SAVONIA

OPINNÄYTETYÖ - **AMMATTIKORKEAKOULUTUTKINTO**
TEKNIIKAN JA LIIKENTEEN ALA

VALOKUVAAJAN TYÖKALU-PAKKI

Opinnäytetyö

TEKIJÄ/T: Mika Korhonen

SAVONIA-AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Tiivistelmä

Koulutusala Tekniikan ja liikenteen ala	
Koulutusohjelma Tietotekniikan koulutusohjelma	
Työn tekijä(t) Mika Korhonen	
Työn nimi Valokuvaajan työkalupakki	
Päiväys 7.5.2013	Sivumäärä/Liitteet 29/1
Ohjaaja(t) lehtori Keijo Kuosmanen	
Toimeksiantaja/Yhteistyökumppani(t) Hiottu Oy	

Tiivistelmä

Tämän opinnäytetyön aiheena on web-pohjainen verkkokauppa valokuvaajille myynnin tukemiseksi. Työ tehtiin Hiottu Oy:lle.

Hiotun asiakkaat tarvitsivat työkalun asiakastietojen ylläpitämiseksi sekä verkkokaupan myynnin edistämiseksi. Sovelluksesta haluttiin Internet-selaimella käytettävä. Sovellus päätettiin jakaa osioihin, joista verkkokauppamoottori käyttöliittymineen toteutetaan opinnäytetyönä.

Valokuvaajan työkalupakki toteutettiin kehittyneellä PHP-kielen työkalulla PhpED-ohjelmistokehityssovelluksella. Ohjelmointikielenä käytettiin PHP:tä ja JavaScriptin jQuery sekä jQwidgets -kirjastoja. Sovellus käyttää web-ohjelmoinnille tyypillistä HTML5-kuvauskieltä sekä CSS-tyylimäärittelyä. Tärkeät tiedot haetaan ja tallennetaan MySQL-tietokannan avulla.

Lopputuloksena saatiin valokuvaajan työkalupakin verkkokauppamoottori, jota tullaan käyttämään lopullisen sovelluksen pohjana. Projektista laadittiin projektisuunnitelma, vaatimusmäärittely sekä tekninen määrittely. Projektista laaditut suunnitelmat ovat luottamuksellisia eikä niitä ole liitetty opinnäytetyöhön. Liitteenä suunnitelma projektin aikataulusta.

Avainsanat
verkkokauppa, valokuvaajan työkalu, verkkosovellus, PHP

SAVONIA UNIVERSITY OF APPLIED SCIENCES

THESIS
Abstract

Field of Study Technology, Communication and Transport			
Degree Programme Degree Programme in Information Technology			
Author(s) Mika Korhonen			
Title of Thesis Photographer's Toolbox			
Date	7 May 2013	Pages/Appendices	29/1
Supervisor(s) Mr. Keijo Kuosmanen, Lecturer			
Client Organisation /Partners Hiottu Oy			

Abstract

The main subject of this thesis was web-based e-commerce software for photographers to improve sales. The whole project was done for Hiottu Oy.

The customers of Hiottu had noticed a demand for a program that would combine all the required software for organizing photographers' work and sales. With Hiottu's project group and supervising teacher, it was chosen to separate the upcoming software in parts. One part of the project was the user-interface combined with the e-commerce software engine and this was chosen to be produced as a bachelor's thesis.

The programming part of this thesis was done with a highly advanced programming tool PhpED. The programming language that was used was PHP and jQuery & jQWidgets which are JavaScript libraries. The program uses typical web-developing components, such as HTML 5 and CSS stylesheets. All critical data is fetched and stored with the help of a MySQL database.

At the end of the project a working version of an e-commerce software engine was built which will be used as the basis for the final application. The documentation made for this project was a project plan, requirement specifications and technical specifications. The above documents are confidential and will not be added in appendices of this thesis. The scheduling plan is available in the appendices of this thesis.

Keywords
photographer's tool, e-commerce, web application, PHP

ESIPUHE

Tämä opinnäytetyö tehtiin keväällä 2013 Hiottu Oy:lle. Työn ohjaajana toimi lehtori Keijo Kuosmanen Savonia-ammattikorkeakoulusta. Haluan kiittää häntä opinnäytetyön ohjaamisesta sekä tarkastamisesta. Lisäksi haluan kiittää Hiottu Oy:n henkilökuntaa aiheesta, sekä yhteistyöstä. Erityiskiitos kuuluu työtapoihin kouluttaneelle toimitusjohtaja Kari Lapinlammelle.

