LAPIN AMK
Lapland University of Applied Sciences

Tradera-verkkopalvelu ja Arcticas SOAP -asiakassovellus

Jorma Vettainen

Kaupan ja kulttuurin koulutusalan opinnäytetyö
Tietojenkäsittelyn koulutusohjelma
Tradenomi

TORNIO 2014

TIIVISTELMÄ

LAPIN AMMATTIKORKEAKOULU, Kauppa ja kulttuuri

Koulutusohjelma:	Tietojenkäsittelyn koulutusohjelma
Opinnäytetyön tekijä:	Jorma Vettainen
Opinnäytetyön nimi:	Tradera-verkkopalvelu ja Arcticas SOAP -asiakassovellus
Sivuja:	34
Päiväys:	15.05.14
Opinnäytetyön ohjaaja:	Yrjö Koskenniemi

Tämän opinnäytetyön tavoitteena on toteuttaa Arcticas Oy:lle rajapinta, jonka kautta siirretään XML-muotoisia aineistoja toimeksiantajan oman tietokannan ja Traderan välillä käyttäen SOAP-protokollaa. Opinnäytetyö käsittelee myös Traderan Web Service -rajapinnan ominaisuuksia.

Opinnäytetyössä kuvaillaan näiden edellä mainittujen asioiden toteutusta, käytettyjä teknologioita, protokollia ja standardeja.

Tämä opinnäytetyö toteutettiin syyskuun 2013 – toukokuun 2014 välisenä aikana. SOAP-asiakassovelluksen kehityksessä käytettiin kehitystyökaluna Visual Studio 2010 -työkalua ja Visual Basic -ohjelmointikieltä.

Opinnäytetyö onnistui suhteellisen hyvin. SOAP-asiakassovellus saatiin toteutettua toimeksiantajan vaatimusten mukaisesti, mutta aikataulu venyi suunnitellusta kolmesta kuukaudesta lähes kahdeksaan kuukauteen.

Asiasanat: Verkkopalvelut, WSDL, SOAP, XML

ABSTRACT

LAPLAND UNIVERSITY OF APPLIED SCIENCES, Business and Culture

Degree programme:	Bachelor of Business Administration
Author:	Jorma Vettainen
Thesis title:	Tradera-Web service and Arcticas SOAP -client
Pages:	34
Date:	15.05.14
Thesis instructor:	Yrjö Koskenniemi

The objective of this thesis is to implement an interface for Arcticas Oy. The interface to be developed is to handle XML data transported between the client's own database and Tradera web service, using the SOAP protocol. The thesis also discusses the Tradera Web Service interface features.

This thesis describes the implementation of the above issues, used technologies, protocols and standards.

This thesis was carried out between September 2013 and May 2014. As a developing tool in this SOAP client application development was Visual Studio 2010 and Visual Basic programming language.

The thesis was relatively successful. SOAP client application was completed in accordance with the requirements of the client. However, the schedule planned was stretched from three months to almost eight months.

Keywords: Web Services, WSDL, SOAP, XML

SISÄLLYS

1 JOHDANTO

Jotta ymmärtäisi perin pohjin termin verkkopalvelu, on ymmärrettävä sen käyttämät teknologiat, protokollat ja standardit. Siksi käsittelen tässä opinnäytetyössäni aluksi teoriaa muun muassa verkkopalvelun protokollista, standardeista ja termeistä, ja havainnollistan keskustelua esimerkkien avulla. Vaikka nykyajan ohjelmistonkehitystyökalut helpottavat ohjelmistojen kehityksessä, on tärkeää ymmärtää myös ne asiat jotka eivät näy itse ohjelmoijalle. Näitä asioita voisi verrata esimerkiksi html-sivun tekemiseen. On olemassa hyviä ja jopa ilmaisia web-sivun luontityökaluja ja niiden avulla web-sivun tekeminen on nopeaa ja helppoa ilman, että täytyy ymmärtää html-kieltä. Ongelmia voi syntyä jos kuva tai teksti ei asetu oikealle paikalle tai kun teksti venyy kuvan alle vaikka sen pitäisi olla kuvan reunalla. Tällaisten ongelmien vuoksi jokaisen joka haluaa ymmärtää syvällisemmin web-sivuja on heidän ymmärrettävä html-kieltä jollain tasolla.

Tämä sama asia koskee myös verkkopalveluja. On ymmärrettävä verkkopalvelujen taustalla käytettävät teknologiat. Tässä opinnäytetyössä keskityn esittelemään kyseisiä teknologioita sekä teorian että käytännön tasolla. Käytännön osalta sovellan teoriaa käytäntöön esimerkein.

1.1 Toimeksiantajan esittely

Tein opinnäytetyöni hankkeistettuna Arcticas Oy:lle. Arcticas Oy on pieni kemiläinen perheyritys, joka myy erilaisia nahkatuotteita sekä ajovarusteita moottoripyöräilijöille. Suosituimpia tuotteita ovat nahkatakit, -housut, -liivit ja -laukut. Arcticas Oy:llä on niin sanottu kivijalkamyymälä Nahka Sipilä ja kaksi verkkokauppaa, joissa se myy tuotteitaan. Nämä verkkokaupat ovat Ziglara.com ja Motoasu.com.

1.2 Opinnäytetyön tavoite

Tämän opinnäytetyön tavoitteena oli toteuttaa Arcticas Oy:lle rajapintasovellus, jonka kautta voidaan siirtää XML-muotoisia aineistoja SOAP-viesteinä toimeksiantajan oman

toiminnanohjausjärjestelmän ja Traderan välillä käyttäen Traderan tarjoamaa verkkopalvelua.

Oppimistavoitteena opinnäytetyössäni oli oppia tietämään, jos ei perusteellisesti niin perusteet teknologioista, protokollista ja standardeista jotka perustuvat SOAP-verkkopalveluihin.

1.3 Opinnäytetyön tausta

Toimeksiantajan eli Arcticas Oy:n suunnitelmissa on ollut laajentaa myyntiä Ruotsiin. Ruotsissa toimii Tradera.se palvelu, joka on osa yhdysvaltalaista eBay huutokauppaa. Tämän palvelun kautta toimeksiantaja voisi myydä tuotteitansa. Ongelmaksi toimeksiantaja koki kuitenkin myytävien tuotteiden määrien päivityksen. Manuaalisesti se olisi hankalaa ja aikaa vievää, puhumattakaan inhimillisen virheen mahdollisuudesta. Tuotemäärien päivitykseen on käytettävissä Traderan tarjoama verkkopalvelu, jonka kautta tuotemäärien päivitys voitaisiin automatisoida tekemällä rajapintasovellus. Tämä rajapintasovellus hakisi toimeksiantajan toiminnanohjausjärjestelmän tietokannasta ajantasaiset tuotemäärät ja päivittäisi nämä Traderan verkkopalvelun kautta Traderan tietokantaan.

1.4 Työvälineet

Itse opinnäytetyön kirjoittamiseen olen käyttänyt OpenOffice Writer -ohjelmaa ja kaavioiden piirtoon OpenOffice Draw -ohjelmaa. SOAP-asiakassovelluksen kehityksessä olen käyttänyt Visual Studio 2010 -työkalua, koska se tukee suoraan verkkopalveluja. Ohjelmointikielenä on käytetty Visual Basic -ohjelmointikieltä. Yhteydenpitoon Traderan kanssa olen käyttänyt sähköpostia ja toimeksiantajan kanssa sähköpostin lisäksi puhelinta ja olemme pitäneet muutamia palavereita prosessin aikana.

2 VERKKOPALVELUT

Järvisen mukaan verkkopalveluilla (Web Services) tarkoitetaan hajautettuja ja ohjelmallisesti käytettäviä palveluita. Verkkopalveluiden keskeisiin periaatteisiin kuuluu, että niitä hyödyntävät sovellukset, eivätkä niinkään käyttäjät. (Järvinen 2002, 2.) Verkkopalveluilla voidaan tarkoittaa myös verkon kautta tarjottavaa palvelua tavallisille käyttäjille (Network Service), mutta tässä opinnäytetyössä keskitytään nimenomaan Web Service -toteutukseen.

Verkkopalvelut perustuvat rakenteelliseen XML-tiedonesitystapaan. Verkkopalveluiden välillä siirrettävä tieto on monesti XML-muotoista, ja näinollen tietojenkäsittelyllisesti verkkopalvelut nauttivat samanlaisista eduista kuin XML-dokumentit muutenkin. Koska XML on suunniteltu alkujaan laajennettavaksi merkintäkieleksi, verkkopalvelutkaan eivät ota kantaa itse tietoon mikä niiden välillä kulkee. Tyypillinen asiakasohjelma (client) joka käyttää verkkopalvelua, voisi esimerkiksi ottaa yhteyttä verkkopalvelua tarjoavaan palvelimeen (server) ja kysyä uusimpia uutisotsikoita tai ajantasalla olevia valuuttakursseja. Näin ollen Järvisen mukaan kyseessä on perinteinen pyyntö ja vastaus (request and response) arkkitehtuuri. (Järvinen 2002, 2.)

