



RIA -KÄYTTÖLIITTYMÄN SUUNNITTELU JA TOTEUTUS

Opinnäytetyö

Simo Jokela

Tietotekniikan koulutusohjelma
Ohjelmistotekniikka

Hyväksytty ____ . ____ . ____ _____

SAVONIA-AMMATTIKORKEAKOULU TEKNIikka KUOPIO

Koulutusohjelma

Tietotekniikan koulutusohjelma

Tekijä

Simo Jokela

Työn nimi

RIA käyttöliittymän suunnittelu ja toteutus

Työn laji

Opinnäytetyö

Päiväys

27.01.2011

Sivumäärä

27

Työn valvoja

lehtori Jussi Koistinen

Yrityksen yhdyshenkilö

Ari-Pekka Raudaskoski

Yritys

Tiivistelmä

Tämän insinööriyön aiheena oli suunnitella ja toteuttaa RIA -käyttöliittymä. Työn tavoitteena oli luoda sekä käyttöliittymän, että palvelimen toiminnallisuudet. Työssä etsittiin sopiva RIA -teknologia toteutusta varten ja päädyttiin lopulta Ext JS -teknologiaan, joka on JavaScript RIA -kirjasto. Ext JS -kirjastoon päädyttiin muun muassa siksi, että se ei vaadi mitään lisäasennuksia käyttäjältä, vaan toimii suoraan kaikissa moderneissa selaimissa. Palvelinpuolen teknologiaksi valittiin Yii framework, joka on nopea ja helppokäyttöinen php-kehys.

Sovellusta varten toteutettiin myös oma JavaScript MVC -kehys, jotta sovelluksen koodi saatiin pysymään mahdollisimman hyvin organisoituna.

Sovelluksen tietoturva oli tärkeää, koska sovelluksessa käsitellään asiakkaiden arkaluontoisia tietoja sekä raha-asioita. Koodin mahdollisimman hyvä kommentoiminen oli myös tärkeää, jotta muiden ohjelmoijien olisi myös mahdollista työskennellä koodin parissa ja että koodit olisivat myöhemmin helposti ymmärrettävissä.

Avainsanat

RIA, verkkokauppa, Yii, Ext JS, php, JavaScript, MVC

Luottamuksellisuus

salainen

SAVONIA UNIVERSITY OF APPLIED SCIENCES

Degree Programme

Information Technology

Author

Simo Jokela

Title of Project

Designing and Implementation of RIA user Interface

Type of Project

Final Project

Date

27th January 2011

Pages

27

Academic Supervisor

Mr Jussi Koistinen, Lecturer

Company Supervisor

Mr Ari-Pekka Raudaskoski

Company

Abstract

The topic of this final project was to design and execute a RIA user interface. The goal of this project was to create the user interface and the server functionalities. The technology used for the user interface was the Ext JS which is a JavaScript library and for the server the technology was Yii framework. Ext Js was chosen because it does not require any extra installations from the user and it works directly in every modern browser.

A JavaScript MVC framework was created for the application so the code could be organized as well as possible.

The security of the information was very important because the software deals with customers' personal and payment data. It was also very important to comment the code properly so that other programmers could be able to comprehend the code later.

Keywords

RIA, e-commerce, Yii, Ext JS, php, JavaScript, MVC

Confidentiality

confidential

ALKUSANAT

Haluan kiittää insinööriyön ohjaajaa Jussi Koistista työn aikana saamistani neuvoista ja ohjeista, sekä asiakas Ari-Pekka Raudaskoskea päättöyön aiheesta.

SISÄLLYS

ALKUSANAT	4
1 JOHDANTO	6
2 ASIAKKAAN TARPEET	7
3 VALITTAVAT TEKNOLOGIAT	8
3.1 Tietokanta	8
3.2 Palvelinpuolen teknologia.....	8
3.3 Yii	9
3.4 Käyttöliittymä	9
3.4.1 Käyttöliittymäteknologian valinta	10
3.4.2 JavaScript -kirjaston valinta.....	10
4 SUUNNITTELU.....	12
4.1 Suunnitteluohjelma	12
4.2 Vaiheet	13
5 TOTEUTUS.....	15
5.1 Työkalut	15
5.2 Tietoturva.....	15
5.3 Koodin kommentointi	16
5.4 Käyttöliittymä	16
5.5 Suunnitteluohjelma	16
5.5.1 Kehys	16
5.5.2 Tapahtumat	17
5.5.3 Nimiavaruus.....	17
5.5.4 Käynnistys	17
5.5.5 Suunnitteluohjelman pääkomponentit	18
5.5.6 Lopputuotteen luontilogiikka.....	21
6 POHDINTA	23
LÄHTEET	24

1 JOHDANTO

Tarkoituksena on luoda asiakkaalle käyttöliittymä RIA-tekniikkaa käyttäen. Tämä tarkoittaa sitä, että käyttöliittymän pitäisi olla dynaaminen ja helppokäyttöinen. Se ei saisi toimia kuten perinteinen verkkosivu, joka ladataan aina uudelleen kun halutaan siirtyä eri näkymään, vaan sovelluksen pitäisi toimia kuten perinteinen työpöytäsovellus.

Työn aihe rajautuu siten, että työhön kuuluu käyttöliittymän ja sen vaatiman palvelinlogiikan suunnittelu ja toteutus. Työssä käsitellään myös tietoturvaa ja hieman kommentointia, sekä miten kommentointi vaikuttaa sovelluksen jatkokehitykseen.

