Haaga-Helia
ammattikorkeakoulu Oy

DevParcel - kehitystyökalu prosessien ja palveluiden testaamiseen

Harri Simonen

Opinnäytetyö
Tietojenkäsittelyn koulutusohjelma
2016

Haaga-Helia ammattikorkeakoulu Oy

Tiivistelmä

9.2.2016

Tekijä
Harri Simonen

Koulutusohjelma
Tietojenkäsittelyn koulutusohjelma

Opinnäytetyön otsikko
DevParcel - kehitystyökalu prosessien ja palveluiden testaamiseen

Sivu- ja liitesivumäärä
43 + 15

Opinnäytetyön otsikko englanniksi
DevParcel - Development tool for testing processes and services

Tämä toiminnallinen opinnäytetyö tehtiin toimeksiantona Posti Oy:lle ja sen tavoitteena oli toteuttaa kehitysprojektina sovellus, DevParcel, jonka avulla järjestelmätestaajat pystyvät tuottamaan Postin tietojärjestelmiin lähetyksistä ennakkotietoja sekä tulostamaan osoitekortteja viivakoodillisilla lähetystunnuksilla. Sovellusta käyttämällä Postin tavoitteena on yksinkertaistaa järjestelmätestausta, nopeuttaa järjestelmäkehitystä ja parantaa tuotantojärjestelmien laatua.

Opinnäytetyössä selvitettiin miten sovellukseen voidaan toteuttaa vaaditut toiminnollisuudet Postin vaatimusten mukaisesti. Sovelluksen toiminnollisuus rajattiin Postin liiketoiminnan osalta koskemaan lähetysseurannan piirissä olevia Postin kotimaan paketti- ja kirjetuotteita sekä kansainvälisiä postilähetyksiä. Toteutukseen liittyviä teknisiä rajauksia olivat Postin työasemiin liittyvät tietoturvavaatimukset, työasemissa käytettävissä olevat varusohjelmat sekä oheislaitteet, joiden vuoksi ohjelmointi toteutettiin Microsoft Office 2013 -ohjelmiston Excel-työkirjan päälle Visual Basic for Application -ohjelmointina.

Tietoperustassa perehdyttiin VBA-ohjelmointikielen objektimalliin, tiedostojen käsittelyyn sekä tiedonsiirtoon siltä osin, kun ne liittyivät sovellukseen toteutettuihin toimintoihin. Lisäksi selvitettiin liiketoiminnan vaatimusten osalta erilaisiin lähetystunnuksiin ja Code 128- ja GS1-128 -viivakoodeihin liittyviä standardeja, joista pystyttiin hyödyntämään lähetystunnusten muodostamiseen liittyvien tarkistelukujen laskentakaavoja ja viivakoodin muodostamiseen liittyvää merkkikoodausta.

Opinnäytetyö aloitettiin lokakuussa 2015 ja valmistui helmikuussa 2016. Produkti toteutettiin marraskuussa 2015 ja otettiin käyttöön Postissa joulukuun alussa 2015. Käyttöönottoa jouduttiin aikaistamaan, koska sovellusta tarvittiin Postissa meneillään olevan strategisen tuoteuudistushankkeen järjestelmätestaukseen.

Opinnäytetyön tuloksena kehitetty DevParcel-sovellus on järjestelmätestaajien käytössä Postissa ja täyttää sille asetut alkuperäiset vaatimukset. Sovellukselle on tullut paljon jatkokehitysehdotuksia, joista osa on jo toteutettukin. Sovelluksesta on tällä hetkellä käytössä neljäs versio.

Onnistuneen kehitysprojektin toteutus osoitti, että Excel VBA-ohjelmoinnin avulla on mahdollista toteuttaa nopeasti ja kevyesti vaativiakin ratkaisuja liiketoiminnan tarpeisiin.

Asiasanat
Posti, integraatio, ftp, Excel, Visual Basic for Applications, viivakoodit

Haaga-Helia
University of Applied Sciences Ltd.

Abstract

9 February 2016

Author
Harri Simonen

Degree programme
Business Information Technology

Report/thesis title	**Number of pages and appendix pages**
DevParcel - Development tool for testing processes and services	43 + 15

This functional thesis was assigned by Posti Oy and its target was to implement an application development project, DevParcel, which allows the system testers to produce advance information to Posti's IT systems, as well as to print address labels with barcodes. By using this application, Posti's goal is to simplify the system testing, speed up IT development projects and improve the quality of production systems.

The thesis studied how the required functionalities of the application can be implemented in accordance with the requirements of Posti. Application functionality with relation to business was limited to include only Posti's traceable shipments like Posti's domestic parcel and letter products as well as international mail. Technical limitations of the implementation were related to the security requirements associated with Posti's workstations, available office suite in workstations and connected peripheral devices, which are the reasons why the application was implemented as an Excel workbook with Visual Basic for Application programming in Microsoft Office 2013.

The theoretical part of the thesis discussed the object model of VBA programming language and handling the files and file transfers when related to the activities of the application. In addition, with regard to the business requirements, a connection to different structures of tracking IDs was investigated and found, as well bar code-related standards for Code 128 and GS1-128, which were able to take advantage of the calculation formulas of check digits for tracking IDs and barcode associated character encoding.

The thesis was started in October 2015 and it was completed in February 2016. The DevParcel application was implemented in November 2015 and was introduced at the beginning of December 2015. The introduction had to be done early, because the application was needed for system testing of an ongoing strategic project of product reform in Posti.

The developed DevParcel application, which is the outcome of the thesis, is used by system testers in Posti and fulfills its original requirements. The application has received a lot further development proposals and most of them have already been implemented. The application is currently using the fourth version of it.

The successful implementation of this development project showed that by using the Excel VBA programming, it is possible to implement solutions quickly and lightly for even the most demanding business needs.

Keywords
Posti, integration, ftp, Excel, Visual Basic for Applications, barcodes

Sisällys

1 Johdanto

Ohjelmistokehityksen yksi tärkeimmistä vaiheista on toteutettavien ratkaisuiden riittävä testaaminen, riippumatta toteutettavan ratkaisun koosta. Kun yrityksessä on useita tietojärjestelmiä, muutosten testaus pitää ulottaa myös kaikkiin integraatioiden kautta liittyviin tietojärjestelmiin ja niihin liitettyihin oheislaitteisiin. Integroiduissa järjestelmäkokonaisuuksissa yhden lähdejärjestelmän toiminnallinen tulos on toiselle järjestelmälle tiedonlähde. Kun tietoja jalostetaan järjestelmäketjuissa, myös eri organisaatioiden välillä, niin tietojen merkitys ei saa muuttua alkuperäisestä. Testauksessa on tärkeää varmistaa juuri tuon tiedon oikeellisuus ketjun alusta loppuun. Vaikka tehtyjä muutoksia testataan riittävästi ja hyväksytysti muutoksen kohteena olevan tietojärjestelmän kehitysympäristössä, niin ikäviä yllätyksiä saattaa esiintyä, kun muutokset siirretään lopulliseen testi- tai tuotantoympäristöön hyväksymistestaukseen.

Onnistuneen testauksen edellytyksenä on, että testaajilla on riittävät tiedot muutoksen kohteena olevasta tietojärjestelmästä, siihen toteutettavista muutoksista ja niiden avulla saavutettavista parannuksista yrityksen palveluprosesseihin ja sitä kautta yrityksen tuottamiin palveluihin. Testauksen alkaessa ensimmäisenä haasteena on riittävän hyvän testiaineiston saatavuus tai sen muodostaminen. Testiaineiston pitää olla kohdejärjestelmän rajapinnan määrittelemän mukaisessa muodossa ja sisältää testisuunnitelman mukaisia testitapauksia, joilla testataan määriteltyjen muutosten tavoitteen mukainen toteutuminen kohdejärjestelmässä ja sen muissa integraatiorajapinnoissa.

Käytännössä ongelma on usein siinä, että liiketoiminnan asiantuntijat kyllä tietävät mitä tapauksia pitää testata ja mitkä ovat toivotut testitulokset, mutta heidän taitonsa eivät riitä muodostamaan teknisiä integraatioiden vaatimia tietoja testattavan tietojärjestelmän rajapintaan. Muutosten toteuttajat taas tuntevat tietojärjestelmärajapinnat mutta eivät tunne riittävästi niitä sääntöjä, joita liiketoiminta on asettanut testattaville liiketoimintatapahtumille. Näiden ongelmien ja tiukkojen projektiaikataulujen takia testaus jää usein pintapuoliseksi ja tuotantoon pääsee puutteellisesti testattuja ohjelmia/ohjelmistoja, usein jopa katastrofaalisin seurauksin.

1.1 Opinnäytetyön tavoite

Tämä toiminnallinen opinnäytetyö tehdään toimeksiantona Posti Oy:lle ja sen tavoitteena on toteuttaa kehitystyö, jonka tuotoksena on sovellus, jonka avulla pyritään ratkaisemaan edellä kuvattuja järjestelmätestaukseen liittyviä ongelmia. Toteutettavan sovelluksen, DevParcel-kehitystyökalu, käyttäjinä ovat Postin liiketoiminnan asiantuntijat, Postin ICT-

asiantuntijat sekä Postille ohjelmakehitystä tekevien ulkopuolisten toimittajien ohjelmoijat, jotka pystyvät sovellusta käyttämällä muodostamaan helposti kattavaa ja oikeanlaista testiaineistoa integraatiotesteihin sekä tulostamaan viivakoodattuja osoitekortteja lähetysten käsittelyprosessien testaamiseen. Testaukset liittyvät yleensä Postin liiketoiminnan Pakettipalveluiden ydinprosessien kehittämiseen, joita ovat pakettilajitteluprosessi, nouto- ja jakeluprosessit sekä asiakaslaskutusprosessi.

Postissa ei ole ollut vastaavaa sovellusta käytettävissä ja sen puute on hidastanut tuote- ja palvelukehitystä muutosten testaamisen osalta. Myös puutteelliset ja kuvitteelliset testiaineistot ovat muodostaneet riskin, kun arvioidaan toteutettujen muutosten vaikutuksia integraation kautta liittyviin muihin tietojärjestelmiin.

Postissa on tällä hetkellä menossa Pakettipalveluiden tuotteisiin liittyvä strateginen uudistushanke, jonka tulokset on tarkoitus saada tuotantoon vuoden 2016 aikana. Muutoksia prosesseihin ja tietojärjestelmiin ollaan toteuttamassa nyt ja ensimmäiset järjestelmätestaukset ovat alkaneet joulukuun 2015 puolessa välissä. DevParcel-kehitystyökalu toteutettiin Postissa kehitysprojektina marraskuussa 2015 ja se on ollut testaajien käytössä joulukuun 2015 alusta alkaen.

1.2 Tehtävän asettelu ja opinnäytetyön rakenne

Opinnäytetyön tarkoituksena on selvittää miten sovellus voidaan toteuttaa niin, että sen avulla pystytään muodostamaan vaaditut tunnukset ja tulostamaan ne viivakoodattuina Postin osoitekortteihin sekä muodostamaan asiakasintegraatiota jäljittelevän, Postin määritysten mukaisen rajapintatiedoston Postin lähetystenseurantajärjestelmään. Lähtökohtana on, että sovellus toteutetaan Excel-taulukkona ja Visual Basic for Application – ohjelmointina.

Johdannossa esittelin ongelman, jonka ratkaisemiseen perehdyn opinnäytetyössäni, lisäksi siinä esitellään tutkittavaan alueeseen liittyvä terminologia. Tietoperusta jakautuu kahteen kappaleeseen, joissa selvitän ja perehdyn toimeksiannon määrittelemiin teknisiin vaatimuksiin viivakooditekniikan ja osalta, sekä Visual Basic – ohjelmointikielen mahdollisuuksiin tarvittavan osaamisen saavuttamiseksi. Empiirisessä osuudessa käyn läpi kohdeorganisaation sekä sovellukselle asetetut tavoitteet, ongelmat, toteutussuunnitelman ja toteutuksen. Lopuksi kerron tuloksista ja sovelluksen jatkokehitysmahdollisuuksista.

1.3 Rajaukset

Sovelluksen toiminnollisuus rajataan toimeksiannon perusteella koskemaan Postin Pakettipalveluiden tuotteita, Postin Kirjattu Kirjettä ja kansainvälisiä Postin pakettituotteet (Priority Parcel ja EMS Parcel). Lisäksi tutkitaan mahdollisuutta tulostaa GS1:n määrittelemä kollilappu. Viivakoodit toteutetaan ainoastaan Code 128 -esitysmuodolla, joka sopii kaikkiin Postin ja GS1:n määrittelemiin tunnuksiin ja jota myös pystytään lukemaan kaikilla Postissa käytössä olevilla viivakoodilukijoilla.

Sovelluksen käyttöliittymänä toimii Microsoft Office 2013/Excel 2013-työkirja, jonka päälle toiminnot ja automatisoinnit toteutetaan Visual Basic for Application -ohjelmointina. Syitä tähän rajaukseen ovat käyttäjien vaatimukset, Postin työasemien tietoturvavaatimukset ja toteuttajan omat ohjelmointitaidot. Käyttäjät vaativat helppokäyttöistä sovellusta, joka on helppo kopioida käyttäjältä toiselle. Käyttäjät eivät saa asentaa omiin työasemiinsa ohjelmia, mutta Excel 13 on käytettävissä kaikilla työasemilla. Tietoturvasyistä myöskään suoria tietokantayhteyksiä ei saa toteuttaa, vaan tietoyhteydet on toteutettava ftp/sftp-protokollilla Postin integraatioarkkitehtuurin mukaisesti. Tulostus toteutetaan Postin A5-kokoisille osoitekorttitarra-arkeille, jotka tulostetaan laser-kirjoittimilla. Lämpösiirtotulostusta ei toteuteta tähän sovellukseen, koska lämpösiirtokirjoittimia on käytettävissä ainoastaan Postin tuotantotiloissa ja tulostukset vaatisivat tulostinkohtaiset sovellusratkaisut.

1.4 Toimeksiantajana Posti Oy

Toimeksiantaja kehitystyölle on Posti Oy, jonka ICT-yksikön Customer and Integration -tiimissä toimin Solution Manager -tehtävässä. Olen työskennellyt Postin ICT:ssä vuodesta 1996 lähtien integraatio-asiantuntijana ja pääosin keskittyen Postin Pakettipalveluiden asiakasintegraatioihin. Taustastani johtuen, ongelman aihe ja siihen ratkaisun kehittäminen, kiinnostaa minua erityisesti.

Posti tekee liiketoimintaa neljässä liiketoimintaryhmässä: Postipalvelut, Paketti- ja Logistiikkapalvelut, Itella Venäjä ja OpusCapita. Postitoiminnan historia Suomessa yltää lähes 400 vuoden päähän ja nykyään se palvelee kuluttaja-asiakkaiden lisäksi noin 200 000 yritysasiakasta 10 eri maassa, Suomessa Posti-brändillä ja ulkomailla Itella-brändillä. Liikevaihto oli vuonna 2014 1859 miljoonaa euroa, joka tehtiin noin 23000 henkilön voimin ja jotka edustavat noin 85 eri kansalaisuutta. (Posti Group 2015.)

Opinnäytetyönä toteutettava sovellus keskittyy Postin pakettipalveluiden tuotteisiin ja prosesseihin sekä niihin liittyviin tietojärjestelmäkehityshankkeisiin. Postin yritysasiakkaat lähettävät noin 95 % kaikista paketeista, joista he myös lähettävät ennakkotietoja Postin järjestelmille, ennen kuin paketit saapuvat Postin lajittelukeskuksiin.

Postin järjestelmät vastaanottavat ennakkotietoja yli 300 asiakasjärjestelmästä, joista vastaanotetaan yli 30 miljoonan paketin tiedot vuosittain. Posti hyödyntää tätä tietoa mm. pakettien automaattisessa lajittelussa, kuljetussuunnittelussa, sähköisissä saapumisilmoituksissa, laskutuksessa. Toimivat asiakasintegraatiot ovat avainasemassa automaation toimivuuden kannalta ja siten palvelulupauksen toteuttamisessa. Kun Postin järjestelmiä kehitetään, silloin yleensä myös integraatiorajapinnat muuttuvat ja koko integroitu järjestelmäkokonaisuus on testattava äärimmäisen huolellisesti.

1.5 Opinnäytetyössä käytetty terminologia, keskeiset käsitteet ja lyhenteet

Asiakasintegraatio	Liiketoimintakumppaneiden välinen sopimus yhteistyömenettelystä, jossa asiakkaan tietojärjestelmä muodostaa määrämuotoista tietoa ja siirtää ne toimittajan järjestelmään sovitulla tavalla. Tietoja siirretään usein molempiin suuntiin. Menetelmän tavoitteena on parantaa palvelun toimivuutta sekä tuottaa automaation seurauksena rahallista hyötyä molemmille osapuolille.
CEN	European Committee for Standardization, joka on kaikkien EU- ja EFTA-maiden standardoimisjärjestöjen yhteistyöelin. Suomea CEN:ssä edustaa SFS. CEN laatii EN-standardeja.
DevParcel	Opinnäytetyössä toteutettava sovellus.
GS1	Kansainvälinen voittoa tavoittelematon järjestö, joka on tunnettu mm. EAN-, GTIN- ja SSCC-tunnusten hallinnasta. Suomessa järjestön toiminnasta vastaa Keskuskauppakamarin omistama tytäryhtiö GS1 Finland Oy.
ISO	International Organization for Standardization, joka on maailmanlaajuinen standardoimisjärjestö. ISO:lla on jäseniä 164 maasta ja Suomea edustaa SFS. Järjestö laatii ISO-standardeja.

Kolli	Pienin kokonaisena kuljetettava lähetykseen kuuluva logistinen yksikkö, johon toimitettavat tuotteet on pakattu. Postissa jokainen kolli yksilöidään lähetystunnuksella.
Lähetys	Lähettäjän kuljetettavaksi antama tavaraerä, joka kuljetetaan yhtenä kuljetuksena yhdelle vastaanottajalle. Lähetys voi sisältää yhden tai useamman kollin.
Lähetystiedot	Paketti- tai kirjelähetyksien tiedot sisältävät sekä lähettäjän että vastaanottajan nimi-, osoite- ja yhteystiedot sekä kuljetuspalvelun toteuttamiseen ja laskuttamiseen liittyviä tietoja.
Lähetystunnus	Yhden kuljetettavan pakkauksen eli kollin (paketti, kirje, lava, rullakko) yksilöivä ja ainutkertainen tunnus, joka on tulostettu lähetyksen päälle kiinnitettyyn osoitekorttiin ja luettavissa myös viivakoodina. Tunnetaan myös seurantatunnuksena (Tracking ID). Postissa lähetystunnus ilmoitetaan kollitasolla.
Makro	Makro sisältää automaattisesti toistettavan sarjan ohjelman omia komentoja.
Osoitekortti	Paketeissa ja kirjeissä käytettävä tarrapintainen tuloste, johon on tulostettu lähetyksen käsittelyyn ja toimittamiseen liittyviä tietoja.
Postin lähetystenseurantajärjestelmä	Tietovarasto seurattavien pakettien ja kirjeiden lähetys- ja tapahtumatiedoille.
S10	UPU:n standardi 13-merkkiselle lähetystunnukselle, jota käytetään mm. Priority- ja EMS-lähetyksille sekä Kirjatulle Kirjeelle.
SFS	Suomen Standardisoimisliitto ry, joka ohjaa ja koordinoi standardisointia Suomessa.
SSCC-tunnus	Serial Shipping Container Code, jota käytetään kollien yksilöimiseen ja joka vastaa Postissa lähetystunnusta.
Tapahtumatiedot	Paketti- tai kirjelähetyksien tapahtumatiedot sisältävät kuljetuspalvelun aikana kerättyä tietoa lähetyksiin kohdistuneista käsittelytoimenpiteistä sekä niiden tuloksista.
UAT-testaus	Hyväksymistestaus, joka suoritetaan asiakkaan laitteistossa ja asiakkaan työympäristössä.

UPU	Universal Postal Union, Postien toimintaa säätelevä ja ohjaava YK:n alajärjestö, joka palvelee sen 192 jäsenmaataan muodostaen maailman suurimman jakeluverkon yli 660000 postitoimipisteellä ja yli 5 miljoonalla postityöntekijällä. Posti Oy:n toimiluvassa on velvoite huolehtia Suomessa UPU-valtiosopimusten velvoitteista postiliikenteessä.
VB	Visual Basic on Microsoftin 4. sukupolven tapahtumaohjattu ohjelmointikieli ja siihen integroitu kehitysympäristö, jonka viimeisin versio on 6.0.
VBA	Visual Basic for Application on VB:n päälle toteutettu tulkkaava ohjelmointikieli, joka käännetään konekielelle (P-code) ohjelman suorituksen yhteydessä. VBA sisältää MS Office ohjelmiin integroidun kehitysympäristön.
VBE	Visual Basic Editor on MS Officeen integroitu VBA sovellusten kehitysympäristö, joka on ollut käytettävissä MS Office 97 -versiosta lähtien.
WinAPI	Windowsin tarjoama avoin ohjelmointirajapinta, Application programming interface, yleisimmille toiminnoille.

2 Visual Basic for Applications

Aluksi on tärkeää ymmärtää, että Visual Basic (VB) ja Visual Basic for Application (VBA) ovat erillisiä ohjelmointivälineitä, vaikka VBA muistuttaa vanhempaa ja laajempaa sukulaistaan Visual Basicia. (Shepherd 2006, ix.)

