Web Services -palvelun suunnittelu ja toteutus

Harri Oksa

Opinnäytetyö
Marraskuu 2012

Mediatekniikan koulutusohjelma
Tekniikan ja liikenteen ala

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖN
KUVAILULEHTI

Tekijä(t) OKSA, Harri	Julkaisun laji Opinnäytetyö	Päivämäärä 29.11.2012
	Sivumäärä 25	Julkaisun kieli Suomi
	Luottamuksellisuus ( ) saakka	Verkkojulkaisulupa myönnetty (X)

Työn nimi
Web Services -palvelun suunnittelu ja toteutus

Koulutusohjelma
Mediatekniikka

Työn ohjaaja(t)
PELTOMÄKI, Juha

Toimeksiantaja(t)
TOSSAVAINEN, Seppo - Tietosuunta Oy

Tiivistelmä

Opinnäytetyön tarkoituksena oli suunnitella ja toteuttaa Web Services -palvelu, jonka avulla verkkolaskuja pystytään lähettämään pankkien maksuliikennepalveluun. Työn toisena tavoitteena oli palvelun helppo liitettävyys valmiiseen laskutusjärjestelmään. Web Services -palvelun rakenteesta tuli tehdä sellainen, että mahdollinen jatkokehitys olisi helppoa.

Työssä tutkittiin XML- ja Web Service -tekniikoita, joiden pohjalta lähdettiin rakentamaan Web Services -palvelua. Palvelun toteuttaminen vaati tarkempaa tutustumista XML-tekniikoihin ja erityisesti XML-dokumenttien digitaalisiin allekirjoituksiin. Web Services -yhteyden toteuttamiseksi tutkittiin SOAP-tekniikkaa ja yhteyden toteuttamista sen avulla.

Työn toteutuksen tuloksena saatiin toimiva Web Services -palvelu, joka voitiin liittää laskutusjärjestelmään. Palvelun ohjelmointi toteutettiin luokkarakenteella, jotta se olisi mahdollisimman helppokäyttöinen ja rakenteeltaan selkeä. Palvelun ansiosta laskutusjärjestelmästä pystyy lähettämään verkkolaskuja.

Avainsanat (asiasanat)

Web Service, SOAP, SEPA, XML, PHP, digitaalinen allekirjoitus

Muut tiedot

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

DESCRIPTION

Author(s)	Type of publication	Date
OKSA, Harri	Bachelor´s Thesis	29.11.2012
	Pages 25	Language Finnish
		Permission for web publication (X)

Title
PLANNING AND IMPLEMENTATION OF WEB SERVICE

Degree Programme
Media Engineering

Tutor(s)
PELTOMÄKI, Juha

Assigned by
TOSSAVAINEN, Seppo - Tietosuunta Oy

Abstract

The purpose of the Bachelor's Thesis was to plan and implement a Web Services -service that allows invoices to be sent into banks Web Services channel. An additional goal was easy connectivity to an existing invoicing system. The structure of the service was to allow simple additional development of the software.

In the thesis XML- and Web Service -techniques were researched and Web Service service was built based on the results. The implementation of the service required further research on XML techniques and specifically on digital signatures. The implementation also required understanding of basic SOAP principles.

As a result of the thesis, a functional Web Services service was implemented with easy connectivity to the invoicing system. The programming of the service was conducted with class structure in order for it to be easy to use and also to have clear code structure. With the help of the service, invoices can now be sent from the invoicing system.

Keywords

Web Service, SOAP, SEPA, XML, PHP, digital signature

Miscellaneous

SISÄLTÖ

KUVIOT

SANASTO JA KÄYTETYT TERMIT

APIX

Rajapintajärjestelmä, jonka kautta voidaan lähettää uusia SEPA-standardin verkkolaskuja.

HTTP

HTTP eli Hypertext Transfer Protocol on selaimien ja WWW-palvelimien käyttämä protokolla tiedonsiirtoon.

ISO 20022

ISO-organisaation määrittelmä XML-pohjainen sanomavälitysjärjestelmä. ISO 20022 -sanomat on tarkoitettu yritysten ja pankkien väliseen viestintään.

MySQL

SQL-standardia noudattava suosittu SQL-tietokannan hallintajärjestelmä.

PKI

Public Key Infrastucture. Asymmetrisiin algoritmeihin perustuva salausjärjestelmä. Järjestelmä pitää sisällään myös julkisten avainten jakelun ja hallinnan varmenteiden muodossa.

URI

URI eli Uniform Resource Identifier on merkkijono, jonka avulla identifoidaan abstrakti tai fyysinen resurssi. URI:t joiden avulla tieto paikannetaan internetistä, ovat nimeltään URL.

WSO2

PHP:lle kehitetty lisäosa, jonka komponenttien avulla voidaan helposti kehittää laajoja Web Services -palveluita

1 TYÖN LÄHTÖKOHDAT

1.1 Toimeksiantaja

Tietosuunta Oy on vuonna 1989 perustettu yritys. Yrityksen toimiala on ohjelmistojen suunnittelu ja toteutus pk-yrityksille. Yhtiön toimipiste sijaitsee Jyväskylässä Kypärätiellä.

Tietosuunnan tuotteisiin kuuluu taloushallinnon ohjelmia, joita ovat palkanlaskenta, kirjanpito, laskutus, vesilaskutus, sopimuslaskutus, ostoreskontra ja jäsenrekisteri.

Taloushallinnon ohjelmat ovat saatavissa myös etäyhteytenä Windows Server -virtuaalipalvelimilla, jolloin asiakkaille on tarjottavissa myös käyttöjärjestelmäriippumaton vaihtoehto. Yrityksen henkilöstömäärä on tällä hetkellä viisi ja asiakkaita yrityksellä on yli 2000. Tietosuunta Oy:llä on myös Iso-Britanniaan rekisteröity yhtiö AC Balance Limited, joka toimii Internetissä markkinointi- ja jakeluyhtiönä ja toimittaa Tietosuunnan taloushallinnon ohjelmia EU-alueelle englannin-, saksan-, ranskan-, italian-, espanjan-, ruotsin- ja suomenkielisinä versioina. (Tossavainen. 2012)

Yrityksellä on myös vankkaa osaamista erilaisista GPS-tietoja hyödyntävistä ohjelmista, joita ovat mm. Route-Tracker-ajoneuvopaikannus ja ajopäiväkirja, jonka avulla yritykset pystyvät seuraamaan ajoneuvojansa ja työntekijät helposti raportoimaan työajojansa. GPS-tietoja hyödyntäviin ohjelmiin kuuluu myös Paikkari Koira GPS, jonka avulla metsästäjät pystyvät seuraamaan koiransa kulkua maastossa reaaliajassa selainyhtettä tukevalla puhelimella maastokartat.fi palvelun kautta, jossa käytetään maanmittauslaitoksen karttoja Google Maps -ympäristössä.

