

KÄYTTÖLIITTYMÄMOOTTORIN TOTEUTUS EXT JS - SOVELLUSKEHYKSELLÄ

Ville Seppälä

Opinnäytetyö
Toukokuu 2012
Tietojenkäsittely
Ohjelmistotuotanto

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma
Ohjelmistotuotannon suuntautumisvaihtoehto

SEPPÄLÄ, VILLE:

Käyttöliittymämoodorin toteutus Ext JS -sovelluskehyksellä

Opinnäytetyö 39 sivua, josta liitteitä 5 sivua
Toukokuu 2012

Opinnäytetyönä toteutettiin käyttöliittymämoodori Solteq Oyj:n kehittämään Merx-toiminnanohjausjärjestelmään. Käyttöliittymämoodori on selaimella käytettävä graafinen käyttöliittymä, joka osaa luoda käyttöliittymäruutuja kuvaustiedostojen pohjalta. Sen tavoitteena on olla mahdollisimman muokattava ja helpottaa uusien käyttöliittymäruutujen luomista ilman web-ohjelmointikielten osaamista.

Käyttöliittymämoodorin avulla voidaan luoda uusia ja vaihtoehtoisia tapoja käyttää Merxiä, nykyisen tekstipohjaisen käyttöliittymän lisänä. Käyttöliittymämoodorilla Merxiin saadaan aivan uusia mahdollisuuksia, joilla voidaan parantaa asiakkaiden liiketoimintaa ja helpottaa päivittäistä työtä.

Opinnäytetyön kirjallisessa osuudessa kerrotaan käyttöliittymämoodorin toimintaperiaatteesta, sen ominaisuuksista ja kehityksen aikana tehdyistä teknisistä ratkaisuista. Opinnäytetyössä tarkastellaan miksi käyttöliittymämoodori toteutettiin, miten se toimii ja mitä hyötyä graafisesta käyttöliittymästä on asiakkaalle.

Käyttöliittymämoodorin toteutuksessa onnistuttiin ja sille asetetut tavoitteet saavutettiin. webMerx-käyttöliittymämoodori on tuotantokäytössä useilla Merx-asiakkailta ja tulossa käyttöön vielä useammille asiakkaille. Käyttöliittymämoodori on helpottanut asiakkaiden päivittäistä työtä tarjoamalla uusia ominaisuuksia, kuten tiedon venti- ja tuontiominaisuudet.

ABSTRACT

Tampere University of Applied Sciences
Degree programme in Business Information Systems
Specialization of Software Engineering

SEPPÄLÄ, VILLE:

User interface engine implementation using Ext JS -framework

Bachelor's thesis 39 pages, appendices 5 pages
May 2012

The objective of this thesis work was to develop a web-based user interface engine for Solteq Oyj's Merx-enterprise resource planning software. The user interface engine is a graphical user interface used by a browser that knows how to create user interface screens on the basis of a description file. The purpose of the interface engine is to be as customizable as possible and to make it easy to create new graphical user interface screens without the understanding of web technologies. The user interface engine offers new features and alternative ways to use Solteq Merx.

The purpose of this thesis is to report the user interface engines operating principles, its properties and to find out what are the benefits of a web-based graphical user interface to the customer.

The user interface engine was developed and all the goals were achieved. WebMerx user interface engine is used by many Merx customers and will be implemented for more customers in future. The user interface engine has helped our customer's everyday work by offering new features such as data import and export feature.

Key words: merx, extjs, erp, php, javascript, user interface

SISÄLLYS

1	JOHDANTO.....	8
1.1	Tausta.....	8
1.2	Tavoitteet ja tarkoitus	9
2	TOIMEKSIANTAJAN ESITTELY.....	11
2.1	Solteq Oyj	11
2.2	Solteq Merx.....	11
3	KÄYTETYT TEKNIIKAT JA KIELET.....	13
3.1	PHP	13
3.2	JavaScript.....	13
3.3	XML.....	14
3.4	Ext JS -sovelluskehys	14
3.5	Ext Direct.....	15
3.6	Zend Framework -sovelluskehys	18
3.7	Zend Core for i5/OS -palvelinohjelmitopaketti	19
4	KÄYTTÖLIITTYMÄMOOTTORIN TOIMINTAPERIAATE	20
4.1	Käyttöliittymämoottorin rakenne.....	20
4.2	Käyttöliittymämoottorin asetustasot	23
4.3	Käyttöliittymäruutujen luominen.....	25
4.4	Aineiston tuonti- ja vientiominaisuus	29
5	TULOKSET JA POHDINTA	31
5.1	Tulokset	31
5.2	Tuotantokäyttö	32
5.3	Jatkokehitys	32
	LÄHTEET	34
	LIITTEET	35
	Liite 1. Aiheeseen liittyviä linkkejä ja lisätiedon lähteitä.	35
	Liite 2. Myyntitilausten selailu -käyttöliittymäruudun XML-kuvaustiedosto.	36
	Liite 3. Passiivituotteiden vienti ja tuonti -ominaisuuden sapluuna.....	38

LYHENTEET JA TERMIT

Apache	Apache on useille eri käyttöjärjestelmille saatava avoimen lähdekoodin HTTP-palvelinohjelma.
API	Ohjelmointirajapinta (Application programming interface) on määritelmä, jonka mukaan eri ohjelmat voivat keskustella keskenään tekemällä pyyntöjä ja vaihtamalla tietoja. Esimerkiksi kaikki sovelluskehukset tarjoavat ohjelmointirajapinnan, jonka avulla ohjelmoija voi keskustella sovelluskehysten kanssa.
ERP	ERP eli toiminnanohjausjärjestelmä on yrityksen tietojärjestelmä, joka sisältää esimerkiksi materiaalin, resurssien, huollon, omaisuuden ja projektien hallinnan sekä varastonhallinnan, kirjanpidon ja palkkalaskennan.
Framework	Ohjelmistokehys on ohjelmisto, jonka muodostaa rungon sen päälle rakennettavalle tietokoneohjelmalle. Se on apuväline, jonka tarkoituksena on nopeuttaa ja helpottaa ohjelmistojen rakentamista.
i5 toolkit	Aura Equipmentsin kehittämä PHP-lisäosa, joka mahdollistaa RPG-ohjelmien suoran kutsumisen PHP:stä. i5 toolkit tulee Zend Core for i5/OS – palvelinohjelmistopakettina.
JSON	JSON (JavaScript Object Notation) on tekstipohjainen, selväkielinen ja yksinkertainen tiedonsiirtomuoto. Nimestään huolimatta JSON ei ole JavaScript-riippuvainen vaan sitä voidaan käyttää myös muissa ohjelmointikielissä.
MVC	MVC-arkkitehtuuri on ohjelmistoarkkitehtuurityyli, joka tulee sanoista model-view-control (malli-näkymä-käsittelijä). Sitä käytetään yleisesti ohjelmissa, jotka omaavat graafisen käyttöliittymän.
PHP	(PHP: Hypertext Preprocessor) on komentosarjakieli, jota käytetään yleisesti web-palvelinympäristöissä dynaamisten verkkosivujen luomiseen. PHP-ohjelmakoodia ei esikäännetä, vaan se tulkitaan ajon aikana.

RIA	Rikkaat internet-sovellukset (Rich Internet Applications) ovat internet-sovelluksia, joiden ominaisuudet ja toiminnallisuus muistuttavat työpöytäsovellusta. Sovelluksen latauksen jälkeen suurin osa toiminnallisuudesta tapahtuu selaimessa. Esimerkiksi Googlen omistama YouTube voidaan laskea rikkaaksi internet-sovellukseksi.
RPG	RPG (Report Program Generator) on IBM:n alun perin vuonna 1959 julkaisema korkean tason ohjelmointikieli helpottamaan ja yksinkertaistamaan ohjelmistokehitystä. Vuosien saatossa se on kehittynyt paljon ja on yhä laajalti käytössä. Ohjelmointikielen uusin versio, RPG IV (ILE RPG) julkaistiin 1994 ja se toi mukanaan paljon uusia ominaisuuksia. RPG IV:n uusin versio 7 julkaistiin vuonna 2010.
SASS	SASS (Syntactically Awesome Stylesheets) on Hamilton Catlin suunnittelema ja Nathan Weizenbaumin kehittämä tyylikieli, jonka tarkoituksena on helpottaa CSS tyyli-tiedostojen luomista ja ylläpitämistä.
SDK	SDK (Software development kit) on kokoelma ohjelmistokehitystyökaluja, joiden avulla voidaan luoda ohjelma tietylle laitteisto- tai ohjelmistoalustalle.
Solteq Merx	Solteq Merx on Solteq Oyj:n alusta lähtien itse kehittämä toiminnanohjausjärjestelmä, jonka kehitys on aloitettu 1980-luvun lopulla. Merx on kehittynyt vuosien saatossa erittäin laajaksi järjestelmäksi, jota kehitetään edelleen aktiivisesti.
IBM i / System i	System i, AS/400, i5 on IBM:n midrange-laitteisto (keskisuuri keskuskone), joka julkaistiin ensimmäistä kertaa vuonna 1988. Laitteiston käyttöjärjestelmänä on IBM i, OS/400, i5/OS, i/OS. Laitteiston ja käyttöjärjestelmän nimi on vaihdettu useita kertoja kun sen teknologia on muuttunut merkittävästi.
XML	XML (Extensible Markup Language) on World Wide Web Consortiumin vuonna 1996 julkaisema merkintäkieli. Sitä käytetään yleisesti formaattina tiedon välityksessä järjestelmien välillä, sekä tiedon tallentamisessa. XML on teksti-

muotoista ja muistuttaa ulkonäöltään hyvin paljon HTML-kieltä, jolla www-sivut rakennetaan.

YUI

YUI (The Yahoo! User Interface Library) on avoimen lähdekoodin JavaScript kirjasto, jolla voidaan toteuttaa rikkaita internet-sovelluksia. YUI:n kehityksen aloitti vuonna 2005 Yahoo!, joka julkaisi sen yleisölle seuraavana vuonna.

1 JOHDANTO

1.1 Tausta

Rikkaat internet-sovellukset (RIA), eli selaimessa toimivat sovellukset, muistuttavat toiminnaltaan työpöytäsovelluksia ja ovat yleistyneet viime vuosina. Yhä useampi järjestelmä tarjoaa selainkäyttöliittymän, jonka avulla järjestelmän käyttäminen on mahdollista ilman asennettavaa asiakasohjelmistoa. Toiminnanohjausjärjestelmät eivät ole poikkeus tässä suhteessa. Useissa toiminnanohjausjärjestelmissä on mahdollisuus käyttää järjestelmää selaimen kautta web-sovelluksella.

Opinnäytetyön toimeksiantaja ja nykyinen työnantajani Solteq Oyj on kehittänyt omaa Merx-toiminnanohjausjärjestelmää jo yli 20 vuotta suomalaisille kaupan alan toimijoille. Merxin perinteinen tekstipohjainen käyttöliittymä on erittäin tehokas ja nopea tapasyöttää tietoa. Tekstipohjainen käyttöliittymä ei kuitenkaan kykene kaikkeen. Esimerkiksi tilastoinnin graafien ja tuotekuvien näyttäminen tekstipohjaisella käyttöliittymällä on vaikeaa, ellei mahdotonta. Näitä ominaisuuksia silmällä pitäen on toteutettu Solteq webMerx -käyttöliittymämooottori, jolla voidaan tukea nykyistä tekstipohjaista käyttöliittymää ja tarjota aivan uusia mahdollisuuksia graafisella käyttöliittymällä.