Järvisen mukaan verkkopalvelut perustuvat kolmeen standardiin. Nämä standardit ovat SOAP, WSDL ja UDDI ja on olennaista, että verkkopalveluissa sekä tiedon välitystavat, että esitystavat ovat standardoitu siten, että annettua tietoa voidaan käsitellä ohjelmallisesti. Koska verkkopalvelut eivät välitä siitä, millä välineillä ne on kehitetty tai millä ohjelmisto- tai laitealustoilla ne toimivat, verkkopalveluita voidaan pitää väline- ja alustariippumattomina, kunhan mainittuja standardeja noudatetaan. (Järvinen 2002, 3-5.)

Tässä opinnäytetyössäni tarkastelen tarkemmin XML:ää, SOAP:ia, WSDL:ää ja UDDI:a omissa luvuissaan.

2.1 XML

Koska esimerkiksi WSDL-dokumentit pohjautuvat vahvasti XML-muotoiseen esitystapaan, on mielestäni luonnollista lähteä tässä opinnäytetyössäni liikkeelle XML:stä. Traderan verkkopalvelun käyttämät XML-dokumentit noudattavat XML 1.0 -version määrityksiä, joten käyn läpi näitä määrityksiä niiltä osin kuin se on tarpeellista. W3C:n julkaisema määritysdokumentti löytyy osoitteesta http://www.w3.org/TR/REC-xml/. XML 1.0 -määritysdokumentista on käytössä jo viides painos ja siihen on tehty muutoksia viimeksi 7.2.2013.

XML on lyhenne sanoista eXtensible Markup Language. Järvisen mukaan XML on rakenteisten dokumenttien standardi ja se on tarkoitettu merkkipohjaisen tiedon tallentamiseen niin, että tiedon käsittely on automatisoitavissa (Järvinen 2002, 43). Puolestaan W3C:n mukaan ”The Extensible Markup Language (XML) is a simple text-based format for representing structured information” (W3C XML Essentials 2010, hakupäivä 8.5.2014). Eli XML on yksinkertainen tekstipohjainen muoto jäsennellyn tiedon esittämiseen.

Koska XML-dokumentti voi sisältää minkälaista tietoa tahansa, voisin julkaista koko opinnäytetyöni myös XML-dokumenttina. Tätä en kuitenkaan tee, mutta lyhyt XML-tiedosto opinnäytetyöstäni voisi näyttää seuraavanlaiselta:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<opinnäytetyö>
  <tietoja>
    <aihe>Tradera-verkkopalvelu ja Arcticas SOAP -asiakassovellus</aihe>
    <tekijä>Jorma Vettainen</tekijä>
    <ohjaavaopettaja>Yrjö Koskenniemi</ohjaavaopettaja>
    <oppilaitos>Lapin Ammattikorkeakoulu</oppilaitos>
  </tietoja>
</opinnäytetyö>
```

Edellä mainittu XML-dokumentti on niin sanotusti hyvin muodostettu (well-formed), mutta ei validi (valid) koska se ei sisällä rakennekuvausta. Esimerkissä on ensimmäisellä rivillä prosessointiohje joka kertoo XML-kielen version ja dokumentin

käyttämän merkistön. Merkistö on tässä tapauksessa ISO-8859-1. Tällä merkistöllä voidaan käsitellä lähes kaikkia länsieurooppalaisissa kielissä esiintyviä erikoiskirjaimia. Ilman merkistön määrittelyä Ä ja Ö kirjaimet aiheuttaisivat virheen dokumenttia käsiteltäessä. Juurielementtinä on opinnäytetyö. Lisäksi esimerkissä on tietoja-elementti ja sen lapsielementteinä aihe, tekijä, ohjaava opettaja ja oppilaitos-elementit. Itse tieto on elementtien välissä.

XML-dokumentti on silloin hyvin muodostettu, kun se sisältää vain yhden juurielementin ja kaikki muut elementit alkavat ja päättyvät saman elementin sisällä. Näinollen elementit ovat niinsanotusti tasapainossa. Lisäksi XML-dokumentin on vastattava XML-määrittelyn asettamia hyvin muodostetun dokumentin rajoituksia, jotka löytyvät W3C:n sivuilta osoitteesta http://www.w3c.org. Kolmanneksi hyvinmuodostetun XML-dokumentin jokainen jäsennetty entiteetti, johon dokumentissa viitataan, täytyy olla hyvin muodostunut. (Walkama & Laakkonen 2004, 4.)

XML-dokumentti on validi eli rakennekuvausten mukainen, kun se on hyvinmuodostettu ja siihen on liitetty rakennekuvaus. Lisäksi sen on noudatettava tätä rakennekuvausta ja muita XML-määrittelyissä asetettuja validiusrajoituksia. (Walkama & Laakkonen 2004, 4.)

XML-dokumentin validius tai hyvinmuodostuneisuus voidaan tarkistaa niiden prosessoimiseen tarkoitetuilla parsereilla. Validoinniksi kutsutaan prosessia, jossa XML-dokumentin validius tarkistetaan. Kun parseri prosessoi XML-dokumenttia se tarjoaa samalla ohjelmille rajapinnan dokumentin rakenteeseen ja sisältöön. (Walkama & Laakkonen 2004, 4.)

2.2 WSDL

WSDL (Web Services Description Language) on alun perin Microsoftin ja IBM:n kehittelemä, jonka viimeisimmän version 2.0 W3C on hyväksynyt standardiksi vuonna 2007 (W3C WSDL, hakupäivä 21.3.2014). W3C eli World Wide Web Consortium on kansainvälinen yhteisö, missä jäsenorganisaatiot, päätoiminen henkilökunta ja suuri yleisö työskentelevät yhdessä kehittääkseen Web-standardeja (W3C Consortium,

hakupäivä 22.3.2014). Käsittelen tässä luvussa WSDL-määrityksiä niiltä osin, kun se on Traderan verkkopalvelun ymmärtämiseksi tarpeellista.

WSDL on kieli, jolla kuvataan esimerkiksi jotakin tiettyä SOAP-palvelua. WSDL-dokumentti on puolestaan XML-muotoinen dokumentti, joka kertoo muun muassa jonkin SOAP-palvelun URI-osoitteen, sen tukemat metodit (method) ja näiden kutsuparametrit (call parameters). WSDL-dokumentti siis kertoo sitä ymmärtävälle ohjelmistolle, mistä SOAP-palvelu löytyy ja mitä palveluita se tukee. (Järvinen 2002, 3.)

WSDL-dokumentit ovat yleensä hyvin pitkiä ja niiden rakenne vaikuttaa tottumattomaan silmään sekavalta. Dokumentit ovat kuitenkin jaettavissa selkeisiin lohkoihin. Järvisen mukaan WSDL-dokumenttien 5 tärkeintä lohkoa ovat, types-, message-, portType-, binding- ja service-lohkot. (Järvinen 2002, 61-62.)

WSDL-dokumentin alussa eli juurielementissä (root element) määritellään kuitenkin käytetyt nimiavaruudet (namespaces) URI-osoittein. Nimiavaruuksista W3C on julkaissut suosituksen (Namespaces in XML), ja niistä on kirjoitettu kokonaisia kirjojakin. En kuitenkaan tässä opinnäytetyössäni käy nimiavaruuksia tarkemmin läpi, mutta lyhyesti määriteltynä Walkaman ja Laakkosen mukaan nimiavaruus on eräänlainen joukko yksiselitteisesti ja ainutlaatuisesti nimettyjä tunnisteita (Walkama & Laakkonen 2004, 9). XML-dokumentissa, joka WSDL-dokumenttikin on, nimiavaruudet tarjoavat menetelmän, jonka avulla elementtien ja attribuuttien nimet voidaan sovelluksissa yksikäsitteisesti nimetä (Walkama & Laakkonen 2004, 16).

WSDL-dokumenttien alusta voi löytyä myös tyyppimääritykset eli types-elementti, jonka avulla voidaan määritellä omia rakenteisia tietotyyppejä. Types-elementti ei kuitenkaan ole aina käytössä, sillä usein sisäänrakennetuilla tietotyypeillä pärjätään varsin pitkälle. (Järvinen 2002, 61, 64.)

Message-lohkojen avulla WSDL-dokumentissa määritellään viestit, joita voidaan välittää sovellusten välillä. Nämä viestit ovat rakenteellisia ja käyttävät types-lohkossa määriteltyjä tai sisäänrakennettuja tietotyyppejä. Jokaisella viestillä on yksi tai useampi

osa (part). (Järvinen 2002, 61, 62, 71.) Esimerkiksi koodissa 1 Traderan GetOfficalTime-metodin message-lohko sisältää yhden osan.

```
<wsdl:message name="GetOfficalTimeSoapIn">
<wsdl:part name="parameters" element="tns:GetOfficalTime"/>
</wsdl:message>
```

Koodi 1. Traderan GetOfficalTime message-lohko (Tradera Public Service WSDL 2010, hakupäivä 23.3.2014).

PortType-lohkossa määritellään ne operaatiot, joita palvelu tukee ja jokainen operaatio sisältää sekä syöte- että tulosteviestin (input message, output message) (Järvinen 2002, 62). Alla olevassa koodissa 2 on GetOfficalTime-operaatio portType-lohkossa.

