VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Niilo Hakoniemi

Web Services käytännössä

Liiketalous
2018

VAASAN AMMATTIKORKEAKOULU
Tietojenkäsittely

TIIVISTELMÄ

Tekijä	Niilo Hakoniemi
Opinnäytetyön nimi	Web Services käytännössä
Vuosi	2018
Kieli	Suomi
Sivumäärä	34+7 liitettä
Ohjaaja	Kenneth Norrgard

Opinnäytetyön tehtävänä oli toteuttaa viitemaksuaineistojen nouto Osuuspankin yrityksille tarkoitetusta Web Services-palvelusta toimeksiantajan laskutusohjelman käytettäväksi ja prosessien automatisoimiseksi. Lisäksi toteutettiin varmenteiden haku pankin varmennepalvelusta Web Servicen käyttöä varten.

Web Service ja yrityksen ohjelmisto keskustelevat keskenään XML-tekniikkaan perustuvien SOAP-sanomien avulla. Lisäksi opinnäytetyössä käsitellään SOAP-sanomien allekirjoituksessa käytettäviä PKI-tekniikoita.

Tuloksena oli PHP-ohjelmointikielellä toteutettu viitesiirtojen automaattinen haku Osuuspankin Web Service palvelusta. Pankkien yhteisten sanoma- ja tietoturvamääritysten vuoksi työn tuloksia voi soveltaa, myös muiden pankkien Web Services-palveluihin

Avainsanat Web Services, XML, SOAP, WSDL

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Tietojenkäsittely

ABSTRACT

Author	Niilo Hakoniemi
Title	Web Services in Practice
Year	2018
Language	Finnish
Pages	34 + 7 Appendices
Name of Supervisor	Kenneth Norrgard

The objective of the thesis was to collect reference payment information from the Web Services of Osuuspankki bank and save the information into an invoicing software program. Another objective of the thesis was to make it possible to fetch Web Services certificate from the Osuuspankki certificate service.

The Web Service and the software communicate with each other using XML-based SOAP messages. In addition, the thesis reviewed PKI techniques which are used to sign SOAP messages.

The thesis result was an automatic reference payments fetching feature from the Web Services of Osuuspankki, which was built using PHP programming language. Due to the common message and data security requirements of the banks, the results of the thesis work can be applied to the Web Services of other banks as well.

Keywords Web Services, XML, SOAP, WSDL

SISÄLLYS

KUVIO- JA TAULUKKOLUETTELO

LIITELUETTELO

KÄSITTEET

C2B	Customer to Bank
PHP	PHP: Hypertext Preprocessor, ohjelmointikieli
SOAP	Simple Object Access Protocol, tietoliikenne protokolla, jota käytetään kommunikaatio välineenä Web Services-palveluun
SSH	Secure Shell, tietoliikenneprotokolla salattuun tietoliikenteeseen
Web Services	Verkossa oleva palvelu, jota voidaan käyttää internetin yli
WSDL	Web Service Description Language, kuvaa Web Services-palvelun toiminnot ja rakenteen
WS-kanava	Osuuspankin Web Services palvelu
XML	Extensible Markup Language, merkintäkieli

1 JOHDANTO

Suomessa toimivien pankkien ja yritysten ohjelmistojen maksuliikeaineistojen siirto on toteutettu Web Services-palvelulla. Pankit ovat yhdessä sopineet palvelun sanoma- ja tietoturvamäärityksistä, jolloin yhteydet perustuvat yhdessä sovittuihin standardeihin käytäntöihin. Yhteisiin käytöntöihin on siirrytty yhteisen euromaksualueen SEPAn siirtymisen myötä.

Opinnäytetyönä on toteutettu viitemaksuaineistojen nouto Osuuspankin Web Services palvelusta toimeksiantajan ohjelmiston käytettäväksi. Toteutuksessa on haettu myös Osuuspankin varmennepalvelusta Web Servicen käyttöön vaadittu pankin myöntämä varmenne.

Työn toimeksiantaja on taloustoimisto, jolla on oma verkossa toimiva laskutusohjelma. Opinnäytetyön aluksi tarkastellaan toimeksiannon taustaa, tarkoitusta, sekä kerrotaan työn tavoitteista.

Web Services on XML-tekniikoihin perustuva kokonaisuus, jonka kanssa yrityksen ohjelmisto pystyy keskustelemaan verkon yli. Teoriaosuudessa käsitellään näitä tekniikoita, sekä lisäksi avataan Web Services-palveluita yleisemmällä tasolla.

Teoriaosuuden jälkeen käsitellään pankkien yhteisiä Web Services käytänteitä sekä tarkastellaan työn toteutukseen tarvittavia osuuspankin määrityksiä. Näiden pohjalta on kuvattu tuotokset osiossa varmennepalvelusta haettavan varmenteen hakuprosessi, sekä toimeksiantajan ohjelmistoon luotu viitemaksuaineiston haun toiminto. Työ on toteutettu PHP-ohjelmointikielellä.

2 PROJEKTIN TAUSTA, TARKOITUS JA TAVOITTEET

Opinnäytetyön toimeksiantaja on taloustoimisto, jolla on verkossa toimiva oma laskutusohjelma. Yrityksen asiakkaat pystyvät laskutusohjelman avulla luomaan omille asiakkailleen laskuja. Laskujen lähetys on automatisoitu siten, ettei laskun luojan tarvitse itse lähettää laskuja, vaan laskutusohjelman kautta luodut laskut lähtevät eteenpäin automaattisesti. Ohjelma tarjoaa myös maksujen seurannan, jolloin ohjelmaa käyttävä yritys pystyy seuraamaan, onko laskuja maksettu.

Asiakkaiden laskut maksetaan toimeksiantajan asiakasvaratilille, mikä mahdollistaa reaaliaikaisen maksuvalvonnan, yli- ja alisuoritusten selvittämisen sekä viitteettömien suoritusten kohdistamisen. Tarvittavien toimenpiteiden jälkeen suorituksen tilitetään asiakkaille saman päivän aikana.

Viitemaksuaineisto muodostuu pankissa päivittäin. Osuuspankin viitemaksuaineistot on haettu Kultalinkki-pankkiyhteysohjelman avulla, mikä on tuottanut manuaalista työtä. Toimeksiantaja haluaa automatisoida tämän prosessin.

Prosessi pystytään automatisoimaan pankkiyhteyskanavassa, tästä eteenpäin käytetään nimitystä WS-kanava. Opinnäytetyön tarkoituksena on luoda yhteys toimeksiantajan ohjelmiston ja WS-kanavan välille ja toteuttaa toiminto, joka hakee viitemaksuaineistot automaattisesti. Koska haun tulee olla automaattinen ei sille tarvitse luoda käyttöliittymää. Toteutus tulee luoda PHP-ohjelmointikielellä, jolla toimeksiantajan ohjelmisto on myös toteutettu.

3 TEORIATAUSTA

3.1 XML

XML-merkintäkieli on kehitetty helppokäyttöiseksi, tekstimuotoisen rakenteisen tiedon kuvauskieleksi (Nykänen 2001, 8). Alun perin se kehitettiin elektronisia julkaisuja varten. XML-kielellä on myös merkittä rooli erilaisten tietojen siirtämisessä verkossa (W3C 2016). Rakenteinen tieto sisältää esimerkiksi osoitteita, transaktioita ja teknisiä piirustuksia.

