MERX-TOIMINNANOHJAUSJÄRJESTELMÄN INTEGROIMINEN ULKOISIIN JÄRJESTELMIIN SOAP WEB SERVICEIDEN AVULLA

Timo Karvinen

Opinnäytetyö
Joulukuu 2011
Tietojenkäsittelyn koulutusohjelma
Ohjelmistotuotannon suuntautumisvaihtoehto
Tampereen ammattikorkeakoulu

TAMPEREEN AMMATTIKORKEAKOULU
Tampere University of Applied Sciences

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma
Ohjelmistotuotannon suuntautumisvaihtoehto

KARVINEN, TIMO: Merx-toiminnanohjausjärjestelmän integroiminen ulkoisiin järjestelmiin SOAP Web Serviceiden avulla

Opinnäytetyö 55 sivua, liitteet 16 sivua
Joulukuu 2011

Yrityksissä on nykyisin todella harvoja täysin itsenäisiä tietojärjestelmiä, lähes jokainen järjestelmä liittyy tavalla tai toisella johonkin toiseen järjestelmään. Tämän vuoksi helppokäyttöisistä ja toimivista järjestelmäintegraatiovälineistä ja -tekniikoista on tullut yksi tärkeimmistä yritysjärjestelmien ominaisuuksista. Opinnäytetyönä toteutettiin tällainen järjestelmäintegraatioväline, jolla tuotiin mahdollisuus luoda ja käyttää Web Service -palveluja Solteq Oyj:n Merx-toiminnanohjausjärjestelmässä.

Opinnäytetyön kirjallisessa osuudessa käsitellään Web Serviceitä yleensä sekä toteutettua Web Services -rajapintaa. Toimeksiantajan esittelyssä kerrotaan taustajärjestelmästä, jonka jälkeen työssä tarkastellaan lyhyesti SOAPia ja Representational State Transferia, jotka ovat nykyisin kaksi suosituinta tapaa toteuttaa Web Servicejä. Työssä keskitytään käsittelemään toteutetussa rajapinnassa käytettyjä ratkaisuja ja tekniikoita. Työssä käsitellään toteutuksessa käytetyt ohjelmistot ja luokkakirjastot, Zend Core ja Zend Server, NuSOAP ja Zend Framework, ja se, miten näitä sekä konfiguraatiotiedostoja avuksi käyttäen on saatu toteutettua hyvin helppokäyttöinen integraatiorajapinta.

Tämän opinnäytetyön tarkoituksena oli selvittää, miksi ja miten Solteq i Web Services -rajapinta on toteutettu ja miten se toimii. Opinnäytetyönä toteutetun järjestelmän tarkoituksena ja tavoitteena oli toteuttaa integraatiorajapinta, joka tekee Merx-toiminnanohjausjärjestelmän tulevista integraatioprojekteista nykyistä kustannustehokkaampia sekä helpompia ja nopeampia toteuttaa. Tässä onnistuttiin: Web Services -rajapinta on tuotantokäytössä useilla Merx-asiakkailla ja tulossa käyttöön vielä useammille asiakkaille lähitulevaisuudessa. Rajapinta on mahdollistanut projekteja, jotka eivät muuten olisi olleet mahdollisia tai olisivat toteutuneet hyvin erilaisessa muodossa.

Kuitenkin, niin kuin usein tietojärjestelmien kanssa, myös Web Services -rajapinnalle tulee tulevaisuudessa uusia haasteita; uusia ominaisuuksia tarvitaan ja niitä tullaan kehittämään tarpeiden ilmetessä. Näitä jo nyt uusien projektien kehityssuunnittelun kautta tunnistettuja tulevaisuudessa tarvittavia lisäominaisuuksia käsitellään opinnäytetyössä.

Asiasanat: Merx, toiminnanohjausjärjestelmä, integrointi, SOAP, REST, Web Service, WSDL, PHP, NuSOAP, RPG, IBM, System i, Zend Server, Zend Framework, i5 toolkit.

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Programme in Business Information Systems
Specialisation of Software Engineering

KARVINEN, TIMO: Integrating Merx Enterprise Resource Planning Software with External Systems Using SOAP Web Services

Bachelor's thesis 55 pages, appendices 16 pages
December 2011

There are very few completely autonomous information systems nowadays; almost every system is connected to other systems in one way or another. Therefore, easy-to-use and functional system integration tools and techniques have become one of the most important features of the enterprise systems. This thesis is about implementing one such system integration tool, which brings about the possibility to create and use Web Services with Solteq Oyj's Merx-enterprise resource planning software.

The thesis addresses topics with the Web Services in general and with the implemented Web Services -interface. The introduction to the client discusses briefly the backend system, after which there is a short review of the SOAP and Representational State transfer, which are currently the two most commonly used methods of implementing Web Services. Thesis focuses on the solutions and techniques used in creating the interface. There are also reviews of the software used and class libraries; Zend Core and Zend Server, NuSOAP and Zend Framework, and how the very easy-to-use integration interface was created with the use of these and additional configuration files.

The purpose of this thesis is to report why and how the Solteq i Web Services -interface was created and how it works. The purpose and goal of the implemented system was to create an integration interface that makes it more cost-effective, easier and quicker to implement new integration projects with Merx-software. This goal was achieved; the Web Services -interface is in use with many Merx customers, and will be implemented with even more customers in the near future. The interface has enabled projects that would otherwise not have been possible, or would have taken a very different form.

However, as often is with information systems, there are new challenges for the Web Services -interface in the future, new features will be needed, and these features will be developed as these needs arise. Already there are identified needs for additional features that have become evident with planning of new projects, which are also be covered in the thesis.

Key words: Merx, ERP, integration, SOAP, REST, Web Service, WSDL, PHP, NuSOAP, RPG, IBM, System i, Zend Server, Zend Framework, i5 toolkit.

SISÄLLYS

LYHENTEET JA TERMIT

.NET	Microsoftin kehittämä ohjelmistokomponenttikirjasto ja ajoympäristö, jolla helpotetaan sovellusten sekä esimerkiksi SOAP Web Service -komponenttien kehittämistä Windows-käyttöjärjestelmän eri versioille.
Apache	Apache HTTP Server on avoimen lähdekoodin HTTP-palvelinohjelmisto, jota kehittää Apache Software Foundation. Apache on maailman suosituin web-palvelinohjelmisto, viimeisimmän tutkimuksen mukaan yli 64 prosentin markkinaosuudella.
B2B	Business to Business eli yritysten välinen kaupankäynti.
B2C	Business to Consumer eli kaupankäynti kuluttajille.
BizTalk	Microsoftin kehittämä Windows-käyttöjärjestelmissä toimiva järjestelmäintegraatio-ohjelmisto. Käyttää eri järjestelmien integroinnissa muun muassa Web Service -tekniikoita.
CD	Solteqin autoliikkeille kehittämä toiminnanohjausjärjestelmä, joka on Merxin lisäksi toinen ohjelmisto, joka tällä hetkellä käyttää Solteq i Web Services -rajapintaa.
CRM	Customer Relationship Management eli asiakkuudenhallintajärjestelmissä ylläpidetään tietoja yrityksen asiakkaista ja mahdollisista asiakkaista liiketoimintasuhteiden jatkuvuuden ja onnistumisen takaamiseksi.
datajono	Yksi Web Service rajapinnassa käytetty IBM i:n tapa tallentaa tietoa. Käytetään rajapinnassa parametrien tai paluuarvojen siirtämiseen, jos kyseiset arvot ovat hierarkkisia taulukoita.
DB2	IBM:n kehittämä tietokantajärjestelmä, System i version tunnettiin ennen nimellä DB2/400, mutta nykyisin sitä kutsutaan DB2 for i:ksi.
Document/literal	Yksi SOAP Web Serviceissä käytetyistä mahdollisista WSDL:n muodoista.
EBCDIC	Extended Binary Coded Decimal Interchange Code on IBM System i:ssä käytössä oleva merkistötyyppi, vastaa PC-koneista tutumpaa ASCII-merkistöä.

ERP	Enterprise Resource Planning eli toiminnnanohjaus. Toiminnanohjausjärjestelmät ovat nykyaikana lähes pakollinen osa yrityksen toimintaa. Toiminnanohjausjärjestelmä voi sisältää esimerkiksi varaston-, tuotannon-, huollon-, materiaalin-, ja resurssienohjauksen, kirjanpidon, reskontran ja palkanlaskennan tarvitsemat ohjelmat jne.
Framework	Ohjelmistokehitystä helpottava luokkakokoelma / komponenttikirjasto. Nykyisin on hyvin yleistä, että ohjelmointikielille tehdään frameworkejä, jotka sisältävät yleisimmin käytetyt ominaisuudet ja funktiot, jotta niitä ei tarvitse joka kerta uutta ohjelmistoa tehtäessä kehittää uudestaan.
HTTP(/S)	Hypertext Transfer Protocol on internetissä käytetty tiedonsiirtoprotokolla. Tavallisten web-sivujen lisäksi esimerkiksi Web Serviceissä käytetään yleensä siirtoprotokollana HTTP:tä tai sen suojattua versiota HTTPS:ää.
i5 toolkit	PHP:n lisäosa, joka mahdollistaa muun muassa RPG-ohjelmakutsujen tekemisen PHP:stä. PHP:n lisäosan on kehittänyt ranskalainen AURA Equipments ja sitä jaellaan sekä Zend Core for i5/os että Zend Server for IBM i -ohjelmistokokonaisuuksien mukana.
IBM i / System i	AS/400, iSeries, i5, System i -laitteisto, jossa on käyttöjärjestelmänä OS/400, i5/OS, i/OS, IBM i. Vuosien varrella IBM on vaihtanut midrange-palvelimensa ja sen käyttöjärjestelmän nimeä useita kertoja, yleensä silloin kun laitteen teknologia on muuttunut merkittävästi. Ensimmäistä kertaa AS/400 midrange-(keskisuuri keskuskone) laitteisto julkaistiin jo vuonna 1988.
Java	Suosittu alustariippumaton ohjelmointikieli ja ajoympäristö. Java julkaistiin ensimmäisen kerran vuonna 1995 ja se perustuu C ja C++ kieliin, mutta niistä eroten Javassa muistinhallinta on automatisoitu. Javan käännetty ohjelmakoodi ajetaan Java-virtuaalikoneessa (JVM).
JSON	JavaScript Object Notation on kevyt tiedon esitysmuoto, jolla voidaan esittää taulukoita ja objekteja. Vaikka nimi viittaa Ja-

	vaScriptiin, suurin osa muistakin ohjelmointikielistä tukee nykyisin tätä tiedon esitysmuotoa. Sama tietomäärä on JSON-muodossa tiiviimmässä tilassa kuin esimerkiksi XML:llä esitettynä.
Merx	Solteqissa kehitetty suurimmaksi osaksi RPG-ohjelmointikielellä ohjelmoitu kaupan toimialan toiminnan- ja materiaalinohjausjärjestelmä.
MIME	Multipurpose Internet Mail Extensions on standardi, joka määrittelee sähköpostissa käytettäviä sisältötyyppejä ja merkistöjä. Nimestään huolimatta se ei rajoitu sähköpostiviestintään vaan samaa tekniikkaa käytetään myös esimerkiksi web-sivuilla ja muussa HTTP-viestinnässä.
MVC	Model-View-Controller (malli-näkymä-käsittelijä) on yleisin tällä hetkellä käytössä oleva ohjelmointimalli. Tämän arkkitehtuurin tarkoituksena on irrottaa käyttöliittymä ohjelmalogiikasta.
NuSOAP	SOAP Web Serviceiden käytön helpottamiseksi tehty avoimen lähdekoodin PHP-luokkakirjasto.
PHP	PHP: Hypertext Preprocessor on suosittu, ellei suosituin ohjelmointikieli web-palvelimissa. PHP:llä ohjelmoituja sovelluksia ei perinteisten ohjelmointikielten tapaan käännettä etukäteen vaan vasta suoritusvaiheessa. PHP toimii nykyisin kaikissa yleisimmissä web-palvelimissa, joskin sen yleisin pari on Apache HTTP-palvelin.
Provider / server	Palvelu / palvelin on Web Servicen osa, joka käsittelee pyynnön ja lähettää vastauksen asiakkaalle.
Requester / client	Pyyntö / asiakas on Web Servicen osa, joka pyytää palvelua toteuttamaan operaation.
REST	Representational State Transfer on uusin tulokas Web Serviceiden toteutustavoissa; toisin kuin muut Web Service -tekniikat, REST on enemmänkin arkkitehtuurinen tyyli kuin standardi.
RPC/encoded	Remote Procedure Call eli proseduurin etäkutsu on yksi WSDL-tyypeistä.
RPG	Report Program Generator on korkean tason ohjelmointikieli,

	jonka juuret johtavat reikäkorttikoneisiin. Ensimmäisiä versioita kielestä on käytetty IBM:n midrange -palvelimissa jo 1950-luvun lopulta lähtien. RPG kehitettiin IBM:n keskitason palvelinkoneisiin liiketoimintasovellusten tarpeisiin helpottamaan ja yksinkertaistamaan ohjelmistokehitystä. Kielen nykyisin käytössä olevaa versiota kutsutaan nimellä RPG IV ja uusimman IBM i -käyttöjärjestelmän julkaisun yhteydessä vuonna 2010 myös RPG-kieleen tuli lukuisia uusia ominaisuuksia.
SAP	Maailman suurin yritysohjelmistojen valmistaja, joka on erikoistunut toiminnanohjausjärjestelmiin.
SOA	Service Oriented Architecture eli palvelukeskeinen arkkitehtuuri.
SOAP	SOAP oli alun perin lyhenne sanoista Simple Object Access Protocol, mutta nykyisin sitä kutsutaan vain SOAPiksi ilman muuta merkitystä. SOAP on W3C:n kehittämä Web Service -määritys, joka kertoo miten Web Serviceiden kuuluu toimia. SOAP perustuu vahvasti XML:n käyttöön.
URI	Uniform Resource Identifier on yksikäsitteinen merkkijono, jolla voidaan tunnistaa tietty resurssi internetissä. URI on URL:n yläluokka.
URL	Uniform Resource Locator on osoite, jonka avulla löydetään haluttu resurssi internetistä.
W3C	World Wide Web Consortium on Tim Berners-Leen perustama yritysten ja yhteisöjen yhteenliittymä, joka määrittelee internetissä käytettäviä standardeja.
Web Service	Web Service voidaan suomentaa esimerkiksi "verkkopalveluksi", mutta tämä on huono suomennos, koska sillä voidaan tarkoittaa melkein mitä tahansa - siksi tässä työssä käytetään vain englanninkielistä termiä "Web Service". Toinen mahdollinen suomennos olisi "www-sovelluspalvelu", joka kuvaa ko. asiaa huomattavasti paremmin kuin verkkopalvelu, mutta myös tämä termi voi olla harhaanjohtava. Web Serviceillä tarkoitetaan kahden tietojärjestelmän keskinäistä kommunikointia tarkoitukseen tehdyn rajapinnan avulla, ilman käyttäjän erityisiä toimia.

wrapped convention	Microsoftin kehittämä ja sen tuotteissa käytetty Document/literal WSDL-tyylin erikoistapaus.
WSDL	Web Service Description Language eli Web Service kuvauskieli, jolla ohjataan ja määrätään Web Servicen käyttäytyminen, sanomien muoto ynnä muut tarvittavat ominaisuudet.
XML	Extensible Markup Language on standardoitu tapa esittää dataa tietokoneen luettavassa mudossa. XML:n tarkoitus on esittää ja siirtää dataa internetissä yleisessä ja yksinkertaisessa muodossa. XML on yleisin sanomaliikenteen tapa siirtää tietoa muun muassa Web Serviceiden välillä.

1 JOHDANTO

SOA (Service Oriented Architecture) eli palvelukeskeinen arkkitehtuuri on asia, josta puhutaan yrityksissä. Yritykset ovat havainneet, että liiketoiminta koostuu prosesseista, joita tietojärjestelmien tulee tukea. Palvelukeskeinen arkkitehtuuri mahdollistaa liiketoimintaprosessien kuvauksen toimintoketjuna, joka muodostuu eri järjestelmien tarjoamista palveluista. Järjestelmäintegraatioiden toteuttaminen SOA:n periaatteiden mukaisesti tarkoittaa sitä, että yrityksen eri järjestelmien on tuettava esimerkiksi Web Servicejä, joka on standardi tapa paljastaa järjestelmän jokin toiminto palveluna muille järjestelmille. Nykyisin järjestelmät, jotka eivät tue millään tapaa palvelukeskeistä arkkitehtuuria, voidaan mieltää vanhanaikaisiksi ja nykypäivän liiketoimintaa sekä sen vaatimuksia tukemattomiksi. Tämän vuoksi järjestelmät tarvitsevat joustavat nykyaikaiset rajapinnat integrointia varten, muuten voi olla vaarana, että järjestelmään ryhdytään tekemään päivityksiä suoraan tietokantoihin, ja järjestelmän omat ominaisuudet jäävät käyttämättömiksi.

Työnantajani ja tämän opinnäytetyön tilaaja Solteq Oyj on kehittänyt Merx-nimistä toiminnanohjausjärjestelmää jo yli 20 vuotta suomalaisille asiakkaille. Kun minä tulin yritykseen töihin, oli olemassa tarve kehittää järjestelmää nykyaikaisemmaksi toteuttamalla siihen tuki uusille järjestelmäintegraatioille Web Serviceiden avulla. Ne liiketoimintaa koskettavat kysymykset, joiden pohjalta integraatiorajapintaa ryhdyttiin toteuttamaan, olivat seuraavat:

- Kuinka pystytään toteuttamaan järjestelmäintegraatioprojektit nykyistä nopeammin?
- Kuinka järjestelmäintegraatioprojektit saadaan nykyistä helpommiksi?
- Miten peitetään System i -järjestelmän näkyminen järjestelmäintegraatioissa; poistaen System i:n negatiiviset puolet, kuten esimerkiksi EBCDIC-merkistö, ja korostaen sen positiivisia puolia, kuten kykyä käsitellä suuria tietomassoja nopeasti ja Unix-järjestelmiä parempaa käytettävyysaikaa (uptime)?
- Miten IBM i -alustalla toimiva Merx integroituu tehokkaammin Solteqin kokonaistarjontaan?
- Miten Merxin käyttöä laajennetaan erilaisiin automaatiojärjestelmiin ja käyttäjäpäätteisiin?