Solteq webMerx -käyttöliittymämooottori ei ole perinteinen graafinen käyttöliittymä. Sen avulla voidaan luoda Merxin ohjelmille selainkäyttöliittymäruutuja kuvaamalla ne XML-kuvauskielillä. Käyttöliittymämooottori helpottaa ja nopeuttaa uusien käyttöliittymäruutujen luomista Merxin ohjelmiin. Uusien käyttöliittymäruutujen kuvaamiseen tarvitaan XML-kuvauskielen osaamista ja valmiiden mallien ansiosta ruudun kuvaaminen onnistuu minuuteissa. Käyttäjälle webMerx näyttää selaimessa käytettävältä käyttöliittymältä, mutta todellisuudessa käyttöliittymämooottori luo käyttöliittymäruudut kuvaustiedostojen pohjalta käyttäjän niitä pyytäessä.

Solteq webMerx -käyttöliittymämooottorin luoma käyttöliittymä ei ole ensimmäinen selainkäyttöliittymä Merxiin. Ensimmäinen Merxin selainkäyttöliittymä on tehty IBM WebFacing:llä vuonna 2008, jonka avulla voidaan luoda käyttöliittymäruutuja selaimen DDS-tiedostojen pohjalta. DDS-tiedostoja käytetään RPG-ohjelmoinnissa kuvaamaan ohjelman käyttöliittymäruutuja (IBM, [www-sivu](#) 2012). Solteq webMerxin

seuraava versio toteutettiin vuoden 2010 alussa, käyttäen Zend Framework -sovelluskehystä, PHP:tä, JavaScriptiä, jQuery- ja jQueryUI JavaScript -kirjastoja. Uusi selainkäyttöliittymä ei kuitenkaan ollut käyttöliittymämoottori, jonka avulla käyttöliittymäruutujen kehittäminen olisi ollut mahdollista ilman web-tekniikoiden osaamista.

1.2 Tavoitteet ja tarkoitus

Toukokuussa 2011, kun aloitin harjoittelujaksoni Solteq Oyj:ssä, oli tarve kehittää Merxiin selaimella käytettävä uusi graafinen käyttöliittymä, käyttöliittymämoottori. Tavoitteena oli toteuttaa selainkäyttöliittymä Solteq Merx toiminnanohjausjärjestelmään, jota on mahdollisimman helppo muokata eri asiakkaiden tarpeisiin, sekä mahdollistaa helppo tapa tehdä uusia käyttöliittymäruutuja ilman PHP- tai JavaScript-osaamista.

Uusien käyttöliittymäruutujen luominen on tehtävä mahdollisimman helpoksi ja niin, että ilman minkäänlaista web-ohjelmointiosaamista on mahdollista luoda uusia ruutuja. Ruutujen luomiseen valittiin XML-kuvauskieli, jonka avulla kuvataan halutunlainen käyttöliittymäruutu. Toinen mahdollinen vaihtoehto olisi voinut olla JSON. Valinta osui XML-kuvauskieleen, koska sen koettiin olevan helpompilukuista ja Solteq Merxissä on käytössä XML-sanomat, joiden kautta XML-kieli on jo tuttua. Käyttöliittymämoottori osaa lukea näitä kuvaustiedostoja ja muodostaa niiden pohjalta käyttöliittymäruudun. Kuvaustiedoston pohjalta käyttöliittymämoottorin tulee osata myös luoda ruudulle toiminnallisuus. Näin pyritään minimoimaan ohjelmoinnin tarve uusien ruutujen luomisen yhteydessä, sekä nopeuttaa uusien ruutujen luomista.

Käyttöliittymämoottorin muokattavuus asiakaskohtaisesti on erityisen tärkeää. Useilla asiakkailla on erilaisia vaatimuksia, jotka muokkaavat käyttöliittymämoottorin toiminnallisuutta, henkilöiden valtuuksia, sekä oletusasetuksia. Asetustasoja on oltava tarvittava määrä, jotta voidaan varmistaa käyttöliittymämoottorin muokattavuus asiakkaan tarpeiden mukaan. Asetustasojen käytöllä pyritään pois sulkemaan asiakaskohtaista ohjelmointia, jonka ylläpidettävyys on hankalaa ja aikaa vievää.

Modulaarisuus on käyttöliittymämoottoria toteutettaessa erittäin tärkeässä asemassa. Moottoria kehitettäessä on otettava huomioon, että kaikkia mahdollisia käyttötarkoituksia ei vielä tiedetä. Uusien ominaisuuksien löytyessä on moottorin laajentaminen oltava

mahdollisimman helppoa. Modulaarisuuden saavuttamisen apuna voidaan käyttää sovelluskehysten tarjoamaa MVC-arkkitehtuuria. MVC-arkkitehtuurissa ohjelma jaetaan kolmeen eri osaan: malli, näkymä ja käsittelijä. Sen avulla saadaan hajautettua ohjelma pienempiin palasiin, joiden ylläpito ja kehittäminen on helppoa. MVC-arkkitehtuuria käytetään yleisesti ohjelmissa, joissa on graafinen käyttöliittymä. Sen avulla voidaan erottaa käyttöliittymä ohjelmalogiikasta.

Tässä opinnäytetyössä kirjallisessa osuudessa vastataan seuraaviin kysymyksiin:

- Miksi Merxiin toteutettiin käyttöliittymämoottori?
- Miten käyttöliittymämoottori toimii?
- Mitä ominaisuuksia käyttöliittymämoottorin luoma selainkäyttöliittymä tarjoaa nykyiseen tekstipohjaiseen käyttöliittymään verrattuna?
- Onko asiakkaille hyötyä uudesta Web-käyttöliittymästä?

2 TOIMEKSIANTAJAN ESITTELY

2.1 Solteq Oyj

Opinnäytetyön toimeksiantajana toimi Solteq Oyj, joka on tamperelainen ohjelmisto-palveluyhtiö. Se on perustettu vuonna 1982 nimellä Tampereen Tiedonhallinta Oy. Solteq tarjoaa toiminnan- ja taloudenohjauksen palveluja teollisuuden, kaupan, logistiikan ja julkishallinnon toimijoille. Helsingin pörssissä yhtiö on noteerattu vuodesta 1999 lähtien, jolloin se vaihtoi nimensä nykyiseen muotoon. Solteq Oyj työllistää noin 290 henkilöä, joista suurin osa työskentelee Tampereen toimipisteessä. Muut toimipisteet sijaitsevat Helsingissä ja Lahdessa. (Solteq Oyj, www-sivu 2012)

Solteq Oyj jakautuu kahteen eri liiketoiminta-alueeseen: toiminnanohjauksen erityisratkaisut sekä toiminnan- ja taloudenohjaus. Toiminnanohjauksen erityisratkaisut liiketoiminta-alueeksi yhdistyi vuoden 2012 alusta kolme yksikköä: EAM, STORE ja DATA. Se pitää sisällään kunnossapidon, huoltopalvelun ja kenttätyön hallinnan, erikoiskaupan, sekä masterdatan hallinnan ratkaisut. Toiminnan- ja taloudenohjaus on toiminut nimellä ERP-liiketoimintayksikkö vuoden 2012 alkuun asti. Se kattaa toiminnanohjausjärjestelmät, taloushallinnon sekä asiakkuuden- ja toimitusketjun hallinnan. Itse työskentelyn toiminnan- ja taloudenohjauksen yksikössä Solteq Merx nimisen toiminnanohjausjärjestelmän parissa. (Solteq Oyj, www-sivu 2012; Solteq Oyj -vuosikertomus 2011.)

Opinnäytetyötä kirjoittaessa Solteq Oyj osti Aldata Solution Finland Oy:n suomen liike-toiminnan. Yrityskaupan yhteydessä henkilöstön määrä kasvoi noin 240 henkilöstä noin 290 henkilöön. Kaupan yhteydessä Solteq Oyj:stä tuli kaupanalan markkinajohtaja suomessa. (Solteq Oyj, www-sivu 2012)

2.2 Solteq Merx

Solteq Merx on Solteq Oyj:n alusta lähtien itse kehittämä toiminnanohjausjärjestelmä tukku- ja vähittäiskaupan toimijoille. Järjestelmän kehittäminen on aloitettu 1980-luvun loppupuolella ja jatkuu edelleen aktiivisesti päivittäin. Merxin alkuaajoista järjestelmä on kehittynyt runsaasti ja nykyisin siihen on saatavilla useita liitännäistuotteita ja se sisäl-

tää laajan rajapinnan erilaisille liittymille eri tietojärjestelmiin. Merxin modulaarinen rakenne mahdollistaa järjestelmän käyttöönoton asiakkaan tarpeiden ja aikataulun mukaisesti. (Solteq Merx -tuotekortti 2011.)

Merx on kehitetty IBM Power Systems -laitteistolle ja IBM i -käyttöjärjestelmälle suurimmaksi osaksi käyttäen RPG-ohjelmointikieltä, joka on IBM:n vuonna 1959 lanseeraama korkean tason ohjelmointikieli. Vuonna 1994 RPG -ohjelmointikielestä julkaistiin versio IV, joka tunnetaan paremmin nimellä ILE RPG. Kielen uusin versio RPG IV versio 7, julkaistiin 2010, joka antoi ohjelmointikielelle taas uusia ominaisuuksia. Modernit ohjelmointikielet tekevät tulemistaan kovaa vauhtia myös System i-alustalle. Muutaman vuoden aikana Merxissä on alettu hyödyntämään paljon PHP-ohjelmointikieltä erilaisissa ratkaisuissa. PHP:n avulla Merxistä voidaan muodostaa nykyaikaisia tiedostomuotoja kuten PDF ja Excel-dokumentteja, sekä se on mahdollistanut WebService ja XML-sanomaliikenteen. (Solteq Merx -tuotekortti 2010.)

Merxin tekstipohjaista käyttöliittymää käytetään telnet-yhteyden yli yleensä IBM:n Client Access -sovelluksella. Tekstipohjaiset käyttöliittymät alkavat olla nykyään harvinaisen näky ja rajoittavat osittain käyttöliittymän käyttömahdollisuuksia. Henkilölle joka on työskennellyt Merxin parissa vuosia, tekstipohjaisen käyttöliittymän käyttäminen on mutkatonta ja nopeaa. Työmaailmaan on kuitenkin saapumassa uusi sukupolvi, joka ei ole tottunut käyttämään tekstipohjaisia käyttöliittymiä, vaan he ovat tottuneet graafisiin käyttöliittymiin. Merxin graafinen selainkäyttöliittymä tarttuu tähän haasteeseen ja tuo uudenlaisen tavan käyttää Merxiä, sekä aivan uusia asiakkaan kannalta hyödyllisiä ominaisuuksia.

3 KÄYTETYT TEKNIIKAT JA KIELET

3.1 PHP

PHP (PHP: Hypertext Preprocessor) on avoimen lähdekoodin ohjelmointikieli, joka on erityisesti tarkoitettu web-kehitykseen. PHP-ohjelmakoodia voidaan sisällyttää HTML-kielen sekaan ja näin voidaan luoda dynaamisia www-sivuja. PHP on komentosarjakie-
li, jonka ohjelmakoodi tulkitaan ajon aikana. Kielen vahvuuksia on helppokäyttöisyys ja monipuolisuus. Toisin kuin esimerkiksi Java, PHP on heikosti tyypitetty kieli, joka tuo omalta osaltaan kieleen helppokäyttöisyyttä. (The PHP Group. [www-sivu](#). 2012.)