```
<wsdl:operation name="GetOfficalTime">
<wsdl:input message="tns:GetOfficalTimeSoapIn"/>
<wsdl:output message="tns:GetOfficalTimeSoapOut"/>
</wsdl:operation>
```

Koodi 2. Traderan GetOfficalTime operaatio portType-lohkossa (Tradera Public Service WSDL 2010, hakupäivä 23.3.2014).

WSDL-dokumentin lopusta löytyy service-lohko. Tämä lohko sisältää porttimäärityksiä (port specification), jotka kertovat WSDL-dokumenttia prosessoivalle sovellukselle sen verkko-osoitteen mistä tietyt palvelut löytyvät (Järvinen 2002, 62). Kuten kuvasta 1 nähdään, Traderan PublicServicen service-lohkosta löytyy vain yksi palvelu PublicService, mutta neljä erilaista lapsielementteinä olevaa port-elementtiä, jotka viittaavat binding-attribuuttinsa kautta johonkin aikaisemmin määritettyyn sidontaan ja sitä kautta johonkin porttityyppiin.

Kuva 1. Traderan PublicService WSDL-dokumentin service-lohko (Tradera Public Service WSDL 2010, hakupäivä 21.3.2014).

WSDL-dokumentin sidos-lohkossa (binding) määritellään muun muassa teknisiä yksityiskohtia, joilla viestejä käytännössä välitetään ja esimerkiksi määritellään viestikohtaisesti, mitä tiedon esitystapaa (data format) käytetään (Järvinen 2002, 62.)

2.3 SOAP

SOAP (Simple Object Access Protocol) on tiedonvälitykseen käytettävä protokolla. Viimeisin W3C:n SOAP-standardista julkaistu versio on 1.2.

SOAP-määritykset koostuvat seuraavasta neljästä osasta: SOAP-kirjekuoresta (envelope), tiedon koodaustavasta (encoding), RPC (Remote Procedure Call)-etäkutsujen määrityksistä ja HTTP (HyperText Transfer Protocol)-sidonnasta. Microsoft määrittelee RPC:n seuraavalla tavalla ”A process, such as a program or task, that requests a service provided by another program (Microsoft TechNet 2003, hakupäivä 9.5.2014).” eli RPC on ohjelman tai tehtävän prosessi, joka pyytää palvelua toiselta ohjelmalta.

SOAP:issa määritellyt viestit ovat yhdensuuntaista tiedonsiirtoa, joista pystytään kokoamaan suurempia kokonaisuuksia, kuten pyyntö ja vastaus-pareja. SOAP ei itsessään ota kantaa sen suhteen, miten viestit tulisi välittää verkossa. Tiedonsiirtoon voidaan käyttää HTTP:n sijaan esimerkiksi sähköpostia tai pelkkää TCP/IP:tä. Koska

SOAP-viestejä kuljetetaan yleisimmin HTTP:n päällä SOAP määritteleekin HTTP-sidonnan (HTTP binding). HTTP-sidonta kertoo, kuinka HTTP:tä tulisi käyttää SOAP-viestien välittämiseen. (Järvinen 2002, 79.)

Kun SOAP-client lähettää pyynnön Traderan verkkopalvelulle, HTTP-protokollan mukana kulkeva pyyntö voisi näyttää samanlaiselta kuten esimerkkinä olevassa koodissa 3. Esimerkissä SOAP-pyyntö koskee GetOfficalTime-metodia.

```
POST /v3/publicservice.asmx HTTP/1.1
Host: api.tradera.com
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://api.tradera.com/GetOfficalTime"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <AuthenticationHeader xmlns="http://api.tradera.com">
      <AppId>int</AppId>
      <AppKey>string</AppKey>
    </AuthenticationHeader>
    <ConfigurationHeader xmlns="http://api.tradera.com">
      <Sandbox>int</Sandbox>
      <MaxResultAge>int</MaxResultAge>
    </ConfigurationHeader>
  </soap:Header>
  <soap:Body>
    <GetOfficalTime xmlns="http://api.tradera.com" />
  </soap:Body>
</soap:Envelope>
```

Koodi 3. Sample SOAP 1.2 request (Tradera GetOfficalTime method 2010, hakupäivä 20.3.2014).

Ensimmäiset neljä riviä kuuluvat HTTP-protokollan mukaisiin tietoihin. Tässä esimerkissä käytetään HTTP-protokollan metodia POST, joka osoittaa resurssiin "/v3/publicservice.asmx" ja HTTP-protokollan versio on 1.1. Otsikkotiedoista (header)

löytyy muun muassa kohteena oleva palvelin (host) sekä tietosisällön tyyppi (content-type) ja tietosisällön koko (content-length).

Itse SOAP-protokollan mukainen tieto kulkee HTTP:n runko-osassa (body). SOAP-pyyntö on XML-muotoinen, joten ensimmäinen rivi kertoo XML:n version ja käytetyn koodaustavan, joka on tässä esimerkissä utf-8. Seuraava rivi aloittaa SOAP-kirjekuoren (envelope), jonka sisällä muut tiedot pyynnöstä ovat.

SOAP-kirjekuoren runko-osassa (body) mainitaan haluttu metodi, joka tässä esimerkissä on GetOfficalTime. Jos ja kun edellä oleva SOAP-pyyntö on kelvollinen ja palvelin ymmärsi pyynnön, palvelimen vastaus olisi koodin 4 kaltainen.

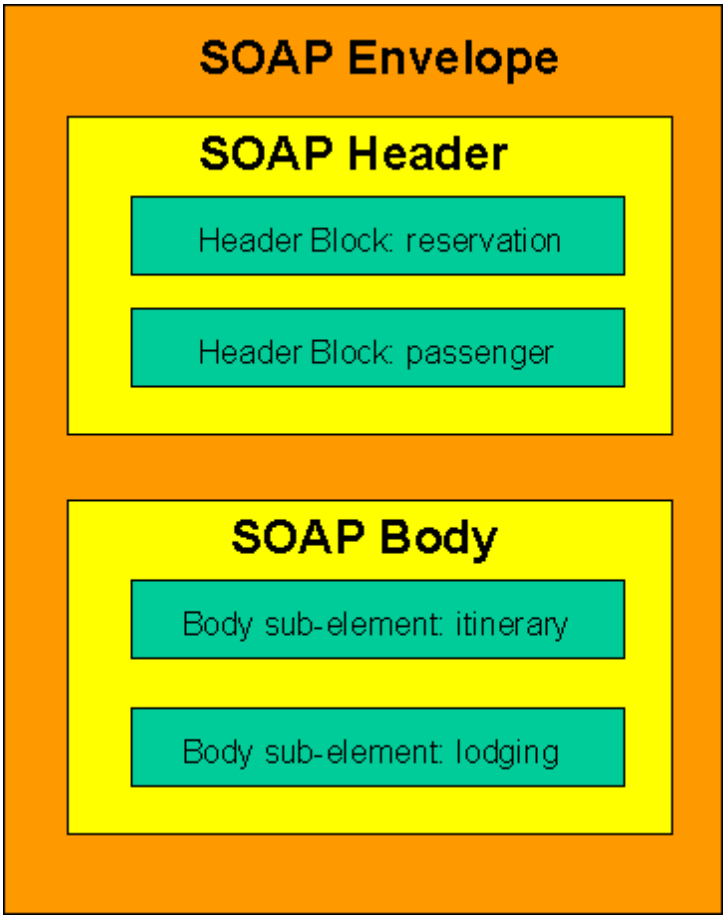
```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <GetOfficalTimeResponse xmlns="http://api.tradera.com">
      <GetOfficalTimeResult>dateTime</GetOfficalTimeResult>
    </GetOfficalTimeResponse>
  </soap:Body>
</soap:Envelope>
```

Koodi 4. Sample SOAP 1.2 response (Tradera GetOfficalTime method 2010, hakupäivä 20.3.2014).

Palvelimen vastaus on myös pyynnön mukaan jaettu kahteen osaan eli HTTP:n mukaisiin tietoihin sekä SOAP-protokollan määrittelemään runkoon eli XML-dokumenttiin. Ensimmäinen rivi kertoo, että pyyntö käsiteltiin onnistuneesti, joten HTTP-protokollan mukainen ns. tilakoodi (status code) on "200 OK". Kuten pyynnössäkin, runko-osan muodostaa SOAP-protokollan kirjekuori. Metodin GetOfficalTime-paluuarvo löytyy GetOfficalTimeResponse-elementin sisältä.

SOAP-viestejä voidaan kutsua myös kirjekuoriksi (envelope), joka on aina XML-muotoinen dokumentti. SOAP-kirjekuori alkaa aina Envelope-elementillä, joka on pakollinen, ja mikäli elementin määrityksissä on käytetty nimiavaruuksia, on nimiavaruuden oltava muotoa http://schemas.xmlsoap.org/soap/envelope/. (Järvinen 2002, 80.)

Kaavio 1. Esimerkki SOAP-kirjekuoresta (W3C SOAP 2007, hakupäivä 20.3.2014).



Alla koodissa 5 on yksinkertainen esimerkki SOAP-kirjekuoresta joka on osa Traderan GetOfficalTime-metodin SOAP-pyyntöä. SOAP-kirjekuoressa on oltava pakollinen Body-elementti, mutta se voi sisältää myös valinnaisen Header-lementin, kuten alla olevassa esimerkissä on esitetty.

