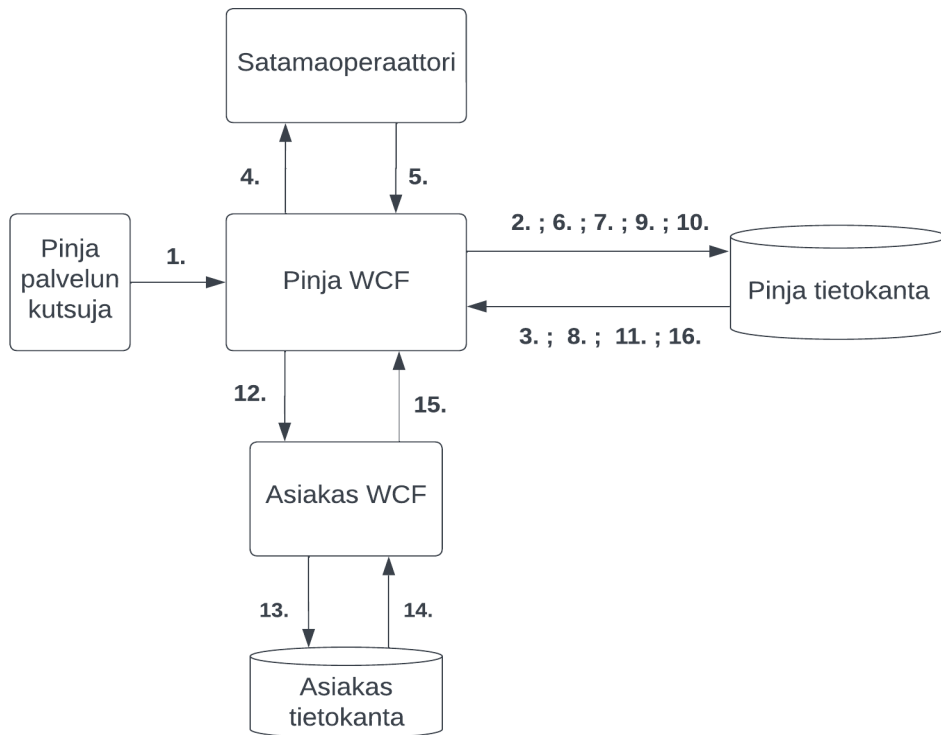
Oliomuotoinen data tallennetaan Timber by Pinja -asiakastietokantaan ja onnistuneesta tallennuksesta palautetaan kuittaus asiakkaan palveluun, joka palauttaa kuittauksen Pinjan palveluväylään. Palveluväylän tietokantaan tallennetaan tieto onnistuneesta tallennuksesta, sekä asiakastietokantaan viittaava ID-arvo, jolla tieto voidaan todentaa. Vastaavasti mikäli tallennus ei onnistu palautetaan virheilmoitus, joka tallennetaan palveluväylän tietokantaan. Näin varmistutaan siitä, että jokainen rivi tulee tallennetuksi asiakastietokantaan.

8.3 Toteutus ja testaaminen

Kuten aiemmissa luvuissa todettiin, projektin ohjelmointiosuus toteutettiin pääosin C#-kielellä ja .NET-sovelluskehyksessä. Ohjelmointiympäristönä toimi Visual Studio 2019. Projektia ja datan lähetysputkea testattiin ajamalla paikallisesti omalla koneella Visual Studion Debug-moodissa kahta eri WCF-palvelua, jotka simuloivat Pinjan ja asiakkaan palveluita. Palvelut olivat yhteydessä kehitysympäristön tietokantoihin. Konttitietojen SFTP-hakuun johtava kutsu toteutettiin testausvaiheessa Postman-ohjelmiston avulla, jolla kutsuttiin paikallisesti simuloitua Pinja-palvelun päätepisettä. SFTP-palvelimena testausvaiheessa toimi Pinjan oma SFTP-palvelin, josta XML-tiedostoja haettiin. Projektin loppuvaiheessa Postmanin avulla testattiin dataputken toimivuus myös tuotantoympäristössä ja satamaoperaattorin SFTP-palvelimelle. Automaattinen ajastetusti toistuva konttitietojen haku aktivoitiin onnistuneen testin jälkeen.

8.4 Toiminta kokonaisuutena

Kuvassa 12 on havainnollistettu konttitietojen haku, lähetys asiakkaalle ja tallennus asiakkaan tietokantaan kokonaisuutena. Kuvan alle on koottu numeroitu selvitys toiminnoista niiden tapahtumajärjestyksessä. Liite 1 havainnollistaa integraatiota kokonaisuutena.