Tämä päättötyö on osoittain salainen ja tämä on työn julkinen versio.

2 ASIAKKAAN TARPEET

Asiakkaan tarpeena oli saada tuotteen suunnitteluovellus. Suunnitteluovelluksen avulla käyttäjä voisi suunnitella tuotteen itse omilla mitoillaan. Sovelluksessa olisi ennalta määritetty tuotteet, joista käyttäjä voisi suunnitella oman lopputuotteensa.

Sovelluksessa pitäisi olla mahdollista määrittää tuotteiden ominaisuudet, kuten tuotteen mitat ja myös monimutkaisempia ominaisuuksia, kuten tuotteessa mahdollisesti olevat reijät.

Ominaisuuksille pitäisi pystyä määrittämään raja-arvoja. Sovelluksessa pitäisi olla mahdollista esimerkiksi määrittää, että jonkin tuotepohjan leveys vähintään 20 senttimetriä ja korkeintaan 50 senttimetriä.

Sovelluksen suunnitteluohjelman käyttöliittymän pitäisi olla helppokäyttöinen ja sen toimintojen pitäisi olla mahdollisimman itsensä selittäviä. Käyttöliittymän pitäisi olla yhdenmukainen ja sulavasti toimiva, ilman turhia hidasteita.

Sovellukseen pitäisi myös suunnitella tuki reaaliaikaisesti tuotteesta kuvaa piirtävälle komponentille.

3 VALITTAVAT TEKNOLOGIAT

Käytettävän teknologian valinnassa pitää ottaa huomioon se, että sovellus jaetaan kolmeen eri osaan: tietokantaan, palvelinpuolen logiikkaan, sekä käyttöliittymään. Tämä tarkoittaa, että pitää määrittää kolme eri teknologiaa, joita käytetään sovellusta toteutettaessa. Ensiksi pitää määrittää mitä tietokantateknologiaa käytetään sovelluksen tarvitseman tiedon tallentamiseen.

Seuraavaksi pitää määrittää teknologia, jolla toteutetaan sovelluksen palvelinpuolen logiikka. Tämä tulee olemaan sovelluksen tärkein teknologia, koska tietoturvan vuoksi on tärkeää, että kaikki tärkeimmät päätökset ja rajoitukset toteutetaan palvelimelle, johon loppukäyttäjällä ei ole pääsyä.

Viimeiseksi päätetään, millä teknologialla varsinainen käyttöliittymä toteutetaan. Teknologian pitää olla erittäin suorituskykyinen, koska käyttäjien tietokoneiden suorituskyky voi olla heikko ja lisäksi asiakas haluaa, että sovellus saavuttaa mahdollisimman suuren osan mahdollisista asiakkaista.

3.1 Tietokanta

Sovelluksen käyttämäksi tietokantateknologiaksi valittiin MySQL. Tähän päädyttiin, koska MySQL on ilmainen teknologia ja se on yksi eniten käytetyistä ja yhteensopivimmista teknologioista.