```
<soap12:Envelope          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://www.w3.org/2003/05/soap-envelope">
  <soap12:Header>
    <AuthenticationHeader xmlns="http://api.tradera.com">
      <AppId>int</AppId>
      <AppKey>string</AppKey>
    </AuthenticationHeader>
    <ConfigurationHeader xmlns="http://api.tradera.com">
      <Sandbox>int</Sandbox>
      <MaxResultAge>int</MaxResultAge>
    </ConfigurationHeader>
  </soap12:Header>
  <soap12:Body>
    <GetOfficalTime xmlns="http://api.tradera.com" />
  </soap12:Body>
</soap12:Envelope>
```

Koodi 5. Sample SOAP-envelope (Tradera GetOfficalTime method 2010, hakupäivä 20.3.2014).

SOAP-kirjekuori alkaa kuitenkin aina Envelope-elementillä, jossa määritellään dokumentin nimiavaruudet. Tässä SOAP-kirjekuoren esimerkissä on siis valinnainen otsikko (header), joka määritellään Header-elementin avulla. Tämän elementin tarkoituksena on tarjota sovelluksille tavan laajentaa SOAP-viestin toiminnallisuutta, kuten välittää sellaisia parametreja, joita ei voida välittää itse runko-osassa. Tässä esimerkissä lähetetään AuthenticationHeader -elementissä sovelluskohtaisia tietoja parametreissa AppId ja AppKey. Näillä parametreilla verkkopalvelu tunnistaa yhteyttä ottavan sovelluksen. Traderan verkkopalvelun käyttö edellyttää näitä tietoja, jotta palvelua voidaan käyttää.

SOAP-kirjekuoren otsikko alkaa Header-elementillä. Tämä elementti on valinnainen, mutta tarjoaa sovelluksille helpon ja yksinkertaisen tavan laajentaa SOAP-viestin toiminnallisuutta. SOAP-standardi ei kuitenkaan määrittele, mitä otsikon pitäisi sisältää.

Kaikki otsikossa olevat tiedot voidaan halutessa määritellä sovelluskohtaisesti. Niinpä otsikkotietojen ensisijainen tarkoitus eli laajennettavuus toteutuu hyvin. Header-elementin lapsielementtejä kutsutaan SOAP-määrityksissä otsikkomerkinnöiksi, joissa on käytettävä XML:n nimiavaruuksia. Nimiavaruuksia on käytettävä nimiavaruuksien törmäämisten (namespace collision) ehkäisemiseksi. (Järvinen 2002, 81.)

SOAP-viestissä tärkein osa on sen runko-osa (body), joka on aina pakollinen SOAP-viestissä. Runko-osassa välitetään halutunlaista XML-muotoista tietoa, RPC:n parametreja ja virheilmoituksia. Itse SOAP-standardi ei ota kantaa siihen mitä runko-osa saa sisältää. Elementtejä, jotka ovat runko-osassa sanotaan runkomerkinnöiksi. (Järvinen 2002, 84.)

SOAP:issa Fault-elementtiä käytetään virhe- ja tilatietojen ilmoituksiin. Virheitä voi tulla, jos verkkopalvelu ei kykene käsittelemään SOAP-kutsua. SOAP-standardin mukaan Fault-elementin pitää olla runko-osassa ja se ei saa esiintyä kuin kerran. Standardi määrittelee myös neljä eri lapsielementtiä jotka ovat faultcode, faultstring, faultfactor ja detail. Faultcode-elementin tarkoitus on antaa sovelluksille alkoritmeihin perustuva mekanismi vian tunnistamiseen. Faultstring-elementtiä käytetään luettavissa olevan selityksen ilmoittamiseen, esimerkiksi ”Tiedostoa ei löytynyt.” Faultfactor-elementti puolestaan kertoo mikä aiheutti virheen ja detail-elementti kertoo tarkemman kuvauksen virheestä. (Brian E. Travis, 383.)

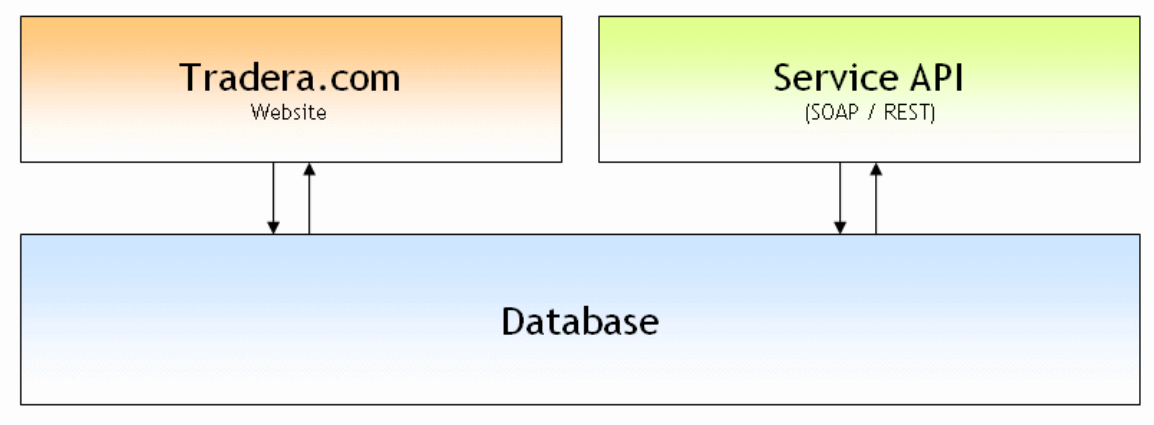
2.4 UDDI

UDDI (Universal Description, Discovery And Integration) on avoin standardoitu rajapinta, jonka avulla voidaan julkaista verkkopalveluita siten, että ne ovat verkkopalveluita tarvitsevien löydettävissä. Rajapinnasta voidaan hakea halutunlaisia verkkopalveluita erilaisin hakuehdoin. Esimerkiksi palvelun tyypin, maantieteellisen sijainnin tai palvelun tuottaneen yrityksen nimen mukaan. UDDI:n viimeisimmän standardin eli 3.0.2 määritykset löytyvät osoitteesta https://www.oasis-open.org/committees/uddi-spec/doc/spec/v3/uddi-v3.0.2-20041019.htm. (Järvinen 2002, 42, 99; OASIS 2014, UDDI, hakupäivä 9.5.2014.)