PHP-ohjelmointikieli sai alkunsa 1994, kun Rasmus Lerdorf kirjoitti kokoelman CGI-skriptejä C-ohjelmointikielellä ja nimesi sen "Personal Home Page Tools" eli "PHP Tools". Rasmus Lerdorf julkaisi kokoelman GPL-lisenssillä vuonna 1995 nimellä PHP/FI (Personal Home Page / Forms Interpreter). Marraskuussa 1997 julkaistiin versio 2.0 ja sillä oli useita tuhansia käyttäjiä ympäri maailmaa. Seuraavan vuoden kesäkuussa PHP sai nykyisen nimensä (PHP: Hypertext Preprocessor), kun siitä julkaistiin kolmas versio. PHP 3.0 oli kirjoitettu lähes kokonaan uudestaan Andi Gutmansin ja Zeev Suraskin toimesta, koska he totesivat sen olevan riittämätön internet-kauppasovelluksien tarpeisiin. Gutmans ja Suraski kirjoittivat PHP:n ytimen uudestaan, jotta se tukisi kolmansien osapuolien ohjelmointirajapintoja. Tämä uusi ydin tuli käyttöön vuoden 2000 marraskuussa julkaistuun PHP 4:ään. PHP:n uusin versio 5 julkaistiin heinäkuussa 2004 ja se tukee olio-ohjelmointia, sekä sisältää sisäänrakennetun tietokantamoottorin. PHP:n versio 5.4 julkaistiin vuonna 2012. (The PHP Group. [www-sivu](#). 2012.)

3.2 JavaScript

JavaScript on alustariippumaton olio-ohjelmointikieli, jota ei ole tarkoitettu käytettäväksi yksin, vaan se on suunniteltu niin, että se on helppo sisällyttää muihin ohjelmistoihin, kuten selaimiin. Vuonna 1995 Brendan Eich kehitti JavaScriptin Netscapelle nimellä Mocha. Myöhemmin nimi vaihdettiin LiveScriptiksi ja lopulta markkinointisystä JavaScriptiksi. (Mozilla Developer Network. [www-sivu](#). 2012.)

JavaScriptillä ei ole Java-ohjelmointikielen kanssa nimestään huolimatta kovinkaan paljon yhteistä. JavaScript muistuttaa syntaksiltaan hyvin paljon Javaa, mutta se ei ole vahvasti tyyditetty kieli kuten Java. JavaScriptissä ei tarvitse esitellä käytettäviä muuttujia, luokkia, eikä metodeita. JavaScript on hyvin vapaa kieli verrattuna Java-ohjelmointikieleen. (Mozilla Developer Network. [www-sivu](#). 2012.)

3.3 XML

XML (Extensible Markup Language) on merkintäkieli, joka alun perin suunniteltiin vastaamaan isojen elektronisten julkaisuiden haasteisiin. Nykyisin XML on isossa roolissa tiedonvälityksessä verkossa ja muualla. XML-kielen kehitti XML Working Group, joka muodostettiin World Wide Web Consortiumin (W3C) alaisuuteen 1996. Tavoitteena oli luoda merkkauškieli, joka olisi mahdollisimman helppolukuista ja -käyttöistä. W3C on kehittänyt joukon XML-pohjaisia teknologioita, joiden tarkoituksena on välittää tietoa järjestelmien välillä. Yleisesti käytettyjä viestintämenetelmiä ovat mm. Web Services ja SOAP, jotka ovat käytössä myös Solteq Merxissä. (W3C Recommendation. [www-sivu](#). 2008.)

XML dokumentti koostuu elementeistä, jotka voivat sisältää toisia elementtejä. Elementteillä voi olla myös attribuutteja, jotka ovat nimi-arvo pareja. Kuvauksessa tulee aina olla ns. juurielementti, joka sisältää kaikki dokumentin elementit. Elementit merkitään HTML-kielestä tutulla tagi-merkinnällä ja jokaisella elementillä pitää olla aloitus- ja lopetus-tagit. Kehittäjä voi itse määritellä haluamiaan elementtejä haluamansa määrän, kunhan elementit muistetaan aloittaa ja lopettaa oikein. (W3C Recommendation. [www-sivu](#). 2008.)

3.4 Ext JS -sovelluskehys

Ext JS JavaScript -sovelluskehys on tarkoitettu rikkaiden internet-sovelluksien (RIA) tekemiseen. Ext JS -sovelluskehys tarjoaa laajan kirjaston käyttöliittymäkomponentteja, joiden avulla on helppo toteuttaa monimutkaisiakin internet-sovelluksia. (Groner. 2011. 7-8)

Ext JS sai alkunsa vuonna 2006, kun Jack Slocum teki YUI-JavaScript kirjastoon uuden tyyppisen taulukkokomponentin, joka tarjosi käyttäjälle useita työpöytäsovelluksista tuttuja ominaisuuksia. Noin vuosi myöhemmin 2007, Slocum julkaisi Ext JS 1.0 nimisen kirjaston, joka perustui YUI-JavaScript kirjastoon. Myöhemmin samana vuonna julkaistiin Ext JS 1.1, joka ei ollut enää riippuvainen ulkoisista kirjastoista ja Ext JS JavaScript-sovelluskehys oli syntynyt. Vaikka Ext JS sai alkunsa vain yhdestä taulukkokomponentista, se ei ole tyyppillinen JavaScript-kirjasto, joka tarjoaa ohjelmoijalle suuren määrän valmiita toiminnallisuuksia. Ext JS on erittäin paljon enemmän. Se tarjoaa ohjelmoijalle kaiken mitä JavaScript-kirjastolta odotetaan ja lisäksi helposti periytettävän, visuaalisesti näyttävän komponenttikirjaston, joka voidaan toteuttaa MVC-arkkitehtuurilla. (Orchard, Pehlivanian & Jones. 2009, 336)

Ext JS -sovelluskehysten uusin versio 4.0 tarjoaa paljon uudistuksia edelliseen versioon Ext JS 3.x verrattuna. Uusi versio toi mukanaan 50 uutta luokkaa, 350 uutta ohjelmointirajapintaa (API) ja lisäsi dokumentaatiota huomattavasti 65 prosentilla. Yksi suurimmista muutoksista on uusi MVC-arkkitehtuuriin perustuva sovelluskehys. Ext JS:n tarjoamassa MVC-arkkitehtuurissa on mukana perinteisten mallin (model), näkymän (view) ja käsittelijän (controller) lisäksi varasto (store), jonka tarkoituksena on säilöä välimuistissa säilytettävä tieto. Uusin versio toi mukanaan myös uudet diagrammikomponentit, jotka aikaisemmassa versiossa olivat lisäosana ja toteutettuna flash-tekniikalla. Uudet diagrammit tehtiin käyttäen SVG- ja VML-tekniikkaa ja ne toimivat kaikilla selaimilla. (Sencha Ext JS, www-sivu 2012.)

Ext JS 4 JavaScript -sovelluskehys tarjoaa ohjelmoijalle kaikki monipuolisen käyttöliittymän tarvitsemat komponentit. Komponenttien joukosta löytyy mm. monipuolinen taulukkokomponentti, Useita muokattavia lomakekomponentteja, useita erilaisia asettelumahdollisuuksia ja mahdollisuuden muokata kaikkien komponenttien ulkoasua helposti käyttäen SASS-tyylikieltä. (Sencha Ext JS, www-sivu 2012.)

3.5 Ext Direct

JavaScript on tulkittava oliopohjainen komentosarjakieli, jonka suoritus tapahtuu aina asiakasohjelmistossa. Web-ohjelmien tekemisessä käytetään yleensä palvelin-asiakas -ohjelmointimallia. Asiakasohjelmasta tehdään kutsuja palvelimelle käyttäen Ajax-

tekniikoita, joiden avulla voidaan päivittää tietoa www-sivulla ilman koko sivun uudelleenlatausta. (Sencha Ext Direct, www-sivu 2012.)

Ext Direct on luotu helpottamaan keskustelua asiakasohjelmiston ja palvelimen välillä. Se on laitealusta- ja ohjelmointikieliriippumaton teknologia, jolla voidaan paljastaa palvelinohjelmasta metodeita asiakasohjelmalle. Ext Direct on saatavilla kaikkiin tunnetuimpiin palvelinohjelmointiympäristöihin, kuten PHP, Java, .Net, ColdFusion, Ruby ja Perl. Ext Direct koostuu kolmesta avainkomponentista, jotka ovat konfiguraatio, ohjelmointirajapinta (API) ja reititin. (Sencha Ext Direct, www-sivu 2012.)

Konfiguraatio

Konfiguraation avulla kerrotaan, mitkä metodit halutaan paljastaa asiakasohjelmiston käyttöön. Konfiguraatiosta ilmenevät asiakasympäristölle paljastetut metodit, luokat joissa ne sijaitsevat ja metodille välitettävien parametrien määrä. Ext Direct tarjoaa neljä eri tapaa konfiguraation toteuttamiseen: ohjelmallinen, JSON, XML tai metadata. Ohjelmallisessa konfiguraatiossa luodaan taulukko palvelinohjelmassa käytetyllä ohjelmointikielellä, jossa konfiguraatio muodostetaan käyttäen avain-arvo pareja. Yksinkertainen esimerkki konfiguraatiosta on luokan "User" metodi "getName". Metodi paljastetaan kertomalla moniulotteisessa taulukossa luokka, jossa metodi sijaitsee, metodin nimi ja metodin saamien parametrien määrä. (Sencha Ext Direct, www-sivu 2012.)

```
$API = array(
    "User" => array(
        "methods" => array(
            "getName" => array(
                "len" => 0
            )
        )
    )
);
```

Konfiguraatio voidaan toteuttaa myös JSON tai XML kielellä, jolloin ennen käyttöä se luetaan ja puretaan palvelinohjelmointikielen ymmärtämään muotoon. Osa palvelinohjelmistoista vaatii vähemmän informaatiota, kuten ColdFusion, koska ne osaavat tulkita omia metodeitaan ohjelman ajonaikana. Tässä tapauksessa on mahdollista käyttää kon-

figuraation tekemiseen metadata -lähestymistapaa. (Sencha Ext Direct, [www-sivu 2012.](#))

```
<User>
  <methods>
    <method name="getName" len="0" />
  </methods>
</User>
```

```
{
  User: {
    methods: {
      getName: {
        len : 0
      }
    }
  }
}
```

Ohjelmointirajapinta

Ohjelmointirajapinta (API) komponentin tarkoituksena on käyttää konfiguraatiota ja luoda sen pohjalta JavaScript kuvaus paljastetuista metodeista asiakasohjelmistolle. Ohjelmointirajapinnan luomien välityspalvelin -metodien ansiosta asiakasohjelmistossa voidaan kutsua suoraan palvelinohjelmiston metodeita. (Sencha Ext Direct, [www-sivu 2012.](#))

```
Ext.app.REMOTING_API = {
  "url": "remote/router.php",
  "type": "remoting",
  "actions": {
    "User": [{
      "name": "getName",
      "len": 0
    }]
  }
};
```

Reititin

Reititin ottaa vastaan asiakasohjelmiston pyynnöt ja ohjaa ne konfiguraatiossa paljastettujen luokkien metodeille, välittäen kutsun mukana lähetetyt parametrit. Reititin osaa ottaa vastaan useita asiakkaalta tulevia pyyntöjä yhdellä kertaa. Asiakasohjelmistossa pyynnöt kasataan taulukkoon ja lähetetään reitittimelle, joka purkaa taulukon ja jakaa pyynnöt oikeille metodeille. Kun metodit on suoritettu, reititin yhdistää vastaukset taas taulukoksi ja välittää ne takaisin asiakasohjelmistolle, jossa taulukko puretaan. (Sencha Ext Direct, [www-sivu 2012.](#))

Reititin saa pyynnön yhteydessä tarvittavat tiedot sen käsittelemiseen. Pyyntön mukana tulee tietysti tieto kutsutusta metodista ja luokasta, jossa se sijaitsee. Mikäli kutsussa on mukana metodille tarkoitettuja parametreja, välitetään ne ”data”-lohkossa avain-arvo pari taulukkona. Mukana tulee myös kutsun tyyppi, joka on aina Ext Directin tekemissä kutsuissa ”rpc” ja kutsun tunnistenumero (tid), joka yksilöi kutsun. (Sencha Ext Direct, [www-sivu 2012.](#))

```
{<a href=""User", "method": "getName", "data": [],  
"type": "rpc", "tid": 2">action</a>}
```