- Miten mahdollistetaan järjestelmäintegraatioprojektit kokonaisvaltaisesti aiempaa kustannustehokkaammin?
- Miten luodaan standardit rajapinnat Merxiin siten, että uusia liittymiä ulkoisiin järjestelmiin ei tarvitse tehdä aina uudestaan tapauskohtaisesti, toisen osapuolen ehdoilla?

Web Services -rajapinnan tarkoituksena on siis tuoda SOA-tuki muun muassa Solteq Merx -toiminnanohjausjärjestelmään tarjoamalla mahdollisuus paljastaa olemassa olevia kyseisen ERP:n sisältämiä toimintoja muille järjestelmille sekä luomalla uusia palveluja Merxiin. Vastaavasti toteutetaan rajapintaan mahdollisuus kutsua Merx-ohjelmista ulkoisissa järjestelmissä toteutettuja Web Serviceitä, joissa rajapinnan Requester kutsuu toisen järjestelmän palvelua ja palauttaa tietoa Merxiin. Solteqilla on tuote- ja ratkaisuvalikoimassaan lukuisia erilaisille alustoille, erilaisilla ohjelmointikielillä ja erilaisiin käyttötarkoituksiin tehtyjä sovelluksia ja järjestelmiä, Web Serviceillä on tarkoitus mahdollistaa helppo, standardi ja yleiskäyttöinen rajapinta näiden järjestelmien väliseen tiedon kulkuun. Muut järjestelmät pystyvät keskustelemaan Merxin kanssa reaaliaikaisesti, joko niin, että Merx pyytää informaatiota tai että siltä pyydetään sitä. Web Serviceillä pyritään myös vähentämään järjestelmäintegraatioissa tarvittavaa työn määrää ja kustannuksia. Merx järjestelmänä on jatkuvasti kehittyvä: uusia ominaisuuksia tehdään koko ajan ja näitä ominaisuuksia toteutettaessa syntyy uusia moduuleja, joita voidaan suoraan hyödyntää paljastettaessa ominaisuuksia muille järjestelmille Web Serviceiden kautta.

Tässä opinnäytetyön kirjallisessa osuudessa käydään läpi Web Services -rajapinnan toteutus, siinä käytetyt tekniikat, välineet ja rajapinnan käyttö tuotannossa. Tämän selvityksen avulla voidaan levittää tietoa Web Serviceistä ja niiden käyttämisestä järjestelmäintegraatioissa Solteqin sisäisesti; tarkoituksena on helpottaa ymmärtämistä, miten ja missä yhteyksissä Web Services -rajapinta toimii integraatioiden apuna. Työssä tarkastellaan myös jatkokehitystarpeita, joita uudet integraatiotarpeet ja keskustelukumppanit tuovat esille. Työssä esitellään myös kaksi hallitsevaa toteutustapaa Web Serviceille sekä se, miten toinen näistä on toteutettu IBM System i -alustalla. Työn aihepiiristä on kirjoitettu paljon; kaikkia löytämiäni lähteitä en käytä työssäni, mutta olen kerännyt hyödyllisiä asiantuntijadokumenttien ja käytettyjen ohjelmistomanuaalien linkkejä liitteeseen 1.

2 TOIMEKSIANTAJAN ESITTELY

2.1 Solteq Oyj

Työnantajani Solteq Oyj on ohjelmistopalveluyhtiö, joka toimittaa ohjelmistoja ja ratkaisuja liiketoiminnan kehitykseen ja toiminnan ohjaukseen. Solteq on perustettu vuonna 1982 ja tunnettiin silloin Tampereen Tiedonhallintana. Vuonna 1999 yritys listautui Helsingin pörssiin ja muutti nimensä nykyiseen muotoon Solteq Oyj. Solteqin pääkonttori sijaitsee Tampereella, ja lisäksi toimipaikkoja on Helsingissä, Lahdessa ja Hämeenlinnassa. Solteqilla on tällä hetkellä noin 240 työntekijää, joista suurin osa työskentelee Tampereella Klingendahlin rakennuksessa Eteläpuisto 2 C:ssä. Solteqin liikevaihto vuonna 2010 oli noin 27 miljoonaa euroa. (Solteq Oyj -vuosikertomus 2010; Solteq Oyj, www-sivu 2011.)

Solteq on organisaatiokaavion mukaan jaettu neljään vahvuusalueeseen, jotka ovat ERP, EAM, DATA ja STORE. ERP kattaa toiminnanohjausjärjestelmät, taloushallinnon ja johdon raportoinnin, toimitusketjun ja henkilöstöhallinnon. EAM sisältää kunnossapitojärjestelmät, käyttöomaisuuden optimoinnin, varaston ja töiden hallinnan sekä kenttähuollon hallinnan. DATA keskittyy perustietojen laadun parantamiseen, ylläpidon ulkoistuspalveluihin, tiedonkeruuseen investointiprojekteissa ja integraatioihin. STORE pitää sisällään kassajärjestelmät, myymäläjärjestelmät ja myymäläteknologiat. Itse työskentelen ERP-osastolla, koska Merx, järjestelmä, jonka yhteyteen Web Service -rajapinta toteutetaan, on toiminnanohjausjärjestelmä. (Solteq Oyj -vuosikertomus 2010; Solteq Oyj, www-sivu 2011.)

2.2 Merx

Solteq Merx -toiminnanohjausjärjestelmä kattaa yrityksen koko materiaalihallinnon, myynti-, osto- ja varastotoiminnot. Merx on täysin Solteqissa kehitetty järjestelmä, jonka ensimmäisiä asiakkaita on ollut muun muassa Toyota Tammer-Auto 1980-luvun loppupuolella. Merx on siis suomalainen, suomalaisten asiakasyritysten kanssa yhteistyössä kehitetty toiminnanohjausjärjestelmä, joka sisältää kaikki suomalaisessa ja eurooppa-

laisessa kaupankäynnissä tarvittavat toiminnot, tämän lisäksi Merx on käytössä joissakin yrityksissä Ruotsissa, Virossa ja Puolassa. Merx on toteutettu IBM:n Power Systems -laitteiston ja IBM i -käyttöjärjestelmän päälle enimmäkseen käyttäen RPG-ohjelmointikieltä, mikä tekee siitä erittäin tehokkaan, nopean ja luotettavan. (Solteq Merx -tuotekortti 2010; Solteq Oyj, www-sivu 2011.)

Merx on aktiivisesti asiakkaiden kanssa yhteistyössä kehittyvä järjestelmä, johon lisätään uusia ominaisuuksia lähes päivittäin. Asiakkailta tulee paljon pyyntöjä uusien ominaisuuksien toteuttamiseksi, ja myös uusien teknologioiden käyttöönottamiseksi. Yksi suuri Merxin etu muihin kansainvälisten yritysten toteuttamiin toiminnanohjausjärjestelmiin nähden on juuri Merxin mahdollisuus mukautua käyttäjien tarpeisiin nopeassa päivityssyklissä. Asiakkaiden tilaamat ja toteutetut ominaisuudet ja muutokset hyödyttävät myös toisia asiakkaita, aina seuraavan Merx-julkaisun yhteydessä tai tarvittaessa jo aikaisemmin. (Solteq Merx -tuotekortti 2010.)

Koska Merxiä normaalisti käytetään tekstipohjaisen käyttöliittymän kautta, telnet- yhteyden yli, on tarvetta ja halukkuutta uusien tekniikoiden käyttöönottoon ilmennyt runsaasti viime aikoina. Esimerkiksi uudenaikaisen käyttöliittymätarpeen vuoksi on Solteqissa otettu käyttöön IBM i -alustalla Zendin ja IBM:n yhteistyössä tarjoamat PHP-ratkaisut. PHP:tä on ryhdytty käyttämään lyhyen ajan sisällä hyvin monipuolisesti Merxissä. PHP:n kautta muodostuvat monet erityyppiset nykyaikaiset tiedostoformaatit, kuten PDF ja Excel, vaikka pyyntö näiden tiedostojen muodostamiseen lähteekin perinteiseltä tekstipohjaiselta käyttöliittymäruudulta. PHP:tä käytetään myös paljon sanomaliikenteen modernisointiin, esimerkiksi vanhahtavien EDI-sanomien rinnalle on tuotu vaihtoehdoksi nykyaikaisempi XML-muoto, ja tämän lisäksi kokonaan uutena asiana sanomaliikenteeseen on tuotu Web Servicet.

3 WEB SERVICET JÄRJESTELMÄINTEGRAATION VÄLINEENÄ

3.1 Mahdolliset toteutustavat

Web Servicet ovat rajapintoja, jotka on kehitetty helpottamaan järjestelmien välistä, tietoverkkojen yli tapahtuvaa kommunikointia. Web Serviceitä voidaan toteuttaa lukuisilla eri tavoilla, selkeästi suosituimpia tapoja ovat SOAP ja REST. SOAP on W3C:n eli World Wide Web Consortiumin kehittämä standardi, joka määrittelee, miten Web Serviceiden kuuluu käyttäytyä ja toimia. REST on enemmänkin ajatusmalli kuin standardi, REST-filosofiaan nojautuvia Web Serviceitä voidaan toteuttaa monella tapaa REST-ajattelumallin antaessa vain suuntaviivat mahdolliselle toteutukselle. Yleistyksenä voidaan todeta, että REST-tyyppiset Web Servicet ovat helppoja toteuttaa ja vaikeita käyttää, kun taas SOAP-palvelut ovat vaikeita toteuttaa ja helppoja käyttää. Tässä tulee kuitenkin ottaa huomioon, että Solteq i Web Services -rajapinnan avulla SOAP-palveluiden toteuttaminen on tehty todella helpoksi. SOAPilla ja RESTillä on paljon yhteistä, molemmat käyttävät lähes poikkeuksetta käytännössä tiedonsiirtoprotokollana HTTP:tä tai sen suojattua versiota HTTPS:ää, vaikka protokollaa ei ole kummassakaan määrätty. Molemmat ovat alusta- ja ohjelmointikieliriippumattomia, Web Servicejä voi sekä SOAPilla että RESTllä tehdä esimerkiksi Linux, Windows, IBM i -alustoilla ja käyttäen ohjelmointikielenä esimerkiksi Javaa, C#:a tai PHP:tä. (Richards 2006, 9–14.)

3.1.1 SOAP

Standardeiksi luettavista Web Service -tekniikoista selkeästi käytetyin on SOAP. SOAP Web Serviceiden peruskomponenttina on WSDL-tiedosto, se ei kuitenkaan ole pakollinen. WSDL (Web Service Description Language), joka perustuu XML-kieleen, kertoo palvelun ominaispiirteet standardoidulla tavalla. WSDL kertoo palvelusta sen tarjoamat operaatiot, viesteissä tarvittavat parametrit ja paluuarvot hyvin yksityiskohtaisella tasolla, viestien muodon, yhteyskäytännön ja sijainnin. WSDL:n tarkoitus on tehdä palvelun käyttäminen mahdollisimman helpoksi, asiakassovelluksen tekijän ei tarvitse tietää mitään rajapinnan toteutuksesta, viestien muodosta tai edes ymmärtää WSDL:ää voidakseen käyttää palvelua. Richards väittää kirjassaan Pro PHP XML And Web Services

tällä olevan haittapuolen, hänen mielestään WSDL:n kirjoittaminen on "mustan magian harrastamista". Kuitenkin koska WSDL:n kielioppi on tarkkaan määritelty, on se mahdollista automatisoida, kuten Solteq i Web Services -rajapinnassa on tehty. Rajapintaa käyttämällä ei palvelua pystytettäessäkään tarvitse ymmärtää mitään WSDL:stä, koska kyseiset tiedostot luodaan automaattisesti. Liiteessä 3 on esimerkki rajapinnalla automaattisesti generoidusta WSDL-dokumentista, joka on asiakkaalla tuotantokäytössä. Kyseisestä dokumentista voi nähdä WSDL:n vaadittavat elementit ja esimerkkinä, miten vastaussanoma muodostuu; siinä ei kuitenkaan ole käytetty kaikkia WSDL:n ominaisuuksia, kuten esimerkiksi tarkkoja rajoituksia parametrien pituudelle jne. (Richards 2006, 673–696.)

WSDL on XML-kielinen dokumentti, joka kuvaa Web Servicen ominaisuudet. WSDL:stä on olemassa erilaisia tyyppejä; WSDL:ssä pitää kertoa palvelun sidonta (binding), joka voi olla joko RPC tai Document, sekä palvelun käyttötapa (USE), joka voi olla joko encoded tai literal. Edellä mainittuja voidaan yhdistellä haluttaessa ristiin, jolloin saadaan seuraavat vaihtoehdot:

- RPC/encoded,
- RPC/literal,
- Document/encoded ja
- Document/literal.

Käytännössä kuitenkin näistä käytetään vain RPC/Encoded tai Document/Literal -tyylejä. Näiden lisäksi on vielä olemassa viides tyyli eli Document/literal wrapped convention, joka valittiin Solteq i Web Serviceiden perustyyliksi, koska kyseinen tyyli on vahvasti suosittu Microsoftin tuotteissa, joita rajapinnan tuli ensi tilassa tukea. Tämä aiheutti kuitenkin rajapintaa kehitettäessä yhden ongelman; PHP:n mukana tuleva SOAP-lisäosa ei tue tätä WSDL-tyyliä, jonka takia rajapinnassa päädyttiin käyttämään avoimen lähdekoodin NuSOAP-kirjastoa natiivin SOAP-lisäosan sijaan. (Butek R. 2003; Richards 2006, 673–696.)

SOAP-sanomalla on tietyt vaaditut komponentit: SOAP-sanoman tulee aina olla käärittynä kirjekuoreen (envelope), tämä kuori voi sisältää tarvittaessa otsikkotieto-elementin (header) ja sen täytyy sisältää runkoelementti (body). Sanomassa täytyy myös määritellä tarvittavat nimiavaruudet (namespace), kirjekuorella on oma määrätty nimiavaruutensa ja varsinaisen viestin elementeille on yleensä annettu yksi tai useampia nimiavaruuksia.

Valinnaisessa otsikko-elementissä voidaan kertoa esimerkiksi ohjaustietoja palvelimille, jotka käsittelevät viestiä matkalla sen lopulliselle vastaanottajalle. Kun palvelin, joka ei ole sanoman lopullinen vastaanottaja, saa sanoman, jossa on sille osoitettuja ohjaustietoja otsikko-elementissä, se tekee toiminnot, jotka sille on määrätty, poistaa kyseisen merkinnän otsikkotiedoista ja lähettää viestin eteenpäin. Otsikkoelementissä kuljetetaan myös usein tunnistetietoja, kuten käyttäjätunnus ja salasana, lopulliselle vastaanottajalle. Runko-elementti sisältää varsinaisen sanomassa kuljetettavan tiedon: jos sanoma on pyyntö, siellä on palvelun kutsuun tarvittavat parametrit, ja jos kyseessä on vastaus, runko sisältää palvelun palauttamat paluuarvot. Runko-elementin sisältämien elementtien vaadittu muoto ja tyyli on määrätty WSDL-dokumentissa. Runko-elementti voi asianmukaisen vastausviestin sijaan sisältää myös virhe (fault) -elementin, mikäli annetuilla arvoilla - tai jostain muusta syystä - palvelu ei pystynyt suorittamaan haluttua toimintoa oikein, eikä varsinaisia paluuarvoja voida palauttaa. (Richards 2006, 696–705.)

3.1.2 REST

Tohtori Roy Fielding kirjoitti väitöskirjan[1] RESTistä vuonna 2000. REST ei ole standardi, se on arkkitehtuurityyli, jossa informaatiolla eli resurssilla on esitysmuoto (Representation), tila (State) ja tämä tietyssä esitysmuodossa oleva resurssin tila voidaan siirtää (Transfer) toiseen järjestelmään. REST-ajattelun nopea nousu suosituksi arkkitehtuurimalliksi selittyy paljolti muun muassa sillä, että se ei tuo uusia teknologioita opeteltavaksi, vaan ohjelmoijan pitää vain osata käyttää vanhoja tuttuja teknologioita REST-tyylisesti. RESTin peruskomponentteihin kuuluvat XML, HTTP-protokollan GET, HEAD, POST, PUT, DELETE -metodit, URI ja MIME. (Richards 2006, 633–639.)

Datan esitysmuoto RESTissä on yleensä XML, mutta toisin kun SOAPissa tai muissa vanhemmissa Web Service -standardeissa, sillä ei ole tarkkaan määriteltyä muotoa ja sääntöjä. Periaatteessa REST-tyyppinen XML-sanoma voi olla minkä muotoinen tahansa, tämä on täysin kyseisen palvelun ohjelmoijan päätettävissä. Juuri tämän takia REST-palvelut ovat helppoja toteuttaa ja vaikeita käyttää, ja koska palvelun ohjelmoija voi

[1] Kyseinen teos löytyy PDF-dokumenttina internetistä osoitteesta
http://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf

tehdä rajapinnassa täysin omavaltaisia ratkaisuja, täytyy palvelun käyttäjän aina mukautua näihin ratkaisuihin ja tietää tarkalleen, kuinka palvelua teknisesti käytetään. Vaikka kaksi eri RESTillä toteutettua palvelua palauttaisivat asiakasohjelmalle täsmälleen saman tiedon, näillä palveluilla on todennäköisesti myös kaksi eri esitysmuotoa, jos niillä on kaksi eri ohjelmoijaa. Näin ollen asiakasohjelman ohjelmoija joutuu tekemään kaksi eri asiakassovellusta näitä kahta palvelua varten, vaikka ne toteuttavat saman toiminnon. (Richards 2006, 633–639.)