2.5 Tradera API

Traderan tarjoama API (Application Programming Interface) on ohjelmointirajapinta, jonka avulla sovelluskehittäjä pääsee käyttämään Traderan tarjoamia verkkopalveluita (web services), niiden metodeita (methods) ja luokkia (classes). Traderan API on jaettu kahteen SOAP-palveluun, PublicService (julkinen palvelu) ja RestrictedService (rajattu palvelu). Julkinen palvelu sisältää metodeja, joita sovellus voi käyttää ilman tunnistautumista. Julkinen palvelu sisältää erilaisia hakumetodeja, joilla voi hakea esimerkiksi julkisia tuote- tai myyjäkohtaisia tietoja. Rajattu palvelu puolestaan vaatii käyttäjän tunnistautumisen eli kirjautumisen järjestelmään ja sisältää metodeja, joiden kautta sovellus voi esimerkiksi lisätä tuotteita myyntiin tai hakea ostajan osoitetietoja. Näiden verkkopalveluiden muodolliset määrittelyt (formal definitions) löytyvät WSDL-dokumenteista eli palveluiden kuvauksista. Nämä WSDL-dokumentit löytyvät osoitteista http://api.tradera.com/v3/publicservice.asmx?WSDL ja http://api.tradera.com/v3/restrictedservice.asmx?WSDL. Alla kaaviossa 2 kuvataan Traderan tekninen ympäristö.

Kaavio 2. Technical overview (Tradera Technical overview 2010, hakupäivä 20.3.2014).

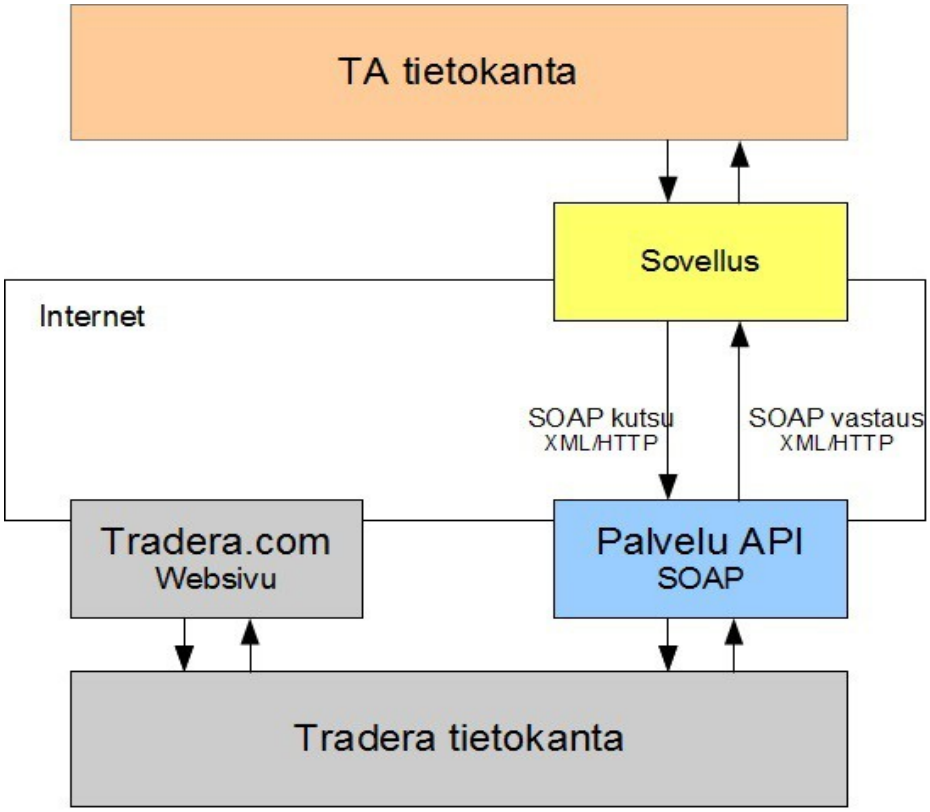


3 RAJAPINNAN TOTEUTUS

3.1 SOAP-asiakassovellus

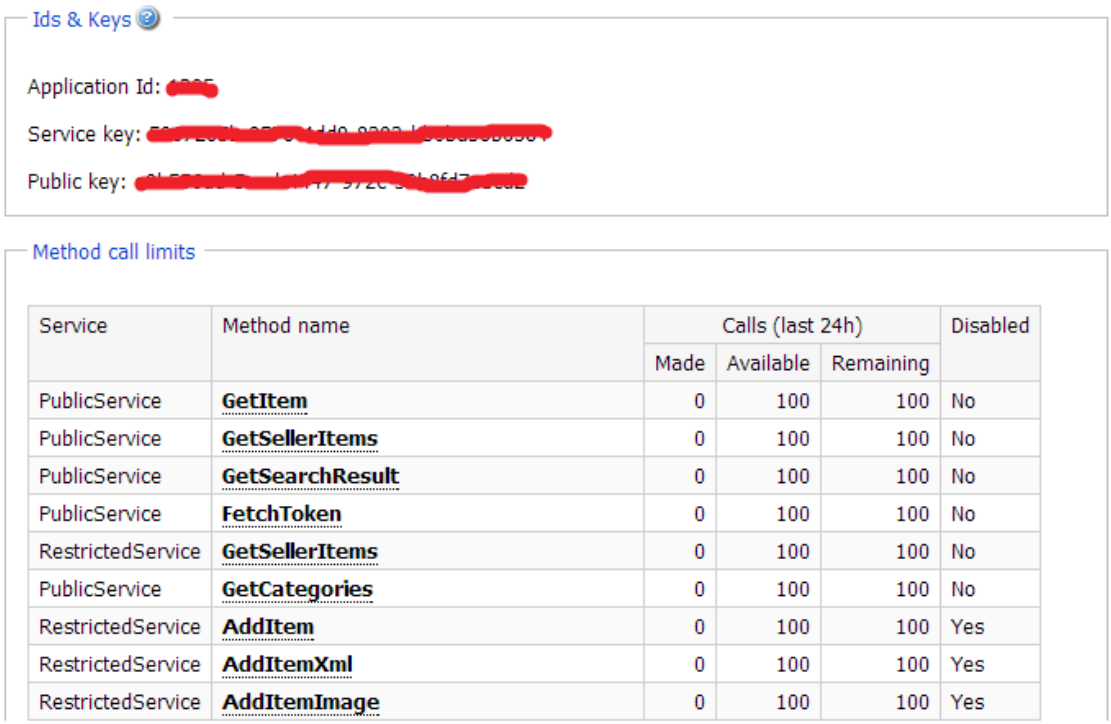
SOAP-asiakassovellus on sovellus, joka käyttää standardoitua SOAP-protokollaa kommunikoidessaan verkkopalvelun kanssa. SOAP-asiakassovellus voi Järvisen mukaan olla minkätyyppinen sovellus tahansa, esimerkiksi perinteinen Windows sovellus, komentorivisovellus, web-sovellus tai jopa toinen verkkopalvelu (Järvinen 2002, 223). Arcticas Oy:n SOAP-asiakassovelluksen kehityksessä käytin Visual Studio 2010 -ohjelmoitityökalua. Tavoitteena oli, että sovellus pystyisi päivittämään ajantasaiset tuotemäärät Traderan verkkopalvelun kautta Traderan tietokantaan. Kaaviossa 3 näkyy SOAP-asiakassovelluksen sijainti keltaisella suhteessa muuhun järjestelmään. Kaaviossa TA-tietokanta tarkoittaa toimeksiantajan toiminnanohjausjärjestelmän tietokantaa. Kehitetty sovellus on yhteydessä toimeksiantajan tietokantaan ja Traderan palveluun. SOAP-asiakassovellus kutsuu (request) palvelimella olevan etäolion metodia HTTP:tä käyttäen. Metodikutsussa käytetään protokollana HTTP:tä ja datan esitysmuotona on XML. Palvelin vastaa (response) metodikutsuun HTTP/SOAP-viestillä, joka sisältää kuittauksen metodin etäkutsun onnistumisesta, sekä metodin palauttamien paluuarvojen välittämisen takaisin.

Kaavio 3. Arcticas asiakassovelluksen sijainti suhteessa muuhun järjestelmään.



3.2 Sovelluksen rekisteröinti

Traderan verkkopalvelun käyttö edellyttää yhteyttä ottavan SOAP-asiakassovelluksen rekisteröintiä. Vain rekisteröimällä asiakassovelluksen, saa Application Id:n eli sovellustunnuksen sekä Service- ja Public-avaimet. Nämä tiedot täytyy aina liittää Traderan verkkopalvelun SOAP-kutsuihin. Sovelluksen rekisteröinti tapahtuu Traderan web-sivujen kautta, luomalla Developer Account ja rekisteröimällä sen alle New Application. Developer Accountin kautta voi tarkastella muun muassa tehtyjä metodien kutsuja, kutsurajoituksia ja onko metodi käytettävissä. Oletusarvoisesti metodien kutsumäärät ovat rajoitettu 100 kappaleeseen vuorokaudessa ja jotkin metodit ovat poissa käytöstä kuten kuvasta 1 näkyy. Kutsumääriä voi kasvattaa vain olemalla yhteydessä Traderan asiakaspalveluun ja sama koskee jos haluaa käyttöön metodin joka on poissa käytöstä.



Ids & Keys

Application Id: [redacted]

Service key: [redacted]

Public key: [redacted]

Method call limits