3.1.1 XML dokumentti

XML-dokumentti koostuu *prologista* ja *elementeistä* eli tägeistä. XML muistuttaa hyvin paljon HTML merkintäkieltä. Erona kielillä on se, että XML kieltä käytetään järjestämään ja varastoimaan tietoa, kun taas HTML kielellä määritellään kuinka tietoa esitetään. Lisäksi erottavana tekijänä ovat tägit. HTML-kielessä tägit on määritelty valmiiksi ja XML-kielessä ne määritellään itse. (Tutorials Point India Private Limited)

XML-dokumentti alkaa prologilla. Prologissa määritetään tiedosto XML-muotoiseksi. Prologilla voi olla parametrit version, encoding ja standalone. *Version* parametri määrittää tiedoston XML-standardin käytettävän version. *Encoding* parametrillä määritellään mitä merkistöstandardia tiedosto käyttää. Parametrillä *standalone* määritetään, onko XML-tiedosto itsenäinen dokumentti vai käyttääkö se tietoa ulkopuolisesta lähteestä. Prologi ei ole pakollinen XML-dokumentissa. (Tutorials Point India Private Limited)

Elementti tarkemmin selitettynä on kokonaisuus, joka koostuu aloitustägistä ja lopetustägistä. Tägien väliin asetetaan tieto, joka halutaan dokumenttiin varastoida. Dokumentin ensimmäistä elementtiä kutsutaan *juurielementiksi*. Elementtejä voi olla sisäkkäin tai rinnakkain. Elementtien sisällä olevia elementtejä kutsutaan *lapsielementeiksi*. (Tutorials Point India Private Limited)

Elementeille pystyy antamaan myös attribuutteja. *Attribuutteja* käytetetään antamaan elementille lisätietoa ja niiden avulla pystytään erottamaan saman nimiset

elementit toisistaan. Attribuutin määrityksessä annetaan attribuutin nimi ja attribuutin arvo. Yhdellä elementillä voi olla useampia attribuutteja. (Tutorials Point India Private Limited)

```xml
<?xml version="1.0" encoding="UTF-8"?>
<varasto>
    <tuote kategoria="elektroniikka">
        <id>1</id>
        <nimi>puhelin</nimi>
        <hinta>200</hinta>
    </tuote>
    <tuote kategoria="urheilu">
        <id>1</id>
        <nimi>polkupyörä</nimi>
        <hinta>500</hinta>
    </tuote>
</varasto>
```

Kuvio 1. Esimerkki XML-dokumentista

3.1.2 XML Schema

XML-Schemaa käytetään kuvaamaan ja validoimaan XML-dokumentin rakennetta ja tietoa. Schemoilla määritetään dokumentin sallitut elementit, attribuutit ja tietotyypit. Schema on erillinen XML-pohjainen dokumentti, johon määritykset on tehty. (Tutorials Point India Private Limited)

Seuraavia määrityksiä voidaan tehdä schemoilla XML-dokumentille:

- Elementit ja attribuutit, joita voidaan käyttää XML-dokumentissa.
- Lapsielementtien määrää ja esiintymisjärjestys.
- Elementtien ja attribuuttien tietotyypit.
- Elementtien ja attribuuttien oletusarvot ja määrätyt arvot (fixed values). (w3schools)

Kuviossa 2 on esiteltynä schema-dokumentti. Schemassa määritellään, että XML-dokumentin juuri elementin tulee olla nimeltään ”note”, jolla on lapsielementtejä

”to”, ”from”, ”heading” ja ”body”. Lapsielementeille on määritelty hyväksyttäviksi tietotyypiksi merkkijono(string).

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="https://www.w3schools.com"
xmlns="https://www.w3schools.com"
elementFormDefault="qualified">

<xs:element name="note">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="to" type="xs:string"/>
      <xs:element name="from" type="xs:string"/>
      <xs:element name="heading" type="xs:string"/>
      <xs:element name="body" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

</xs:schema>
```

Kuvio 2. Esimerkki Schema dokumentista. (w3schools)

Scheman käyttöä XML-dokumentissa on mallinnettu kuviossa 3. SchemaLocation-attribuutilla määritetään mistä käytettävä schema löytyy. Tämä voi olla esimerkiksi verkko-osoite tai tiedostopolku. Kuvion 3 XML-dokumentin rakenne ja tietotyypit kuvin 2 scheman mukaiset.

```
<?xml version="1.0"?>

<note
xmlns="https://www.w3schools.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="https://www.w3schools.com/xml/note.xsd">
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

Kuvio 3. Schemaa käyttävä XML-dokumentti. (w3schools)

3.2 PKI

PKI eli *Public key infrastructure* on varmenteiden myöntämiseen, jakeluun, hallinnointiin ja ylläpitoon kuuluva kokonaisuus. (Järvinen 2003, 165.) *Sertifikaatti*, eli varmenne on tiedosto, johon on tallennettu varmenteen tietoja. Tietoverkkojen kautta tapahtuvassa tunnistamisessa, salauksessa ja sähköisen allekirjoituksen tekemisessä käytetään varmenteita. (Viestintävirasto 2017)

Jokaisella varmenteella on julkinen ja yksityinen avain, näitä kutsutaan avainpariksi. Julkinen avain on vapaasti jaettava osa sertifikaattia ja yksityinen avain pitäisi olla aina sillä henkilöllä tai taholla tallessa, jolle varmenne on myönnetty. (Centero 2012)

3.3 WEB SERVICES

Tässä luvussa käsittelen Web Serviceä yleisellä tasolla. Tarkoituksena ei ole keskittyä mihinkään tiettyyn Web Serviceen ja sen toimintaan. Tarkoitus on selventää, mitä Web Services tarkoittaa, ja mitä tekniikoita ja mahdollisia toiminnallisuuksia se mahdollistaa.

3.3.1 Web Services yleisesti

Web Service voi olla mikä tahansa ohjelma, johon saadaan yhteys internetin yli. Web Servicet tarjoavat ominaisuuksia, jotka ovat muitten ohjelmien käytettävissä internetin ja protokollien avulla (Chatterjee, Webber 2004). Kommunikaatio tapahtuu XML-pohjaisten standardoitujen sanomien avulla, jotka lähetetään useimmiten HTTP protokollaa käyttäen. Koska kommunikaatio tapahtuu XML-pohjaisilla sanomilla, niin Web Servicet eivät ole ohjelmointikielisidonnaisia, vaan käytännössä jokaisella ohjelmointikielellä pystyy tekemään Web Serviceen palvelupyynnön.

Web Serviceä ei kuitenkaan käytetä suoranaisesti itse minkään käyttöliittymän avulla. Yksinkertaisesti sanottuna Web Serviceä pyydetään suorittamaan tehtävä ja onnistuessaan Web Service suorittaa tehtävän ja palauttaa viestin onnistumisesta. Jos jokin menee vikaan Web Service palauttaa viestin epäonnistumisesta. Kui-

tenkaan missään vaiheessa pyynnön tekijä ei käytä Web Service-ohjelmaa, vaan se on rajapinta, josta toimintoja kutsutaan.

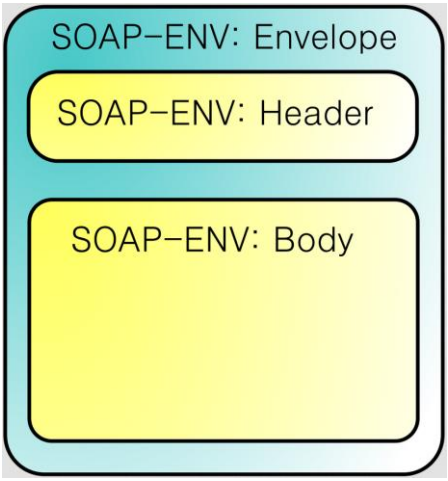
Tärkeimpinä käsitteitä ovat *pyyntösanoma* (request), jolla pyydetään Web Serviceä tekemään jotain. Sekä *vastaussanoma* (response), joka palauttaa dataa tai viestin onnistumisesta tai kumpaakin.

Web Service mahdollistaa useampia erilaisia toimintoja. *Funktio-toiminnoilla* voidaan esimerkiksi suorittaa palvelussa laskutoimituksia. Toiminnot voivat myös palauttaa tietoa esimerkiksi varastotilanteesta. *Prosessi-toiminnot* suorittavat prosesseja, kuten laskun hyväksyminen ja laskun lähettäminen. Erilaisia mahdollisuuksia toiminnolle on paljon, ja ne vaihtelevat aina palvelun tuottajan mukaan. (Chatterjee, Webber 2004, 3)

3.4 SOAP

SOAP-protokolla on XML-pohjainen dokumentti, jota käytetään sovellusten välillä vaihtamaan tietoa. Web Services-palvelupyynnöt ovat SOAP-sanomia. (Chatterjee, Webber 2004)

SOAP-sanoma voi sisältää neljä erilaista elementtiä. Nämä elementit ovat *Envelope-elementti*, *Header-elementti*, *Body-elementti* ja *Fault-elementti*. Näistä elementeistä muodostuu SOAP-sanoma, mutta kaikki elementit eivät ole kuitenkaan pakollisia.



Kuvio 4. SOAP kirjekuoren rakenne

SOAP Envelope-elementti on SOAP sanoman juurielementti, jonka avulla XML dokumentti tunnistetaan SOAP sanomaksi. Tämä elementti on pakollinen SOAP-sanomassa. (Chatterjee, Webber 2004, 77.)