Yksi RESTin perusajatuksista liittyy HTTP-protokollan metodien käyttämiseen niiden alkuperäisen tarkoituksen mukaisesti. HTTP-protokolla sisältää yleisesti käytettyjen GET- (hae) ja POST (lähetä) -metodien lisäksi myös tuntemattomamman HEAD (hae otsikkotiedot) -metodin, ja lähes käyttämättömiksi jääneet PUT- (tallenna) ja DELETE (poista) -metodit. REST web serviceissä näitä tulisi käyttää ohjaamaan ja osoittamaan palvelun toiminto, jolloin palvelun nimenä käytettäisiin substantiivia kuvaamaan resurssia, ilman verbiä osoittamassa toimintoa. Kuten yleisesti muuallakin HTTP-protokollan käyttökohteissa, myös REST-palveluissa on näiden metodien käyttö kuitenkin jäänyt vähäiseksi, ja yleisimmin RESTissä käytetään vain GET- ja POST-metodeja, ja palvelun toiminnon ohjaaminen tapahtuu niiden parametreilla. Kuitenkin GET- ja POST-metodien välillä tehdään yleensä selvä pesäero, jolloin näitä käytetään REST-serviceissä siten, että GET on aina pelkkää tiedon hakemista varten, ja jos tietoa muokataan jollain tapaa eli lisätään, poistetaan tai muutetaan, niin silloin käytetään POST-metodia. (Richards 2006, 633–639.)

REST-ajatusmaailma on helppo omaksua, koska periaatteessa normaaleja web-sivujakin voidaan ajatella REST-tyyppisinä Web Serviceinä. REST-palvelu on myös usein kevyempi kuin vastaava SOAP Web Service, koska palautetun XML:n muoto voi standardien puutteen takia olla hyvin minimalistinen. Tämän vapauden mukana kuitenkin tulee varjopuoli, palvelun käyttäjän on vaikeampaa sopeutua lukuisiin erilaisiin, omia standardejaan toteuttaviin palveluihin. (Richards 2006, 633–646.)

3.2 Keskustelukumppanit

Web Serviceitä voidaan nykyisin käyttää ja toteuttaa lähes kaikissa suosituimmissa järjestelmissä ja ohjelmointikielissä. Koska SOAP on XML-pohjainen standardi, joka ei ota kantaa esimerkiksi siihen, minkä protokollan yli tiedonsiirto sanomia vaihdettaessa tehdään, tarkoittaa tämä sitä, että SOAP on aidosti alusta- ja ohjelmointikieliriippumaton tapa tehdä Web Serviceitä.

Ensimmäisessä vaiheessa Web Service -rajapinnan toteutuksessa otetaan huomioon joitakin todennäköisiä keskustelukumppaneita eli viestinnän toisella puolella olevia tietojärjestelmiä. Todennäköisimpiä pareja Web Service -rajapinnalle ovat SOAP Web Servicet, jotka on toteutettu Microsoftin välineillä, esimerkiksi .NET -ohjelmistokomponenttikirjastolla ohjelmoiduilla sovelluksilla tai Microsoft BizTalk -serverin päälle tehdyillä Web Serviceillä. Tämän lisäksi on tiedossa, että tulemme jossain vaiheessa kommunikoimaan järjestelmien kanssa, joissa ohjelmointikielinä on muun muassa Java, C++ tai PHP. Muita valmiita sovellusjärjestelmiä, joiden kanssa integraatioita tullaan tekemään, ovat Lotus Notes -alustan järjestelmät, SAP-toiminnanohjausjärjestelmät ja IBM WebSphere -perheen tuotteet. Myös järjestelmätyyppejä käynnissä olevissa ja tulevissa integraatioissa on lukuisia, esimerkiksi taloushallintajärjestelmät, asiakkuudenhallintajärjestelmät (CRM), intranet- ja extranet-järjestelmät sekä erilaiset verkkokaupat suunnattuina sekä kuluttajille (B2C) että asiakasyrityksille (B2B).

Tulevissa järjestelmäintegraatioissa tulee siis olemaan hyvin paljon erilaisia osapuolia tekniseltä kannalta ajatellen. Järjestelmiä tulee myös olemaan eri toimittajilta, todennäköisesti suurin osa on kolmannen osapuolen järjestelmiä, mutta Web Services -rajapintaa käytetään myös Solteqin omien tuotteiden integroimiseen toisiinsa. Yksi esimerkki tällaisesta on jo nyt toteutettu integraatio Merxin ja Solteq Kassan kanssa, kassajärjestelmä siis kommunikoi Merxin kanssa Web Serviceiden avustuksella.

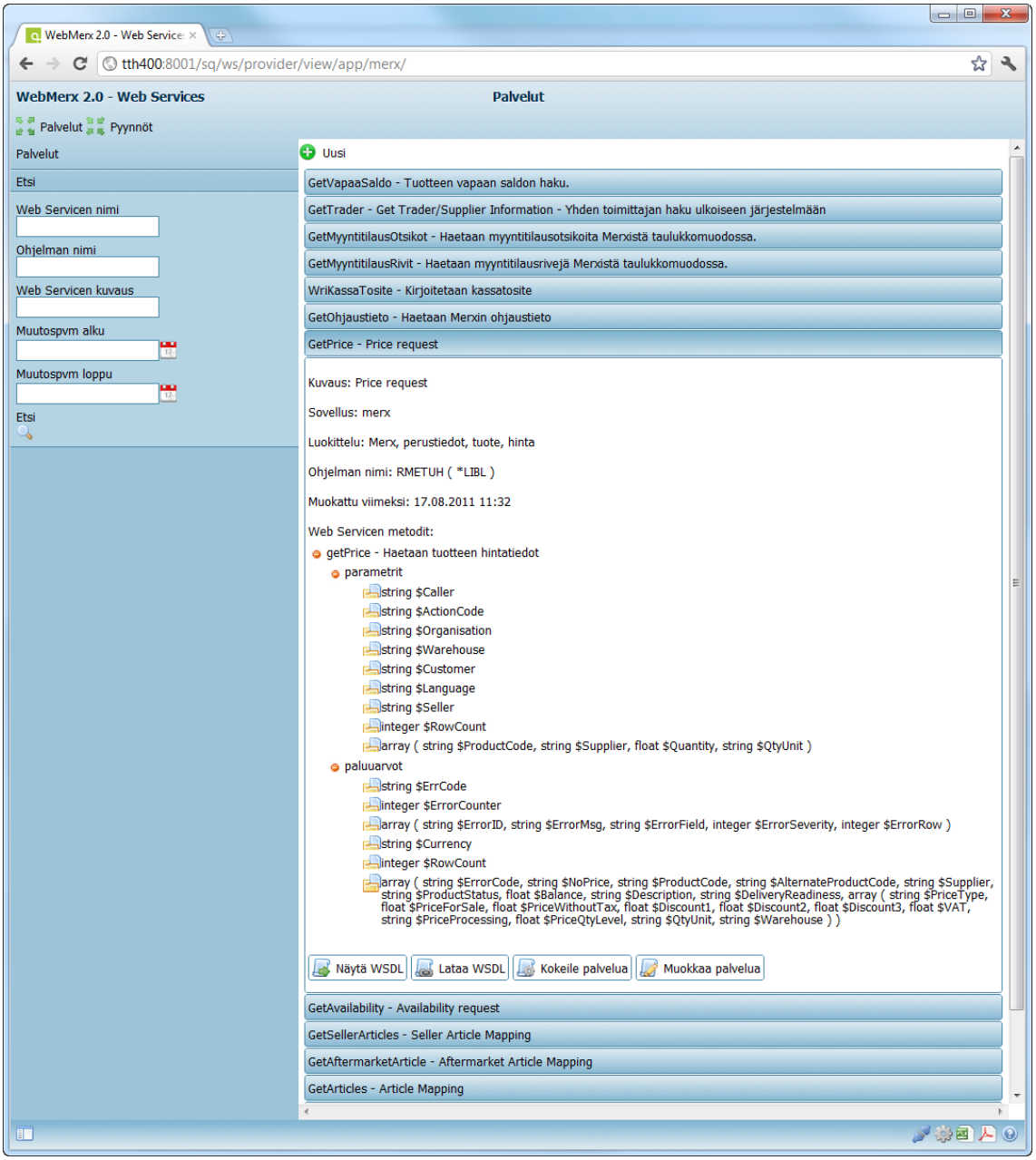
4 SOLTEQ I WEB SERVICES -RAJAPINTA

4.1 Tavoite ja toimintaperiaate

Solteq i Web Serviceiden tarkoitus on saada Merxiin yleiskäyttöinen rajapinta, jonka avulla Merx pystyy keskustelemaan muiden järjestelmien kanssa. Rajapinnan on tarkoitus toimia lähes automaattisesti niin, että uusi palvelu voidaan rakentaa ja paljastaa muille järjestelmille lähes ilman ohjelmointitaitoja tai tarkempaa tietoa Web Serviceistä itsestään. Tämä tavoite saavutetaan toteuttamalla rajapinta niin, että se generoi tarvittavat osat automaattisesti ajonaikaisesti, pelkästään yksinkertaisen konfiguraatiotiedoston avulla. Siis jos Merxissä on olemassa valmis moduuli, joka halutaan paljastaa palveluna, tehdään vain konfiguraatiotiedosto, jossa kerrotaan, mitä tietoja ja mitä parametreja ohjelmasta paljastetaan ulospäin Web Servicen avulla. Solteq i Web Serviceitä ohjaavia konfiguraatiotiedostoja voidaan tehdä yksinkertaisen web-selaimella käytettävän käyttöliittymän avulla tai haluttaessa millä tahansa tavallisella tekstieditorilla. Web Services -rajapinta toteutettiin Merxiä varten, mutta koko ajan ottaen huomioon, että järjestelmää tullaan käyttämään myös muiden System i -järjestelmässä toimivien sovellusten yhteydessä. Yksi näistä järjestelmistä on autokaupoille tehty Solteq CD -toiminnanohjausjärjestelmä, jonka parissa Web Services -rajapintaa käytetään irrallisena lisäosana ilman Merxiä. Tämän takia järjestelmän nimi on Solteq i Web Services eikä Merx Web Services.

Konfiguraatiotiedoston avulla Web Services -rajapinta generoi siis kaiken tarvittavan ohjelmakoodin ja kuvaukset, muun muassa WSDL-dokumentin, josta palvelua käyttävä osapuoli saa selville tarvittavat asiat siitä, miten heidän asiakassovelluksensa tulee kutsua palvelua, ja siitä, minkä muotoista tietoa Merx Web Servicet palauttavat takaisinpäin. WSDL-kuvaustiedosto ja muut ohjelmakooditiedostot pystytään rajapinnassa muodostamaan ajonaikaisesti, kun palvelua kutsutaan, mutta palvelun suoritukseen kuluvan ajan säästämiseksi tiedostot generoidaan vain, jos konfiguraatiotiedostoa on muokattu edellisen generoinnin jälkeen. Olemassa olevista Web Serviceistä muodostetaan myös rekisteri. On siis olemassa Web-selaimella käytettävä sivusto, josta voidaan nähdä, mitä Merx-moduuleita on julkaistu Web Serviceinä. Sivustolta saa selväkielisenä luettavan kuvauksen lisäksi ladattua tarvittavan WSDL-kuvauksen, joka helpottaa kut-

suvan asiakasohjelman toteuttamista useissa eri järjestelmissä. Esimerkki tästä rekisteristeristä nähdään kuvassa 1.



KUVA 1. Palvelurekisteri eli katalogi järjestelmässä olevista palveluista ja niiden ominaisuuksista

Web Serviceiden toteutuksen yhtenä lähtökohtana on alusta lähtien ollut se, että uusia palveluita voitaisiin paljastaa eli tehdä mahdollisimman helposti, ja niitä voisi tehdä kuka tahansa, joka ymmärtää Merxin toimintalogiikkaa. Näin ollen uuden palvelun tekeminen ei olisi pelkästään yhden tai kahden asiaan perehtyneen ja koulutetun ihmisen varassa. Esimerkiksi asiakasvastaavat voisivat perustaa uuden Web Serviceillä toteute-

tun palvelun valmiin RPG:llä tehdyn ohjelman päälle ilman ylimääräistä apua. Tämän ongelman ratkaisuksi kehitettiin konfiguraatiotiedostot, jotka ohjaavat koko Web Serviceiden toimintaa.

4.2 Alusta ja välineet

4.2.1 Zend Core for i5/OS ja Zend Server for IBM i

PHP IBM i -alustalla on kohtuullisen uusi asia, ensimmäinen versio virallisesta julkaisusta tuli vuonna 2006. IBM ja Zend Technologies lyöttäytyivät tuolloin yhteen ja näin syntyi Zend Core for i5/OS. Koska perinteisesti System i:llä ohjelmoidaan joko RPGllä, Cobolilla tai C:llä, on IBM ottanut vahvasti huomioon System i -toimittajien tarpeen moderneille ohjelmointikielille ja järjestelmien käytön siirtymiselle selainpohjaiseksi. Tämän takia IBM maksaa Zendille PHP:n System i version kehityksestä, ja näin ollen i-talot voivat siirtyä käyttämään PHP:ta sovelluksissaan täysin ilmaiseksi Zend Core for i5/os:n avulla. PHP:n käyttömahdollisuus IBM i:ssä on otettu todella positiivisesti vastaan i-maailmassa, ja sitä on ryhdytty hyödyntämään hyvin paljon ja monipuolisesti. Myös Solteq Merx kulkee tässä asiassa etujoukoissa, ja PHP:lle keksitään jatkuvasti uusia käyttökohteita ja tarpeita. (Anderson, Kosturjak & Mullen-Schultz 2007; Seiden 2010: Zend Server – PHP Server for IBM i 2011.)

Zend Core for i5/OS sisältää perinteiset muiltakin alustoilta tutut Apache HTTP Serverin ja PHP:n, lisäosineen, lisäksi sen mukana tulee MySQL, jonka voi asentaa tarvittaessa. Lisämaksullisessa versiossa tulee lisäksi käyttäjätuki sekä Zend Platform -tuote, joka nopeuttaa PHP:ta ja antaa joitakin lisäominaisuuksia käyttöön. Zend Core for i5/OS siis muistuttaa tuttua LAMP-ohjelmistokokonaisuutta, missä Linuxia tarkoittava L on vain vaihdettu System i:ksi. Tärkeimpänä erona muilta alustoilta tuttuihin PHP:n versioihin Zend Core for i5/OS:ssä tulee mukana lisäosa, joka mahdollistaa tietokantayhteydet System i:n DB2 for i -tietokantaan. Tämä lisäosa on AURA Equipmentsin kehittämä PHP Toolkit for IBM i, tai lyhemmin i5 toolkit. (Anderson, Kosturjak & Mullen-Schultz 2007; Seiden 2010: Zend Server – PHP Server for IBM i 2011.)

Merxin parissa suoria tietokantayhteyksiä huomattavasti tärkeämpänä ominaisuutena i5 toolkit tarjoaa mahdollisuuden kutsua suoraan RPG-ohjelmia. Koska Merxiä on kehitetty vuosikymmeniä, toiminnanohjausjärjestelmän tarvitsema liiketoimintalogiikka on olemassa RPG-ohjelmissa. Joten kun ominaisuuksia ryhdytään modernisoimaan PHP:llä, ei ole järkevää kirjoittaa logiikkaa uudelleen PHP:llä, vaan helpointa on vain ajaa jo olemassa olevia ohjelmia PHP:n avulla. Web Serviceiden perusajatus pohjautuu juuri tähän konseptiin; koska liiketoimintalogiikka on olemassa RPG-ohjelmista, käytetään vain i5 toolkitin tarjoamaa mahdollisuutta kutsua näitä moduuleita PHP:stä ja saadaan aikaan rajapinta, jonka avulla data liikkuu Merxistä verkon kautta ulkoisiin järjestelmiin. (Anderson, Kosturjak & Mullen-Schultz 2007; Seiden 2010: Zend Server – PHP Server for IBM i 2011.)

Vuonna 2010 Zend yhdisti Zend Core ja Zend Platform -tuotteet ja näin syntyi Zend Server for IBM i. Zend Server for IBM i sisältää kaikki samat ominaisuudet kuin Zend Core for i5/OS, ja lisäksi joitakin uusia ominaisuuksia ja parannuksia. Zend Serverin mukana tulee edelleen esimerkiksi AURAn i5 toolkit, mutta tämän lisäksi IBM on ryhtynyt kehittämään omaa versiotaan kyseisestä lisäosasta, tosin tämä XML toolkitin - nimellä kulkeva lisäosa on vielä tällä hetkellä beta-vaiheessa, mutta virallista julkaisua odotetaan vuoden 2011 loppuun mennessä. Zend Core for i5/OS toimii kahden Apache-palvelimen järjestelmänä, jossa toinen palvelimista on proxy-palvelin. Tämä ominaisuus on kuitenkin Zend Server for IBM i:ssä hylätty, ja käytössä on enää sisempi, System i:n natiivi Apache-palvelin; tämä ratkaisu helpottaa konfigurointia ja nopeuttaa palvelimen toimintaa. MySQL:n siirryttyä Oraclen omistukseen Oracle lopetti MySQL:n System i-version julkaisemisen. Tämän vuoksi IBM on kehittämässä Zendin kanssa yhteistyössä myös uutta MySQL-versiota Zend Serverin mukana toimitettavaksi; tämä tietokanta kantaa nimeä Zend DBi. (Anderson, Kosturjak & Mullen-Schultz 2007; Seiden 2010; Pavlak 2011; Zend Server – PHP Server for IBM i 2011.)

4.2.2 Zend Framework

PHP-maailmassa on kehitetty hyvin paljon avoimia komponenttikirjastoja, jotka toimivat runkona PHP:llä kehitettäville sovelluksille. Tunnetuimpia ja suosituimpia tällaisia frameworkejä ovat CodeIgniter, Symphony, CakePHP ja Zend Framework. Kompo-

nenttikirjastoihin on kirjoitettu PHP:llä yleisimmin käytetyt toiminnallisuudet valmiiksi, jotta voidaan välttää niin kutsuttu pyörän uudelleen keksiminen kehitettäessä uutta PHP-sovellusta. Kirjastot ovat siten helposti laajennettavissa sovelluksen tarvitsemien erikoispiirteiden mukaan. Zend Frameworkiä kehittävät Zend Technologies Ltdssä muun muassa samat tahot, jotka kehittävät ja julkaisevat myös Zend Serveriä, virallista kaupallista versiota itse PHP:stä, minkä takia se tulee myös Zend Serverin ja Zend Server Community Editionin mukaan paketoituna valmiiksi. (Brady 2009; Seiden 2010; Zend Framework 2011.)