3.6 Zend Framework -sovelluskehys

Zend Framework on avoimen lähdekoodin olio-ohjelmointi -periaatteella toteutettu PHP 5 sovelluskehys verkkosovelluksien kehittämiseen. Sen tarkoituksena on nopeuttaa ja helpottaa ohjelmistokehitystä, sekä selkeyttää ohjelman rakennetta tarjoamalla erittäin modulaarinen MVC-arkkitehtuuri. MVC-arkkitehtuurin ansiosta ohjelmakoodi on uudelleen käytettävää ja helpommin ylläpidettävää. Sovelluskehysten tarkoituksena on helpottaa ohjelmoijan työtä ja auttaa suorittamaan yleisimpiä ohjelmointitehtäviä. (Zend Framework, [www-sivu 2012.](#))

Zend Framework -sovelluskehys tarjoaa ohjelmoijan käyttöön laajan luokkakirjaston, joka sisältää ohjelmoinnissa yleisimmin tarvittuja komponentteja. Luokkakirjaston kaikki luokat on kirjoitettu olio-ohjelmointi periaatteen mukaisesti. Sovelluskehys on toteutettu niin, että ohjelmoija voi ottaa käyttöönsä vain ne luokat ja ominaisuudet, joita

hän tarvitsee. Zend Framework -sovelluskehys tarjoaa ohjelmoijan käyttöön laajan rajapinnan useille yleisimmille relaatiotietokantaohjelmistoille, kuten MySQL, IBM DB2, Microsoft SQL Server, SQLite, Informix Dynamic Server ja MariaDB. (Zend Framework, www-sivu 2012.)

3.7 Zend Core for i5/OS -palvelinohjelmistopaketti

PHP IBM i-alustalla on vielä uusi tuttavuus. Vuonna 2005 IBM ja Zend aloittivat yhteistyön, jonka johdosta vuonna 2006 julkaistiin Zend Core for i5/OS. Julkaisu oli ensimmäinen virallinen ja tuettu PHP IBM i-alustalla. System i:llä ohjelmoidaan perinteisesti RPG-, COBOL- ja C-ohjelmointikielillä, mutta Zend Core for i5/OS:n julkaiseminen antoi paljon uusia mahdollisuuksia käyttää moderneja ohjelmointikieliä, kuten PHP ja JavaScript. Vuonna 2007 julkaistiin Ohjelmointiympäristö (IDE) Zend Studio, sekä Zend Platform -tuote, joka parantaa PHP:n suoritussykyä ja antaa joitakin lisäominaisuuksia. Vuonna 2008 Zend Framework -sovelluskehukseen tuli tuki IBM i:n DB2-relaatiotietokannalle ja vuonna 2010 julkaistiin Zend Server, joka yhdisti Zend Core:n ja Zend Platform:n parhaat puolet. (Anderson, Kosturjak & Mullen-Schultz. 2007; Seiden A. pdf-dokumentti. 2010)

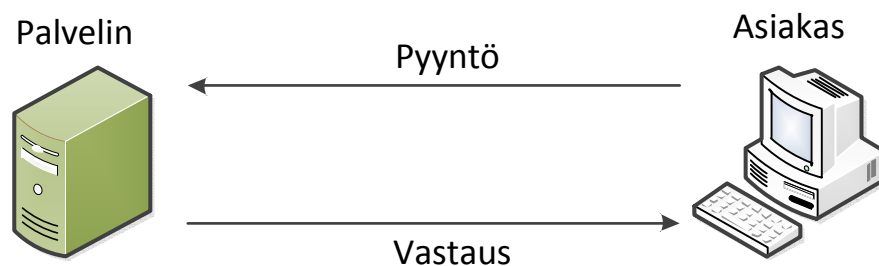
Zend Core for i5/OS koostuu Apache HTTP -palvelimesta, PHP:sta ja sen lisäosista, sekä MySQL:stä.. Zend Core for i5/OS:n erikoisuutena verrattuna kahteen edellä mainittuun on, että siinä on kaksi Apache HTTP -palvelinta. Toinen palvelimista toimii pelkkänä välityspalvelimena jonka liikenne ohjataan toiseen palvelimeen. Tämän on tarkoitus parantaa Zend Coren tietoturva. (Anderson, Kosturjak & Mullen-Schultz. 2007; Seiden A. pdf-dokumentti. 2010)

Zend Core for i5/OS:n mukana tulee AURA equipmentsin kehittämä PHP Toolkit for IBM i -lisäosa, jonka avulla voidaan mm. kutsua suoraan PHP:stä RPG-ohjelmia. Tämä lisäosan tuoma mahdollisuus on Merxin kannalta erityisen tärkeä. Merxiä on kehitetty useita vuosia ja kaikki liiketoimintalogiikka sijaitsee RPG-ohjelmissa. Lisäosan ansiosta tätä liiketoimintalogiikkaa voidaan käyttää PHP-ohjelmissa ja webMerx -käyttöliittymämoottorissa. (Anderson, Kosturjak & Mullen-Schultz. 2007; Seiden A. pdf-dokumentti. 2010)

4 KÄYTTÖLIITTYMÄMOOTTORIN TOIMINTAPERIAATE

4.1 Käyttöliittymämoottorin rakenne

Käyttöliittymämoottori perustuu palvelin-asiakas -malliin (client-server). Asiakas on tässä mallissa ohjelma, joka toimii käyttäjän tietokoneella selaimessa. Palvelin on ohjelma, joka toimii iSeries-palvelinkoneessa ja tarjoaa asiakkaalle vastauksia sen lähettämiin pyyntöihin.

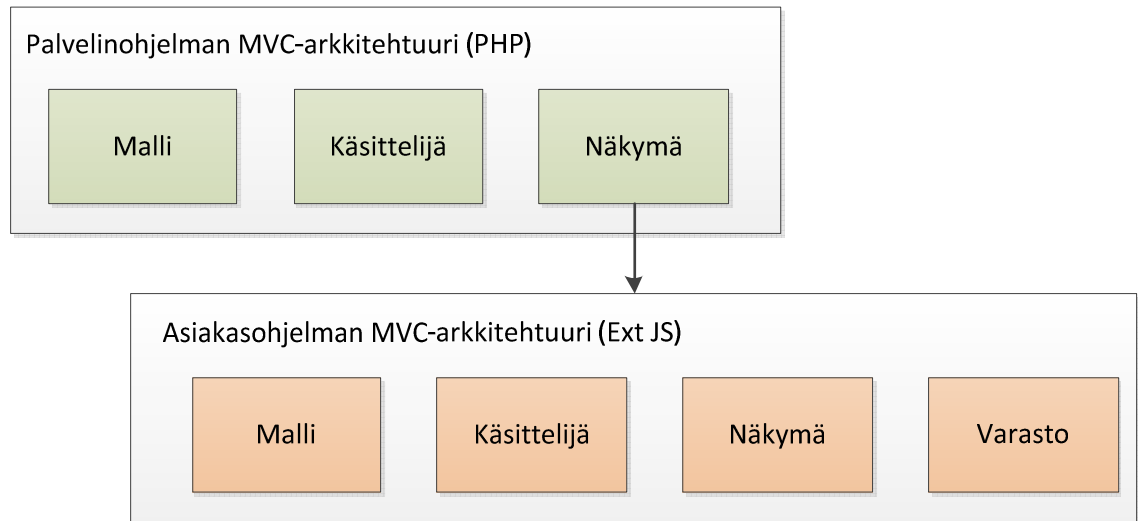


Palvelinohjelma on toteutettu käyttäen PHP-ohjelmointikieltä. Palvelinohjelman toteutuksessa on käytetty MVC-arkkitehtuuria, jotta ohjelman modulaarisuus ja jatkokehitys olisi mahdollisimman helppoa. Se vastaa konfiguraatitiedostojen käsittelystä, käyttöliittymäruutujen kuvauksen käsittelystä, keskustelusta Merxin kanssa, sekä asiakasohjelman kutsujen käsittelystä. Ohjelma keskustelee Merxin kanssa kutsuen RPG-ohjelmia i5toolkitin avulla. Tämän keskustelun toteutuksessa on käytetty apuna pientä osaa Zend Framework -sovelluskehiksestä.

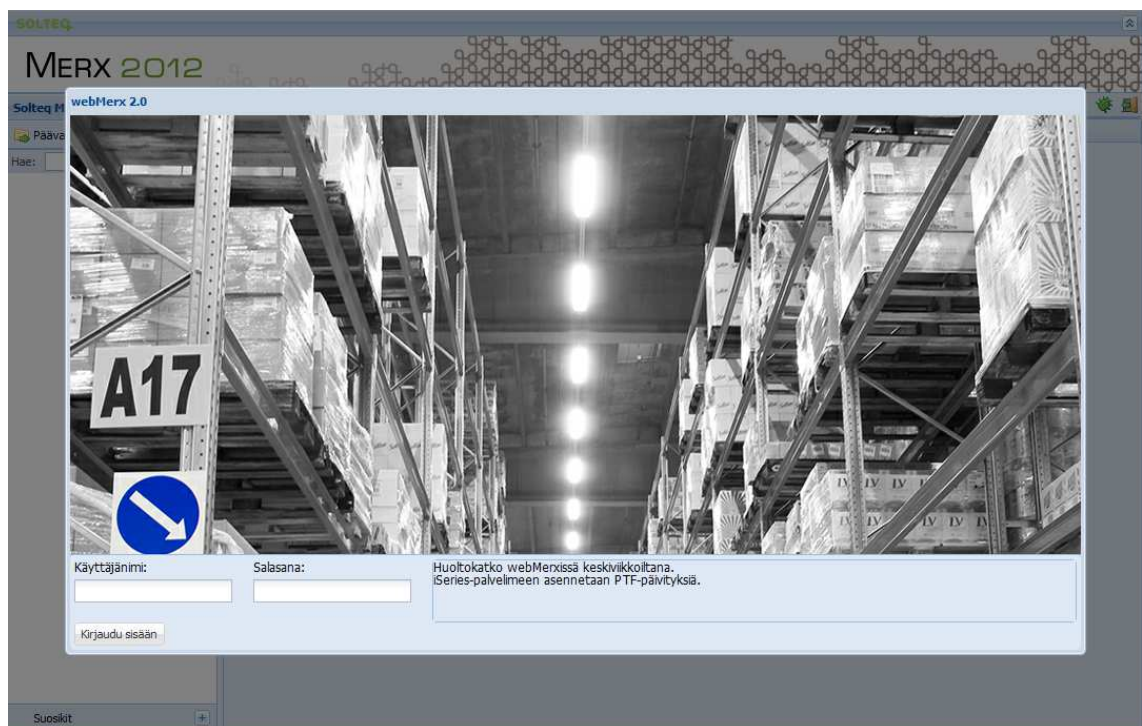
Asiakasohjelma on toteutettu Ext JS -sovelluskehiksen versiolla 4.0, käyttäen sovelluskehiksen tarjoamaa MVC-arkkitehtuuria. Asiakasohjelma tekee pyyntöjä palvelinohjelmalle ja vastauksien perusteella näyttää saamaansa tietoa tai luo käyttöliittymän moottorin kuvauksen perusteella.

Asiakkaan ja palvelimen välisen keskustelun toteutuksessa on hyödynnetty Ext Direct -tekniikkaa, jossa palvelinohjelman metodeita paljastetaan asiakasohjelmalle ja voidaan kutsua suoraan asiakasohjelmasta. Asiakasohjelma lähettää pyynnön palvelinohjelmalle, joka prosessoi pyynnön ja lähettää vastauksen takaisin asiakkaalle.