Oulussa 16.4.2013
Mika Korhonen

SISÄLTÖ

1 JOHDANTO 7

2 TERMIT JA LYHENTEET 8

3 TEKNIIKAT JA TYÖKALUT 9

3.1 Työkalut 9

3.1.1 PhpED 9

3.1.2 GanttProject 9

3.1.3 sdedit 9

3.1.4 VssConnect Client 9

3.1.5 Xampp 9

3.1.6 SQL-Yog 9

4 OPINNÄYTETYÖPROJEKTI 10

4.1 Suunnittelu 10

4.2 Määrittely 10

4.3 Toteutus 11

5 KÄYTETYT TEKNIIKAT 12

5.1 Ajax-rajapinta ja JSON 12

5.2 Kerrostettu MVC-malli 13

5.3 jQWidgets 14

6 VALOKUVAAJAN TYÖKALUPAKKI 15

6.1 Sisäänkirjautumattoman käyttäjän käyttöliittymä 15

6.2 Ylläpitäjän käyttöliittymä 15

6.2.1 Asiakkaanhallinta 16

6.2.2 Tuotteiden ylläpito 16

6.2.3 Kampanjoiden ylläpito 19

6.2.4 Tilausten ylläpito 20

6.3 Asiakkaan käyttöliittymä 20

6.3.1 Kampanjat-sivu 21

6.3.2 Tuotteet-sivu 22

6.3.3 Tilaukset-sivu 23

6.4 Tietokanta 24

7 TESTAUS 26
8 TYÖN ARVIOINTI 27
9 POHDINTA 28
LÄHTEET 29
LIITTE 1 Opinnäytetyön aikataulutus

1 JOHDANTO

Tämän opinnäytetyön aiheena on toteuttaa Internetissä toimiva valokuvaajan työkalu myynnin edistämiseksi. Toimeksianto on saatu Hiottu Oy:ltä. Hiottu Oy on vuonna 2005 perustettu yritys, joka sisältää ohjelmistosuunnitteluyksikön sekä grafiikkasuunnitteluyksikön. Hiottu Oy:n toimipisteet sijaitsevat Oulussa ja Seinäjoella.

Idea opinnäytetyön aiheesta syntyi Hiotun asiakkailta – valokuvaajilta -, joiden työn organisointi perustui lähes täysin sähköpostiin ja muistilappuihin. Asiakkaat kokivat tarpeelliseksi sovelluksen, joka yhdistäisi asiakkaiden hallinnoinnin sekä tuotteiden myynnin.

Projekti päätettiin suunnitella huolellisesti ennen työn varsinaista aloittamista. Suunnitelmassa tulee käydä ilmi sovelluksen käyttötapauskaaviot, vaatimukset sekä aikataulutus. Suunnitteluosuutta pidettiin sovelluksen kehityksen kannalta niin tärkeänä, ettei sovelluksen ohjelmointia saisi aloittaa ennen suunnittelua.

Työn ohjelmointiosuuteen kaavailtiin käytettäväksi Hiotun web-ohjelmoinnissa käytettyä PhpED-sovelluskehitintä. Ohjelmoitaessa tulisi myös käyttää Hiotun alati kehittyviä kantaluokkakirjastoja sekä kirjoittaa uusia kantaluokkia tarpeen mukaan. Työn ohjelmointikieleksi määriteltiin PHP yhdistettynä JavaScriptin jQuery-kirjastoon. Web-ohjelmointiympäristössä oletuksena pidettiin HTML 5 -kuvauskielen sekä CSS 3 tyylimääritteiden luomaa Internet-sivurakennetta.

2 TERMIT JA LYHENTEET

2.1 PHP

PHP (rekursiivinen lyhenne PHP: Hypertext Preprocessor) on laajalti käytetty avoimen lähdekoodin yleiskäyttöinen scriptikieli, joka soveltuu erinomaisesti web-kehitykseen ja johon voidaan upottaa HTML-kuvauskieltä. PHP suoritetaan palvelimella. (verkkodokumentti. PHP.NET.2013.PHP Manual).

2.2 MySQL

MySQL (Structured Query Language) on paljon käytetty tietokanta web-sovelluksissa.

2.3 HTML

HTML (Hyper Text Markup Language) on Internet-selaimen ymmärtämä kuvauskieli, jota käytetään pääasiassa verkkosivujen tekemiseen.

2.4 JavaScript

JavaScript on web-kehityksessä paljon käytetty komentokieli, jota käytetään lisäämään dynaamisuutta Internet-sivulle. JavaScript suoritetaan Internet-selaimessa.

2.4.1 jQuery

jQuery on JavaScriptin kirjasto.

2.4.2 jQWidgets

jQWidgets on jQueryyn perustuva koodikirjasto, joka sisältää useita käyttöliittymäkomponentteja.

2.5 UML

UML (Unified Modelin Language) on graafinen mallinnuskieli. UML-kaavioita käytetään paljon suunnitteluvaiheessa, kun suunnitellaan käyttötapauskaavioita.

2.6 AJAX

AJAX (Asynchronous JavaScript And Xml) on tekniikka nopeiden ja dynaamisten Web-sivujen luomiseen. AJAX mahdollistaa sivujen päivityksen asynkronisesti vaihtamalla pieniä määriä tietoa palvelimen kanssa mahdollistaen vain sivun tiettyjen osien päivittämisen päivittämättä koko sivustoa.

2.7 JSON

JSON (JavaScript Object Notation) on syntaksi tiedon vaihtamiseen ja tallentamiseen tekstimuodossa. JSON:a käytetään usein yhdessä Ajaxin kanssa (verkkosivu. JSON).