Zend Framework on, toisin kuin esimerkiksi CakePHP, heikosti yhteenliitetty (loosely coupled) kirjasto. Tällä tarkoitetaan sitä, että lähes kaikkia Zend Frameworkin yksittäisiä komponentteja voidaan käyttää riippumatta toisista komponenteista. Näin ollen Zend Frameworkin kattavasta komponenttivalikoimasta voidaan valita ainoastaan ne osat, joita omassa sovelluksessa todella tarvitaan. Toisaalta Zend Frameworkiä voidaan käyttää myös ”täyden pinon” (full stack) frameworkinä, jos halutaan ottaa MVC:stä lähtien kaikki sen ominaisuudet käyttöön. (Brady 2009; Seiden 2010; Zend Framework 2011.)

Zend Frameworkin moduuleista on Web Serviceiden parissa hyödynnetty muun muassa Zend_CodeGenerator -komponenttia, jota käytetään Web Serviceiden PHP-luokkien luomiseen ajonaikaisesti, aiemmin mainittujen konfiguraatiotiedostojen perusteella. Myös Zend Frameworkin tarjoamaa MVC-arkkitehtuuria on käytetty hyväksi Web Servicen toimintojen ohjauksessa ja erityisesti konfiguraatiotiedostojen luomista ja muokkaamista varten tarkoitetussa käyttöliittymässä. (Brady 2009; Seiden 2010; Zend Framework 2011.)

4.2.3 NuSOAP

NuSOAP on alun perin Dietrich Ayalan kehittämä PHP-luokkakirjasto SOAP Web Serviceiden tekemisen ja käyttämisen helpottamiseksi. Nykyisin luokkakirjastoa ylläpitää satunnaisesti vapaa-ajallaan Scott Nichol. Ayala kehitti NuSOAPin PHP:n versiolle 4, jolloin PHP:ssä ei vielä ollut sisäistä tukea Web Serviceille, nykyisin PHP:n mukana toimitetaan lisäosa, joka toteuttaa natiivisti suuren osan NuSOAPin toiminnoista, mutta NuSOAPin käytölle on vieläkin olemassa perusteet tietynlaisia Web Serviceitä käytettä-

essä. PHP:n omassa SOAP-toteutuksessa tuetaan lähinnä RPC/encoded -tyyppisiä SOAP Web Serviceitä, jotka taas eivät sovi kovin hyvin yksiin Microsoftin tuotteiden Web Serviceiden kanssa, koska ne käyttävät Document/literal wrapped -tapaa. (Butek 2003; Nichol 2004; Dietrich & Nichol 2004-2011.)

NuSOAPissa on hyvät ja huonot puolensa. Huonoina puolina on muun muassa hitaus, viimeistelemättömyys, tiettyjen ominaisuuksien puuttuminen ja vakituisen ylläpidon puute. Toisaalta koska NuSOAP on PHP-luokkakirjasto, se tarkoittaa myös sitä, että vaikka sitä ylläpidetään ja uusia ominaisuuksia kehitetään hyvin satunnaisesti, voidaan siihen tehdä näitä toimenpiteitä myös Solteqissa. Jos kyseessä olisi PHP:n sisään rakennettu ominaisuus tai lisäosa, ei tällainen moduulin muokkaaminen omiin tarpeisiin onnistuisikaan niin helposti. Koska kyseinen muokkaaminen on NuSOAPin kohdalla mahdollista, on sitä myös käytetty hyödyksi; olen itse lisännyt joitakin uusia ominaisuuksia ja virheiden korjauksia suoraan NuSOAPin koodiin. Olen myös ilmoittanut nämä puutteet ja virheet NuSOAPin foorumeilla sekä lisännyt posteihin omat ratkaisuni asioiden korjaamiseksi. Suuri osa näistä koodimuunnoksista on näin myös päätynyt itse NuSOAPin julkaisun koodiin ajan kuluessa. NuSOAPin hyviin puoliin kuuluu tietenkin myös se, että se tukee suoraan jo hyvin Merxin Web Service -rajapinnan eniten tarvitsemia ominaisuuksia keskusteltaessa Microsoftin .NET sekä muiden vastaavien Web Serviceiden kanssa. (Nichol 2004; Dietrich & Nichol 2004-2011.)

4.3 Ominaisuudet

4.3.1 Palvelut (Provider)

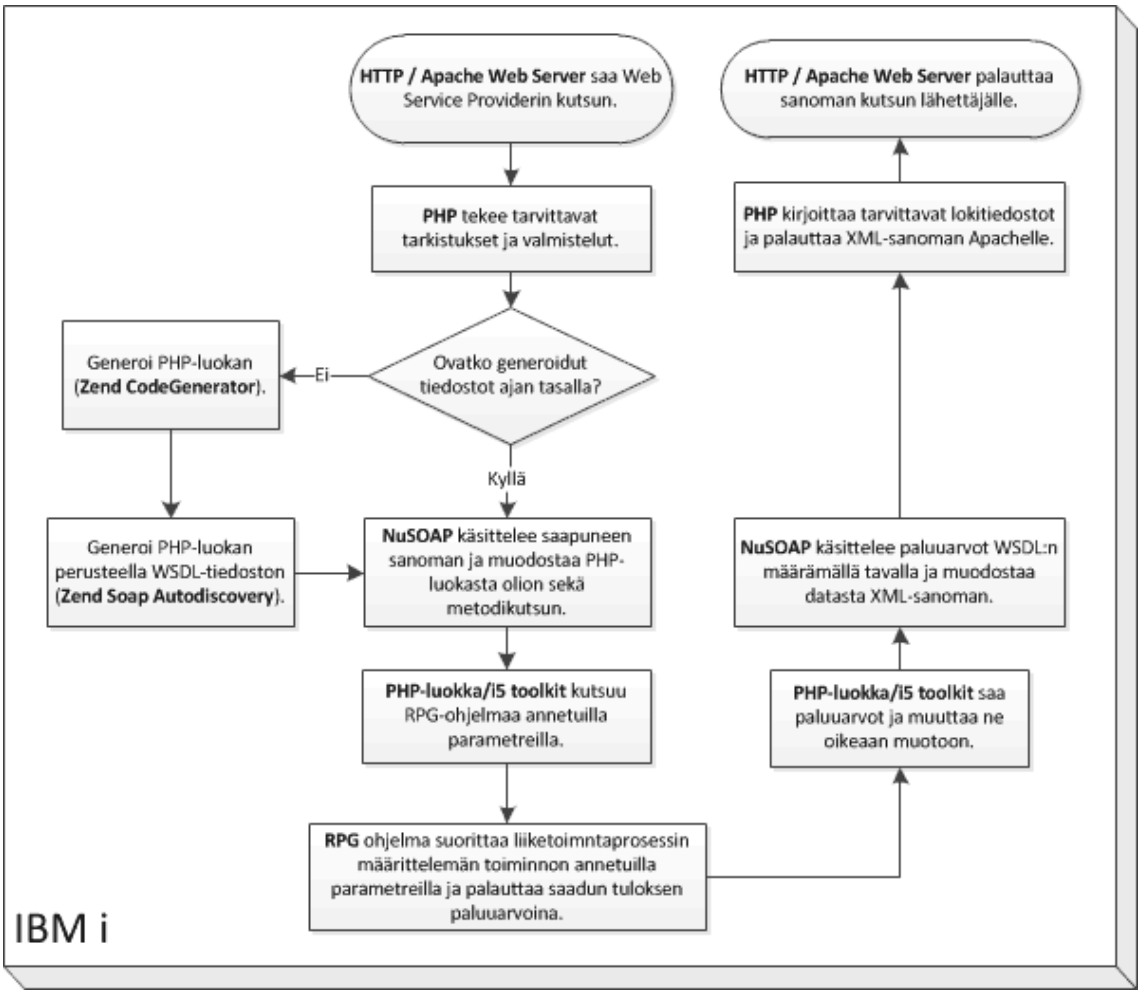
Web Service Provider on Web Service -rajapinnan toteutus palveluista, joita Merxistä tai jostain toisesta RPG:llä ohjelmoidusta IBM i -ohjelmistosta paljastetaan ulkoisille sovelluksille ja järjestelmille. Provider on alkuvaiheessa Solteq i Web Service -rajapinnan tärkein ominaisuus, koska sille on jo olemassa tarkkaan määriteltyjä käyttökohteita. Palvelut voidaan jakaa kahteen eri päätyyppiin, Merxistä tietoa hakeviin palveluihin eli ”pull”-tyyppisiin, mikä on selvästi yleisin käyttötapa, mutta myös Merxiin tietoa tuoviin palveluihin eli ”push”-tyyppisiin palveluihin. Yleisin providerin käyttöta-

pa on tosiaan se, että Merxin sisältämää tietoa tarvitaan jossain ulkoisessa järjestelmässä, jolloin tehdään palvelu, joka ottaa vastaan hakuparametreja ja palauttaa parametreilla haetun datan. Toisaalta välillä on niin, että jostain ulkoisesta järjestelmästä pitää tuoda tietoa Merxiin siten, että kommunikoinnin aloittajana on ulkoinen järjestelmä. Tällöin ei käytetä Web Services -rajapinnan Requester-ominaisuutta, vaan käytetään ”push”-tyyppistä palvelua, jolla mahdollistetaan Merx-datan päivittäminen ulkoisen järjestelmän ollessa aloitteellinen.

Provider toimii siis niin, että joko käsin tai toteutetun käyttöliittymän avulla muodostetaan konfiguraatiotiedosto, jossa on kerrottu kaikki tiedot, jotka tarvitaan Web Servicen muodostamiseen. Konfiguraatiotiedoston ominaisuuksista ja sisällöstä kerrotaan tarkemmin luvussa 4.3.3 Käyttöliittymä. Jotta RPG-ohjelman päälle voidaan tehdä Web Service, pitää muodostaa PHP-luokka, jolla ohjelmaa voidaan i5 toolkitin tai vastaavan RPG/PHP-rajapinnan avulla kutsua, tämän lisäksi tarvitaan myös WSDL-tiedosto, joka kuvaa Web Servicen ominaisuudet ulospäin sovellukselle, joka kutsuu kyseistä palvelua. Esimerkit rajapinnalla generoiduista PHP-luokkatiedostosta ja WSDL-tiedostosta löytyvät liitteistä 2 ja 3.

Kun konfiguraatiotiedosto on muodostettu, pystyy rajapinta ajonaikaisesti tuottamaan tarvittavat PHP-luokan ja WSDL-tiedoston. PHP-luokan generoimisessa käytetään apuna Zend Frameworkin mukana tulevaa Zend CodeGenerator -moduulia, joka on tarkoitettu helpottamaan ajonaikaisesti tehtävien luokkien, metodien ja ohjelmakoodin muodostamista. Muodostetussa PHP-tiedostossa on kyseisen palvelun käyttämien tietorakenteiden oliomalliset kuvaukset sekä tarvittavat ohjelmakoodit RPG-ohjelman kutsumiseen i5 toolkitin avulla; kaikki nämä muodostetaan konfiguraatiotiedoston määräämien asetusten perusteella. Muistiin ajonaikaisesti luotu PHP-ohjelmakoodi tallennetaan myös palvelimen kovalevylle, tavalliseksi PHP-tiedostoksi. Palvelun toiminnan nopeuttamiseksi tarvittava ohjelmakoodi luetaan palvelua kutsuttaessa suoraan tallennetusta tiedostosta, jos palvelun kuvaava konfiguraatiotiedosto ei ole muuttunut sen jälkeen, kun kyseinen väliaikaistiedosto on viimeksi tallennettu. Tarvittava WSDL-tiedosto muodostetaan samaan tapaan käyttäen apuna Zend Frameworkin Zend Soap AutoDiscovery -moduulia. Tämä moduuli helpottaa huomattavasti SOAP-tyylisten Web Serviceiden vaatiman WSDL-tiedoston generoimista ajonaikaisesti; kun edellä mainittu automaattisesti luotu PHP-luokka on muodostettu oikein ja sisältää riittävät tiedot kerrot-

tuna muun muassa kommenttilohkoissa, AutoDiscovery osaa luoda WSDL-tiedoston automaattisesti. Kuviossa 1 havainnollistetaan Web Services -rajapinnan toimintaa prosessikaaviolla. (Programmer's Reference Guide: Zend_CodeGenerator 2011; Programmer's Reference Guide: Zend_Soap_Autodiscovery 2011.)



KUVIO 1. Web Services -rajapinnan toimintaa kuvaava prosessikaavio

4.3.2 Pyynnöt (Requester)

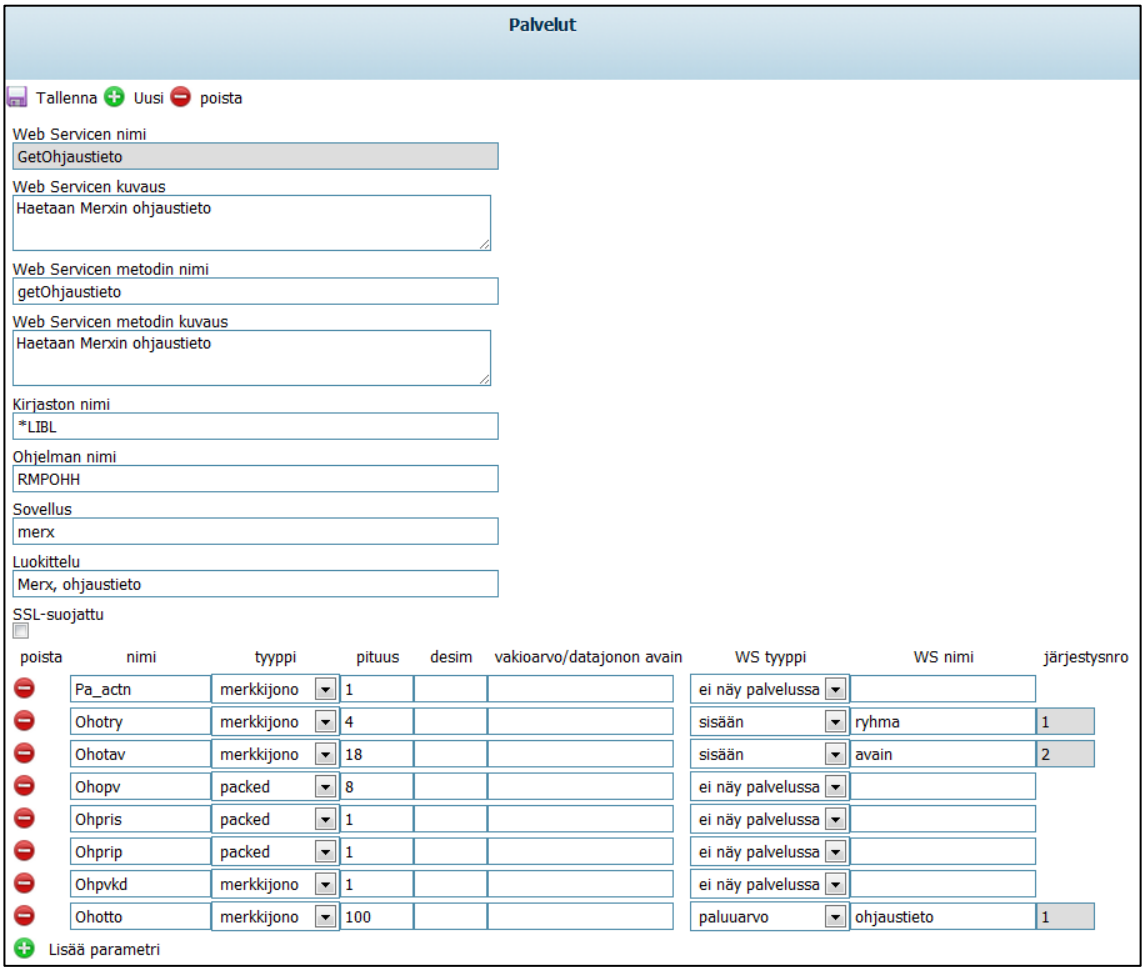
Web Services -rajapinta tulee myös omien ulospäin paljastettavien palveluiden lisäksi toteuttamaan samaan tapaan muissa järjestelmissä paljastettujen palvelujen hyödyntämisen Merxissä. Nämä Requester Web Servicet ovat siis asiakassovelluksia, jotka pyytävät tietoa jossain toisessa järjestelmässä toteutetulta palvelulta. Web Services -rajapinnassa nämä on toteutettu samaan tapaan konfiguraatiotiedostojen avulla. Web Service -pyynnön tekevä ohjelma toimii siis taas automaattisesti, eikä varsinaisista Web Service -

tekniikoista tarvitse tässäkään tapauksessa ymmärtää mitään rajapintaa käytettäessä. Toisin kuin palveluissa, pyyntöihin pitää kuitenkin tehdä uudet sovellukset Merxiin, koska vaikka rajapinta huolehtii varsinaisesta pyynnöstä ja tarjoaa standardin tavan suorittaa varsinainen kutsu, pitää parametrien lähetys palvelulle ja paluudatan keruu suorittaa Merxissä hieman totutusta poikkeavasti. Tosin tämä voidaan todennäköisesti osassa tapauksista toteuttaa käyttämällä olemassa olevan sovelluksen ja Web Services -rajapinnan välissä olevaa "kuorta", joka hoitaa tarvittavat muutokset. Myös Requester-serviceistä tulee web-selaimella käytettävä rekisteri ja konfiguraatiotiedostojen muokkaukseen tarkoitettu käyttöliittymä. Asiakas-servicen luonnissa auttaa myös palvelun toteuttajalta saatava WSDL-kuvaus, samaan tapaan kuin palveluissa. Myös Requester-puolella on luonnollisesti mahdollisuus käyttää joko tiedon hakemiseen tarkoitettuja "pull"-tyyppisiä palveluja tai lähettää Merxistä tietoa ulkoiseen järjestelmään "push"-tyyppiselle palvelulle.