1.2 Työn taustat

Euromaksualueella ollaan siirtymässä SEPA-aikaan. SEPA-aikaan siirtymisen jälkeen pankit eivät ota vastaan enää muita kuin SEPA-muotoisia XML-maksutiedostoja. Tämän vuoksi Tietosuunta Oy:lla oli tarve päivittää heidän uusi web-pohjainen laskutusjärjestelmänsä SEPA-yhteensopivaksi.

Tietosuunta Oy on tähän asti käyttänyt SEPA-maksujen lähetykseen APIX-rajapintapalvelua, jonka kautta maksut on lähetetty. Rajapintapalvelun heikkouksia ovat maksullisuus ja se, että maksut kulkevat APIX-palvelun palvelimien kautta pankkiin, joten ongelmatilanteissa yrityksellä ei ole mahdollisuutta selvittää virheen syytä.

Yrityksen oman Web Services -pankkiyhteyden ansiosta maksujen koko lähetysprosessi on yrityksen hallinnassa ja ongelmatilanteisiin pystytään reagoimaan nopeasti. Myöskään palvelun kehitys ei ole enää riippuvainen ulkoisesta palveluntarjoajasta.

1.3 Työn tavoitteet

Työn tavoitteena oli luoda rajapinta Tietosuunta Oy:n laskutusjärjestelmän ja pankkien Web Services -palveluiden välille. Tavoitteena oli, että laskutusjärjestelmään ei tarvitsisi tehdä suuria muutoksia, vaan laskut voitaisiin tallentaa tietokantaan olemassa olevaa mallia käyttäen ja toteutettu Web Services -asiakasohjelma poimisi tiedot tietokannasta ja muodostaisi SEPA-standardin mukaisen XML-maksusanoman ja lähettäisi sen pankkiin ja välittäisi saadun vastauksen laskutusohjelmalle.

Toisena tavoitteena oli tiliotteiden ja muiden aineistojen haku Web Services -asiakasohjelmalla ja näiden aineistojen tallentaminen tietokantaan muotoon, josta laskutusjärjestelmä voisi lukea tiedot ja näyttää ne käyttäjälle.

Rajapinnan toteutuksessa piti käyttää PHP:tä, koska Tietosuunta Oy:n laskutusjärjestelmä oli rakennettu PHP:llä. Tietokantana laskutusjärjestelmässä toimii MySQL, joten sitä täytyi käyttää myös rajapinnan tietokantayhteyksissä.

2 SEPA

2.1 Yleistä

SEPA eli Single Euro Payment Area tarkoittaa yhtenäistä euromaksualuetta. SEPAn tarkoituksena on yhtenäistää euromaksualueella maksamisen peruspalveluiden standardit ja käytännöt. Näihin peruspalveluihin kuuluvat mm. tilisiirrot, maksukortit ja suoraveloitukset. SEPA yhdistää myös maksamisen osapuolien oikeudet ja velvollisuudet. (Mitä SEPA tarkoittaa?. 2012)

Helmikuussa 2012 EU-parlamentti hyväksyi asetuksen, joka asettaa takarajan SEPA-palveluiden käyttöönotolle ja vanhoista suoraveloituksista ja kansallisista tilisiirroista luopumiselle. Suomessa suoraveloituksesta luovutaan viimeistään 31.1.2014 ja tämän tilalle suositellaan e-laskua ja uutta suoramaksua. Tilisiirtojen osalta Suomi on jo siirtynyt SEPA-aikaan ja tilisiirrot ovat SEPA-yhteensopivia. (Mitä SEPA tarkoittaa?. 2012)

2.2 SEPA-maksujen standardit

Yhtenäisen euromaksualueen yksi tavoitteista on standardien yhtenäistäminen. Tämä helpottaa asiakkaan toimintaa sillä tavoin, ettei enää ole väliä, mitä pankkia käytetään ja tapahtuuko maksu maan sisällä vai toiseen SEPA-alueen maahan. SEPA-suoraveloitusten ja SEPA-tilisiirtojen välitykseen valittiin standardi ISO 20022 XML-pohjainen sanoma. Myös SEPA suoraveloituspalvelu toimii XML-standardilla. (Yhtenäisen euromaksualueen toteutuminen Suomessa. 2012)

2.3 SEPAn vaikutus pankkiyhteyksiin

Pankkien eräsiirron tiedonsiirtomenetelmä (FTP) ja turvamenettely (PATU) korvataan uusilla kehittyneemmillä ja turvallisemmilla standardeilla. Web Services täyttää edellämainitut vaatimukset, ja useimmat pankit tukevat jo Web Services -yhteyskäytäntöpalvelua. Palvelun kuvaukset löytää Finanssialan Keskusliiton sivuilta. SEPA-aika tarkoittaa käytännössä siirtymistä XML-muotoiseen maksuaineistoon. (Yhtenäisen euromaksualueen toteutuminen Suomessa. 2012)

3 XML

3.1 Mitä on XML?

XML on tekstimuotoista rakenteellista kuvauskieltä, jossa tiedon lisäksi voidaan kuvata myös tiedon merkitys. XML-kieli muistuttaa HTML-kieltä, jota käytetään WWW-sivujen tekemiseen. XML-kielen ja HTML-kielen ero on siinä, että HTML:ssä on ennalta määrätyt elementit, joita voidaan käyttää, kun taas XML-kielessä elementit voidaan nimetä halutulla tavalla, joka kuvaa elementin sisäistä tietoa parhaiten. Yksinkertaistettuna voidaan siis sanoa, että XML on yleistä tiedon kuvaamista varten, kun taas HTML on vain WWW-sivujen sisällön kuvaamista varten. XML-kielen on kehittänyt World Wide Web Consortium eli W3C. (Rouse. 2007; Extensible Markup Language. 2008)