2.8 Kerrostettu MVC-malli

Kerrostettu MVC (Model View Controller) on sovellusarkkitehtuurimalli, jota tässä työssä käytettiin kehittyneesti. MVC-malli jakaa sovelluksen suorittamisen osiin, ja ohjelmistorakenteen kerrostaminen parantaa sovelluksen jatkokehityspotentiaalia.

2.9 TDD

TDD (Test-Driven Developement) on ohjelmistonkehitysmenetelmä jossa testitapaukset luodaan ennen testattavan kohteen ohjelmointia (Karjalainen, V. Testivetoinen web-sovelluskehitys).

3 TEKNIIKAT JA TYÖKALUT

Koska kyseessä on web-ympäristön ohjelmisto, sovelluksen pääkielenä käytettiin palvelimella suoritettava PHP:tä, jonka tueksi otettiin käyttöön MySQL-tietokanta ja JavaScriptin jQuery-kirjasto. Käytetyt ohjelmointikielet valittiin niiden erinomaisten web-ohjelmointiin soveltuvien ominaisuuksien vuoksi. Ohjelmiston rakenteessa noudatettiin vahvasti kerrostettua MVC-mallia, joka mahdollisti AJAX- & JSON-rajapinnan monipuolisen käytön.

3.1 Työkalut

3.1.1 PhpED

PhpED on NuSpheren toteuttama integroitu kehitysympäristö PHP, CSS, HTML, XML, XHTML sekä muille Web-ympäristön ohjelmointikielille (verkkosivu. PhpED).

3.1.2 GanttProject

GanttProject on ilmainen avoimen lähdekoodin työkalu projektin aikataulutukseen (verkkosivu. GanttProject).

3.1.3 sdedit

sdedit (Sequence Diagram Edit) on ilmainen UML-kaavioiden piirtotyökalu. Sdedittiä käytettiin suunnitteluvaiheessa käyttötapausten piirtämiseen. (verkkosivu. Sdedit.).

3.1.4 VssConnect Client

VssConnect Client on versionhallintasovellus, jota käytettiin opinnäytetyössä koodin ja dokumenttien versioinnin ylläpitämiseen (verkkosivu. VssConnect Client).

3.1.5 Xampp

Xampp on helppokäyttöinen Apache-palvelin, jonka asennuspaketti sisältää Apachen lisäksi mm. PHP:n, MySQL:n, phpMyAdmin (verkkosivu, Xampp).

3.1.6 SQL-Yog

SQL-Yog on tietokannan monipuoliseen käsittelyyn tehty maksullinen sovellus, josta on käytettävissä myös 30 päivän maksuton kokeiluversio. SQL-yogia käytettiin sovelluksen tietokantarakennetta suunniteltaessa. Sovellus mahdollistaa monimutkaisempienkin tietokantojen rakentamiseen, mitä tässä sovelluksessa ei kuitenkaan nähty tarpeellisena.

4 OPINNÄYTETYÖPROJEKTI

Opinnäytetyön tekeminen aloitettiin keväällä 2013 Kuopiossa etätyöskentelynä. Etätyöskentelyn mahdollistivat ruudun jakaminen videoneuvotteluissa ja Hiotun hankkimat laadukkaat ohjelmistot, joilla web-sovellus ohjelmoitiin.

Projekti toteutettiin perinteisellä ohjelmistojen kehitysperiaatteella: esisuunnittelu, määrittely, ohjelmistosuunnittelu, toteutus ja testaus. Projektiosuus aloitettiin projektisuunnitelmalla. Projektisuunnitelma sisälsi alustavat määritelmät, jotka kirjattiin tarkemmin määrittelyvaiheessa. Toteutusvaihe alkoi välittömästi suunnitelmien valmistuttua. Toteutusvaiheessa ohjelmoitiin suunnitelmien mukainen sovellus. Sovelluksen ohjelmointiin liittyi tekijälle vähemmän käytetyn rajapinnan opiskelu sekä ammattimaisen ohjelmistorakenteen sisäistäminen.

Testausvaiheessa sovellukseen tehtiin vain yksikkötestaus, sillä käyttöliittymän komponenttien päivitys on myöhemmin ajankohtaista. Käytöliittymän päivitytty ohjelmiston toiminta muuttuu sen verran, että tarkemmat testaukset koettiin tarpeelliseksi vasta käyttöliittymän päivitysten jälkeen.

4.1 Suunnittelu

Projekti aloitettiin suuntaa antavan projektisuunnitelman kirjoittamisella. Suunnitelmaa kirjoittaessa alkoivat hahmottua ohjelman toiminnot sekä tietokannan rakenne.
Suunnitteluvaiheessa piirrettiin käyttötapauskaaviot, laadittiin aikataulutus, projektisuunnitelma, vaatimusmäärittely sekä tekninen määrittely. Projektisuunnitelmaa päivitettiin tarvittaessa projektin edetessä. Tämä opinnäytetyödokumentti sisältää vain tiivistelmän suunnitelmista.