Käyttöliittymämoodorin hakemistorakennetta tarkkaillessa huomaa, että palvelinohjelman MVC-arkkitehtuurin näkymä-osa (view) sisältää asiakasohjelman MVC-arkkitehtuurin. Käyttöliittymämoodorissa on siis kaksi MVC-arkkitehtuuria sisäkkäin.



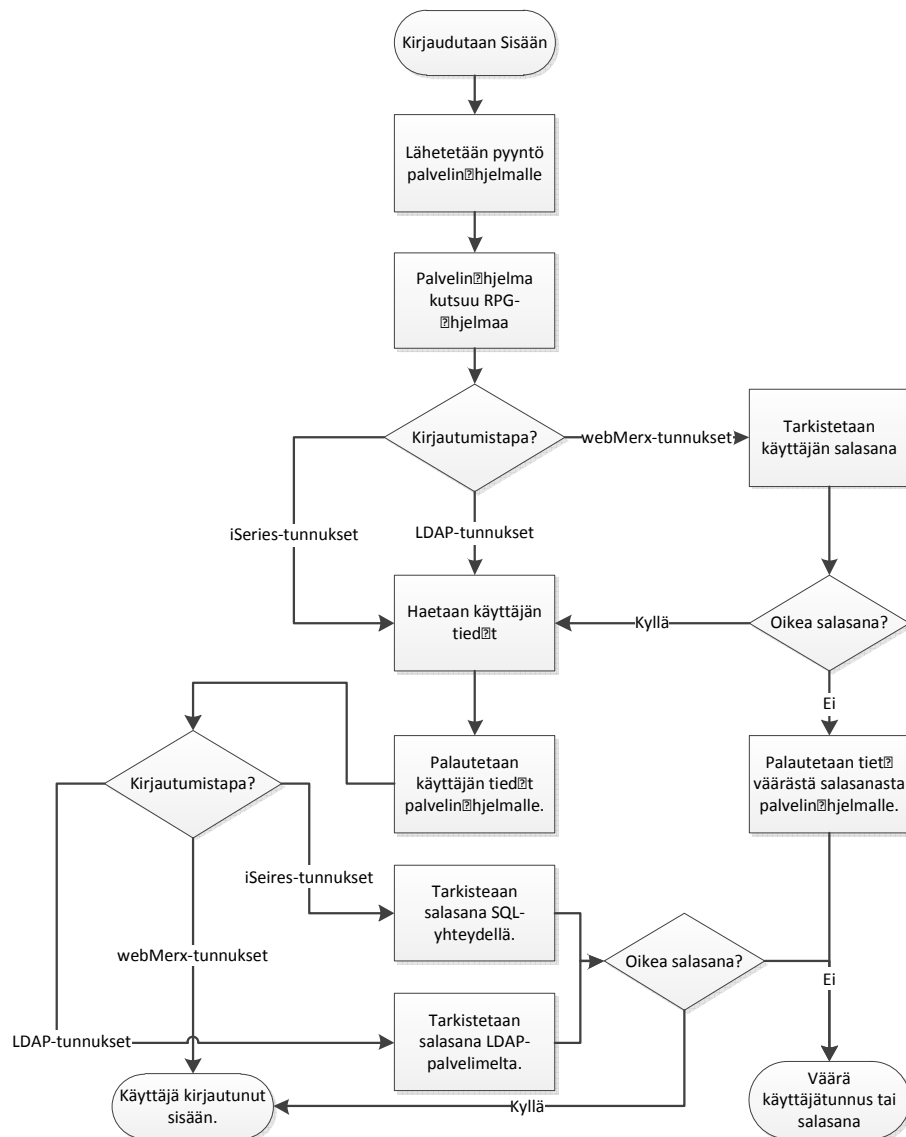
Web-käyttöliittymää avatessa ennen kun käyttöliittymä ladataan, kutsutaan ensimmäisen kerran RPG-ohjelmaa. Ohjelmalta kysytään onko webMerx-käyttöliittymä käytössä ja muita järjestelmän kannalta merkittäviä tietoja. Mikäli RPG-ohjelma vastaa webMerxin olevan toimintavalmiudessa ladataan käyttöliittymän runko ja kirjautumisikkuna (Kuvio 1.).



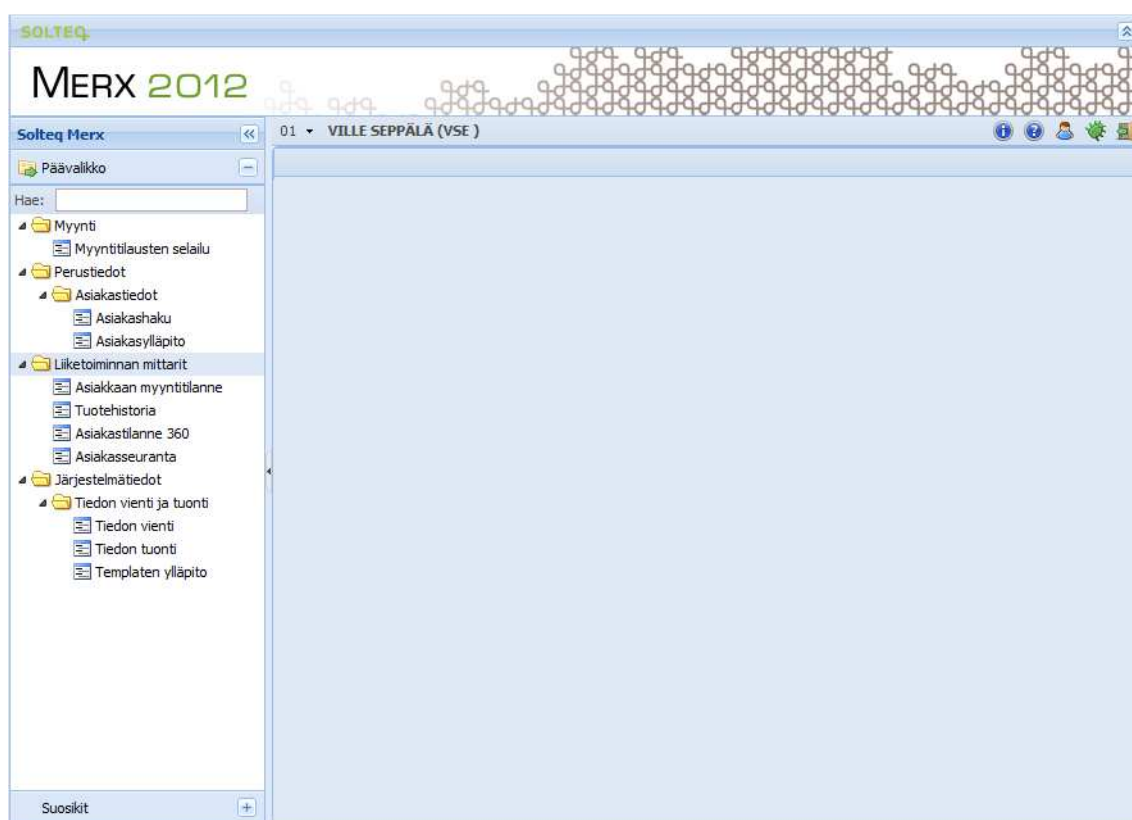
Kuvio 1. webMerx käyttöliittymän runko ja kirjautumisikkuna.

Web-käyttöliittymään on mahdollistettu kolme eri kirjautumistapaa. Järjestelmään voidaan kirjautua samoilla tunnuksilla, joilla kirjaudutaan perinteiseen Merxiin. Toinen mahdollisuus on kirjautua webMerx-tunnuksilla, jotka on linkitetty henkilön Merx-profiiliin. Mahdollistettu on myös kirjautuminen LDAP-tunnuksilla, jotka ovat samat tunnukset kuin joilla henkilö kirjautuu yrityksen verkkoon.

Käyttäjän kirjautuessa kutsutaan RPG-ohjelmaa, jolla käyttäjä tunnistetaan. RPG-ohjelma tarkistaa, että käyttäjä on olemassa ja mikäli käyttäjä on kirjautumassa webMerx-tunnuksilla, tarkistetaan myös käyttäjän salasana. LDAP- tai Merx-tunnuksilla kirjautuessa RPG-ohjelma ei tarkista salasanaa vaan ilmoittaa PHP:lle, että salasana on tarkistamatta, jolloin PHP tarkistaa sen. Käyttäjän tunnistamisen jälkeen RPG-ohjelma palauttaa käyttäjän tiedot kuten valikon-tunnuksen, käyttäjän nimen ja yksikön.



Käyttöliittymään kirjautumisen jälkeen käyttöliittymämooottori luo käyttäjälle asetukset periyttämällä kaikki asetustasot aina Solteq-tasosta käyttäjä-tasoon. Asetustiedot säilytetään PHP:n istuntoon tulevaa käyttöä varten. Kirjautumisen yhteydessä saadun valikko-tunnuksen avulla pyydetään RPG-ohjelmalta käyttäjälle tarkoitettua valikkoa. Valikoita voi olla useita erilaisia eri käyttäjäryhmille ja eri käyttötarkoituksiin. Valikon kautta käyttäjä pääsee käsiksi käyttöliittymäruutuihin, joiden kautta webMerxiä käytetään. Käyttöliittymäruutuja voidaan ajatella ohjelmina. Jokainen käyttöliittymäruutu on oma ohjelmansa, jolla on tietty käyttötarkoitus. Valikon ladattuaan käyttöliittymän runko on valmis ja käyttäjä voi aloittaa webMerxin käytön (Kuvio 2.).



Kuvio 2. webMerx-käyttöliittymä kirjautumisen jälkeen.

4.2 Käyttöliittymämooottorin asetustasot

Käyttöliittymämooottoriin on tarve usealle asetustasolle, joiden avulla voidaan muokata käyttöliittymän toimintaa. Asiakkailla on usein erilaisia käytäntöjä turvallisuuden ja toimintatapojen suhteen. Konfiguraation avulla voidaan mukauttaa käyttöliittymämooottori asiakkaiden tarpeiden mukaan ja välttää asiakaskohtaista ohjelmointia. Myös käyt-

täjillä on omia mieltymyksiä ja asetusten avulla voidaan antaa heille mahdollisuus muokata omaa käyttöliittymäkokemustaan.

Käyttöliittymämoottorin asetustasot on toteutettu XML-kuvaskielellä, kuten kaikki muutkin moottorin kuvaustiedostot. Käyttöliittymämoottorin palvelinohjelmisto osaa lukea, periyttää ja päivittää asetustiedostoja tarvittaessa. Asetustiedostojen muokkaamiseen ei ole olemassa vielä käyttöliittymää, lukuun ottamatta käyttäjätasoa, jota voidaan muokata käyttöliittymän asetusruudulla ja käyttäjän tekemien toimintojen mukaan.

Käyttöliittymämoottorissa on mahdollista suorittaa asetukset neljällä eri tasolla. Kaikilla tasoilla on oma merkityksensä ja järjestyksessä seuraava taso ylikirjoittaa edellisellä tasolla määritellyt asetukset. Käyttöliittymämoottorissa olevat tasot ovat: oletus, yritys, ryhmät ja käyttäjä (Kuvio 3.).



Kuvio 3. Käyttöliittymämoottorin asetustasot.