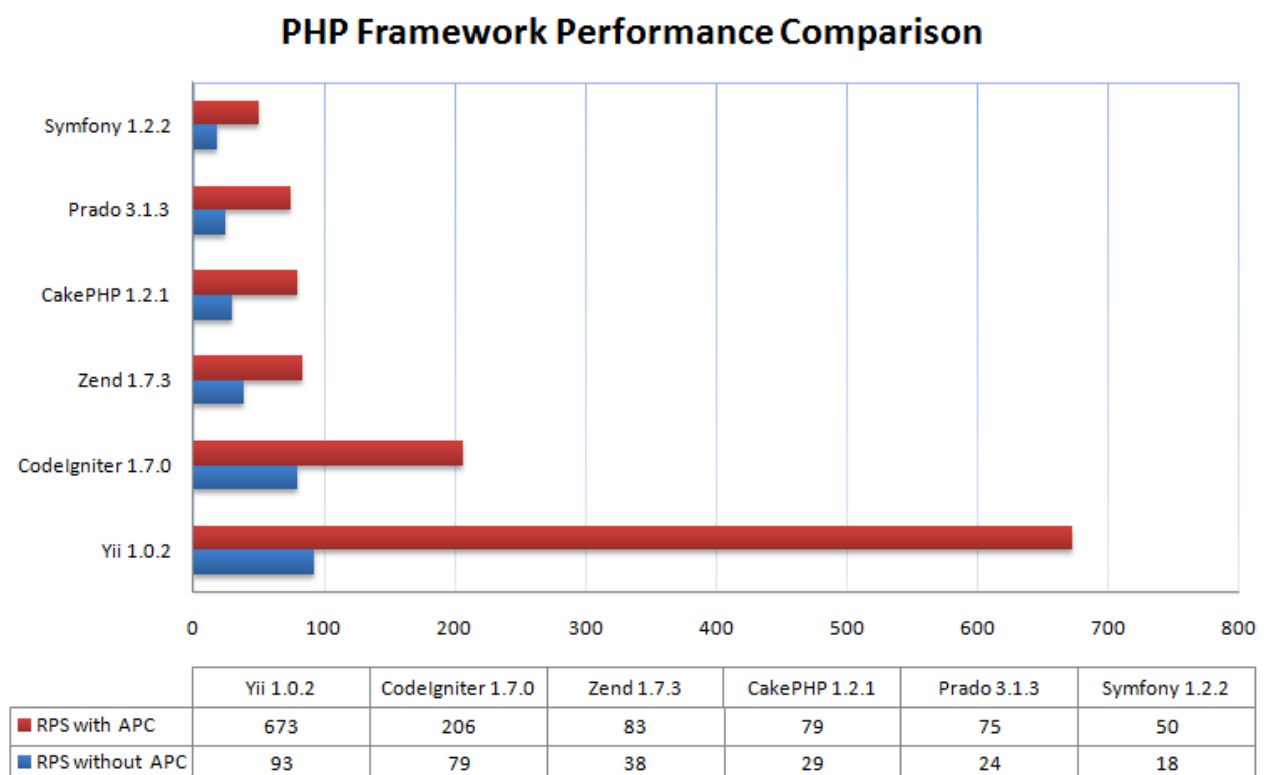
3.2 Palvelinpuolen teknologia

Palvelinpuolen teknologiaksi valittiin PHP. PHP:n valinta oli luontevaa, koska teknologia oli jo entuudestaan tuttu koulusta ja työpaikalta. Koska sovellus tulee olemaan suhteellisen laaja, sovelluksessa haluttiin käyttää MVC-suunnittelumallia. Jotta MVC-suunnittelumalli toteutuisi mahdollisimman hyvin, päätettiin projektissa käyttää jonkinlaista PHP-kehystä. PHP-kehysten valinta oli selkeä, koska minulla oli työn kautta kokemusta kehyksestä,

jonka nimi on Yii PHP framework ja työskentelen Yii:n kanssa lähes päivittäin. Lisäksi Yii:n valintaa puolsi vielä se, että se noudattaa hyvin MVC-suunnittelumallia.

3.3 Yii

Yii on MVC-suunnittelumallia toteuttava PHP -kehys, joka sisältää erittäin paljon ominaisuuksia ja sen ajatellaan olevan yksi nopeimmista PHP -kehyksistä, joita on olemassa (kuva 1). Yii tarjoaa paljon ominaisuuksia, joista voi olla asiakkaalle tulevaisuudessa hyötyä, kuten tuki tekstien kääntämiselle eri kielille.



Kuva 1. PHP -kehysten nopeusvertailu (Yii Framework)

3.4 Käyttöliittymä

Käyttöliittymän suunnittelussa piti aluksi kartoittaa, mitä erilaisia teknologioita oli mahdollista käyttää käyttöliittymän komponenttien toteuttamiseen. Ensimmäinen vaihtoehto oli toteuttaa myös käyttöliittymä Yii:llä eli PHP:ta käyttäen, mutta koska haluttiin saada käyttöliittymä mahdollisimman sulavaksi, sitä ei haluttu käyttää.

Mahdollisia teknologioita olivat myös JavaFX, Adobe Flex ja Silverlight. Näiden teknologioiden huonoin puoli on se, että ne vaativat toimiakseen jonkinlaisen kehyksen tai selainliitännäisen asennettuna käyttäjän koneelle. Ylimääräiset tiedostojen tai sovellusten asennukset web-sovelluksessa, ja varsinkin verkkokauppasovelluksessa, saattaa karkottaa mahdollisia asiakkaita. Tämä haluttiin tietenkin välttää. Lisäksi Silverlightin ikävä puoli oli vielä se, että sen yhteensopivuus kaikkien käyttöjärjestelmien ja selainten kanssa ei ollut täysin varmaa.

3.4.1 Käyttöliittymäteknologian valinta

Käyttöliittymän teknologian valinnassa tärkeimmät seikat olivat mahdollisimman hyvä yhteensopivuus internet selainten ja käyttöjärjestelmien kanssa sekä se, että käyttäjän ei tarvitsisi asentaa konelleen mitään ylimääräistä. Käyttöliittymän teknologian valinnassa otettiin huomioon myös mobiililaitteiden toimivuus teknologian kanssa, koska haluttiin saavuttaa mahdollisimman laaja asiakaskunta.

Käyttöliittymä päätettiin toteuttaa XHTML:a ja JavaScriptiä käyttäen. Näihin teknologioihin päädyttiin, koska kaikki yleisimmät selaimet tukevat molempia ilman, että käyttäjän tarvitsee asentaa mitään sovelluksia tai kirjastoja koneelle. Yleisimmät mobiililaitteet tukevat myös JavaScriptiä, mukaan lukien Applen mobiililaitteet kuten iPhone ja iPad. JavaScriptin huono puoli on se, että sillä on mahdollista ohjelmoida, ilman että henkilö oikeasti osaisi ohjelmoida (Douglas Crockford 2008). Toinen huono puoli on se, että sitä käytettäessä tulee ottaa huomioon, että jotkin sen toiminnot eivät välttämättä toimi samalla tavalla tai lainkaan kaikissa selaimissa.

3.4.2 JavaScript -kirjaston valinta

Harkitsimme että käytettäisiinkö projektissa jotain valmista JavaScript -kirjastoa vai kirjoitettaisiinko kaikki puhtaalla JavaScriptillä. Aluksi tarkoituksena oli käyttää jQuery JavaScript -kirjastoa, mutta tutustuttuani Ext JS JavaScript -kirjastoon päädyin käyttämään sitä, koska sillä oli helppo luoda hyvin toimivia käyttöliittymiä.

