Petteri Antikainen

UPS-kuljetusten tilausautomaation toteutus ja integraatio

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tietotekniikka

Insinöörityö

20.5.2013

Metropolia

Tekijä Otsikko Sivumäärä Aika	Petteri Antikainen UPS-kuljetusten tilausautomaation toteutus ja integraatio 37 sivua 20.5.2013
Tutkinto	insinööri (AMK)
Koulutusohjelma	tietotekniikka
Suuntautumisvaihtoehto	ohjelmistokehitys
Ohjaajat	kehityspäällikkö Misa Vahvelainen yliopettaja Hannu Laine

Insinöörityön tarkoituksena oli korvata asiakasyrityksen vanha UPS-kuljetusten tilausautomaatio nopeammalla ja luotettavammalla. UPS eli Universal Parcel Service on kuljetusyritys, joka toimittaa tavaraa ympäri maailmaa. Asiakkaana oli tekniikan alan huoltoyritys. Automaatio on osa suurempaa järjestelmää, joka pitää kirjaa kaikista yrityksen huoltotapauksista. Automaation tehtävä on tilata tarvittavat UPS kuljetukset, kirjata lokimerkinnät tietokantaan ja tulostaa tarpeelliset rahtikirjat.

Työssä hyödynnettiin UPS:n tarjoamia XML-rajapintoja, joiden avulla tilaukset voidaan tehdä verkon yli. Rajapintojen avulla rakennettiin C#-kirjasto, jonka avulla rajapinnat saatiin integroitua yrityksen loppukäsittelyautomaatioihin.

Järjestelmä saatiin käyttöön määräajassa ilman vanhan järjestelmän ongelmien uusiutumista. Järjestelmää nopeutettiin myös huomattavasti. Joitain pieniä ongelmia huomattiin projektin aikana ja sen päätyttyä, mutta niistä pahimmat saatiin korjattua ja loput pystytään kiertämään.

Näiden kriteerien perusteella projektia voidaan kuitenkin pitää onnistuneena.

Avainsanat	UPS, ohjelmistokehitys, automaatio, kuljetus, XML

Author	Petteri Antikainen
Title	Automation and integration of UPS shipping orders
Number of Pages	37 pages
Date	20 May 2013
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Software Development
Instructors	Misa Vahvelainen, Development Director Hannu Laine, Principal Lecturer

The purpose of this project was to replace the old UPS shipping order system the company was using with a new, faster and more reliable system. UPS, meaning Universal Parcel Service, is a package delivery company that delivers packages of all sizes around the world. XML APIs provided by UPS were used in the project.

Using the APIs, it was possible to build a C# library that could be used to order transportation from UPS. This library could be integrated into the company automations. The system automatically orders shipping from UPS, writes logs of all the cases and prints the necessary labels.

The system was completed without any major problems. Most of the problems of the older systems were avoided in the new one. The new system also appears to be considerably faster. Some minor problems were found during and after the project but the worst of them were fixed and alternative solutions were found for the rest of them. These problems were mostly related to installation of new operating systems and hardware. With these parameters in mind, the project can be considered a success.

Keywords	UPS, Software development, Automation, Delivery, XML

Sisällys

Lyhenteet

API	*Application Programming Interface* ohjelmointirajapinta.
AutoIt	Rakenteeltaan Visual Basicin kaltainen kevyt, mutta samalla monipuolinen, skriptauskieli.
C#	Microsoftin .NET:ia varten kehittämä ohjelmointikieli.
SciTe	Yrityksellä käytetty AutoIt kehitysympäristö.
SOAP	*Simple Object Access Protocol* on XML-pohjainen verkkotoimintojen kuvauskieli.
UPS	*United Parcel Service* on maailmanlaajuisesti toimiva tavaran kuljetusyritys.
WSDL	*Web Services Description Language* on XML-pohjainen verkkopalveluiden kuvauskieli.
XML	*Extensible Mark-up Language* on yleisesti käytetty kuvauskieli.

1 Johdanto

Projektin tarkoituksena on ollut korvata yrityksen vanha UPS (*United Parcel Service* on maailman laajuisesti toimiva tavaran kuljetusyritys) -kuljetustilausjärjestelmä uudemmalla ja paremmin yrityksen tarpeisiin sopivalla sisäisesti toteutettavalla ohjelmistolla.

Asiakkaana oleva yritys, on kasvava huoltoalan yritys, jolla on oma logistiikkaosasto, jonka kautta kaikki yrityksen huoltamat laitteet kulkevat. Yritys käyttää UPS:n lisäksi Itellaa ja omaa kuljetuspalveluaan asiakkaasta riippuen. Yrityksellä on oma automaatio, joka tekee kaikille huoltotapauksille tarvittavat tarkastukset, varmistaen ettei niiden tiedoista löydy virheitä tai puutteita. Tätä automaatiota kutsutaan yrityksessä loppukäsittelyksi. Automaatio päättää myös minne ja kuinka eri laitteet tulee lähettää.

Vanhana järjestelmänä UPS -kuljetusten tilauksessa toimi UPS World Ship-ohjelmisto ja sen käyttöliittymän automaatio. Tämä automaatio jäljittelee ihmisen tapaa käyttää ohjelmaa kirjoittamalla tietoja asianmukaisiin kenttiin ohjelmassa. Automaatioon kuuluu huomattava määrä lyhyitä viiveitä, koska ohjelman pitää antaa käsitellä saamansa tiedot ennen kuin sille voi turvallisesti antaa lisää tietoja. Tämä tarkoittaa teknikolle pahimmillaan reilun minuutin odottelua, jolloin hän ei voi käyttää konetta mihinkään muuhun.

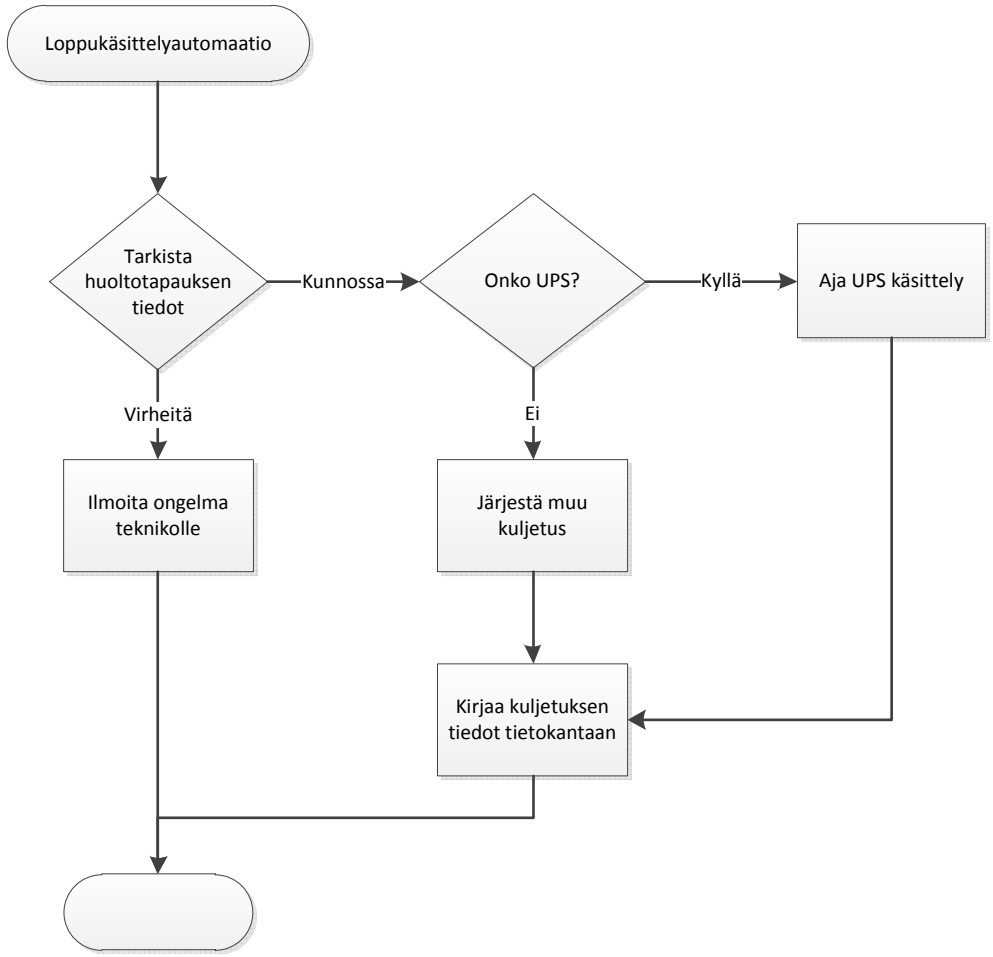
Yrityksellä on ennestään myös pieni sisäinen IT-tuki ja ohjelmistokehitysosasto tukemassa muiden osastojen, kuten huollon ja logistiikan, toimintaa. Yrityksen sisäisesti on toteutettu useita pieniä apuohjelmia ja automaatioita, jotka nopeuttavat ja helpottavat yrityksen asiakaspalvelua ja sisäisiä toimintoja.

2 Huoltotapausten kulku

Yritys huoltaa useita erilaisia laitteita tiloissaan ja toimittaa sitten huoltamansa laitteet takaisin asiakkailleen.

Kun laitteet ensin tulevat yritykselle erinäisten asiakasyritysten kautta, päätyvät ne logistiikkaosastolle. Logistiikkaosastolla jokainen laite kirjataan yrityksen järjestelmään ja

viedään sen jälkeen huolto-osastojen teknikoille huollettavaksi. Teknikko tarkastaa laitteen, tilaa tarpeelliset varaosat ja huoltaa laitteen.



Kuva 1. Loppukäsittelyautomaation vuokaavio.

Huollettuaan laitteen teknikko ajaa yrityksen oman loppukäsittelyautomaation. Automaatio tarkastaa kuvan 1 mukaisesti, että laitteen kaikki tiedot löytyvät yrityksen järjestelmistä eikä laitteeseen ole käytetty vääriä osia. Kun loppukäsittelyautomaatio on saanut tarkastuksensa tehtyä, se tilaa useiden lajittelusääntöjensä perusteella kuljetuksen, jolla laite palautetaan asiakkaalle. Näihin lajittelusääntöihin kuuluu useita eri mahdollisuuksia, jotka automaatio tarkastaa tietyssä järjestyksessä: onko kyseessä yksityis- vai yritysasiakas, mitä lisäpalveluita tapauksella on, minkä merkin ja mallin laite on kyseessä yms.

Tässä vaiheessa automaatio on ennen joutunut UPS:n kuljetuksilla käynnistämään UPS World Ship-ohjelman ja automatisoimaan sen käyttöliittymää. UPS World Ship-ohjelma ja sen käyttöliittymän automaatio käydään tarkemmin läpi luvussa 4.2.

Kun automaatio on saanut kuljetuksen tilattua, antaa teknikko laitteen takaisin logistiikkaosastolle. Sieltä laite lähtee tilatulla kuljetuksella takaisin asiakkaalle.

3 Kehitysympäristö

Yrityksellä on jo kattavat kirjastot yrityksen sisäisesti toteutettuja tukiohjelmistoja, kuten graafisella käyttöliittymällä varustettu viestienvälitysohjelma, jota yritys käyttää asiakkaiden tiedottamiseen laitteiden tilasta yms.

Yrityksen omat ohjelmat on koodattu suurimmalta osin AutoIt- ja C#-kielillä.

AutoIt

”AutoIt v3 on ilmainen Basic -sukuinen skriptauskieli, joka on suunniteltu Windowsin graafisen käyttöliittymän automatisointiin ja muuhun skriptaukseen” [1]. Se sisältää tarvittavat komennot useimpiin Windows -ikkunoiden automaatioihin. Komentoja löytyy mm. kontrollien klikkaamiseen ja tekstin syöttämiseen. AutoIt on hyvä työkalu Windows-ikkunoiden tehostamiseen ja helppokäyttöinen ikkunatietojen infotyökalu helpottaa tätä tehostamista. Työkalu kertoo mm. millä nimellä eri ikkunoita ja niiden kontrolleja voidaan kutsua.