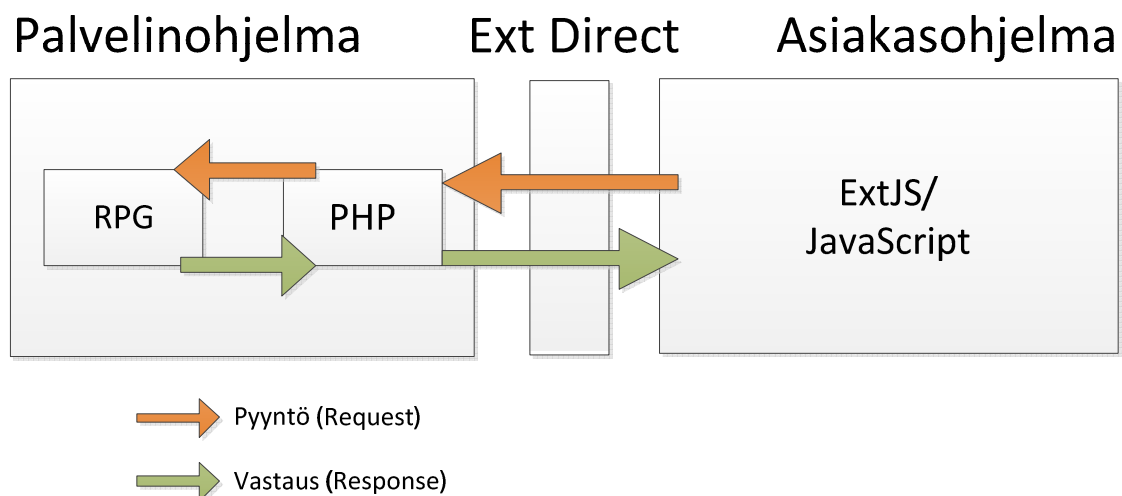
Oletus-taso sisältää oletusasetukset koko käyttöliittymämoottoriin. Oletusasetusten ylikirjoittaminen on mahdollista seuraavilla tasoilla. Seuraava taso on yritys-taso, jonka tarkoituksena on mahdollistaa yrityksen asetusten määrittely. Yritys-tasolla voidaan asettaa oletusarvoja, rajoituksia, sekä käyttöliittymän ulkoasuun vaikuttavia asetuksia. Ryhmä-taso muistuttaa hyvin paljon yritystasoa, mutta se ei välttämättä koske kaikkia käyttäjiä. Käyttäjät voivat kuulua eri ryhmiin ja saada tätä kautta oikeuksia tai rajoituksia käyttöliittymän käyttöön. Alin taso, eli käyttäjätaso sisältää käyttäjän asetukset.

Käyttäjän asetuksiin kuuluvat mm. teema ja käyttöliittymässä piilotetut komponentit. Periaatteessa millä tahansa tasolla voidaan ylikirjoittaa mikä tahansa edellisen tason asetus. Tätä mahdollisuutta kuitenkin rajoitetaan käyttöliittymässä annettavien asetusmahdollisuuksien perusteella.

4.3 Käyttöliittymäruutujen luominen

Käyttöliittymämoottorin päätavoitteena oli helpottaa uusien käyttöliittymäruutujen tekemistä, sekä mahdollistaa niiden tekeminen ilman PHP- tai JavaScript -osaamista. Käyttöliittymäruutujen luomisessa päädyttiin ratkaisuun, jossa ruudut kuvataan XML-kielellä mahdollisimman yksinkertaisesti ja käyttöliittymämoottori luo tämän kuvauksen perusteella käyttöliittymäruudun ja sen toiminnallisuuden.

Käyttöliittymäruudun luominen alkaa siitä, kun käyttäjä valitsee valikosta ohjelman, joka halutaan avata käyttöliittymään. Jokainen valikon kohta pitää sisällään tunnisteiden, jonka avulla haluttu käyttöliittymä voidaan tunnistaa. Esimerkiksi myyntitilausten selailu -ruutu on tunnisteeltaan MXMTH. Tämän tunnisteiden avulla käyttöliittymän asiakasohjelma lähettää pyynnön palvelinohjelmalle uudestaan käyttöliittymäruudusta (Kuvio 4.). Palvelinohjelma etsii tunnisteiden avulla käyttöliittymän kuvaustiedostoa ja lukee sen itselleen käyttöön. Mikäli tunnuksella ei löydetä kuvaustiedostoa, palautetaan asiakasohjelmalle virheviesti.



Kuvio 4. Palvelin- ja asiakasohjelmiston yhteydet.

Kun käyttöliittymämooottori on lukenut ruudun kuvaustiedoston (Liite 2.) moniulotteiseksi PHP taulukoksi, aloitetaan sen muokkaaminen. Mooottori etsii kuvauksesta attribuutteja jotka se korvaa useista muista XML-kuvaustiedostoista saamallaan kuvauksella. Koska XML:ssä kaikki arvot ovat PHP:n kannalta merkkijonoja, on numeeriset- ja totuusarvot mahdoton tunnistaa. Tästä johtuen näiden arvojen antaminen kuvaustiedostoissa tapahtuu attribuuttien avulla (Kuvio 5.). Attribuuttien avulla on mahdollista myös liittää erilaisia yleisesti käytettyjä komponentteja XML-kuvaukseen. Työkalurivi on hyvä esimerkki usein toistuvasta komponentista, joka esiintyy jokaisella käyttöliittymäruudulla. Jotta voidaan välttää jokaisen ruudun kuvaustiedostoon työkalurivin kuvaaminen, mahdollistetaan sen kuvaaminen omassa tiedostossaan. Käyttöliittymämooottorin törmätessä attribuuttiin, joka ei kerro arvon tyypistä, yrittää se etsiä XML-kuvaustiedostoa attribuutin nimen perusteella. Löytyneen attribuutin ”toolbar” kohdalla yritetään etsiä toolbar.xml -tiedostoa. Tiedoston löydettyään mooottori etsii sieltä elementtiä, jonka nimi on attribuutin arvo, eli ”basic” ja korvaa alkuperäisen käyttöliittymäruudun elementin ”basic”-elementin sisällöllä. Mikäli ”basic”-elementin sisältä löydytty attribuutteja, tehdään niille samoin. Näin voidaan muodostaa monimutkaisiakin rakenteita XML-kuvaustiedostoissa.

```
<!-- merkkijono -->
<elementti>merkkijono</elementti>

<!-- totuusarvo -->
<elementti boolean="true" />

<!-- numeerinen arvo -->
<elementti int="3" />

<!-- lokalisaatio -->
<elementti languagekey="lokalisaatioTunnus" />

<!-- viitattu kenttä -->
<elementti element="TUNO" type="searchfield" />

<!-- yleinen työkalurivin kuvaus -->
<elementti toolbar="basic" />
```

Kuvio 5. Käyttöliittymäruudun XML-kuvauksessa käytetyt attribuutit.

Käyttöliittymämooottori ymmärtää myös kahden attribuutin käyttöä elementissä. Kuviossa 5 on viitattu kenttä, jossa on käytetty kahta attribuuttia. Ensimmäinen attribuutti ”element” tarkoittaa, että haetaan element.xml -tiedostosta elementtiä nimeltä ”TUNO”

(tuotenumero). Tässä kyseisessä XML-kuvaustiedostossa on kuvattu kaikki mahdolliset kentät, joita voidaan käyttää käyttöliittymämoottorissa. Kenttien kuvauksessa kerrotaan niiden tyyppi, pituus, sekä muta asetustietoja (Kuvio 6.). Tämän ansiosta, jos esimerkiksi tuotenumeron maksimipituus Merxissä muuttuisi pidemmäksi, riittää kun muutos kuvataan yhteen paikkaan. Viitatus kentän toinen attribuutti tarkoittaa ”TUNO”-elementin sisällä olevaa elementtiä, eli tässä tapauksessa ”searchfield”-elementtiä. Käyttöliittymämoottori osaa ottaa käyttöönsä vain tämän elementin sisällä olevat elementit luodessaan käyttöliittymäruutua.

```
<TUNO>
  <searchfield>
    <enableKeyEvents boolean="true"/>
    <actionActionKey>TUNO</actionActionKey>
    <ilength>20</ilength>
    <maxLength int="20"/>
    <xtype>formField</xtype>
    <fieldLabel languagekey="TUNO"/>
  </searchfield>
  <column>
    <dataIndex>TUNO</dataIndex>
    <header languagekey="TUNO"/>
  </column>
  <i5>
    <type>I5_TYPE_CHAR</type>
  </i5>
  <itemId>TUNO</itemId>
</TUNO>
```

Kuvio 6. Tuotenumeron kuvaus kenttien XML-kuvaustiedostosta.

Käyttöliittymämoottorin käytyä koko ruudun XML-kuvauksen läpi palautetaan asiakasohjelmalle moottorin prosessoima kuvaus. Kuvauksen perusteella moottorin asiakasohjelma luo tarvittavat komponentit, joilla on niille kuuluvat toiminnallisuudet. Jokaisella käyttöliittymäruudulla on tyyppi, joka määrää sen käyttötarkoituksen. Erilaisia tyyppejä ovat selailu, diagrammi, vienti, tuonti ja editori. Ruudun tyyppin perusteella asiakasohjelma osaa luoda ruudulle kuuluvat komponentit ja muokata niitä palvelinohjelmalta saadun kuvauksen perusteella. Kun ruutu on kokonaisuudessaan valmis, tuodaan se käyttöliittymään (Kuvio 7. ja 8.).

Myyntitilausten selailu

Hae Tyhjennä kentät Lisää Tallenna Poista

Myyntitilausten selailu - mxmth

Tilausnumero: Asiakkaan tilaus: Asiakas: 181 Lähetenumero: Tilaustyyppi: Tuotenumero: Myyjä: Toimitustapa: Tilakoodi: Tilauspäivä alku: Tilauspäivä loppu: Varasto:

Lisää hakuheitoja...