VB-projektissa ohjelmoija määrittelee itse kaikki ohjelmassa käytettävät komponentit, jotka lopulta käännetään suoritettavaksi exe-ohjelmaksi. Ohjelmiston asennus toteutetaan usein niin, että ohjelmoija luo ohjelmassa käytetyistä komponenteista asennuspaketin, joka sisältää käännetyn ohjelman lisäksi myös VB:n yleiset suorituksenaikaiset komponentit. VB:llä voidaan myös tehdä ohjelmakirjastoja (DLL) muiden Windows-ohjelmien käyttöön. (Merensalmi 2007, 4.)

VBA toimii sen sijaan aina isäntänä olevan Office-sovelluksen päällä. VBA-projektin Visual Basic Editorilla kirjoitettu ohjelmakoodi tallentuu aina isäntäsovelluksen yhteyteen, esimerkiksi Excel-työkirjaan, samoin tallennetaan myös käyttäjän nauhoittamat makro-ohjelmat. VBA-ohjelma on siirrettävissä toisille käyttäjille isäntädokumentin mukana yhtenä Office-tiedostona, esimerkiksi Excel-työkirjana. Yhdessä VBA-ohjelmassa voidaan käyttää myös muiden Office-ohjelmien komponentteja. Vaikka VBA:lla ei voida toteuttaa DLL-ohjelmakirjastoja, niin VBA pystyy kuitenkin käyttämään kaikkia Windowsin avoimia ohjelmakirjastoja API-rajapinnan kautta. (Merensalmi 2007, 4-5; Shepherd 2006, 189–191.)

Samoin erillisiä ohjelmointiympäristöjä ovat myös Microsoft Visual Studio Tools for Office (VSTO), Visual Basic .NET ja Microsoft Visual C#, joista VSTO:lla voidaan myös toteuttaa Exceliä ja muita Office-tuotteita hallinnoivia ohjelmia. Myös Visual Basic .NET-kehitysympäristössä voidaan tuottaa VBA-yhteensopivia COM–lisäosia (Component Object Model), joiden tiedostopäätteet ovat .exe tai .dll. Niitä voi lisätä ja käyttää VBA-projektissa, vaikka niiden lähdekoodia ei pystykään katsomaan. (Walkenbach 2013, Kindle Location 715, 19720.)

2.1 VBA:n kehityspolku

Microsoft julkaisi ensimmäisenä tuotteenaan BASIC-kääntäjän jo vuonna 1975, jonka jälkeen IBM julkaisi BASICA-tulkin ja Microsoft vastaavan GWBASIC-tulkin. Näiden tulkkien jälkeen Microsoftin ja IBM:n ulkopuolella kehitettiin QuickBASIC-kääntäjä vuonna 1983, jonka avulla käännettyjä BASIC-ohjelmia voitiin suorittaa monta kertaa nopeammin kuin aikaisempien BASIC-tulkkien avulla. Vuonna 1991 Microsoft julkaisi 4. sukupolven Visual

Basic 1.0 -ohjelmointikielen (VB), joka pohjautui aiempaan graafisen käyttöliittymän sisältävään QuickBASIC-kääntäjään, jolla käännettiin BASIC-ohjelma suoritettavaksi exe-tiedostoksi. VB sisälsi kääntäjän lisäksi myös ohjelmoinnin aikaisen tulkin. VB saavutti laajan suosion ja Microsoftin 1990-luvun lopulla suorittaman tutkimuksen mukaan noin kaksi kolmesta PC-sovelluksesta oli toteutettu Visual Basicilla. Viimeisin VB:n versio 6.0 on julkaistu vuonna 1998, mutta se on edelleen laajassa käytössä, vuonna 2002 julkaistusta VB.NET:stä huolimatta. (Mack 2002; Walkenbach 2013, Kindle Location 3264–3279.)

Excel on maailman eniten käytetyin taulukkolaskentaohjelma, joka alun perin kehitettiin Macintosh-tietokoneisiin vuonna 1985 ja PC:lle ensimmäinen Windows-versio Excel 2.0 julkaistiin vuonna 1987. Tällä hetkellä uusin Windows-käyttöjärjestelmään tehty versio on Excel 2016 (versio 16), joka sisältyy lokakuussa 2015 julkaistuun Office 2016-pakettiin. (Walkenbach 2013, Kindle Location 869.)

Microsoft julkaisi Visual Basic for Application (VBA) ohjelmointikielen vuonna 1994 Excel 5.0:n yhteydessä, mutta sen käyttö laajeni vasta Excel 97:n yhteydessä. VB ja VBA ovat täysin erillisiä ohjelmointivälineitä, joita yhdistävänä tekijänä on kuitenkin ohjelmointikieli VB 6.0. VBA toimii kuitenkin aina Microsoftin Office-sovellusten päällä ja sitä ei tarvitse erikseen kääntää suoritettavaksi ohjelmaksi. Kaikki Office-sovellukset (Excel, Access, Word, Powerpoint, Outlook, MS Project) sisältävät VBA-ympäristön, jonne voidaan siirtyä sovelluksesta käynnistämällä Visual Basic Editorin (VBE). Makroja ja VBA-ohjelmakoodia sisältävän Excel-työkirjan tunnistaa siitä, että tavallinen Excel-työkirja tallennetaan yleensä *.xlsx – muodossa, mutta VBA-ohjelmakoodia tai makroja sisältävä Excel-työkirja tallennetaan *.xlsm – muodossa. (Merensalmi 2007, 4-5; Walkenbach 2013, Kindle Location 3279.)

2.2 Excel VBA-ohjelmoinnin perusteet

Ohjelmoinnin rakenteet ovat ohjelmointikielestä riippumatta samanlaisia, ainoastaan komennot kirjoitetaan erilaisilla syntakseilla. Kaikissa ohjelmointikielissä rakenne perustuu ohjelmointiprojektiin, johon määritellään yksi tai useampia moduuleita, jotka sisältävät aliohjelmia. Aliohjelmissa määritellään muuttujat, asetetaan muuttujille arvoja, rakennetaan ehtolauseita, tehdään silmukkarakenteita ja käytetään muita tietojen hallintaan liittyviä komentoja. Kun Exceliin perustetaan VBA-projekti, niin myös silloin toimitaan samoin. . (Merensalmi 2007, 49.)

2.2.1 Excelin VBA-projektin rakenne

Kun Excelin työkirjasta avaa VBA-projektin Visual Basic Editorilla (VBE), niin saa näkyviin työkirjaan (Workbook) liittyvän VBAProjectin rakenteen ja siihen kuuluvat komponentit. VBA on objektiorientoitunut ohjelmointikieli ja ensimmäinen komponenttikin on nimeltään Microsoft Excel Objects, joka sisältää vähintään yhden taulukon (Worksheet) ja itse työkirjan. Seuraavat komponentit lisätään projektiin ohjelmointitarpeen mukaisesti. Toinen komponentti sisältää ohjelmoijan määrittelemät lomakkeet (Forms), jotka sisältävät käyttäjälomakkeiden ulkoasut näyttökontrolleineen sekä niitä ohjaavat ohjelmakoodit. Kolmas komponentti sisältää ohjelmamoduulit (Modules), joihin ohjelmoija määrittelee haluttuja toimintoja suorittavat aliohjelmat eli proseduurit ja funktiot. Aliohjelmat muodostavat rajapinnan työkirjan kanssa, koska työkirjasta voi kutsua ainoastaan aliohjelmia. Neljäs komponentti sisältää luokkamoduulit (Class Modules), jossa ovat ohjelmoijan määrittelemät omat uudet objektit ja kokoelmat, joita ei suoriteta muiden moduulien lailla vaan niihin viitataan muiden moduulien koodeista. (Merensalmi 2007, 50–51; Walkenbach 2013, Kindle Location 3442–3522, 25083; Shepherd 2006, 199.)

2.2.2 Excelin VBA-objektimalli ja objektien hierarkia

VBA on ohjelmointikielenä objektiorientoitunut (Object-oriented Programming, OOP), jossa ohjelmointi perustuu käytettävän sovelluksen objektimalliin. Esimerkiksi Excelin objektimalli perustuu työkirjoihin ja taulukoihin sekä niiden ominaisuuksiin, menetelmiin ja tapahtumiin, mutta samalla voi myös laajentaa Excelin objektimallia käyttämällä muiden Office-sovellusten objekteja. Tästä esimerkkinä on sähköpostin kirjoittaminen ja lähettäminen, johon Excel voi käyttää Outlookin objektimallia tai Access, jonka avulla voi luoda tietokannan tiedoista Excel-taulukon ilman kohdeohjelman avaamista. (Shepherd 2006, 127, 167–172.)

Excel-sovellus rakentuu kolmesta tasosta, joita ovat käyttöliittymä, objektimalli ja tiedonkäsittelytoiminnot. Yleensä työkirjan taulukko toimii käyttöliittymänä, joka kommunikoi käyttäjän kanssa. Objektimalli ottaa vastaan käskyjä käyttöliittymästä ja käynnistää tehtävän hoitamiseksi tarvittavat objektit, joita ovat esimerkiksi työkirja-objektilla työkirjan avaaminen, sen tallentaminen tai sen poistaminen. Excelin objektimalli sisältää yli 200 erilaista objektia, joita voidaan hallita ohjelmakoodilla ja jotka sisältävät erilaisia menetelmiä. Objektit rakentuvat hierarkkisesti Application-objektin siis Excelin alle, jolloin seuraavan tason muodostaa Workbook-objekti (työkirja), jonka alla on Worksheet-objekti (taulukko), jonka alla on Range-objekti (solu tai solualue). Objekti on ohjelmoinnin perusyksikkö,

jolla voi olla omia asetuksia, ominaisuuksia ja menetelmiä. Objekti erotetaan ominaisuudesta tai toisista objekteista pisteellä alisteisuuden mukaan ja aina hierarkkisessa järjestyksessä. (Merensalmi 2007, 41; Shepherd 2006, 127–128; Walkenbach 2013, Kindle Location 883, 3329-3355.)

2.2.3 Kokoelmat, ominaisuudet, metodit ja tapahtumat

Kokoelmat (Collection) tarkoittaa objekteja, joiden sisältö koostuu samanlaisten objektien ryhmästä. Tästä esimerkkinä on taulukko (Worksheet), joita voi olla yhdessä työkirjassa useampia ja ne esitetään yhtenä taulukkokokoelmana (Worksheets). Kokoelmatyyppisten objektien nimet esitetään monikossa ja yksittäiseen objektiin viitataan yleensä sen nimellä tai järjestysnumerolla.

Ominaisuus (Property) on skalaarinen attribuutti, joka määrittää objektille mahdolliset parametrit. Esimerkiksi taulukon ominaisuuksia ovat nimi ja näkyvyys. Jos objektin ominaisuus ei ole kirjoitussuojattu, niin sitä voidaan muuttaa.

Metodi (Method) tai menetelmä on objektiin liittyvä toiminnollisuus, joka muuttaa objektin jotain ominaisuutta. Sen avulla voidaan toiminta kohdistaa juuri oikeaan objektiin. Esimerkiksi solun kopiointi tapahtuu Range-objektin Copy-metodilla.

Tapahtuma (Event) on objektiin kohdistunut muutos, jolloin objekti itse käynnistää tapahtuman ja objektin tapahtumaan kirjoitettu koodi käynnistää automaation. Esimerkkinä tapahtumasta on työkirja-objektin sulkeminen, joka voi käynnistää työkirjan automaattisen tallentamisen.

(Merensalmi 2007, 42; Shepherd 2006, 128–136.)

2.3 Tekstitiedostojen muodostaminen VBA:ssa

Kun ohjelmoidaan Windows-ympäristössä, niin on tärkeää että hakemistoja ja tiedostoja pystytään luomaan, siirtämään ja poistamaan. Vaikka Excelin pystyy lukemaan ja kirjoittamaan useita erilaisia tekstitiedostoja, niin usein nämä Excelin sisäänrakennetut toiminnot eivät täytä vaativan ohjelmoinnin tarpeita. VBA-ohjelmoinnissa voidaan käyttää VB:n toimintoja, jotka mahdollistavat levyasemien, hakemistojen ja tiedostojen käsittelyn kahdella tavalla: käyttämällä uudempaa objektimallilla File System Object (FSO), joka toimii

Excel 2000 alkaen tai käyttäen vanhempia alemman tason input/output (I/O) -lausekkeita, jotka toimivat kaikissa Excelin versioissa ja tarjoavat kattavamman tiedostohallinnan kuin Excelin omat tiedostokäsittelytoiminnat. (Microsoft 2015a; Walkenbach 2013, Kindle Location 23232–23658.)

2.3.1 File System Object Model (FSO)

VB:n FSO-malli mahdollistaa objekti-pohjaisen työkalun hakemistojen ja tiedostojen käsittelyyn ja tarjoaa tutun objekti.metodi-syntaksin myös hakemistojen ja tiedostojen ominaisuuksien, metodien ja tapahtumien käsittelyyn. FSO yksinkertaistaa tiedostojen käsittelyä tallentamalla tiedostot tilaa ja laitteen resursseja säästäen sekä helposti käsiteltävässä muodossa. FSO sisältyy Microsoftin Scripting Runtime -kirjastoon (scrrun.dll), joka saadaan VBA-projektin käyttöön VBE:n Tools/References-työkalun kautta. (Microsoft 2015a.)

Tosin, Walkenbachin (2013, Kindle Location 23488) mukaan, Windows Scripting -kirjaston käyttö saattaa levittää viruksia ja muita haittaohjelmia, joten sen käyttäminen saattaa olla estetty joissain ympäristöissä. Lisäksi hän kehottaa olemaan varovainen Windows Scripting -kirjaston käytössä, silloin kun sovellusta on tarkoitus käyttää eri ympäristöissä, koska jotkut virustorjuntaohjelmistot häiritsevät Windows Scripting -kirjaston käyttöä.

FSO-malli sisältää seuraavat objektit (Microsoft 2015a):

- Drive-objekti tarjoaa tietoja levy- tai verkkoasemista tai muista järjestelmän fyysistä ja loogisista asemista esimerkiksi GetDrive- tai GetFolder-metodeilla.
- Folder-objekti mahdollistaa kansioiden käsittelyn ja tarjoaa tietoja niiden ominaisuuksista esimerkiksi Folder.Move-, Folder.Name- ja Folder.Delete-metodeilla.
- File-objekti mahdollistaa tiedostojen käsittelyn ja tarjoaa tietoja niiden ominaisuuksista, käyttämällä esimerkiksi metodeja File.Copy tai File.Delete.
- FileSystemObject-objektiryhmä sisältää kaikki FSO-mallin objektit.
- TextStream-objekti mahdollistaa tekstitiedostojen lukemisen ja kirjoittamisen esimerkiksi Read-, ReadLine- tai ReadAll -metodeilla.

FSO-malli ei tue vielä Random- ja Binary-tyyppisten tiedostojen luontia mutta niitä voi luoda perinteisillä input/output-lausekkeilla (Microsoft 2015a).

2.3.2 Perinteiset tiedoston input/output-lausekkeet ja toiminnot

Perinteiset tiedoston I/O-lausekkeet ja -toiminnot ovat olleet käytössä VB:n ensimmäisestä versiosta alkaen ja ovat edelleenkin täysin tuettuja VB 6.0:ssa, vaikka ne on jätetty pois FSO-mallista (Microsoft 2015a).

Levyllä sijaitseva tiedosto koostuu sarjasta tavuja, joilla esitetään esimerkiksi merkkejä, kokonaislukuja, tietueita, merkkijonoja. Riippuen tiedoston sisällöstä, tiedosto voidaan avata kolmella eri tavalla (Microsoft 2015a):

- Sequental-tyyppinen käyttö sopii tekstitiedostoihin, jossa tiedosto sisältää ainoastaan ANSI-merkkejä ja rivit päättyvät Newline-merkkiin (NL).
- Random Access-tyyppinen käyttö sopii tiedostoihin, joissa kaikilla tietueilla on sama tietuerakenne ja kaikki tietueet ovat yhtä pitkiä. Tiedot tallentuvat tiedostoon kuitenkin Binary-tyyppisesti.
- Binary-tyyppisessä käytössä tiedostoihin voi tallentaa kaiken tyyppisiä tietoja lähes rajoituksetta mutta tiedostoja luettaessa pitää tietää tiedoston tarkka rakenne.

Tiedoston käyttötavasta riippuen, tiedostoa voidaan käsitellä erilaisilla lausekkeilla ja toiminnoilla (taulukko 1). Kuitenkin kaikille tiedostoille, riippumatta tiedoston avaustavasta, ovat käytettävissä seuraavat toiminnot: Dir, EOF, FileCopy, FileDateTime, FileLen, FreeFile, GetAttr, Loc, LOF, Seek, ja SetAttr. (Microsoft 2015a).

Taulukko 1. Tiedostojen avaustyyppeihin liittyvät tiedostojen käsittelylausekkeet ja toiminnot (Microsoft 2015a).

Statements & Functions	Sequential	Random	Binary
Close	X	X	X
Get		X	X
Input( )	X		X
Input #	X		
Line Input #	X		
Open	X	X	X
Print #	X		
Put		X	X
Type...End Type		X	
Write #	X		

Ennen kuin tiedostoja luetaan tai kirjoitetaan, niin ne on avattava tai luotava Open-komennolla ja käyttötarkoituksen mukaisella avaamistavalla (luku, kirjoitus tai lisäys). Lisäksi avaamisen yhteydessä on annettava tiedostonimi ja tiedoston numero (1-511). Tämän jälkeen kaikissa luku- ja kirjoitus-operaatioissa viitataan ainoastaan tiedoston numeroon ja tiedosto myös suljetaan (Close) käytön jälkeen viittaamalla tiedoston numeroon. Varsinaiset tiedoston luku- ja kirjoitusoperaatiot suoritetaan edellä mainitun taulukon (taulukko 1) mukaisilla komennoilla, tiedoston avaustavasta riippuen. Tarvittaessa avaamisen yhteydessä voidaan määrittää myös halutaanko tiedosto lukita muilta käyttäjiltä tai ohjelmilta siksi ajaksi kun tiedosto on auki. Lisäksi muita tarvittavia parametreja voi olla esimerkiksi tietueenpituus (Len), luettava tietueen järjestysnumero, konversio ANSI:sta UNICODE:een. (Merensalmi 2007, 165–168; Microsoft 2015a.)

2.3.3 Postin lähetystenseurantajärjestelmän rajapinta vastaanotettaville ennakkotiedoille

Postin lähetystenseurantajärjestelmän sisäinen integraatiorajapinta perustuu määrämuotoiseen peräkkäistiedostoon, jossa tietueet esitetään hierarkkisesti, tietuerakenteet vaihtelevat ja joissa tietueiden kentät ovat positiosidonnaisia. Tietueet päättyvät aina CRLF-rivinvaihtoon (0x13 ja 0x10) ja käytetyt merkit perustuvat ISO-8859-1 (Latin-1) -merkistöön. (Liite 2.)

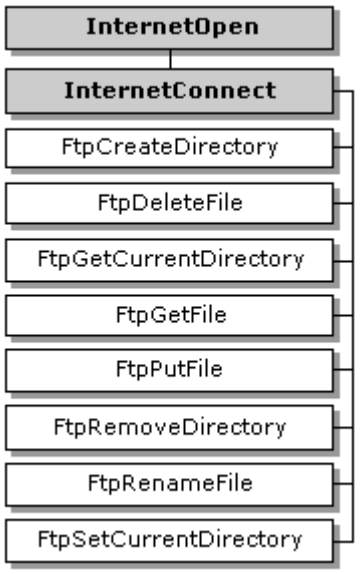
2.4 Automaattisen ftp-tiedonsiirron toteuttaminen VBA-ohjelmoinnilla

Windowsin Internet (Wininet) ohjelmointirajapinta (API) mahdollistaa sovellusten Internet-käytön mm. ftp- ja http-protokollilla ja se toimitetaan Windows-käyttöjärjestelmän mukana wininet.dll -tiedostona. Siten se on myös käytettävissä kaikissa Windows-käyttöjärjestelmän sisältämissä laitteissa. Alun perin Wininet API oli tarkoitettu toimimaan MS Internet Explorer-selaimen kanssa mutta sitä käytetään myös laajasti muissa verkkoresursseja käyttävissä sovelluksissa. (Microsoft 2015b.)

Wininet-ohjelmointirajapinnan kautta, sovellukset voivat ftp-istunnon (FTP session) aikana

- liikkua palvelimen hakemistoissa ja listata hakemistojen sisältöjä
- luoda ja poistaa hakemistoja sekä nimetä niitä uudelleen
- siirtää tiedostoja palvelimelle ja palvelimelta
- poistaa tiedostoja ja nimetä niitä uudelleen. (Microsoft 2015c.)

Ftp-istunnon muodostaminen alkaa HINTERNET handles (kahvojen) luomisesta. Ensin pitää luoda root handle InternetOpen-toiminnolla ja sen jälkeen FTP session handle InternetConnect-toiminnolla. Kuviossa 1 on esitetty hakemistojen ja tiedostojen käsittelytoiminnot (vaaleat laatikot), joita käytettäessä tarvitaan InternetConnect-toiminnon palauttamaa HINTERNET-kahvaa, jota luodessa taas tarvitaan InternetOpen-toiminnon palauttamaa HINTERNET-kahvaa. (Microsoft 2015c.)



Kuvio 1. Ftp-istuntoon liittyvät toiminnot (Microsoft 2015c)

Kun sovellus on suorittanut ftp-toiminnot, niin molemmat HINTERNET-kahvat on myös suljettava InternetCloseHandle-toiminnolla, jolloin ensin suljetaan InternetConnect-kahva ja sen jälkeen InternetOpen-kahva. (Microsoft 2015c.)