Kieli on tyypitön, eli muuttujien sisällön käsittelytapa päätetään vasta niitä käytettäessä. Tämän takia kielessä on hyvin vähän eroa vaikka string- (merkkijonon) ja integer (tasaluku)-muuttujien käsittelyssä. Koska muuttujia ei tyypitetä mitenkään, on niitä mahdollista käyttää ristiin lähes miten haluaa, mutta se ei ole turvallista. Tämä tarkoittaa sitä, että oli muuttujassa mitä tietoa tahansa sitä voidaan käsitellä, millä komennoilla halutaan. Esimerkiksi voidaan käsitellä merkkijonoa kokonaislukuna. Tästä seuraa omat riskinsä, joihin kehittäjän pitää varautua. Jokaisen muuttujan sisältämä tieto pitää aina tarkastaa ennen muokkaamista. Muuten muuttujien sisältö voi käydä arvaamattomaksi.

Kielessä ei myöskään ole kunnon oliorakenteita, mutta niitäkin voidaan luoda sopivan rajapinnan avulla. Esimerkiksi voidaan kutsua ulkoisia dll kirjaston funktioita COM rajapintaa hyödyntävän objCreate-funktiolla ja sitä kautta käyttää vaikka C#:lla toteutettuja luokkia. Olion luomisen onnistuminen kannattaa aina tarkastaa, jos haluaa välttää ongelmat sitä kutsuttaessa.

Yrityksen AutoIt-ohjelmistojen kehityksessä on käytetty SciTe -nimistä ohjelmaa. SciTe on kevyt ja helppokäyttöinen ympäristö, joka muistuttaa käyttöliittymältään monin tavoin Notepad++-ohjelmaa. Ohjelma on muokattavissa useilla eri lisäpalveluilla, kuten kielikohtaiset koodin kääntäjät ja linkittäjät sekä funktion ja virheentilojen seuranta toiminnot. Keveydestään huolimatta se on täysiverinen kehitysympäristö. Se soveltuu käytettäväksi muidenkin kielien kanssa, mutta olen itse tutustunut vain AutoIt:tä varten valmiiksi muokattuun versioon.

C# -kieli

C# on Anders Hejlsbergin ja Scott Wiltamuthin johtaman tiimin Microsoftilla varta vasten. NET ympäristöä varten kehittämä kieli. Se juontaa juurensa C-, C++- ja Java-kieliin omaksuen kaikkien hyvät ominaisuudet ja lisäten vielä omiaan. C# on oliopohjainen kieli, johon kuuluu tehokas ennalta rakennettu luokkakirjasto, jonka avulla ohjelmoijat voivat kehittää ohjelmia nopeasti [5. s.4].

WSDL – *Web Services Description Language*

WSDL on XML-pohjainen verkkopalveluiden kuvauskieli, jonka avulla voidaan määrittää minkä tahansa verkossa toimivan palvelun toiminta. Nämä kuvaukset ovat niin abstraktissa muodossa, että niistä saadaan rakennettua konekieliset toiminnot verkkopalvelun käyttöön [6]. WSDL määrittää verkkopalvelun täysin. Kaikki palvelun vaatimat parametrit ja sen lähettämä paluusanoma tyyppeineen voidaan ja järjestelmän toiminnan kannalta täytyykin määrittää tarkkaan. WSDL-kielellä määritetyn palvelun suurin etu on mahdollisuus määrittää palvelu palvelimen päässä kerran. Tämän jälkeen siitä voidaan johtaa käyttäjäpäässä lähes mille tahansa ohjelmointikielelle oma rajapintansa ilman, että palvelimen päässä tarvitaan mitään toimenpiteitä. Oli käyttäjäpäässä millä kielellä toteutettu ohjelma tahansa, palvelinpuolella ei näy mitään eroa. XML-sanomat, joita rajapinnan ja palvelimen välisessä kommunikoinnissa käytetään, ovat identtisiä.

Verkkopalveluiden kuvauskielenä WHDL kuvaa juuri verkon kautta lähetettävien kutsujen ja vastausten avulla toimivaa rajapintaa. Rajapinta lähettää HTML-dokumentin kohteeseensa ja palauttaa saamansa vastauksen käyttäjälle.

Muut automaation osat

UPS World Ship-ohjelma on osa yrityksen loppukäsittelyautomaatiota ja se käynnistetään automaattisesti, kun huolletun laitteen tiedot sitä vaativat, luvun 2 mukaisesti. Automaatio suorittaa useita erilaisia tarkastuksia ennen asianmukaisen kuljetuksen tilaamista.

Koska yrityksellä ei ole ollut UPS World Ship-ohjelmaan käytössä suoraa rajapintaa, käyttöliittymäautomaatio on ollut tähän asti ainoa vaihtoehto. Käyttöliittymäautomaatio tarkoittaa tässä tapauksessa sitä, että automaatio jäljittelee ihmisen tapaa käyttää ohjelmaa. Se käy hakemassa huoltotapauksen tarvittavat tiedot tietokannasta ja syöttää ne asianmukaisiin kenttiin ohjelmassa.

4 Järjestelmä ennen muutoksia

4.1 Järjestelmän kuvaus

Yrityksellä on jo olemassa käytännössä toimiva kuljetustilausjärjestelmä. Siinä käytetään UPS World Ship-ohjelmaa ja tilataan käyttöliittymäautomaatiolla kuljetus yrityksen logistiikkaosastolta asiakkaalle.

4.2 Kuljetusten tilausohjelmisto

Ohjelman kuvaus

UPS World Ship on kattava ja monipuolinen ohjelmisto, mutta se kärsii muutamasta pienestä ongelmasta, joita ei manuaalisessa käytössä edes huomaa. Ikävä kyllä ongelmat vaikuttavat sitäkin enemmän automaatioon. Suurin ongelma automaation kannalta on se, miten kauan ohjelmalla kestää käsitellä saamansa tiedot, ennen kuin se on valmis vastaanottamaan lisää tietoja.

Automaatio vaatii vähän huoltoa joidenkin UPS World Ship-ohjelman päivitysten jälkeen. Ohjelma tarkastaa jokaisen sille annetun tiedon aina näitä tietoja saadessaan. Yleisin ongelma onkin jonkin näistä tarkastuksista hidastuminen, mikä vaatii asian mukaisen viiveen pidentämistä. Yksittäisinä nämä viiveet eivät ole ongelma, mutta niitä kertyy ajan myötä niin paljon, että suurin osa loppukäsittelyyn menevästä ajasta kuluu näiden viiveiden odotteluun. Läheskään kaikki päivitykset eivät hidasta ohjelmaa, mutta hidastavia päivityksiäkin tulee joskus.

Toinen hankalampi ongelma on tapa, jolla ohjelma päivittää itsensä. Ohjelma tarkastaa saatavilla olevat päivitykset aina käynnistyessään ja asentaa kaiken mahdollisen automaattisesti. Yleensä tämä ei ole ongelma vaan juuri näin nykyaikaisen ohjelman kuuluukin toimia. Ongelmana on siinä, ettei automaatio voi tarkastaa, onko ohjelma jo valmis vai päivittääkö se vielä itseään, eikä se ole millään tavoin käytettävissä päivitysten aikana, koska ohjelma asentaa kaikki päivityksensä ennen käynnistyksen viimeistelyä. Ennen kuin päivitys on valmis, ohjelma ei edes avaa pääikkunaansa. Kun näitä päivityksiä tulee viikoittain ja jokainen päivitys kestää muutaman minuutin ja sammuttaa samalla automaation, on ongelma jo aika selkeä. Suurin osa päivityksistä on käsittääkseni lähinnä polttoaineen hinnan muutosten aiheuttamia korjauksia ohjelman sisäiseen hinnoitteluun. Normaalikäytössä on hyvä nähdä mahdollisimman tarkka arvio kuljetuksen hinnasta, mutta yrityksen käyttöön se lisäsi päivitystiheyttä aivan liikaa.

Automaation suorittamat toimenpiteet

Automaatio syöttää jokaisella ajokerralla useita tietoja ohjelmaan. Se hakee ensin yrityksen tietokannasta huoltotapauksen tiedot ja alkaa sitten täyttää ohjelmaan yrityksen ja asiakkaan osoite- ja yhteystietoja. Kun yhteystiedot on riittävällä tarkkuudella määritetty ohjelmaan, alkaa ohjelma täyttää kuljetettavan paketin tietoja.

Jokaiseen ohjelmaan syötettävään tietoon on lisättävä pieni viive (100–500 ms), jotta ohjelma ehtii käsitellä saamansa tiedot ja tarkastaa, mitä muutoksia annetut tiedot aiheuttavat muihin tietoihin. Jos ohjelmalle ei anna aikaa käsitellä tietoja, voi automaatio käydä epävakaaksi, koska se ei saa tarvitsemiaan kenttiä muokattua ohjelmassa. Yksittäin nämä viiveet eivät ole ongelma, mutta niistä kertyy varsin nopeasti noin minuutin mittainen tauko automaation ajokertaa kohden ennen seuraavaan huoltotapaukseen siirtymistä. Minuuttiin lisätään vielä muutama sekunti kaikista tarkastuksista ja muutama lisää ohjelman käynnistämiseen, jonka automaatio suorittaa ennen UPS World

Ship-ohjelmiston käyttämistä. Kaiken tämän jälkeen ohjelman voi jo sanoa hidastavan työntekoa. Automaatio varaa koneen, eikä teknikko sen takia kykene katsomaan edes seuraavan huoltotapauksen tietoja.

Automaatiota suoritetaan jokaiselle huoltotapaukselle. Jokainen teknikko käsittelee useita huoltotapauksia päivittäin, joten aikaa kuluu odotteluun suhteellisen runsaasti.

Ohjelma-arvio

Kaiken kaikkiaan ohjelma on erinomainen yksittäisten kuljetusten manuaaliseen tilaamiseen, mutta hitaan käynnistyksen, häiritsevien päivitysten ja tietojen käsittelyviiveiden takia se ei sovellu yrityksen tarpeisiin.

Ohjelma myös tarjoaa työkalut kuljetustilausten peruuttamiseen ennen, kuin ne on noudettu yrityksen tiloista.

5 Järjestelmän vaatimukset

Järjestelmän täytyy kyetä automaattisesti tilaamaan UPS:ltä kuljetus ja tulostaa UPS:n rahtikirja. Tarvitaan myös menetelmä kuljetusten tilanteen tarkkailuun ja tarpeen tullen myös peruuttamiseen.

Järjestelmää on myös nopeutettava ja vakautettava huomattavasti. Luvun 4 ohjelmakuvauksen pohjalta voidaan kuitenkin sanoa, ettei UPS World Ship-ohjelman käyttöliittymän automatisointi sovellu tähän. Ratkaisuna oli ottaa käyttöön UPS:n rajapinnat ja rakentaa yrityksen käyttöön paremmin soveltuva järjestelmä.

6 UPS rajapinta ja tuki

UPS tarjoaa kattavaa ja helppokäyttöistä rajapintaa kaikille sitä tarvitseville. Nämä rajapinnat ovat WSDL (*Web Services Description Language*)-kielellä kirjoitettuja kuvauksia. WSDL on XML-pohjainen verkkopalveluiden kuvauskieli, jolla kirjoitetuista kuvauksista saa useimmilla nykyaikaisilla ohjelmointiympäristöillä suhteellisen vaivattomasti rakennettua luokkakirjaston.

Rajapinnan saamiseksi riittää kirjautuminen UPS:n verkkosivuille. Sen löytää teknisestä tuesta. Rajapinnan käyttämiseen tarvitaan myös käyttäjäkoodeja, jotka saa ottamalla yhteyttä UPS:ään. Projektiin osallistunut UPS:n edustaja oli erittäin avulias, mikä nopeutti projektin etenemistä.

UPS tarjoaa muutaman rajapinnan, mutta yrityksen tarpeisiin riitti lähetys-, seuranta- ja poistorajapinnat. Kaikki nämä rajapinnat voidaan jakaa kolmeen pääluokkaan, jotka ovat palvelu (lähetyksessä ShipService), pyyntö (lähetyksessä ShipmentRequest) ja vastaus (lähetyksessä ShipmentResponse). Näistä palveluluokka sisältää varsinaisen toiminnallisuuden, pyyntöluokka sisältää tiedon siitä, mitä käyttäjä UPS:ltä haluaa ja vastausluokka määrittää nimensä mukaisesti vastauksen muodon.