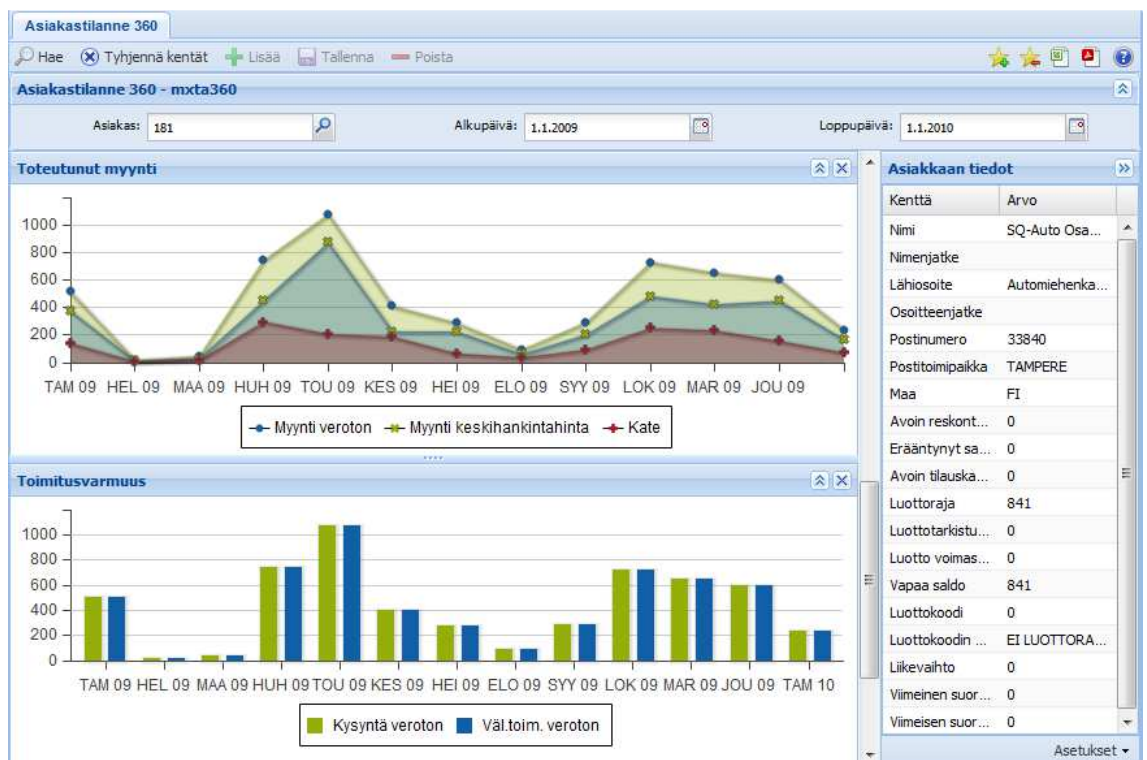
Myyntitilaukset

Tilausnumero	Kustpaik.	Toim. tapa	Toimitustapa	Til.tyyppi	Hyv/vel	Tilast päivä	Tilaskdo
3707470	90315	1	NOUDETAAN/AVHÄMTAS	LL	V	26.08.2011	08:39:04
3694978	90315	1	NOUDETAAN/AVHÄMTAS	LL	V	16.08.2011	12:22:04
3692646	30321	1	NOUDETAAN/AVHÄMTAS	LL	V	15.08.2011	08:16:47
3688948	20314	1	NOUDETAAN/AVHÄMTAS	LL	V	10.08.2011	12:41:15
3646434	30321	1	NOUDETAAN/AVHÄMTAS	LL	V	30.06.2011	14:53:32
3644509	30321	1	NOUDETAAN/AVHÄMTAS	LK	V	29.06.2011	12:17:17
3643912	30321	1	NOUDETAAN/AVHÄMTAS	LL	V	29.06.2011	08:37:13
3607153	20314	1	NOUDETAAN/AVHÄMTAS	LL	V	27.05.2011	11:48:54
3602592	20314	1	NOUDETAAN/AVHÄMTAS	VM	V	24.05.2011	13:09:12
3589584	30313	1	NOUDETAAN/AVHÄMTAS	LL	V	13.05.2011	09:03:51
3577947	30312	1	NOUDETAAN/AVHÄMTAS	LL	V	04.05.2011	08:23:19
3536591	90315	1	NOUDETAAN/AVHÄMTAS	LL	V	22.03.2011	08:22:54
3529472	30312	1	NOUDETAAN/AVHÄMTAS	LL	V	14.03.2011	10:32:29
3525314	90315	1	NOUDETAAN/AVHÄMTAS	LL	V	09.03.2011	07:44:27

Sivu 1 / 2

Näytetään 1 - 100 / 121

Kuvio 7. Käyttöliittymämoottorin luoma selailu-ruutu (myyntitilausten selailu).



Kuvio 8. Käyttöliittymämoottorin luoma diagrammi-ruutu (asiakasnäkö 360).

4.4 Aineiston tuonti- ja vientiominaisuus

Ensimmäisenä ominaisuutena joka löytyy vain käyttöliittymämoodorista, toteutettiin tiedon tuonti- ja vientiominaisuus. Tarkoituksena oli toteuttaa ominaisuus, jolla voidaan tuoda Excel-tiedostoista aineistoa Merxiin. Ominaisuus toteutettiin yhteishankkeena, jossa oli mukana useita asiakkaita. Ensimmäinen tarve aineiston tuonnissa oli aktiivi- ja passiivituotteet. Ominaisuuden avulla voidaan tuoda tai päivittää Excel-tiedostosta useita tuhansia tuotteita helposti käyttöliittymän kautta. Samaan aikaan toteutettiin myös aineiston vienti, jolla voidaan viedä tuotetietoja Excel-tiedostoon. Toiminto mahdollistaa esimerkiksi tuotteiden viennin Excel-tiedostoon, jossa niitä muokataan ja tämän jälkeen tuodaan muokattu aineisto takaisin Merxiin.

Aineiston tuonti ja vienti perustuu kahden eri tason sapluunaan (template). Sapluunat on toteutettu XML-kuvauskielellä (Liite 3.), kuten kaikki muutkin asetustiedostot käyttöliittymämoodorissa. Ylemmän tason sapluuna (SQTemplate) on Solteq Oyj:n tekemä kuvaus, jossa kerrotaan mihin tiedostoon aineisto viedään ja mitkä RPG-ohjelmat käsittelevät aineistoa. Alemman tason sapluuna on käyttäjän tekemä ja se pohjautuu aina johonkin ylemmän tason sapluunaan. Siinä liitetään Merxissä olevan tiedoston sarakkeet Excel-tiedoston sarakkeisiin. Näin aineistoa vietäessä tai tuotaessa tiedetään, että esimerkiksi Excel-tiedoston sarakkeessa A on aina tuotenumero jne. Käyttäjä luo sapluunan aina Excel-tiedoston rakenteen mukaan ja käyttää pohjana ylemmän tason sapluunaa. Käyttöliittymämoodorissa on sapluunoiden tekemiseen ja muokkaamiseen tarkoitettu ruutu (Kuvio 9.), jonka avulla niiden tekeminen on hyvin yksinkertaista. Sapluuna pitää sisällään liitettyjen sarakkeiden lisäksi myös Excel-tiedoston versiosta, siitä onko sapluuna tarkoitettu aineiston vientiä, tuontia vai molempia varten, sekä pienen vapaamuotoisen kuvauksen sen käyttötarkoituksesta.

Templaten ylläpito

Hae (X) Tyhjennä kentät + Lisää - Tallenna - Poista

Templaten ylläpito - mxpdataatemplate

Template: FMPTIH; Aktiivituotteiden

FMPTIH; Aktiivituotteiden tuonti/vienti

SQLTemplate: FMPTIH; Aktiivituotteiden

Nimi: FMPTIH; Aktiivituotteiden

ID: fmptih_Merx

Kuvaus: Merx; Aktiivituotteiden tuonti/vienti, oletuspohja.

Excel-versio: Excel 2007

Otsikointi: Merx-kuvaus

Datarivi: 2

Otsikkorivi: 1

Tuonti (Import): ☒

Vienti (Export): ☒

FMPTIH; Aktiivituotteiden tuonti/vienti

Kentän nimi	Kentän kuvaus	Pituus	Tuonnin pakkoar	Tuonnin oletusa	Isot kirj.	Excel-sarja
Pakolliset kentät						
TWPYKS	YKSIKKÖ	2 A			<input checked="" type="checkbox"/>	C
TWTURY	TUOTERYHMÄ	5 P 0				B
TWTNIS	TUOTTEEN NIMI,SUOM.	20 A			<input checked="" type="checkbox"/>	F
Valinnaiset kentät						
TWTUNO	TUOTENUMERO	20 A			<input checked="" type="checkbox"/>	A
TWHJNO	HINTAKIRJAN JÄRJESTYSNU...	5 P 0				D
TWTTUN	TOIMITTAJAN TUOTENUMERO	20 A			<input checked="" type="checkbox"/>	E
TWMAL1	MALLIKOODI 1	7 A			<input checked="" type="checkbox"/>	
TWVTU	TUOTTEEN NIMI, SUOM. OSA 2	20 A			<input checked="" type="checkbox"/>	G
TWTNML	HAKU 1	10 A			<input checked="" type="checkbox"/>	
TWLIIT	HAKU 2	20 A			<input checked="" type="checkbox"/>	
TWKTTU	HAKU 3	20 A			<input checked="" type="checkbox"/>	
TWTUTY	TUOTETYYPPI	1 A			<input checked="" type="checkbox"/>	
TWKAAT	KAATOKOODI	1 A			<input checked="" type="checkbox"/>	I
TWMYKS	MYNTIYKSIKKÖ	3 A			<input checked="" type="checkbox"/>	H
TWPAMY	PAKKAUKSESSA MYNTIYKS.	5 P 2				
TWLAMY	LAATIKOSSA MYNTIYKSIKÖITÄ	5 P 2				
TWLVMY	LAVALLA MYNTIYKSIKÖITÄ	5 P 2				
TWKÄYK	KÄYTTÄYTYMISKOODI	1 A			<input checked="" type="checkbox"/>	

Kuvio 9. Käyttöliittymäruutu sapluunoiden luomiseen ja muokkaamiseen.

5 TULOKSET JA POHDINTA

5.1 Tulokset

Nykyään yhä useampi sovellus on siirtymässä verkkoon. Rikkaat internet-sovellukset lisääntyvät ja niiden käyttö yleistyy myös yritysmaailmassa. Nyt myös Solteq Merx voi tarjota asiakkailleen selainkäyttöliittymän, jonka avulla saadaan tuotua Merxiin aivan uusia ominaisuuksia ja helpottaa asiakkaiden päivittäisiä työtehtäviä. Käyttöliittymämoottori eli webMerx täydentää omilla ominaisuuksilla Merxin nykyistä käyttöliittymää.

Käyttöliittymämoottorin yhtenä tavoitteena oli helpottaa uusien käyttöliittymäruutujen tekemistä ilman JavaScript- tai PHP-osaamista. Tavoite toteutui ja uusien käyttöliittymäruutujen luominen saatiin tehtyä mahdollisimman helpoksi XML-kuvauskielen avulla. Käyttöliittymämoottoriin voidaan luoda uusia ruutuja erittäin nopeasti ja ruutujen luominen onnistuu henkilöltä, jolle web-ohjelmointikielet eivät ole tuttuja.

Käyttöliittymämoottorin toinen päätavoite oli saada siitä mahdollisimman muokattava eri asiakkaiden tarpeiden mukaan. Asiakkailla on yleensä erilaisia vaatimuksia ja rajoituksia järjestelmän toiminnan suhteen. Muokattavuudella voidaan ottaa nämä tarpeet huomioon ja vähentää asiakaskohtaisia ohjelmaratkaisuja. Käyttöliittymämoottoriin toteutettiin usean tason asetusmahdollisuus, jolla onnistuttiin saamaan moottorista mahdollisimman mukautuva.

Jatkokehityksen kannalta käyttöliittymämoottori oli toteutettu käyttäen modulaarista ohjelmointia. MVC-arkkitehtuurin ja hyvien ohjelmointitapojen ansiosta ohjelman modulaarisuus pystyttiin säilyttämään hyvällä tasolla. Tulevaisuudessa voi kuitenkin olla tarpeen pilkkoa ohjelmaa vielä pienempiin paloihin ja näin lisätä sen modulaarisuutta.

Muutamit Merx-asiakkaat ovat ottaneet webMerx -käyttöliittymämoottorin käyttöön. Sen ensimmäinen oma ominaisuus aineiston tuonti- ja vienti on otettu erittäin positiivisesti vastaan. Ominaisuus on helpottanut asiakkaiden työtä tuotetietojen päivittämisessä, sekä uusien tuotteiden tuomisessa Merxiin.

5.2 Tuotantokäyttö

Käyttöliittymämoottori otettiin tuotantokäyttöön kahden asiakkaan toimesta keväällä 2012. Se on myös toimitettu testikäyttöön kahdelle asiakkaalle. Asiakkailla on käytössä käyttöliittymämoottori, sekä sen ensimmäinen oma tuonti ja vienti -ominaisuus. Uusi käyttöliittymä ja sen tuoma tuonti- ja vienti-ominaisuus on otettu lämpimästi vastaan. Käyttöliittymästä saadun palautteen mukaan se on helpottanut ja nopeuttanut asiakkaiden työtä. Tuonti- ja vientiominaisuuteen on tullut asiakkaiden taholta tarpeita uusille sapluunoille, kuten asiakaskohtaisten myyntihintojen hallintaan.

Tulevaisuudessa tavoitteena on saada asiakkaat miettimään lisää uusia mahdollisuuksia, joita voitaisiin toteuttaa käyttöliittymämoottorin avulla. Moottori on nyt käytössä vain osalla Merx-asiakkaista, mutta tavoitteena on saada se tuotantokäyttöön kaikille Merxin asiakkaille.