Niblack esittää blogi-sivuillaan neljän kohdan tiivistetyn toimintalistan ftp-yhteyden toteuttamiseksi, kun tarkoituksena on siirtää tiedosto työasemasta palvelimelle (Niblack 2008):

- Määritetään laiteympäristö InternetOpen-toiminnolla.
- Otetaan yhteys palvelimeen InternetConnect-toiminnolla.
- Siirretään tiedosto palvelimelle FtpPutFile-toiminnolla.
- Suljetaan alussa luodut kahvat yksi kerrallaan InternetCloseHandel-toiminnolla.

Ennen istunnon avaamista pitää olla selvillä otetaanko yhteys suoraan palvelimeen vai proxy-palvelimen kautta, palvelimen yhteystiedot, tarvitaanko aktiivinen vai passiivinen ftp-yhteys ja siirretäänkö ASCII- vai binääri-muotoisia tiedostoja. (Microsoft 2015c; Niblack 2008.)

3 Licence Plate-, S10- ja SSCC-lähetystunnukset sekä Code 128 -viivakoodi

Tietojenkäsittely voidaan jakaa neljään osa-alueeseen: perinteiseen tietojenkäsittelyyn, toimistoautomaatioon, tietoliikenteeseen ja tuotantoautomaatioon. Lähetysten seuranta kuuluu kuljetusyrityksen näkökulmasta ensisijaisesti tuotantoautomaation alueelle, jossa käsitellään kaikkea tuotantoprosesseihin liittyvää tietojenkäsittelyä. Ongelma ei johdu tarjolla olevan tiedon puutteesta, koska tietoa on saatavilla runsaasti eri lähteistä, vaan siitä, miten tietoja pystytään tunnistamaan, tallentamaan ja käsittelemään tehokkaasti, niin että oikeat tiedot ovat käytettävissä oikealla hetkellä ja oikeassa paikassa. Logistisen toimitusketjun aikana tietoja voidaan kerätä useilla eri menetelmillä. Lähetyksen mittatietoja saadaan automaattisilta painon ja tilavuuden mittaavilta laitteilta, sijaintitietoja autojen GPS-laitteilta ja kuvia luovutuskuittauksia jakelukuljettajien mobiililaitteista. Tärkeänä tekijänä on käsiteltävän lähetyksen tunnistaminen, johon muut kerätyt tiedot voidaan yhdistää, käytettävissä olevien, lähettäjän ilmoittamien sähköisten ennakkotietojen lisäksi. Kuljetusyrityksissä käytetyimpiä tekniikoita lähetyksen tunnistamiseen ovat radiotaajuuteen perustuvat RFID-tunnisteet ja optiseen tunnistamiseen perustuvat viivakoodit, joista yksiulotteinen (1D) GS1:n määrittelemä SSCC-koodi on vakiintuneessa käytössä ympäri maailman. (Hokkanen & Karhunen 2014, 225-245; Sakki 2014, 14-18.)

Universal Postal Union (UPU) on julkaissut useita standardeja lähetystunnuksien muodostamiseen ja niiden esittämiseksi yksiulotteisella viivakoodilla (1D), kaksiulotteisella viivakoodilla (2D) sekä erilaisilla RFID-tunnisteilla. Kansainvälisissä postilähetyksissä (kirjeet ja paketit) on vakiintuneessa käytössä UPU/S10-standardin mukaiset lähetystunnukset, jotka esitetään Code 128- tai Code 39-viivakoodeilla. Suomessa Posti on käyttänyt vuodesta 2000 alkaen myös UPU/S24-standardiin perustuvaa Licence Plate-tunnusta. (Universal Postal Union 2014; Universal Postal Union 2015.)

3.1 UPU:n Licence Plate -lähetystunnus

UPU on määritellyt lähetyksissä käytettävän Licence Plate -lähetystunnuksen (standardit ISO 15459 ja EN 1572) rakenteen UPU:n S24-6-stadardissa seuraavasti (liite 3):

- Data identifier (DI) (ISO 15418), jossa arvo “J” ilmoittaa tunnuksen olevan ISO:n määrittelemän Licence Plate -rakenteen mukainen.
- Issuing Agency Code (IAC), jossa arvolla ”J” ilmoitetaan UPU:n hallinnoivan tunnuksen loppuosaa.

- Defining Authority, jossa arvolla "FI" ilmoitetaan Suomen postihallinnon (Posti) hallinnoivan tunnuksen loppuosaa.
- Format identifier -arvoa Posti ei käytä.
- Data value -arvo muodostuu Postin ohjeiden mukaan asiakkaan sopimustunnuksesta (6 numeroa) sekä juoksevasta asiakkaan muodostamasta yksilöllisestä tunnisteesta (11 numeroa).

Lopputuloksena saadaan lähetyksen yksilöivä tunnus, joka on aina 21 merkkiä pitkä. Esimerkkinä "**JJFI**654321**01234567890**", jossa "JJFI" on aina vakio, "654321" on Postin asiakkaalle ilmoittama asiakaskohtainen sopimustunnus ja "01234567890" on asiakkaan muodostaman lähetyksen yksilöivä osa. Lähetystunnus voidaan käyttää uudelleen 18 kuukauden jälkeen.

Posti käyttää edellä kuvattua Licence Plate -lähetystunnusta Suomen sisäisissä kuljetuksissa sekä vähäisin määrin myös Suomen ulkopuolelle lähtevissä kuljetuksissa. Suurimmassa osassa Suomen ulkopuolelle lähtevissä lähetyksissä käytetään UPU:n S10-standardin mukaisia lähetystunnuksia. Licence Plate -lähetystunnus tulostetaan osoitekortille aina Code 128 -viivakoodina.

3.2 UPU:n S10 -lähetystunnus

UPU määrittelee S10-standardissa 13-merkkisen lähetystunnuksen seuraavasti (Universal Postal Union 2014, 4):

- Standardi koskee paketteja, rekisteröitäviä kirjeitä ja EMS-lähetyksiä, jotka kuljetaan kansainvälisinä postilähetyksinä postihallintojen välillä.
- 13-merkkinen lähetystunnus muodostuu kahdesta kirjaimesta (tuotekoodi), yhdeksästä numerosta ja kahdesta kirjaimesta (maakoodi, ISO 3166-1).

Yhdeksän numeroinen luku, muodostuu kahdeksan numeron pituisesta yksilöivästä tunnisteesta ja yhden numeron pituisesta tarkistenumerosta, joka lasketaan painotetun modulo 11 -kaavan mukaisesti. Lähetystunnus voidaan tulostaa osoitekortille Code 128 -viivakoodina. Esimerkkinä EMS-lähetystunnus "EE99900005**1**FI". (Universal Postal Union 2014, 3,7.)

3.3 GS1:n SSCC-tunnus

Serial Shipping Container Code (SSCC) on standardimuotoinen lähetyksen yksilöivä tunnistenumero, joka on uudelleen käytettävissä vuoden kuluttua. SSCC-koodin luo aina lähettävä yritys, joka hankkii GS1:ltä Company Prefix -tunnuksen (GCP), jonka avulla yritys pystyy itse määrittelemään oman yksilöllisen SSCC-tunnussarjan lähetyksiään varten. (GS1 2015.)

SSCC-koodi on kokonaisuudessaan aina 20-numeroinen tunnus, joka koostuu sovellustunnuksesta, laajennustunnuksesta, GS1:n yritystunnuksesta (GCP), sarjanumerosta ja tarkistenumerosta. Sovellustunnuksena (AI) on aina ”00”, jota käytetään silloin kun SSCC-tunnus tulostetaan viivakoodina tai siirretään data-integraatiossa. Laajennustunnus on yhden numeron pituinen ja se on asiakasyrityksen vapaasti valittavissa. Yritystunnus voi olla 6-, 7- tai 9-numeroinen ja yhdessä lähetyksen yksilöivän sarjanumeron kanssa ne muodostavat aina 16-numeroisen luvun. Näin muodostetusta 17-numeroisesta tunnuksesta, poissulkien alun ”00”, lasketaan loppuun liitettävä tarkistenumero. Lähetystunnus tulostetaan osoitekortille aina Code 128 -viivakoodina, jolloin tunnukseen alkuun lisätään vielä Function Code 1 (FNC1, arvo 102). Esimerkkinä SSCC-lähetystunnus ”**00**9**643004871**00000005**5**”. (GS1a; GS1b; Bar Code Graphics 2013.)

3.4 Code 128 -viivakoodi (Code 128)

Yksiulotteisissa (1D) viivakoodeissa informaatio on koodattu leveydeltään vaihtelevien, tummien ja vaaleiden juovien muodostamaan yhdistelmään. Kun viivakoodi luetaan lukulaitteella, niin lukulaite mittaa juovien leveydet ja viivojen kombinaatiot, tunnistaa käytetyn standardin, tarkistaa mahdollisen tarkistenumeron ja palauttaa taustajärjestelmään luetun arvon. Viivakooditekniikka perustuu globaalisti standardoituun teknologiaan, jota käyttämällä pyritään saavuttamaan prosessihyötyä tietojen tallennuksen nopeuden, sen oikeellisuuden, luennan helppouden ja teknologian edullisuuden kautta. (Logistiikan Maailma 2015; Sakki 2014, 16 - 17.)

Kirjassaan Sakki (2014, 17) pohtii, että viivakooditekniikkaa pidetään usein tietojenkäsittelyn yleisenä ratkaisuna kaikkiin ongelmiin, vaikka koodit ja lukijalaitteet eivät irrallisina asioina ratkaise mitään, vaan viisaus on tapauskohtaisesti ratkaistuissa taustajärjestelmissä. Lisäksi hän toteaa, että lukijalaite vain korvaa käsin tehtävän tallennuksen ja sen avulla pitkänkin viivakoodin tunnistaminen tapahtuu nopeasti ja virheettömästi.

3.4.1 Code 128:n rakenne

Code 128 on lineaarinen, yksiulotteinen viivakoodi, jonka avulla voidaan esittää numeerisia arvoja, kirjaimia ja erikoismerkkejä tiiviissä muodossa. Koodi sisältää merkkikohtaisen pariteettitarkistuksen ja viivakoodiin lisättävän tarkistemerkin. Code 128:lla voidaan esittää 128 erilaista ASCII-arvoa ja sen tilantarve on erilaisista yksiulotteisista viivakoodeista pie-

nin, silloin kun esitettävä arvo on pidempi kuin kuusi merkkiä. Code 128 on määritelty ISO:n standardissa (ISO 15417). (Adams Communications 2013; BarCodeIsland.com, 2006.)

Code 128:ssa jokainen merkki muodostuu 11 mustasta tai valkoisesta moduulista, poikkeuksena loppumerkki (Stop) joka muodostuu 13 moduulista. Moduuleista muodostettu viivakoodi rakentuu kolmesta mustan ja kolmesta valkoisen moduulin yhdistelmästä, jotka toistuvat mustasta yhdistelmästä alkaen vuorotellen. Jokainen yhdistelmä sisältää 1- 4 moduulia. Yhden moduulin leveys (x-arvo tai x-dimensions) on usein 0,5 mm, jolloin yhtä merkkiä vastaavan yhdistelmän pituus on 5,5 mm. Moduulin korkeus on vähintään 15 % koko viivakoodin pituudesta. (Adams Communications 2013.)

Code 128:ssa käytettävä merkistö muodostuu kolmesta merkkitaulukosta (Adams Communications 1995):

- Code A -taulukko sisältää merkit (0-9, A-Z, toimintokoodeja), erikoismerkkejä ja FNC 1-4.
- Code B -taulukko sisältää merkit (0-9, A-Z, a-z), erikoismerkkejä ja FNC 1-4.
- Code C -taulukko sisältää numerot 00-99 ja FNC1.

Code C -merkkitaulukkoa käytetään ainoastaan numeeristen arvojen esittämiseen, jolloin arvot esitetään numeropareina ja viivakoodin tilantarve vähenee lähes puoleen. Kuvassa 1 on esitelty Code 128:n rakenne ja sen komponentit.

Kuva 1. Code 128:n rakenne (Adams Communications 2013)

Rakenne muodostuu kuudesta komponentista:

- Aloittava hiljainen alue (Quiet zone), joka jätetään tyhjäksi ja jonka pituus on vähintään 10 kertaa X-arvo.
- Aloitusmerkki (Start) ilmoittaa koodauksessa käytettävän merkistön (Code A, B tai C).
- Koodattavat merkit (Data) sisältävät viivakoodin sisällön varsinaisen arvon.
- Tarkistemerkki (Check character) sisältää kaikista viivakoodin sisältämistä arvoista lasketun tarkistemerkin.
- Lopetusmerkki (Stop) päättää viivakoodin.

- Lopettava alue (Quiet zone), joka jätetään tyhjäksi ja jonka pituus on vähintään 10 kertaa X-arvo.

3.4.2 Code 128:n merkistö ja sen vaihto

Esitettävästä arvosta riippuen, viivakoodin aloitusmerkistöksi voidaan valita tarkoitukseen sopivin, ellei sitä ole sovelluskohtaisesti erikseen määritelty. Jos viivakoodiksi muutettava arvo sisältää numeroita ja kirjaimia, niin merkistöksi voidaan valita Code B. Jos arvo sisältää ainoastaan isoja kirjaimia, niin merkistöksi voidaan valita Code A.

Kun arvo on kokonaan numeerinen, niin merkistöksi voidaan valita Code C, jolloin numerot koodataan tilaa säästäen numeropareina. Jos viivakoodiksi muutettava luku on pariton, niin silloin yhtä numeroa varten merkistöksi on vaihdettava Code A tai Code B. Tämä vaihto voidaan tehdä aluksi, jolloin voidaan aloittaa B-merkistöllä ja ensimmäisen merkin jälkeen vaihtaa C-merkistöön, tai aloittaa C-merkistöllä ja vaihtaa B-merkistöön ennen viimeistä numeroa. Yhden viivakoodin sisällä voidaan tehdä useita vaihtoja merkistöstä toiseen mutta jokainen vaihtomerkki lisää tulevan viivakoodin pituutta yhden moduuliyhdistelmän verran (11 x X-arvo). Voidaan myös käyttää vain yhtä merkkiä toisesta merkistöstä käyttämällä Shift-komentoa mutta ainoastaan A- ja B-merkistöjen välillä. Usein tavoitteena on muodostaa mahdollisimman lyhyt viivakoodi, jonka pituutta voidaan optimoida valitsemalla tarkoitukseen sopiva merkistö ja sijoittamalla merkistöjen vaihdot oikeisiin paikkoihin. (Adams Communications 2013; BarCodeIsland.com, 2006.)

3.4.3 Code 128:n tarkistemerkin laskenta

Ennen kuin viivakoodin arvo voidaan tulostaa, niin sille on laskettava tarkistemerkki, joka tulee osaksi viivakoodia, vaikka sitä ei kuitenkaan tulosteta näkyviin (ja jota viivakoodinlukija ei myöskään palauta, vaikka tarkastaakin luetun arvon oikeellisuuden). Tarkistemerkin laskenta perustuu modulo 103 -kaavaan ja merkkiarvojen painotettuihin summiin.

Tarkistemerkin laskenta tapahtuu merkkitaulukosta saatuihin arvoihin (Adams Communications 1995):

- Aloitusmerkin arvosta (103, 104 tai 105) saadaan summaan alkuluku, jonka kerroin on yksi.
- Ensimmäiselle koodattavalle merkille haetaan taulukosta arvo (tai käytetään numeroparin arvoa), jonka kerroin on myös yksi.
- Haetaan kaikille lopuille koodattaville merkeille (tai numeropareille) arvo taulukosta ja kasvatetaan kerrointa aina yhdellä.

Taulukossa 2 on esimerkki tarkistemerkin laskennasta, jossa viivakoodattavaksi arvoksi on annettu ”HI345678”. Koodaus alkaa alkumerkillä ”START A” (arvo 103), jonka jälkeen

koodataan H-kirjain (40) ja I-kirjain (41). Koska seuraavat kuusi merkkiä ovat numeroita, niin ne koodata kolmena C-merkistön numeroparina. Vaihto C-merkistöön tehdään toiminnolla ”Code C” (99), jonka jälkeen koodataan numeroparit (34), (56) ja (78).

Taulukko 2. Esimerkki tarkistemerkin laskennasta (BarCodeIsland.com, 2006)

Barcode	START-A	H	I	CODE-C	34	56	78
Character Value	103	40	41	99	34	56	78
Character Position	-	1	2	3	4	5	6
Calculation	103	40 * 1	41 * 2	99 * 3	34 * 4	56 * 5	78 * 6
Weighted Sum	103	40	82	297	136	280	468

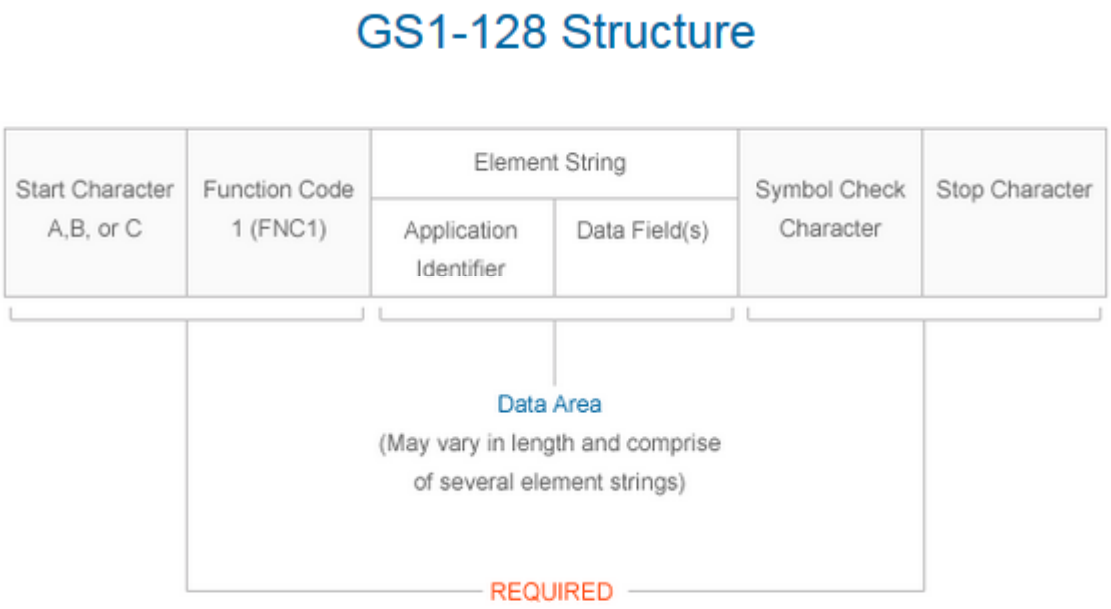
Tarkistenumeron laskenta aloitetaan laskemalla yhteen merkkiarvojen ja kertoimien tulot. Yhteenlaskettu summa jaetaan 103:lla, jonka jakojäännöksestä tulee tarkistemerkki ja jonka arvo liitetään koodisarjan perään. Kaavana tämä voidaan esittää käyttäen kongruenssia: $1406 \equiv 67 \pmod{103}$ eli kun 1406 jaetaan 103:lla, niin jakojäännös on 67.

3.4.4 Code 128:n tulostus

Kun edellä mainittu merkkien arvoista koostettu koodisarja on valmis, niin loppuun liitetään vielä 13 moduulin pituinen lopetusmerkin koodi (106). Ennen tulostusta koodisarjan arvot muutetaan vielä ASCII-arvoiksi lisäämällä 32 jokaiseen koodisarjan arvoon. Tämän jälkeen koodisarja on valmis tulostettavaksi erillisillä ohjelmilla tai käyttämällä jotain lukuisista Code 128 -fonteista.

3.4.5 GS1:n SSCC-tunnus GS1-128 -viivakoodina (GS1-128)

GS1:n kehittämä GS1-128 -standardi ei määrittele viivakoodin tuottamista samalla tavalla kuin Code 128 -standardi, vaan se määrittelee ainoastaan viivakoodissa käytettävät komponentit ja muilta osin se hyödyntää täysin Code 128 -standardia. Kuvassa 2 esitetään GS1-128:n rakenne, jossa Code 128:sta poikkeavana on ainoastaan pakollinen Function Code 1 (FNC1) -toiminto. (Bar Code Graphics 2013.)