3.2 XML-tiedoston rakenne

XML-tiedostoa voisi kuvailla hierarkkisena dokumenttipuuna. Tämä dokumenttipuu sisältää solmukohtia, jotka ilmentävät yksittäistä elementtiä dokumentissa. Solmukohtia voi olla rajattomasti dokumentin sisällä.(2kmediat. 2012b)

Kuviossa 1 on esitetty yksinkertaisen XML-dokumentin rakenne. Dokumentin alussa rivillä yksi on dokumentin prosessointikäsky, jossa on määritelty dokumentin yleisiä ominaisuuksia, kuten XML-versio ja käytetty merkistökoodaus. Rivin kaksi elementti on dokumentin juurielementti, eli juurielementtinä on elementti nimeltä kirjahylly. Juurielementin sisällä on kaksi kirja-elementtiä, jotka ovat kirjahylly-elementin lapsielementtejä. Kirjoilla on tässä tapauksessa kolme lapsielementtiä, jotka kuvaavat kirjan ominaisuuksia. Kuviossa 1 riveillä kolme ja kahdeksan voi nähdä, että Kirja-elementeillä on attribuuttina id. XML-attribuuteilla voi elementtien ohella kuvata tietoa. XML-attribuuttien heikkous on se, että ne eivät voi sisältää useita arvoja toisin kuin elementit, joille voi lisätä lapsielementtejä. XML-attribuuteilla on kuitenkin hyvä kuvata tietoa, joka ei olennaisesti liity elementtiin. Kuvion 1 tapauksessa attribuutti id on ainoastaan kirjan järjestysnumero, joten sillä ei ole samanlaista arvoa kuten nimellä, julkaisuvuodella tai kustantajalla. Alemmasta kirja-elementistä voi nähdä, että kustantaja-elementillä ei ole arvoa. XML-elementeissä ei tarvitse olla arvoa, vaan ne voivat olla myös tyhjiä. Tyhjän elementin voi merkitä XML-dokumentissa kuten Kustantaja-elementti on merkitty eli ilman lopputagia ja lisäämällä kauttaviivan ensimmäisen tagin loppuun. (Rouse. 2007; 2kmediat. 2012a; 2kmediat. 2012b; Extensible Markup Language. 2008)

```
1  <?xml version="1.0" encoding="UTF-8">
2  <Kirjahylly>
3      <Kirja id="1">
4          <Nimi>XML-perusteet</Nimi>
5          <Vuosi>2003</Vuosi>
6          <Kustantaja>Otava</Kustantaja>
7      <Kirja>
8      <Kirja id="2">
9          <Nimi>WSDL-alkeet</Nimi>
10         <Vuosi>2013</Vuosi>
11         <Kustantaja/>
12     <Kirja>
13 </Kirjahylly>
14
```

KUVIO 1. XML-dokumentin rakenne

4 WEB SERVICES

4.1 Web Services yleisesti

Web Service on ohjelmistojärjestelmä, jonka avulla keskenään yhteensopivat tietokoneet voivat kommunikoida tietoverkon yli. Yksinkertaistettuna Web Service tarkoittaa WWW-pohjaisia ohjelmointirajapintoja. Web Servicessä palvelun tarjoaja ja käyttäjä kommunikoivat keskenään erilaisten XML-pohjaisten protokollien avulla. Protokolliin kuuluu seuraavat kolme komponenttia: Simple Object Access Protocol eli SOAP, Web Services Description Language eli WSDL sekä Universal Description Discovery and Integration eli UDDI. (Web Services Architecture. 2004; W3Schools)

4.2 SOAP

4.1.1 Yleistä

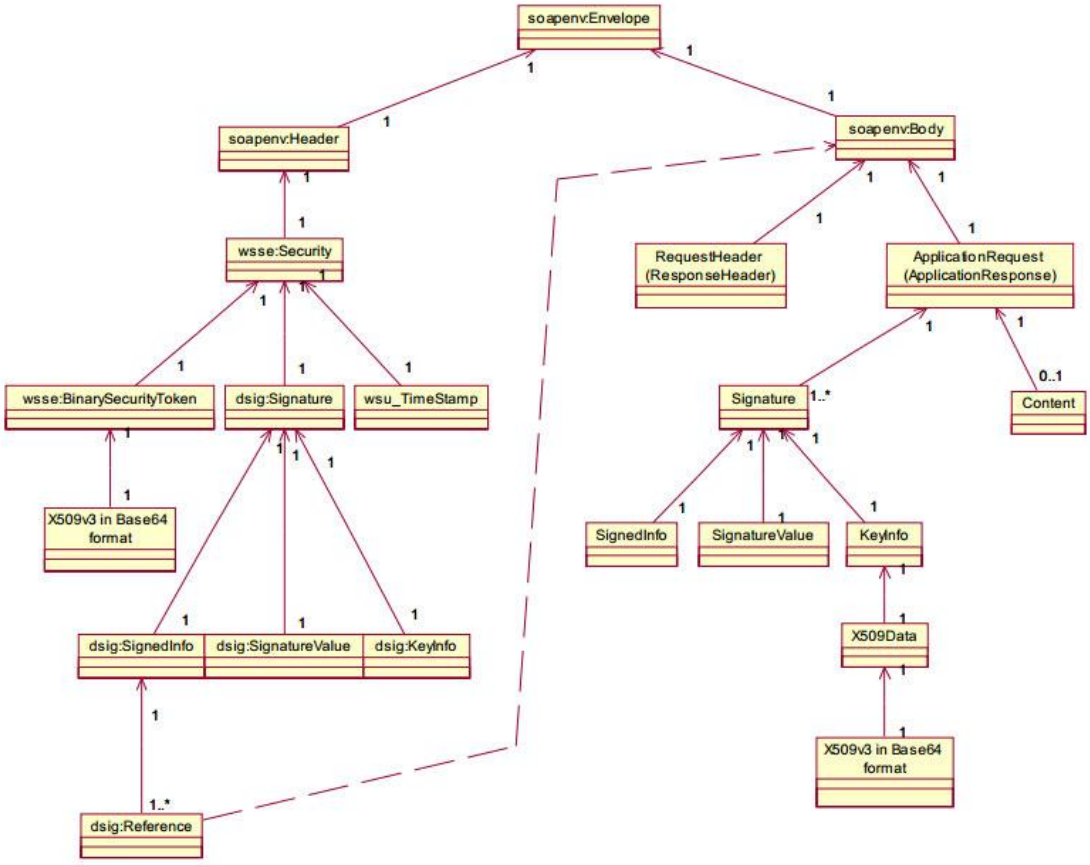
SOAP on tiedonsiirtoprotokolla, jota käytetään sovellusten väliseen yhteydenpitoon. SOAP rakentuu kolmesta eri osasta ja on rakenteeltaan XML-kuvauskieltä. SOAP on riippumaton kuljetusprotokollasta. Useimmiten käytössä on HTTP, mutta myös esimerkiksi SMTP ja FTP ovat mahdollisia kuljetusprotokollia SOAP-sanomille.(Simple Object Access Protocol. 2000; Service-oriented architecture and Web services. 27)