SOAP Header-elementti on envelopen ensimmäinen lapsielementti. Header-elementti sisältää sovelluskohtaista informaatiota. Sillä voi olla lapsi elementtejä joita kutsutaan header-blockeiksi. Header ei ole pakollinen SOAP-kirjekuoressa. (Chatterjee, Webber 2004, 77.)

Body-elementti on pakollinen osa SOAP-kirjekuorta, joka sisältää sovelluskohtaista tietoa. Body-elementti tulee määritellä Header-elementin jälkeen SOAP-sanomassa ja se on pakollinen SOAP-sanomassa. (Chatterjee, Webber 2004, 81.)

Mikäli prosessin aikana tapahtuu virhe ja pyyntösanoma epäonnistuu, niin tulee Fault element Body-elementin sisälle, jolloin Fault-element palautuu vastaussanoman mukana. Fault-elementti sisältää tietoa siitä, mikä on ollut virhe. Virhe ilmoitetaan ennalta määritettynä koodina ja viestinä. Tämä elementti ei ole pakollinen. (Chatterjee, Webber 2004, 81.).

3.5 WSDL

WSDL eli Web Services Description Language on XML-pohjainen kieli, jolla Web servicen Toiminnallisuus ja käyttö. WSDL määritykset kuvaavat kuinka saa-

daan pääsy Web Serviceen, ja mitä toimintoja Web Servicessä on, sekä kuinka näiden kanssa kommunikoidaan. (Tutorials Point India Private Limited)

WSDL-tiedostoa käytetään yhdessä SOAP ja XML Scheman kanssa tarjoamaan Web Services toimintoja internetin yli. Ohjelma, jolla otetaan yhteys Web Serviceen lukee WSDL-tiedoston ja sen avulla tietää mitä toimintoja kyseisessä palvelussa on. WSDL käyttää tietotyyppien validoinnissa XML Schemaa ja SOAP-sanomaa käytetään kutsumaan WSDL-dokumentissa määritettyjä toimintoja Web Services palvelusta.

4 PROJEKTIN LÄHESTYMISTAPA JA PERIAATTEET

4.1 Pankkien Web Services palvelu

SEPA tarkoittaa yhtenäistä euromaksualuetta. SEPA on osa eurooppalaista yhdentymiskehitystä, josta eurovaluutta on yksi esimerkki. Euroopan ja EU-maiden välillä on suuria eroja maksukäytännöissä ja nopeudessa. Yhteisen euromaksualueen tarkoitus on edistää eurooppalaisten yritysten liikkumavapautta ja Euroopan taloudellista kilpailukykyä. Tällä hetkellä SEPA-alueeseen kuuluu 32 maata. (OP)

SEPAn tavoitteena on, että kaikki pankkipalveluja käyttävät voivat maksaa ja vastaanottaa maksuja samoin ehdoin, riippumatta siitä onko maksu maan sisäinen tai maiden välinen. Kaikille peruspalveluilla on tarkoitus olla yhtenäiset käytännöt ja standardit. (Finanssiala ry 2008)

SEPA muutos Suomessa on toteutettu Web Services-palvelulla. Web Services palvelun sanoma- ja tietoturvamääritykset on tehnyt yhteistyössä Nordea, OP-Pohjola Group ja Sampo pankki, jotka myös vastaavat sanomamääritysten ajanmukaisuudesta. Suomessa kaikki pankit käyttävät näitä yhdessä luotuja määrityksiä. Sanomamääritysten uusin versio löytyy aina Finanssialan Keskusliiton sivuilta, kun kirjoittaa hakukenttään ”web services”. Pankkien vastuulla on tehdä pankkikohtaiset ohjeet Web Servicen käyttöönotosta ja käytöstä. (Finanssiala ry 2008)

Seuraavat sanomamääritykset ja kuvaukset ovat kaikille pankeille samat.

- Web Sevices Security and message Specification
- Web Services Description Language
- ApplicationRequest Schema
- ApplicationResponse Schema
  (Finanssiala ry 2008)

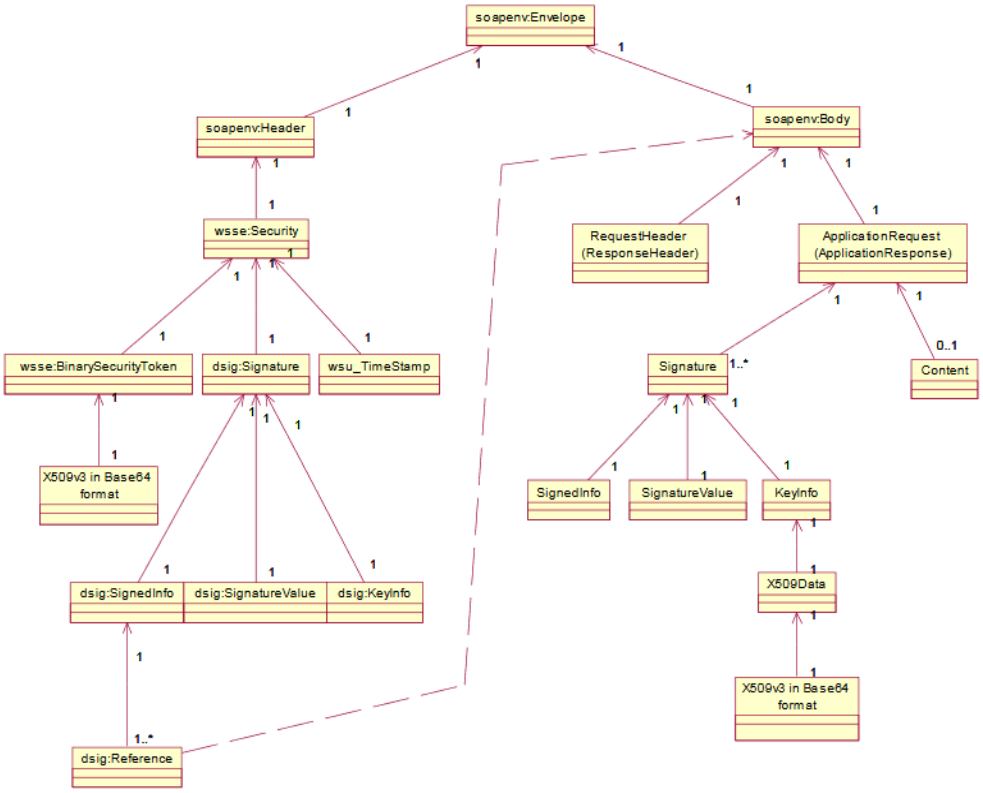
Seuraavat ohjeet ja kuvaukset pankit tekevät itse:

- PKI-dokumentaatiot, josta tulee löytyä tapa, jolla sertifikaatit toimitetaan asiakkaille.

- Testipalveluiden kuvaukset.
- Listaus tuetuista tietotyypeistä, tiedosto tyypeistä, toiminnoista ja palveluista.
- Pankkikohtaiset ohjeistukset vastaus elementtien käytölle.
  (Finanssiala ry 2008)

4.2 Palvelupyyntö

Palvelupyynnöt pankkien Web Service-palveluihin tehdään SOAP-sanomalla. SOAP-sanoman rakenne on kaikilla pankeilla sama.



Kuvio 5. Pankkiin lähetettävän SOAP-sanoman rakenne. (Finanssiala ry 2008)

Pankin Web Service-palvelimelle lähetettävän SOAP-sanoman juurielementtinä toimii SOAP Envelope-elementti, jolla on kaksi lapsielementtiä SOAP Header ja SOAP Body. SOAP Body-elementti on jaettu kahteen lapsielementtiin Request-Header ja ApplicationRequest. Jos sanoman mukana lähetetään tietoa pankin Web

Service palvelimelle, niin se sijoitetaan ApplicationRequestin sisälle Content elementtiin. Content elementin sisältö tulee olla aina base64 enkoodattu ja sen voi myös pakata. RequesHeader sisältää sanoman erikseen määriteltyä tietoa. (Finanssiala ry 2008)

Jokainen pankkiin lähetettävä SOAP-sanoma sisältää toiminnon, lähetettävän datan ja toiminnon vaatimat parametrit. Nämä tiedot asetetaan ApplicationResponse tai ApplicationRequest XML-elementin sisälle. Web Serviceä käyttävän asiakkaan ohjelmiston lähettämä SOAP-sanoma sisältää ApplicationRequest-elementin ja pankin Web Servicen palauttama SOAP-sanoma sisältää ApplicationRequest elementin. (Finanssiala ry 2008)

Jokainen ApplicationRequest-elementti allekirjoitetaan palvelupyynnön tekijän yksityisellä avaimella. Tätä allekirjoitusta pankki käyttää varmistavaan sanoman toimintopyynnön ja tietojen oikeellisuuden, sekä tietojen muuttumattomuuden. (Finanssiala ry 2008)

Liitteestä 1 löytyy ApplicationRequest-elementin rakenne ja liitteessä 2 puolestaan on ApplicationResponse-elementin rakenne.