Ext JS on kaikilla nykyaikaisilla selaimilla toimiva rikas internetsovelluskirjasto. Sillä voidaan luoda helposti erittäin näyttäviä ja monipuolisia käyttöliittymiä. Ext JS:n yksi erittäin hyvä ominaisuus on se, että sillä on mahdollista toteuttaa varsin helposti

dynaamista dataa, koska sen käyttöliittymäkomponentteja voidaan luoda suoraan JSON-datasta. Täten on mahdollista luoda sovellus, jonka lähes kaikki komponenttien data on tietokannassa. Ext JS:n hyvä puoli on myös se, että sen on luvattu toimivan Internet Explorer 6:ssa sekä sen uudemmissa versioissa.

4 SUUNNITTELU

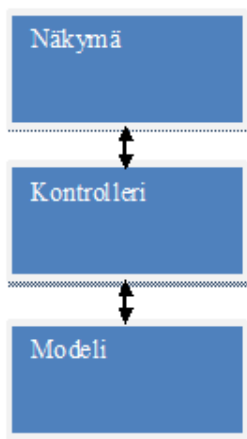
Tietokanta on salaista tietoa, se sisältää vain tietoa tuotteesta, lopputuotteesta ja ominaisuuksista.

4.1 Suunnitteluohjelma

Jokaiselle tuotteelle luodaan oma suunnitteluohjelma. Suunnitteluohjelma koostuu useasta eri vaiheesta. Näissä suunnitteluohjelman vaiheissa käyttäjä määrittää haluamansa määrittymiset tuotteen jo ennalta määritetyille ominaisuuksille.

Suunnitteluohjelma toteutetaan JavaScriptiä ja Ext JS:ää käyttäen. Ohjelman on tarkoitus käynnistyä, kun sivustolla siirrytään tuotteen omalle suunnittelusivulle.

Suunnitteluohjelmaa varten toteutetaan pienimuotoinen oma framework, joka toimii MVC-suunnittelumallia mukaillen (Kuva 7). Framework koostuu itse sovelluksesta (Application), kontrollereista (Controller), malleista (Model) ja näkymistä (View). Kuvassa 8:n näkyy esimerkki kontrollerin koodista. Kontrolleri linkittää näkymän ja modelin toisiinsa siten, että niiden ei tarvitse olla lainkaan tietoisia toisistaan (Den Odell 2009).



Kuva 2. MVC-suunnittelumallin rakenne

```

/**
 * Runs action with given name and parameters
 * @param {String} actionName, needs to be in CamelCase
 * @param {Object} parameters
 */
run: function(actionName, parameters){

    // Confirm that we have action name
    if( actionName!==undefined ){

        // Make sure that action name is capitalized.
        // Because action is always in form e.g "actionActionName" not "actionactionName"
        actionName = Ext.util.Format.capitalize(actionName);

        // If method did not receive parameters then define them to empty object
        if( parameters===undefined ){
            parameters = {};
        }

        // If action exist, run it
        if( this['action' + actionName]!==undefined ){
            this['action' + actionName](parameters);
        }
        // If not throw error
        else{
            throw new Error('No action named ' +actionName+ ' was found!');
        }
    }
    // If not, throw error
    else{
        throw new Error('Action name is undefined!');
    }
}

```

Kuva 3. Kontrollerin koodia

Kun frameworkin kriittisimmät toiminnallisuudet on toiminnassa, voidaan alkaa suunnittelemaan tuotteiden suunnitteluohjelmille kantaluokkia, joihin kuuluu esimerkiksi suunnittelija ja vaihe. Tarkoituksena on saada kantaluokista tarpeeksi yleispätevät, jotta jatkossa uusien suunnitteluohjelmien luonti olisi mahdollisimman nopeaa ja vaivatonta.

4.2 Vaiheet

Sovelluksen tuotteet sisältävät lukuisia ominaisuuksia, joten olisi erittäin epäkäytännöllistä sijoittaa kaikki tuotteen ominaisuuksien määritykset näkymään samaan aikaan näytölle. Tästä syystä tuotteen suunnittelu hajoitetaan useampaan pienempään osaan eli vaiheeseen.

Vaihe sisältää kentät, joiden avulla voidaan syöttää arvot tuotteen ominaisuuksille. Suunnitteluohjelma ei ole sidoksissa varsinaiseen tuotteeseen, vaan suunnitteluohjelman luojan on pidettävä huoli siitä, että ominaisuudet jotka ovat ohjelmassa, täsmäävät varsinaisen tietokannassa toteutetun tuotteen ominaisuuksien kanssa. Mikäli ohjelmassa olevat ominaisuudet eivät täsmää tietokannassa oleviin tuotteen ominaisuuksiin, lopputuotteen luonti palvelimen sisäisessä logiikassa ei luultavasti onnistu, koska palvelin ei tällöin saa tarvittavia tietoja luontioperaatiota varten.

Vaiheelle määritetään lisäksi suunnitteluohjelmaan informaatioteksti, joka voi sisältää myös kuvia ja linkkejä. Tämän tekstin tarkoituksena on auttaa asiakasta, jos hän ei tiedä, mitä pitäisi tehdä.

Informaatiolaatikkoon lisätään myöhemmin tuotteen kuva, joka päivittyy tuotteen määritysten vaihtuessa. Tämä tarkoittaa sitä, että suunnitteluohjelmaan on tehtävä tuki ajax-kutsuille, jotka välittävät tuotteen senhetkiset ominaisuudet palvelimelle ja palvelin sitten välittää uuden kuvan käyttöliittymään. Tämä ominaisuus ei tule vielä sovelluksen ensimmäiseen versioon.