Yksinkertaisimmillaan SOAPia voidaan käyttää yksisuuntaiseen kommunikointiin, jossa asiakassovellus lähettää SOAP-pyynnön palvelimelle, jossa palvelin käsittelee pyynnön ja lähettää SOAP-vastauksen lähetetyn tiedon perusteella. Tietosuunta Oy:lle toteutetussa Web Services -järjestelmässä käytetään yksisuuntaista kommunikointia, jossa sovelluksesta lähetetään pyyntöjä pankin palvelimille, joka vastaa pyyntöihin lähetetyn tiedon perusteella.

(Simple Object Access Protocol. 2000; Service-oriented architecture and Web services. 27)

4.1.2 SOAP-sanomien rakenne

SOAP-sanoma on XML-dokumentti, jonka pakollisia osia ovat SOAP-kirjekuori ja SOAP-viesti. SOAP-sanoma voi sisältää myös SOAP-otsikkotiedot. Otsikkotiedot ovat valinnainen osa SOAP-sanomaa, mutta SEPA Web Services -pyynnöissä otsikkotiedot ovat pakollisia. SOAP-sanoman rakenne SEPA ja sanoman osien järjestys pankkien Web Services -kanavassa on kuvattu kuviossa 2. (Service-oriented architecture and Web services. 28)



KUVIO 2. SOAP-sanoman rakenne pankkien Web Services -kanavassa

SOAP-kirjekuori

SOAP-kirjekuori määrittelee XML-dokumentin SOAP-viestiksi. Kirjekuori on XML-dokumentin juurielementti, jonka sisälle otsikkotiedot ja viesti tulevat.

SOAP-otsikkotiedot

Otsikkotiedot määrittävät, kuinka vastaanottajan tulisi sanoma käsitellä. SEPA Web Services -järjestelmässä otsikkotiedoissa välitetään lähettäjän digitaalinen allekirjoitus, jonka avulla vastaanottaja pystyy tunnistamaan lähettäjän ja varmistamaan viestin oikeellisuuden.

SOAP-viesti

Viesti on SOAP-sanoman osa, joka sisältää varsinaisen tiedon, joka halutaan välittää vastaanottajalle.

4.3 REST

Web Services -palvelun voi toteuttaa myös tekniikalla nimeltä REST eli Representational State Transfer. REST-tyylisen palvelun toteuttaminen on huomattavasti helpompaa kuin esimerkiksi SOAP-tyylisen palvelun toteutus. Tämän vuoksi REST-palveluja suositaan kuluttajille suunnatuissa rajapintapalveluissa. Esimerkiksi Twitter tarjoaa mahdollisuuden käyttää Web Services -rajapintaa, joka mahdollistaa Twitterin käytön omissa sovelluksissa. (Elkstein. 2008a; Elkstein. 2008b)

REST-palvelun tekee yksinkertaiseksi se, että toisin kuin SOAP-palveluissa, joissa muodostetaan XML-muotoinen SOAP-kirjekuori ja lähetetään se Web Services -palvelimelle WSDL-tiedoston määritysten mukaan, REST-palveluissa pyynnöt ovat URI-pohjaisia. Näiden tekniikoiden ero selviää kuviosta 3, jossa ylempänä on SOAP-tekniikalla muodostettu pyyntö, jolla haetaan käyttäjän 12345 tiedot palvelimelta. SOAP-sanoman alla on sama pyyntö toteutettu käyttäen REST-tekniikkaa, jossa HTTP-protokollaa ja HTTP:n GET-

komentoa käyttäen haetaan tiedot ja haettavan tiedon tyyppi ja tunniste on määritelty URL-osoitteessa. Myös pyyntöjen vastauksissa on eroja. SOAP-pyyntö palauttaa vastauksena SOAP-kirjekuoren, kun taas REST-pyynnön vastauksena saadaan vain haettu data, jonka ympärillä ei ole mitään SOAP-kirjekuoren tyylisiä kehyksiä. (Elkstein. 2008a; Elkstein. 2008b)

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
 <soap:body pb="http://www.acme.com/phonebook">
  <pb:GetUserDetails>
   <pb:UserID>12345</pb:UserID>
  </pb:GetUserDetails>
 </soap:Body>
</soap:Envelope>
```