Suunnitteluosuuteen käytettiin aikaa noin kuukausi ennen ohjelmointivaihetta. Aikaa päätettiin varata suunnittelulle kunnolla, sillä sovellusta jatkokehitetään vielä opinnäytetyön jälkeenkin. Suunnittelun hyöty jatkokehityksen kannalta on suunnitteluvaiheessa syntyneet dokumentit, joista jatkokehittäjien on helppo ymmärtää ohjelman toimintatapoja.

Suunnitteluvaiheessa otettiin vastaan ideoita myös valokuvaajilta, joille sovellusta oltiin alun perin kohdistamassa. Näin he pääsivät vaikuttamaan sovellukseen ohjelmoitaviin toimintoihin. Suunnitteluvaiheessa päätettiin käyttää lopputestaajana näitä samoja valokuvaajia, jotta sovelluksesta saataisiin mahdollisimman miellyttävä asiakkaan näkökulmasta.
Suunnittelussa kirjattiin keskeisiä toimintoja, joita sitten määrittelyvaiheessa jalostettiin hieman pidemmälle.

4.2 Määrittely

Määrittelyvaihe alkoi projektin suunnitteluvaiheen kanssa taustaprosessina. Työn määrittelyvaiheesta kirjoitettiin vaatimusmäärittely, josta ilmenevät ohjelman toiminnot. Vaatimusmäärittelyn kirjoittaminen alkoi ensimmäisen projektisuunnitelman version valmistuttua.
Tämä menettely koettiin parhaaksi, sillä projektin vaatimukset laajenivat alkusuunnitelmista.
Määrittelyvaiheen lopuksi toteutettiin tekninen määrittely vaatimusmäärittelyn ja projektisuunnitelman pohjalta.

Kirjoitetuissa dokumenteissa käytettiin Hiotun dokumentointipohjia. Pohjat sisälsivät dokumentointiin tarvittavan rakenteen, joka mahdollistaa dokumentoinnin yhdenmukaisuuden muiden projektien dokumenttien kanssa.

Aluksi oli tarkoitus toteuttaa käyttöliittymä vain asiakkaalle sekä ylläpitäjälle. Myöhemmin ilmeni tarve verkkokaupan toiminnoille, jotka pystyttäisiin toteuttamaan hallintaliittymää tehtäessä. Toteutuksen aikaraja kuitenkin rajasi sovelluksen osa-alueet niin, että opinnäytetyö sisältää seminaariin mennessä toteutetut käyttöliittymäkomponentit sekä sovelluksen moottorin.

4.3 Toteutus

Toteutus aloitettiin vasta järjestelmällisen suunnittelun jälkeen. Suunnitteluvaihetta pidettiin erittäin tärkeänä, sillä sen tarkoitus on lyhentää ohjelmointiin käytettävää aikaa sekä vähentää tulevien hätäratkaisuiden tarvetta.

Opinnäytetyön toteutus aloitettiin aikataulutukseen tarkoin määritettyjen etappien seuraamisella. Aikataulutukseen oli merkitty toteutettavaksi ensimmäiseksi tietokannan rakentaminen.
Alustava tietokanta rakentui nopeasti huolellisesti suunniteltujen projektisuunnitelmien sekä käyttötapauskaavioiden pohjalta. Tietokantaa jouduttiin laajentamaan ohjelmiston toimintojen kehittyessä. Tietokannasta haluttiin nopeasti muunneltava ja vierasavaimia sekä tietokantataulujen välisiä relaatioita pyrittiin välttämään. Mahdolliset liitokset päätettiin tehdä MySQL-kielen *Join*-metodeilla.

Aikataulussa pysyttiin hyvin, kun työn suunnitteluvaiheessa työpäivien pituutta pidennettiin usealla tunnilla. Tämä mahdollisti toteutusvaiheen aloittamisen ennen sille määrättyä päivämäärää. Ajankäyttösuunnitelmasta poikettiin ohjelmointivaiheen alkaessa. Ensiksi rakennettiinkin ylläpitäjän käyttöliittymä toimintoineen, sillä sen uskottiin olevan nopeammin tehty. Näin työn vaikeammalle osuudelle, asiakkaan käyttöliittymälle, jäi enemmän aikaa.

Ylläpitäjän käyttöliittymään pyrittiin rakentamaan mahdollisimman tehokkaat metodit verkkokaupan hallinnointiin, jotta asiakkaan käyttöliittymää rakennettaessa tietokannassa olisi jo sisältöä testitapauksia varten.

Käyttöliittymästä haluttiin vuorovaikutteisempi, minkä vuoksi täytyi opiskella Ajax-rajapinta ja sen käyttäminen JSON-formaatin kanssa. Ajax-rajapinnan opiskelu vei aikaa, sillä opinnäytetyön tekijällä ei aikaisemmin ole ollut mahdollisuutta toteuttaa kyseisen rajapinnan avulla toimivia sivustoja.