Rajapinnan hyödyntäminen on yksinkertaista. Microsoft Visual Studio ja monet muutkin nykyaikaiset ohjelmistot pystyvät muodostamaan toimivan luokkakirjaston WHDL-koodista. Kun luokkakirjasto on muodostettu, kehittäjän tarvitsee vain täyttää asianmukaiset tiedot oikeisiin olioihin ja kutsua sitten palveluluokasta luodun olion metodia, joka muodostaa saamistaan tiedoista XML-dokumentin ja lähettää sen määritettyyn osoitteeseen jääden sitten odottamaan vastausta.

Rajapinnat myös toiminta on myös yksinkertaista, vaikka sen vaatimien tietojen määrä saattaakin ensin ihmetyttää. Kaikki työssä käytetyt UPS:n rajapinnat toimivat samalla tavalla. Rajapinnan pyyntöluokkaan täytetään tarpeelliset tiedot, että UPS:n palvelimet tietävät, mitä käyttäjä tarvitsee. Palveluluokka sisältää varsinaisen toiminnallisuuden mukaan lukien metodit, joilla ensin muodostetaan XML-kysely pyyntöluokan sisältämistä tiedoista ja sitten lähettää tämä kysely UPS:n palvelimille. UPS:n palvelimet vastaavat lähettämällä vastausluokan mukaisen olion, josta käyttäjä voi etsiä haluamansa tiedot.

UPS myös tarjoaa jokaiselle rajapinnalle testipalvelimen, etteivät kehitysvaiheen alussa kohdattavat ongelmat ja virheet kuormita tai kaada varsinaisia tuotantopalvelimia. Nämä testipalvelimet antavat tuotantopalvelimiin verrattuna lähes identtiset palautteet käyttäjälle. Ainoat erot ovat rahtikirjan viivakoodi, jonka päälle testipalvelin lisää tekstiä, eikä testipalvelimelle tulleita tapauksia kirjata UPS:n palvelimelle tilauksina.

Seuraavaksi käydään läpi yrityksen järjestelmiin tarvitut osiot lähetysrajapinnasta.

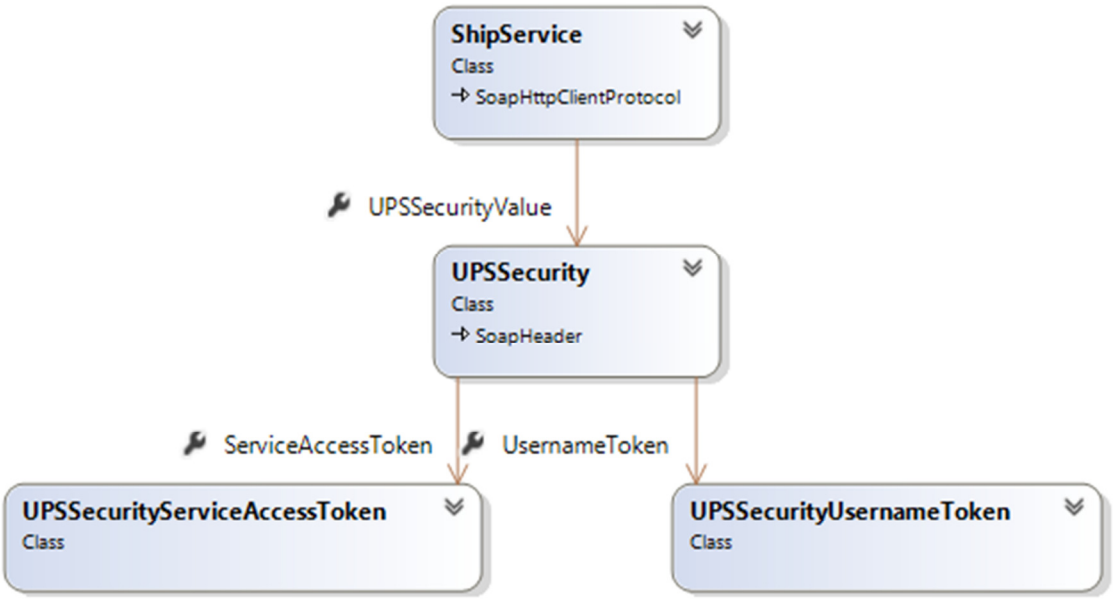
Lähetysrajapinta oli ensimmäinen ja ehdottomasti järjestelmän kannalta tärkein rajapinta. Se on yrityksen toiminnan kannalta kriittisin, ja sen toimintaa tarkkailemalla selvitettiin myös muiden rajapintojen toimintaa.

Lähetysrajapinnan avulla tilataan pakettien kuljetus paikasta toiseen. Sen luokat sisältävät satoja muuttujia, joista rajapinnan metodit muodostavat kutsuttaessa XML-tiedostoja. Rajapinta käyttää näitä XML -tiedostoja UPS:n palvelimien kanssa keskusteluun.

Rajapinnasta on syytä mainita yksi asia. Jos jotain kenttää ei tarvita, sitä ei pidä edes alustaa. Muutama kenttä on sellaisia, joihin ei sijoiteta mitään tietoa, vaan palvelin reagoi jo niiden olemassaoloon.

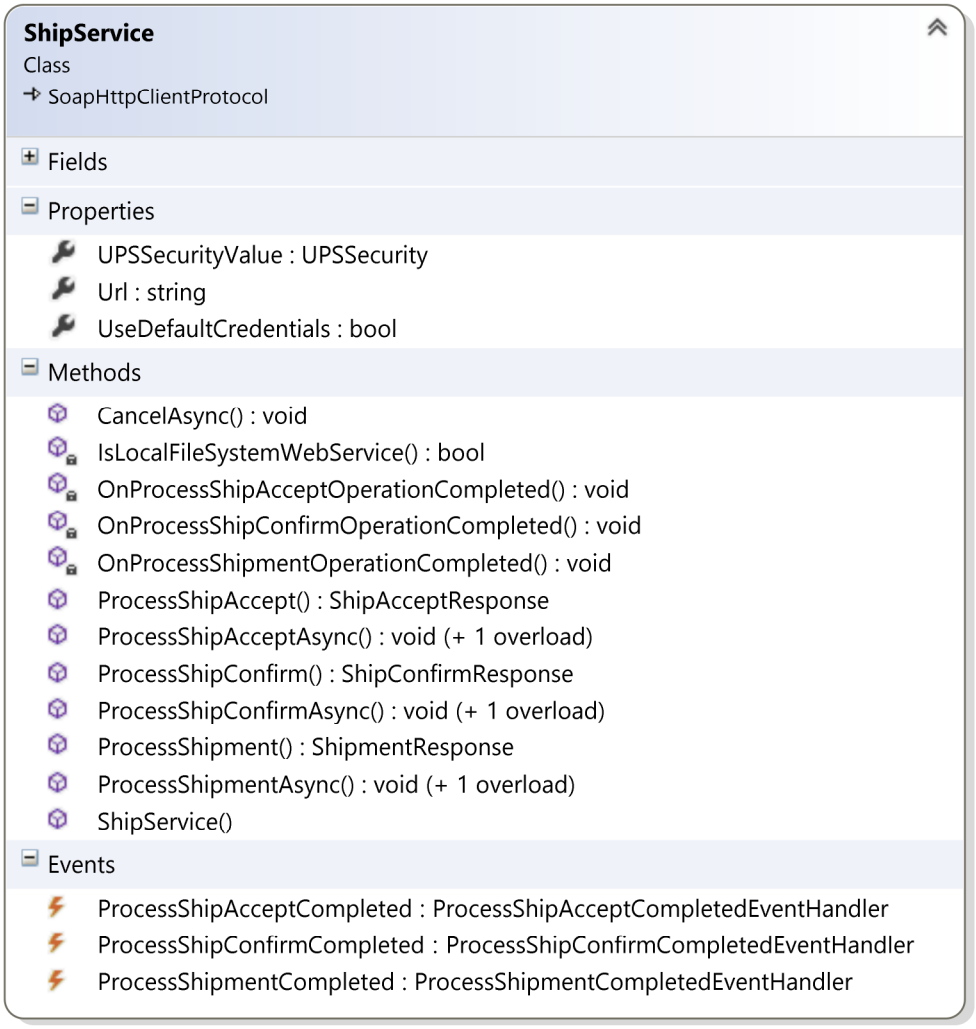
Aloitan rajapinnan tarpeellisten luokkien esittelyn ylimmältä tasolta palveluluokasta lähtien ja käyn niitä läpi aina niin syvälle kuin tarvitaan, ennen kuin siirryn seuraavaan haaraan. Käyn läpi vain ne osiot, joita tarvittiin yrityksen haluaman järjestelmän toteutukseen, koska rajapinnat sisältävät niin paljon eri mahdollisuuksia, ettei niitä kaikki kannata tässä lähteä erittelemään.

6.1 Palveluluokka



Kuva 2. Palveluluokan kokonaiskuva.

Kuvassa 2 on kuvattuna lähetysrajapinnan palveluluokka ShipService ja muut sen käyttämät luokat, sisältävät toiminnallisuuden, turvallisuusrakenteet ja tiedot siitä, mihin pyynnöt lähetetään.

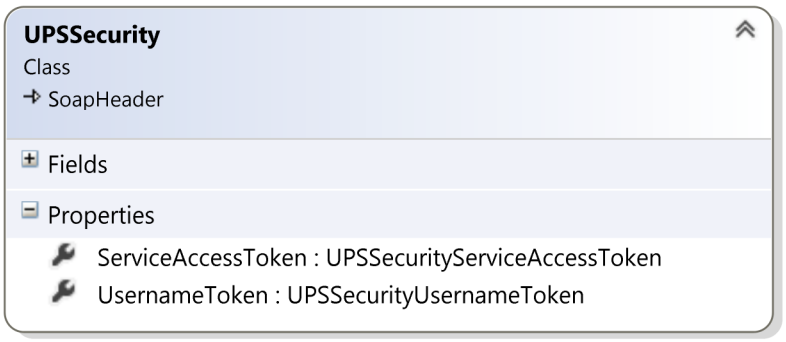


Kuva 3. ShipService-luokka.

Kuvassa 3 on ShipService, joka on lähetysrajapinnan toiminnallisuuden ydin. Se sitoo yhteen muut rajapinnan rakenteet ja sisältää varsinaisen toiminnallisuuden, kuten kuvan metodilistasta näkee. Url -kenttään asetetaan käytettävän palvelimen osoite. Tähän osoitteeseen rajapinta lähettää HTML-kyselyyn pakatun XML-koodinsa. Tämä osoitteen muutos mahdollistaa UPS:n tarjoamien testipalvelimien käytön kehitystyön alkuvaiheessa ja varmistavat testiajot ilman laskutusta myöhemmin.

uPSSecurityValueField sisältää tunnukset ja salasanat, joilla UPS varmentaa asiakkaansa.

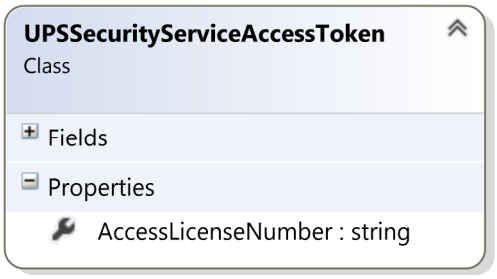
ProcessShipment -metodi lähettää sille annetun ShipmentRequest -olion ja antaa paluuarvona ShipmentResponse-olion, jossa on kaikki tiedot kuljetustilauksesta.



Kuva 4. UPSSecurity-luokka.

Kuvassa 4 oleva UPSSecurity on varmennus ja kirjautumistietojen asettamiseen tarkoitettu luokka. Sen sisältämät oliot tarvitaan, että UPS tietää, kuka on tilaamassa kuljetuksen. UserNameToken sisältää käyttäjätunnuksen ja salasanan UPS:n järjestelmiin ja ServiceAccessToken sisältää pitkähkön UPS:ltä saadun merkkijonon, jota tarvitaan rajapintaa käyttävän ohjelman varmentamiseen. Ilman näitä tietoja UPS ei hyväksy kuljetuspyyntöäsi.