5 TOTEUTUS

5.1 Työkalut

Ohjelmiston kehityksessä käytettiin ilmaisia kehitystyökaluja. PHP:n HTML:n ja CSS:n kirjoittamiseen käytettiin ilmaista sovellusta nimeltä Notepad++. JavaScriptin kirjoittamiseen käytettiin Aptana Studiota.

Aptana Studioon asennettiin myös JavaScript -kehityksessä lähes välttämätön koodin validointi -liitännäinen nimeltä JSLint, joka validoi kirjoitettua JavaScript -koodia ja etsii siitä lukemattomia eri virheitä, kuten puuttuvia puolipisteitä ja aaltosulkeita.

5.2 Tietoturva

Tietoturva on verkkokauppasovellukselle erittäin tärkeä asia ja erityisen tärkeä tässä sovelluksessa, koska siinä käytetään JavaScript -teknologiaa luomaan suunnitteluohjelman käyttöliittymä. JavaScript -koodi ajetaan käyttäjän selaimessa lokaalisti ja sitä on helppo muokata sovelluksen ajon aikana. Tämän vuoksi esimerkiksi suunnitteluohjelman validaatioita voidaan muokata ja palvelimelle voitaisiin lähettää tiedot tuotteesta, jota ei voida valmistaa. Tällaisten tapausten takia pitää palvelinpuolen logiikkaan luoda tarpeeksi hyvät validaatiot, Näin ollen tiedot validoidaan kahteen kertaan sekä JavaScript -puolella että palvelimen logiikassa ennen tuotteen tietojen tallentamista tietokantaan.

Yksi tärkeimmistä kohdista huomioida tietoturva on käyttäjien salasanat. Tässä sovelluksessa käyttäjien salasanat salataan käyttämällä SHA1-salausmenetelmää sekä kehittämällä oma salaus -toiminnallisuus palvelimelle. Käyttäjien salasanat täytyy salata erittäin hyvin, jotta kenenkään ei ole mahdollista tarkastella salasanoja selkeäkielisenä tietokannassa.

5.3 Koodin kommentointi

Kommentoinnilla tarkoitetaan sitä, että koodiin kirjoitetaan selitystekstejä sovelluksen kehittäjiä varten. Koodin kommentoiminen on todella tärkeää, koska ilman kunnollista kommentointia on erittäin vaikeaa saada myöhemmin selvää, mitä kirjoitetun koodin on tarkoitus tehdä, ja ohjelmoija joutuu käyttämään turhaa aikaa tämän selvittämiseen. Isommissa ohjelmistoprojekteissa, joissa on enemmän kuin yksi ohjelmoija, kommentoinnin tärkeys korostuu entisestään, koska ilman kommentointia eivät kanssaohjelmoijat välttämättä ymmärrä, mitä koodin kirjoittaja on tarkoittanut.

Tässä sovelluksessa kommentointeihin on käytetty PHPDoc -standardin mukaista kommentointi -syntaksia palvelinpuolen koodeissa ja JavaScriptin puolella JSDoc -standardin mukaista syntaksia.

5.4 Käyttöliittymä

Ennen käyttöliittymän toteutuksen aloittamista päätettiin, että sovelluksen pienin tuettu resoluutio on 1024 pikseliä kertaa 768 pikseliä. Tämä tarkoittaa sitä, että sovelluksen tärkeimpien komponenttien pitää mahtua 960 pikseliä leveään alueeseen. Sovellus toimii pienemmilläkin resoluutioilla, mutta käyttöliittymän käyttö voi olla tällöin hankalampaa.

5.5 Suunnitteluohjelma

5.5.1 Kehys

Suunnitteluohjelman toteutus aloitettiin toteuttamalla asiakkaalle oma kevyt JavaScript MVC -kehys. Kehys koostuu malleista, kontrollereista ja näytöistä.

5.5.2 Tapahtumat

Kehys hyödyntää Ext JS:n tapahtumia (Event). Tapahtumat ovat hyödyllisiä varsinkin käyttöliittymäkomponentteja luotaessa, koska on mahdotonta ennalta arvata mitä käyttäjä tekee käyttöliittymässä ja missä järjestyksessä. Tapahtumat ovat tällaisessa tapauksessa hyvä ratkaisu, koska tapahtumiin voidaan liittää tapahtumankuuntelijoita (Listener). Esimerkiksi luokalla A voi olla tapahtuma painettu. Luokka B luo tapahtumankuuntelijan luokan A tapahtumaan painettu ja yhdistää tähän tapahtumaan jotain toimintoa, kuten tallentaa tietokantaan rivin painalluksen ajankohdasta. Kun luokka B ampuu tapahtuman painettu, siitä ilmoitetaan kaikille tapahtumankuuntelijoille. Tässä tapauksessa luokka B:n tapahtumankuuntelijalle, joka suorittaa tähän tapahtumaan sidotun toiminnon ja tallentaa tietokantaan uuden rivin.

5.5.3 Nimiavaruus

Kehykselle luotiin oma nimiavaruus, jotta saataisiin ryhmiteltyä eri komponenttien nimeämiset hyvin eri aihealueiden mukaan, kuten OmaKehys.model, OmaKehys.controller, OmaKehys.view. Tällä tavalla saadaan maksimoitua esimerkiksi muuttujien, luokkien ja metodien eri nimien määrä sovelluksessa, koska esimerkiksi OmaKehys.model ja OmaKehys.controller voivat molemmat sisältää luokan nimeltä LuokkaA. Hyvin järjestelty nimiavaruus myös selkeyttää tiedostostruktuuria, jos tiedostot on ryhmitelty nimiavaruuden mukaan, kuten tässä sovelluksessa on tehty. Nimiavaruudet päätettiin luoda Ext JS:n omalla Ext.namedspace-metodilla, jota on selkeintä käyttää, kun suunnitteluohjelma on tehty Ext JS:llä (Jorge Ramon 2009).