Työn ohjelmointivaihe vei suuren osan opinnäytetyöstä myös sen kerrostettuun MVC-malliin pohjautuvan ohjelmistoarkkitehtuurin takia, jota noudatettiin tulevien muutosten sekä integrointien helpottamiseksi.

Ohjelmointimetodeihin perehdyttiin toteutuksen yhteydessä, mistä syystä työpäivien pituus vaihteli kymmenestä neljääntoista tuntiin. Työpäivien pidentäminen mahdollisti aikataulussa pysymisen viikkotasolla. Aikataulusta päätettiin myöhemmin joustaa, sillä ylläpitäjän käyttöliittymään koettiin tarpeelliseksi lisätä ominaisuuksia, mikä luonnollisesti viivästytti asiakkaan käyttöliittymän toteuttamista. Ohjelmistoa suunniteltaessa valitut jQWidgets-käyttöliittymäkirjastot koettiin keskeneräisiksi, minkä vuoksi sovelluksen moottorille päätettiin siirtää enemmän resursseja. Moottorilla tarkoitetaan sovellusta pyörittävää ohjelmistokokonaisuutta, johon käyttöliittymäkomponentit myöhemmin liitetään.

5 KÄYTETYT TEKNIIKAT

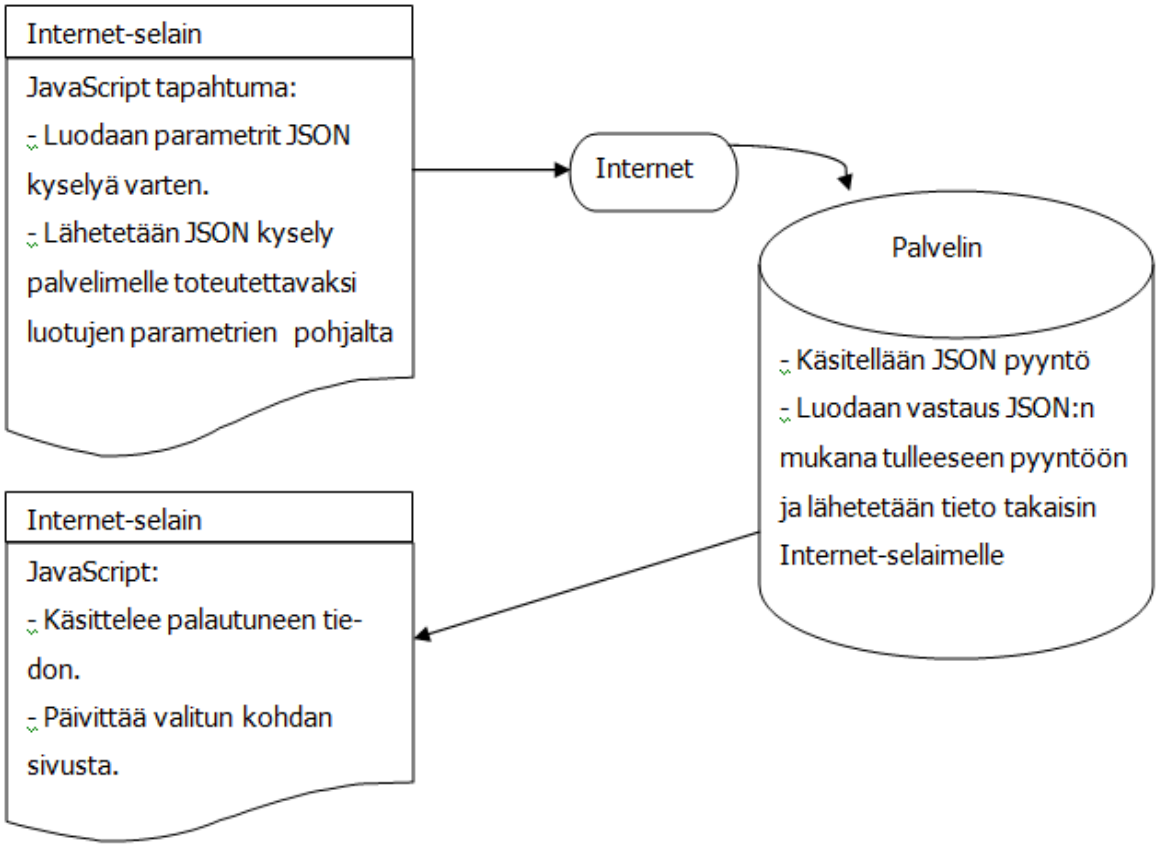
5.1 Ajax-rajapinta ja JSON

Tämä luku käsittelee työssä käytettyä AJAX-rajapintaa.
AJAX-rajapinta toimii yleensä XML-merkintäkielen kanssa, mutta kuitenkin päätettiin käyttää kevyempää ja nopeampaa JSON-tiedonsiirtomuotoa. JSON pystyy käsittelemään AJAXiin yhdistettynä samat asiat kuin XML (verkkosivu. Ajax) .