Rajapinnassa UPSSecurity sijaitsee ShipService-luokan UPSSecurityValue-kentässä.

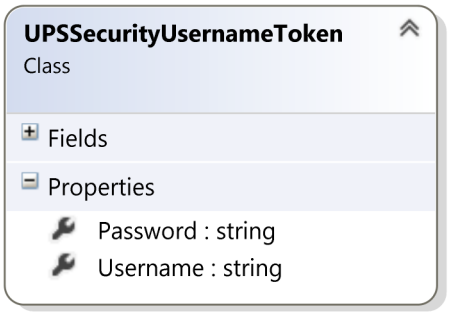


Kuva 5. UPSSecurityServiceAccessToken-luokka.

UPSSecurityServiceAccessToken sisältää kuvan 5 mukaisesti vain yhden erittäin tärkeän varmistusarvon. Ilman AccessLicenseNumber-kenttää on turha yrittääkään käyt-

tää palvelua. Ilman kenttään annettua arvo UPS ei hyväksy mitään pyyntöjä, koska se arvo yhdistettynä käyttäjätunnuksen ja salasanan kanssa varmentaa UPS:n silmissä pyytäjän henkilöllisyyden.

Rajapinnassa UPSSecurityServiceAccessToken sijaitsee UPSSecurity-luokan ServiceAccessToken-kentässä.

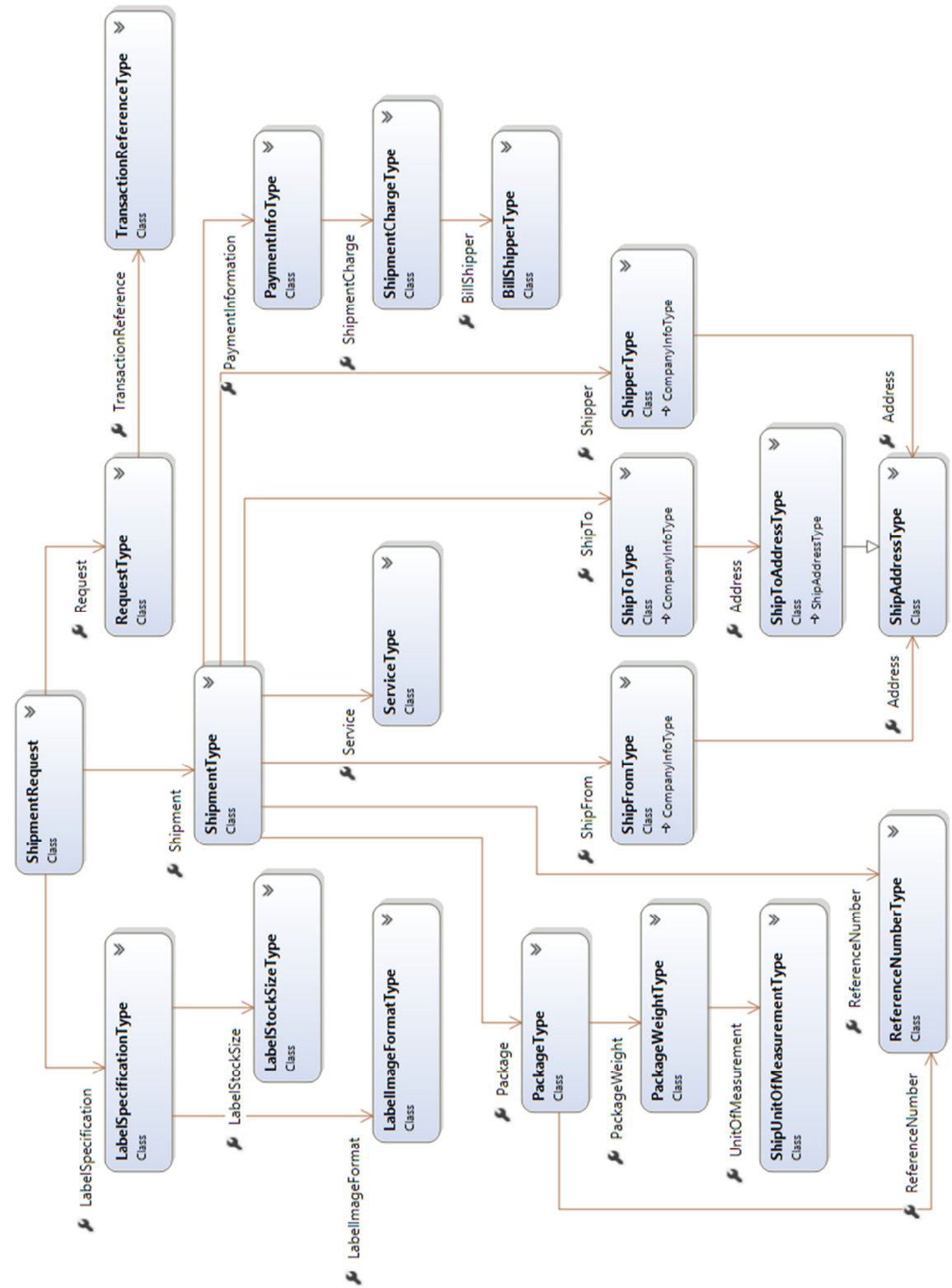


Kuva 6. UPSSecurityUsernameToken-luokka.

Kuvassa 6 näkyvä UPSSecurityUsernameToken sisältää palvelussa tunnistautumisen kannalta kriittiset käyttäjätunnuksen ja salasanan. Nämä ovat käytännössä samat kuin ne joilla voit kirjautua UPS:n sivuille.

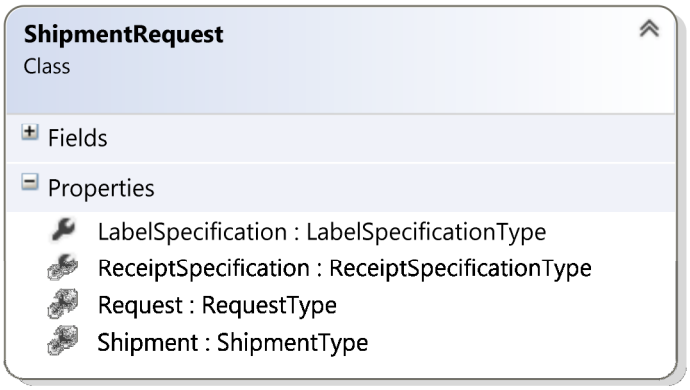
Rajapinnassa UPSSecurityUsernameToken sijaitsee UPSSecurity-luokan UsernameToken-kentässä.

6.2 Pyyntöluokka



Kuva 7. Pyyntöluokan kokonaiskuva.

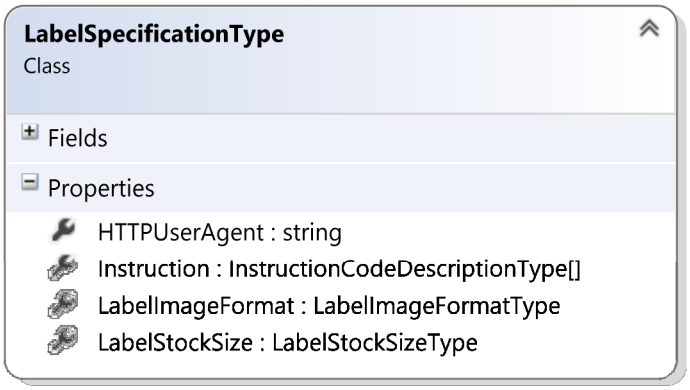
Pyyntöluokka, eli tilausrajapinnassa kuvan 7 mukaisesti ShipmentRequest, sisältää tilattavan kuljetuksen tiedot.



Kuva 8. ShipmentRequest–luokka.

ShipmentRequest sisältää varsinaisen kuljetustilauksen tiedot. Siihen on koottu kaikki rahtikirjan muodosta kuljetus- ja maksutapaan ja maksajaan.

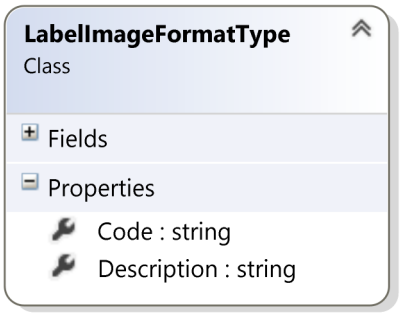
Rajapintaa käytettäessä ShipmentRequest-olio annetaan ShipService-olion ProcessShipment-metodin parametriksi.



Kuva 9. LabelSpecificationType –luokka.

Kuvan 9 LabelSpecicationType -luokasta, joka määrittää rahtikirjan tyypin ja koon, ei tähän projektiin tarvinnut määrittää kuin rahtikirjan koodaustapa ja koko. Koska rajapinta tukee yrityksen Zebra-mallisia tarratulostimia, päätettiin käyttää niiden omaa EPL-tulostinkieltä. Zebra-tulostimet eivät ole ainoita rajapinnan tukemia tulostimia vaan yksi esimerkki.

Rajapinnassa LabelSpecificationType-luokka sijaitsee ShipmentRequest-luokan LabelSpecification-kentässä.

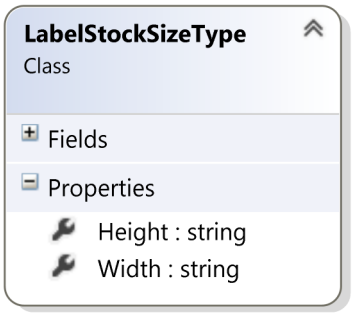


Kuva 10. LabelImageFormatType-luokka.

Kuvan 10 LabelImageFormatType-luokan avulla määritetään paluusanomassa tulevan rahtikirjan koodikieli. Rahtikirjan koodausmuoto pitää määrittää asettamalla sen lyhenne Code-kenttään.

Rajapinta tukee joitain tulostinmerkkejä ja -malleja. Tukemilleen tulostimille rajapinta voi lähettää rahtikirjan kuvauksen tulostimen omalla kuvauskielellä, mikä nopeuttaa ja tarkentaa näin rahtikirjan tulostamista. Tuki löytyy yrityksen käyttämille Zebra-tulostimille.

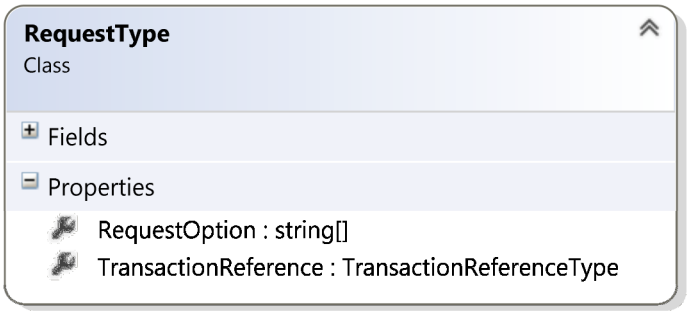
Rajapinnassa LabelImageFormatType sijaitsee LabelSpecificationType-luokan LabelImageFormat-kentästä.



Kuva 11. LabelStockSizeType-luokka.

Kuvan 11 LabelStockSizeType-luokka määrittää rahtikirjan fyysisen koon tuumina. Sen avulla voidaan rajata UPS:n palvelimilta saatavan rahtikirjan koko sopivaksi käytössä olevalle tulostimelle.

Rajapinnassa LabelStockSizeType sijaitsee LabelSpecificationType-luokan LabelStockSize-kentässä.

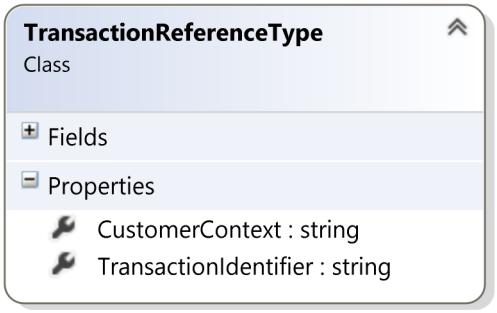


Kuva 12. RequestType-luokka.