4.3 OP-Pohjola-ryhmän Web Services kanava

WS-kanavaa käyttäen voidaan turvallisesti välittää konekielisiä aineistoja pankin ja yrityksen välillä. WS-kanan välityksellä eri järjestelmät voivat lähettää ja vastaanottaa C2B-maksuaineistoja, tiliotteita, e-laskuaineistoja ja niiden ilmoitussanomia. (OP ryhmä 2018)

SOAP sanoma WS-kanavaan lähetetään SSL-suojattuna HTTPS protokollaa käyttäen. SOAP-sanoman tulee olla allekirjoitettu yksityisellä avaimella. Palvelupyynnön tekevä järjestelmä muodostaa XML muotoisen ApplicationRequest palvelupyynnön, joka sisällytetään SOAP-sanomaan. ApplicationRequest sisältää pankkiin lähetettävän aineiston. (OP ryhmä 2018)

OP Pohjola-ryhmällä on käytössä neljä toimintoa, joita voidaan kutsua WS-kanavasta

- UploadFile – Tätä käytetään, kun halutaan lähettää tietoa pankkiin.
- DownloadFileList – WS-kanava palauttaa palvelupyynnön onnistuessa listauksen pankista haettavista tiedostoista.
- DownloadFile – Palauttaa palvelupyynnön onnistuessa halutun tiedoston. Palvelu vaatii, että ensin on haettu listaus tiedostoista, joita voidaan hakea. Jokaisella tiedostolla on yksilöllinen viite, jota haussa täytyy käyttää.
- DeleteFile – Poistaa tiedoston.
  (Finanssiala ry 2008)

4.4 Tunnistepalvelu

Sanomien ja palvelupyyntöjen muuttumattomuus ja aitous varmistetaan digitaalisella allekirjoituksella. Allekirjoitukset varmistetaan julkisella avaimella, eli toisin sanoen varmenteella. Jokainen pankki vastaa itse varmenteiden jakelusta. Tunnistepalvelun avulla luodaan ja hallinnoidaan varmenteita. (OP ryhmä 2018)

Ensimmäisenä kun yritys haluaa ottaa WS-kanava käyttöön, niin asiakkaan tulee luoda avainpari ja OP-Pohjola-ryhmän myöntämä varmenne. Varmenne hankitaan WS-kanavasta. Asiakkaan tulee hankkia pankista ohjelmistollensa siirtoavain, jota käytetään varmenteen hakemisessa. (OP ryhmä 2018)

Siirtoavaimen saa pankista tekemällä paikan päällä rekisteröinti henkilökunnan toimesta. Henkilökunta myös tarkistaa asiakkaan valtuutuksen ja tunnistaa henkilön. Pankista saa mukaansa asiakirjan, josta löytyy siirtoavaimen ensimmäinen osa, joka on pituudeltaan 8 numeroa. Siirtoavaimen toinen osa toimitetaan asiakkaalle joko tekstiviestillä tai kirjeitse asiakkaan valinnan mukaan. Siirtoavaimen toisen osan pituus on myös 8 numeroa. Ensimmäinen ja toinen osa siirtoavaimesta muodostavat yhdessä 16-numeroisen siirtoavaimen (OP ryhmä 2018)

Seuraavaksi tulee luoda avainpari. Avaimen pituus tulee olla 2048 bittiä ja algoritmi on RSA. SHA-1 on allekirjoituksen tiivistealgoritmi.

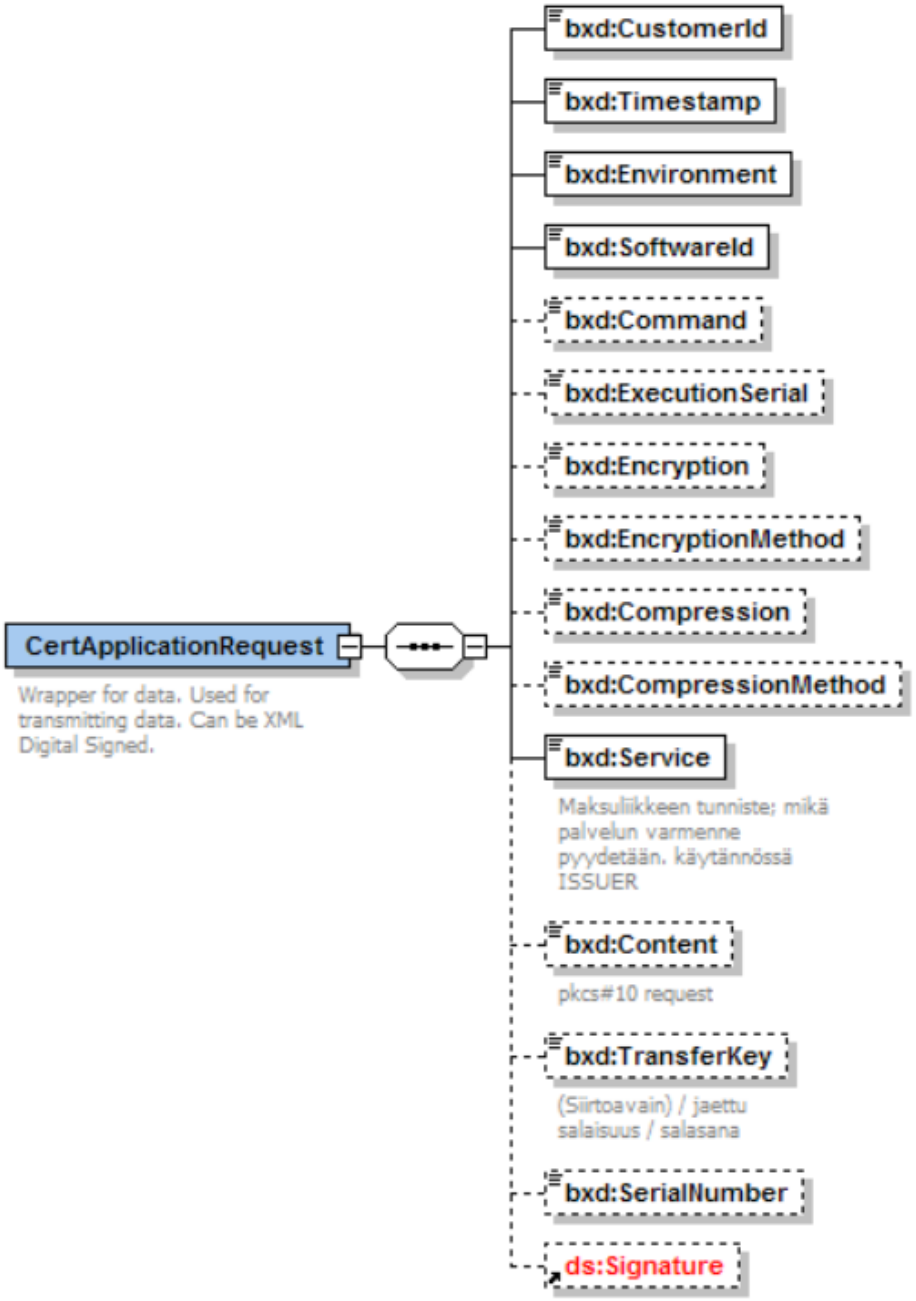
Varmenne haetaan WS-kanavasta lähettämällä SOAP-sanoma WS-kanavan tunnistepalveluun. Tunnistepalvelu lähetettävä SOAP sanoma luodaan tunnistepalvelun WSDL-dokumentissa kuvatulla tavalla. Tunnistepalvelun WSDL-tiedoston voi ladata osuuspankin sivulta. Ensimmäisellä kerralla kun haetaan pankista varmennetta, niin SOAP-sanomaa ei allekirjoiteta. Tähän tarvitaan pankista saatu käyttäjätunnus ja siirtoavain. (OP ryhmä 2018)

Pankista saatu varmenne on voimassa kaksi vuotta. Varmenne täytyy uusia, ennen kuin se vanhenee. Varmenteen uusiminen voidaan tehdä aikaisintaan 60 päivää ennen kuin varmenne vanhenee. Varmennetta uusittaessa joudutaan tekemään uusi avainpari.