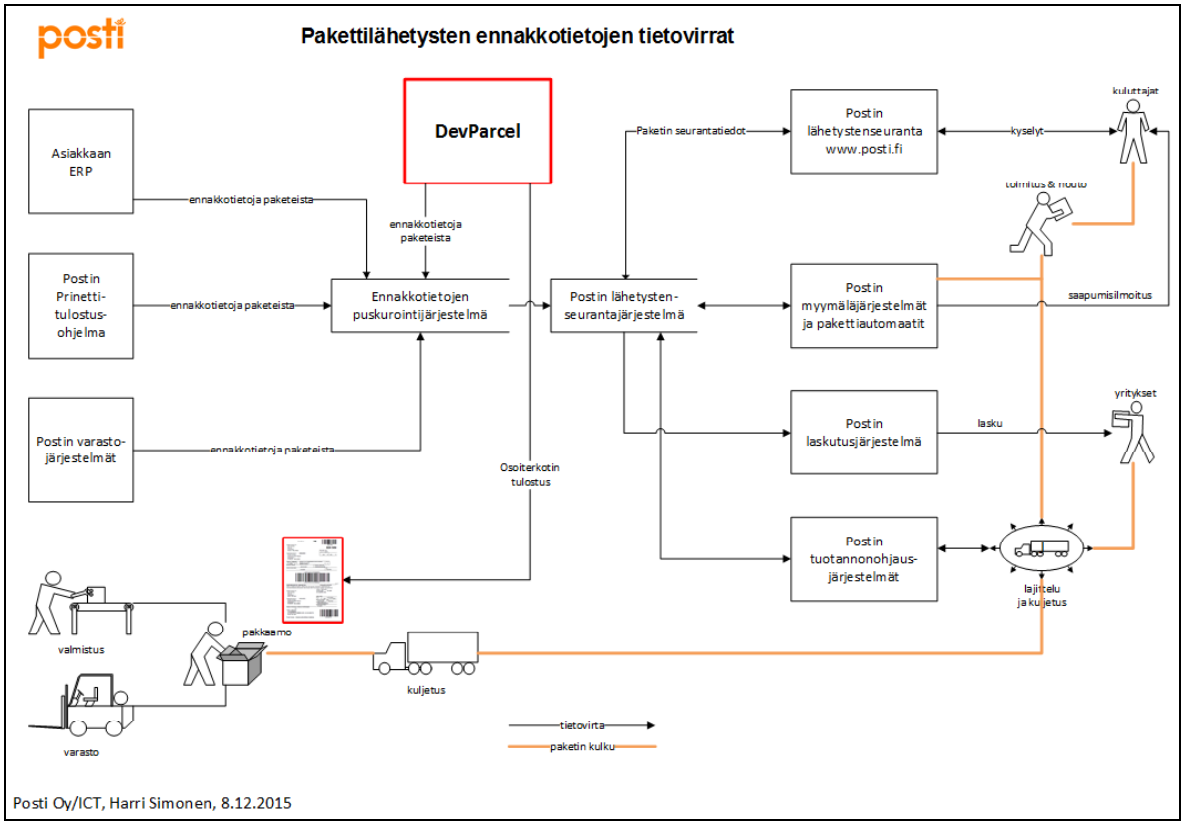


Kuva 2. GS1-128:n rakenne (Bar Code Graphics 2013)

Luvussa 3.3 on selvitetty miten luodaan GS1:n SSCC-tunnus, joka sijoittuu GS1-128:n rakenteessa kohtaan ”Element String” (Kuva 2). Koska SSCC-tunnus on kokonaan numeerinen ja parillinen, niin Code C -merkistö (arvo 105) sijoitetaan kohtaan ”Start Character”. Kohtaan ”Function Code” sijoitettaan aina FNC1 (arvo 102). Muilta osin GS1-128:n muodostaminen tapahtuu edellä mainituissa luvuissa 3.4.3 ja 3.4.4 mukaisesti.

4 DevParcel-kehitystyökalusovelluksen toteuttaminen

Produktin tavoitteena oli toteuttaa Postille DevParcel-sovellus, jolla voidaan tuottaa testiaineistoja Postin liiketoiminnan kehitysprojekteissa suunniteltujen ja toteutettujen muutosten testaamiseksi. Tuotettavan testiaineiston on tarkoitus simuloida asiakkaan toimintaa heidän lähettämien ennakkotietojen ja tulostamien osoitekorttien osalta. Kuviossa 2 on havainnollistettu DevParcel-sovelluksen rooli Postin toimintaympäristössä, jossa näkyvät sovelluksen tuottamien aineistojen käyttötarkoitus ja niiden sulautuminen tuotantoprosessiin sekä integroituihin tietojärjestelmiin. DevParcel-sovellusta on tarkoitus käyttää kuitenkin vain testiympäristössä.



Kuvio 2. Lähetysten ennakkotietojen tietovirrat Postissa

4.1 Kehystystyökalun toimintaympäristö Postissa

Postin lähetystenseurantajärjestelmä perustuu paketti- ja kirjelähetyksissä käytettävien lähetystunnusten tunnistamiseen ja niille tehtyihin seurantatapahtumiin, joita tehdään kuljetus- ja jakeluprosessien aikana. Lähetysten asiakaslaskutus perustuu tehtyihin seurantatapahtumiin sekä lähetyksistä vastaanotettuihin ennakkotietoihin. Lähetyksen päälle kiinnitetyssä osoitekorteissa oleva lähetystunnus yhdistää fyysisen lähetyksen vastaanotettuun ennakkotietoon. Koska lähetykselle tehdyt seurantatapahtumat kohdistuvat osoitekortissa

olevaan lähetystunnukseen, niin lähettäjien on muodostettava se ainutkertaisena. Lähetystunnuksen muodostamista ohjaavat erilaiset standardit, jotka riippuvat käytetystä kuljetuspalvelusta. Edellytyksenä onnistuneelle lähetystenseurannalle on myös se, että osoitekortissa ilmoitettu lähetystunnus on tulostettu myös viivakoodina.

4.2 Lähtötilanne

Postissa liiketoimintaprosessit ovat jatkuvassa kehityksessä, jolloin muutoksia toteuttavissa IT-projekteissa muutetaan usein myös olemassa olevien tietojärjestelmien rajapintoja. Uusia toiminnollisuuksia testattaessa, myös muuttuneet rajapinnat on testattava mahdollisimman realistisella testiaineistolla, joka taas muodostaa lähtötiedot laajemmalle järjestelmäintegraatiokokonaisuudelle.

Perinteisesti projektissa määritetyn testaussuunnitelman mukaiset testitapaukset on valmistettu manuaalisesti vaihtelevin adhoc-menetelmin, jolloin osoitekortteja ja ennakkotietoja on tuotettu usein minimivaatimuksin. Testiaineistojen tuottaminen on ollut hidasta ja niiden laatu vaihtelevaa. Vuoden 2015 loppuun asti testaamisessa on pystytty käyttämään tuotannosta kopioituja ennakkotietoja mutta uudistuneen EU tietoturvadirektiivin vuoksi, niiden käyttäminen testi- ja kehitysympäristöissä on nykyään ehdottomasti kielletty ennakkotietojen sisältäminen henkilöihin liittyvien tunnistetietojen vuoksi.

4.3 DevParcel-sovelluksen toteutussuunnitelma

Postissa päätettiin, että DevParcel-sovellus voidaan toteuttaa kehitysprojektina, joka samalla toteuttaa myös opinnäytetyöni produktin. Suunnittelun lähtökohdaksi asetettiin, että toteutettava sovellus pystyy tuottamaan asiakasintegraation testaamisessa käytettävän testiaineiston eli ennakkotiedot Lähetystenseurantajärjestelmään ja osoitekortit tuotantoprosessien testaamiseen. Koska Postissa oli tiedossa, että meneillään olevat kehityshankkeet tulevat muuttamaan olemassa olevia järjestelmärajapintoja, niin sovellus piti suunnitella niin, että siihen olisi helppo toteuttaa rajapintoihin kohdistuvat muutokset tai kokonaan uudet rajapinnat. Suunnittelu perustuu nykyisen rajapinnan vaatimuksiin ja tuleviin muutokset, siltä osin kun ne olivat tiedossa suunnitteluvaiheen alussa. Sovelluksen nimeksi annettiin DevParcel, koska sovellusta on tarkoitus käyttää pääosin Postin Pakettipalvelut -liiketoimintaan liittyvien kehitysprojektien tuotosten testaamiseen.

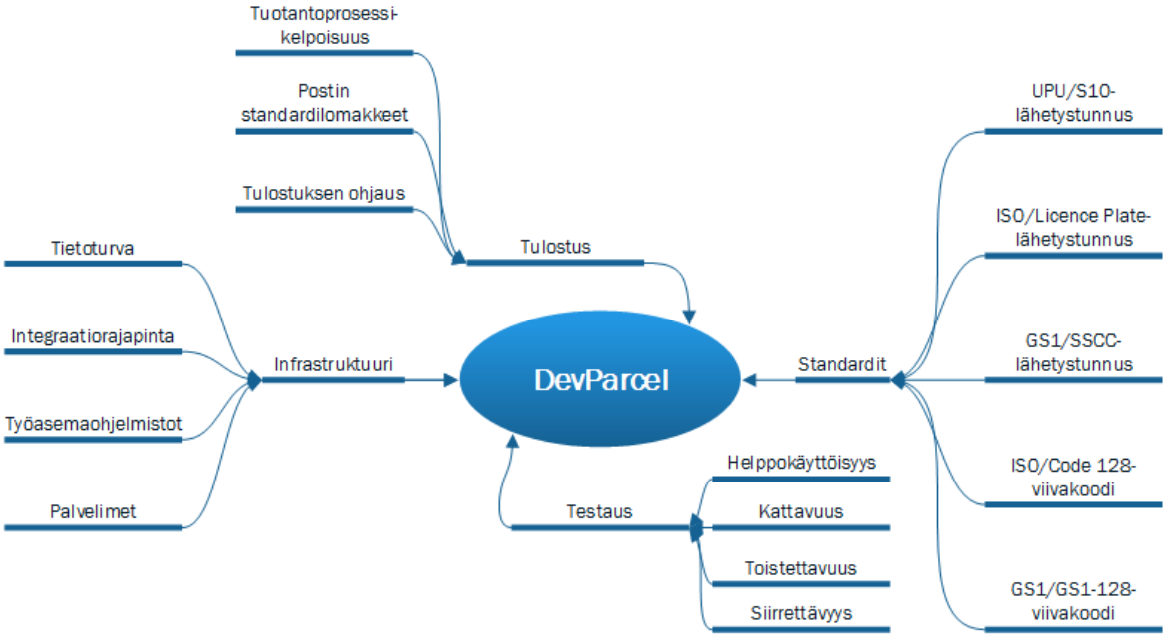
4.4 DevParcel-toteutusprojektin tavoitteet ja sovellukselle asetetut vaatimukset

Projektin liiketoiminnallisena tavoitteena on tuottaa työkalu, joilla testaajat pystyvät muodostamaan totuudenmukaista ja oikeanmuotoista testiaineistoa järjestelmäkehityksen yh-

teydessä, koskien Postin Paketti- ja Kirjepalveluiden liiketoimintoja ja Postin Lähetysten-seurantajärjestelmää. Työkalun käytöstä ei odoteta tulevan projekteille lisäkustannuksia vaan säästöjä, jotka syntyvät tehokkaamman ja nopeamman testausprosessin ansiosta, jolloin testaamiseen käytettyjen sisäisten ja ulkoisten resurssien tarve vähenee.

Projektin toiminnallisena tavoitteena oli nopeuttaa järjestelmäkehitysprosessia ja parantaa tuotantojärjestelmien laatua. Työkalulla testaajat pystyvät muodostamaan totuudenmukaista ja oikeanmuotoista dataa testattavaan järjestelmään sekä tuottamaan viivakoodattuja osoitekortteja lähetysten fyysisen käsittelyprosessin testaamiseen. Testiaineistojen avulla voidaan simuloida asiakkaan lähetystoimintaa eri järjestelmätoiminnoissa ja käsittelyprosesseissa, jotka esiteltiin kuviossa 2. Työkalulla on pystyttävä tuottamaan Postin Lähetystenseurantajärjestelmän määritysten mukaisia integraatio-tiedostoja, jotka vastaavat asiakkaiden lähettämiä ennakkotietoja.

Projektin suora asiakas on Postin ICT-yksikkö, joka testaa tietojärjestelmiin toteutetut muutokset yhdessä liiketoiminnan asiantuntijoiden kanssa liiketoimintavaatimusten mukaisesti. Sovellukseen liittyviä teknisiä vaatimuksia on esitelty kuviossa 3, jossa vaatimukset on jaettu neljään ryhmään. On otettava huomioon toteutukseen liittyvät standardit ja Postin infrastruktuurin asettamat mahdollisuudet ja rajoitteet. Lisäksi on huomioitava sovelluksen käytettävyys testiaineistoja muodostettaessa, niin että testaussuunnitelman mukaisten testitapausten syöttö ja osoitekorttien tulostukset onnistuvat laadukkaasti ja tehokkaasti.



Kuvio 3. DevParcel-sovellukselle asetettuja vaatimuksia

4.5 Projektisuunnitelma

DevParcel-kehitystyökalu toteutettiin marraskuun 2015 aikana Postissa kehitysprojektina, josta Postin projektimallin mukainen projektisuunnitelma on liitteessä 1.

Projektisuunnitelman päävaiheet:

1. Projektin tausta ja yhteydet strategioihin
2. Projektin tavoitteet
3. Projektin laajuus
4. Projektin tuotokset
5. Liiketoimintahyödyt
6. Projektin kustannukset
7. Projektin aikataulu ja vaiheet
8. Projektiorganisaatio
9. Organisaation muutoksenhallinta
10. Projektin riskianalyysi
11. Riippuvuudet
12. Laadunvarmistus
13. Ympäristövaikutusten arviointi
14. Projektin hallintokäytännöt
15. Projektin päättäminen

4.6 Projektin tulokset

Opinnäytetyön produktin tavoite onnistui eli kehitysprojektissa tuotettiin DevParcel-sovellus, joka toteuttaa sille asetetut vaatimukset ja toimeksiantajan tarpeen. Sovellus otettiin Postissa käyttöön joulukuussa 2015 ja sitä käyttäviltä testaajilta on saatu ohjelmasta hyvää palautetta.

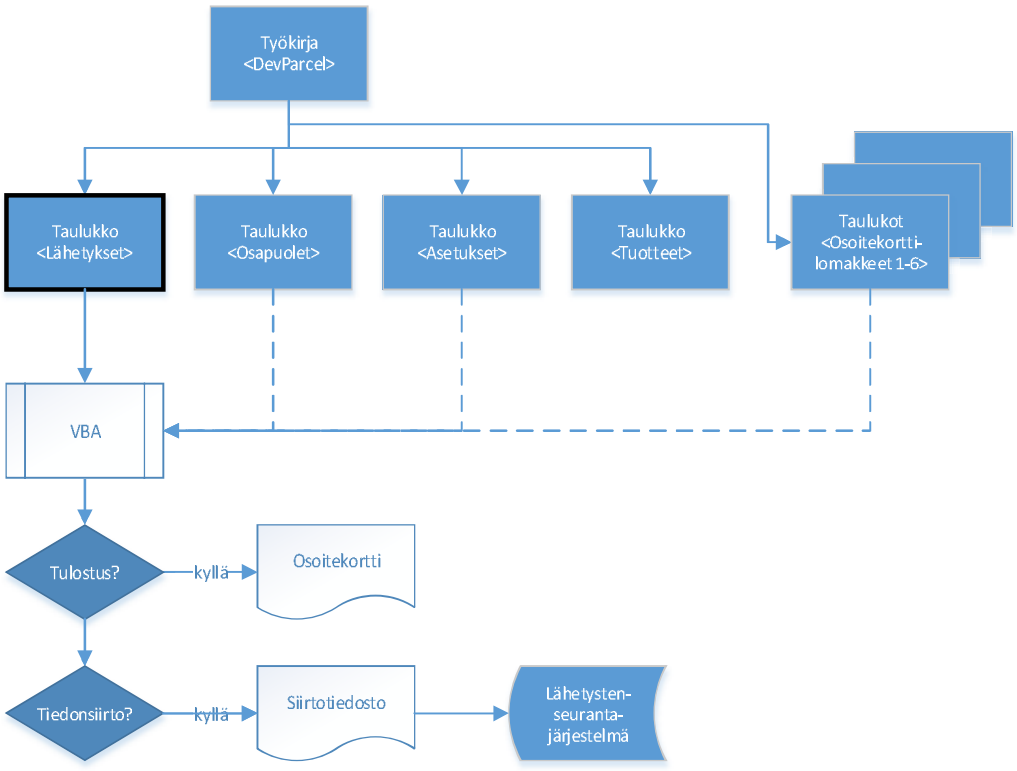
Sovelluksella saa syötettyä yhden lähetyksen perustiedot muutamassa sekunnissa ja samoin tulostus sekä tiedonsiirto tapahtuvat myös noin sekunnissa. Kun yhden lähetyksen tiedot kopioidaan Excel-taulukossa tuhannelle riville, niin sovellus tulostaa noin kolme osoitekorttia sekunnissa, kun oletuskirjoittimeksi on asetettu osoitekortit tiedostoiksi tulostava ”PDF-XChange Printer 2012”. Kun sama testi suoritetaan ilman osoitekorttien tulostusta, niin tuhat lähetystä käsitellään noin kuudessa sekunnissa, jolloin lähetyksille generoidaan vain yksilölliset lähetystunnukset ja luodaan tuhat lähetystietoa sisältävä lähetystiedosto. Tiedonsiirron suoriutumiseen testatuilla määrillä eli tiedoston koolla ei ole huomattavaa vaikutusta (1 lähetys = 2 kt, 1000 lähetystä = 1438 kt). Koska käsitellyt testitapaukset jäävät talteen Excel-taulukon riveille, niin testi on helposti toistettavissa mahdollisten kohdejärjestelmään tehtyjen ohjelmakorjausten jälkeen kokonaan tai tarvittavin osin.

DevParcel-sovelluksen (DevParcel_p1.xlsm) koko on noin 400 kt, joka sisältää talletettuja kuljetuspalveluiden tuotetietoja, osapuolitietoja, asetuksia, kaksi käyttäjälomaketta sekä kuusi piilotettua taulukkomuodossa olevaa tulostuslomaketta. Sovellus on kokonsa puolesta helppo jakaa sähköpostissa ja se sisältää vain yhden työkirjan. Varsinaista asennusta ei tarvitse tehdä mutta työkirjan käynnistyksen yhteydessä tulee varoitus työkirjan sisältämistä makroista, jotka tulee hyväksyä. Muodostetut lähetystiedostot arkistoidaan omaan hakemistoon, joka muodostetaan tarvittaessa samaan hakemistoon josta työkirja avattiin.

4.7 DevParcel-sovelluksen rakenne

Sovelluksen rakenne on esitetty kuviossa 4 ja se perustuu Excel-työkirjaan (DevParcel_p1.xlsm), jossa käyttöliittymänä toimii neljä erilaista taulukkoa: Lähetykset (lähetystietojen syöttö), Osapuolet (lähettäjä- ja vastaanottajatietojen ylläpito, Tuotteet (Palvelujen tuotetietojen ylläpito) ja Asetukset (sovelluksen asetukset). Lähetystietojen ja asetusten syöttämiseksi toteutettiin myös tietojensyöttämistä helpottavat käyttäjälomakkeet.

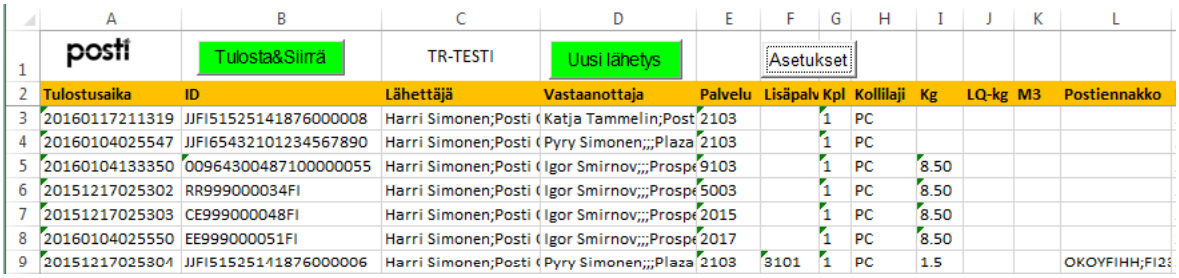
Lisäksi työkirja sisältää kuusi käyttäjiltä piilotettua taulukkoa, jotka toimivat erilaisten tulostettavien osoitekorttilomakkeiden pohjina.



Kuvio 4. DevParcel-sovelluksen rakenne

4.8 DevParcel-sovelluksen toteutus

Sovellus on toteutettu kuviossa 4 esitetyn rakenteen mukaisesti, jossa keskeisenä käyttöliittymänä toimii Lähetykset-taulukko, joka sisältää sarakkeita lähetyksen tiedoille ja jossa jokainen taulukon rivi muodostaa yhden lähetyksen tiedot (kuva 3).

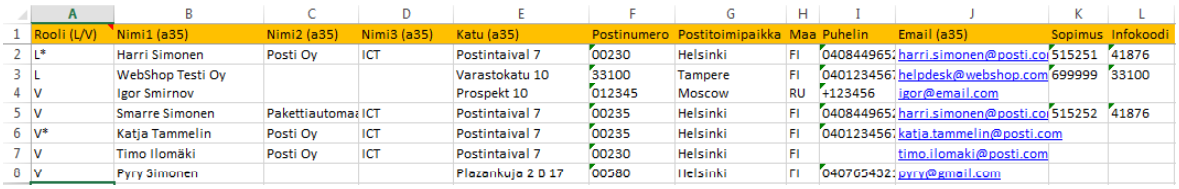


	A	B	C	D	E	F	G	H	I	J	K	L
1	posti	Tulosta&Siirrä	TR-TESTI	Uusi lähetys		Asetukset						
2	Tulostusaika	ID	Lähettäjä	Vastaanottaja	Palvelu	Lisäpalv	Kpl	Kollilaji	Kg	LQ-kg	M3	Postiennakko
3	20160117211319	JJFI51525141876000008	Harri Simonen;Posti (	Katja Tammelin;Post	2103		1	PC				
4	20160104025547	JJFI65432101234567890	Harri Simonen;Posti (	Pyry Simonen;;;Plaza	2103		1	PC				
5	20160104133350	00964300487100000055	Harri Simonen;Posti (	Igor Smirnov;;;Prospe	9103		1	PC	8.50			
6	20151217025302	RR999000034FI	Harri Simonen;Posti (	Igor Smirnov;;;Prospe	5003		1	PC	8.50			
7	20151217025303	CE999000048FI	Harri Simonen;Posti (	Igor Smirnov;;;Prospe	2015		1	PC	8.50			
8	20160104025550	EE999000051FI	Harri Simonen;Posti (	Igor Smirnov;;;Prospe	2017		1	PC	8.50			
9	20151217025304	JJFI51525141876000006	Harri Simonen;Posti (	Pyry Simonen;;;Plaza	2103	3101	1	PC	1.5			OKOYFIHH;FI2

Kuva 3. Lähetykset-taulukko

Lähetystiedot voidaan syöttää suoraan lähetystaulukkoon, jossa niitä voidaan myös muokata ennen tulostusta ja tiedonsiirtoa. Tietojensyöttöön ei haluttu liittää erityisiä virhetarkistuksia, koska testatessa on tarpeen tuottaa myös viallista materiaalia. Kuitenkin, koska tavoitteena oli toteuttaa helppo käyttöliittymä testaajille, niin työkirjaan toteutettiin myös perustietoja sisältäviä taulukoita (osapuolet, tuotteet ja asetukset), jotka toimivat apuna uusia lähetystietoja syötettäessä kuvassa 3 näkyvällä Uusi lähetys-toiminnolla.