Service	Method name	Calls (last 24h)			Disabled
		Made	Available	Remaining	
PublicService	**GetItem**	0	100	100	No
PublicService	**GetSellerItems**	0	100	100	No
PublicService	**GetSearchResult**	0	100	100	No
PublicService	**FetchToken**	0	100	100	No
RestrictedService	**GetSellerItems**	0	100	100	No
PublicService	**GetCategories**	0	100	100	No
RestrictedService	**AddItem**	0	100	100	Yes
RestrictedService	**AddItemXml**	0	100	100	Yes
RestrictedService	**AddItemImage**	0	100	100	Yes

Kuva 1. Tradera application properties (Tradera application properties 2014, hakupäivä 5.5.2014).

3.3 Sovelluksen toiminnot ja metodit

Kehitettyyn SOAP-asiakassovellukseen tuli 3 eri päätoimintoa. Nämä toiminnot olivat myytävissä olevien tuotteiden haku Traderasta, valitun tuotteen selitteen haku Traderasta ja tuotteiden määrien päivitys Traderaan. Näistä sovelluksen päätoiminnoista luotiin myös kaaviot.

Muita sovelluksen ominaisuuksia olivat rajausmahdollisuus haetaanko vain avoimet, sulkeutuneet vai kaikki myynnissä olevat tuotteet kuten kuvassa 2. Sovellus näyttää haetussa tuotelistassa punaisella ne tuotteet, joiden tuotemäärätiedot poikkeavat toimeksiantajan tuotemäärästä ja harmaalla sulkeutuneet kohteet. Lisäksi sovelluksella voidaan päivittää vain valitun tuotteen tiedot tai kaikkien tuotteiden tiedot.

Kuva 2. Tuotehaun rajaus SOAP-asiakassovelluksessa.

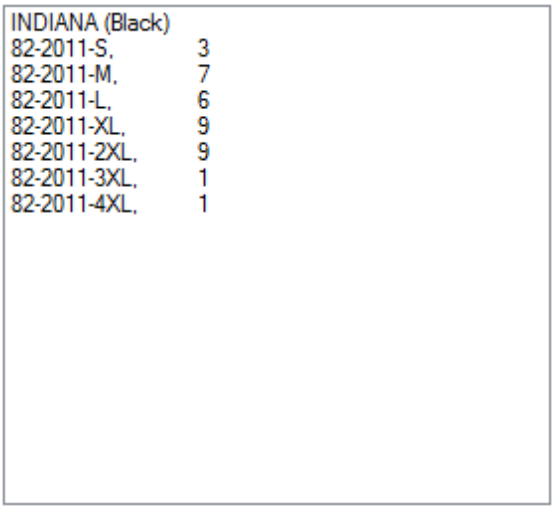
Ennen sovelluksen kehitystä perehdyin Traderan verkkopalvelun eri metodeihin ja luokkiin. Mitä metodeja ja luokkia tulisin ohjelmassa tarvitsemaan ja miten niitä käytettäisiin. Tässä apuna oli Traderan Developer Program -sivuston dokumentaatio, joka löytyy osoitteesta http://api.tradera.com/v3/default.aspx.

3.3.1 haeTuotteet -toiminto

Toimintokaaviossa haetaan Traderan verkkopalvelun kautta SOAP-kutsuna myynnissä olevat tuotteet ja palvelimelta SOAP-vastauksena saatujen tietojen perusteella haetaan toimeksiantajan tietokannasta SQL-kyselynä tuotteiden ajantasaiset lukumäärät. Traderan palvelimelta saadun vastauksen ja SQL kyselyssä saatujen tietojen perusteella järjestetään tiedot ja näytetään ne sovelluksen käyttäjälle listamuodossa kuvan 3 kaltaisesti.

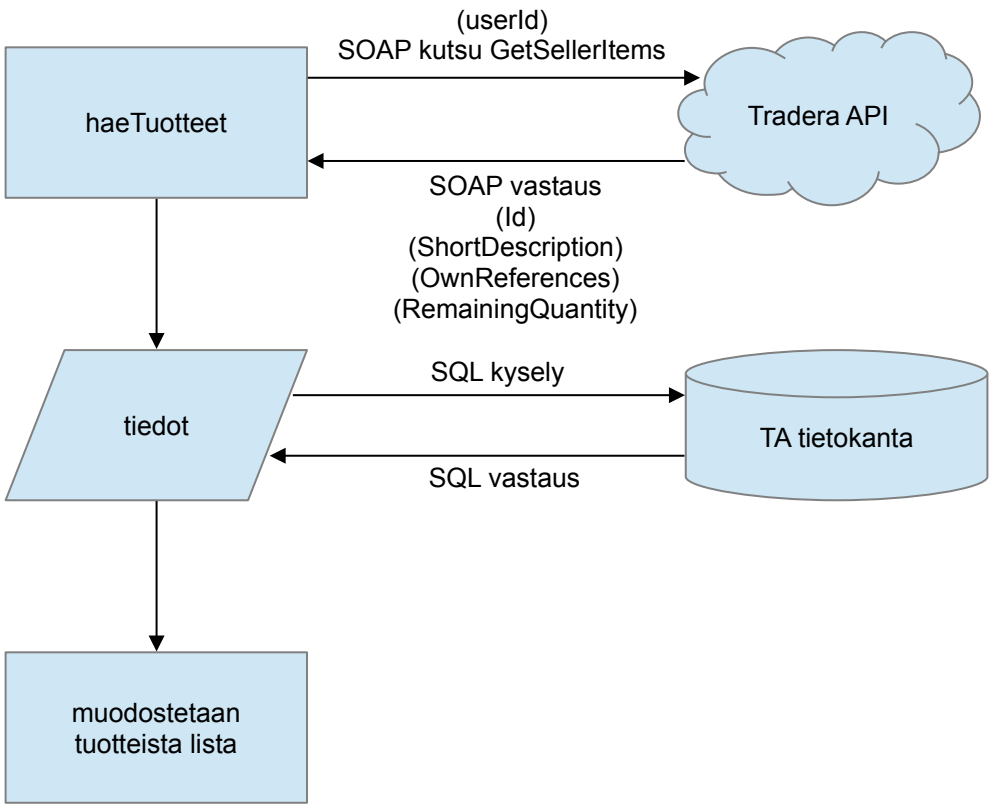
Kuva 3. Tuotteet SOAP-asiakassovelluksen listassa.

Listassa olevaa tuotetta klikkaamalla saa esiin tuotteen lukumäärät kokojen mukaan kuvan 4 kaltaisesti.



Kuva 4. Tuotteen Indiana (Black) tuotekoot ja lukumäärät SOAP-asiakassovelluksessa.

Kaavio 4. haeTuotteet -toimintokaavio.



Kuten kaaviosta 4 näkee haeTuotteet -toiminto sisältää myös tietokantahaun. haeTuotteet toiminnossa käytetään Traderan verkkopalvelun PublicServicen GetSellerItems -metodia ja parametreina pyynnössä lähetetään appId, appKey ja userId arvot. appId on yhteyttä ottavan SOAP-asiakassovelluksen tunnus, appKey sovelluspalvelun avain ja userId on myyjän tunnus Tradera.se palvelussa, joka tässä tapauksessa tarkoittaa toimeksiantajan tunnusta. AppId ja appKey arvot lähetetään aina SOAP-kutsun header-osassa ja userId puolestaan runko-osassa.

Metodikutsun vastauksena saadaan myytävissä olevista tuotteista kymmeniä eri tietoja, mutta sovelluksen kannalta tärkeimmät tiedot ovat tuotteen Id, ShortDescription, OwnReferences ja RemainingQuantity -arvot. Id on myynnissä olevan tuotteen yksilöivä tunnus ja ShortDescription on tuotteen lyhyt selite. OwnReferences on toimeksiantajan Traderaan syötetty viite, jonka perusteella haetaan tuotteen vastaavat tiedot toimeksiantajan toiminnanohjausjärjestelmän tietokannasta. RemainingQuantity tarkoittaa tuotteen jäljellä olevaa lukumäärää Tradera.se palvelussa. Taulukossa 1 on sovelluksen kannalta tärkeimmät GetSellerItems -metodin parametrit taulukkomuodossa tietotyyppeineen.