Kuvan 12 RequestType-luokka sisältää pyynnön tyypin ja erikoistapausten määrittelyt. RequestOption-tauluun laitetaan mahdolliset tiedot osoitteen tarkistustasosta. Sallittuja arvoja on ”blank”, ”validate” ja ”nonvalidate”. Mikään muu kuin arvo kuin ”nonvalidate” ja UPS:n palvelin lähettää virheilmoituksen takaisin käyttäjälle, jos osoite ei heidän tarkastuksissaan kelpaa. Yleensä arvoksi voi antaa ”nonvalidate”, koska osoitteen oikeellisuus on aina käyttäjän vastuulla.

TransactionReference on tarkoitettu helpottamaan tilausten erottamista toisistaan sekä palvelin että loppukäyttäjäpäässä.

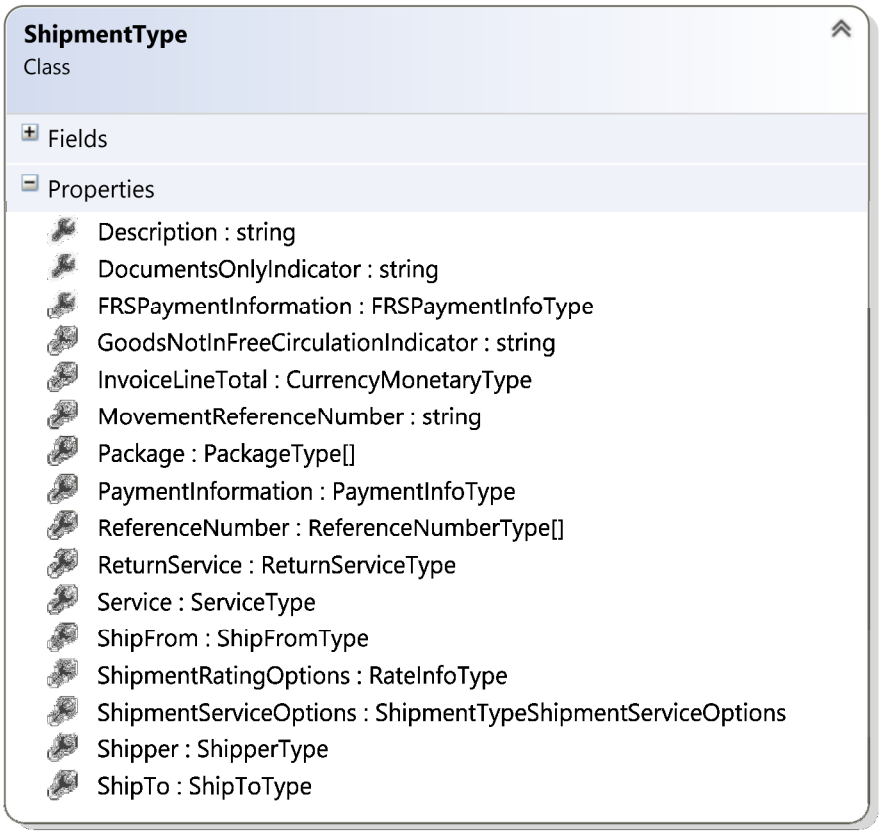
Rajapinnassa RequestType sijaitsee ShipmentRequest-luokan Request-kentästä.



Kuva 13. TransactionReferenceType-luokka.

Kuvan 13 TransactionReferenceType-luokka auttaa yksilöimään kuljetuksia. Vain CustomerContext-kenttä tarvitaan, eikä siihen sijoitettavalla merkkijonolla ole mitään erityisiä rajoitteita. Ainoa rajoite on, ettei kenttään mahdu yli 512:ta merkkiä. Rajapinta käyttää CustomerContext-kenttää pyynnön ja vastauksen synkronointiin.

Rajapinnassa TransactionRaferenceType sijaitsee RequestType-luokan TransactionReference-kentässä.

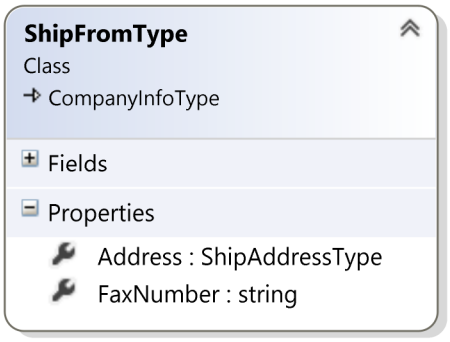


Kuva 14. ShipmentType-luokka.

Kuvan 14 ShipmentType-luokkaan kootaan kaikki kuljetettavan paketin tiedot. Kenttiin voidaan sisällyttää paljon erilaisia tietoja, tärkeimpinä mm. lähtö- ja kohdeosoite ja maksaja. Kaikki UPS:n paketin fyysistä perille saamista varten tarvittavat tiedot sisälletään tähän luokkaan.

Käytännössä luokasta tarvitaan yhden paketin kotimaankuljetuksiin vähintään ShipFrom-, Shipper-, ShipTo-, PaymentInformation- ja Package-kentät.

Rajapinnassa ShipmentType sijaitsee ShipmentRequest-luokan Shipment-kentässä.

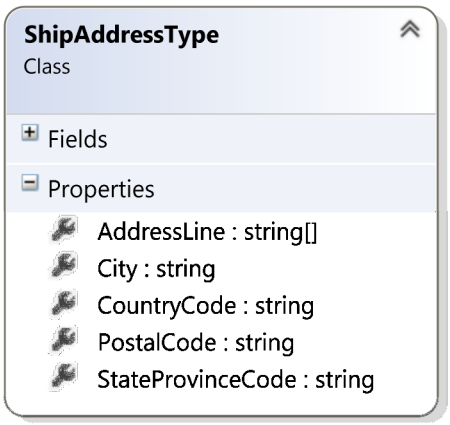


Kuva 15. ShipFromType-luokka.

Kuvan 15 ShipFromType-luokka sisältää nimensä mukaan lähetysosoitteen tiedot. Näistä vain Address- ja tässä näkymättömissä olevat Name- ja Phone-kentät ovat pakollisia. Phone-kenttäkään ei ole pakollinen, ellei käytä UPS Next Day Air Early A.M. -palvelua tai jos lähtö- ja kohdeosoitteet ovat eri maissa.

Name-kenttään laitetaan lähettäjän nimi (yleensä lähettävän yrityksen). Phone -kentän vaatimukset löytyvät PhoneType-luokan kohdalta.

Rajapinnassa ShipFromType sijaitsee ShipmentRequest-luokan ShipFrom-kentässä.



Kuva 16. ShipAddressType-luokka.

Kuvan 16 ShipAddressType-luokka on käytössä lähettäjä-, kohde- ja laskutusosoitteen määrittämisessä. Tästä luokasta ainoa, joka kannattaa jättää määrittämättä, on StateProvinceCode-kenttä. Siihen ei pitäisi kotimaankuljetuksissa olla mitään syytä koskea.

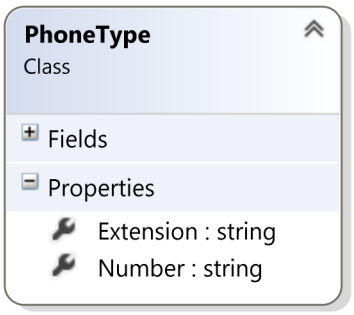
City-kenttään kirjataan kaupungin nimi. Kenttään saa kirjoittaa enimmillään 30 merkkiä, mutta rahtikirjaan tulee näkyviin vain ensimmäiset 15.

CountryCode-kenttään asetetaan viralliset kaksi merkkiset valtion nimi lyhenteen. Eli suomen tapauksessa ”FI”.

PostalCode-kenttään asetetaan postinumero. Kenttään mahtuu enimmillään yhdeksän merkkiä.

Kannattaa huomioida se, miten rajapinta sallii useamman rivin osoitteet. AddressLine-kenttä on String-taulu. Jokainen merkkijono saa olla enimmillään 35 merkkiä pitkä, mutta vain ensimmäisen merkkijonon ensimmäiset 30 merkkiä näkyvät rahtikirjassa.

Rajapinnassa ShipAddressType sijaitsee ShipFromType-, ShipperType- ja ShipToType-luokkien Address-kentissä.



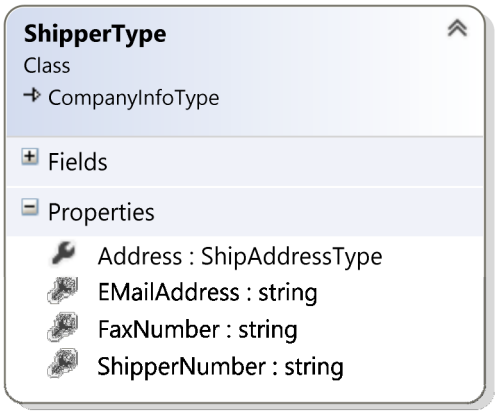
Kuva 17. PhoneType–luokka.

Kuvan 17 PhoneType-luokka sisältää nimensä mukaisesti puhelinnumeron. Numero on jaettu kahteen osaan. Kumpaankaan kenttään ei saa laittaa mitään muuta kuin numeroita. Erikoismerkit aiheuttavat UPS:n palvelimilla virheen, joka estää kuljetustilausta menemästä läpi.

Number-kenttään mahtuu enimmillään 15 merkkiä. Extension-kenttää ei tarkasteta millään tavalla. Sitä ei ole edes pakko käyttää, jos saa puhelinnumeron kokonaisuudessaan mahtumaan Number-kenttään.

PhoneType-luokka ei ole yleensä pakollinen, mutta sen käyttö on suositeltavaa erinäisten kuljetusten ongelmatilanteiden ratkaisemisen kannalta.

Rajapinnassa PhoneType sijaitsee ShipFromType-, ShipperType- ja ShipToType-luokkien Phone-kentässä. Ikävä kyllä Visual Studio ei näytä Phone-kenttää minkään näistä luokista luokkakaaviossa.



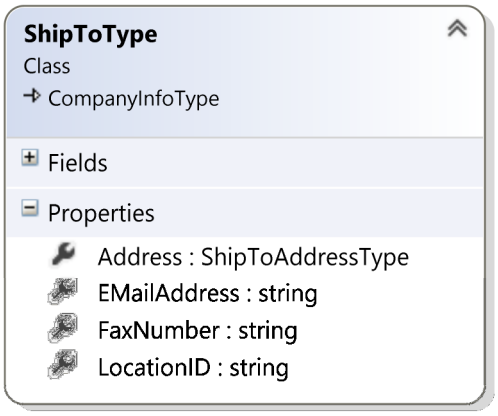
Kuva 18. ShipperType-luokka.

Kuvan 18 ShipperType-luokka sisältää lähettäjän laskutustiedot osoitetta myöten. Laskutusosoite pitää määrittää erikseen, koska paketin lähetysosoite ei ole aina sama kuin maksajan laskutusosoite.

Osoitteen lisäksi tarvitaan myös Name- ja ShipperNumber-kentät, joita ilman UPS ei hyväksy tilausta. Name-kenttään laitetaan lähettäjän nimi (yleensä lähettävän yrityksen) ja ShipperNumber on kuusimerkkinen tunnistenumero. Tunnistenumero on UPS:n määrittämä tunniste, joka on sidottu käyttäjätunnukseen. Tämä on erillinen tunniste, koska näitä tunnisteita voi olla samalla tilillä useita.

Luokkaan kuuluu myös Phone -kenttä, joka ei ole pakollinen.

Rajapinnassa ShipperType sijaitsee ShipmentType-luokan Shipper-kentästä.

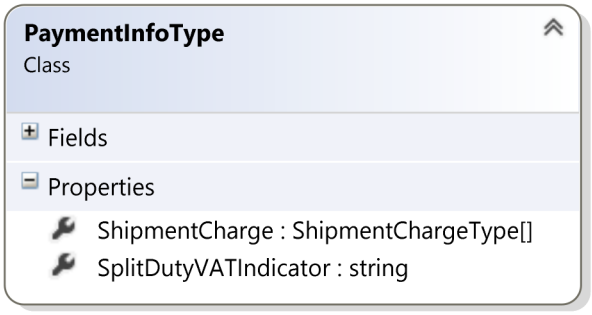


Kuva 19. ShipToType-luokka.

Kuvan 19 ShipToType-luokka sisältää lähetettävän paketin vastaanottajan tiedot. Näistä tarvitaan Address-kenttä ja tässä näkymättömissä oleva Name-kenttä.