Mikäli uusimista ei tehdä ajoissa ja varmenne pääsee vanhenemaan, niin joudutaan tällöin hakemaan pankin WS-kanavasta kokonaan uusi varmenne, jonka luomiseen tarvitaan uudet siirtoavaimet. Pankki ei ilmoita varmenteiden vanhenemisesta, vaan asiakas joutuu itse pitämään huolen siitä, ettei varmenne pääse vanhenemaan. Varmenteiden vanhenemista pystyy seuraamaan esimerkiksi asiakkaan ohjelmistolla. (OP ryhmä 2018)

Varmenteiden uusiminen suoritetaan samalla CertApplicationRequest-palvelupyynnöllä, erona kuitenkin, ettei siirtoavainta käytetä varmenteen uusimisessa. Palvelupyyntö allekirjoitetaan sen hetkisellä yksityisellä avaimella, johon on olemassa voimassa oleva varmenne. (OP ryhmä 2018)

Varmenteen haun palvelupyyntö on nimeltään CertApplicationRequest, joka on kuvattu tunnistepalvelun WSDL-tiedostossa. Palvelupyyntöön sijoitettavat elementit riippuvat siitä, ollaanko tekemässä ensimmäistä varmenteen hakua vai ollaanko uusimassa varmennetta. Kuviossa 6 on kuvattu CertApplicationRequest palvelupyynnön rakenne. (OP ryhmä 2018)



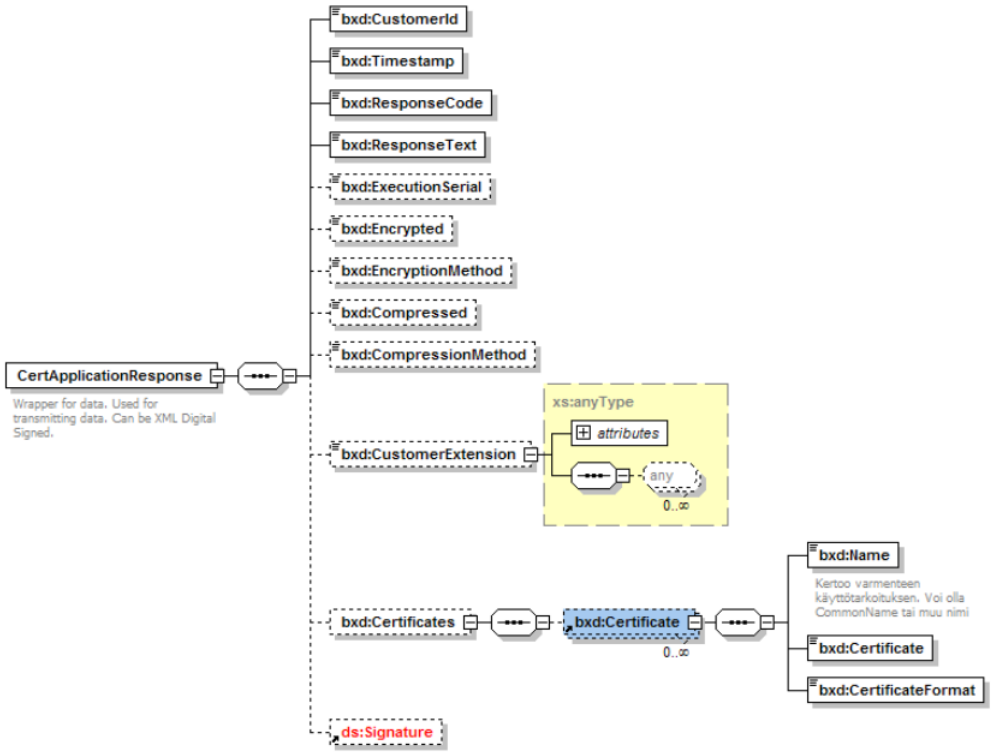
Kuvio 6. CertApplicationRequest palvelupyynnön rakenne (OP ryhmä 2018)

Kaikkia CertApplicationRequest -palvelupyynnön rakenteeseen kuuluvia elementtiä ei tarvitse käyttää. Osuuspankin palvelupyynnössä käytetyt elementit ovat:

- CustomerId – Varmenteen pyytäjän WS-kanvavan käyttäjä tunnus, joka on pituudeltaan 10 numeroa.

- Content – Varmennepyyntö, joka on Base64 enkoodattu
- TransferKey – Pankista saatu 16-numeroinen siirtoavain. Elementtiä käytetään vain, jos tehdään varmennepyyntöä ensimmäistä kertaa.
- Signature – XML-allekirjoitus, jota käytetään vain, jos uusitaan varmennetta.
- Timestamp – Aikaleima jolloin palvelupyyntö on muodostettu. Pakollinen tieto.
- Environment – Mikäli palvelupyyntö tehdään tuotanto ympäristöön, niin käytetään arvoa ”PRODUCTION”. Testiympäristössä käytetään arvoa ”TEST”.
- SoftwareId – Palvelupyynnön tehneen ohjelmiston nimi.
- Service – Arvoksi asetetaan ”MATU”

Palvelupyyntö varmennepalveluun palauttaa CertApplicationResponse-palveluvastauksen.



Kuvio 7. CertApplicationResponse palveluvastauksen rakenne (OP ryhmä 2018)

4.4.1 Testiympäristö

Ohjelmiston kehittäjille Osuuspankilla on WS-kanavan testausympäristö. WS-kanavan käyttöönotto vaatii, että yritys on tehnyt sopimuksen Osuuspankin kanssa WS-kanavan käytöstä. Testausympäristön käyttö vaatii myös erillisen sopimuksen tekemistä pankin kanssa. Sopimuksen teon yhteydessä tilataan testiympäristölle oma siirtoavain. Testausympäristöä varten täytyy hakea omat varmenteet varmennepalvelusta. (OP ryhmä 2018)

Testausympäristö on tuotantopalvelua vastaavaympäristö, jonne lähetetään palvelupyyntöjä tuotantopalvelun kaltaisesti. Testiympäristö myös palauttaa takaisin palveluvastauksia, jolloin pystytään tarkastelemaan palvelupyyntöjen toimivuutta. Kuitenkaan kaikkia tuotantopalvelun ominaisuuksia ei pysty testiympäristössä testaamaan. (OP ryhmä 2018)

5 PROJEKTIN TUOTOKSET

5.1 Varmenteen hakeminen varmennepalvelusta

5.1.1 Avainparin luominen

Kuka tahansa voi luoda itse varmenteen. Opinnäytetyössä varmenne luotiin toimeksiantajan järjestelmään käyttäen openssl-tekniikkaa. SSH-yhteyden avulla toimeksiantajan järjestelmän palvelimelle luotiin avainpari openssl-komennolla (Kuvio 8). Komento luo uuden salaisen avaimen ja DER-muotoisen varmennepyynnön, joka sisältää julkisen avaimen. Komennon suorituksen aikana täytyy alaiselle avaimelle antaa *passprhase*, jonka avulla salaista avainta voi käyttää tietojen salaamiseen.

Varmennepyyntöä varten täytyy antaa pyyntöön tallennettavia. tietoja. Tämä tietoosio on nimeltään subjekti. Näistä tiedoista Osuuspankille tehtävää varmennepyyntöä varten täytyy antaa kaksi pakollista tietoa. Jotka ovat Country Name ja Common Name. Country Name tiedoksi asetetaan ”FI” ja Common Name tiedoksi asetetetaan 10 merkin pituinen WS-kanavan käyttäjätunnus. (OP ryhmä 2018)

```
$ openssl req -new -keyout privatekey.pem -out pkcs10.csr -outform DER
```

Kuvio 8. Avainparin ja sertifikaatti pyynnön luonti

5.1.2 Palvelupyynnön luonti varmennepalveluun

Seuraavana avainparien luomisen jälkeen luodaan palvelupyyntö, jolla pankin myöntämä varmenne haetaan WS-kanavasta. Varmennetta haettaessa palvelupyyntö on nimeltään CertApplicationRequest. Sanoman muodostos aloitetaan tekemällä PHP-ohjelmointikielellä XML-dokumentti käyttäen SimleXMLElement-luokkaa, jonka juurielementiksi määritetään CertApplikationRequest. Juurielementille luodaan WS-kanavan määrittelyn mukaiset lapsielementit.

```php
$xml = new SimpleXMLElement('<CertApplicationRequest xmlns="http://op.fi/mlp/xmldata/"/>');
$xml->addChild('CustomerId', $your_customer_id);
$xml->addChild('Timestamp', date("c"));
$xml->addChild('Environment', 'PRODUCTION');
$xml->addChild('SoftwareId', $your_customer_id);
$xml->addChild('Service', 'MATU');
$xml->addChild('Content', base64_encode($pkcs10_certificate));
$xml->addChild('TransferKey', $your_trasfer_key);
```

Kuvio 9 Esimerkki CertApplicationRequest XML-tiedoston luonnista

CertApplicationRequest-elementin lisäksi täytyy muodostaa RequetHeader elementti, joka myös tulee SOAP-sanoman Body-elementin sisälle. SOAP sanoman muodostuksessa ja lähetyksessä käytetään PHP:n SoapClient luokkaa, jonka avulla SOAP sanoma muodostetaan ja sanoma lähetetään pankin WS-kanavaan.