5.3 Jatkokehitys

Käyttöliittymämoottori on tuotantokäytössä muutamalla asiakkaalla ja käytännössä katsoen valmis. Sen kehitys ei kuitenkaan lopu tähän, vaan jatkuu toivottavasti vielä pitkään. Uusia ominaisuuksia ja käyttöliittymäruutuja kehitetään ja luodaan tulevaisuudessa lisää.

Ext JS -sovelluskehityksen uusi versio 4.1 julkaistaan kesän 2012 aikana. Käyttöliittymämoottorin päivittäminen käyttämään tätä uutta versiota vaatii kehitystä. Kehitykseen vaadittu aika riippuu paljon sovelluskehityksen muutoksista. Ext JS -sovelluskehitys on historiansa aikana muuttunut usein paljonkin versioiden välillä ja voidaan olettaa, että sama toistuu myös version 4.1 kohdalla. Uusi versio saattaa tuoda myös uusia mahdollisuuksia, joita voidaan hyödyntää käyttöliittymämoottorissa.

Käyttöliittymämoottorista puuttuu vielä Merxin kannalta oleellinen käyttöliittymäruututyyppi. Ylläpito-ruutujen luomista moottorin avulla ei ole vielä toteutettu. Tälle ruututyyppille on tarve tulevaisuudessa ja se tullaan kehittämään osaksi käyttöliittymämoottoria. Modulaarisen rakenteen ansiosta uusien ruutu-tyyppien kehittäminen on pyritty tekemään mahdollisimman helpoksi.

Käyttöliittymäruutujen luomiselle ei ole vielä kehitetty käyttöliittymää, vaan niiden luominen tapahtuu kirjoittamalla uusi XML-kuvaus. Tämä olisi kuitenkin mahdollista tehdä osaksi käyttöliittymämootoria, jolloin ruutujen luominen helpottuisi huomattavasti. Ominaisuus poistaisi myös tarpeen ymmärtää XML-kuvauskieltä.

Käyttöliittymämootoriin on suunniteltu pikaviestintä-ominaisuutta, jonka avulla käyttäjät voisivat viestiä keskenään ja olisi mahdollista muistuttaa käyttäjiä esimerkiksi tilauksien vahvistuksesta, sekä uusista tilauksista. Ominaisuuden toteutus on vielä suunnitteluvaiheessa, mutta saadaan toivottavasti vauhtiin jo kuluvan vuoden aikana.

LÄHTEET

Anderson M., Kosturjak V., Mullen-Schultz G. 2007. PHP: Zend for i5/OS. [pdf-dokumentti]. Luettu 22.4.2012.
<http://www.redbooks.ibm.com/redbooks/pdfs/sg247327.pdf>

Groner, L. 2011. Ext JS First Look. Briemingham, UK : Packt Publishing Ltd

IBM. WebSphere Development Studio Client for System i. 2012. [www-sivu]. Luettu 3.5.2012. <http://www-01.ibm.com/software/awdtools/wdt400/about/webfacing.html>

Mozilla Developer Network. 2012. JavaScript Overview. Luettu 13.5.2012.
https://developer.mozilla.org/en/JavaScript/Guide/JavaScript_Overview

Orchard, L.M., Pehlivanian, A. & Jones, H. 2009. Professional JavaScript frameworks : Prototype, YUI, Ext JS, Dojo and MooTools. Hoboken, NJ, USA : Wrox

Seiden A. 2010. Zend Server and Zend Framework on IBM i. [pdf-dokumentti]. Luettu 20.4.2012.
<http://www.alanseiden.com/presentation%20slides/Zend-Server-and-Zend%20Framework-for-IBM%20i.pdf>.

Sencha Ext Direct. 2012. [www-sivu]. Luettu 10.4.2012.
<http://www.sencha.com/products/extjs/extdirect>

Sencha Ext JS. 2012. [www-sivu]. Luettu 18.3.2012.
<http://www.sencha.com/products/extjs/>

Solteq Merx-tuotekortti. 2010. Solteq Oyj. [pdf-dokumentti]. Luettu 1.2.2012.

Solteq Oyj. 2012. [www-sivu]. Luettu 1.2.2012. <http://www.solteq.com/>

Solteq Oyj -vuosikertomus 2011. [pdf-dokumentti]. Luettu 5.3.2012.

The PHP Group. 2012. History of PHP. Luettu 12.5.2012.
<http://se2.php.net/manual/en/history.php.php>

The PHP Group. 2012. What is PHP?. Luettu 12.5.2012.
<http://se.php.net/manual/en/intro-what-is.php>

W3C Recommendation. 2008. Extensible Markup Language (XML) 1.0 (Fifth Edition). Luettu 12.5.2012. <http://www.w3.org/TR/REC-xml/>

Zend Framework. 2012. [www-sivu]. Luettu 16.4.2012. <http://framework.zend.com/>

LIITTEET

Liite 1. Aiheeseen liittyviä linkkejä ja lisätiedon lähteitä.



Sencha Ext JS -sovelluskehys

<http://www.sencha.com/products/extjs/>

Sencha Ext JS -sovelluskehysten latausosoite

<http://www.sencha.com/products/extjs/download/>

Sencha Ext JS -ohjelmointirajapintadokumentaatio

<http://docs.sencha.com/ext-js/4-0/>



Ext Direct

<http://www.sencha.com/products/extjs/extdirect>



Solteq Oyj

<http://www.solteq.com/>

Liite 2. Myyntitilausten selailu -käyttöliittymäruudun XML-kuvaustiedosto.

```

<?xml version="1.0" encoding="UTF-8" ?>

<root>
  <view>

    <basicInformation>
      <type>Browsing</type>
      <id>mxmth</id>
      <title languagekey="title" />
    </basicInformation>

    <searchPanel>
      <config>
        <title languagekey="searchTitle" />
      </config>

      <fields>
        <field element="TINO" type="searchfield">
          <hideTrigger boolean="true" />
        </field>
        <field element="ASTN" type="searchfield">
          <hideTrigger boolean="true" />
        </field>
        <field element="ASNO" type="searchfield" />
        <field element="LAHN" type="searchfield" />
        <field element="TITY" type="searchfield" />
        <field element="TUNO" type="searchfield" />
        <field element="MYIJ" type="searchfield" />
        <field element="TOTA" type="searchfield" />
        <field element="TILK" type="searchfield" />
        <field element="TIPVD" type="searchfield">
          <fieldLabel languagekey="TIPVA" />
          <itemId>TIPVA</itemId>
        </field>
        <field element="TIPVD" type="searchfield">
          <fieldLabel languagekey="TIPVL" />
          <itemId>TIPVL</itemId>
          <actionActionKey>TPVL</actionActionKey>
        </field>
        <field element="TOVA" type="searchfield" />
        <field element="TOPI" type="searchfield" />
        <field element="TOEH" type="searchfield" />
        <field element="MAEH" type="searchfield" />
        <field element="TIKL" type="searchfield" />
      </fields>

    </searchPanel>

    <grid>
      <config>
        <store>mxmthStore</store>
        <paging boolean="true" />
      </config>

      <columns>
        <column element="TINO" type="column" />
        <column element="KUST" type="column" />
        <column element="TOTA" type="column" />
      </columns>
    </grid>
  </view>
</root>

```

```

        <column element="TOTX" type="column" />
        <column element="TITY" type="column" />
        <column element="HVKD" type="column" />
        <column element="TIPVD" type="column" />
        <column element="TIAI" type="column" />
        <column element="ASNO" type="column" />
        <column element="ASNI" type="column" />
        <column element="TIMK" type="column" />
        <column element="VALK" type="column" />
        <column element="TOPVD" type="column" />
        <column element="TILK" type="column" />
        <column element="TASN" type="column" />
        <column element="TANI" type="column" />
        <column element="VII1" type="column" />
        <column element="VII1" type="column" >
            <itemId>VII2</itemId>
        </column>
        <column element="TOEH" type="column" />
        <column element="TOET" type="column" />
        <column element="MAEH" type="column" />
        <column element="MAET" type="column" />
        <column element="TSPR" type="column" />
        <column element="KMKD" type="column" />
        <column element="KESK" type="column" />
        <column element="TOKK" type="column" />
    </columns>

</grid>

</view>

<stores>
    <store>
        <name>mxmthStore</name>
        <controller>MyyntiController</controller>
        <method>mxmthSearchAction</method>
        <paging boolean="true" />
    </store>
</stores>
</root>

```

Liite 3. Passiivituotteiden vienti ja tuonti -ominaisuuden sapluuna.

```

<?xml version="1.0" encoding="UTF-8"?>

<root>
  <template>
    <displayName>
      FMPTTIH; Passiivituotteiden tuonti/vienti
    </displayName>

    <file>FMPTTIH</file>
    <library>*LIBL</library>
    <model>Class_PgmRmpexttih</model>
    <errorlogsupport boolean="true" />

    <dataRow int="2" />
    <titleRow int="1" />
    <titlesType>title</titlesType>
    <version>07</version>
    <rowColumn>BIRIVI</rowColumn>

    <sqdescription>
      FMPTTIH; Passiivituotteiden tuonti/vienti
    </sqdescription>

    <typeTemplate>
      <importT boolean="true" />
      <exportT boolean="true" />
    </typeTemplate>

    <dataSource>
      <exportRPG>
        <name>RMPEXTTIH</name>
        <library>*LIBL</library>
      </exportRPG>
      <importRPG>
        <model>Class_PgmRmpimttih</model>
      </importRPG>
    </dataSource>

    <fields>
      <BITTUN>
        <sqllock boolean="true" />
      </BITTUN>
      <BILVVK>
        <sqllock boolean="true" />
      </BILVVK>
      <BIPYKS>
        <value>01</value>
        <sqllock boolean="true" />
      </BIPYKS>
      <BITURY>
        <logicCheck boolean="true" />
        <sqllock boolean="true" />
        <defaultvalue>1</defaultvalue>
      </BITURY>
      <BITNIS>
        <logicCheck boolean="true" />
        <sqllock boolean="true" />
        <uppercase boolean="true"/>
      </BITNIS>
    </fields>
  </template>
</root>

```

```

        </BITNIS>
        <BITOIM>
            <logicCheck boolean="true" />
            <sqllock boolean="true" />
        </BITOIM>
    </fields>

    <preventFields></preventFields>

    <searchFields>
        <items element="TUNO" type="searchfield">
            <itemId>TUNO1</itemId>
            <fieldLabel>Tuotenumero alku</fieldLabel>
            <hideTrigger boolean="true" />
        </items>
        <items element="TUNO" type="searchfield">
            <itemId>TUNO2</itemId>
            <fieldLabel>Tuotenumero loppu</fieldLabel>
            <hideTrigger boolean="true" />
        </items>
        <items element="TURY" type="searchfield">
            <itemId>TURY1</itemId>
            <fieldLabel>Tuoteryhmä alku</fieldLabel>
            <hideTrigger boolean="true" />
        </items>
        <items element="TURY" type="searchfield">
            <itemId>TURY2</itemId>
            <fieldLabel>Tuoteryhmä loppu</fieldLabel>
            <hideTrigger boolean="true" />
        </items>
        <items element="TOIM" type="searchfield">
            <itemId>TOIM1</itemId>
            <fieldLabel>Toimittaja alku</fieldLabel>
            <hideTrigger boolean="true" />
        </items>
        <items element="TOIM" type="searchfield">
            <itemId>TOIM2</itemId>
            <fieldLabel>Toimittaja loppu</fieldLabel>
            <hideTrigger boolean="true" />
        </items>
    </searchFields>

</template>
</root>

```