```
http://www.acme.com/phonebook/UserDetails/12345
```

KUVIO 3. SOAP- ja REST-tyyliset pyynnöt

4.4 WSDL

Web Service Description Language eli WSDL on XML-perustainen kieli, jonka avulla kuvataan web-teknologioihin perustuva palvelu tietoverkossa. WSDL-kuvaus koostuu kahdesta eri osasta. Abstrakti osa kuvaa vain palvelun rajapinnan eli toiminnot, joita palvelussa voi suorittaa. WSDL:n konkreettinen osa kuvaa taas yhteyden osoitteen ja protokollan, jota käytetään. WSDL on siis yksinkertaistettuna Web Services palvelun tekninen kuvaus. (Web Services Description Language. 2001)

5 WEB SERVICES -YHTEYDEN TOTEUTUS

5.1 Yleistä

Jotta ohjelma pankin ja yrityksen välille voidaan luoda, täytyy pystyä hahmottamaan Web Services -yhteyden kulku pyyntöä

lähetettäessä. Seuraavassa on kuvattu yksinkertaistetusti vaiheet, joita tarvitaan onnistuneen pyynnön suorittamiseen.

Ensimmäiseksi luodaan ApplicationRequest XML-rakenne, jossa on pankkiin lähetettävän pyynnön kaikki tiedot mukaan lukien mahdolliset maksuaineistot jos ollaan lähettämässä aineistoja pankkiin. Valmiille XML-rakenteelle suoritetaan digitaalinen allekirjoitus käyttäen asiakkaan varmennetta ja sen yksityistä avainta, jotta käyttäjän oikeellisuus ja aineiston muuttumattomuus lähetyksen aikana voidaan taata. Tämän jälkeen ApplicationRequest XML-rakenne base64-koodataan. (Security and Message Specification for Financial Messages using Web Services. 2008)

Seuraavaksi luodaan SOAP-sanoma käyttäen tähän tarkoitukseen luotua yhteysohjelmaa. SOAP-sanoman viestin ApplicationRequest-elementtiin sijoitetaan enkoodattu ApplicationRequest. SOAP-sanoma allekirjoitetaan myös käyttäen käyttäjän yksityistä avainta. SOAP-sanoma lähetetään pankkiin ja odotetaan pankin vastausta. Kun pankilta saadaan vastaus, ensimmäiseksi tarkistetaan saadun vastauksen allekirjoitus ja sen oikeellisuus. Jos allekirjoitus on oikea, siirretään vastauksen tiedot sisältävä ApplicationResponse XML-rakenne mahdolliseen jatkokäsittelyyn. (Security and Message Specification for Financial Messages using Web Services. 2008)

5.2 Tekniikoiden valinta

Alussa rajapintajärjestelmä oli tarkoitus toteuttaa WSO2 Web Services -ohjelmistokehystä käyttäen. WSO2-moduuli täytyy asentaa palvelimelle, ennen kuin sen komponentteja pystyy käyttämään sovelluksessa.

Sovelluksen käyttöönotto oli vaikeaa siinä mielessä, että se täytyi asentaa jokaiselle palvelimelle, jossa sitä haluttiin käyttää.

WSO2:n vahvuuksiin kuului ehdottomasti yhteysloki, josta pystyi näkemään lähetettyjen pyyntöjen ja pyyntöjen vastausten eri

vaiheet ja mahdolliset virheet. XML-dokumenttien ja SOAP-sanomien digitaalinen allekirjoitus kävi myös helposti kyseistä moduulia käyttäen. Kun yhteys oli saatu muodostettua ja virheet koodissa korjattua, ilmeni kuitenkin ylitsepääsemätön ongelma itse moduulissa. Vastausviestien pituudesta johtuen vastausten tarkistussumman muodostamisessa tapahtui virhe, kun koko XML-dokumentti ei mahtunut moduulin sisäiseen muuttujaan ja tarkistesumman vertaus viestiin epäonnistui joka kerta, kun pankin vastaus oli riittävän pitkä. Moduulin kehittäjiltä ei saatu vastausta, joten moduulin käyttö rajapintajärjestelmässä jouduttiin hylkäämään kokonaan.

Nykyisessä toimivassa ratkaisussa päädyttiin käyttämään PHP:n SoapClient-luokkaa, josta tehtiin oma versio, joka periytyy SoapClient-luokasta. Allekirjoitusten muodostus ohjelmoitiin itse omaan TSSoapClient-luokkaan.

5.3 ApplicationRequest

5.3.1 Yleistä

ApplicationRequest on XML-tiedosto, joka sisältää pankista haettavat tai pankkiin lähetetettävät tiedot. Pakollisiin tietoihin kuuluivat pyynnön tyyppi, pyynnön muodostavan sovelluksen tunniste ja aikaleima. Tiedoissa, joita täytyy lähettää, on pieniä eroja pankkien välillä, joten tämä täytyy ottaa huomioon pyyntöjä muodostettaessa. ApplicationRequest lähetetään pankkiin SOAP-sanoman muun sisällön mukana base64-enkoodattuna.

ApplicationRequest-tiedoston muodostamista varten tehtiin oma PHP-luokka. Luokan sisällä XML-tiedosto rakennetaan käyttäen PHP:n DOMDocument-luokkaa, josta ApplicationRequest -luokka periytyy. Kuviossa 4 näkyy, mitä elementtejä ApplicationRequest-tiedostoon voidaan sijoittaa. Kuviossa 4 katkoviivoin reunustetut

kentät eivät ole pakollisia kaikissa tapahtumissa, joita pankkiin lähetetään, kun taas tasaisen reunuksen omaavien kenttien täytyy esiintyä jokaisessa pyynnössä, mikä pankkiin tehdään. Sovelluksen ApplicationRequest-luokassa tarkastetaan, minkä tyyppinen pyyntö on kyseessä, ja tämän pohjalta valitaan tiedostoon lisättävät XML-elementit.

KUVIO 4. ApplicationRequestin rakenne

5.3.1 Pyynnön allekirjoittaminen

Jotta pankki voisi varmistua, että sinne lähetetyt pyynnöt tulevat juuri oikealta lähettäjältä, tulee pyynnöstä käydä ilmi, kuka on

lähettäjä. Pankki hylkää automaattisesti kaikki pyynnöt, joiden autentikointi epäonnistuu. Käyttäjän autentikointi järjestelmässä on toteutettu XML-allekirjoituksella, jossa ApplicationRequest-dokumentti allekirjoitetaan tiedoilla, jotka ovat ainoastaan käyttäjän ja pankin tiedossa. Kuviossa 5 on esitetty XML-dokumentin digitaalinen allekirjoitus.