SOAP-sanoman muodostuksessa luodaan ensin uusi SoapClient objekti. Objektille annetaan parametriksi varmennepalvelun WSDL-tiedoston, jonka avulla objekti pystyy luomaan oikean muotoisen SOAP-sanoman. Objektilla on soapCall-metodi, kutsumalla SOAP-sanoma luodaan ja lähetetään. Metodia kutsuttaesa sille annetaan kaksi parametriä. Ensimmäiseksi parametriksi annetaan toiminto, jota haluamme WS-kananvan tunnistepalvelussa käyttää. palvelussa käytettävät toiminnon löytyy WSDL-tiedostosta. Tässä tapauksessa haluamme käyttää varmenteenhakutoimintoa, joka on varmennepalvelussa ”getSertificate”.

Toisena parametrina annetaan metodille argumentit, jotka ovat tiedot, joiden avulla avulla SOAP-sanoma muodostetaan oikean muotoiseksi varmennepalvelun toimintoa varten. Tässä tapauksessa nämä argumentit ovat. ReguestHeader ja ApplicationRquest, nämä on myös määritelty WSDL-tiedostossa. Argumentit laitetaan array muuttujaan ja RequestHeaderille Annetaan sisäkkäiseen arrayhin parametrit joiden avulla objekti luo ReguestHeader-elementin. ApplicationRequestin tiedoksi annetaan aiemmin luotu XML-tiedoston CertApplicationRequest palvelupyynnön. SoapCall-metodi muodostaa SOAP-sanoman ja lähettää sen varmennepalveluun. Jos palvelupyyntö on onnistunut, niin metodi palauttaa varmennepalvelun SOAP-muotoisen vastaussanoman. Kuviossa vastaussanoma tallennetaan muuttujaan jatkokäsittelyä varten.

```php
$arguments = array(
    'RequestHeader' => array(
        'SenderId' => $your_customer_id,
        'RequestId' => time().'-'.uniqid(),
        'Timestamp' => date("c")
    ),
    'ApplicationRequest' => $xml->asXML()
```

Kuvio 10. Argumenttien array-muuttujan luonti

```php
$client = new SoapClient($wsdl);

$response = $client->__soapCall("getCertificate", array($arguments));
```