5.5.4 Käynnistys

Suunnitteluohjelma käynnistetään luomalla ensin sovellusluokasta (AppClass) instanssi. Kun sovelluksesta on instanssi, kutsutaan sen start-metodia antamalla sille käytettävän tuotteen id, josta sovellus osaa päätellä, mitä suunnitteluohjelmaa käytetään. Start-metodi kutsuu sovelluksen initialize-metodia, joka valmistelee sovelluksen käyttöä varten ja palauttaa lopuksi totuusarvon, joka kertoo sovellukselle, onnistuiko metodin suorittaminen

ja voidaanko sovelluksen suorittamista jatkaa. Mikäli initialize-metodi palauttaa false-totuusarvon, on valmistelussa tapahtunut jokin virhe ja sovelluksen suoritus on lopetettava sekä näytettävä virheilmoitus. Jos initialize-metodin suoritus onnistui, kutsuu sovellus suunnitteluohjelman kontrollerin start-toimintoa. Sovellus myös ampuu start-tapahtuman, jotta sovellukseen liitetyt tapahtumakuuntelijat tietävät, että sovellus on nyt käynnistetty. Suunnitteluohjelman kontrollerin start-toiminto käynnistää varsinaisen suunnitteluohjelman.

5.5.5 Suunnitteluohjelman pääkomponentit

Suunnitteluohjelma koostuu kolmesta pääkomponentista:

- Designer-luokka
- Phase-luokka
- PhaseNavigator-luokka.

Sovelluksen Designer-luokan tarkoituksena on toimia itse suunnitteluohjelman pohjana. Se kokoaa kaikki suunnitteluohjelman komponentit ja sisältää paljon logiikkaa. Designer-luokkaa voidaan pitää eräänlaisena säiliönä muille suunnitteluohjelman komponenteille (kuva 9). Kun ohjelmoija luo uuden suunnitteluohjelman uudelle tuotteelle, hän perii tätä luokkaa ja tekee tarvittavat lisämääritykset, joita suunnitteluohjelma tarvitsee. Näihin määrityksiin kuuluu esimerkiksi nimi ja otsikkoteksti. Ohjelmoijan pitää myös luoda uudelle suunnitteluohjelmalle metodi, joka luo ja palauttaa ohjelman käyttämät vaiheet.

```

/**
 * Creates phases to phase list
 */
buildPhaseList: function(){
    // Get phases via method
    var phases = this.getPhases();

    for( var i=0; i<phases.length; i++ ){
        // Take single phase to a temporary variable
        var phase = phases[i];

        // Push phase to phase list
        this.phaseList.push(phase);
    }

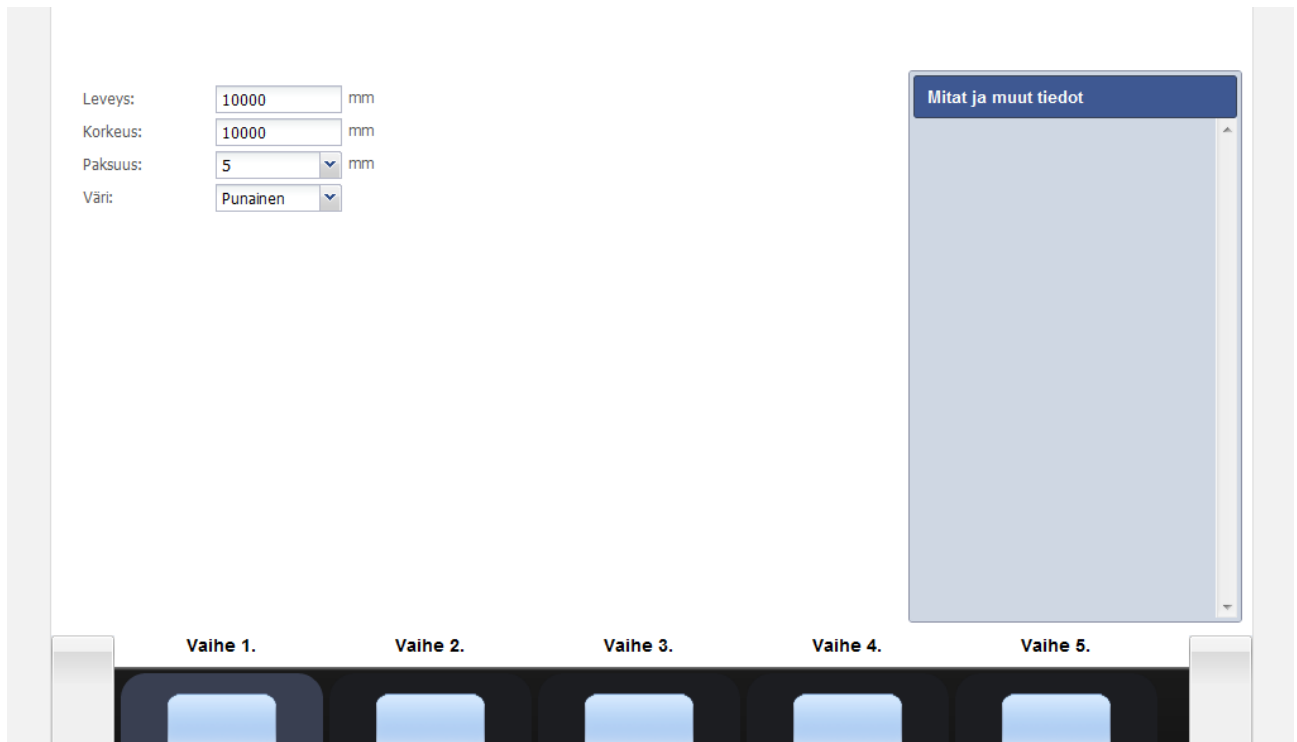
    // Create a last phase and set its title
    var lastPhase = new Mittatilaus.view.designer.LastPhase();
    //lastPhase.title = 'Vaihe ' +(i+1)+ '. ' +lastPhase.name;

    // Push last phase to last of phase list
    this.phaseList.push(lastPhase);
},