Vaikka JSON on JavaScript-perustainen tiedonsiirtomuoto, se on silti riippumaton JavaScriptistä. JSON:n JavaScript-riippumattomuuden ansiosta sitä voidaan käyttää myös monissa muissa ohjelmointikielissä.

Ajaxin toiminta perustuu Internet-selaimen ja palvelimen vuorovaikutussuhteeseen. Valokuvaajan työkalupakissa AJAX:a käytettiin yhdessä JSON:n kanssa esimerkiksi aina silloin, kun käyttäjä valitsi näytettävän sivun navigointipalkista. Valitun sivun tieto lähetetään JSON-parametrinä AJAXin kautta palvelimelle. Palvelin tulkitsee saamistaan JSON-parametreista navigointipalkista valitun sivun ja suorittaa sisällön luomisen sekä tarvittavat tietokantakyselyt. Kun palvelin on luonut käyttäjälle näytettävän sisällön Internet-selaimen ymmärtämään muotoon, lähetetään sivu näytettäväksi käyttäjälle. Internet-selain muodostaa saadusta HTML-koodista näytettävän sivuston.
AJAX mahdollistaa yllä olevan esimerkin mukaisesti sivun päivittämisen vain ohjelmallisesti määrättyihin Internet-sivun osiin.

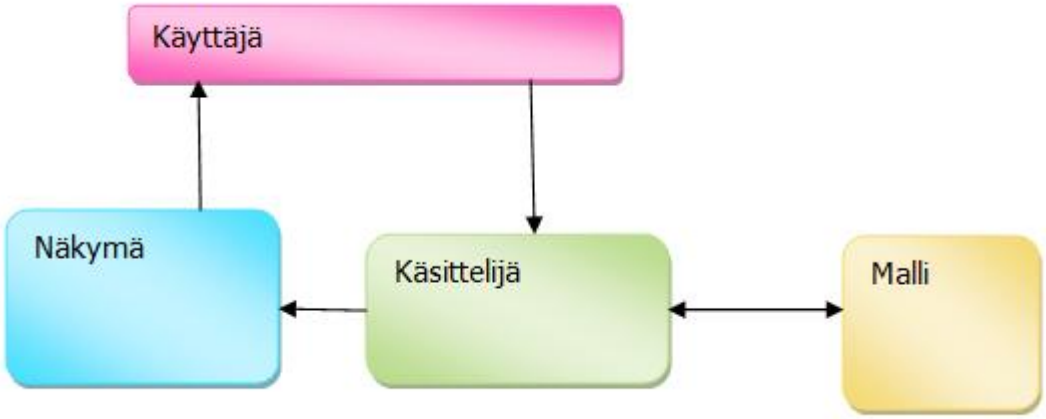
Kuvio 1 selventää AJAX:n toimintaa Internet-selaimen ja palvelimen välissä.



KUVIO 1. Ajaxin toimintaperiaate

5.2 Kerrostettu MVC-malli

MVC (Model View Controller) on kolmen komponentin yhteen liitetty ohjelmistojen suunnittelumalli (Pitt, Chris. Pro PHP MVC). Kyseistä suunnittelumallia käytetään erittäin usein olio-ohjelmoinnissa. Kuviosta 2 ilmenee näiden kolmen komponentin väliset yhteydet.



KUVIO 2. MVC-suunnittelumalli

MVC-mallilla pyritään jakamaan sovellus kolmeen eri komponenttiin (kuvio 2). Nämä komponentit ovat:

- Malli (Model), jokin osa sovellusdatasta tai loogisesta sovelluksen tilasta
- Näkymä (View), jokin osa näkyvästä käyttöliittymästä
- Käsittelijä (Controller), toimii sovittimena mallien ja näkymien välillä.

Käsittelijä lähettää komentoja Malliin. Malli käsittelee saadut komennot ja päivittää Näkymän. Päivitetty näkymä näytetään käyttäjälle.

Kun yllä kuvattua MVC-mallia kerrostetaan, voidaan jokainen kerros ajatella luokkana, jonka sisältö koostuu Mallista, Näkymästä ja Käsittelijästä. Kerrosarkkitehtuuri koostuu nousevaan järjestykseen lajitelluista kerroksista siten, että ylempi kerros käyttää aina hyväkseen alemman kerroksen palveluita. Alempi kerros ei kuitenkaan ole riippuvainen ylemmästä tasosta.
MVC-mallia käytetään usein vain laajemmissa ohjelmistoissa, sillä se jakaa ohjelman eri osa-alueet omiin komponentteihinsa, jonka hyödyt tulevat esiin ohjelmistoon tehtävien muutosten yhteydessä. Pienempiin sovelluksiin ei MVC-mallin käyttö ole suositeltua, sillä se saattaa tehdä yksinkertaisesta ohjelmasta turhan monimutkaisen.

Käytännön esimerkki MVC-mallista on aikaisemmin tässä dokumentissa mainittu AJAX-rajapinta, joka voidaan käsittää MVC-mallin mukaiseksi sovellusarkkitehtuuriksi.