Kuvio 11. SoapClient objektin luonti ja objektin soapCall metodin kutsu

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
    <env:Header/>
    <env:Body>
        <opc:getCertificatein xmlns:opc="http://mlp.op.fi/OPCertificateService">
            <opc:RequestHeader>
                <opc:SenderId>1000012222</opc:SenderId>
                <opc:RequestId>123</opc:RequestId>
                <opc:Timestamp>2010-01-26T14:32:43.800+02:00</opc:Timestamp>
            </opc:RequestHeader>
            <opc:ApplicationRequest>PD94bWwgdmVy...
            GlvblJlcXVlc3Q+</opc:ApplicationRequest>
        </opc:getCertificatein>
    </env:Body>
</env:Envelope>
```

Kuvio 12. Esimerkki varmennepalveluun lähetetystä SOAP-sanomasta (OP ryhmä 2018)

5.1.3 Varmennepalvelun palveluvastaus

Varmennepalvelusta saatava vastaus tulee käsitellä, tarkistaa ja tallentaa asianmukaisesti. Kuviossa muuttujaan tallennettu vastaussanoman ResponseHeader-elemintissä oleva ResponseCode-elementin arvo tarkistetaan. Jos palvelupyyntö on onnistunut arvona 00. Jos arvona on jokin muu, niin palautetaan käyttäjän nähtäville ResponseCode ja ResponseText. Näiden perusteella pystyy katsomaan pankin virhesanomalistauksista tapahtuneen virheen syyn.

Seuraavaksi tarkistetaan ReguestId arvo. Tämä arvo tarkistetaan palvelupyynnössä lähetettyä RequestId arvoa vasten. Jos nämä arvot ovat samat voidaan silloin todeta, että saatu palveluvastaus on lähetetyn palvelupyynnön palveluvastaus. Palveluvastaus palauttaa Certificate elementissä base64 enkoodatun varmenteen. Varmenne base64 dekoodataan ja tallennetaan haluttuun turvalliseen paikkaan.

5.2 Viitemaksuaineistojen haku WS-kanavasta

Viitemaksuaineistojen haussa käytettiin apuna toimeksiantajan ohjelmassa olevaa WS-kanava luokkaa ja sen metodeja. Luokan objektia luotaessa haetaan ohjelmiston omat ohjelmistokohtaiset turvallisesti säilytettävät parametrit, varmenteet ja avaimet luokan käyttöön. Apuna on käytetty myös ohjelmiston tietokantaluokkaa, jonka avulla muodostetaan tietokantayhteys ja tallennetaan viitemaksuaineisto tietokantaan. Viitemaksuaineistojen haun PHP-skripti löytyy liitteestä 4.

Viitemaksuaineistojen haku WS-kanavasta voidaan neljään eri osakokonaisuuteen. Haettavien aineistojen listaukseen, aineiston hakuun, aineiston käsittelyyn tietokantaa varten ja aineiston tietokantaan tallentamiseen.

5.2.1 WS-kanavasta haettavien aineistojen listaus

Jokaisella haettavissa olevalla aineistolla on oma yksilöllinen viite. Aineistolistaus täytyy tehdä aina, jotta saadaan selville haettavan aineiston viite, jolla aineistohaku kohdistetaan oikeaan aineistoon.

Aineistonlistauksessa käytettiin apuna WS-kanava luokan *getFileList-metodia* (LIITE 5). Metodille lähetetään kaksi parametriä ”NEW” ja ”TL”. ”NEW” aineistolistauksessa sitä, että aineistolistaukseen haetaan vain sellaiset aineistot, joita ei ole vielä haettu WS-kanavasta. ”TL” on haettava aineisto tyyppi. Yhdessä näillä parametreilla listauksessa haetaan uusia viitemaksuaineistoja.

WS-kanava luokka ja sen metodit muodostavan ensin ApplicationRequest-elementin, joka allekirjoitetaan yksityisellä avaimella (LIITE 7). Tämä jälkeen muodostetaan RequestHeader-elementti, jotka sijoitetaan SOAP-sanoman body-

elementin sisälle. SOAP-sanoma allekirjoitetaan ja lähetetään WS-kanavaan HTTPS tekniikkaa käyttäen.

Palautussanoman FileDescriptions-elementin sisällä on jokaista WS-kanavasta haettavaa aineistoa kohden FileDescriptor-elementti. Toteutuksessa tehdään loop-pi, joka hakee jokaista FileDescriptor-elementtiä kohden elementin sisällä olevan FileReference- elementin arvon, joka on haettavan aineiston viite.

5.2.2 Aineiston haku

Kun ohjelmalla on tiedossa aineiston viite, voidaan aineisto hakea WS-kanavasta. Aineiston hakuun käytetään ohjelmiston WS-kanava luokan getFile-metodia (LIITE 6), jolle annetaan parametriksi automaattisesti haettu viite. GetFile-metodi luo SOAP-sanoman samalla tavalla kuin getFileList-metodi, mutta parametrit ja WS-kanavan käytetty toiminto on eri.

Palautusanoman ApplicationResponse-elementin sisällä oleva Content-elementti sisältää pakatun ja base64 koodatun viitemaksuaineiston, joka dekoodataan ja pu-retaan.

5.2.3 Aineiston käsittely

Ohjelmiston WS-kanavaan on liitetty useampi eri tili, jonka viitemaksuaineiston palautuu samaa aineistossa. Tarkoituksena on hakea vain toimeksiantajan asiakas-varatilin viitemaksuaineistot. Siitä johtuen jokainen palautunut viitemaksuaineisto joudutaan purkamaan osiin erätietueen kohdalta, jolloin saadaan eriteltyä eri tilien viitemaksuaineistot. Eritellyt aineistot käydyt läpi ja tarkistetaan vastaako aineis-tojen viitemaksujen tili yrityksen tiliin. Oikea aineisto tallennetaan ohjelmiston tietokantaan tietokantaluokkaa apuna käyttäen.

5.2.4 Lopputulos

Lopputuloksena on PHP-skripti, joka voidaan suorittaa automaattisesti haluamaan aikaan. Pankkien aineistojen luonti tapahtuu monesti ilta-aikaan, joten skriptin

suoritus on hyvä ajastaa vaikka yölle. Yöllä WS-kanava ja toimeksiantajan ohjelmisto on myös pienemmällä kuormituksella.

6 JOHTOPÄÄTÖKSET JA POHDINTA

Opinnäytetyön aihe, oli haastava ja vaati paljon paneutumista teoriatasolla työssä käytettäviin tekniikoihin. Työn tekeminen edellyttää ymmärrystä ohjelmoinnista, XML rakenteista. Ehdottomasti työn hankalin osuus oli teorian ja käsitteiden ymmärtämien.

Haastavinta toteutuksen kannalta oli pankin palveluiden dokumentaatio. Esimerkiksi osuuspankin dokumenteista ei löydy tarkkaa ohjetta tietyssä toiminnon palvelupyyntöön asetettavista elementeistä. Monesti työtä piti viedä eteenpäin ”yritä ja erehdy-menetelmällä”. Myöskin Osuuspankin WS-kanavan testiympäristön toiminnallisuuksien puuttuminen teki testauksesta haastavampaa.

Toimeksiantajan puolesta toteutukselle ei ollut tarkkaa aikataulua. Työ kuitenkin sujui aikataulullisesti hyvää vauhtia eteenpäin toteutuksen osalta. Raportointivaihe osoittautui kuitenkin ongelmalliseksi, ja siinä en pysynyt itselleni asettamissa aikataulussa. Paremmalla raportointivaiheen suunnittelulla olisin varmasti näiltä ongelmilta välttynyt.

Työn tuloksissa onnistuin mielestäni hyvin. Onnistuin luomaan viitemaksuaineistojen hakuun toimeksiantajan tarpeisin sopivan ratkaisun ja pystyin hakemaan varmenne palvelusta haettavan varmenteen.

Pankkien yhteiset sanoma- ja tietoturvamääritykset mahdollistavat tulosten hyödyntämisen helposti muidenkin pankkien Web Services palveluiden käyttöön.

LÄHTEET

Centero. 2012. PKI for Dummies. viitattu 4.5.2018.
https://www.centero.fi/blogi/pki-for-dummies/)

Chatterjee, S. Webber, J. 2004. Developing Enterprices Web Services An Architect's Guide. 1.painos. New Jersey. Prentice Hall PTR

Finanssiala ry. 2008. Security and Message Specification for Financial Messages using Web Services. Viitattu 5.5.2018.
http://www.finanssiala.fi/maksujenvalitys/dokumentit/WebServices_Messages_20081022_105.pdf

Järvinen, P. 2003. Salausmenetelmät. 1.painos. Jyväskylä. Docenco

Nykänen, O. 2001. XML. 1. painos. Jyväskylä. Docenco

OP ryhmä. Mikä on SEPA-maksu. Viitattu 4.5.2018
https://uusi.op.fi/henkiloasiakkaat/paivittaiset/maksaminen/sepa-maksu

OP ryhmä. 2018. Web Services -kanavan ja sen tunnistepalvelun sovellusohje. Viitattu 5.5.2018.
https://uusi.op.fi/documents/20556/65450/Yrityksen+pankkiyhteyskanavan+sovellusohje+.pdf/35f6967f-c31f-4807-9109-f7dda9da73d9

Tutorials Point India Private Limited. WSDL-introduction. viitattu 3.5.2018.
https://www.tutorialspoint.com/wsdl/wsdl_introduction.htm

Tutorials Point India Private Limited. XML-Attributes viitattu 6.5.2018.
https://www.tutorialspoint.com/xml/xml_attributes.htm

Tutorials Point India Private Limited. XML-Declaration. viitattu 6.5.2018.
https://www.tutorialspoint.com/xml/xml_declaration.htm

Tutorials Point India Private Limited. XML-Elements viitattu 6.5.2018.
https://www.tutorialspoint.com/xml/xml_elements.htm

Tutorials Point India Private Limited. XML-Overview. viitattu 6.5.2018. https://www.tutorialspoint.com/xml/xml_overview.htm

Tutorials Point India Private Limited. XML-Schemas viitattu 6.5.2018. https://www.tutorialspoint.com/xml/xml_schemas.htm

Viestintävirasto. 2017. Vahva sähköinen tunnistaminen, sähköinen allekirjoitus ja varmenne. Viitattu 4.5.2018. https://www.viestintavirasto.fi/kyberturvallisuus/sahkoinentunnistaminenjaallekirjoitus.html

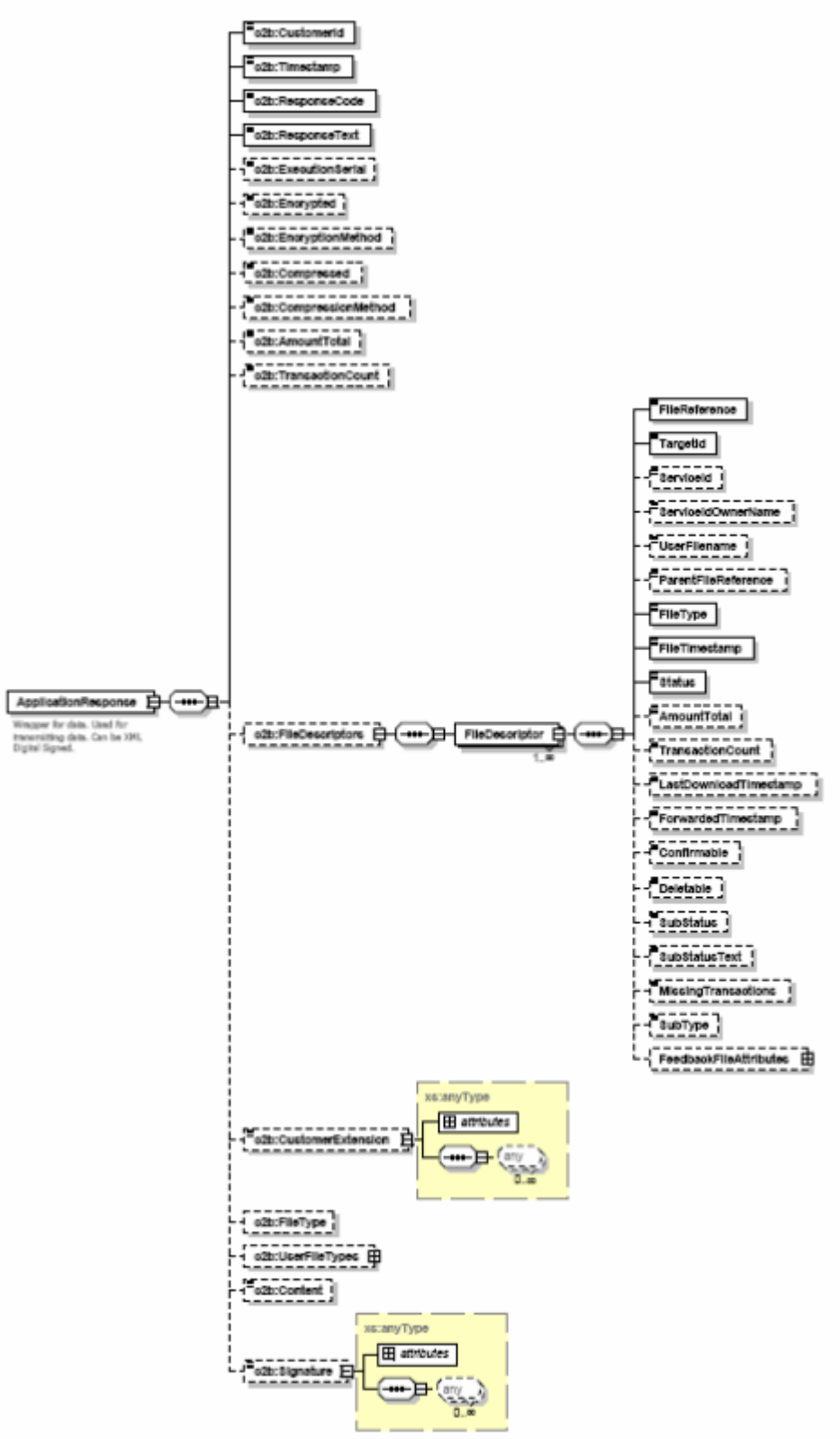
W3C. 2016. Extensible Markup Language (XML). viitattu 5.5.2018. https://www.w3.org/XML/

w3schools. XML Schema Tutorial. viitattu 6.5.2018. https://www.w3schools.com/xml/schema_intro.asp

w3schools. XSD How To? viitattu 6.5.2018. https://www.w3schools.com/xml/schema_howto.asp

LIITE 1 ApplicationRequest rakenne

LIITE 2 ApplicationResponse rakenne



LIITE 3 Viitemaksujen nouto skripti osa 1