KUVA 12. Rajapinnan ja palveluväylän toiminnallisuus ja datan käsittely

Seloste kuvan 12 tapahtumajärjestyksestä:

1. Palvelun kutsuja kutsuu varsinaista Pinja-palvelua hakemaan konttitiedot.
2. Haetaan satamaoperaattorin päätepisteen tiedot ja autentikointitiedot tietokannasta.
3. Saadaan satamaoperaattorin päätepisteen tiedot ja autentikointitiedot.
4. Haetaan XML-tiedostot satamaoperaattorin palvelimelta SFTP-protokollan avulla.
5. Saadaan XML-tiedostot.
6. Tallennetaan XML-tiedostojen sisältö raakadatana tietokantaan
7. Haetaan tietokannasta kaikki käsittelemättömät raakadatarivit käsiteltäviksi.
8. Tietokannasta saatu raakadata deserialisoidaan ja muunnetaan oliomuotoon.
9. Tallennetaan konttitieto-oliomuotoinen data serialisoituna tietokantaan.
10. Haetaan kaikki lähettämättömät konttitieto-muotoiset rivit tietokannasta lähetettäväksi.
11. Saadaan lähettämättömät konttitiedot ja asiakkaan päätepisteen tiedot.
12. Lähetetään konttitiedot listana olioita asiakkaan päätepisteeseen.
13. Tallennetaan konttitiedot Timber by Pinja -asiakkaan tietokantaan.
14. Kuittaus onnistuneesta tallennuksesta tai virhetilanteessa virheilmoitus.
15. Kuittaus onnistuneesta tallennuksesta tai virhetilanteessa virheilmoitus.
16. Kuittaus onnistuneesta tallennuksesta tai virhetilanteessa virheilmoitus.

9 TULOKSET JA POHDINTA

Merikonttitietoja satamaoperaattorilta integraatiopalveluväylän välityksellä hakeva rajapinta saatiin toteutettua ja toimitettua asiakkaalle käyttöön joulukuun 2022 alkuun mennessä. Muutamia pieniä korjauksia tehtiin vielä testausvaiheessa. Kirjoitushetkellä rajapinta on käytössä tuotantoympäristössä ja konttierittelytiedot liikkuvat automaattisesti. Rajapinnan hakemat ja asiakastietokantaan tallentamat konttitiedot ovat täsmänneet asiakkaan satamaoperaattorilta erikseen sähköpostilla saamiin kontitusraportteihin ja palaute asiakkaalta rajapinnan toiminnasta on ollut myönteistä. Aikataulullisesti projekti venyi hieman odotettua pidemmäksi, mutta kokonaisuutena lopputulosta voi kuitenkin pitää varsin onnistuneena.

Itse projekti kokonaisuutena ja konttierittelytietojen siirtoon käytettävät luokat ja oliot haluttiin toteuttaa mahdollisimman geneerisenä, jotta konttitietojen haku myös muilta satamaoperaattoreilta vaatisi mahdollisimman vähän lisäyksiä ja muutoksia. Kirjoitushetkellä onkin jo suunnitteilla konttierittelytietojen haun laajentaminen koskemaan myös erästä toista satamaoperaattoria. Todennäköistä on, että lisäksi myös muita satamaoperaattoreita saatetaan mukaan tulevaisuudessa. Yhtenä johtajatuksena projektissa olikin alusta saakka, että ainoastaan tiedonhakuun satamaoperaattorilta käytettävä tapa vaihtelee, mutta siitä eteenpäin rajapinnan toiminta on mahdollisimman yhdenmukaista.

Suunnitelmallisuus ja asioiden jakaminen pienempiin osakokonaisuuksiin korostuivat läpi projektin, määrittelystä toteutukseen ja raportointivaiheeseen saakka. Myös huolellinen testaaminen ja poikkeustilanteiden huomioonottaminen olivat tärkeässä roolissa, jotta rajapinnasta saataisiin mahdollisimman varmatoiminen.

Teknologiat, joita projektissa käytettiin, olivat osittain tuttuja jo entuudestaan aiemmista samantyyppisistä kehitystehtävistä Timber by Pinja -tuotteen parissa. Ymmärrys järjestelmän ja siihen rakennettujen integraatioiden toiminnasta syveni ja myös erilaiset integraatiotavat ja niiden toimintamallit tulivat projektin myötä tutuiksi. SFTP-tiedonsiirto Renci SSH -kirjaston avulla oli suhteellisen helppoa omaksua ja palveluväylän toiminta, sekä siinä toimivien palveluiden hyödyntäminen avautuivat varsinaisen koodausvaiheen ohella testaamisen avulla konkreettisesti.

Suunnitelmallisuuden, teknologisen osaamisen ja ohjelmointitaidon parantumisen ohella tärkein oppi, joka projektista tuli, on ehdottomasti ymmärrys aktiivisen kommunikoinnin tärkeydestä asiakkaiden ja sidosryhmien kanssa, mutta myös kehitystiimin sisäisesti. Aikataulut saattavat venyä huomattavastikin, kun tärkeää tietoa jää puuttumaan tai sitä ei osata kysyä ajoissa. Kehitystiimissä on paljon teknologista osaamista ja tietoutta toimialasta, joita täytyy oppia hyödyntämään oikeassa paikassa ja kysyä neuvoa kokeneemmilta työtovereilta tarvittaessa. Oma tietämykseni Timber by Pinja -järjestelmän asiakkaiden, sahojen ja puunjalostuslaitosten liiketoiminnasta ja prosesseista syveni huomattavasti projektin aikana. Osaltaan myös tämä auttaa ymmärtämään järjestelmän toimintaa paremmin ja auttaa kehittymään paremmaksi ohjelmistokehittäjäksi. Myös asiakkaiden kanssa kommunikointi on sujuvampaa, kun omat valmiudet ”puhua samaa kieltä” ovat paremmat. Seuraavia projekteja ja pienempiäkin kehitystehtäviä silmällä pitäen opinnäytetyöprojekti antaa paljon eväitä ja tuo myös itseluottamusta ja varmuutta tulevia haasteita varten.