Yksi Web Service -rajapinnan Requesterin tärkeimpiä komponentteja on RMWSREQ-niminen (nimi tulee sanoista RPG-Merx-Web-Service-Requester) RPG-ohjelma. Tämä ohjelma toimii välittäjänä PHP:llä tehdyn rajapinta-osuuden ja Web Service -kutsun tekevän RPG-ohjelman välissä. RMWSREQ ottaa parametreina vastaan kutsuttavan Web Servicen nimen, jonka avulla tunnistetaan oikea konfiguraatiotiedosto ja 2048 merkkiä pitkä merkkijono, jossa voidaan välittää parametrit erotettuina toisistaan määrätyillä erotinmerkeillä. Tätä parametrien välitystapaa käytetään, jos kutsuttavan palvelun kutsuparametrit eivät sisällä taulukoita. Jos parametrit ovat tai sisältävät taulukoita, viedään ne PHP:n käsiteltäväksi datajonon avulla, tietyssä positiosidonnaisessa formaatissa. RMWSREQ ottaa yhteyden Web Services -rajapinan PHP-käsittelijään ja NuSOAPiin normaalin HTTP-socket -yhteyden kautta, socket- yhteys muodostetaan paikalliseen koneeseen ja Apachelle määriteltyyn porttiin. Tämän jälkeen parametrit siirretään tavalliseen tapaan POST-parametreina PHP:lle. RMWSREQ siis hoitaa kaiken kommunikoinnin PHP:n kanssa, joten tässäkään ei Web Servicen tekijän tarvitse oppia uusia tekniikoita ja teknologioita, riittää kun kutsuu normaalista RPG-ohjelmasta RMWSREQ:a ja välittää parametrit oikeassa muodossa. Paluuparametrit kutsuneelle ohjelmalle tulevat joko samaan tapaan kuin kutsuttaessa, 2048 merkin jonossa, tai taulukoiden ollessa kyseessä datajonon kautta.

4.3.3 Käyttöliittymä

Web Services -rajapinnan tarvitsemia konfiguraatiotiedostoja voidaan tehdä käsin editoimalla tavallisia tekstitiedostoja millä tahansa tekstieditorilla. Koska konfiguraatiotiedostojen täytyy kuitenkin sisältää lukuisia asioita juuri tietyillä tavoilla määriteltyinä, oikein muotoillun ja pakolliset asiat sisältävän konfiguraatiotiedoston luomista varten toteutettiin myös web-selaimella käytettävä editori, jonka ulkoasu nähdään kuvasta 2.



KUVA 2. Palveluiden konfiguraatiotiedostoja voidaan muokata web-selaimella toimivassa käyttöliittymässä

Konfiguraatiotiedostossa täytyy olla kerrottuna tiettyjä pakollisia asioita, näihin lukeutuvat seuraavat:

- Kutsuttavan RPG-ohjelman nimi ja kirjasto tai kirjastolista, jolta kyseinen ohjelma löytyy. Yleensä kirjastolista on *LIBL, joka tarkoittaa sitä, että käytetään

kirjastolistaa, joka on valmiiksi asetettu käytettävän IBM i -käyttäjäprofiilin asetuksiin.

- Web Servicen nimi, jolla se näkyy ulkoisiin järjestelmiin; nimenä käytetään yleisesti Web Servicen konfiguraatiotiedoston nimeä.
- Web Servicen metodi, joka paljastetaan ulkoisille järjestelmille. Metodeja voi olla useampiakin yhdessä palvelussa, mutta rajapintaa toteutettaessa tehtiin päätös, että käytetään vain yhtä metodia jokaista palvelua kohti; metodin nimi on yleensä sama kuin servicen nimi, mutta pienellä alkukirjaimella.
- Web Servicessä ulospäin paljastettavien parametrien ja paluuarvojen konfiguraatiotiedot. Näihin kuuluvat kyseisten muuttujien yhdistäminen RPG-ohjelman kutsuparametreihin, tietotyyppi, pituus ja järjestys kutsussa sekä mahdollinen vakioarvo. Kaikkia RPG-ohjelman parametreja ei tarvitse paljastaa ulospäin palvelussa.
- RPG-ohjelman *ENTRY-parametrit tarkkoine määrityksineen. Nämä toimivat samaan tapaan kuin PHP-funktio-kutsujen parametrit, sillä erotuksella, että kyseisessä määrityksessä sekä parametrit että paluuarvot määritellään samalla ja ne ovat kaikki sekä ohjelmaan sisään tulevia että paluuarvoina takaisin tulevia muuttujia.

Lisäksi Web Serviceiden konfiguraatiotiedostossa voidaan kertoa tarvittaessa joitakin valinnaisia ominaisuuksia, joista seuraavassa tärkeimmät:

- Sanallinen kuvaus palvelun tarkoituksesta ja ominaisuuksista.
- Kyseisen palvelun SSL-suojauksen käyttö tiedonsiirrossa.
- Käyttäjätunnus ja salasana, jos palvelu on salasanasuojattu.
- Sovelluksen nimi, johon Web Service kuuluu, esim. Merx, CD yms.
- Tagit eli tunnisteet, joilla voidaan lyhyesti kertoa, mihin kategorioihin kyseinen palvelu kuuluu; näin helpotetaan tietyn palvelun hakemista palvelurekisteristä.

Koska SOAPin luonteeseen kuuluu se, että palvelun tarjoajan muodostama WSDL-kuvaustiedosto kertoo kaiken tarvittavan kyseisen palvelun käyttämisestä, on Requesterin konfiguraatiotiedoston tekeminen helpompaa kuin Providerin. Requesterin konfiguraatiotiedostossa voidaan kertoa seuraavaa:
WSDL:n osoite.

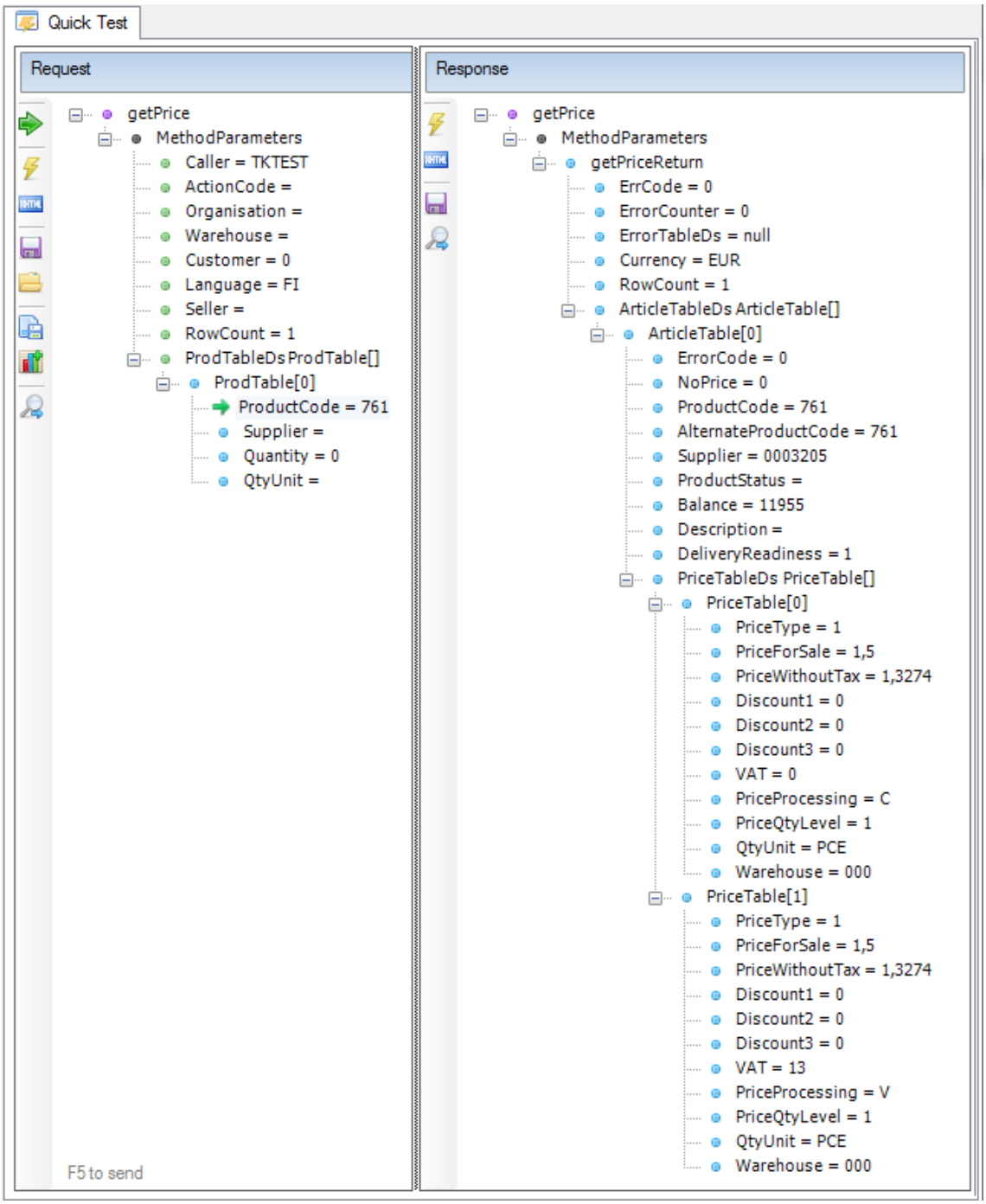
- Käyttäjätunnus ja salasana, jos palvelu on salasanasuojattu.

- Web Servicen nimi.
- Sovelluksen nimi, johon Web Service kuuluu, esim. Merx, CD yms.
- Käytettävän metodin nimi. Ulkoisissa järjestelmissä voi, toisin kuin Solteq i Web Services Providerissa, olla samalla palvelulla useita metodeja.
- Sanalliset kuvaukset palvelusta ja käytettävästä metodista.
- Tagit eli tunnisteet.

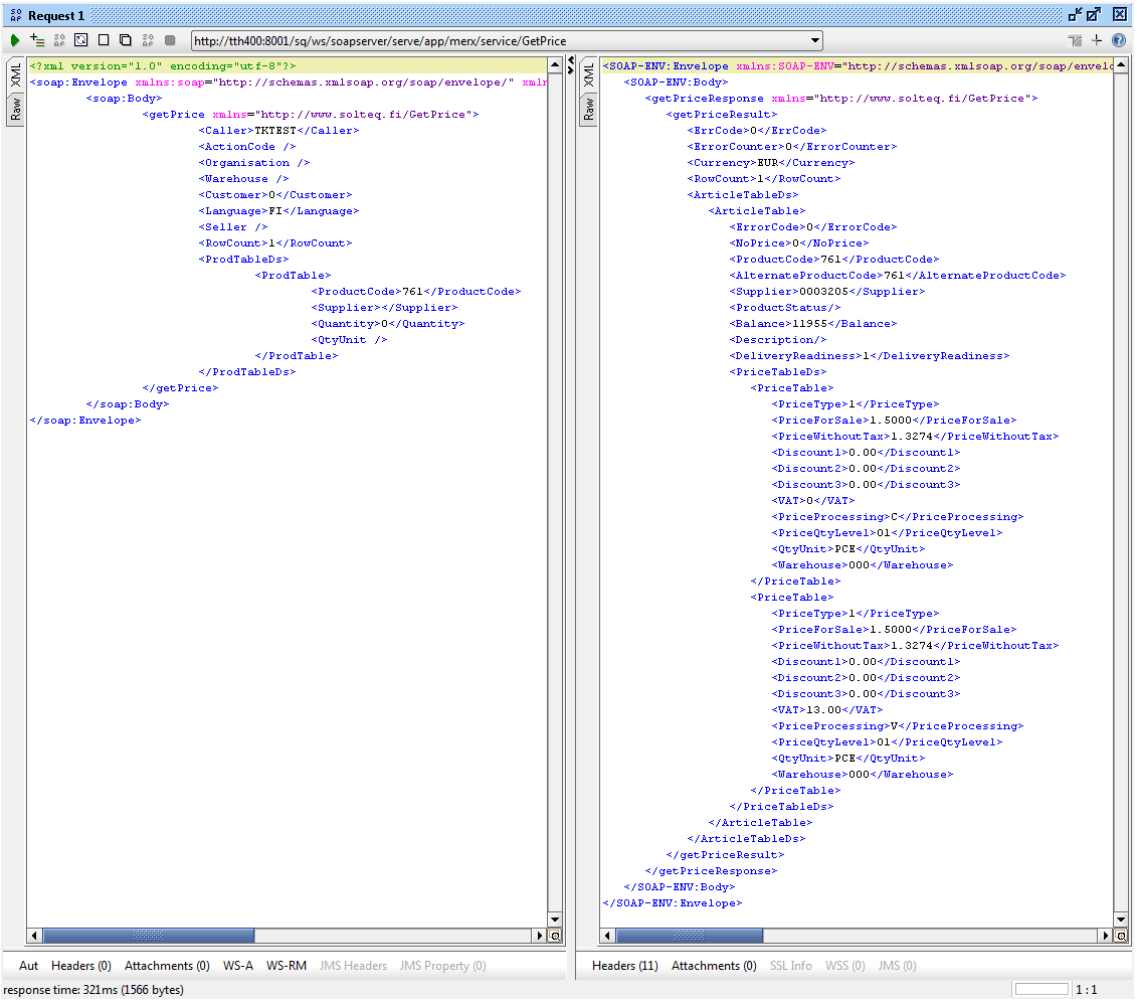
Näiden lisäksi pitää yhdistää Merxistä ulospäin lähtevät parametrit WSDL:n kuvauksessa annettuihin parametreihin: parametrit välitetään rajapinnalle joko HTTP:n GET-parametreina, jos kyseessä on vain yksittäisiä parametreja, tai luetaan PHP:llä datajonosta, johon ne on rajapinnan RPG-ohjelmalla kirjoitettu.
Konfiguraatiotiedostossa pitää siis olla kerrottuna joko, miten GETllä tulleet parametrit ohjataan kutsuttavaan palveluun tai, millä avaimella parametrit luetaan datajonosta, jotta tarvittava Requester-sanoma voidaan muodostaa.

4.4 Testaus

Yhtenä SOAP Web Serviceiden hyvänä puolena on valmiiden testaukseen tarkoitettujen työkalujen saatavuus. Koska asiakasohjelma perustuu WSDL-kuvauksen käyttöön, on olemassa lukuisia yleisluontoisia client-ohjelmia, niin web-selaimella käytettävinä kuin koneelle asennettavina versioinakin. Seuraavilla sivuilla löytyvissä kuvissa nähdään esimerkit kahdesta koneelle asennettavasta sovelluksesta, joilla Web Serviceitä voidaan helposti testata. Kun sovellukseen annetaan URL, josta halutun Web Servicen WSDL-kuvaus löytyy, muodostavat molemmat sovellukset valmiin sanomapohjan, johon käyttäjä voi syöttää halutut arvot. Kuvan 3 WCFStorm-sovellus esittää sekä annettavat parametrit että vastauksena saadut paluuarvot graafisesti. Hieman monipuolisemmassa SoapUI-sovelluksessa, joka näkyy kuvassa 4, on taas pelkistetympi käyttöliittymä sanomien esityksen suhteen. Toisaalta tämän esitysmuodon selkeänä etuna on se, että molemmat sanomat näytetään täydellisinä, eli juuri sellaisina kuin ne verkossa liikkuvat.



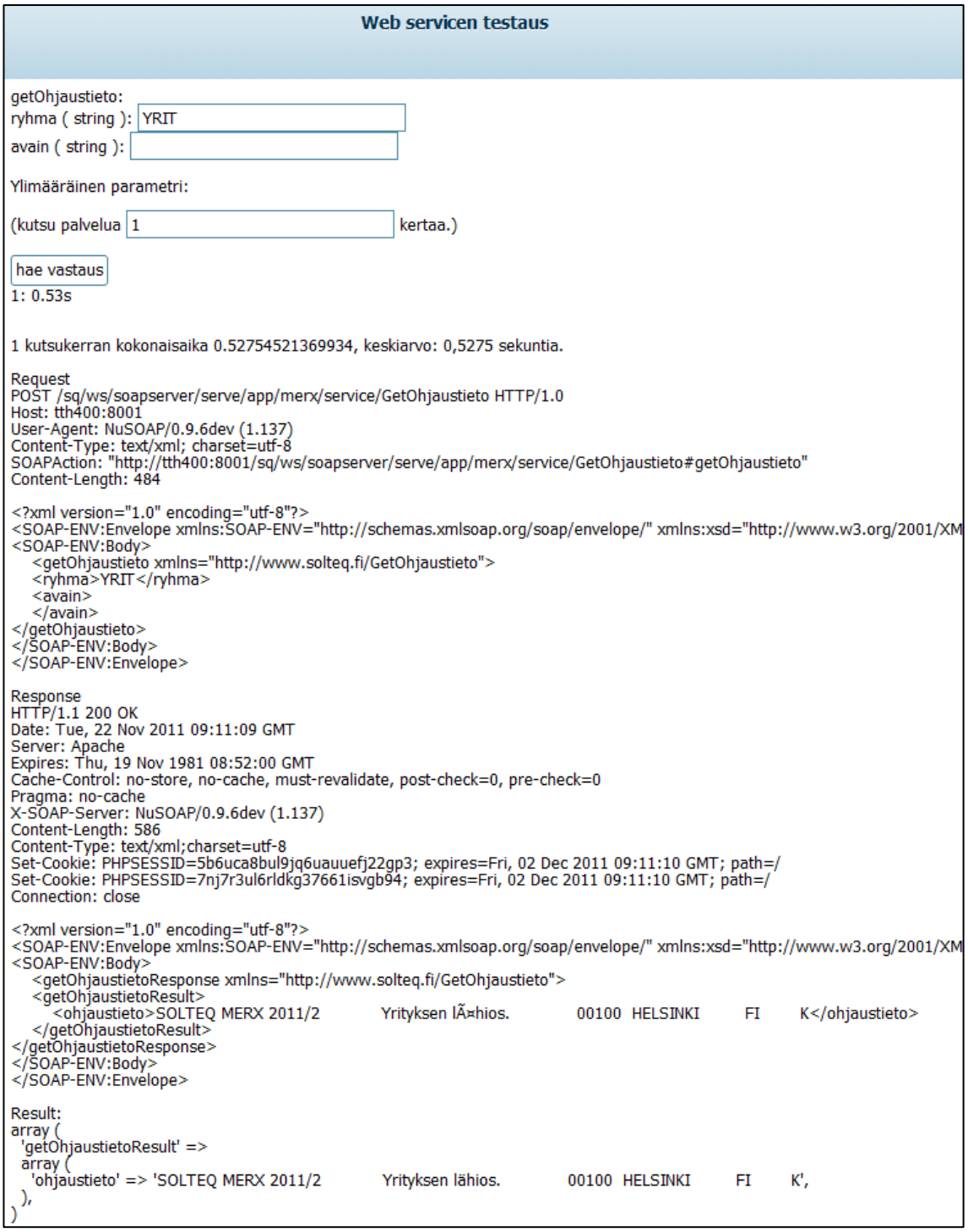
KUVA 3. Esimerkki WCFStorm-ohjelmalla suoritetusta Web Service -kyselystä



KUVA 4. Esimerkki SoapUI-ohjelmalla suoritetusta Web Service -kyselystä

Testauksessa on edellä mainittujen ohjelmien lisäksi käytetty myös esimerkiksi Requester – Provider -paria, jossa RPG:llä tehty Web Services -rajapinnan Requester -puolta käyttävä sovellus kutsuu palvelua, joka on tehty rajapinnan Providerilla, eli kutsuu siis samassa järjestelmässä olevaa toista RPG-ohjelmaa. Tästä testitapauksestahan ei ole siis mitään varsinaista käytännön hyötyä, mutta siinä tulee yhdellä RPG-ohjelman kutsulla todistettua, että Web Services -rajapinnan täydellinen kutsuketju menee kokonaisuudessaan läpi. Lisäksi testauksessa ja ongelmanselvityksessä on huomattavaa apua siitä, että rajapinta kirjoittaa lokitiedostoja kaikista sanomista halutulla tasolla. Järjestelmästä löytyvät siis lokitiedostot palvelukohtaisesti saapuneille sanomille, vastaussanomille sekä erillinen lokitiedosto sille, kuinka kauan palvelun suorittamiseen kului aikaa. Esimerkit lokitiedostoista, joihin on kirjattu GetPrice-palveluun tehty kysely ja lähetetty vastaus, löytyvät liitteistä 4 ja 5. Lisäksi Solteq i Web Serviceiden käyttöliittymään on tehty hyvin yksinkertainen, palvelun nopeaa testausta helpottamaan tarkoitettu PHP-sovellus. Tämä sovellus tekee WSDL-tiedoston perusteella HTML-lomakkeen, jolla voidaan lä-

hettää palvelulle parametrit ja saadaan näkyviin kyselysanoma ja vastaussanoma sellaisinaan. Esimerkki tällä testaussovelluksella tehdystä kyselystä ja vastauksesta nähdään kuvassa 5. Sovelluksella voidaan myös testata palvelun vastausnopeutta ajamalla useampia peräkkäisiä kyselyitä kerralla.