```
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
  <SignedInfo>
    <CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-
    20010315#WithComments"/>
    <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
    <Reference URI="">
      <Transforms>
        <Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
      </Transforms>
      <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <DigestValue>o9/bmBaH58Phw01oiQS/ttrP/sY=</DigestValue>
    </Reference>
  </SignedInfo>
  <SignatureValue>NwNRa...dTtMMqvg==</SignatureValue>
  <KeyInfo>
    <X509Data>
      <X509Certificate>MIIC9TC...nIv3xpHPU=</X509Certificate>
    </X509Data>
  </KeyInfo>
</Signature>
```

KUVIO 5. XML-dokumentin digitaalinen allekirjoitus

ApplicationRequest allekirjoitetaan käyttäen käyttäjän salaista avainta, joka on ainoastaan käyttäjän tiedossa. Pankin päässä käyttäjän tunnistus tapahtuu vertaamalla avainta pankin hallussa olevaan käyttäjän julkiseen avaimeen. Allekirjoituksen metodi on nimeltään Public Key Infrastructure, eli PKI, joka tunnetaan myös nimellä julkisen avaimen menetelmä.

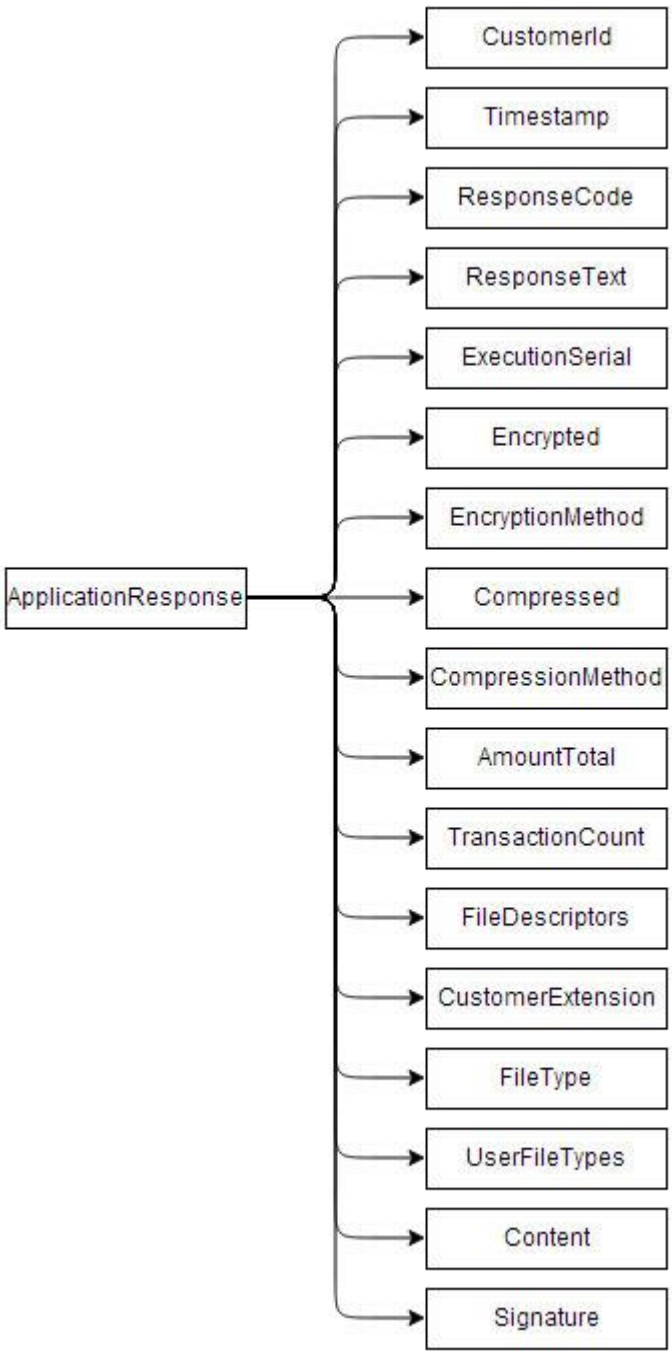
Julkisen avaimen menetelmässä sekä asiakas että pankki muodostavat oman avainparinsa, joka muodostuu julkisesta ja salaisesta avaimesta. Salaiset avaimet tulee säilyttää vain itsellään, mutta julkiset avaimet lähetetään toiselle osapuolelle. Toisen osapuolen julkista avainta käytetään viestin salaukseen, jonka voi purkaa ainoastaan kyseisen osapuolen salaisella avaimella.

Edellä mainitulla järjestelmällä voidaan varmistua siitä, että tieto tulee oikealta lähettäjältä. Tätä ei pidä sekoittaa XML-tiedostojen digitaaliseen allekirjoittamiseen, jossa käytetään viestin salaukseen

omaa salaista avainta ja jonka tarkoituksena on varmistaa lähetetyn viestin eheys, eli ettei viesti ole muuttunut matkalla.

5.4 ApplicationResponse

Web Services -yhteyden tietojen vastaanottoa varten tehtiin ApplicationResponse-luokka, joka ApplicationRequestin tavoin periytyy DomDocument-luokasta. Luokka käsittelee vastauksen, joka saadaan pankista, kun sinne tehdään pyyntö. ApplicationResponse voi sisältää XML-tagien sisällä yksinkertaisen kuittauksen, jonka pankki on lähettänyt tai esimerkiksi kokonaisen maksutiedoston, jos käyttäjä on sitä pankilta pyytänyt. ApplicationResponsen vastauken XML-rakenteen mahdolliset elementit on esitetty kuviossa 6.



KUVIO 6. ApplicationResponsen rakenne

5.5 TSSoapClient

Web Services -rajapinnan pyyntöjen lähettämistä varten tehtiin SOAP-asiakasohjelma. Ohjelma vastaa kommunikoinnista laskutusjärjestelmän ja pankin Web Services -palvelinohjelman välillä. Ohjelma toteutettiin omana luokkanaan, ja se periytyy PHP:n

SoapClient luokasta. Luokan tehtävänä on lähettää laskutusjärjestelmän antamat pyynnöt pankkiin ja vastaanottaa pankin lähettämät aineistot.

Luokan alustuksessa annetaan parametreina käyttäjän tunnistamisessa käytetty varmenne eli käyttäjän salainen ja julkinen avain. Varmenteen avulla kaikki lähetetyt SOAP-pyynnöt allekirjoitetaan. Kaikki pyynnöt joudutaan allekirjoittamaan, koska yhteys pankkiin ei sisällä minkäänlaista istuntoa, jossa autentikointi suoritetaan vain ensimmäisellä kerralla, vaan jokaista pyyntöä varten muodostetaan uusi SOAP-yhteys.

5.6 Apuluokat

5.6.1 BankAccount

BankAccount-luokan päätehtävä on käyttäjän varmenteen muodostus, tallennus ja haku. Luokka hakee alustuksessa käyttäjän nykyiset varmenteet tietokannasta ja purkaa niiden salauksen jos käyttäjällä on olemassa olevat varmenteet. Luokka pystyy myös luomaan uuden avainparin ja muodostamaan varmennepyynnön avainparin pohjalta. Kun varmennepyynnölle saadaan vastaus pankista, salataan käyttäjän salainen ja julkinen avain ja tallennetaan ne tietokantaan.

5.6.2 BankConnection

BankConnection-luokka vastaa oikeiden WSDL-tiedostojen hausta TSSoapClientille. Jokaisella pankilla on omat WSDL-tiedostonsa, jotka eroavat hieman toisistaan niin abstraktin kuin konkreettisen osion osalta.

5.7 TSWebServices

Kaikkien edellämainittujen luokkien käsittelyä varten tehtiin TSWebServices-luokka, joka on rajapintasovelluksen pääluokka. Luokan tehtävänä on yhdistää luokkien ApplicationRequest, ApplicationResponse ja TSSoapClient toiminta. Tavoitteena oli tehdä

pääluokasta mahdollisimman yksinkertainen, jotta kirjoitetun koodin määrä laskutusjärjestelmässä olisi mahdollisimman pieni rajapintasovelluksen osalta.

TSWebServices-luokka toimii rajapintana laskutusjärjestelmän ja Web Services -yhteyden välillä. Kaikki tieto, mitä halutaan lähettää, välitetään TSWebServices-luokan kautta, ja kaikki pankista palautunut tieto voidaan hakea suoraan saman luokan kautta. Kuviosta 7 nähdään toimenpiteet, joita vaaditaan laskutusjärjestelmän puolelta, kun halutaan tehdä pyyntö pankkiin, joka palauttaa tiedostolistauksen tietyllä aikavälillä, tietystä aineistosta ja tietyllä aineiston tilakoodilla. Koska TSWebServices käyttää samaa tietokantaa kuin laskutusjärjestelmä, alustuksessa ei tarvitse antaa kuin tunniste, joka yhdistää tietyn pankin ja käyttäjän ja TSWebServices hoitaa avainparien ja muiden käyttäjäkohtaisten tietojen haun. Tämän jälkeen kutsutaan vain downloadFileList-funktiota halutuilla parametreilla, ja ohjelma muodostaa automaattisesti ApplicationRequest-tiedoston ja SOAP-sanoman ja liittää tiedoston siihen ja tekee pyynnön pankkiin. Pankin vastaus tallentuu muodostettuun TSWebServices-luokan olioon, jonka jälkeen laskutusjärjestelmä voi tehdä halutut toimenpiteet vastauksen perusteella.