LÄHTEET

HiQ Finland Oy 2022. Integraatiot ja integraatioalustat – lyhyt oppimäärä. Hakupäivä 16.11.2022.

<https://hiq.fi/ajankohtaista/integraatio/>.

Itewiki.fi 2022. Integraatiot. Hakupäivä 7.11.2022. <https://www.itewiki.fi/opas/integraatiot/>.

Linux.fi 2016. SFTP. Hakupäivä 26.10.2022. <https://www.linux.fi/wiki/SFTP>.

Microsoft 2021. What is Windows Communication Foundation. Hakupäivä 7.12.2022.

<https://learn.microsoft.com/en-us/dotnet/framework/wcf/whats-wcf>.

Microsoft 2022a. Common Language Runtime (CLR) overview. Hakupäivä 10.1.2023.

<https://learn.microsoft.com/en-us/dotnet/standard/clr>.

Microsoft 2022b. Visual Studio. Hakupäivä 11.1.2023. <https://visualstudio.microsoft.com/>.

Microsoft 2022c. Language Integrated Query (LINQ) (C#). Hakupäivä 18.1.2023. <https://learn.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/linq/>.

<https://learn.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/linq/>.

Microsoft 2023. Namespaces. Hakupäivä 25.1.2023. <https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/language-specification/namespaces>.

<https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/language-specification/namespaces>.

Solita Oy 2018. Integraatio-opas. Hakupäivä 10.11.2022. <https://hub.solita.fi/hubfs/Op-paat%20ja%20tiedostot/Integraatio-opas%202018/solita-integraatio-opas.pdf>.

<https://hub.solita.fi/hubfs/Op-paat%20ja%20tiedostot/Integraatio-opas%202018/solita-integraatio-opas.pdf>.

Spiceworks 2022. What Is Cloud Computing? Definition, Benefits, Types, and Trends. Hakupäivä 6.12.2022.

<https://www.spiceworks.com/tech/cloud/articles/what-is-cloud-computing/>.

SSH Academy 2022. SSH Keys. Hakupäivä 3.11.2022. <https://www.ssh.com/academy/ssh-keys>.

<https://www.ssh.com/academy/ssh-keys>.

Stackify 2022. What is IIS? Hakupäivä 13.12.2022. <https://stackify.com/iis-web-server/>.

Tavu.io 2022. SSH-avaimien käyttö Windowsilla Putty-ohjelmistolla. Hakupäivä 1.11.2022. <https://tavu.io/fi/tutorials/ssh-avaimien-kaytto-windowsilla-putty-ohjelmistolla>.

Techopedia 2022. Framework Class Library (FCL). Hakupäivä 10.1.2022. <https://www.techopedia.com/definition/24212/framework-class-library-fcl-net>.

Wikipedia 2021. Järjestelmäintegraatio. Hakupäivä 7.11.2022. <https://fi.wikipedia.org/wiki/J%C3%A4rjestelm%C3%A4integraatio>.

Wikipedia 2022a. SSH. Hakupäivä 1.11.2022. <https://fi.wikipedia.org/wiki/SSH>.

Wikipedia 2022b. Secure Shell. Hakupäivä 1.11.2022. https://en.wikipedia.org/wiki/Secure_Shell.

Wikipedia 2022c. SSH File Transfer Protocol. Hakupäivä 26.10.2022. https://en.wikipedia.org/wiki/SSH_File_Transfer_Protocol.

Wikipedia 2022d. Tietokanta. Hakupäivä 22.11.2022. <https://fi.wikipedia.org/wiki/Tietokanta>.

Wikipedia 2022e. SQL. Hakupäivä 23.11.2022. <https://fi.wikipedia.org/wiki/SQL>.

Wikipedia 2022f. Pilvipalvelu. Hakupäivä 6.12.2022. <https://fi.wikipedia.org/wiki/Pilvipalvelu>.

Wikipedia 2022g. Windows Communication Foundation. Hakupäivä 7.12.2022. https://fi.wikipedia.org/wiki/Windows_Communication_Foundation.

Wikipedia 2022h. Internet Information Services. Hakupäivä 13.12.2022. https://en.wikipedia.org/wiki/Internet_Information_Services.

Wikipedia 2022i. .NET Framework. Hakupäivä 10.1.2023. https://fi.wikipedia.org/wiki/.NET_Framework.

Wikipedia 2022j. Visual Studio. Hakupäivä 11.1.2023. https://en.wikipedia.org/wiki/Visual_Studio.

Wikipedia 2022k. C sharp. Hakupäivä 18.1.2023. https://fi.wikipedia.org/wiki/C_sharp.