5.3 jQWidgets

jQWidgets on JavaScript-ohjelmointikielen lisäkirjasto, joka keskittyy käyttöliittymäkomponentteihin. jQWidgetsistä on tarjolla 30 päivän kokeiluversio, mutta kirjaston käyttö kaupallisessa tarkoituksessa on maksullista.

jQWidgets lupaa sivuillaan olevansa suorituskykyinen ja optimoitu toimimaan useiden laitteiden, alustojen ja selainten välillä. Tukea povataan useille eri kosketus- sekä mobiililaitteille unohtamatta tietokoneen Internet-selaimia. Versiosta 2.8.2 löytyi yli 30 käyttöliittymäkomponenttia malliesimerkkeineen.

jQWidgets pohjautuu JavaScriptin yleisesti käytössä olevaan jQuery-koodikirjastoon. jQueryn etuina JavaScriptiin verrattuna on ohjelmointinopeus sekä koodin selkeämpi rakenne.

jQWidgets-koodikirjasto otettiin käyttöön sovelluksessa, ja sillä pyrittiin tekemään kaikki käyttöliittymän vaatimat dynaamiset komponentit. Näitä varten tarvittiin useita päivityksiä Hiotun tekeillä olleeseen jQWidgetsiä koskevaan kantaluokkakirjastoon. jQWidgetsistä ilmestyi uusi ohjelmistoversio lähes poikkeuksetta kantaluokan muutosten jälkeen. Osa päivityksistä toi mukanaan tarpeen tehdä suuria muutoksia koko jQWidgets-kantaluokkaan.

6 VALOKUVAAJAN TYÖKALUPAKKI

Tässä luvussa käsitellään opinnäytetyönä toteutetun Valokuvaajan työkalupakin ominaisuuksia, sekä toteutus-ratkaisuja. Valokuvaajan työkalupakki on valokuvaajalle suunniteltu web-sovellus myynnin parantamiseksi. Sovelluksen valmistuessa markkinakelpoiseksi on sitä tarkoitus käyttää myös Hiotun CMS-sovelluksen verkkokauppa-palveluna.

Valokuvaajan työkalupakki sisältää ylläpitäjän käyttöliittymän, sekä asiakkaan käyttöliittymän. Ylläpitäjän käyttöliittymä sisältää työkalut asiakkaiden, kuvien, tuotteiden, kategorioiden, kampanjoiden, sekä asiakkaiden tekemien tilausten hallintaan. Asiakkaan käyttöliittymä koostuu verkkokaupan osalta kategorioiden ja niihin kuuluvien tuotteiden selaamisesta, asiakkaalle luotujen kampanjoiden esikatselusta, tuotteiden tilaamisesta, sekä omien tilausten seurannasta. Valokuvaajan työkalupakkia voi myös käyttää kirjautumatta, tällöin näkyvillä on vain julkiseksi määriteltу sisältö. Käyttöliittymistä ja niiden toiminnoista löytyy lisätietoa myöhemmissä luvuissa.

6.1 Sisäänkirjautumattoman käyttäjän käyttöliittymä

Sovellusta suunnitellessa ilmeni tarve, että sivustoa pitää pystyä käyttämään myös sisäänkirjautumatta. Tällöin käyttäjälle näytetään ylläpitäjän luoma sivusto välilehtineen. Sivuston ylläpitäjä (valokuvaaja) on luonut Hiotun CMS-sovelluksella oman kotisivunsa, joka voi sisältää esimerkiksi etusivun, referenssit, yhteystiedot, sekä verkkokauppa-sivuston. Oletuksena on että verkkokauppa-sivusto näkyy sisäänkirjautumattomalle asiakkaalle vain julkiseksi määrätyiltä osilta. Näitä voivat olla esimerkiksi julkiset kategoriat, julkiset tuotteet tai valokuvat.

Tuotteiden tilaaminen vaatii kuitenkin sisäänkirjautumisen, ja sisäänkirjautuneille asiakkaille näytetään vain heille tarkoitetut tuotteet, kuvat ja kategoriat. Näiden lisäksi näytetään myös kaikki julkinen sisältö.

Sivuston käyttöliittymä on toteutettu jQWidgets JavaScript-kirjaston avulla, sekä AJAX- & JSON-rajapinnan avulla. Edellä mainitut rajapinnat takaavat sivun käyttöystävällisyyden päivittäen vain valitut sivuston osat. Rajapinnat mahdollistavat sen, ettei koko sivua tarvitse ladata aina uudestaan.

6.2 Ylläpitäjän käyttöliittymä

Ylläpitäjän käyttöliittymä vaatii sisäänkirjautumisen. Sisäänkirjautumistiedot tarkastetaan opinnäytetyötä tehdessä turvallisimmaksi havaitun salausalgoritmin avulla. Tämä algoritmi sisältää sisäänkirjautumisissa jo vakiintuneen ”salt”-merkkijonon, joka generoidaan jokaiselle käyttäjälle erikseen, sekä 160-bittisen ”SHA-1”-kryptausmetodin. SHA-1-metodia käytetään yhdessä saltin kanssa, jossa salt toimii kryptausalgoritmin avaimena. Molempia merkkijonoja tarvitaan, kun käyttäjätietoja tarkistetaan tietokannasta.