Web servicen testaus

getOhjaustieto:
ryhma (string): YRIT
avain (string):

Ylimääräinen parametri:

(kutsu palvelua 1 kertaa.)

hae vastaus
1: 0.53s

1 kutsukerran kokonaisaika 0.52754521369934, keskiarvo: 0,5275 sekuntia.

```
Request
POST /sq/ws/soapserver/serve/app/merx/service/GetOhjaustieto HTTP/1.0
Host: tth400:8001
User-Agent: NuSOAP/0.9.6dev (1.137)
Content-Type: text/xml; charset=utf-8
SOAPAction: "http://tth400:8001/sq/ws/soapserver/serve/app/merx/service/GetOhjaustieto#getOhjaustieto"
Content-Length: 484

<?xml version="1.0" encoding="utf-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://www.w3.org/2001/XM
<SOAP-ENV:Body>
   <getOhjaustieto xmlns="http://www.solteq.fi/GetOhjaustieto">
   <ryhma>YRIT</ryhma>
   <avain>
   </avain>
</getOhjaustieto>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>

Response
HTTP/1.1 200 OK
Date: Tue, 22 Nov 2011 09:11:09 GMT
Server: Apache
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
X-SOAP-Server: NuSOAP/0.9.6dev (1.137)
Content-Length: 586
Content-Type: text/xml;charset=utf-8
Set-Cookie: PHPSESSID=5b6uca8bul9jq6uauuefj22gp3; expires=Fri, 02 Dec 2011 09:11:10 GMT; path=/
Set-Cookie: PHPSESSID=7nj7r3ul6rldkg37661isvgb94; expires=Fri, 02 Dec 2011 09:11:10 GMT; path=/
Connection: close

<?xml version="1.0" encoding="utf-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://www.w3.org/2001/XM
<SOAP-ENV:Body>
   <getOhjaustietoResponse xmlns="http://www.solteq.fi/GetOhjaustieto">
   <getOhjaustietoResult>
      <ohjaustieto>SOLTEQ MERX 2011/2          Yrityksen lÃ¤hios.          00100  HELSINKI       FI       K</ohjaustieto>
   </getOhjaustietoResult>
</getOhjaustietoResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>

Result:
array (
  'getOhjaustietoResult' =>
  array (
    'ohjaustieto' => 'SOLTEQ MERX 2011/2          Yrityksen lähios.          00100  HELSINKI       FI       K',
  ),
)
```

KUVA 5. Rajapintaan toteutetulla testaus-sovelluksella tehty kysely GetOhjaustieto-palveluun

5 TULOKSET JA JOHTOPÄÄTÖKSET

5.1 Tuotantokäyttö

Nykypäivän liiketoimintaa harjoittavissa yrityksissä on pakko olla käytössä järjestelmäintegraatiovälineitä; samaa tietoa tarvitaan useissa eri järjestelmissä, mutta sitä ei ole järkevää tallentaa jokaiseen järjestelmään erikseen. Jos yksi järjestelmä käsittelee tiettyä tietoa ja sillä on olemassa metodit tuon tiedon käsittelyyn sekä kyseinen tieto, on järkevin ja tehokkain tapa paljastaa tuo tieto ja prosessi ulkoisille järjestelmille esimerkiksi Web Serviceiden avulla. Web Services -rajapinta on nyt ollut tuotantokäytössä Solteqin asiakkailla pisimmillään reilusti yli vuoden, ja kaikkiaan Web Service -rajapinta on toimitettu ja joko otettu tuotantokäyttöön tai tulossa tuotantokäyttöön lähiaikoina kuudella Solteq Merx -asiakkaalla ja kolmella Solteq CD -asiakkaalla. Web Service -rajapintaan halutaan jatkuvasti uusia ominaisuuksia, ja uusia palveluita perustetaan tiuhaan tahtiin. Web Services -rajapinnan olemassa oleminen on tuonut Solteqille uusia projekteja, jotka eivät olisi muuten ehkä toteutuneet ollenkaan tai olisivat toteutuneet hyvin erilaisessa muodossa. Näiden seikkojen pohjalta voidaan sanoa, että Web Services -rajapinnan toteuttaminen on ollut hyvin onnistunut ja tuottanut niitä tuloksia, joita siltä kaivattiin, kun toteutusta lähdettiin suunnittelemaan.

Työn johdannossa esiteltiin liiketoimintakysymyksiä, joihin pyrittiin vastaamaan Web Services -rajapinnan toteutuksella. Nyt kun rajapinta on asiakkailla tuotantokäytössä, voidaan sanoa, että näihin kysymyksiin ollaan pystytty vastaamaan. Kun järjestelmäintegraatioprojektissa voidaan käyttää avuksi Web Services -rajapintaa, ei varsinaista integraatiota varten tarvitse tehdä kuin rajapinnan konfiguraatiotiedosto, jos tarvittavan toiminnon toteuttava RPG-ohjelma on jo olemassa. Kyseisen RPG-ohjelman toteuttaminen on Solteq Merx -ohjelmoijille tuttua jokapäiväistä työtä, joten integraatiosta selvitään ilman erityisosaamista ja resurssit ovat saatavilla, mikä tietenkin helpottaa ja nopeuttaa järjestelmäintegraatioprojektien toteuttamista sekä tekee niistä halvempia. Myöskään System i:n negatiiviset puolet eivät enää näy integraation toiselle osapuolelle; sanomat kulkevat Web Services -rajapinnan avulla ulospäin standardeina XML-sanomina, joiden merkistö on ASCII ja koodaus UTF-8. Näiden käsitteleminen onnistuu nykypäivänä lähes jokaisessa järjestelmässä, toisin kuin System i:n natiivin EBCDIC-

merkistön. Solteq Merx on nyt helpompi liittää myös Solteqin omaan tarjontaan, esimerkkinä tästä toimii hyvin integraatio Solteq Kassan kanssa. Microsoftin .NET -ohjelmistokirjastolla ohjelmoitu, Windows-koneessa toimiva kassajärjestelmä keskustelee reaaliaikaisesti Merxin kanssa käyttäen avuksi Web Services -rajapinnan tarjoamia palveluita. Tämä toimii samalla esimerkkinä siitä, miten Web Services -rajapinnan avulla voidaan käyttää Merxiä käyttäjäpääte-tyyppisellä järjestelmällä. Samanlaisia integraatioita Merxin ja muiden Solteqin tuotteiden välille tullaan varmasti toteuttamaan jatkossa lisää.

Loistava esimerkki siitä, miten Web Services -rajapinta on tehnyt integraatioprojekteista kokonaisvaltaisesti kustannustehokkaampia ja miten Merxiin saadaan standardit, uudelleen käytettävissä olevat rajapinnat, on juuri käyntiin lähdössä oleva uusi asiakasprojekti. Rajapintaan on tehty yhdelle asiakkaalle tuotantokäyttöön palvelut jo aiemmin mainittua kassan rahastuspäätettä varten, ja toisella asiakkaalla Web Servicejä käytetään suuren suomalaisen yrityksen B2B-verkkokaupassa muun muassa tuotteiden hintojen ja saldojen hakemiseen Merxistä. Kolmannelle asiakkaalle tehdään palveluja kuluttaja-verkkokauppaa varten, joten tässä voidaan hyödyntää jo B2B-verkkokauppaa varten toteutettuja palveluita suoraan. Nyt ollaan käynnistämässä neljännelle asiakkaalle uutta integraatioprojektia, jossa tullaan käyttämään avuksi kaikkia edellä mainittuja, muille asiakkaille tehtyjä Web Servicejä; iso osa projektin tarvitsemasta työmäärästä on siis jo valmiiksi tehty projektin alkaessa.

5.2 Jatkokehitys

Kuten tietojärjestelmät yleensä, myöskään Web Services -rajapinta ei tule koskaan olemaan valmis. Muuttuvia ja lisääntyviä tarpeita uusille ominaisuuksille syntyy ja ilmenee jatkuvasti. Kasvavat tiedonsiirtotarpeet, haastavammat ja aikakriittisemmät palvelut edellyttävät vaihtoehtoisten toimintametodien tutkimista, vaikka nykyiset toimintamallit on todettu toimiviksi ja käytännöllisiksi. Uusien projektien ja kehityssuunnitelmien kautta on jo nyt tunnistettu suuri määrä asioita, joita rajapintaan pitää tulevaisuudessa toteuttaa, jotta asiakkaita voidaan palvella paremmin. Tässä luvussa esitellään lyhyesti kyseisiä tarpeita.

Rajapinnassa on nyt jo kutsupuolella mahdollisuus valita SOAP Web Servicen ja tietynlaisen REST-tyyppisten Web Servicen välillä konfiguraatiotiedoston parametreilla. Jatkossa voitaisiin nopeuttaa myös osaa SOAP-tyyppisistä Serviceistä käyttämällä PHP:n natiivia lisäosaa silloin, kun se on mahdollista, ja käyttää nykyistä NuSOAP -rajapintaa vain silloin, kun keskustelukumppanin ominaisuudet tätä vaativat.

On havaittu, että SOAP Web Serviceiden suorittaminen NuSOAPilla on hyvin raskasta, jos kutsussa tai paluudatassa on paljon sisäkkäisiä taulukoita. Tällöin on järkevää harkita rajapintaan toteutettavaa lisäominaisuutta, jonka avulla palvelupuolella (provider) voidaan toteuttaa haluttu palvelu tarvittaessa REST-tyyppisiä tekniikoita käyttämällä. Tämän toiminnallisuuden toteuttaminen rajapintaan pitäisi itse asiassa olla verraten helppoa. Ainoat asiat, jotka kutsuketjussa muuttuvat, ovat sisään tulevan parametrisanoman jäsennys jo olemassa olevan generoidun PHP-luokan kutsun vaatimaan muotoon ja verraten taas takaisinpäin tulleen PHP-luokan paluuarvoista muodostettavan paluusanoman käsitteleminen haluttuun muotoon. Paluusanoman muotokin voitaisiin tässä tapauksessa vapaasti valita, esimerkiksi XML:n ja JSON:n välillä konfiguraatiotiedoston parametreilla. Näin voitaisiin leikata siivu pois PHP:n kuluttamasta Web Servicen suoritusajasta sekä pienentää lähetettävän sanoman kokoa valittaessa paluudatan muodoksi JSON.

Vaikka konfiguraatiotiedostojen muokkaamista varten on tehty käyttöliittymä, on se aikapulan vuoksi jäänyt hieman vajavaiseksi. Nykyinen käyttöliittymä ei tue sellaisen konfiguraatiotiedoston luomista tai muokkaamista, jossa on käytetty monimutkaisia taulukkohierarkioita. Tämä ominaisuus on yksi asioista, joita on järjestelmän helppokäyttöisyyden takaamiseksi pakko kehittää tulevaisuudessa. Merxin muussa webselaimia hyödyntävässä käyttöliittymäkehityksessä käytetään nykyisin ExtJS-kirjastoon perustuvaa uutta käyttöliittymämoottoria, joten todennäköisesti jossain vaiheessa myös Web Services -käyttöliittymä tullaan kirjoittamaan kokonaan uudelleen samaa moottoria käyttäen. On todennäköisesti hyödyllistä samalla tarkistaa konfiguraatiotiedostojen nykyistä muotoa paremmaksi. Nykyiset konfiguraatiotiedostot ovat pseudo-PHP:tä eli suoraan PHP-muuttujiin tallennettua tietoa; tämän luettavuutta tekstimuotoisena voitaisiin parantaa huomattavasti vaihtamalla tiedostojen formaatti esimerkiksi XML:ksi tai käyttämällä avuksi Zend Frameworkin tarjoamia INI-tiedostojen käsittelyominaisuuksia.

Yksi NuSOAPin puutteista on vaillinainen tuki käytettäessä WSDL:ssä ja sanomissa useampia eri nimiavaruuksia (namespace) parametreille. Tälle ominaisuudelle tiedetään olevan tarvetta, joten se pitää lisätä rajapintaan tulevaisuudessa. Myös WSDL-tiedoston ja PHP-luokan versiointi on yksi tarpeellinen rajapinnan ominaisuus. Kun eri asiakkailla on samoja palveluita, voi joillakin asiakkailla olla tarpeita muuttaa palvelua vain hieman, esimerkiksi lisäämällä yksi parametri. Jotta tämä voidaan toteuttaa helposti ja tehokkaasti, pitäisi olla mahdollista lisätä kyseinen parametri konfiguraatiotiedostoon, ja tätä kautta rajapintaan, rikkomatta muilla asiakkailla toiminnassa olevaa versiota kyseisestä palvelusta. Tämä mahdollistetaan versioinnilla, jossa yhtenä ? palvelulle tulevista parametreista kerrotaan, mitä versiota kyseisestä palvelusta halutaan käyttää. Jos parametria ei tule, käytetään perusversiota, muuten valitaan konfiguraatiotiedostosta parametrina tulleen version mukaiset asetukset.

Yksi haaste rajapinnalle on myös siinä, että vuoden 2011 loppuun mennessä ilmestyy uusi versio Zend Serveristä ja mikä tärkeintä, sen mukana tulee IBM:n kehittämä uusi i5 toolkit, joka toimii periaatteiltaan samoin kuin vanha AURA:n toolkit, mutta vaatinee jonkin verran muutoksia rajapintamoottorin toimintaan. Tämän lisäksi yksi tarvittava lisäominaisuus eli jonkin aikaa sitten NuSOAPiin tullut liitetiedostojen lähettäminen Web Service -sanomien mukana, pitää myös toteuttaa rajapintaan lähitulevaisuudessa.

Edellä on mainittu tärkeimpiä rajapinnasta puuttuvia ominaisuuksia, mutta kuten sanottu, näitä tarpeita tulee aina uusien projektien mukana lisää. Solteq i Web Services -rajapinta siis elää ajan mukana ja kehittyy tarvittaessa, kuten monet muutkin sovellukset.

LÄHTEET

Anderson M., Kosturjak V., Mullen-Schultz G. 2007. PHP: Zend for i5/OS [pdf-dokumentti]. Luettu 25.10.2011.
http://www.redbooks.ibm.com/redbooks/pdfs/sg247327.pdf

Brady P. 2009. Zend Framework: Surviving The Deep End. [www-sivu]. Luettu 22.11.2011.
http://survivethedeepend.com/zendframeworkbook/en/1.0

Butek R. 2003. Which style of WSDL should I use? [www-sivu]. Luettu 20.10.2011.
http://www.ibm.com/developerworks/webservices/library/ws-whichwsdl/

Dietrich A., Nichol S. 2004-2011. NuSOAP – SOAP Toolkit for PHP. [www-sivu]. Luettu 22.11.2011.
http://sourceforge.net/projects/nusoap/

Nichol S. 2004. Introduction to NuSOAP. [www-sivu]. Luettu 22.11.2011.
http://www.scottnichol.com/soap/nusoapintro.html

Pavlak M. 2011. Zend DBi to Rescue. [www-sivu]. Luettu 22.11.2011.
http://mikepavlak.blogspot.com/2011/09/zend-dbi-to-rescue.html

Programmer's Reference Guide: Zend_CodeGenerator. 2011. Zend Technologies Ltd. [www-sivu]. Luettu 22.11.2011.
http://framework.zend.com/manual/en/zend.codegenerator.introduction.html

Programmer's Reference Guide: Zend_Soap_Autodiscovery. 2011. Zend Technologies Ltd. [www-sivu]. Luettu 22.11.2011.
http://framework.zend.com/manual/en/zend.soap.autodiscovery.html

Richards R. 2006. Pro PHP XML and Web Services. Berkeley: Apress.

Solteq Merx -tuotekortti. 2010. Solteq Oyj. [pdf-dokumentti]. Luettu 22.11.2011.

Solteq Oyj. 2011. [www-sivu]. Luettu 22.11.2011.
http://www.solteq.com/

Solteq Oyj -vuosikertomus 2010. [pdf-dokumentti]. Luettu 22.11.2011.