```php
// WS-kanava objektin luonti
$opws = new OPWS();

$db=new Database();
if (!$db->open()) { echo "Tietokantavirhe!"; exit ;}


//haetaan listaus kaikista uusista viiteluettelo tiedostoista
$responseFileList = $opws->getFileList('NEW', 'TL');

// tallennetaan mahdolliset virheet
$errors = array();

//tarkistetaan onko file list pyynt□ onnistunut
if($responseFileList['code'] > 0) {
    //tehdään xml vastauksesta objekti
    $fileList = simplexml_load_string($responseFileList['msg']->ApplicationResponse);
    //lasketaan montako viiteluetteloaineistoa listauksessa l□ytyi
    if(count($fileList->FileDescriptors->FileDescriptor) > 0) {
        // haetaan jokainen viiteluettelo
        foreach($fileList->FileDescriptors->FileDescriptor as $fileDescriptor) {
            $responseFile = $opws->getFile($fileDescriptor->FileReference);
            //tarkistetaan onnistuiko viiteluettelon haku
            if($responseFile['code'] > 0) {
                //tehdään xml vastauksesta objekti
                $applicationResponse = simplexml_load_string($responseFile['msg']->ApplicationResponse);
                //haetaan vastauksen contentti ja dekoodataan, sekä puretaan se
                $file = gzdecode(base64_decode($applicationResponse->Content));

                $name = '';
                // tarkistetaan ettei teksti ole tyhjä
                if(!strlen($file)){
                    array_push($errors, "Response content is empty");
                    echo 'Response content is empty';
                }
                else{

                    //muodostetaan kantaan vietävä viiteluettelon nimi
                    if(substr($file, 0, 1 ) === '0'){
                        $name = 'viitemaksut'.date('ymd');
                    }
                    // splitataan viiteluettelo aina uuden erätietueen kohdalla. Luettelon ensimmäistä erätietuetta ei noteerata.
                    $fileSplit = explode(PHP_EOL.'0', $file);
                    // loopataan jokainen splitattu luettelo
                    foreach ($fileSplit as $fileRow){
                        // hajotetaan luettelo riveittän
                        $rowSplit = explode(PHP_EOL, $fileRow, 2);
                        print_r($rowSplit);
```

LIITE 4 Viitemaksujen nouto skripti osa 2

```
// tarkistetaan onko rivin 2 tilinumero oikea. Tilinumero poistettu opinnäytetyötä varten
if(substr($rowSplit[1],1,14) === 'xxxxxxxxxxxxxxxxxx'){
    // jos erätietueesta on hävinnyt 0 edestä, lisätään kantaan tallennuksessa puuttuva nolla erätietueeseen.
    if(substr($fileRow,0,1) !== '0'){
        echo '0'.$fileRow;
        //tallennetaan Luettelo palvelimelle tekstitiedostona
        file_put_contents('viitemaksut/'.$name.'.txt', '0'.$fileRow);
        // tallennetaan viiteluettelo kantaan
        $result = $db->sql("INSERT INTO poistettu opinnäytetyöstä");
        if (!$result){
            array_push($errors, $name." insert into database failed. File reference - ".$fileDescriptor->FileReference);
            echo $name." insert into database failed";
        }
    }
    else {
        echo $fileRow;
        //tallennetaan Luettelo palvelimelle tekstitiedostona
        file_put_contents('viitemaksut/'.$name.'.txt', $fileRow);
        // tallennetaan viiteluettelo kantaan
        $result = $db->sql("INSERT INTO poistettu opinnäytetyöstä");
        if (!$result){
            array_push($errors, $name." insert into database failed. File reference - ".$fileDescriptor->FileReference);
            echo $name." insert into database failed";
        }
    }
```

LIITE 5 getFileList metodi

```
function getFileList($status=null, $type=null, $startDate=null, $endDate=null) {
  $soap_function = 'downloadFileList';

  $doc = new DOMDocument('1.0', "utf-8");
  $applicationRequestNode = $doc->createElement('ApplicationRequest');
  $applicationRequestNode->setAttribute('xmlns', 'http://bxd.fi/xmldata/');
  $applicationRequestNode->appendChild($doc->createElement('CustomerId', $this->wsUsername));
  $applicationRequestNode->appendChild($doc->createElement('Timestamp', date("c")));
  if($startDate) {
    $applicationRequestNode->appendChild($doc->createElement('StartDate', $startDate));
  }
  if($endDate) {
    $applicationRequestNode->appendChild($doc->createElement('EndDate', $endDate));
  }
  if($status) {
    $applicationRequestNode->appendChild($doc->createElement('Status', $status));
  }
  $applicationRequestNode->appendChild($doc->createElement('Environment', $this->environment));
  $applicationRequestNode->appendChild($doc->createElement('Compression', 'true'));
  $applicationRequestNode->appendChild($doc->createElement('SoftwareId', 'xxxxxx'));
  if($type) {
    $applicationRequestNode->appendChild($doc->createElement('FileType', $type));
  }

  $doc->appendChild($applicationRequestNode);

  $applicationRequest = $this->signXML($doc);

  $response = $this->request($soap_function, $applicationRequest, $this->wsdl);

  return $response;
}
```

LIITE 6 getFile metodi

```php
// Get single file by reference
function getFile($fileReference) {
  $soap_function = 'downloadFile';

  $doc = new DOMDocument('1.0', "utf-8");
  $applicationRequestNode = $doc->createElement('ApplicationRequest');
  $applicationRequestNode->setAttribute('xmlns', 'http://bxd.fi/xmldata/');
  $applicationRequestNode->appendChild($doc->createElement('CustomerId', $this->wsUsername));
  $applicationRequestNode->appendChild($doc->createElement('Timestamp', date("c")));
  $applicationRequestNode->appendChild($doc->createElement('Environment', $this->environment));

  $fileReferencesNode = $doc->createElement('FileReferences');
  $fileReferencesNode->appendChild($doc->createElement('FileReference', $fileReference));
  $applicationRequestNode->appendChild($fileReferencesNode);

  $applicationRequestNode->appendChild($doc->createElement('Compression', 'true'));
  $applicationRequestNode->appendChild($doc->createElement('SoftwareId', 'xxxxxx'));

  $doc->appendChild($applicationRequestNode);

  $applicationRequest = $this->signXML($doc);

  $response = $this->request($soap_function, $applicationRequest, $this->wsdl);

  return $response;
}
```

LIITE 7 signXML metodi

```php
// Sign DomDocument (ApplicationRequest)
private function signXML($xml) {
  // Create a new Security object
  $objDSig = new XMLSecurityDSig();
  // Use the c14n exclusive canonicalization
  $objDSig->setCanonicalMethod(XMLSecurityDSig::EXC_C14N);
  // Sign using SHA-1
  $objDSig->addReference(
      $xml,
      XMLSecurityDSig::SHA1,
      array('http://www.w3.org/2000/09/xmldsig#enveloped-signature'),
      array('force_uri' => true) // To generate URI="" in ApplicationRequest <Reference>
  );

  // Create a new (private) Security key
  $objKey = new XMLSecurityKey(XMLSecurityKey::RSA_SHA1, array('type'=>'private'));

  // Load the private key
  $objKey->passphrase = $this->passphrase;
  $objKey->loadKey($this->privateKey, TRUE);

  // Sign the XML file
  $objDSig->sign($objKey);

  // Add the associated public key to the signature
  $objDSig->add509Cert($this->certificate);

  // Append the signature to the XML
  $objDSig->appendSignature($xml->documentElement);

  return $xml;
}
```