Kuten ShipperType- ja ShipFromType-luokista, myös täältä löytyy Phone-kenttä, vaikkei olekaan kuvassa näkyvissä. Etenkin tässä luokassa kenttää kannattaa kuitenkin täyttää, koska se helpottaa UPS:n kuljetushenkilöstön toimintaa. Heidän on huomattavasti helpompi toimittaa paketti perille, jos he voivat sopia loppuasiakkaan kanssa toimitus ajasta.

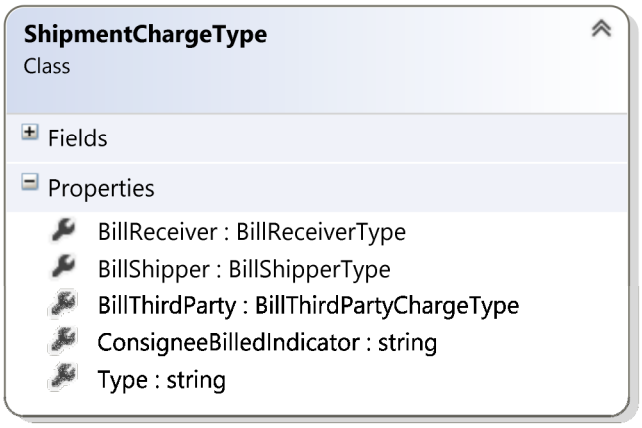
Rajapinnassa ShipToType sijaitsee ShipmentType-luokan ShipTo-kentässä.



Kuva 20. PaymentInfoType-luokka.

Kuvan 20 PaymentInfoType-luokka sisältää tiedot maksajista. Jos tätä ei määritetä oikein, UPS laskuttaa vastaanottajaa toimittaessaan paketin. Yritys on kuitenkin huomioinut nämä toimitukset hinnoittelussaan, joten asiakkaan laskuttaminen kuljetuksesta toistamiseen ei ole hyvää mainosta.

Rajapinnassa PaymentInfoType sijaitsee ShipmentType -luokan PaymentInformation -kentässä.

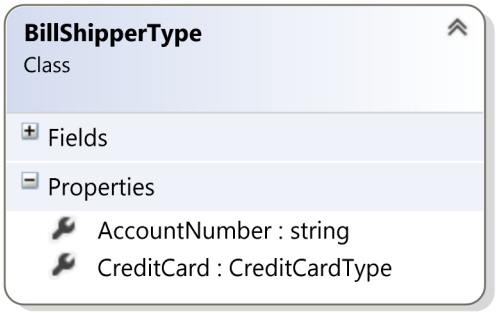


Kuva 21. ShipmentChargeType-luokka.

Kuvan 21 ShipmentChargeType-luokka sisältää tarkemmat tiedot maksun luonteesta. Type-kenttään on annettava joko arvo ”01” eli kuljetus tai arvo ”02” eli tullit ja verot.

Kentistä BillReceiver, BillShipper, BillThirdParty ja ConsigneeBilledIndicator tulee valita yksi. Näillä kentillä määritetään, ketä laskutetaan. Yrityksen käytössä voitiin poikkeuksetta käyttää BillShipper-kenttää, jolloin kuljetus laskutetaan lähettäjältä, joka tässä tapauksessa on yritys itse.

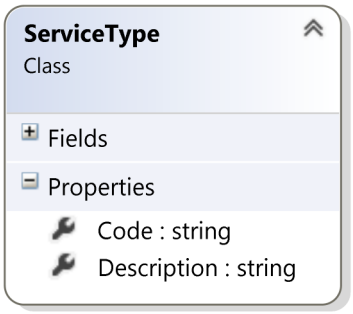
Rajapinnassa ShipmentChargeType sijaitsee PaymentInfoType-luokan ShipmentCharge-kentässä.



Kuva 22. BillShipperType –luokka.

Kuvan 22 BillShipperType-luokka tarjoaa lähettäjän laskutukseen kaksi vaihtoehtoa, tilinumero tai luottokortti. Tilinumeron täytyy olla sama kuin ShipperType-luokan ShipperNumber-kentässä.

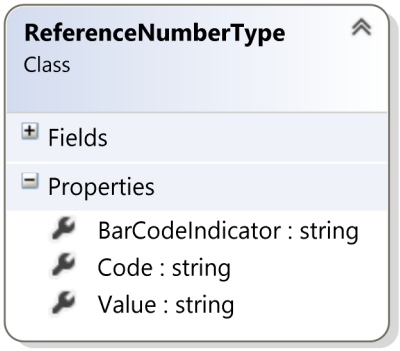
Rajapinnassa BillShipperType sijaitsee ShipmentChargeType-luokan BillShipper-kentästä.



Kuva 23. ServiceType-luokka.

Kuvan 23 ServiceType-luokka määrittää UPS:ltä halutun kuljetustyypin. Kuljetustyyppi määritetään asettamalla Code-kenttään kahden merkin mittainen koodi. Yrityksen käyttöön riitti koodi ”65”, joka tarkoittaa ”UPS Saver” kuljetusta. UPS Saver kuljetus on yleiskäyttöinen peruskuljetus.

Rajapinnassa ServiceType sijaitsee ShipmentType-luokan Service-kentässä.



Kuva 24. ReferenceNumberType-luokka.

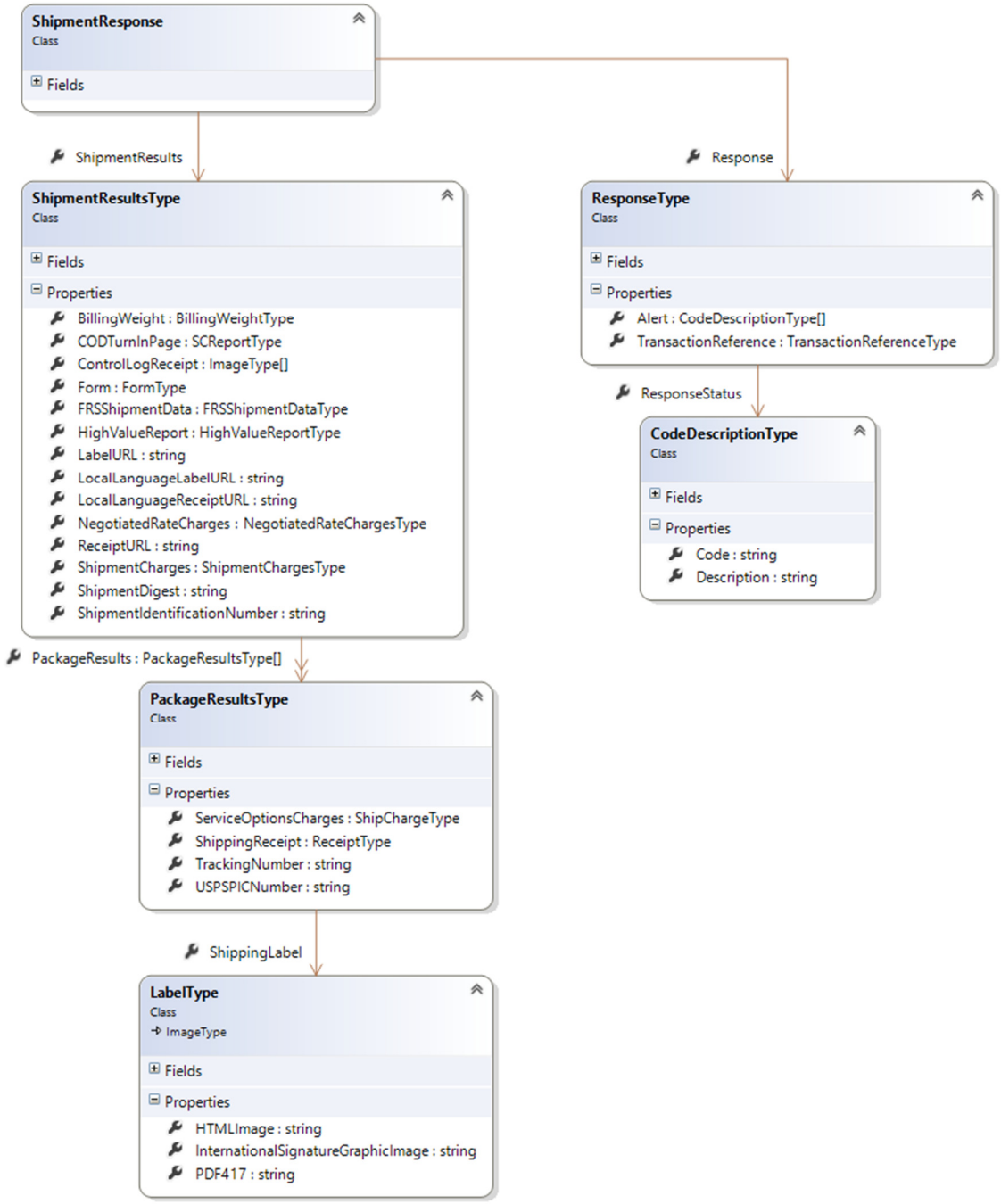
Kuvan 24 ReferenceNumberType-luokassa voidaan antaa tunnistetiedot paketille. Näiden tunnisteiden avulla vastaanottaja ja lähettäjä tietävät tarkalleen, mikä paketti on

kyseessä. Value-kenttään voidaan asettaa enimmillään 35-merkkinen merkkijono erottamaan paketti muista.

Rajapinnassa ReferenceNumberType sijaitsee ShipmentType-luokan ReferenceNumber-kentästä.

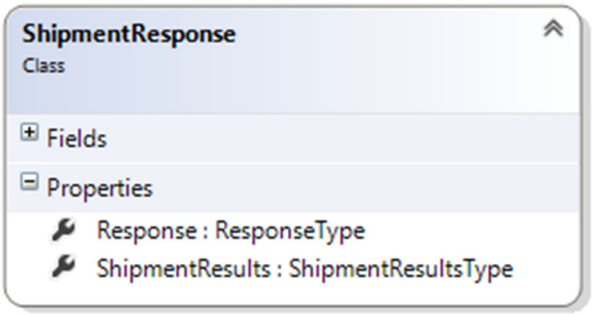
6.3 Vastausluokka

Vastausluokan tarpeellisten osien määrittely on hieman hankalampaa. Luokan olio saadaan aina UPS:n palvelimilta ja oliosta otetaan ne tiedot, joita kulloinkin tarvitaan.



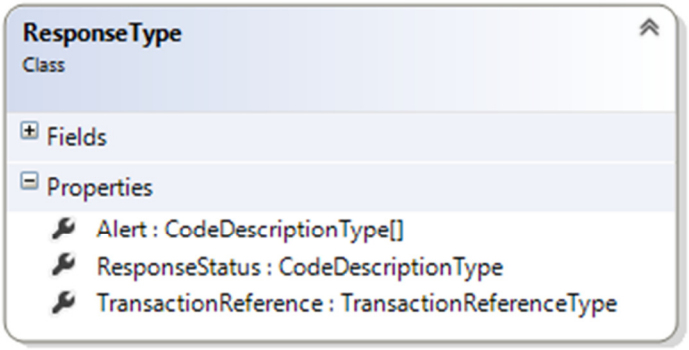
Kuva 25. Vastausluokan kokonaiskuva.

Kuvan 25 vastausluokasta löytyy huomattava määrä tietoja, mutta aivan ensimmäisenä sieltä tarvitaan tieto tilauksen onnistumisesta. Onnistumisen varmistamisen jälkeen voidaan etsiä tarkemmat tiedot. Tärkeimpinä tietoina mainittakoon seurantanumero ja rahtikirjan kuvaus.



Kuva 26. ShipmentResponse-luokka.

Kuvan 26 ShipmentResponse sisältää ShipService-luokan processShipment-metodin palauttaman vastauksen.

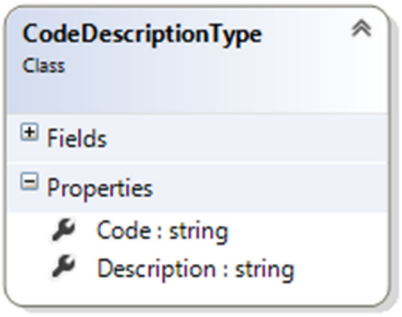


Kuva 27. ResponseType-luokka.