```
1  <?php
2  $ws = new TSWebServices($id);
3  $ws->downloadFileList($status, $filetype, FALSE, FALSE, $start, $end);
```

KUVIO 7. TSWebServices-pyynnön esimerkki

5.8 Ongelmakohdat

Ongelmakohdiksi WSO2-komponentin lisäksi Web Services -yhteyden toteutuksessa muodostuivat epäselvä dokumentaatio yhteyden palauttamista virhetilanteista, sekä pienet erot XML-dokumenttien muodostamisessa pankkien välillä. Virhetilanteissa pankin Web Services -palvelusta palautuu virhekoodi ja lyhyt koodin

kuvaus. Finanssialan keskusliiton ja joidenkin pankkien sivuilla on lista virhekoodeista, mutta kuten kuviosta 8 voi nähdä, niissä ei mainita kovinkaan kattavasti, että missä vika mahdollisesti voisi olla. Virhekoodeista ei aina selviä missä vika oikeasti on. Esimerkiksi kun yhteys pankkiin oli saatu muodostettua, niin pankin Web Services -palvelu palautti virhekoodia 18, joka viittaa sisällön virheelliseen digitaaliseen allekirjoitukseen. Usean päivän testauksen ja yhteyslokin tutkimisen jälkeen huomattiin, että base64-koodattu ApplicationRequest näytti oudolta ja tarkempi tarkastelu paljasti, että SOAP-asiakasohjelma suoritti base64-koodauksen automaattisesti ApplicationRequest-tiedostolle, koska sen tyyppi oli määritetty niin WSDL-tiedostossa. Koska tästä automatiikasta ei ollut tietoa, niin SOAP-asiakasohjelmalle välitettiin valmiiksi koodattu tiedosto, joten ApplicationRequest koodattiin kahteen kertaan. Pankin palvelimella ApplicationRequest-tiedoston koodaus purettiin, mutta koska koodaus oli suoritettu kahdesti, käsittelyyn siirtyi koodattu ApplicationRequest. Pankin palvelin siis käsitteli base64-koodattua tiedostoa XML-tiedostona ja palautti virheen koskien digitaalista allekirjoitusta.

Code	Name	Remarks
00	OK.	
01	Pending.	not used
02	SOAP signature error.	signature verification failed
03	SOAP signature error.	certificate not valid for this id
04	SOAP signature error.	certificate not valid
05	Operation unknown.	
06	Operation is restricted.	
07	SenderID not found.	
08	SenderID locked.	
09	Contract locked.	
10	SenderID outdated	
11	Contract outdated	
12	Schemavalidation failed.	
13	CustomerID not found.	
14	CustomerID locked.	
15	CustomerID outdated.	
16	Product contract outdated.	
17	Product contract locked.	
18	Content digital signature not valid.	
19	Content certificate not valid.	
20	Content type not valid.	
21	Deflate error.	
22	Decrypt error.	
23	Content processing error.	
24	Content not found.	
25	Content not allowed.	
26	Technical error.	
27	Cannot be deleted.	
28	[not used]	not used
29	Invalid parameters.	
30	Authentication failed	
31	Duplicate message rejected.	SOAP.Body.RequestHeader.SenderId + SOAP.Body.ReqhestHeader.RequestId
32	Duplicate ApplicationRequest rejected.	ApplicationRequest.CustomerId + ApplicationRequest.Timestamp

KUVIO 8. Web Services -kanavan virhekoodit

6 YHTEENVETO

Opinnäytetyön tavoitteena oli toteuttaa Web Services -yhteys tietosuunnan laskutusjärjestelmän ja pankkien välille. Yhteyden liittäminen laskutusjärjestelmään tuli olla mahdollisimman yksinkertaista ja nopeaa. Tavoitteet saavutettiin osittain, sillä rajapinta tukee tällä hetkellä täysin vasta yhtä pankkia. Muiden pankkien lisäys järjestelmään on kuitenkin helppoa, koska ohjelman peruslogiikkaa ei tarvitse muuttaa, vaan ainoastaan lisätä sanomien muodostussäännöt muille pankeille.

Web Services -yhteyden toteutusta jälkikäteen miettineenä, voin sanoa, että olisin tehnyt asioita toisin. Suurin kompastuskivi