Osapuolet-taulukko sisältää testauksessa käytettävien lähettäjien ja vastaanottajien tiedot (kuva 4). Koska Lähetys-taulukossa osapuolen tiedot on tiivistetty erotinmerkeillä yhteen soluun, niin osapuolitietojen syöttäminen ennakolta on suositeltavaa. Osapuolet-taulukkoon tallennettuja tietoja on myös helppo jakaa useiden testaajien kesken, kun tuotantotestauksessa käytetään usein ennalta määrättyä vastaanottajajoukkoa lähetyksille.



	A	B	C	D	E	F	G	H	I	J	K	L
1	Rooli (L/V)	Nimi1 (a35)	Nimi2 (a35)	Nimi3 (a35)	Katu (a35)	Postinumero	Postitoimipaikka	Maa	Puhelin	Email (a35)	Sopimus	Infokoodi
2	L*	Harri Simonen	Posti Oy	ICT	Postintaival 7	00230	Helsinki	FI	040844965;	harri.simonen@posti.co	515251	41876
3	L	WebShop Testi Oy			Varastokatu 10	33100	Tampere	FI	040123456;	helpdesk@webshop.com	699999	33100
4	V	Igor Smirnov			Prospekt 10	012345	Moscow	RU	+123456	igor@email.com		
5	V	Smarre Simonen	Pakettiautomaa	ICT	Postintaival 7	00235	Helsinki	FI	040844965;	harri.simonen@posti.co	515252	41876
6	V*	Katja Tammelin	Posti Oy	ICT	Postintaival 7	00235	Helsinki	FI	040123456;	katja.tammelin@posti.com		
7	V	Timo Ilomäki	Posti Oy	ICT	Postintaival 7	00230	Helsinki	FI		timo.ilomaki@posti.com		
0	V	Pyry Simonen			Plazankuja 2 D 17	00580	Helsinki	FI	040765432;	pyry@gmail.com		

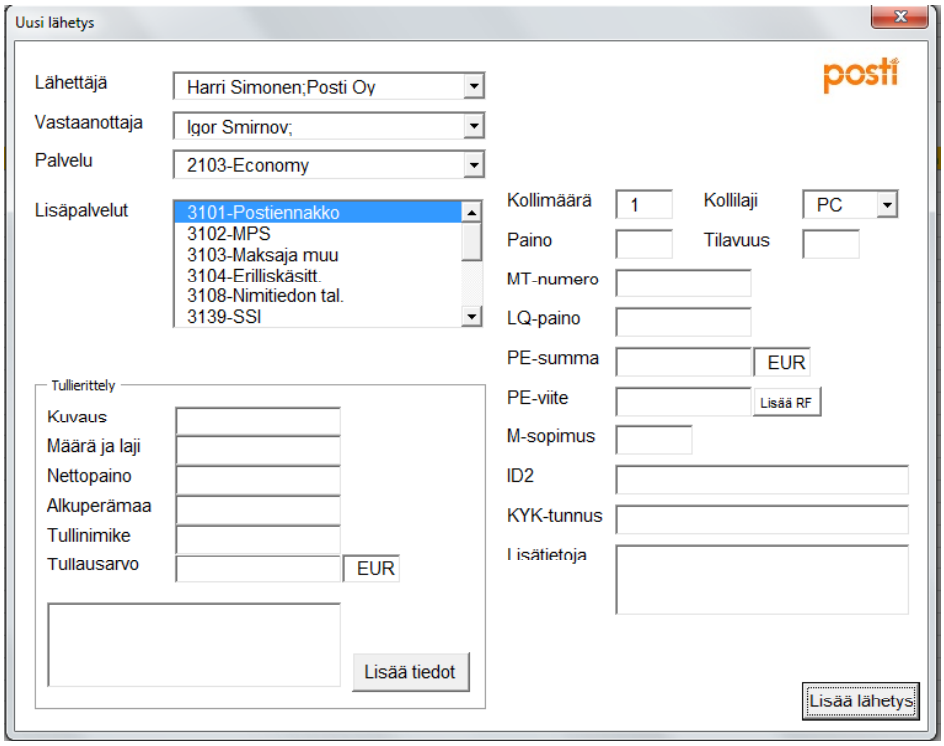
Kuva 4. Osapuolet-taulukko

Kuvassa 5 on esitetty perustietoja sisältävä Tuotteet-taulukko, jossa matriisissa luetellaan Postin tuotteet ja lisäpalvelut sekä niiden sallitut yhdistelmät. Lisäksi taulukossa kerrotaan tuotekohtaisia ohjaustietoja, jotka liittyvät lähetystunnusten muodostamiseen ja osoitekorttien tulostamiseen. Uusien tuotteiden ja lisäpalveluiden sekä näiden sallittujen kombinaatioiden lisäämisen taulukkoon on yksinkertaista ja nopeaa.

	A	B	C	G	H	I	J	K	L	M	N	O	P
1	Tuote/LP	Tuotekoodi	TuotenimiFI	Lyhytkoodi	Viivakoodi	Lisäpalvelut=>	3101	3102	3103	3104	3105	3106	3107
2	T	2015	Priority		S10P								
3	T	2017	EMS		S10E								
4	T	2020	Consumer Parcel	16	S10C		X						
5	T	2101	Express Morning	9	LP			X	X	X		X	
6	T	2102	Express Business	14	LP		X	X	X	X			
7	T*	2103	Economy	16	LP		X	X	X	X			
8	T	2104	Express Flex	21	LP		X	X	X	X			
9	T	2105	Express Point	00/16	LP		X	X			X		
10	T	2106	SmartPost	16	LP		X	X					
11	T	2108	Asiakaspalautus	14	LP					X			
12	T	2116	VAK/ADR		LP		X	X	X	X			
13	T	2124	Express City	00	LP			X	X	X			
14	T	2127	Termo		LP		X	X	X				

Kuva 5. Tuotteet-taulukko

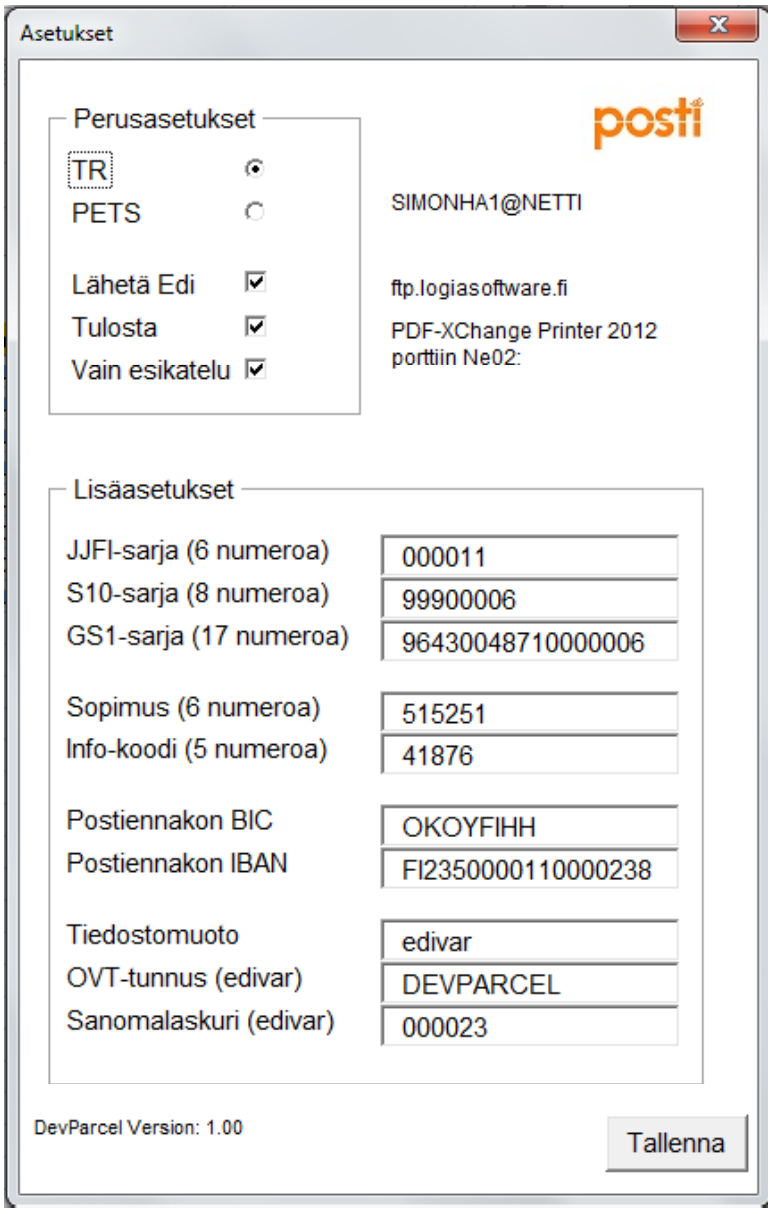
Lähetykset-taulukkoon toteutettu Uusi Lähetys-toiminto (kuva 3) hyödyntää edellä mainittuja perustietoja uutta lähetystä muodostettaessa. Yksinkertainen testitapaus voidaan tehdä valitsemalla vain oikeat arvot avautuvasta käyttäjälomakkeesta (kuva 6), jolloin valintojen (ja muiden annettujen tietojen) perusteella saadaan muodostettua uusi lähetysrivi Lähetykset-taulukkoon käyttämällä lomakkeen Lisää lähetys -toimintoa.



Kuva 6. Uusi lähetys -lomake

Koska eri tuotteet perustuvat erilaisiin standardeihin, joissa lähetystunnukset muodostuvat omilla säännöillään, niin sovelluksen perustietoina toteutettiin myös Asetukset-taulukko.

Kuitenkin paremman käytettävyyden varmistamiseksi, varsinainen Asetukset-taulukko piilotettiin ja Lähetykset-taulukkoon lisättiin Asetukset-toiminto, joka avaa Asetukset-lomakkeen lähetystunnussarjojen hallinnoimiseksi. Kuvassa 7 on esitetty DevParcel-sovelluksen asetukset, jotka yleensä tallennetaan käyttäjä- tai testaajakohtaisesti.

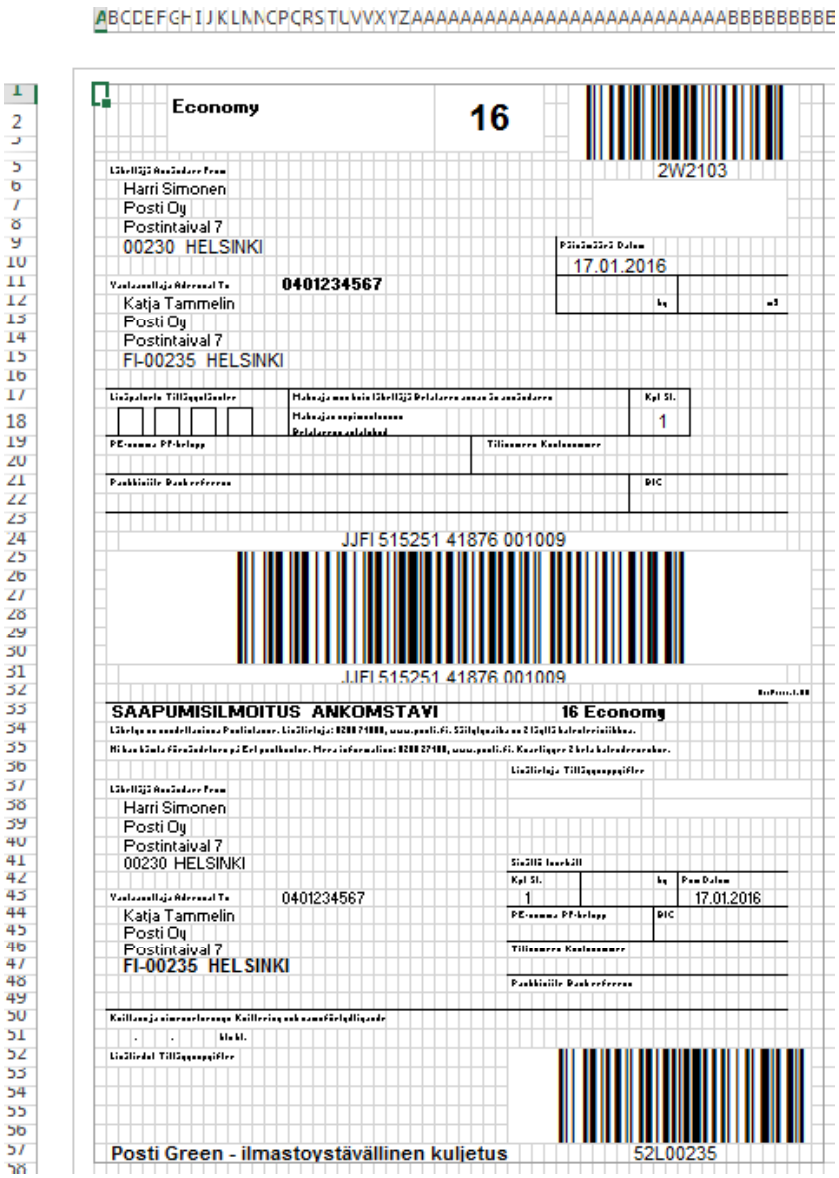


Kuva 7. Asetukset-lomake

Sovelluksen käyttäjän hallinnoimien taulukoiden lisäksi, sovellukseen toteutettiin kuusi käyttäjältä piilotettua taulukkoa, joita sovellus käyttää erilaisten osoitekorttitulosteiden lomakepohjina. Sovellukseen on toteutettu seuraavat lomakepohjat:

- LabelA5_perus, Postin kotimaan tuotteet Licence Plate -lähetystunnuksilla (kuva 8).
- LabelA5_EMS, UPU/EMS-lähetyksen osoitekortti S10-lähetystunnuksella.
- LabelA5_RM, UPU/Registered Letter -osoitekortti S10-tunnuksella.
- LabelA5_PRIO, UPU/Priority-lähetyksen osoitekortti S10-lähetystunnuksella.
- LabelA5_cons, Postin Consumer Parcel -osoitekortti S10-lähetystunnuksella.

- Label_GS1, GS1:n Kollilappu SSCC-lähetystunnuksella.



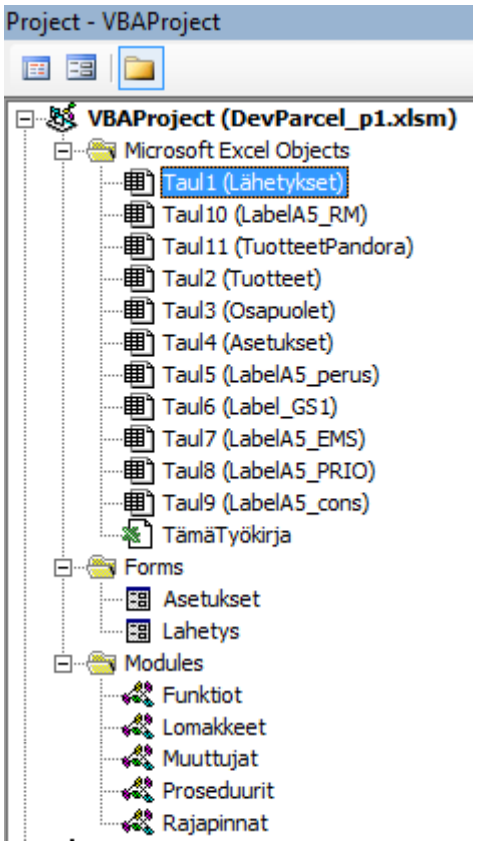
Kuva 8. Esimerkki tulostukseen käytettävästä lomakepohjasta: LabelA5_perus-taulukko

Kuvassa 8 on esitetty malliksi yksi taulukkona toteutettu osoitekorttipohja, jossa näkyy aina edellisen tulostuksen tiedot. Sovellus siirtää Lähetys-taulukon lähetysriveiltä tiedot lomakkeen soluihin, jotka on muotoiltu standardin vaatimusten mukaisesti. Lomakkeet on toteutettu kokonaan Excelin perustoiminnoilla, lukuun ottamatta viivakoodeissa käytettävää JLCode128-fonttia, joka on asennettu valmiiksi kaikkiin Postin työasemiin.

Excel-työkirjan yhteyteen toteutettu VBA-ohjelmointi sitoo edellä esitetyt taulukot ja käyttäjälomakkeet yhteen ja muodostaa kokonaisuutena toimivan DevParcel-sovelluksen. Seuraavissa alakappaleissa esittelen toteutuksen keskeisimmät osat VBA-ohjelmoinnin näkökulmasta.

4.8.1 DevParcel-sovelluksen VBA-projektin rakenne

Toteutetun sovelluksen VBA-projekti jakautuu kolmeen komponenttiin, jotka on esitetty kuvassa 9. Ensimmäinen komponentti sisältää objekteina työkirjan taulukot, toinen käyttäjälomakkeet VBA-koodeineen ja kolmas työkirjaobjekteja käsittelevät ohjelmakoodit ja niihin liittyvät muuttujien määritykset.



Kuva 9. DevParcel-sovelluksen objektimalli

Forms-komponentissa Asetukset-lomakkeen yhteyteen lisätty VBA-koodi toteuttaa tietojen tallennuksen Asetukset-taulukkoon. Lähetys-lomakkeeseen on toteutettu perustietojen haku Osapuolet- ja Tuotteet-tauluista sekä uusien rivien lisäykset Lähetykset-tauluun.

Modules-komponentissa aliohjelmat on sijoitettu funktio- ja proseduuri-moduuleihin. Koska erilaiset osoitekortit käyttävät erilaisia taulukoita lomakepohjina, niihin liittyvät muuttujien käsittelyt ja tulostukset sijoitettiin selvyyden vuoksi omaan erilliseen Lomakkeet-moduuliin. Koko projektia koskevat yleiset muuttujat ja niiden vakioarvot sijoitettiin Muuttujat-moduuliin. Rajapinnat-moduuli, joista nyt toteutettiin Postin Lähetystenseurantajärjestel-

män edivar-rajapinta, muodostuu muuttujista koostuvista käyttäjän määrittelemistä rakenteellisista tietotyypeistä, jotka edustavat erilaisia positiopohjaisia tietueita.

4.8.2 Lähetystunnuksen muodostaminen

Sovellukseen toteutettiin kolmen erilaisen lähetystunnuksen muodostaminen, joista Postin kotimaan lähetyksissä käytettävä UPU/Licence Plate -tunnuksen muodostaminen ei vaatinut erityistä ohjelmointia.

Kappaleessa 3.2 on esitetty UPU/S10-lähetystunnuksen rakenne ja tarkistenumeron laskentaan liittyvä menetelmä. S10-tunnukselle tarkistenumeron laskenta toteutettiin VBA-koodissa aliohjelmana, joka saa kutsuvalta aliohjelmalta parametrina 8-numeroisen alkuarvon, jonka perusteella lasketaan tarkistenumero. Sovelluksen VBA-toteutus on esitetty kuvassa 10.

```
Public Function LaskeS10Tarkenne(ByVal S10koodi As String) As String
    Dim Kertoimet() As String
    Dim Summa
    Dim Tarkiste As Integer
    Dim i As Integer

    Kertoimet = Split("8;6;4;2;3;5;9;7", ";")
    For i = LBound(Kertoimet) To UBound(Kertoimet)
        Summa = Summa + (Val(Kertoimet(i)) * Val(Mid(S10koodi, i + 1, 1)))
    Next i
    Tarkiste = 11 - (Summa Mod 11)
    If Tarkiste = 10 Then Tarkiste = 0
    If Tarkiste = 11 Then Tarkiste = 5
    LaskeS10Tarkenne = Tarkiste
End Function
```

Kuva 10. S10-tarkistenumeron laskenta

LaskeS10Tarkenne-funktiota kutsunut sovellus liittää palautetun tarkistenumeron osaksi S10-lähetystunnusta.