Seiden A. Zend Server and Zend Framework on IBM i. 2010 [pdf-dokumentti]. Luettu 22.11.2011.
http://www.alanseiden.com/presentation%20slides/Zend-Server-and-Zend%20Framework-for-IBM%20i.pdf

Zend Framework. 2011. Zend Technologies Ltd. [www-sivu]. Luettu 22.11.2011.
http://framework.zend.com/

Zend Server – PHP Server for IBM i. 2011. Zend Technologies Ltd. [www-sivu]. Luettu 22.11.2011.
http://www.zend.com/en/products/server/zend-server-ibm-i

LIITTEET

Liite 1: Aiheeseen liittyviä linkkejä, joita ei ole käytetty suoraan lähteenä tekstissä.
Liite 2: Esimerkki Web Service -rajapinnalla ajonaikaisesti generoidusta WSDL-dokumentista.
Liite 3: Esimerkki Web Service -rajapinnalla ajonaikaisesti generoidusta PHP-luokkatiedostosta.
Liite 4: Esimerkki rajapinnan kirjoittamasta lokimerkinnästä, jossa näkyy vastaanotettu kyselysanoma GetPrice-palvelulle.
Liite 5: Esimerkki rajapinnan kirjoittamasta lokimerkinnästä, jossa näkyy GetPrice-palvelun lähettämä vastaussanoma liitteessä 4 esitettyyn kyselyyn.

Liite 1: Aiheeseen liittyviä linkkejä, joita ei ole käytetty suoraan lähteenä tekstissä.

Seuraavana lisää aiheeseen läheisesti liittyviä www-linkkejä.

Työn toimeksiantaja Solteq Oyj

http://www.solteq.com/

IBM

http://www.ibm.com/

Zend Technologies Ltd.

http://www.zend.com/

IBM i

www.ibm.com/systems/i/

Zend Server for IBM i

http://www.zend.com/en/products/server/zend-server-ibm-i

PHP

http://www.php.net/

Zend Framework

http://framework.zend.com/

NuSOAP – SOAP Toolkit for PHP
http://sourceforge.net/projects/nusoap/

(jatkuu)

Työssä käytetyt sovellukset ovat ladattavissa seuraavista www-osoitteista.

Zend Server
http://www.zend.com/en/products/server/downloads

Zend Framework
http://framework.zend.com/download/latest

NuSOAP
http://sourceforge.net/projects/nusoap/files/nusoap/

SoapUI
http://sourceforge.net/projects/soapui/files/

WCFStorm
http://www.wcfstorm.com/wcf/wcfstorm-lite.aspx

Työssä käytettyjen sovellusten manuaalit löytyvät seuraavista osoitteista.

Apache HTTP Server
http://httpd.apache.org/docs/

PHP
http://www.php.net/manual/en/

Zend Framework
http://framework.zend.com/manual/en/

PHP Toolkit for i5/OS
http://files.zend.com/help/Zend-Core-i5/i5_php_api_toolkit.htm

NuSOAP
http://sourceforge.net/projects/nusoap/files/nusoap-docs/

Seuraavassa on muita hyödyllisiä web-sivustoja, joista löytyy materiaaleja työssä käsitellyistä aiheista.

http://www.alanseiden.com/articles-and-publications/
http://www.alanseiden.com/presentations/
http://mikepavlak.blogspot.com/
http://forums.zend.com/
http://sourceforge.net/projects/nusoap/forums
http://php.net/manual/en/book.soap.php
http://www.zend.com/en/resources/webinars/i5-os
http://www.wcfstorm.com/
http://www.soapui.org/
http://www.easycom-aura.com/en/eac_php.asp
http://www.ibm.com/systems/i/software/db2/index.html
http://www.redbooks.ibm.com/redbooks/pdfs/sg246303.pdf

Liite 2. Esimerkki Web Service rajapinnalla ajonaikaisesti konfiguraatiotiedoston perusteella generoidusta GetPrice-palvelun WSDL-dokumentista

```
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:tns="http://www.solteq.fi/GetPrice"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap-enc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" name="GetPrice"
targetNamespace="http://www.solteq.fi/GetPrice">
    <types>
        <xsd:schema elementFormDefault="qualified" targetNamespace="http://www.solteq.fi/GetPrice">
            <xsd:complexType name="ProdTable">
                <xsd:all>
                    <xsd:element name="ProductCode" type="xsd:string"/>
                    <xsd:element name="Supplier" type="xsd:string"/>
                    <xsd:element name="Quantity" type="xsd:float"/>
                    <xsd:element name="QtyUnit" type="xsd:string"/>
                </xsd:all>
            </xsd:complexType>
            <xsd:complexType name="ArrayOfProdTable">
                <xsd:sequence>
                    <xsd:element name="ProdTable" type="tns:ProdTable" minOccurs="1"
maxOccurs="200"/>
                </xsd:sequence>
            </xsd:complexType>
            <xsd:element name="getPrice">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="Caller" type="xsd:string"/>
                        <xsd:element name="ActionCode" type="xsd:string"/>
                        <xsd:element name="Organisation" type="xsd:string"/>
                        <xsd:element name="Warehouse" type="xsd:string"/>
                        <xsd:element name="Customer" type="xsd:string"/>
                        <xsd:element name="Language" type="xsd:string"/>
                        <xsd:element name="Seller" type="xsd:string"/>
                        <xsd:element name="RowCount" type="xsd:int"/>
                        <xsd:element name="ProdTableDs" type="tns:ArrayOfProdTable"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
            <xsd:complexType name="ErrorTable">
                <xsd:all>
                    <xsd:element name="ErrorID" type="xsd:string"/>
                    <xsd:element name="ErrorMsg" type="xsd:string"/>
                    <xsd:element name="ErrorField" type="xsd:string"/>
                    <xsd:element name="ErrorSeverity" type="xsd:int"/>
                    <xsd:element name="ErrorRow" type="xsd:int"/>
                </xsd:all>
            </xsd:complexType>
            <xsd:complexType name="ArrayOfErrorTable">
                <xsd:sequence>
                    <xsd:element name="ErrorTable" type="tns:ErrorTable" minOccurs="1"
maxOccurs="200"/>
                </xsd:sequence>
            </xsd:complexType>
            <xsd:complexType name="PriceTable">
                <xsd:all>
                    <xsd:element name="PriceType" type="xsd:string"/>
                    <xsd:element name="PriceForSale" type="xsd:float"/>
                    <xsd:element name="PriceWithoutTax" type="xsd:float"/>
                    <xsd:element name="Discount1" type="xsd:float"/>
                    <xsd:element name="Discount2" type="xsd:float"/>
                    <xsd:element name="Discount3" type="xsd:float"/>
                    <xsd:element name="VAT" type="xsd:float"/>
                    <xsd:element name="PriceProcessing" type="xsd:string"/>
                    <xsd:element name="PriceQtyLevel" type="xsd:float"/>
                    <xsd:element name="QtyUnit" type="xsd:string"/>
                    <xsd:element name="Warehouse" type="xsd:string"/>
                </xsd:all>
            </xsd:complexType>
            <xsd:complexType name="ArrayOfPriceTable">
                <xsd:sequence>
                    <xsd:element name="PriceTable" type="tns:PriceTable" minOccurs="1"
maxOccurs="5"/>
                </xsd:sequence>
            </xsd:complexType>
            <xsd:complexType name="ArticleTable">
                <xsd:all>
                    <xsd:element name="ErrorCode" type="xsd:string"/>
                    <xsd:element name="NoPrice" type="xsd:string"/>
                    <xsd:element name="ProductCode" type="xsd:string"/>
                    <xsd:element name="AlternateProductCode" type="xsd:string"/>
                    <xsd:element name="Supplier" type="xsd:string"/>
                    <xsd:element name="ProductStatus" type="xsd:string"/>
                    <xsd:element name="Balance" type="xsd:float"/>
                    <xsd:element name="Description" type="xsd:string"/>
```

(jatkuu)

LIITE 2: 2 (2)

```
                    <xsd:element name="DeliveryReadiness" type="xsd:string"/>
                    <xsd:element name="PriceTableDs" type="tns:ArrayOfPriceTable" minOccurs="0"
maxOccurs="1"/>
                </xsd:all>
            </xsd:complexType>
            <xsd:complexType name="ArticleTable">
                <xsd:sequence>
                    <xsd:element name="ArticleTable" type="tns:ArticleTable" minOccurs="1"
maxOccurs="200"/>
                </xsd:sequence>
            </xsd:complexType>
            <xsd:complexType name="getPriceReturn">
                <xsd:all>
                    <xsd:element name="ErrCode" type="xsd:string"/>
                    <xsd:element name="ErrorCounter" type="xsd:int"/>
                    <xsd:element name="ErrorTableDs" type="tns:ArrayOfErrorTable" minOccurs="0"
maxOccurs="1"/>
                    <xsd:element name="Currency" type="xsd:string"/>
                    <xsd:element name="RowCount" type="xsd:int"/>
                    <xsd:element name="ArticleTableDs" type="tns:ArrayOfArticleTable"/>
                </xsd:all>
            </xsd:complexType>
            <xsd:element name="getPriceResponse">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="getPriceResult" type="tns:getPriceReturn"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
        </xsd:schema>
    </types>
    <portType name="GetPricePort">
        <operation name="getPrice">
            <documentation>Haetaan tuotteen hintatiedot</documentation>
            <input message="tns:getPriceIn"/>
            <output message="tns:getPriceOut"/>
        </operation>
    </portType>
    <binding name="GetPriceBinding" type="tns:GetPricePort">
        <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
        <operation name="getPrice">
            <soap:operation
soapAction="http://tth400:8001/sq/ws/soapserver/serve/app/merx/service/GetPrice#getPrice"/>
            <input>
                <soap:body use="literal" namespace="http://www.solteq.fi/GetPrice"/>
            </input>
            <output>
                <soap:body use="literal" namespace="http://www.solteq.fi/GetPrice"/>
            </output>
        </operation>
    </binding>
    <service name="GetPriceService">
        <port name="GetPricePort" binding="tns:GetPriceBinding">
            <soap:address
location="http://tth400:8001/sq/ws/soapserver/serve/app/merx/service/GetPrice"/>
        </port>
    </service>
    <message name="getPriceIn">
        <part name="parameters" element="tns:getPrice"/>
    </message>
    <message name="getPriceOut">
        <part name="parameters" element="tns:getPriceResponse"/>
    </message>
</definitions>
```

Liite 3. Esimerkki Web Service -rajapinnalla ajonaikaisesti konfiguraatiotiedoston perusteella generoidusta GetPrice-palvelun PHP-luokkatiedostosta

```php
<?php

class ProdTable
{

    /**
     * @var string $ProductCode
     */
    public $ProductCode = null;

    /**
     * @var string $Supplier
     */
    public $Supplier = null;

    /**
     * @var float $Quantity
     */
    public $Quantity = null;

    /**
     * @var string $QtyUnit
     */
    public $QtyUnit = null;

    /**
     *
     */
    public function __construct($inArray = array ())
    {
        $this->ProductCode = mb_convert_encoding ( $inArray['PI_Pr_Tuno'], 'UTF-8' );
        $this->Supplier = mb_convert_encoding ( $inArray['PI_Pr_Toim'], 'UTF-8' );
        $this->Quantity = mb_convert_encoding ( $inArray['PI_Pr_Mra'], 'UTF-8' );
        $this->QtyUnit = mb_convert_encoding ( $inArray['PI_Pr_Myks'], 'UTF-8' );
    }


}
class ErrorTable
{

    /**
     * @var string $ErrorID
     */
    public $ErrorID = null;

    /**
     * @var string $ErrorMsg
     */
    public $ErrorMsg = null;

    /**
     * @var string $ErrorField
     */
    public $ErrorField = null;

    /**
     * @var integer $ErrorSeverity
     */
    public $ErrorSeverity = null;

    /**
     * @var integer $ErrorRow
     */
    public $ErrorRow = null;

    /**
     *
     */
    public function __construct($inArray = array ())
    {
        $this->ErrorID = mb_convert_encoding ( $inArray['Err_Id'], 'UTF-8' );
        $this->ErrorMsg = mb_convert_encoding ( $inArray['Err_Msg'], 'UTF-8' );
        $this->ErrorField = mb_convert_encoding ( $inArray['Err_Field'], 'UTF-8' );
        $this->ErrorSeverity = mb_convert_encoding ( $inArray['Err_Sev'], 'UTF-8' );
        $this->ErrorRow = mb_convert_encoding ( $inArray['Err_Row'], 'UTF-8' );
    }


}
class PriceTable
{
```