projektissa oli WSO2-komponentin käyttöönotto. Web Services -tekniikat olivat minulle käytännön toteutuksissa täysin uusi asia, sillä en tietänyt niistä projektin alussa kuin perus toimintaperiaatteet. Tähän kun lisäsi vielä WSO2-komponentin ja digitaalisten allekirjoitusten opettelun niin haasteita oli tiedossa. Näistä haasteista kuitenkin selvittiin ja Web Services -yhteys allekirjoituksineen saatiin muodostettua Nordea-pankkiin testivarmenteilla. Tarkemmassa testauksessa kuitenkin ilmeni virhe, jota ei pystytty korjaamaan, koska se esiintyi WSO2-komponentin sisällä. Pyynnöt pankkiin menivät läpi, mutta WSO2-komponentti hylkäsi pankin vastaukset, koska se ei saanut laskettua oikeaa tarkistetta vastauksesta vastauksen pituudesta johtuen. Tämän vuoksi WSO2 jouduttiin hylkäämään ja itse yhteyden muodostamiseen ja allekirjoittamiseen tehtyä ohjelmaa ei pystynyt käyttämään, koska se oli ohjelmoitu lähes kokonaan käyttäen WSO2-komponentin ohjelmakirjastoja.

WSO2-komponentin hylkäämisen jälkeen koko Web Services -yhteyden toteutus jäi sivuun hetkeksi, koska ilmaantui muita töitä, joilla oli korkeampi prioriteetti. Web Services -yhteyden pariin palattiin vasta noin puoli vuotta WSO2-komponentin hylkäämisen jälkeen. Koska edellisestä kerrasta tämän aihepiirin parissa oli kulunut aikaa, jouduin kertaamaan ensin joitain asioita, jotka olivat unohtuneet ajan saatossa. Näitä asioita olivat muun muassa avainparien ja sertifikaattien muodostukset PHP:ssa.

Uudessa yhteyden toteutuksessa käytettiin yhteyden muodostamiseen PHP:n omaa SoapClient-luokkaa. Oman TSSoapClient-luokan tekemällä uusi Web Service -yhteys saatiin luotua jo muutamassa päivässä sisältäen kaikki dokumenttien digitaaliset allekirjoitukset ja vastausten käsittelyt.Pankkien SEPA-dokumentaatio osoittautui mielestäni myös puutteelliseksi, koska esimerkiksi pankista palautuneiden virheilmoitusten selitykset olivat suppeita tai niitä ei ollut lainkaan.

Vaikeuksienkin jälkeen ohjelmasta saatiin toimiva kokonaisuus. Rajapinta toteutettiin selkeällä luokkarakenteella, joten rajapinnan jatkokehitys on helppoa. Ohjelma on saatu jo osittain käyttöön ja sillä lähetetään verkkolaskuja pankkiin. Ohjelmaa käytetään myös yhden Tietosuunta Oy:n Windows-ohjelman verkkolaskujen välittäjänä.

LÄHTEET

2kmediat. 2012a. XML. Viitattu 23.11.2012. http://www.2kmediat.com/xml/syntaksi.asp

2kmediat. 2012b. XML. Viitattu 23.11.2012. http://www.2kmediat.com/xml/syntaksi-2.asp

Elkstein, M. 2008a. How Simple is REST?. Viitattu 25.11.2012. http://rest.elkstein.org/2008/02/how-simple-is-rest.html

Elkstein, M. 2008b. What is REST?. Viitattu 25.11.2012. http://rest.elkstein.org/2008/02/what-is-rest.html

Extensible Markup Language. 2008. W3C. Viitattu 23.11.2012. http://www.w3.org/TR/REC-xml/

Mitä SEPA tarkoittaa?. 2012. Finanssialan keskusliitto. Viitattu 29.10.2012. http://www.fkl.fi/teemasivut/sepa/sepan_maaritelma/Sivut/default.aspx

Rouse, M.2007. XML (Extensible Markup Language). Viitattu 23.11.2012. http://searchsoa.techtarget.com/definition/XML

Security and Message Specification for Financial Messages using Web Services. 2008. Finanssialan keskusliitto. Viitattu 20.11.2012. http://www.fkl.fi/teemasivut/sepa/tekninen_dokumentaatio/Dokumentit/WebServices_Messages_20081022_105.pdf

Service-oriented architecture and Web services. SOAP. Viitattu 27.11.2012. http://www.cs.tut.fi/kurssit/OHJ-5201/materiaali/4.pdf

Simple Object Access Protocol. 2000. W3C. Viitattu 22.11.2012. http://www.w3.org/TR/2000/NOTE-SOAP-20000508

Tossavainen S. 2012. Tietosuunta Oy. Viitattu 1.11.2012. http://www.tietosuunta.com/yri_yhteystiedot.php

W3Schools. Introduction to Web Services. Viitattu 21.11.2012. http://www.w3schools.com/webservices/ws_intro.asp

Web Services Architecture. 2004. W3C. Viitattu 23.11.2012. http://www.w3.org/TR/ws-arch/

Web Services Description Language. 2001. W3C. Viitattu 20.11.2012. http://www.w3.org/TR/wsdl

Yhtenäisen euromaksualueen toteutuminen Suomessa. 2012. Finanssialan Keskusliitto. Viitattu 20.11.2012. http://www.fkl.fi/teemasivut/sepa/tekninen_dokumentaatio/Dokumentit/SEPA_siirtymasuunnitelma_v5.pdf