Kappaleessa 3.3 on esitetty GS1:n SSCC-lähetystunnuksen muodostaminen ja tarkistenumeron laskentaan liittyvä menetelmä. SSCC-tunnukselle tarkistenumeron laskenta toteutettiin VBA-koodissa aliohjelmana, joka saa kutsuvalta aliohjelmalta parametrina 17-numeroisen alkuarvon, jonka perusteella lasketaan tarkistenumero. Sovelluksen VBA-toteutus on esitetty kuvassa 11.

```
Public Function LaskeSSCCTarkenne(ByVal SSCCkoodi As String) As String
    Dim Kertoimet(1 To 17) As Integer
    Dim Summa
    Dim Tarkiste As Integer, i As Integer

    For i = 1 To 17
        If (i Mod 2) = 0 Then
            Kertoimet(i) = 1
        Else
            Kertoimet(i) = 3
        End If
    Next i
    For i = Len(Trim(Left(SSCCkoodi, 17))) To 1 Step -1
        Summa = Summa + (Kertoimet(i) * Val(Mid(SSCCkoodi, i, 1)))
    Next i
    LaskeSSCCTarkenne = Right((Str(10 - (Summa Mod 10))), 1)
End Function
```

Kuva 11. SSCC-tarkistenumeron laskenta

LaskeSSCCTarkenne-funktiota kutsunut sovellus liittää palautetun tarkistenumeron osaksi SSCC-lähetystunnusta.

4.8.3 Viivakoodien tuottaminen

Luvussa 3.4 esiteltiin Code 128 -viivakoodin rakenne, erilaisia toteutustapoja, tarvittava merkkikoodaus ja tarkistenumeron laskentamenetelmä. Code 128 -viivakoodi voidaan tuottaa mistä tahansa arvosta, joka halutaan esittää mahdollisimman lyhyenä 1D-viivakoodina. Samassa yhteydessä todettiin, että SSCC-tunnus tulostetaan GS1-128 -viivakoodina, joka perustuu täysin Code 128 -viivakoodimenetelmään, mutta GS1-128 -viivakoodiarvoon on lisättävä ennen Code 128 -viivakoodiarvon muodostamista FNC1-merkki aloitusmerkin ja SSCC-koodin väliin.

Postin työasemissa Code 128 -viivakoodit tulostetaan JLCode128-fontilla mutta sitä ennen viivakoodiksi haluttavasta arvosta on muodostettava ISO-standardin mukainen viivakoodiarvo. Sovelluksessa tulostettavaksi kelpaava viivakoodiarvo muodostetaan TeeCode128-aliohjelmalla, joka saa parametrina koodin ja joka palauttaa kutsuvalle ohjelmalle tulostettavan viivakoodiarvon. TeeCode128-funktion VBA-koodi on esitetty liitteessä 4.

Aliohjelman toteuttamisessa pääkohdat olivat seuraavat:

- Päätellään koodin sisältämien merkkien ja niiden sijaintien perusteella optimaalinen Code 128 -aloitusmerkistö ja mahdolliset merkistön vaihdot. Tavoitteena on tuottaa mahdollisimman lyhyt viivakoodi.
- Jos kyseessä on GS1:n SSCC-koodi, niin lisätään aloitusmerkin jälkeen FNC1-merkki, jonka arvo on 102.
- Käsitellään annettu koodi merkki kerrallaan valitun merkistön mukaisesti. Jos valittu merkistö on Code C, niin numeeriset arvot käsitellään pareittain.

- Lasketaan ja lisätään annettuun koodiin tarkistusluku ja loppumerkki.
- Muunnetaan koko annettu koodi tulostettavaksi merkkijonoksi.

4.8.4 Osoitekorttien tulostus

Valitun tuotteen perusteella sovellus ohjaa lähetystiedot oikean osoitekorttitaulukon täyttävään ja tulostavaan aliohjelmaan, jossa myös muodostetaan tuotteen mukainen lähetystunnus ja vastaava Code 128 -viivakoodiarvo. Tulostus voidaan ohjata myös esikatseltavaksi, jolloin tulostus ei ohjaudu kirjoittimelle. Kuvassa 12 on esitetty VBA-koodia lomakepohjana toimivan taulukon tulostamisesta.

```
'TULOSTUS
If Worksheets("Asetukset").Range("Tulostus").Value Then
    If Worksheets("Asetukset").Range("Esikatselu").Value Then
        Worksheets("Label_GS1").PrintPreview
    Else
        Worksheets("Label_GS1").PrintOut
    End If
End If
```

Kuva 12. Osoitekortin tulostus

Liitteissä 5, 6 ja 7 on esitetty malliksi kolmella erilaisella osoitekorttipohjalla DevParcel-ohjelmasta tulostettuja osoitekortteja, joissa on eri menetelmin luodut lähetystunnisteet.

4.8.5 Tiedoston muodostaminen

Kappaleessa 2.3 esittelin kaksi erilaista tapaa, joilla VBA-sovelluksissa voidaan muodostaa määrämuotoisia tekstitiedostoja. Uudempaan objektimalliin perustuva FSO-malli vaikutti ohjelmoinnin näkökulmasta helpommalta ja mallin tarjoamat objektit tuntuivat kattavan kaikki kansioiden ja tiedostojen käsittelyyn tarvittavat metodit. Yrityksistäni huolimatta, en kuitenkaan onnistunut luomaan sen avulla liitteessä 2 esitettyä Postin Lähetystenseurantajärjestelmän integraatiorajapinnan mukaista määrämuotoista ja positiosidonnaista tiedostoa. Vaikutti siltä, ettei FSO-malli tue käyttäjän määrittelemiä tietotyyppejä (User-Defined Data Types), joita olen aiemmin käyttänyt menestyksellisesti tiedoston tietuerakenteiden määrittämiseen.

Toinen tapa muodostaa määrämuotoisia tekstitiedostoja, perustuu VB:n perinteisiin tiedostojen I/O-lauseisiin ja -toimintoihin. Koska tämä malli tukee myös käyttäjän määrittelemiä tietotyyppejä, päätin toteuttaa tiedoston muodostamisen perinteisellä mallilla ja määrittelin kaikista seitsemästä tietuerakenteista uudet tietuekohtaiset tietotyypit. Kuvassa 13 on esitetty malliksi integraatiotiedoston alkutietueen määritys uudeksi käyttäjän määritte-

lemäksi tyypiksi, jossa alkutietueen kaikille kentille on määritetty vastaavat muuttujat ja loppuun on lisätty rivinvaihto. Lopuksi määritellään vielä uusi koko alkutietueen sisältävä muuttuja juuri luodulla tyypillä, jota käytetään myöhemmin tiedostoon kirjoitettaessa.

```
'*** EDIVAR-tiedosto alkaa
Type edivar_alkutietue
    Alk1 As String * 3         'Tietuetunnus
    Alk4 As String * 10        'Alkuperäinen tiedosto
    Alk14 As String * 10       'Alkuperäinen tiedosto
    Alk24 As String * 35       'Sovellustunnus
    Alk59 As String * 12       'Sanoman luontiaika
    Alk71 As String * 17       'Lähettäjän OVT
    Alk88 As String * 17       'Vastaanottajan OVT
    crlf As String * 2        '<CRLF>
End Type
Public EdivarAlku As edivar_alkutietue
```

Kuva 13. Käyttäjän määrittelemä tietotyyppi

Ennen kuin uusi tiedosto avataan, niin on huolehdittava että käytettävä hakemisto on olemassa, muuten suoritus päättyy virheeseen. Kuvassa 14 on esimerkki sovellukseen toteutetusta VBA-ohjelmoinnista, jossa tarkistetaan käytettävän tiedostohakemiston käytettävyys ja jossa uusi tiedosto avataan binary-tyyppisenä. Sovelluksen taulukoista saadut arvot sekä muut kiinteät arvot siirretään muuttujiin, jotka lopuksi kirjoitetaan tiedostoon edellä määritetyn uuden muuttujatyypin avulla. Liitteessä 8 on esimerkkinä edellä kuvattuihin osoitekortteihin liittyvä integraatio-tiedosto (edivar).

```
'Ensimmäinen
Select Case TietueTyyppi
Case "ALKU"
    'testataan hakemisto ja tehdään tarvittaessa uusi
    EdiHakemisto = ThisWorkbook.Path & "\edi"
    EdiTiedosto = Trim(Worksheets("Asetukset").Range("Tiedosto").Value) _
    & "." & Format(Now, "yyyymmdd") _
    & "_" _
    & Worksheets("Asetukset").Range("Infokoodi").Value _
    & "_" _
    & Worksheets("Asetukset").Range("EdivarLaskuri").Value _
    & ".dat"

    On Error Resume Next
        MkDir EdiHakemisto

    EdiRiviLaskuri = 1
    Open EdiHakemisto & "\" & EdiTiedosto For Binary As #1
    EdivarAlku.Alk1 = "001"
    EdivarAlku.Alk4 = "EDIVARD03A"
    EdivarAlku.Alk14 = "DEVPARCEL"
    EdivarAlku.Alk24 = ""
    EdivarAlku.Alk59 = Left(TulostusAika, 12)
    EdivarAlku.Alk71 = Trim(Worksheets("Asetukset").Range("EdivarOvt").Value)
    EdivarAlku.Alk88 = "003715318644200"
    EdivarAlku.crlf = vbCrLf
    Put #1, 1, EdivarAlku
```

Kuva 14. Tiedoston luonti ja kirjoitus

4.8.6 Tiedonsiirto

Sovellukseen toteutettiin automaattinen integraatiotiedoston siirto palvelimelle kappaleessa 2.4 esitetyllä tavalla, jossa hyödynnettiin Windows käyttöjärjestelmän mukana toimitettua Wininet-API-rajapintaa. Tarvittavat Wininet-rajapinnan tarjoamat toiminnot määriteltiin Funktiot-moduulissa, josta kuvassa 15 on esimerkkinä ftp-siirrossa käytettävä put-komennon määritys.

```
'Send a file using FTP
 Public Declare Function FtpPutFile _
   Lib "wininet.dll" _
     Alias "FtpPutFileA" _
       (ByVal hFtpSession As Long, _
        ByVal lpszLocalFile As String, _
        ByVal lpszRemoteFile As String, _
        ByVal dwFlags As Long, _
        ByVal dwContext As Long) As Boolean
```

Kuva 15. Put-toiminnon määritys Wininet-rajapinnan avulla

Tiedoston siirto toteutettiin SiirräFtp-aliohjelmalla (kuva 16), joka sijoitettiin Proseduurit-moduuliin. Kutsuva ohjelma välittää aliohjelmalle siirrettävän tiedoston nimen ja tiedonsiirto tapahtuu Postin palvelimelle binääri-muodossa passiivisella ftp-yhteydellä.

```
Sub SiirraFtp(ByVal FtpTiedosto As String)
    'Luodaan ftp-parametrit
    Dim tmpFile As String, hostFile As String
    Dim ServerName As String, UserName As String, Password As String
    Dim INet As Long, INetConn As Long, RetVal As Long, ftpMode As Long
    Dim Success As Boolean

    Const ASCII = 1
    Const BINARY = 2

    'ftpMode = 0            'active mode FTP
    ftpMode = &H8000000     'passive mode FTP

    UserName = "xxxxxxx"
    Password = "yyyyyyy"
    tmpFile = "in/" & "tmp." & FtpTiedosto
    hostFile = "in/" & FtpTiedosto

    If Worksheets("Asetukset").Range("Edi").Value = True Then
        RetVal = False
        INet = InternetOpen("MyFTP Control", 1&, vbNullString, vbNullString, 0&)
        If INet > 0 Then
            INetConn = InternetConnect(INet, TESTIPALVELIN, 0&, UserName, Password, 1&, ftpMode, 0&)
            If INetConn > 0 Then
                Success = FtpPutFile(INetConn, EdiHakemisto & "\" & FtpTiedosto, tmpFile, BINARY, 0&)
                If Success Then
                    Success = FtpRenameFile(INetConn, tmpFile, hostFile)
                End If
                RetVal = InternetCloseHandle(INetConn)
            End If
            RetVal = InternetCloseHandle(INet)
        End If
    End If
End Sub
```

Kuva 16. Ftp-tiedonsiirto Wininet-rajapinnan avulla

4.9 Yhteenveto

DevParcel-sovellus toteutettiin alkuperäisten tavoitteiden mukaisesti ja valitut ratkaisut noudattivat erittäin hyvin tietoperustassa esitettyjä näkökohtia. Toteutusprojektissa ei kohdattu varsinaisia ongelmia mutta ohjelmoitavaa oli paljon, joka myös vaikutti ohjelmointiin käytettyyn aikaan. Kaikki sovellukseen suunnitellut toiminnot toteutettiin täysimääräisesti.

VBA-ohjelmakoodi toteutettiin mahdollisimman modulaarisesti siten, että jatkokehittäminen olisi yksinkertaista ja uusien ominaisuuksien lisääminen olisi helppoa. Sovelluksen suunnittelua ja toteutusta kevensi huomattavasti se, että sovellusta käytetään ainoastaan testiaineistojen tekoon, jolloin sovellukseen syötettävien tietojen virhetarkastus on tarkoituksellisesti jätetty vähäiseksi.

Sovelluksen tuottamat tiedot on tarkastettu oikeellisiksi kohdejärjestelmässä. Samoin sovelluksesta tulostetut viivakoodit on tarkastettu erilaisilla viivakoodilukijoilla ja niiden on todettu olevan standardien mukaisia sekä täyttävän Postin tuotantoprosessin vaatimukset.

5 Pohdinta

Tässä opinnäytetyössä suunniteltiin ja toteutettiin DevParcel-sovellus, jonka tarkoituksena on yksinkertaistaa ja nopeuttaa IT-järjestelmämuutosten testaamista ja sitä kautta parantaa liiketoimintaprosessien ja niitä palvelevien järjestelmien laatua. Opinnäytetyö suoritettiin Postin toimeksiantona ja samoin sovellus toteutettiin erillisenä Postin IT-kehitysprojektina, joka on esitetty liitteessä 1. DevParcel-sovellus otettiin Postissa käyttöön joulukuun 2015 alussa.

5.1 Tulosten tarkastelu

Johdannossa esittelin järjestelmäkehityksen tuotosten testauksiin liittyviä ongelmia, joita on tarkoitus lieventää tai poistaa DevParcel-sovelluksella.

Liiketoimintaprosessien kehittäminen tuottaa lähes aina IT-vaatimuksia, jotka toteutetaan IT-kehitysprojekteissa. Postissa on jatkuvasti meneillään useita kymmeniä IT-kehitysprojekteja sekä muita pienempiä toteutettavia muutoksia, joista osa kohdistuu suoraan tai integraatioiden kautta myös Postin Lähetystenseurantajärjestelmään. Projektisuunnitelma sisältää usein testaussuunnitelman, joka perustuu vaatimusmäärittelyssä lueteltuihin uusiin tai muuttuneisiin toimintoihin ja niiden hyväksymisehtoihin. Toimitetut ratkaisut voidaan hyväksyä, kun testaussuunnitelman kaikki testitapaukset on hyväksytty. Toteutettu DevParcel-sovellus on tarkoitettu juuri näiden testitapausten tekoon, joilla simuloidaan asiakkaan toimintaa tehtyjen järjestelmä- tai prosessimuutosten jälkeen. DevParcel-sovellus tuottaa tehokkaasti ja helposti oikeamuotoista ”asiakkaan” lähettämää ennakkotietoa lähetyksistä, joiden vaikutuksia voidaan tarkastella testitapauksia vasten kohdejärjestelmissä. DevParcel-sovellus tuottaa lähetystietojen lisäksi myös viivakoodillisia osoitekortteja, joiden avulla voidaan simuloida asiakkaan lähettämiä ”fyysisiä” paketteja tai kirjeitä.

Koska sovelluksen toteuttamisen lähtökohtana oli, että sovellus toteutetaan Microsoftin Office -pakettiin kuuluvan Excelin päälle VBA-ohjelmointina, niin opinnäytetyön ensimmäisenä tavoitteena oli tutkia VBA-ohjelmoinnin perusteita soveltuvin osin. Osiossa keskityttiin perusteiden lisäksi erityisesti määrämittaisen tiedoston muodostamiseen ja ftp-tiedonsiirtoon. Tavoitteena oli muodostaa asiakasintegraatiota jäljittelevä, Postin määritysten mukainen rajapintatiedosto Postin lähetystenseurantajärjestelmään.

Opinnäytetyön toisena tavoitteena oli selvittää miten Postin määrittelemät lähetystunnukset pystytään tuottamaan ja miten ne pystytään tulostamaan vaatimusten mukaisella viivakoodilla.

Empiirisessä osuudessa kävin läpi kohdeorganisaation sekä sovellukselle asetetut tavoitteet, ongelmat, toteutussuunnitelman ja toteutuksen.

Opinnäytetyön tuloksena toteutettiin DevParcel-sovellus, joka on toimeksiantajalla päivittäisessä käytössä useilla testaajilla, jotka osallistuvat nykyisten sekä tulevien järjestelmien uusien tai muuttuneiden toimintojen testaukseen. Sovellus täyttää kaikilta osin toimeksiantajan sille asettamat vaatimukset erinomaisesti.

Opinnäytetyön aikana selvisi myös, miten monipuolinen ja tehokas työkalu Microsoftin Office -paketin sisältämä VBA on, ainakin Excelin yhteydessä. Koska ratkaisukeinoja on aina lukuisia, niin on tärkeä selvittää VBA-ohjelmointiin liittyvä hyvä ohjelmointitapa, joka auttaa muita koodin ylläpitäjiä ymmärtämään ohjelmoidun ratkaisun rakenteet ja helpottaa myös mahdollista virheselvitystä.

5.2 Johtopäätökset sekä mahdolliset kehitys- ja jatkotutkimusehdotukset

Koska yrityksen tietojärjestelmiä kehitetään jatkuvasti, niin myös järjestelmien rajapinnat muuttuvat. DevParcel-sovelluksen käyttö aloitettiin Postissa joulukuussa 2015, jolloin sillä tuotetuilla materiaaleilla testattiin nykyiseen Lähetystenseurantajärjestelmään tehtyjä muutoksia ja testattiin myös tulevaan Lähetystenseurantajärjestelmään toteutettavaa samaa rajapintaa. Kuitenkin tammikuussa 2016 päätettiin, että tulevaan Lähetystenseurantajärjestelmään toteutetaan myös xml-rajapinta, joka on myös testattava. Koska DevParcel-sovelluksen lähtökohtana oli, että rajapinnat tulevat muuttumaan ja niitä voi olla useampia, niin toteutin DevParcel-sovelluksen toiseen versioon lisäksi myös Postin xml-rajapinnan (POSXML) mukaisen siirtotiedoston muodostamisen Microsoftin DOM API-rajapinnan avulla. Uuden rajapinnan lisäys onnistui hienosti ja toteutin sen noin kahdeksan tunnin ohjelmoinnilla.

Excel VBA:lla toteutettuun DevParcel-sovellukseen on sen rakenteen vuoksi helppo toteuttaa uusia tiedostorajapintoja, joiden valinta voidaan tehdä Asetukset-taulussa tai sitä käyttävällä lomakkeella. Muita jatkokehitettäviä kohteita ja tutkittavia asioita voisi olla:

- Miten sftp-siirrot pystytään toteuttamaan VBA-sovelluksessa?

- Tukeeko VBA JSON-tiedostojen muodostamista ja miten se toteutetaan?
- Voisiko 1D-viivakoodit muodostaa VBA/VB-grafiikalla ilman erillisiä ja maksullisia Code 128 -viivakoodi-fontteja?
- Miten VBA tukee 2D-viivakooditn tulostamista?
- Miten VBA:lla toteutetaan SOAP WebService-kutsut?
- Miten VBA:lla toteutetaan REST -kutsut?
- Miten VBA:lla voidaan toteuttaa tehokkaat tietokanta-yhteydet Oracle- ja MySQL-tietokantoihin?
- Kun VBA-sovelluksesta julkaistaan päivitys, niin miten päivityksen voisi automatisoida loppukäyttäjille?

5.3 Opinnäytetyöprosessin ja oman oppimisen arviointi

Opinnäytetyöprosessi oli isona kokonaisuutena erittäin haastava mutta mielenkiintoinen. Opinnäytetyön tekemisestä helpotti huomattavasti, kun sain työnantajaltani toimeksiannon sovelluksen toteuttamiseksi. Tosin, toimeksiantoon sisältyi, että teen koko opinnäytetyöni ja sovellukseen liittyvän koodauksen vapaa-aikanani. Tämä aiheutti itselleni aikatauluongelmia, kun samaan aikaan Postissa eteni muutamia suuria IT-kehityshankkeita, joissa mukana olevat testaajat odottivat DevParcel-sovellusta käyttöönsä helpottaakseen järjestelmätestausta. Alun perin tarkoituksenani oli edetä opinnäytetyöprosessissa ohjeen mukaisesti mutta toimeksiantajani aikataulujen vuoksi jouduin toteuttamaan sovelluksen ensin.

Koska olen toiminut 20 vuotta Postissa integraatioasiantuntijana, tunsin rajapintavaatimukset, käytössä olevat standardit ja ratkaisun tarpeen erinomaisesti. Aikaisemman ohjelmoijataustani lisäksi, minulla oli myös hieman kokemusta Excel/VBA-ohjelmoinnista ja uskoin pystyväni toteuttamaan vaaditun sovelluksen. Vaikka toteutettu sovellus toimiikin hyvin ja täyttää sille asetetut vaatimukset täydellisesti, niin opinnäytetyön tietoperustassa tuli esiin niin paljon itselleni uutta tietoa, että nyt toteuttaisin sovelluksen erilaisin VBA-ohjelmointiratkaisuin.