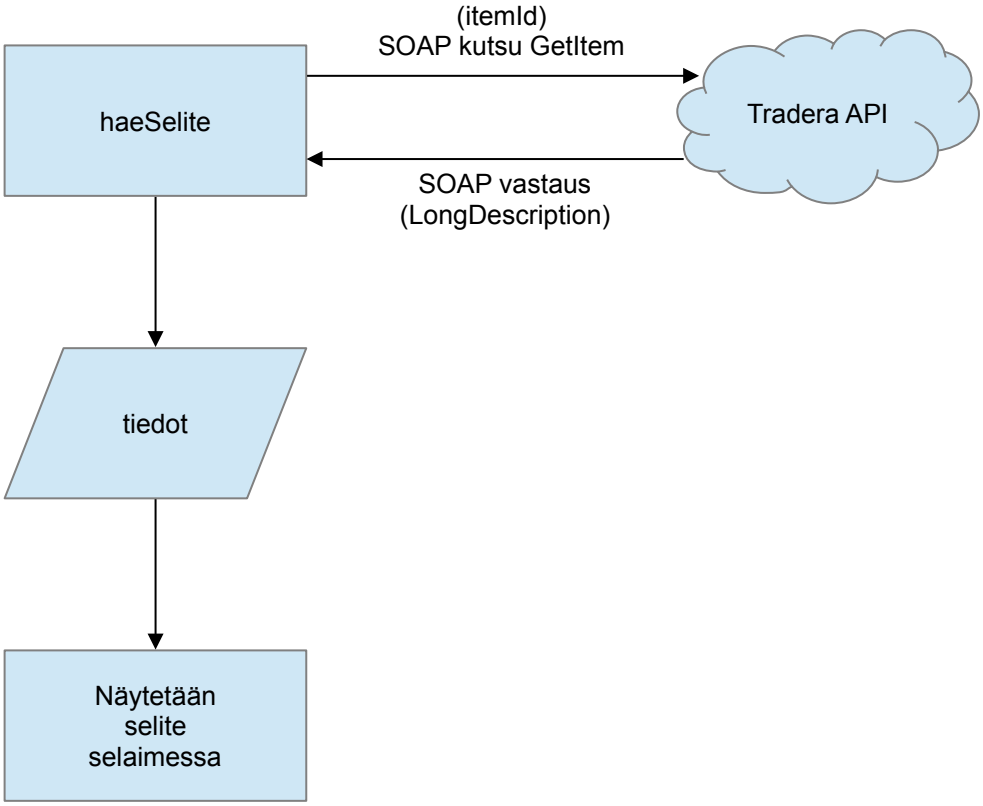
Taulukko 1. GetSellerItems -metodin parametrit tietotyyppeineen.

Service	PublicService	
Method	GetSellerItems	
Request	appId	Int
	appKey	String
	userId	Int
Response	Id	Int
	ShortDescription	String
	OwnReferences	String
	RemainingQuantity	Int

3.3.2 haeSelite -toiminto

Kaaviossa 5 kuvataan kuvataan kuinka haetaan Traderan verkkopalvelun kautta SOAP-kutsuna käyttäjän valitseman tuotteen pitkä selite. Tuotteen pitkä selite voi sisältää maksimissaan 7000 merkkiä ja se voi olla myös html-muodossa. Sovelluksessa sisältö näytetään niin sanotussa WebBrowser ikkunassa, joka osaa käsitellä html-kieltä.

Kaavio 5. haeSelite -toimintokaavio.



haeSelite toiminnossa käytetään Traderan verkkopalvelun PublicServicen GetItem -metodia ja parametreina pyynnössä lähetetään appId, appKey ja itemId arvot. itemId on tuotteen yksilöllinen tunnus Tradera.se palvelussa.

Metodikutsun vastauksena saadaan haetusta tuotteesta erilaisia tietoja, joista sovellus käyttää vain LongDescription -arvoa. Taulukossa 2 on GetItem -metodin parametrit tietotyyppeineen.

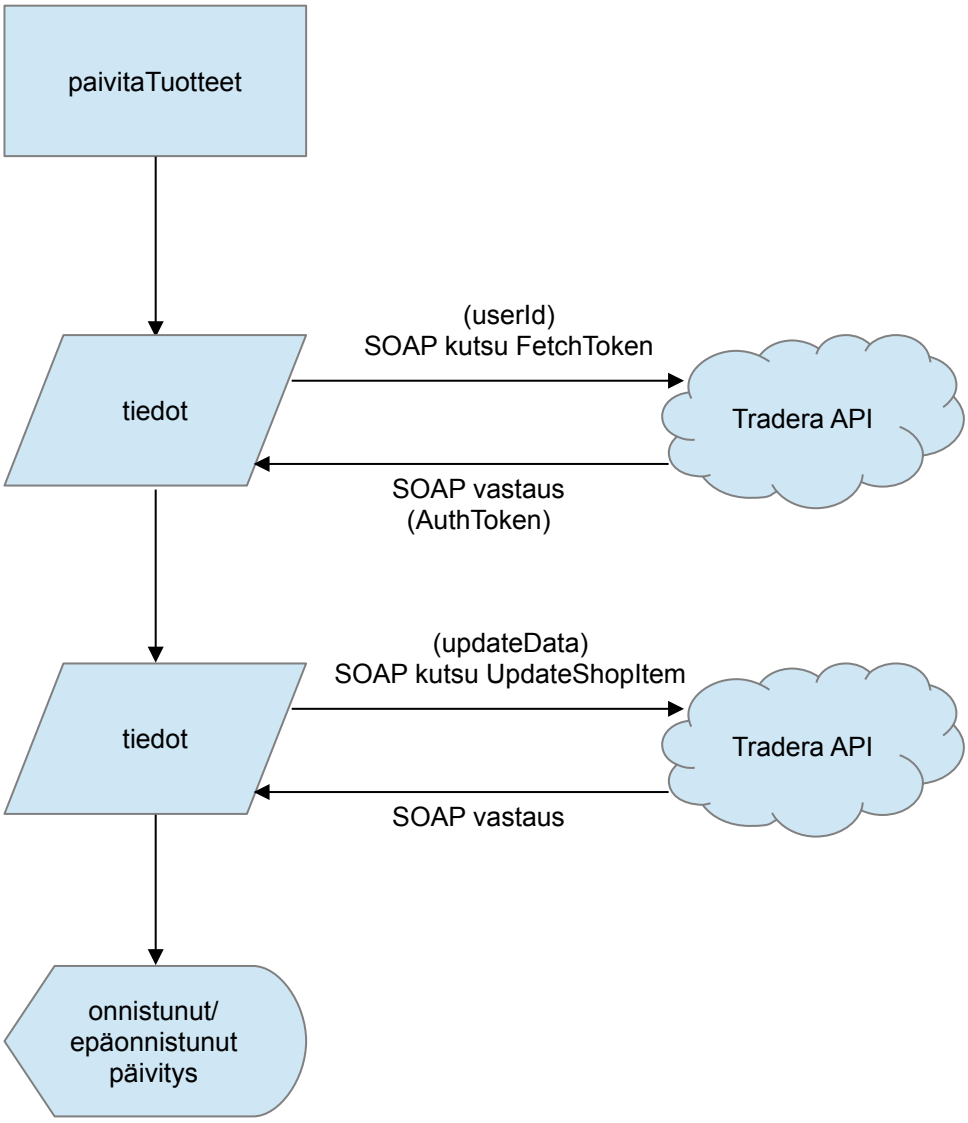
Taulukko 2. GetItem -metodin parametrit tietotyyppeineen.

Service	PublicService	
Method	GetItem	
Request	appID	Int
	appKey	String
	itemId	Int
Response	LongDescription	String

3.3.3 paivitaTuotteet -toiminto

Päivitätuotteet toiminnossa lähetetään SOAP-kutsuna palvelimelle tuotteiden ajantasaiset tuotemäärät ja päivitetään tuoteselite.

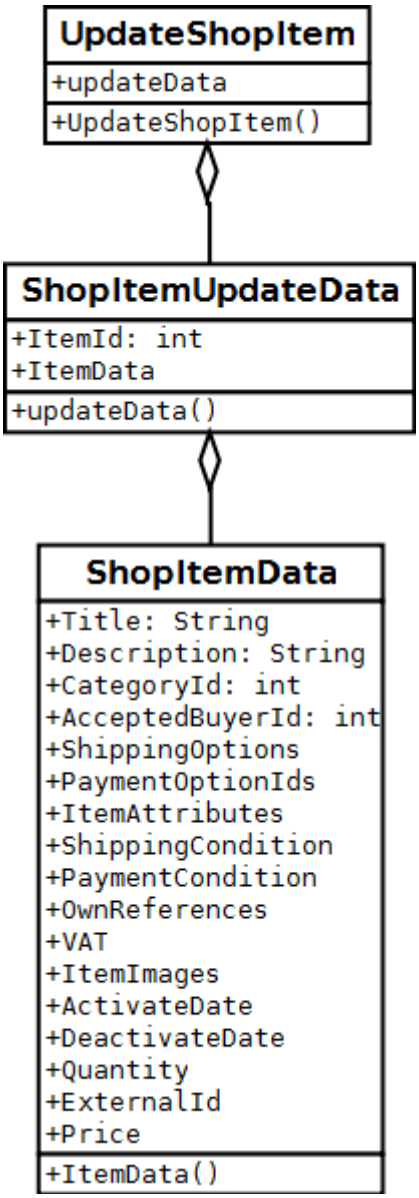
Kaavio 6. paivitaTuotteet -toimintokaavio.



paivitaTuotteet toiminnossa käytetään ensin Traderan verkkopalvelun PublicServicen FetchToken -metodia. FetchToken -metodissa parametreina lähetetään appId, appKey, userId ja secretKey arvot. secretKey on salainen avain jota käytettiin kun token luotiin. Vastauksena metodikutsulle saadaan AuthToken, joka on haetun käyttäjän valtuutus token. Eli AuthTokenin avulla sovelluksella on valtuus tehdä määritettyjä asioita käyttäjän nimissä, esimerkiksi muuttaa tuotetietoja. Itse tuotetietojen päivitys tapahtuu käyttämällä RestrictedServicen UpdateShopItem -metodia ja parametreina pyynnössä lähetetään appId, appKey, userId, token ja updateData arvot. token on se arvo joka saatiin FetchToken vastauksessa eli AuthToken. updateData sisältää luokan ShopItemUpdateData arvot ItemId ja ItemData. ItemData puolestaan sisältää luokan

ShopItemData arvot. Kaaviossa 7 selkeytetään metodin UpdateShopItem ja sen luokkien välisiä suhteita ja periytymistä.

Kaavio 7. UpdateShopItem metodin luokat ja suhteet.



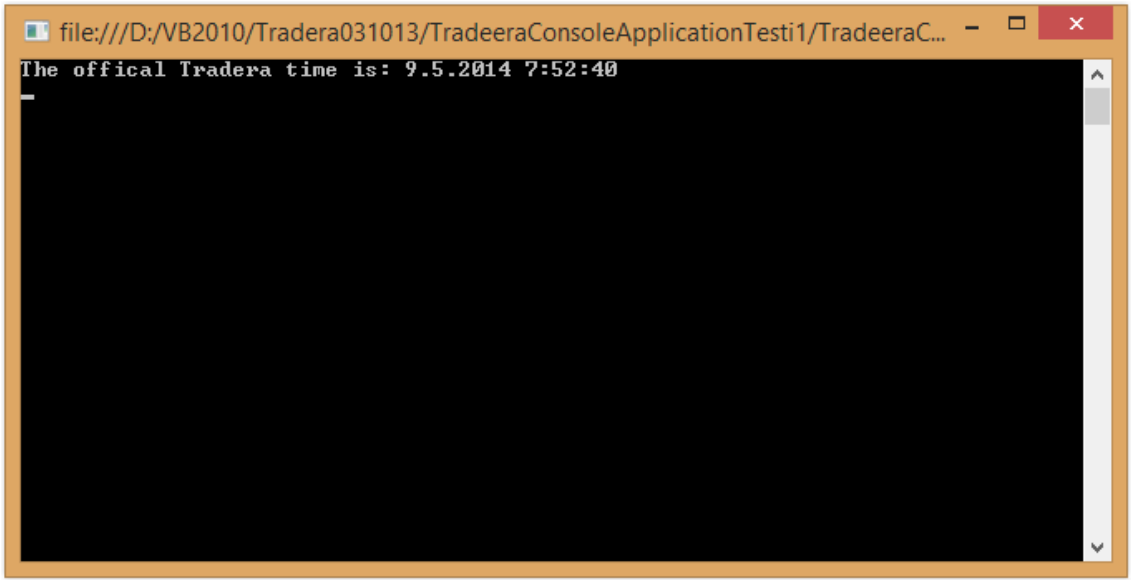
Taulukko 3. FetchToken ja UpdateShopItem -metodin parametrit tietotyyppeineen.

Service	PublicService	
Method	FetchToken	
Request	appId	Int
	appKey	String
	userId	Int
	secretKey	String
Response	AuthToken	String

Service	RestrictedService	
Method	UpdateShopItem	
Request	updateData	array

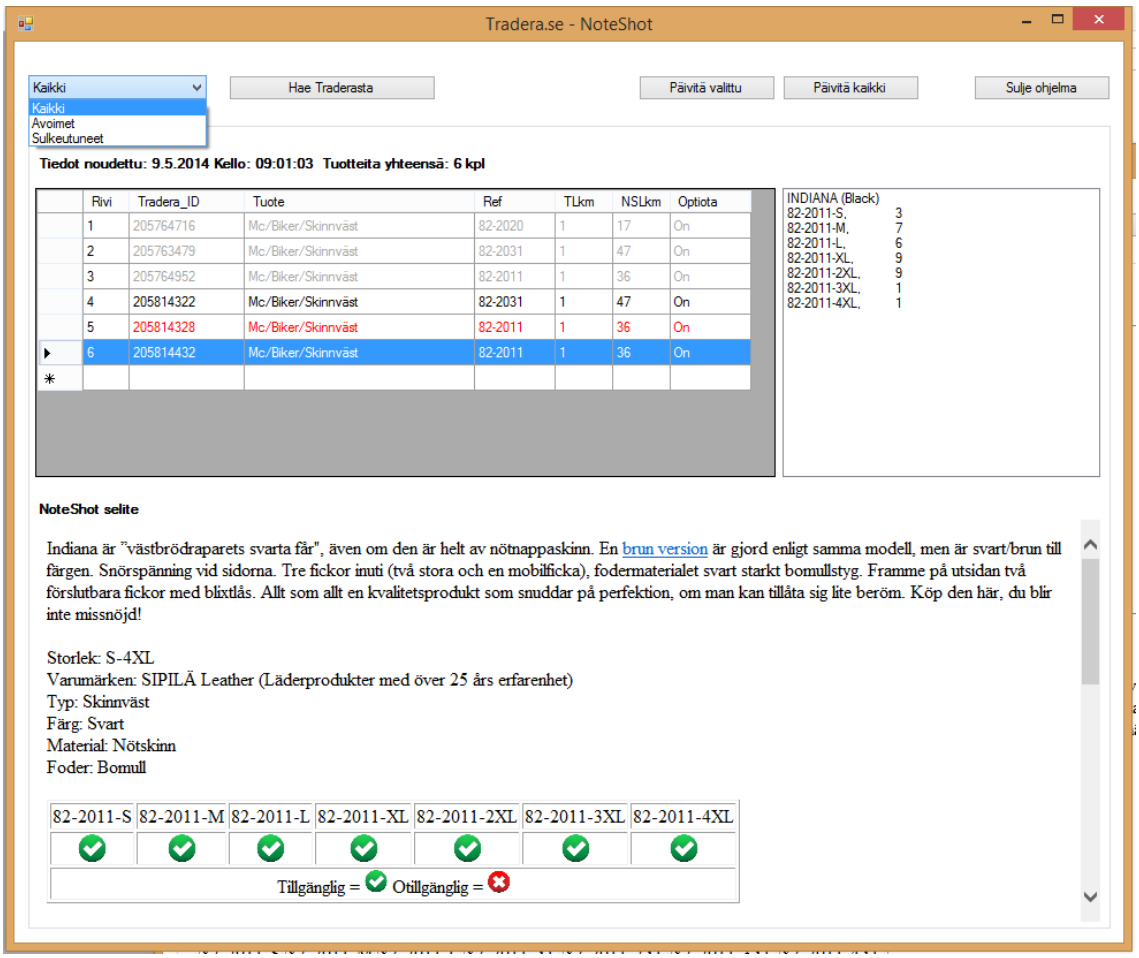
3.4 Sovelluksen kehitys

Aluksi Visual Studio 2010 ohjelmointiympäristöön lisättiin viittaus (Web Reference) Traderan verkkopalveluun. Kun reference oli käytössä voitiin aloittaa itse sovelluksen kehitys. Ensimmäiseksi luotiin testisovellus Traderan ohjeiden mukaan. Testisovellus oli yksinkertainen konsolisovellus jossa SOAP-kutsussa kutsuttiin metodia GetOfficalTime. GetOfficalTime metodi palauttaa vastauksena käyttäjälle palveluntarjoajan kellonajan kuten kuvasta 5 näkee. Testisovelluksen avulla saatiin varmistus, että yhteys Traderan verkkopalveluun toimii ja oli valmiina käytettäväksi.



Kuva 5. Tradera verkkopalvelun vastaus GetOfficalTime -metodille testisovelluksessa.