```

Kuva 4. Esimerkki vaiheiden toteutuksesta suunnitteluohjelman sisällä

Phase-luokka rakentaa suunnitteluohjelman varsinaiset vaiheet ja ohjelmoijan luodessa uutta suunnitteluohjelmaa tämä luokka näyttelee isointa osaa ohjelmointityössä. Phase-luokan tärkeimmät metodit ovat getValues ja isValid. Näistä getValues-metodin tehtävänä on palauttaa vaiheen ominaisuuksille antamat arvot. Suunnitteluohjelma odottaa metodilta palautusarvona objektia, jonka sisällä on ominaisuuden nimeä vastaavassa muuttujassa sille kuuluva arvo. isValid-metodin tarkoituksena on puolestaan validoida vaiheen arvot ja palauttaa totuusarvo, joka kertoo onko vaiheen arvot oikeanlaisia. Tätä metodia käytetään esimerkiksi kun suunnitteluohjelman kontrolleri kokoaa ohjelman arvot ja lähettää ne palvelimelle. Mikäli jonkin vaiheen isValid-metodi palauttaa false totuusarvon, ei tietoja lähetetä palvelimelle ja käyttäjää pyydetään korjaamaan virheelliset tiedot. Jos isValid-metodia ei toteuteta vaiheessa, suunnitteluohjelma olettaa, että arvot ovat oikeanlaiset ja täten metodia ei ole pakollista toteuttaa. Kuvassa 10 näkyy vaiheen esimerkkitoteutus.



Kuva 5. Esimerkki vaiheesta sovelluksessa

Sovelluksen PhaseNavigator-luokan tarkoituksena on toimia suunnitteluohjelmassa työkaluna, jolla käyttäjä voi vaihtaa vaihetta ja josta hän näkee missä vaiheessa hän on menossa. Tämä luokka on täysin suunnitteluohjelman sisäisessä logiikassa käytettävä luokka ja siksi tästä luokasta ei tarvitse välittää uusia suunnitteluohjelmia luotaessa. Navigaattori on rakennettu toimimaan täysin automaattisesti oli suunnitteluohjelmassa kuinka monta vaihetta tahansa. Jotta navigaatiopaneelin rakenne ei hajoaisi kun sovelluksessa on enemmän vaihteita, kuin mitä näytölle mahtuu, täytyi siihen toteuttaa ns. vieritystoiminto (Kuva 11). Kuvassa 12 voi nähdä navigaatiopaneelin toiminnassa.

```

/**
 * Handler for beforeselect event
 * @param {Object} dataView
 * @param {Object} node
 * @param {Array} selections
 */
onBeforeSelect: function(dataView, node, selections){
    // Get container and inner container to variable
    var container = Ext.get('phase-navigator-container');
    var innerContainer = Ext.get('phase-navigator-container-inner');

    // If inner container is wider than container then set scroll
    if( container.getWidth() < (innerContainer.getWidth()-5) ){
        // Get selected items element
        var item = Ext.get('phase-navigator-item-' +this.indexOf(node));

        // Get x offset
        var offsetX = item.getOffsetsTo(innerContainer)[0];

        // Set new scroll
        container.scrollTo('left', offsetX, true);
    }
},