Kuvan 27 ResponseType-luokka sisältää tilauksen tilan. ResponseStatus-kentästä näkee, onko tilaus onnistunut ja Alert -kentästä voi selvittää syitä, jos tilaus ei jostain syystä onnistunutkaan.

TransactionReference -kentästä löytää tilatessa määritetyn CustomerContext-arvon, jos haluaa varmistaa, mistä tilauksesta on kyse. Jollei tee suuria tilausmääriä asynkronisesti yhdellä ajolla, niin todennäköisesti tätä varmistusta ei tarvita.

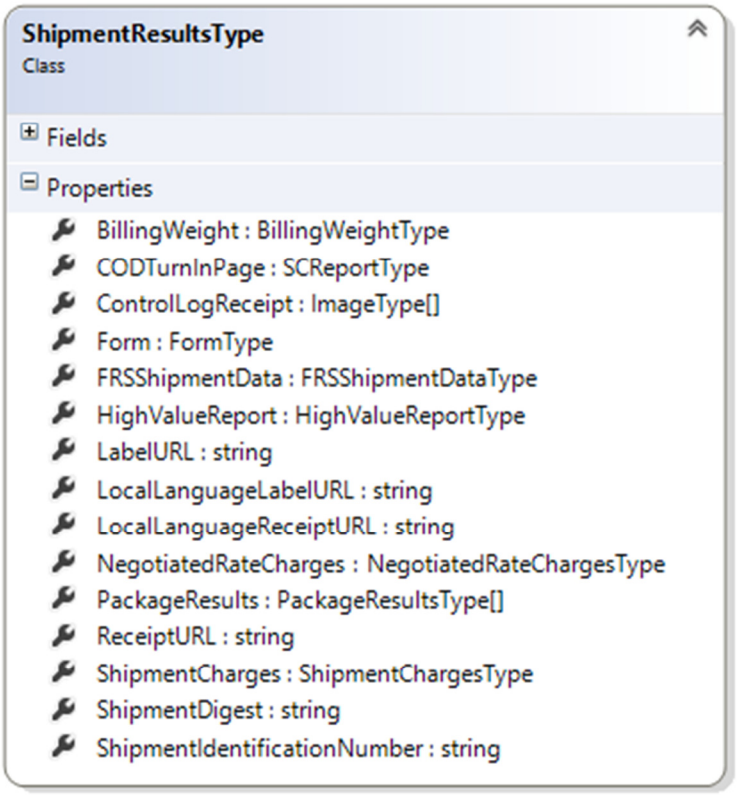
Rajapinnassa ResponseType sijaitsee ShipmentResponse-luokan Response-kentässä.



Kuva 28. CodeDescriptionType-luokka

Kuvan 28 CodeDescriptionType-luokassa on tieto tilaustapahtuman itsensä onnistumisesta. Tila on merkitty sekä kaksimerkkisellä koodilla Code-kenttään, että selkeällä englannilla Description -kenttään. Molempia voi käyttää, mutta yrityksen tarpeisiin Description-kentän katsottiin olevan riittävä. Käytännössä, jos Description-kentästä löytyy jotain muuta kuin teksti ”Success”, pitää tarkastaa ResponseType-luokan Alert-taulusta, mistä ongelma johtuu.

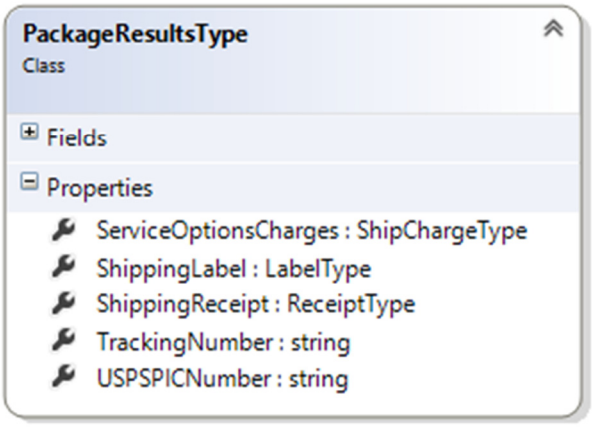
Rajapinnassa CodeDescritionType sijaitsee ResponseType-luokan ResponseStatus-kentästä.



Kuva 29. ShipmentResultsType –luokka.

Kuvan 29 ShipmentResultsType –luokka sisältää kuljetuksen tarkemmat tiedot, kun tilaus on ensin hyväksytty UPS:n palvelimilla. Tärkeimpinä voidaan pitää ShipmentIdentificationNumber -kentän sisältämää 18-merkkistä rahtikirjan numeroa ja syvemmältä PackageResults -taulusta löytyviä rahtikirjoja.

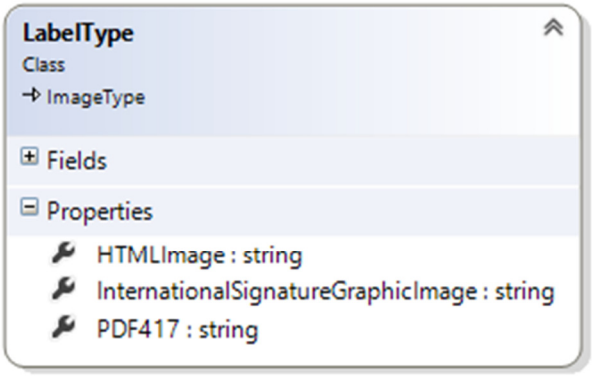
Rajapinnassa ShipmentResultsType sijaitsee ShipmentResponse -luokan ShipmentResults -kentässä.



Kuva 30. PackageResultsType-luokka.

Kuvan 30 PackageResultsType-luokassa on kuljetustilaukseen kuuluvien pakettien pakettikohtaiset tiedot. Jokainen luokan olio sisältää yhden paketin tiedot, rahtikirja ja seurantanumero mukaan lukien. Luokan tärkein tieto on ShippingLabel-kentästä löytyvä rahtikirja.

Rajapinnassa PackageResultsType-olioita sijaitsee ShipmentResultsType-luokan PackageResults-kentässä yksi jokaiselle paketille, joka tilaukseen kuuluu.



Kuva 31. LabelType-luokka.

Kuvan 31 LabelType-luokka sisältää itse rahtikirjan tulostamiseen vaadittavan datan. Tässä kuvassa ei taaskaan ole näkyvillä projektissa käytettyä GraphicImage-kenttää, mutta käytettävä kenttä riippuu täysin tilausta tehdessä annetuista tiedoista. Rahtikirja on säilöttynä valittuun kenttään 64-bittisenä merkkijonona koodattuna. Tämän jälkeen koodaus pitää purkaa ja saatu käskyjono on lähetettävä tulostimelle.

Rajapinnassa LabelType sijaitsee PackageResultsType-luokan ShippingLabel-kentässä.

7 Työn kuvaus

Vanhassa järjestelmässä oli huomattavia ongelmia, joiden korjaukseen katsottiin sopivaksi UPS:n tarjoamaa XML-rajapintaa käyttävä yrityksen sisällä toteutettu tilaus- ja seurantajärjestelmä. Edellisessä luvussa on käyty läpi yksi kolmesta työhön tarvitusta rajapinnasta.

7.1 Vaihe 1 - Tilausautomaatio

Vaatimukset

Järjestelmän tulee kyetä luotettavasti tilaamaan UPS-kuljetus ilman mitään toimenpiteitä käyttäjältä automaation käynnistämisen jälkeen. Yrityksellä on eri asiakkaitten laskutuksessa olevien erojen takia useampi tili UPS:llä ja automaation tulee käyttää tilanteen mukaan oikeaa tiliä laskutuksen tarvitseman kirjanpidon takia. Automaation täytyy myös mukautua ympäröivän järjestelmän muutoksiin ja reagoida oikein erilaisten huoltotapausten vaatimiin toimenpiteisiin.

Automaation UPS-osuuden tulee olla tarpeeksi kevyt, ettei se kuormita turhaan yrityksen järjestelmää tai hidasta loppukäsittelyprosessia.

Automaation on pystyttävä tulostamaan rahtikirja kuljetukselle ja lisättävä seurantanumero yrityksen kirjanpitoon automaattisesti. Toiminnanohjaus-ohjelmistolle riittää pelkkä seurantanumero, mutta tilausjärjestelmä pitää myös itse tarkempaa lokia kaikista kuljetustilauksista. Tähän lokiin sisältyy tiedot tilauksen ajankohdasta ja tilasta, rahtikirjan numero, rahtikirjan kuva yms.

Automaation on myös pystyttävä peruuttamaan vanha kuljetustilaus, jos loppukäsittely on ajettu tapaukselle aiemminkin.

Yrityksen toiminnanohjaus-ohjelmisto on kirjoitettu niin, ettei siihen saa millään minun tuntemallani .NET-työkaluilla niin hyvää kontaktia, että sen käyttöliittymää voitaisiin

käsitellä C#:lla. Ongelma on kontrollien ohjelmallisten nimien selvittämisessä, joita en ainakaan ilman maksullisia lisäpalveluita saa selville .NET-puolella. Yrityksen loppukäsittely on toteutettu AutoIt:llä, koska se sisältää ohjelmallisten nimien selvitystyökalut omaan käyttöönsä.

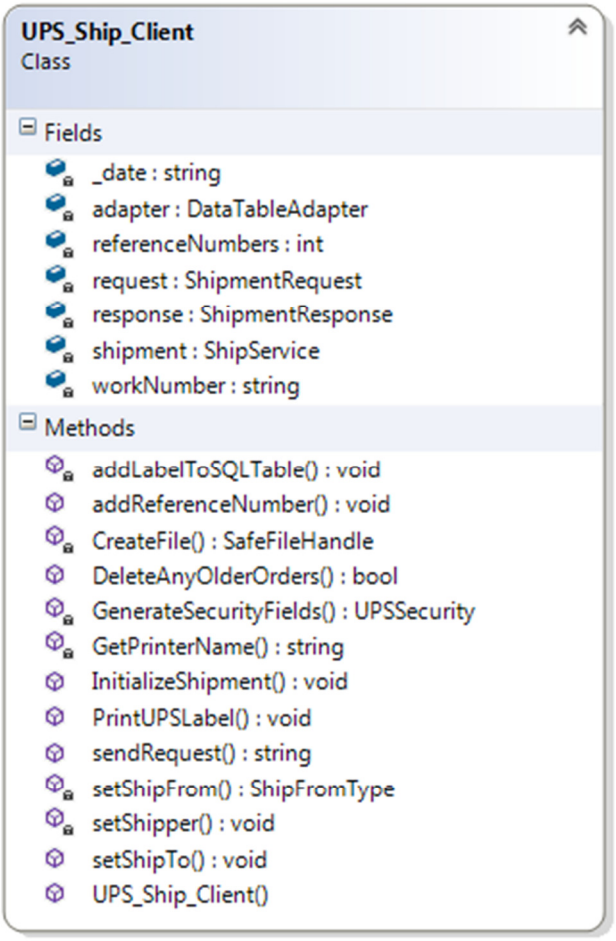
Kuljetustilaus pitää myös integroida vanhaan AutoIt:llä kirjoitettuun loppukäsittelyautomaatioon, jolla on jo todistetusti saatu toiminnanohjausjärjestelmän käyttöliittymän käsittely aikaiseksi.

Toteutus

Koska tilaukset pitää saada automaattisesti eteenpäin, ei tilausta voitu kokonaan eriyttää omaksi ohjelmakseen, sillä ei voida luottaa, että se muistettaisiin ajaa aina tarvittaessa. Automaatio myös pitää lokia kaikista tilaamistaan kuljetuksista SQL-tietokannassa.

Toteutuksena rakennettiin C#-kirjasto, jonka luokkia loppukäsittelyn AutoIt-automaatio voi kutsua. C# on ohjelmointikielenä huomattavasti luotettavampi kuin AutoIt:n kaltaiset skriptikielet, etenkin virheenkäsittelytilanteissa. Toisekseen näin saatiin yhdellä kertaa toteutettua sekä tilauksen että seurannan metodit. Kun kirjastoja on vain yksi, on helpompi ylläpitää sitä.