Varsinainen opinnäytetyöprosessi oli itselleni uusi kokemus, joka osoittautui koodausta lukuun ottamatta vaikeaksi. Henkilökohtaisesti vaikeinta oli tuttuun aiheeseen uuden tulokulman löytäminen, jossa ei enää riittänytkään se, että asioita vain teki parhaan ymmärryksensä mukaisesti. Tietoperustaa selvittäessä kirjastosta lainattu puolenmetrin kirjopinokaan ei aluksi auttanut yhtään, ennen kuin oivalsin lähteiden käytön ja aloin löytämään varteenotettavia näkökulmia. Internet-lähteistä oli paljon apua mutta samoin kuin

kirjalähteissä, huomasin usein tietojen olevan vanhentuneita, puutteellisia tai jopa virheellisiä. Yllättävän paljon aikaa kului lähteiden asiasisällön tarkistamiseen, jolloin ymmärsin myös useiden lähteiden käytön tarpeellisuuden muodostaessani totuuden mukaista kokonaiskuvaa pienestäkin aiheesta.

Onnistuneen kehitysprojektin toteutus vahvisti aikaisempaa mielikuvaani Excel VBA-ohjelmoinnin mahdollisuuksista toteuttaa nopeasti ja kevyesti vaativiakin ratkaisuja liiketoiminnan tarpeisiin, kuitenkaan työnlaadusta ja lopputuloksista tinkimättä.

Uskon opinnäytetyöprosessin kautta oppineeni paljon tutkimustyöstä, varsinkin kun aihe perustuu työelämässä havaittuihin käytännön tarpeisiin. Menetelmä ohjaa tutkijaa lähestymään tutkimusongelmaa rakenteellisesti ja kriittisesti sekä kirjoittamaan loogisesti etenevän raportin perusteluineen. Tulen varmasti jatkossa tarvitsemaan ja käyttämään oppimiani taitoja niin työantajani kuin oman ammatillisen kehittymisenikin hyväksi.

Lähteet

Adams Communications 1995. Code 128 Barcode Table of Characters. Luettavissa: http://www.adams1.com/128table.html. Luettu: 4.1.2016.

Adams Communications 2013. All About Code 128 Barcode. Luettavissa: http://www.adams1.com/128code.html. Luettu: 4.1.2016.

BarCodeIsland.com, 2006. Code 128 Symbology. Luettavissa: http://www.barcodeisland.com/code128.phtml. Luettu: 10.1.2016.

Bar Code Graphics, 2013. GS1-128 Barcodes. Luettavissa: http://www.gs1-128.info. Luettu: 4.1.2016.

GS1 2015. S1 General Specification. Luettavissa: http://www.gs1.org/sites/default/files/docs/barcodes/GS1_General_Specifications.pdf. Luettu: 1.12.2015.

GS1a. SSCC. Luettavissa: http://www.gs1.fi/gs1-jarjestelma/gs1-yksiloinnin-avaimet/sscc. Luettu: 1.12.2015

GS1b. How to calculate a check digit manually. Luettavissa: http://www.gs1.org/how-calculate-check-digit-manually. Luettu: 1.12.2015

Hokkanen, S. & Karhunen, J. 2014. Johdatus logistiseen ajatteluun. Sho Business Development Oy. Jyväskylä.

Logistiikan Maailma 2015. Ohjausjärjestelmät/Viivakooditekniikka. Luettavissa: http://www.logistiikanmaailma.fi/wiki/Viivakooditekniikka. Luettu: 8.1.2016.

Mack, G. 2002. History of Visual Basic. Luettavissa: http://www.ojodepez-fanzine.net/network/qbdl/history_of_visual_basic.html. Luettu: 28.12.2015.

Merensalmi, J. 2007. Excel VBA yrityskäytössä. WSOY. Helsinki.

Microsoft 2015a. Visual Basic Concepts - Processing Drives, Folders, and Files. Luettavissa: https://msdn.microsoft.com/en-us/library/aa241712%28v=vs.60%29.aspx. Luettu: 31.12.2015.

Microsoft 2015b. About WinINet. Luettavissa: https://msdn.microsoft.com/en-us/library/windows/desktop/aa383630%28v=vs.85%29.aspx. Luettu: 31.12.2015.

Microsoft 2015c. FTP Session. Luettavissa: https://msdn.microsoft.com/en-us/library/windows/desktop/aa384180%28v=vs.85%29.aspx. Luettu: 31.12.2015.

Niblack, J. 2008. Creating an FTP Component in Visual Basic Luettavissa: http://www.codeguru.com/vb/vb_internet/article.php/c19511/Creating-an-FTP-Component-in-Visual-Basic.htm. Luettu: 31.12.2015.

Posti Group 2015. Yleisesite. Luettavissa: http://www.posti.com/attachments/financials/Posti_Group_yleispresentaatio.pdf. Luettu:1.12.2015

Sakki, J. 2014. Tilaus-toimitusketjun hallinta, digitalisoitumisen haasteet. Jouni Sakki Oy. Vantaa.

Shepherd, R. 2006. Excel-ohjelmointi – Tehokas hallinta. Readme.fi. Helsinki.

Universal Postal Union 2014. S10, Identification of postal items – 13-character identifier. Luettavissa: http://www.upu.int/uploads/tx_sbdownloader/S10TechnicalStandard.pdf. Luettu: 1.12.2015.

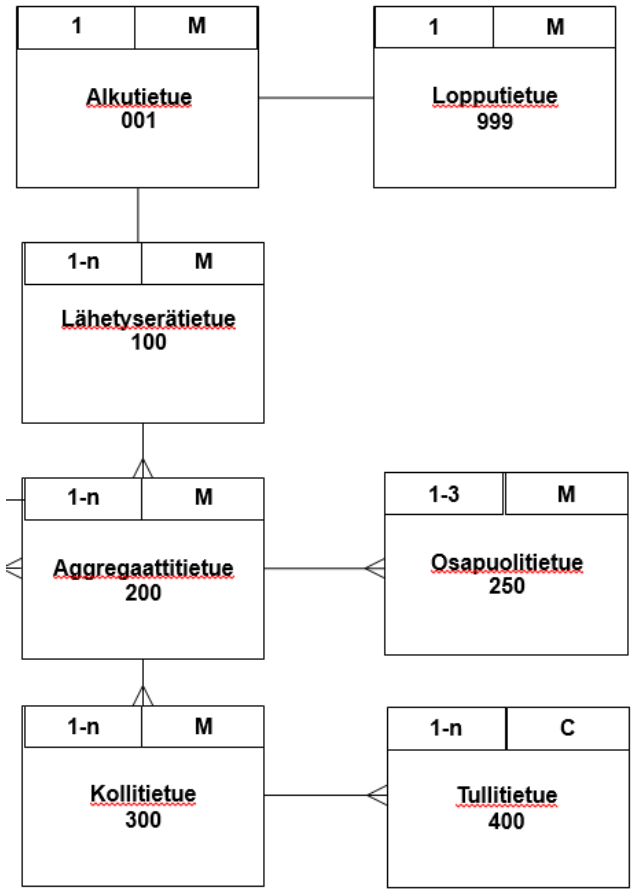
Universal Postal Union 2015. Catalogue of UPU Standards. Luettavissa: http://www.upu.int/uploads/tx_sbdownloader/catalogueStandardsCatalogueOfUpuStandardsEn_01.pdf. Luettu: 4.1.2016.

Walkenbach, J. 2013. Excel ® 2013 Power Programming with VBA. Kindle Edition. Wiley, USA. ISBN 978-1-118-49040-2 (ebk).

Liitteet

Liite 1. Projektisuunnitelma

Liite 2, 1/2. Postin EDIVAR D.03A, siirtotiedoston looginen rakenne



Liite 2, 2/2. Postin EDIVAR D.03A, tietosisältö

Kuljetussanoman tietuera-kenteet							
Alku positio	**Kentän-pituus**	**Kentän muoto**	**Pa-kollisuus (K/E)**	**Kentän nimi**	**Esimerkki/Vakioarvo**	**Huomioitavaa**	**Kohdekenttä**
Alkutietue				**Tiedoston otsikkotiedot**			
1	3	N	K	Tietuetunnus	**001**	**Vakio**	
4	10	A	K	Tiedoston tunniste	EDIVARD03A	**vakio (Sano-matyyppi+versio+release)**	
14	10	A	E	Alkuperäinen tiedoston tunniste	IFCSUM	**vakio (Sano-matyyppi+versio+release)**	
24	35	A	E	Sovellustunnus			
59	12	N	K	Sanoman luontiaika	200109051500	**Ccyymmddhhmm**	
71	17	A	K	Sanoman lähettäjän tunnus	699999	**Asiakkaan sopimustunnus**	
88	17	A	K	Sanoman vastaanottajan OVT-tunnus	003715318644200	**Suomen Posti Oyj**	
Lähetyserätietue				**Lähetyserän yhteiset tiedot**			
1	1	N	K	Tietuetunnus	**100**	**Vakio**	
4	14	A	E	Sanoman ID			
18	20	A		Lähetyserän ID			
38	12	N		Depinsulkuaika/ Kuljetus valmiina kuljetet-tavaksi	200303051500	**Ccyymmddhhmm**	
50	3	A		Kuljetuksen laatu	10	**10, 20 tai 30**	
53	17	A	K	Kuljetusasiakirjan tunnus (Rahtikirjanu-mero)	87654321		
70	12	N		Kuljetusasiakirjan luontiaika	200303051600	**Ccyymmddhhmm**	
82	3	A		Kuljetussopimuksen status	14	**14, 22, tai 16**	
85	3	A		Postilaji	A	**A, B tai C**	
88	17	A		Kuljettaja	AY		
105	17	A		Kuljetuksen numero	812		
122	3	A		Kuljetustapa	4	**1=sea, 2=rail, 3=road tai 4=air**	
125	5	A	K	Kuljetuksen lähtöpaikka	00230	**Postinro / SEARN**	

130	12	N		Kuljetuksen lähtöaika	200303051900	**Ccyymmddhhmm**	
142	5	A	K	Kuljetuksen määräpaikka	01000	**Postinro / FIHEL**	
147	12	N		Kuljetuksen saapumisaika	200303051950	**Ccyymmddhhmm**	
159	5	A	E	Kuljetuksen vaihtopaikka	GEFRA		
Aggregaattitietue				**Aggregaatin tiedot**			
1	3	N	K	Tietuetunnus	**200**	**Vakio**	
4	20	A		Lähetyserän ID			
24	17	A	K	Kuljetusasiakirjan tunnus	87654321		
41	35	A	E	Kuljetusyksikön ID /Pussilapun ID			
76	6	A		Kuljyks. Juokseva nro depin sisällä			
82	2	A	K	Kuljetusyksikkötyyppi	BG	**CN, BG, PU, CG, GU, PC,…**	
84	2	A	E	Kuljetusyksikön luokka	UA	**UA, UI, UL, UN, CN, CE, EM, ED**	
86	6	A		Maakoodi, josta tulossa	SE		
92	8	N	E	Aggregaatin bruttopaino yhteensä, g	00036500	**= 36.5 kg**	
100	5	N	E	Aggregaatin kollien lukumäärä yhteensä	00003	**= 3 kpl**	
105	5	A	E	Info-koodi	50200		
110	4	N	E	Palautettujen kulj.yks. lkm	0004		
114	2	A	E	Kuljetusyksikön kuntokoodi	20	**20 – 25**	
116	12	N	E	Käsittelyynottoaika	200303052030	**Ccyymmddhhmm**	
128	1	A	E	Perillesaamattomuuskoodi	P	**Katso lisätiedot**	
129	18	N	E	Kuljetusvakuutusarvo sentteinä	3400000	**= 34000.00 EUR**	
147	3	A	E	Kuljetusvakuutusarvon valuuttakoodi	EUR	**Vakio**	
150	3	A	E	Sisältökoodi	M	**Katso lisätiedot (G, P,E, S, M, J, K)**	WAYBILL ja Postaalinen huomioitava
153	18	N	E	Kauppatavaran arvo	150000	**= 1500.00 USD**	
171	3	A	E	Kauppatavaran arvon valuuttakoodi	USD		
174	5	A	K	Kuljetuksen lähtöpaikka	00230	**Postinro**	
179	5	A	K	Kuljetuksen määräpaikka	00100	**Postinro, vakio**	
184	1	A	E	Sisältää dokumetit	1	**0=Ei, 1=Kyllä**	
185	1	A	E	Sisältää vaarallisia aineita	1	**0=Ei, 1=Kyllä**	
186	3	A	E	Käsittelyohje	N	**R=Rekisteröity, V=Vakuutettu, N=Normaali**	
189	1	A	E	Sisältää pikapostia	1	**0=Ei, 1=Kyllä**	
190	1	A	E	Vapaa terminaalimaksuista	1	**0=Ei, 1=Kyllä**	

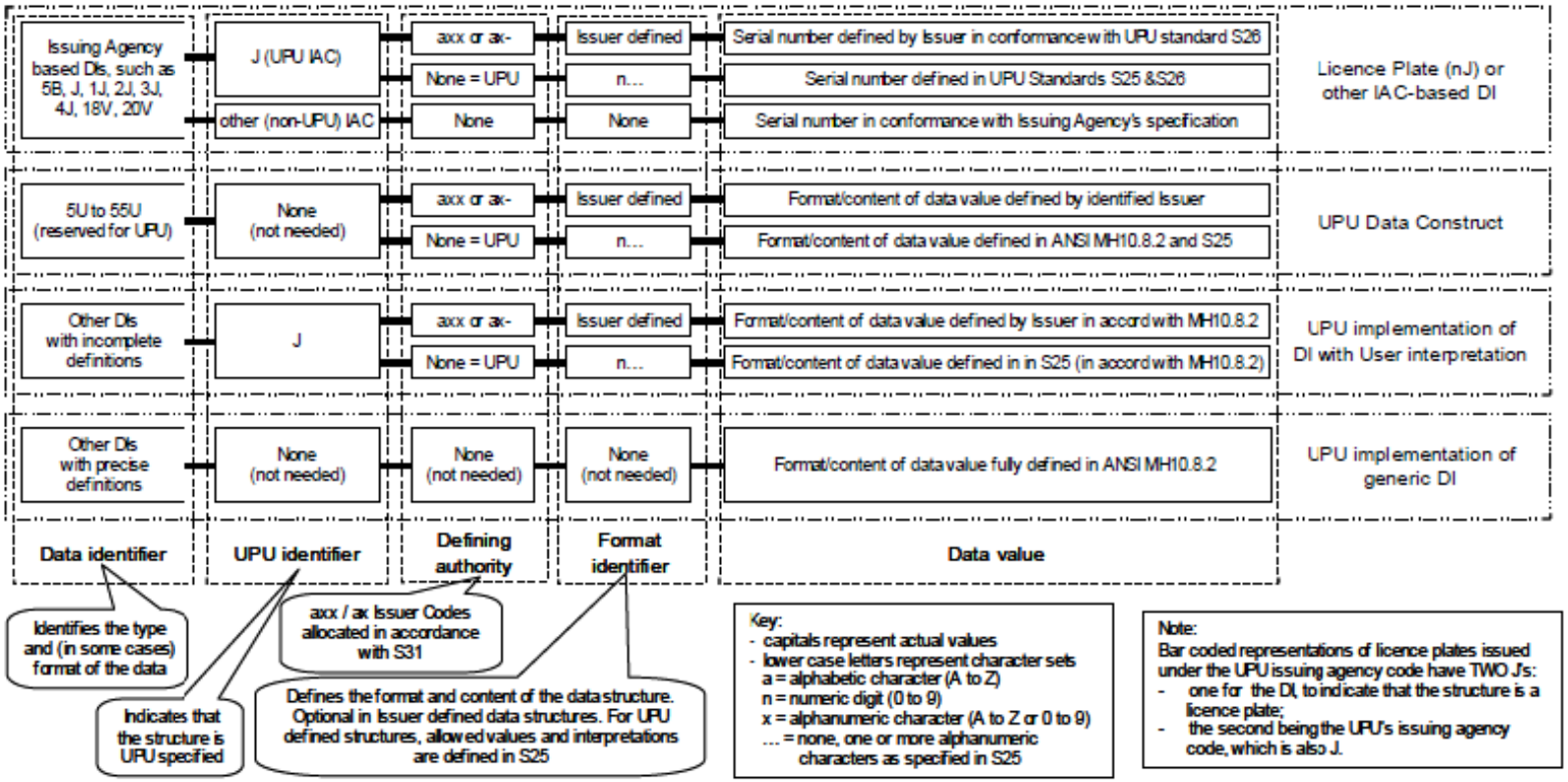
191	1	A	E	Palautetaanko kuljetusyksikkö	1	**0=Ei, 1=Kyllä**	
192	**15**	**A**	**E**	**Info-koodi2**	**1234ABCDEF56789**		**Tulee käyttöön 2010 aikana.**

Osapuo-litietue				**Aggregaatin osapuolet**			
1	3	N	K	Tietuetunnus	**250**	**Vakio**	
4	35	A	E	Kuljetusyksikön ID /Pussilapun ID			
39	2	N	K	Rooli	10	**10=Lähettäjä, 20=Vastaanottaja, 30=Rahdinmaksaja**	
41	6	A	E	Sopimustunnus	699999		
47	35	A	E	Nimi 1	LÄHETTÄJÄ OY		
82	35	A	E	Nimi 2	LOGISTIIKKAPALVELUT		
117	35	A	E	Nimi 3	LAURA LÄHETTÄJÄ		
152	35	A	E	Katusoite	POSTINTAIVAL 7		
187	25	A	E	Postitoimipaikka	HELSINKI		
212	9	A	E	Postinumero	00230		
221	2	A	E	Maakoodi	FI		
223	35	A	E	Puhelinnumero	12345 1234		
258	35	A	E	Sähköpostiosoite	LAURA.LAHETTAJA@XXX.FI		
293	35	A	E	Yhteyshenkilö	JIM KEARNEY		
328	11	A	E	Y-tunnus	01234567		
339	12	A	E	Verotustunnus	01234567		
Kolli-tietue				**Lähetyksen kollitiedot**			
1	3	N	K	Tietuetunnus	**300**	**Vakio**	
4	35	A	E	Kuljetusyksikön ID /Pussilapun ID			
39	35	A	E	Lähetyksen ID	JJFI69999912345678901	**Jos jätetään tyhjäksi, niin TR tekee.**	
74	3	A		Käsittelyohje	N	**R=Rekisteröity, V=Vakuutettu, N=Normaali**	
77	3	A		Jakeluohje		**F, C**	
80	3	A	E	Epäonnistuneen jakelun syykoodi		**10 23, 99**	
83	1	A	E	Pikalähetys	E	**0=Ei, 1=Kyllä**	
84	3	A	K	Kollilajikoodi	PC	**Katso lisätiedot**	

87	3	N	E	MPS-lähetyksen kollimäärä	03	**Tämä kolli on osa 3 kollin lähe-tystä**	
90	4	A	K	Tuotetunnus	2015	**Katso lisätiedot**	
94	28	A	E	Lisäpalvelut, 7 kpl	30033102	**Katso lisätiedot**	
122	8	N	E	Kollin bruttopaino, g	00015530	**= 15.530kg**	
130	8	N	E	Kollin nettopaino, g	00010530	**= 10.530 kg**	
138	8	N	E	Kollin tilavuus, cm3	00115000	**=115 dm3 = 0.115m3**	
146	35	A	E	Postiennakon tilinumero	876543-112233	**Suomalainen tai IBAN-tilinumero**	
181	35	A	E	Postiennakon maksuviite	1234 123456 890		
216	18	N	E	Kollin postiennakkomäärä	00016980	**= 169,80 EUR**	
234	3	A	E	Kollin postiennakon valuuttakoodi	EUR	**Vakio**	
237	35	A	E	Laskun tai proformalaskun numero	123456789/7175		
272	7	N	E	Lähetyksen vakuutusarvo (SDR)	0010003	**= 100,03 SDR**	
279	12	N	E	Jakelu aikaisintaan	200308201230	**Ccyymmddhhmm**	
291	12	N	E	Jakelu viimeistään	200308201300	**Ccyymmddhhmm**	
303	35	A	E	Lähettäjän myyntitilausnumero	12345678	**Tieto välitetään laskulle**	
338	35	A	E	Toinen lähetystunniste		**Tässä välitetään mahdollinen asiakkaan oma lähetystunnus**	
373	11	A	E	Postiennakon BIC	ABCDE123456		Tulee käyttöön 2010 aikana.
384	**8**	**N**	**E**	**LQ paino lähetyksessä, g**	**00010120**	**=10,120 kg**	**Tulee käyttöön 2011 kesällä**
392							
Tulli-tietue				**Lähetyksen sisältöerittely**			
1	3	N	K	Tietuetunnus	**400**	**Vakio**	
4	35	A	E	Tavaran kuvaus	SPORT SHOES		
39	35	A	E	Määrä ja laji	5 PAIRS		
74	8	N	E	Nettopaino, g	3500	**= 3.5 kg**	
82	2	A	E	Alkuperämaa	FI		
84	12	A	E	Tullinimike	64029996		
96	18	N	E	Arvo	54350	**= 543.50 EUR**	
114	3	A	E	Valuuttakoodi	EUR	**EUR/XXX**	
Lopputietue				**Tiedoston lopputietue**			

1	3	N	K	Tietuetunnus	**999**	**Vakio**	
4	6	N	K	Tiedoston tietueiden yhteismäärä	000005	**mukaan luettuna lopputietue**	

Liite 3. UPU Standard S24-5 / Representation of postal information using data identifiers