(jatkuu)

```
    /**
     * @var string $PriceType
     */
    public $PriceType = null;

    /**
     * @var float $PriceForSale
     */
    public $PriceForSale = null;

    /**
     * @var float $PriceWithoutTax
     */
    public $PriceWithoutTax = null;

    /**
     * @var float $Discount1
     */
    public $Discount1 = null;

    /**
     * @var float $Discount2
     */
    public $Discount2 = null;

    /**
     * @var float $Discount3
     */
    public $Discount3 = null;

    /**
     * @var float $VAT
     */
    public $VAT = null;

    /**
     * @var string $PriceProcessing
     */
    public $PriceProcessing = null;

    /**
     * @var float $PriceQtyLevel
     */
    public $PriceQtyLevel = null;

    /**
     * @var string $QtyUnit
     */
    public $QtyUnit = null;

    /**
     * @var string $Warehouse
     */
    public $Warehouse = null;

    /**
     *
     */
    public function __construct($inArray = array ())
    {
        $this->PriceType = mb_convert_encoding ( $inArray['PO_Pr_Hity'], 'UTF-8' );
        $this->PriceForSale = mb_convert_encoding ( $inArray['PO_Pr_Myhi'], 'UTF-8' );
        $this->PriceWithoutTax = mb_convert_encoding ( $inArray['PO_Pr_Vthi'], 'UTF-8' );
        $this->Discount1 = mb_convert_encoding ( $inArray['PO_Pr_Ale1'], 'UTF-8' );
        $this->Discount2 = mb_convert_encoding ( $inArray['PO_Pr_Ale2'], 'UTF-8' );
        $this->Discount3 = mb_convert_encoding ( $inArray['PO_Pr_Ale3'], 'UTF-8' );
        $this->VAT = mb_convert_encoding ( $inArray['PO_Pr_Alvp'], 'UTF-8' );
        $this->PriceProcessing = mb_convert_encoding ( $inArray['PO_Pr_Hikt'], 'UTF-8' );
        $this->PriceQtyLevel = mb_convert_encoding ( $inArray['PO_Pr_Mhalr'], 'UTF-8' );
        $this->QtyUnit = mb_convert_encoding ( $inArray['PO_Pr_Myks'], 'UTF-8' );
        $this->Warehouse = mb_convert_encoding ( $inArray['PO_Pr_Vano'], 'UTF-8' );
    }


}
class ArticleTable
{

    /**
     * @var string $ErrorCode
     */
    public $ErrorCode = null;

    /**
     * @var string $NoPrice
     */
    public $NoPrice = null;
```

```
    /**
     * @var string $ProductCode
     */
    public $ProductCode = null;

    /**
     * @var string $AlternateProductCode
     */
    public $AlternateProductCode = null;

    /**
     * @var string $Supplier
     */
    public $Supplier = null;

    /**
     * @var string $ProductStatus
     */
    public $ProductStatus = null;

    /**
     * @var float $Balance
     */
    public $Balance = null;

    /**
     * @var string $Description
     */
    public $Description = null;

    /**
     * @var string $DeliveryReadiness
     */
    public $DeliveryReadiness = null;

    /**
     * @var PriceTable[1,5] $PriceTableDs[0,1]
     */
    public $PriceTableDs = null;

    /**
     *
     */
    public function __construct($inArray = array ())
    {
        $this->ErrorCode = mb_convert_encoding ( $inArray['PO_Pr_Error'], 'UTF-8' );
        $this->NoPrice = mb_convert_encoding ( $inArray['PO_Pr_NoPrice'], 'UTF-8' );
        $this->ProductCode = mb_convert_encoding ( $inArray['PO_Pr_Tuno'], 'UTF-8' );
        $this->AlternateProductCode = mb_convert_encoding ( $inArray['PO_Pr_Vetn'], 'UTF-8' );
        $this->Supplier = mb_convert_encoding ( $inArray['PO_Pr_Toim'], 'UTF-8' );
        $this->ProductStatus = mb_convert_encoding ( $inArray['PO_Pr_TuSta'], 'UTF-8' );
        $this->Balance = mb_convert_encoding ( $inArray['PO_Pr_Vapaa'], 'UTF-8' );
        $this->Description = mb_convert_encoding ( $inArray['PO_Pr_TuDesc'], 'UTF-8' );
        $this->DeliveryReadiness = mb_convert_encoding ( $inArray['PO_Pr_Tvalm'], 'UTF-8' );
        $empty['PO_PriceTable'] = 0;
        $empty['PO_PriceTable_CELLS'] = 0;
        $dsar = null;
        if (!empty($inArray['PO_PriceTable']) and is_array($inArray['PO_PriceTable'])) {
            foreach ($inArray['PO_PriceTable'] as $v) {
                if (!empty($v) and is_array($v)) {
                    foreach ($v as $data) {
                        if (!empty($data)) {
                            $empty['PO_PriceTable_CELLS'] = 0;
                        } else {
                            $empty['PO_PriceTable_CELLS'] += 1;
                        }
                    }
                    if ( $empty['PO_PriceTable_CELLS'] >= count($v)) {
                        $empty['PO_PriceTable'] += 1;
                        $empty['PO_PriceTable_CELLS'] = 0;
                    } else {
                        $ds = new PriceTable( $v );
                        $dsar[] = $ds;
                    }
                    if ($empty['PO_PriceTable'] > 2) {
                        break;
                    }
                }
            }
        }
            $this->PriceTableDs = $dsar;
    }


}
class getPriceReturn
{
```

```
/**
 * @var string $ErrCode
 */
public $ErrCode = null;

/**
 * @var integer $ErrorCounter
 */
public $ErrorCounter = null;

/**
 * @var ErrorTable[1,200] $ErrorTableDs[0,1]
 */
public $ErrorTableDs = null;

/**
 * @var string $Currency
 */
public $Currency = null;

/**
 * @var integer $RowCount
 */
public $RowCount = null;

/**
 * @var ArticleTable[1,200] $ArticleTableDs
 */
public $ArticleTableDs = null;

/**
 *
 */
public function __construct($inArray = array ())
{
    $this->ErrCode = mb_convert_encoding ( $inArray['PO_OutCode'], 'UTF-8' );
    $this->ErrorCounter = mb_convert_encoding ( $inArray['PO_ErrNbr'], 'UTF-8' );
    $empty['PO_Err'] = 0;
    $empty['PO_Err_CELLS'] = 0;
    $dsar = null;
    if (!empty($inArray['PO_Err']) and is_array($inArray['PO_Err'])) {
        foreach ($inArray['PO_Err'] as $v) {
            if (!empty($v) and is_array($v)) {
                foreach ($v as $data) {
                    if (!empty($data)) {
                        $empty['PO_Err_CELLS'] = 0;
                    } else {
                        $empty['PO_Err_CELLS'] += 1;
                    }
                }
                if ( $empty['PO_Err_CELLS'] >= count($v)) {
                    $empty['PO_Err'] += 1;
                    $empty['PO_Err_CELLS'] = 0;
                } else {
                    $ds = new ErrorTable( $v );
                    $dsar[] = $ds;
                }
                if ($empty['PO_Err'] > 2) {
                    break;
                }
            }
        }
    }
        $this->ErrorTableDs = $dsar;
    $this->Currency = mb_convert_encoding ( $inArray['PO_Valk'], 'UTF-8' );
    $this->RowCount = mb_convert_encoding ( $inArray['PO_Rows'], 'UTF-8' );
    $empty['PO_ProdTable'] = 0;
    $empty['PO_ProdTable_CELLS'] = 0;
    $dsar = array();
    if (!empty($inArray['PO_ProdTable']) and is_array($inArray['PO_ProdTable'])) {
        foreach ($inArray['PO_ProdTable'] as $v) {
            if (!empty($v) and is_array($v)) {
                foreach ($v as $data) {
                    if (!empty($data)) {
                        $empty['PO_ProdTable_CELLS'] = 0;
                    } else {
                        $empty['PO_ProdTable_CELLS'] += 1;
                    }
                }
                if ( $empty['PO_ProdTable_CELLS'] >= count($v)) {
                    $empty['PO_ProdTable'] += 1;
                    $empty['PO_ProdTable_CELLS'] = 0;
                } else {
                    $ds = new ArticleTable( $v );
                    $dsar[] = $ds;
                }
                if ($empty['PO_ProdTable'] > 2) {
```

```
                        break;
                    }
                }
            }
        }
            $this->ArticleTableDs = $dsar;
    }


}


/**
 * Price request
 *
 *
 *
 * @version 1313573525
 */
class GetPrice
{

    private $rpgInParms = array(
        'PI_Caller' => '',
        'PI_Action' => '',
        'PI_Pyks' => '',
        'PI_Vano' => '',
        'PI_Asno' => '',
        'PI_Kiekdi' => '',
        'PI_Myyja' => '',
        'PI_Rows' => '',
        'PI_ProdTable' => array(),
        'PO_OutCode' => '',
        'PO_ErrNbr' => '',
        'PO_Err' => array(),
        'PO_Valk' => '',
        'PO_Rows' => '',
        'PO_ProdTable' => array()
        );

    private $rpgAllParms = array(
        array(
            'name' => 'PI_Caller',
            'io' => 3,
            'type' => 0,
            'length' => '18'
            ),
        array(
            'name' => 'PI_Action',
            'io' => 3,
            'type' => 0,
            'length' => '1'
            ),
        array(
            'name' => 'PI_Pyks',
            'io' => 3,
            'type' => 0,
            'length' => '2'
            ),
        array(
            'name' => 'PI_Vano',
            'io' => 3,
            'type' => 0,
            'length' => '15'
            ),
        array(
            'name' => 'PI_Asno',
            'io' => 3,
            'type' => 0,
            'length' => '15'
            ),
        array(
            'name' => 'PI_Kiekdi',
            'io' => 3,
            'type' => 0,
            'length' => '3'
            ),
        array(
            'name' => 'PI_Myyja',
            'io' => 3,
            'type' => 0,
            'length' => '15'
            ),
        array(
            'name' => 'PI_Rows',
            'io' => 3,
            'type' => 7,
```

```
            'length' => '5.0'
            ),
        array(
            'DSName' => 'PI_ProdTable',
            'count' => 200,
            'DSParm' => array(
                array(
                    'name' => 'PI_Pr_Tuno',
                    'io' => 3,
                    'type' => 0,
                    'length' => '20'
                    ),
                array(
                    'name' => 'PI_Pr_Toim',
                    'io' => 3,
                    'type' => 0,
                    'length' => '15'
                    ),
                array(
                    'name' => 'PI_Pr_Mra',
                    'io' => 3,
                    'type' => 7,
                    'length' => '9.2'
                    ),
                array(
                    'name' => 'PI_Pr_Myks',
                    'io' => 3,
                    'type' => 0,
                    'length' => '3'
                    )
                )
            ),
        array(
            'name' => 'PO_OutCode',
            'io' => 3,
            'type' => 0,
            'length' => '1'
            ),
        array(
            'name' => 'PO_ErrNbr',
            'io' => 3,
            'type' => 7,
            'length' => '3.0'
            ),
        array(
            'DSName' => 'PO_Err',
            'count' => 200,
            'DSParm' => array(
                array(
                    'name' => 'Err_Id',
                    'io' => 3,
                    'type' => 0,
                    'length' => '7'
                    ),
                array(
                    'name' => 'Err_Msg',
                    'io' => 3,
                    'type' => 0,
                    'length' => '132'
                    ),
                array(
                    'name' => 'Err_Field',
                    'io' => 3,
                    'type' => 0,
                    'length' => '30'
                    ),
                array(
                    'name' => 'Err_Sev',
                    'io' => 3,
                    'type' => 7,
                    'length' => '1.0'
                    ),
                array(
                    'name' => 'Err_Row',
                    'io' => 3,
                    'type' => 7,
                    'length' => '5.0'
                    )
                )
            ),
        array(
            'name' => 'PO_Valk',
            'io' => 3,
            'type' => 0,
            'length' => '3'
            ),
```

```
array(
    'name' => 'PO_Rows',
    'io' => 3,
    'type' => 7,
    'length' => '5.0'
    ),
array(
    'DSName' => 'PO_ProdTable',
    'count' => 200,
    'DSParm' => array(
        array(
            'name' => 'PO_Pr_Error',
            'io' => 3,
            'type' => 0,
            'length' => '1'
            ),
        array(
            'name' => 'PO_Pr_NoPrice',
            'io' => 3,
            'type' => 0,
            'length' => '1'
            ),
        array(
            'name' => 'PO_Pr_Tuno',
            'io' => 3,
            'type' => 0,
            'length' => '20'
            ),
        array(
            'name' => 'PO_Pr_Vetn',
            'io' => 3,
            'type' => 0,
            'length' => '20'
            ),
        array(
            'name' => 'PO_Pr_Toim',
            'io' => 3,
            'type' => 0,
            'length' => '15'
            ),
        array(
            'name' => 'PO_Pr_TuSta',
            'io' => 3,
            'type' => 0,
            'length' => '3'
            ),
        array(
            'name' => 'PO_Pr_Vapaa',
            'io' => 3,
            'type' => 7,
            'length' => '9.2'
            ),
        array(
            'name' => 'PO_Pr_TuDesc',
            'io' => 3,
            'type' => 0,
            'length' => '40'
            ),
        array(
            'name' => 'PO_Pr_Tvalm',
            'io' => 3,
            'type' => 0,
            'length' => '3'
            ),
        array(
            'DSName' => 'PO_PriceTable',
            'count' => 5,
            'DSParm' => array(
                array(
                    'name' => 'PO_Pr_Hity',
                    'io' => 3,
                    'type' => 0,
                    'length' => '3'
                    ),
                array(
                    'name' => 'PO_Pr_Myhi',
                    'io' => 3,
                    'type' => 0,
                    'length' => '13'
                    ),
                array(
                    'name' => 'PO_Pr_Vthi',
                    'io' => 3,
                    'type' => 0,
                    'length' => '13'
                    ),
```

```
                        array(
                            'name' => 'PO_Pr_Ale1',
                            'io' => 3,
                            'type' => 0,
                            'length' => '7'
                            ),
                        array(
                            'name' => 'PO_Pr_Ale2',
                            'io' => 3,
                            'type' => 0,
                            'length' => '7'
                            ),
                        array(
                            'name' => 'PO_Pr_Ale3',
                            'io' => 3,
                            'type' => 0,
                            'length' => '7'
                            ),
                        array(
                            'name' => 'PO_Pr_Alvp',
                            'io' => 3,
                            'type' => 0,
                            'length' => '7'
                            ),
                        array(
                            'name' => 'PO_Pr_Hikt',
                            'io' => 3,
                            'type' => 0,
                            'length' => '3'
                            ),
                        array(
                            'name' => 'PO_Pr_Mhalr',
                            'io' => 3,
                            'type' => 0,
                            'length' => '11'
                            ),
                        array(
                            'name' => 'PO_Pr_Myks',
                            'io' => 3,
                            'type' => 0,
                            'length' => '3'
                            ),
                        array(
                            'name' => 'PO_Pr_Vano',
                            'io' => 3,
                            'type' => 0,
                            'length' => '15'
                            )
                        )
                    )
                )
            )
        );

    private $db = null;

    private $rpgProgram = 'RMETUH';

    private $rpgLibrary = '*LIBL';

    private $TODAY = null;

    public function __construct()
    {
        $this->db = new Class_Database();
        $this->TODAY = date('Ymd');
    }

    /**
     * Haetaan tuotteen hintatiedot
     *
     * @param string $Caller
     * @param string $ActionCode
     * @param string $Organisation
     * @param string $Warehouse
     * @param string $Customer
     * @param string $Language
     * @param string $Seller
     * @param integer $RowCount
     * @param ProdTable[1,200] $ProdTableDs
     * @return getPriceReturn
     */
    public function getPrice($Caller = '', $ActionCode = '', $Organisation = '', $Warehouse = '',
$Customer = '', $Language = '', $Seller = '', $RowCount = '', $ProdTableDs)
    {
        if ( isset( $Caller ) and $Caller != '') {
```

```
            $this->rpgInParms['PI_Caller'] = mb_convert_encoding( $Caller, 'ISO-8859-1',
mb_detect_encoding($Caller, 'UTF-8, ISO-8859-1, ISO-8859-15, ASCII', true) );
        }
        if ( isset( $ActionCode ) and $ActionCode != '') {
            $this->rpgInParms['PI_Action'] = mb_convert_encoding( $ActionCode, 'ISO-8859-1',
mb_detect_encoding($ActionCode, 'UTF-8, ISO-8859-1, ISO-8859-15, ASCII', true) );
        }
        if ( isset( $Organisation ) and $Organisation != '') {
            $this->rpgInParms['PI_Pyks'] = mb_convert_encoding( $Organisation, 'ISO-8859-1',
mb_detect_encoding($Organisation, 'UTF-8, ISO-8859-1, ISO-8859-15, ASCII', true) );
        }
        if ( isset( $Warehouse ) and $Warehouse != '') {
            $this->rpgInParms['PI_Vano'] = mb_convert_encoding( $Warehouse, 'ISO-8859-1',
mb_detect_encoding($Warehouse, 'UTF-8, ISO-8859-1, ISO-8859-15, ASCII', true) );
        }
        if ( isset( $Customer ) and $Customer != '') {
            $this->rpgInParms['PI_Asno'] = mb_convert_encoding( $Customer, 'ISO-8859-1',
mb_detect_encoding($Customer, 'UTF-8, ISO-8859-1, ISO-8859-15, ASCII', true) );
        }
        if ( isset( $Language ) and $Language != '') {
            $this->rpgInParms['PI_Kiekdi'] = mb_convert_encoding( $Language, 'ISO-8859-1',
mb_detect_encoding($Language, 'UTF-8, ISO-8859-1, ISO-8859-15, ASCII', true) );
        }
        if ( isset( $Seller ) and $Seller != '') {
            $this->rpgInParms['PI_Myyja'] = mb_convert_encoding( $Seller, 'ISO-8859-1',
mb_detect_encoding($Seller, 'UTF-8, ISO-8859-1, ISO-8859-15, ASCII', true) );
        }
        if ( isset( $RowCount ) and $RowCount != '') {
            $this->rpgInParms['PI_Rows'] = mb_convert_encoding( $RowCount, 'ISO-8859-1',
mb_detect_encoding($RowCount, 'UTF-8, ISO-8859-1, ISO-8859-15, ASCII', true) );
        }
        if ( isset( $ProdTableDs ) and is_array( $ProdTableDs ) ) {
            if ( isset( $ProdTableDs['ProdTable'][0] ) and is_array ( $ProdTableDs['ProdTable'][0] )
) {
                $ProdTableDs = $ProdTableDs['ProdTable'];
            }
            foreach ( $ProdTableDs as $val ) {
                $ProductCode = mb_convert_encoding( $val['ProductCode'], 'ISO-8859-1',
mb_detect_encoding($val['ProductCode'], 'UTF-8, ISO-8859-1, ISO-8859-15, ASCII', true) );
                $Supplier = mb_convert_encoding( $val['Supplier'], 'ISO-8859-1',
mb_detect_encoding($val['Supplier'], 'UTF-8, ISO-8859-1, ISO-8859-15, ASCII', true) );
                $Quantity = mb_convert_encoding( $val['Quantity'], 'ISO-8859-1',
mb_detect_encoding($val['Quantity'], 'UTF-8, ISO-8859-1, ISO-8859-15, ASCII', true) );
                $QtyUnit = mb_convert_encoding( $val['QtyUnit'], 'ISO-8859-1',
mb_detect_encoding($val['QtyUnit'], 'UTF-8, ISO-8859-1, ISO-8859-15, ASCII', true) );
                $ProdTable[] = array (
                    'PI_Pr_Tuno' => $ProductCode,
                    'PI_Pr_Toim' => $Supplier,
                    'PI_Pr_Mra' => $Quantity,
                    'PI_Pr_Myks' => $QtyUnit
                );
            }
            $this->rpgInParms['PI_ProdTable'] = $ProdTable;
        }

        foreach ( $this->rpgInParms as $key => $value ) {
            if ( $value == '$TODAY' ) {
                $this->rpgInParms[$key] = $this->TODAY;
            }
        }

        $ret = null;

        $this->db->i5pConnect ( );
        $this->db->setLibPgm ( $this->rpgLibrary, $this->rpgProgram );
        $this->db->setPlist ( $this->rpgAllParms );
        $this->db->pgmPrepare( );
        $this->db->createIOParms ( $this->rpgInParms );
        $this->db->pgmCall ( );
        $this->db->pgmClose ( );

        $result = $this->db->getResult();
        $ret = new getPriceReturn( $result );

        return $ret;
    }


}

//End of file
?>
```

Liite 4. Esimerkki rajapinnan kirjoittamasta lokimerkinnästä, jossa näkyy vastaanotettu kyselysanoma GetPrice-palvelulle

```
================================================================
= Request below received: 2011.11.01 12:30:46
================================================================
host: 127.0.0.1:8888
content-type: text/xml;charset=UTF-8
user-agent: Jakarta Commons-HttpClient/3.1
max-forwards: 10
x-forwarded-for: 172.17.66.58
x-forwarded-host: tth400:8001
x-forwarded-server: tth400.solteq.fi
content-length: 669

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Body>
    <getPrice xmlns="http://www.solteq.fi/GetPrice">
    <Caller>TKTEST</Caller>
    <ActionCode />
    <Organisation />
    <Warehouse />
    <Customer>0</Customer>
    <Language>FI</Language>
    <Seller />
    <RowCount>1</RowCount>
    <ProdTableDs>
        <ProdTable>
            <ProductCode>761</ProductCode>
            <Supplier>
            </Supplier>
            <Quantity>0</Quantity>
            <QtyUnit />
        </ProdTable>
    </ProdTableDs>
</getPrice>
</soap:Body>
</soap:Envelope>
```

Liite 5. Esimerkki rajapinnan kirjoittamasta lokimerkinnästä, jossa näkyy GetPrice-palvelun lähettämä vastaussanoma liitteessä 4 esitettyyn kyselyyn

```
=====================================================================
Response below sent: 2011.11.01 12:30:47 (for request received 2011.11.01 12:30:46) time: 0.17
seconds
=====================================================================
Server: NuSOAP Server v0.9.6dev
X-SOAP-Server: NuSOAP/0.9.6dev (1.137)
Content-Type: text/xml; charset=utf-8
Content-Length: 1566

<?xml version="1.0" encoding="utf-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/">
<SOAP-ENV:Body>
    <getPriceResponse xmlns="http://www.solteq.fi/GetPrice">
    <getPriceResult>
        <ErrCode>0</ErrCode>
        <ErrorCounter>0</ErrorCounter>
        <Currency>EUR</Currency>
        <RowCount>1</RowCount>
        <ArticleTableDs>
            <ArticleTable>
                <ErrorCode>0</ErrorCode>
                <NoPrice>0</NoPrice>
                <ProductCode>761</ProductCode>
                <AlternateProductCode>761</AlternateProductCode>
                <Supplier>0003205</Supplier>
                <ProductStatus>
                </ProductStatus>
                <Balance>11955</Balance>
                <Description>
                </Description>
                <DeliveryReadiness>1</DeliveryReadiness>
                <PriceTableDs>
                    <PriceTable>
                        <PriceType>1</PriceType>
                        <PriceForSale>1.5000</PriceForSale>
                        <PriceWithoutTax>1.3274</PriceWithoutTax>
                        <Discount1>0.00</Discount1>
                        <Discount2>0.00</Discount2>
                        <Discount3>0.00</Discount3>
                        <VAT>0</VAT>
                        <PriceProcessing>C</PriceProcessing>
                        <PriceQtyLevel>01</PriceQtyLevel>
                        <QtyUnit>PCE</QtyUnit>
                        <Warehouse>000</Warehouse>
                    </PriceTable>
                    <PriceTable>
                        <PriceType>1</PriceType>
                        <PriceForSale>1.5000</PriceForSale>
                        <PriceWithoutTax>1.3274</PriceWithoutTax>
                        <Discount1>0.00</Discount1>
                        <Discount2>0.00</Discount2>
                        <Discount3>0.00</Discount3>
                        <VAT>13.00</VAT>
                        <PriceProcessing>V</PriceProcessing>
                        <PriceQtyLevel>01</PriceQtyLevel>
                        <QtyUnit>PCE</QtyUnit>
                        <Warehouse>000</Warehouse>
                    </PriceTable>
                </PriceTableDs>
            </ArticleTable>
        </ArticleTableDs>
    </getPriceResult>
</getPriceResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```