```

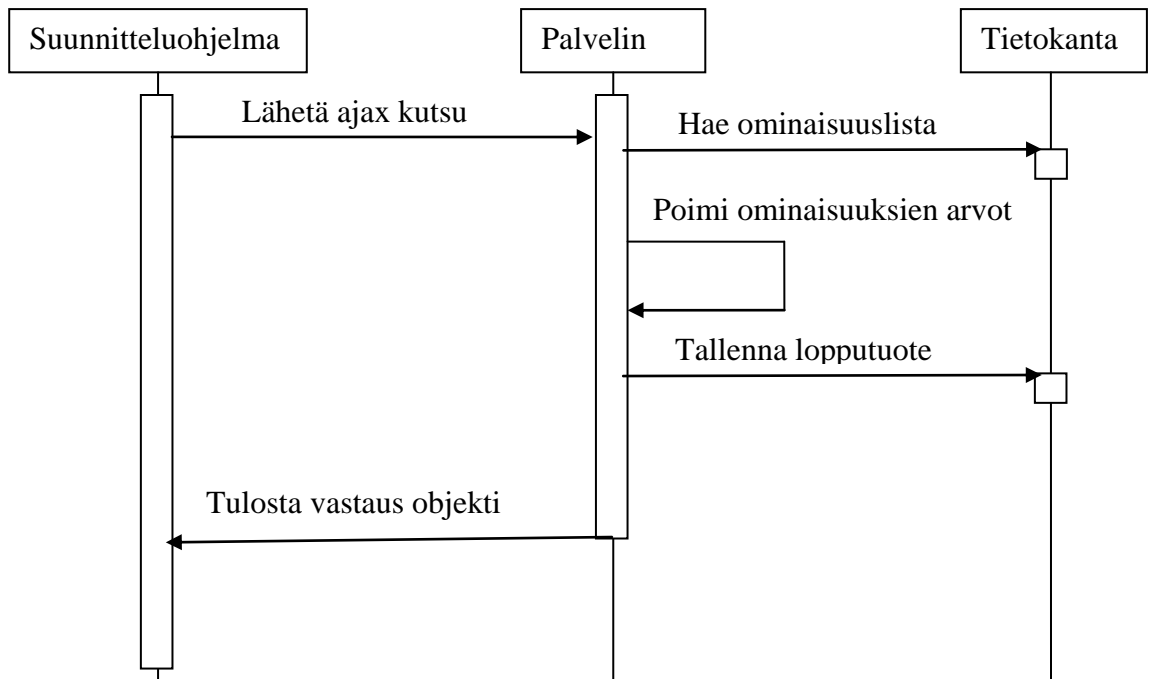
Kuva 6. Navigaationpaneelin vieritystoiminnon koodia

5.5.6 Lopputuotteen luontilogiikka

Suunnitteluohjelman lähettäessä Ajax-kutsuna palvelimelle pyynnön luoda uusi tuote lähettää suunnitteluohjelman oma kontrolleri pyynnön mukana luotavan tuotteen kaikki suunnitellut mitat joilla palvelin luo uuden lopputuotteen. Mutta kuten jo aikaisemmissa luvuissa on mainittu, on JavaScriptin tietoturvan taso heikko, ja minkä vuoksi tietokantaan tallennettavan tiedon validointi jää palvelimen vastuulle.

Palvelimelle toteutettiin logiikka, joka hakee tietokannasta listan tuotteen ominaisuuksista ja tämän jälkeen käy läpi ominaisuuslistan ja poimii suunnitteluohjelmalta saadusta datasta vain listasta löydettyjen ominaisuuksien arvot. Jos tässä vaiheessa kaikki vaaditut ominaisuudet ovat saaneet arvon, yrittää logiikka tallentaa uuden lopputuotteen tietokantaan. Logiikka luo niin sanotun vastausobjektin suunnitteluohjelmalle tulostamista varten, mihin tallennetaan tiedot siitä, onnistuiko tallentaminen, sekä viesti, joka

tulostetaan käyttöliittymässä käyttäjälle. Kuvassa 13 lopputuotteen luonnin sekvenssikaavio.



Kuva 7. Lopputuotteen luonnin sekvenssikaavio

6 POHDINTA

Työ onnistui hyvin, koska asiakkaalle saatiin toteutettua toimiva käyttöliittymä verkkokauppaa varten ja teknologiavalinnat onnistuivat hyvin.

Ext JS oli aluksi hankala teknologia opetella, koska opiskelumateriaalia oli niukasti saatavilla. Lisäksi JavaScript oli minulle suhteellisen tuntematon teknologia, mutta Ext JS:ää opiskeltuani ja kokeneempia ohjelmoijia konsultoituani alkoi Ext JS hiljalleen aueta. Nyt kun teknologia alkaa olla hallussa, on se erittäin tehokas työkalu. Suurimpana negatiivisena puolena voidaan mainita se, että Ext JS:llä jotkin yksinkertaiset asiat voi olla erittäin hankala toteuttaa, kuten alavetovalikon toteutus siten, että se ei sisällä mitään ylimääräisiä toiminnallisuuksia.

Suurimpia projektin hidasteita olivat puutteelliset määrittelydokumentit, joita ei alkuun ollut juuri ollenkaan, sekä ohjelmoijan ajanpuute loppua kohden.

Käyttöliittymän kehitystä nopeutti se, että projektissa oli käytettävissä graafikko, joka suunnitteli käyttöliittymän ulkoasun, joten ohjelmoijana minun ei tarvinnut käyttää siihen aikaa lainkaan. Sovelluksen ulkoasun kehitys ja hiominen on kirjoitushetkellä vielä käynnissä.

Jatkokehityksen kannalta olisi tärkeää palkata useampi osaava ohjelmoija, koska sovelluksen tuotantokäyttöön saattamisessa on aivan liian paljon tehtävää yhdelle ohjelmoijalle. Työtä riittää varmasti ainakin yhdelle käyttöliittymäohjelmoijalle ja yhdelle palvelinpuolen ohjelmoijalle. Lisäksi sovelluksen jatkokehitykseen olisi syytä palkata kokenut ohjelmistotalo, jonka olisi hyvä kartoittaa sovelluksen oikea työmäärä ja sen tekniset ja laitteistotarpeet.

LÄHTEET

Crockford D (2008). JavaScript: The Good Parts – Unearthing the Excellence in JavaScript. O'Reilly Media/Yahoo Press.

Odell D. (2009). Pro JavaScript RIA Techniques: Best Practices, Performance, and Presentation. Apress.

Ramon J (2009). Ext JS 3.0 Cookbook. Packt publishing.

Yii Framework. Haettu internetistä 19.12.2010. <http://www.yiiframework.com>