© UPU 2002 – All rights reserved

Liite 4, 1/3. Function TeeCode128

```
Public Function TeeCode128(AnnettuKoodi As String) As String
    Dim TaulukkoC128(50) As Integer   'taulukko code128-arvoille
    Dim Code128 As String
    Dim OnkoNumero As Boolean
    Dim Csarja As String    'numerot
    Dim Bsarja As String    'merkit
    Dim LaskuriC128 As Integer
    Dim Summa As Integer
    Dim i As Integer

    'poistetaan saadusta parametristä tyhjät
    AnnettuKoodi = Trim(AnnettuKoodi)

    'erotellaan saadusta koodista kirjaimet ja numerot lukemalla koodi loppusta alkuun
    'tarkotuksena tutkia voidaanko annettu koodi tehdä kokonaan b- tai c-merkistöllä
    OnkoNumero = True 'oletetaan että annettu koodi sisältää vain numeroita
    For i = Len(AnnettuKoodi) To 1 Step -1 'luetaan saatu koodi lopusta alkuun
        If IsNumeric(Mid(AnnettuKoodi, i, 1)) And OnkoNumero = True Then
            Csarja = Mid(AnnettuKoodi, i, 1) & Csarja 'löydettiin edelleen numero
         Else
            OnkoNumero = False 'löydettiin kirjain ja kaikki seuraavat merkitään kirjaimiksi
            Bsarja = Mid(AnnettuKoodi, i, 1) & Bsarja
        End If
    Next i

    'varmistetaan että löydettyjä numeroita on parillinen määrä
    If Len(Csarja) > 3 And Len(Csarja) Mod 2 <> 0 Then 'numeroita pitää olla min. 4 ja lukumäärä parillinen
        Bsarja = Bsarja & Mid(Csarja, 1, 1) 'siirretään yksi numero kirjainjonoon että saadaan numeromäärä saadaan parilliseksi
        Csarja = Mid(Csarja, 2)
    End If
```

Liite 4, 2/3. Function TeeCode128

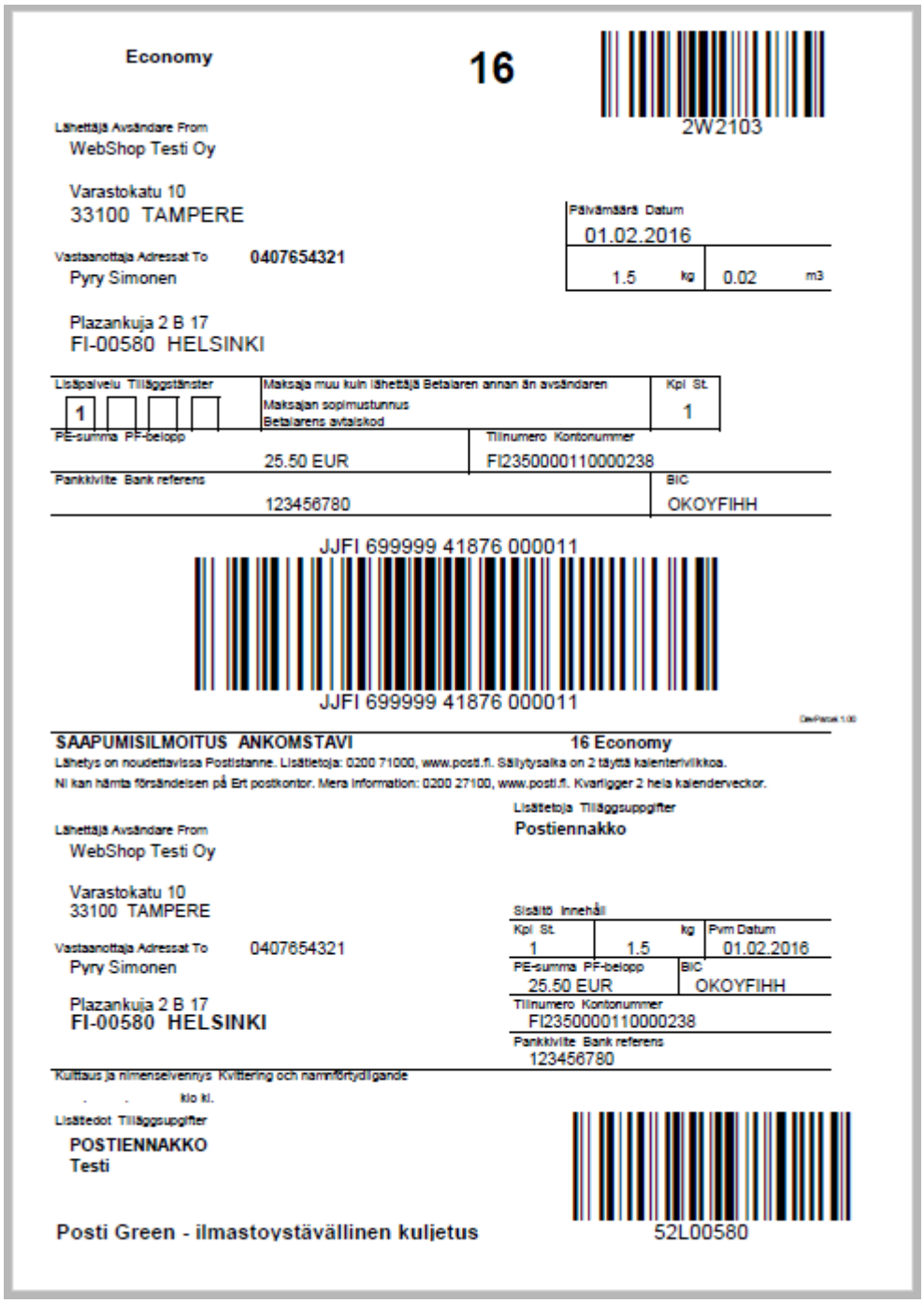
```
'määritellään aloituskoodi
If Csarja = AnnettuKoodi Then    'kaikki oli numeroita
    TaulukkoC128(0) = 105        'tehdään kokonaan 128c-koodilla
    LaskuriC128 = 1
    If Mid(AnnettuKoodi, 1, 2) = "00" And Len(AnnettuKoodi) = 20 Then 'on SSCC
        TaulukkoC128(1) = 102        'lisätään GS1:n "FNC 1"
        LaskuriC128 = 2
    End If
Else
    TaulukkoC128(0) = 104        'aloitetaan 128b-koodilla
    LaskuriC128 = 1
End If
'LaskuriC128 = 1

' muodostetaan lopullinen string Code128 luoduista kirjain- ja numerotaulukoista
' aloitetaan käsittelemällä kirjaimet, mikäli niitä löytyy
For i = 1 To Len(Bsarja) 'tehdään mahdollinen alku 128b-koodilla
    TaulukkoC128(LaskuriC128) = Asc(Mid(Bsarja, i, 1)) - 32 'tallennetaan merkkiarvo - 32 taulukkoon
    LaskuriC128 = LaskuriC128 + 1
Next i
' käsitellään numeerinen jono numeropari kerrallaan
For i = 1 To Len(Csarja) Step 2
    If i = 1 And TaulukkoC128(0) = 104 Then     'aloitettiin 128b-koodilla
        TaulukkoC128(LaskuriC128) = 99          'ja tarvitaan vaihto vaihdetaan 128c-koodiin
        LaskuriC128 = LaskuriC128 + 1
    End If
    TaulukkoC128(LaskuriC128) = Val(Mid(Csarja, i, 2)) 'tallennetaan numeropari taulukkoon
    LaskuriC128 = LaskuriC128 + 1
Next i
'lasketaan tarkistusluku
Summa = TaulukkoC128(0)             'aloitusarvon kerroin on 1
For i = 1 To LaskuriC128 - 1
    Summa = Summa + (TaulukkoC128(i) * i)   'loppusarjan kertoimet ovat 1,2,3,...
Next i
TaulukkoC128(LaskuriC128) = Summa Mod 103   'tarkistusluku on jakojäännös jaettuna tulojen summa 103:lla
LaskuriC128 = LaskuriC128 + 1
TaulukkoC128(LaskuriC128) = 106 ' STOP
```

Liite 4, 3/3. Function TeeCode128

```
    'Käytettäessä "JL code128 korkea" -fonttia, tarvitaan arvoille korjaus, että viivatkoodit tulostuvat oikein
    For i = 0 To LaskuriC128
        Select Case TaulukkoC128(i)
            Case 0
                TaulukkoC128(i) = 204
            Case 103
                TaulukkoC128(i) = 200
            Case 104
                TaulukkoC128(i) = 201
            Case 105
                TaulukkoC128(i) = 202
            Case 106
                TaulukkoC128(i) = 203
            Case Is > 94
                TaulukkoC128(i) = TaulukkoC128(i) + 97
            Case Else
                TaulukkoC128(i) = TaulukkoC128(i) + 32
        End Select
        ' talletetaan alkuperäinen tai korjattu merkki
        Code128 = Code128 & Chr(TaulukkoC128(i))
    Next i
    TeeCode128 = Code128
End Function
```

Liite 5. Postin Economy 16 + Licence Plate -viivakoodi



Economy

16

2W2103

Lähettäjä Avsändare From
WebShop Testi Oy

Varastokatu 10
33100 TAMPERE

Päivämäärä Datum
01.02.2016

1.5 kg | 0.02 m3

Vastaanottaja Adressat To 0407654321
Pyry Simonen

Plazankuja 2 B 17
FI-00580 HELSINKI

Lisäpalvelu Tilläggstänster: 1
Maksaja muu kuin lähettäjä Betalaren annan än avsändaren
Maksajan sopimustunnus
Betalarens avtalskod
Kpl St. 1

PE-summa PF-belopp: 25.50 EUR
Tilinumero Kontonummer: FI2350000110000238
Pankkiviite Bank referens: 123456780
BIC: OKOYFIHH

JJFI 699999 41876 000011

JJFI 699999 41876 000011

SAAPUMISILMOITUS ANKOMSTAVI 16 Economy

Lähetys on noudettavissa Postistanne. Lisätietoja: 0200 71000, www.posti.fi. Säilytysaika on 2 täyttä kalenterivikkoa.
Ni kan hämta försändelsen på Ert postkontor. Mera information: 0200 27100, www.posti.fi. Kvarligger 2 hela kalenderveckor.

Lisätietoja Tilläggsuppgifter
Postiennakko

Lähettäjä Avsändare From
WebShop Testi Oy

Varastokatu 10
33100 TAMPERE

Sisältö Innehåll
Kpl St. 1 | kg 1.5 | Pvm Datum 01.02.2016
PE-summa PF-belopp 25.50 EUR | BIC OKOYFIHH
Tilinumero Kontonummer FI2350000110000238
Pankkiviite Bank referens 123456780

Vastaanottaja Adressat To 0407654321
Pyry Simonen

Plazankuja 2 B 17
FI-00580 HELSINKI

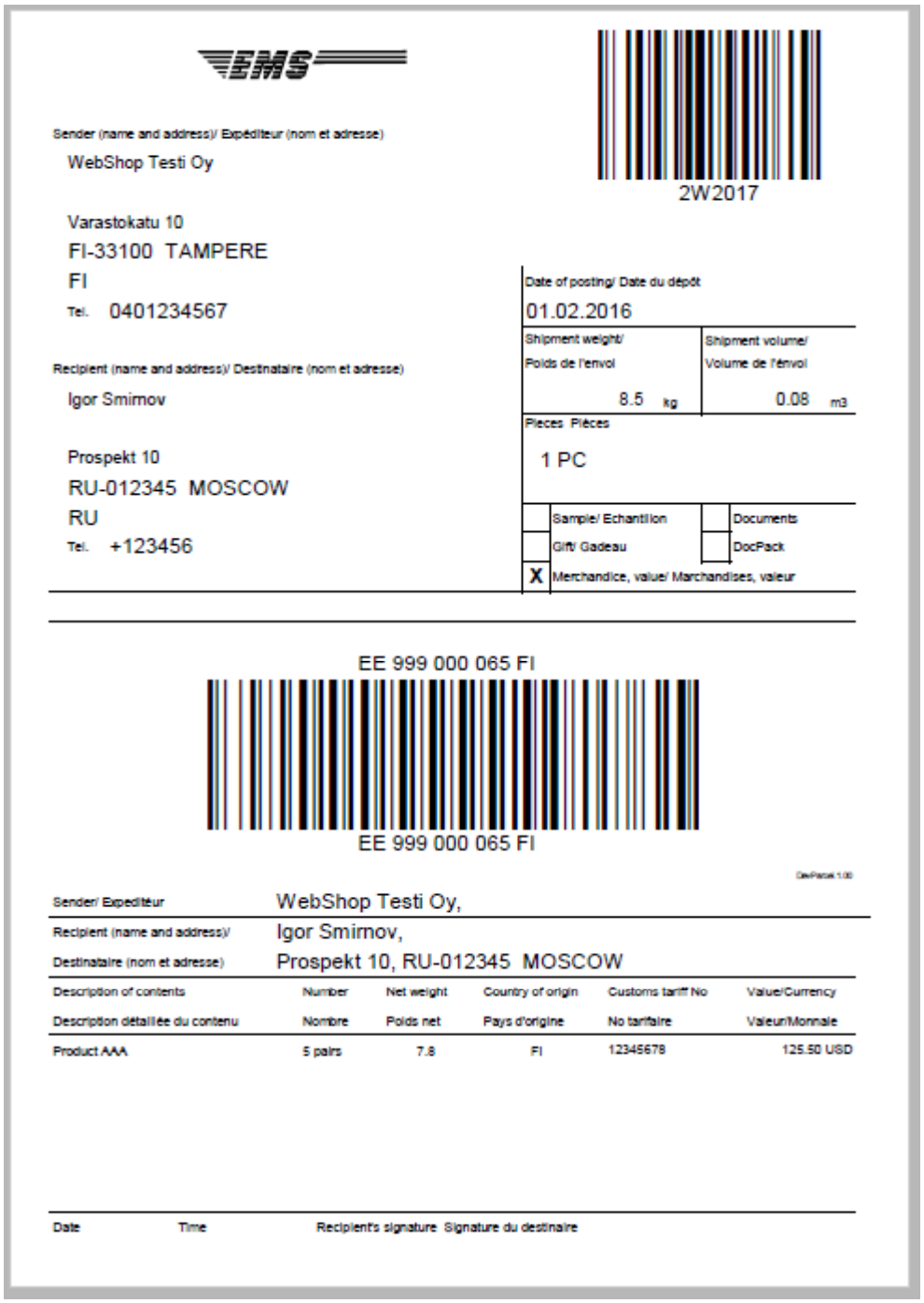
Kuittaus ja nimenselvennys Kvittering och namnförtydligande
. . klo kl.

Lisätiedot Tilläggsuppgifter
POSTIENNAKKO
Testi

52L00580

Posti Green - ilmastoystävällinen kuljetus

Liite 6. UPU EMS Parcel + S10-viivakoodi



EMS

2W2017

Sender (name and address)/ Expéditeur (nom et adresse)

WebShop Testi Oy

Varastokatu 10
FI-33100 TAMPERE
FI
Tel. 0401234567

Recipient (name and address)/ Destinataire (nom et adresse)

Igor Smirnov

Prospekt 10
RU-012345 MOSCOW
RU
Tel. +123456

Date of posting/ Date du dépôt: 01.02.2016

Shipment weight/ Poids de l'envoi: 8.5 kg

Shipment volume/ Volume de l'envoi: 0.08 m3

Pieces Pièces: 1 PC

[ ] Sample/ Echantillon [ ] Documents
[ ] Gift/ Gadeau [ ] DocPack
[X] Merchandice, value/ Marchandises, valeur

EE 999 000 065 FI

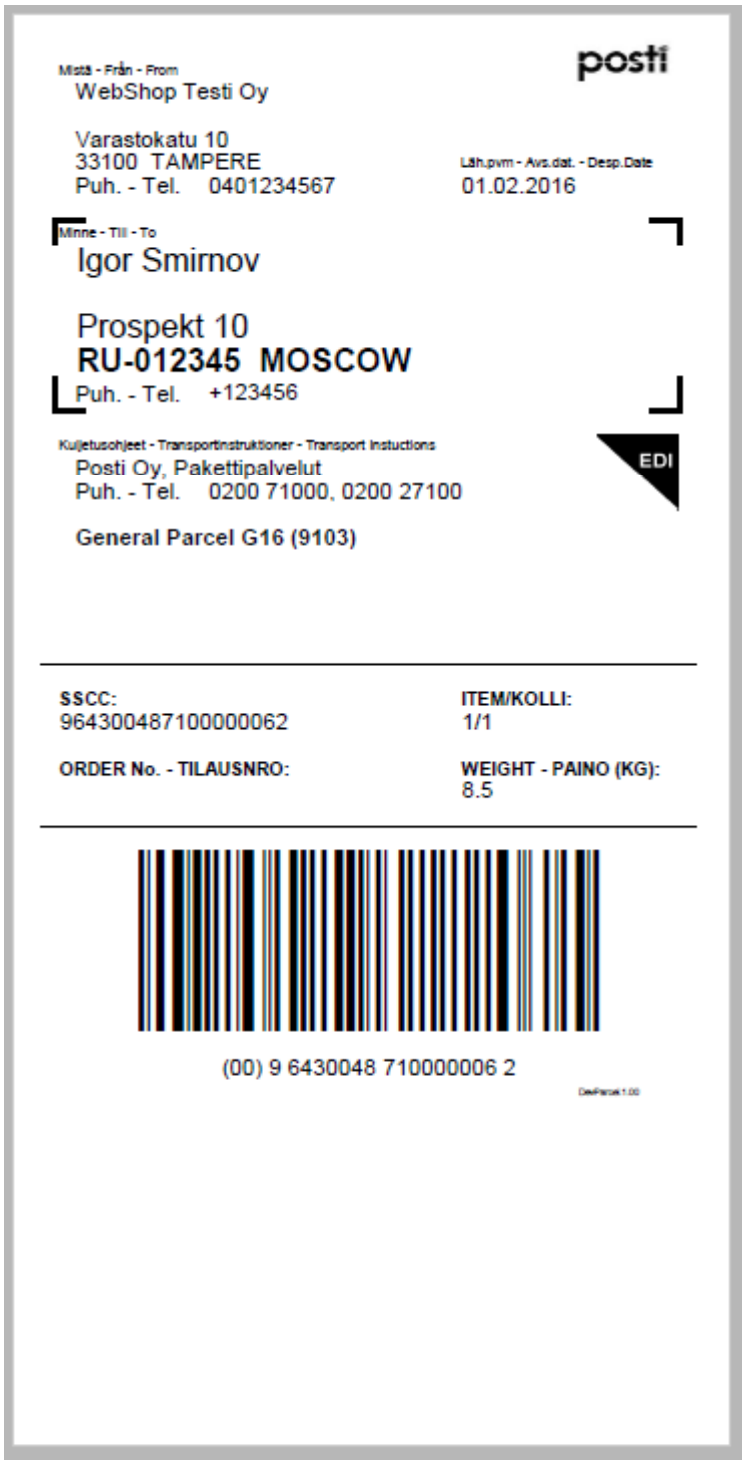
EE 999 000 065 FI

Sender/ Expediteur	WebShop Testi Oy,
Recipient (name and address)/ Destinataire (nom et adresse)	Igor Smirnov, Prospekt 10, RU-012345 MOSCOW

Description of contents / Description détaillée du contenu	Number / Nombre	Net weight / Poids net	Country of origin / Pays d'origine	Customs tariff No / No tarifaire	Value/Currency / Valeur/Monnaie
Product AAA	5 pairs	7.8	FI	12345678	125.50 USD

Date Time Recipient's signature Signature du destinaire

Liite 7. GS1 Kollilappu + SSCC-viivakoodi



Liite 8, Postin edivar-tiedosto

```
001EDIVARD03ADEVPARCEL                                  201602011323DEVPARCEL        003715318644200
100                                                 20160201_ 3      201602011323                                                33100            01000
200                    20160201_ 3                                                  FI    0       1
3310001000        3310C
250                                 10699999WebShop Testi Oy                                                                                          Varastoka-
tu 10                     Tampere                   33100    FI0401234567                          helpdesk@webshop.com
250                                 20      Igor Smirnov                                                                                              Prospekt
1C                        Moscow                    012345   RU+123456                             igor@email.com
300                                 00964300487100000062                           PC 1  9103                                8500             80000
C
400Product AAA                      5 pairs                                  7800    FI12345678    12550             USD
100                                                 20160201_ 3      201602011323                                                33100            01000
200                    20160201_ 3                                                  FI    0       1
3310001000        3310C
250                                 10699999WebShop Testi Oy                                                                                          Varastoka-
tu 10                     Tampere                   33100    FI0401234567                          helpdesk@webshop.com
250                                 20      Igor Smirnov                                                                                              Prospekt
10                        Moscow                    012345   RU+123456                             igor@email.com
30C                                 EE999000065FI                                  PC 1  2017                                8500             8000C
C
400Product AAA                      5 pairs                                  7800    FI12345678    12550             USD
10C                                                 20160201_ 7      201602011323                                                33100            00580
200                    20160201_ 7                                                  FI    0       1
3310000580        3310C
250                                 10699999WebShop Testi Oy                                                                                          Varastoka-
tu 1C                     Tampere                   33100    FI0401234567                          helpdesk@webshop.com
250                                 20      Pyry Simonen                                                                                              Plazankuja
2 B 17                  Helsinki                    00580    FI0407654321                        pyry@gmail.com
300                                 JJFI699999418760000l1                          PC 1  21033101                            150C             20000
FI2350000110000238                123456780                             255C             EUR
OKOYFIHH   C
99919
```