Sovelluksen ohjelmoinnissa toteutettiin yksi toiminnallisuus kerrallaan, jotka olivat myytävissä olevien tuotteiden haku Traderasta, valitun tuotteen selitteen haku Traderasta ja tuotteiden määrien päivitys Traderaan. Ohjelmoinnissa käytettiin apuna toiminnoista luotuja kaavioita. Kuvassa 6 kehitetyn SOAP-asiakassovelluksen graafinen käyttöliittymä kokonaisuudessaan.



Kuva 6. Arcticas SOAP -asiakassovelluksen graafinen käyttöliittymä.

Haastetta ohjelmointiin toi tuotekokojen pienimmästä suurimpaan järjestäminen. Tuotekokoja ei oltu tietokannassa määritelty erikseen vaan ne piti poimia tuotekoodin loppuosasta esimerkiksi 80-2010-XS, 80-2010-M ja 80-2010-4XL tietojen perusteella. SQL lauseessa nousevaan aakkosjärjestykseen järjestetyssä haussa tuotteiden koot, eivät olleet loogisessa järjestyksessä. Tähän järjestämiseen piti käyttää useita If ja Else lauseita.

Ohjelmointimenetelmänä käytettiin ns. ketterää menetelmää. Eli jaettiin sovellus lyhyempiin iteraatioihin ja toteutettiin ne loogisessa järjestyksessä.

3.5 Sovelluksen testaus

Testauksessa käytettiin kopiota toimeksiantajan toiminnanohjausjärjestelmän tietokannasta. Testausta tehtiin ohjelmointiympäristössä, tarvittaessa debuggaamalla eli suorittamalla ohjelmakoodia rivi riviltä. Myös yksikkö- ja integraatio-testausta tehtiin itse ohjelmointiympäristössä aina ohjelmistomoduuli kerrallaan. Järjestelmätestaus tehtiin toimeksiantajan tiloissa toiminnanohjausjärjestelmän tietokannan kanssa.

4 POHDINTA

Opinnäytetyö onnistui suhteellisen hyvin. SOAP-asiakassovellus saatiin toteutettua toimeksiantajan vaatimusten mukaisesti, mutta aikataulu venyi suunnitellusta kolmesta kuukaudesta lähes kahdeksaan kuukauteen. Aikataulu venyi kun en saanut toteutettua asiakassovellusta syksyn aikana opinnäytetyöhön määritetyssä aikataulussa.

Sovelluksen kehitykseen kulunut aika oli odotettua suurempi. Osa sen vuoksi, koska verkkopalvelut ja siihen liittyvät asiat olivat minulle ennestään tuntemattomia, ja osa ajasta meni Traderan asiakaspalvelun hitauteen. Esimerkiksi sähköpostiin vastaaminen saattoi venyä jopa 3 viikkoon.

Opinnäytetyöni oppimistavoite täyttyi. Opin paljon uusia asioita teknologioista, protokollista, standardeista ja termeistä jotka perustuvat SOAP-verkkopalveluihin.

LÄHTEET

Brian E. Travis 2000. XML and SOAP Programming for BizTalk Servers. Redmond, Washington: Microsoft Press.

Järvinen, Jani 2002. Hajautetut verkkopalvelut. Jyväskylä: Docendo Finland Oy.

Microsoft TechNet 2003. What is RPC?. Hakupäivä 9.5.2014.
<http://technet.microsoft.com/en-us/library/cc787851(v=ws.10).aspx>

OASIS. UDDI. Hakupäivä 9.5.2014.
<http://uddi.xml.org/>

Tradera application properties 2014. Tradera Developer Program. Hakupäivä 5.5.2014.
<http://api.tradera.com/applicationedit.aspx>

Tradera GetOfficalTime method 2010. Tradera.com API Dokumentation. Hakupäivä 20.3.2014.
<http://api.tradera.com/v3/publicservice.asmx?op=GetOfficalTime>

Tradera Public Service WSDL 2010. Tradera API Dokumentation. Hakupäivä 20.3.2014.
<http://api.tradera.com/v3/publicservice.asmx?WSDL>

Tradera Technical overview 2010. Tradera Developer Program. Hakupäivä 20.3.2014.
<http://api.tradera.com/>

W3C Consortium 2013. Tietoja W3C:stä. Hakupäivä 22.3.2014.
<http://www.w3c.tut.fi/consortium.html>

W3C SOAP 2007. SOAP Version 1.2 Part 0: Primer (Second Edition). Hakupäivä 20.3.2014.
<http://www.w3.org/TR/soap12-part0/>

W3C WSDL 2007. Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language. Hakupäivä 21.3.2014.
<http://www.w3.org/TR/2007/REC-wsdl20-20070626/>

W3C XML Essentials 2010. Hakupäivä 8.5.2014.
<http://www.w3.org/standards/xml/core>

Walkama, Pekka & Laakkonen, Aapo 2004. Inside XML-Skeema. Helsinki: IT Press.