Kirjastoon kirjoitettiin väliluokka (UPS_Ship_Client seuraavalla sivulla), jonka kautta API:a on huomattavasti helpompi käsitellä, kun kirjastoa kutsutaan AutoIt:n puolelta.



Kuva 32. UPS_Ship_Client-luokka.

Kuvassa 32 oleva UPS_Ship_Client-luokka, on lähetysrajapinnan hyödyntämiseen tehty yläluokka, johon on sisällytetty kaikki yrityksen kannalta tarpeellinen, rajapinnan helppoon hyödyntämiseen. Sen metodit alustavat tarpeelliset kentät ja turvallisuusrakenteet. Luokka sisältää pyynnön ja vastauksen tiedot request- ja response-kenttien muodossa, sekä toiminnallisuuden shipment-kentässä olevan ShipService-olion muodossa. UPS_Ship_Client-luokan olion luominen määrittää ensin turvallisuuskentät, joiden perusteella UPS varmistaa, ettei kukaan yritä huijata vaikkapa tilaamalla kuljetuksia muiden laskuun.

Turvallisuustietojen asettamisen jälkeen luokan metodit asettavat staattiset perustiedot, kuten lähettäjän osoitteen, joita rajapintaan luvun 6 rajapinnan kuvausten mukaan tarvitaan. Tämän lisäksi luokasta löytyy yhden kutsun metodit kohdeosoitteen ja muiden tietojen asettamiseen. Rakenne päätettiin tehdä näin, koska AutoIt:n puolella kutsut on huomattavasti työläämpiä kirjoittaa. Koska AutoIt-koodi kutsuu kirjaston luokkia COM-

rajapinnan kautta, ei rakenteita lähdetty työstämään siellä. Oli huomattavasti helpompi kirjoittaa ja testata rakenteet C#:n puolella, missä oliot ovat sentään kieleen sisäänrakennettuja, eivätkä siten tarvitse erillistä rajapintaa.

Tilausautomaatiolle ei tehty tarkempaa suunnitelmaa, osittain koska se yksinkertaisuudestaan johtuen valmistui API:iin tutustumisen ohessa. Tämä tarkoittaa oikeastaan sitä, että kun tilausautomaation testit alkoivat toimia ja rajapinta alkoi olla niin hyvin hallinnassa, että tarkempia suunnitelmia olisi voitu alkaa tehdä, niin automaatio oli jo käytännössä toimiva.

Eniten ongelmia tuotti rahtikirjan tulostaminen. Koska UPS:n palvelimet tukevat yrityksen tarratulostimia, päätettiin rahtikirjat ottaa tulostimien omassa dataformaatissa. Tämä tarkoittaa, että rahtikirjat tulevat palauteviestissä 64-bit-koodatussa muodossa, josta ne saa C# -ympäristössä helposti purettua luettavampaan 32-bit-muotoon. Purkamisen jälkeen ohjelman täytyy siirtää saamansa tulostinkomentoja sisältävä merkkijono raakadatana tulostimelle. Raakadatan siirtoon tulostimelle ei löytynyt suoraan funktioita C#:sta, mutta onneksi Microsoft on tajunnut ongelman ja tarjoaa tarvittavan koodin tukisivustoillaan RawPrinterHelper-nimisen luokan muodossa [2].

7.2 Vaihe 2 - Seurantaohjelma

Vaatimukset

Yrityksen tarvitsee joskus tietää, missä tilassa sen tilaamat kuljetukset ovat. Tärkeimmät tiedot ovat: onko kuljetus perillä, milloin se on tilattu ja milloin sen tila on viimeksi kysytty. Kuljetuksen tila saadaan UPS:n palvelimilta, mutta viimeisin päivitystieto katsotaan aina oman palvelimen kellosta.

Seurantaohjelman tulee olla käyttöliittymältään selkeä ja helppokäyttöinen, koska se menee useiden teknikoiden käyttöön. Sen tulee kyetä suodattamaan näkyviin kuljetukset vain tietyltä ajalta, teknikolta, huoltotapaukselta tai tietyssä tilassa olevat kuljetukset. Ohjelman tulee myös pystyä tulostamaan hukatut rahtikirjat uudelleen ja tarpeen tullen peruuttamaan kuljetuksia, joita ei ole vielä noudettu.

Ohjelma on vain työkalu kuljetusten tilan tarkkailuun. Se näyttää tietokannassa olevien tilausten tilan ja antaa mahdollisuuden päivittää yksittäisten tilausten tilan tai peruuttaa koko kuljetuksen.

Koska ohjelma on vain yksinkertainen teknikoille tarkoitettu seurantaohjelma, tarvitaan järjestelmään myös keino ylläpitää kuljetustilausten tilaa automaattisesti. Tähän tarkoitukseen toteutettiin vaiheessa 3 tilan ylläpito-ohjelma.

Suunnitelma

Ohjelma toteutettiin täysin C#:lla hyödyntäen samaa kirjastoa kuin automaatiossakin.

Käyttöliittymäksi valittiin yksinkertainen taulukkonäkymä, jossa käyttäjä näkee ne kuljetustilaukset, joita hänen on oikeus nähdä. Ohjelman sisältämät käyttöoikeustiedot saadaan Microsoftin käyttäjätietokannasta (Active Directory) tarkastamalla, kuuluuko käyttäjä korkeamman tason käyttäjäryhmään.

Peruskäyttäjät näkevät ja voivat muokata vain omien kuljetustensa tiloja, mutta korkeamman tason käyttäjät näkevät ja voivat muokata kaikkien kuljetustilausten tilannetta.

Käyttäjillä on ohjelmassa painikkeet kuljetustilausten tilan tarkastamiseen ja päivittämiseen ja kuljetustilauksen poistamiseen. Yksittäisen tilauksen tilaa ei voi päivittää kuin kerran tunnissa, etteivät liialliset päivitys tiedustelut turhaan kuormita UPS:n palvelimia.

Toteutus

Tämän vaiheen toteutukseen tarvittiin seuranta- ja poistorajapintoja: seurantarajapintaa kuljetustilausten tilan tiedustelemiseen UPS:ltä ja poistorajapintaa niiden tilausten peruuttamiseen, joiden tilauksessa oli jotain pielessä.

Tilanteen seurannan takia vaiheen 1 tilausautomaatioon kirjoitettiin mukaan myös lokitietojen ylläpitoa SQL-tietokantaan.

7.3 Vaihe 3 - Tilan ylläpito

Vaatimukset

Järjestelmä vaatii myös sen, että kuljetusten tila pidetään edes suunnilleen ajan tasalla myös ilman manuaalista päivittämistä. Sen täytyy kyetä päivittämään kaikkien kuljetustilausten tila automaattisesti.

Toteutus

Koska UPS:n seurantarajapinta ei tue kaikkien keskeneräisten kuljetusten tietojen hakemista kerralla, päätettiin toteuttaa yksinkertainen ohjelma, joka käy automaation tekemät SQL-rivit läpi ja etsii kaikki kuljetukset, joita ei vielä ole toimitettu asiakkaalle asti. Kun ohjelma on saanut listan valmiiksi, se kyselee UPS:ltä yksi kerrallaan jokaisen kuljetuksen tilan ja päivittää sen tietokantaan. Lopuksi ohjelma kirjaa oman lokinsa SQL-tietokantaan, jotta sen toimintaa on mahdollista seurata.

Tämä päivitinohjelma ajoitettiin yrityksen palvelimelle ajettavaksi joka yö.

UPS tarjoaa kyllä muita keinoja tilatietojen noutamiseen suuremmissa määrissä, mutta koska yrityksellä oli työn tähän pisteeseen edettyä jo toimivaksi todettu tapa saada yksittäisten tilausten tiedot, päätettiin ajan säästämiseksi hyödyntää sitä. Muussa tapauksessa olisi pitänyt ottaa mukaan vielä vähintään yksi uusi rajapinta. Tämän lisärajapinnan käyttöönotto olisi hidastanut projektia huomattavasti.

8 Lopputulos ja päätelmät

Projekti saatiin päätökseensä ajoissa ja sen toiminta saatiin varmistettua. Työt tehtiin vuoden 2012 syksyllä ja niissä kesti noin 4 kuukautta ensimmäisestä kokouksesta käyttöönottoon.

Projektin aikana mitattiin tilaukseen kuluva aika vanhalla ja uudella järjestelmällä. Kuluneen ajan lokitietojen keräyksessä oli ikävä huolimattomuusvirhe, jonka takia suuri osa lokitiedoista jouduttiin jättämään kokonaan käyttämättä. Ongelmana oli se, etten ollut muistanut päivittää käytettävän version tietoja lokiin kirjaamisesta vastaaviin funktioihin ja ne ovat menneet paikoittain sekaisin. Tästä johtuen yrityksellä ei ole varmuut-

ta suuresta osasta lokitiedoista. Ainoa tapa saada mitään luotettavuutta oli ottaa lokitiedot projektin alkupäästä vanhalle järjestelmälle ja loppupäästä uudelle. Kaikki muut jouduttiin jättämään huomiotta. Tämä vähensi huomattavasti käytettävissä olevien näytteiden määrää.

En saanut varmuudella eroteltua kaikkia vanhan järjestelmän lokitietoja uusista, mutta jopa näissä oloissa puhutaan lähes kymmenen sekunnin erosta keskiarvoissa. Vanhan järjestelmän keskiarvo oli 33,078 sekuntia ja uuden järjestelmän 23,342 sekuntia. Hämmentävää tästä tekee se, että kaikissa testitapauksissa, jotka kehitystyön aikana tein, uuden järjestelmän käsittelyaika oli enimmilläänkin huomattavasti alle 20 sekuntia. Vastaavasti vanhalla järjestelmällä testit ja koodissa olevat viiveet viittasivat vähintäänkin 30 sekunnin kestoon. Näissä oloissa todennäköisimmältä näyttäisi, että suuri osa näytteistä on mennyt sekaisin huonon versiomerkinnän takia jo ensimmäisten testien aikana. Näiden versiomerkintöjen ongelmien takia en saanut luotettavaa hajontaa selville.

Uudessa UPS-käsittelyssä on kuitenkin edelleen hiottavia kohtia, joihin minulla ei enää testien jälkeen ollut aikaa vaikuttaa. Ongelmia tuli etenkin tulostimen valinnassa. Tulostimen valintafunktio kun on kirjoitettu niin, että se valitsee ensimmäisen päällä olevan tarratulostimen. Tässä tilanteessa ongelma johtuu tulostimen ajuriasennuksen luomasta haamutulostimesta. Kun ajurin ensimmäistä kertaa asentaa koneeseen, se luo aina päällä olevan haamutulostimen, joka uskottelee koneelle olevansa COM-portissa. Ikävä kyllä UPS-käsittely löytää tämän haamutulostimen ja yrittää käyttää sitä. Ongelmaan on parikin ratkaisua, joista ehdin jättää projektia ylläpitämään tulleelle ohjelmistokehittäjälle ohjeet.

Kaikista edellä mainituista ongelmistaan huolimatta uusi järjestelmä on tuotantokäytössä ja se on korvannut vanhan täysin. Vanha UPS World Ship-ohjelma on edelleen käytössä erikoistapauksia varten.

Lähteet

1 Shipping Package WebServices Developers Guide.pdf. 2011. UPS API documentation

2 How to send raw data to a printer by using Visual C# .NET. 2006. Verkkodokumetti. Microsoft. <http://support.microsoft.com/kb/322091>. Päivitetty 09.05.2006. Luettu 28.09.2012

3 AutoIt. 2012. Verkkodokumentti. Jonathan Bennett & AutoIt Consulting Ltd. <http://www.autoitscript.com/site/autoit/>. Luettu 04.10.2012

4 C Sharp (programming language). 2013. Verkkodokumentti. Wikipedia. <http://en.wikipedia.org/wiki/C_Sharp_(programming_language)>. Päivitetty 06.05.2013. Luettu 06.05.2013.

5 Paul J. Deitel ja Harvey M. Deitel. 2008. C# 2008 for Programmers. Third Edition. ISBN-10: 0-13-714415-6 ja ISBN-13: 978-0-13-714415-0

6 Web Services Description Language. 2013. Verkkodokumentti. Wikipedia. <http://en.wikipedia.org/wiki/Web_Services_Description_Language> Päivitetty 02.05.2013. Luettu 06.05.2013.