Ylläpitäjän käyttöliittymä sisältää toiminnot asiakkaiden, tuotteiden, kategorioiden, kampanjoiden, asiakkaan tekemien tilausten, sekä kuvien hallinnointiin.
Ylläpitäjän kirjautuessa sisälle ohjataan hänet välittömästi asiakashallinta-sivulle.

6.2.1 Asiakkaanhallinta

Asiakashallintasivulla ylläpitäjä voi lisätä ja poistaa asiakkaan, sekä muokata jo olemassa olevan asiakkaan tietoja.
Tältä sivulta pääsee myös asiakkaalle henkilökohtaisen tervehdystekstin muokkausvalikkoon.
Ylläpitäjä voi myös suositella asiakkaan hallintaliittymän kautta haluamiaan tuotteita asiakkaalle.

Asiakkaan-hallinnan tärkein ominaisuus on asiakastietojen ylläpito.
Asiakkaan-hallintapaneelissa on toiminto uuden asiakkaan lisäämiselle. Asiakkaan hallintapaneelista löytyy myös toiminnot olemassa olevien asiakastietojen muuttamiseen, sekä poistamiseen.
Muutettaessa asiakastietoja on myös mahdollisuus muuttaa henkilökohtaista alennusaikaa, sekä alennusprosenttia.

Kun ylläpitäjä haluaa lisätä asiakkaalleen henkilökohtaisen tervehdystekstin, täytyy hänen siirtyä tervehdystekstin muokkaus -toimintoon. Tervehdystekstin tarkoituksena on tuoda asiakkaalle ilmi hänen tärkeytensä asiakkaana.
Tervehdysteksti tallennetaan html-muotoon ja kyseisen asiakkaan kirjautuessa sisään näytetään hänen omalla sivullaan.

Asiakkaan hallintapaneelissa on linkki tuotteiden suosittelu-sivulle, jossa ylläpitäjä voi valita mieleisensä tuotteet ja lisätä niille suosituksensa. Suositeltavista tuotteista voi kasata paketin, esimerkiksi kehystetyn kuvan. Paketti sisältää siis kuvan ja kehykset.

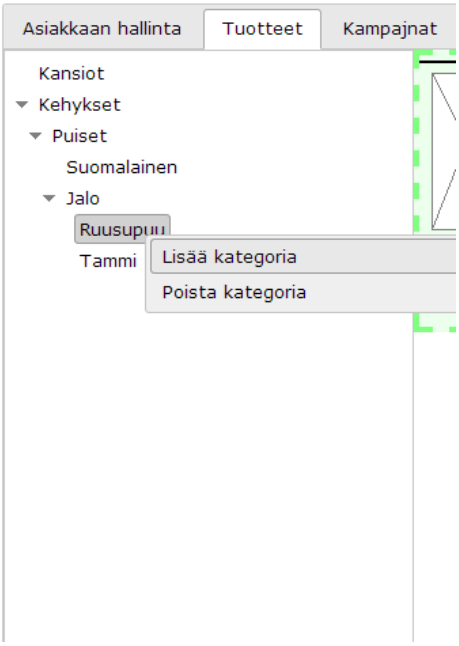
6.2.2 Tuotteiden ylläpito

Tuotteiden ylläpidolle oli aiheellista rakentaa oma välilehti toimintoineen. Tuotteiden ylläpidolle oleellisia toimintoja ovat kategorian lisäys/poisto, tuotteen lisäys/poisto sekä kuvien lisäys/poisto.

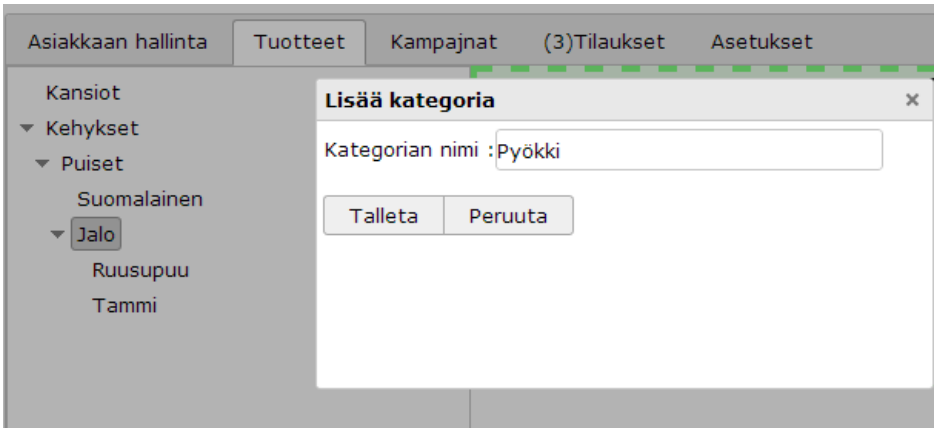
Kategorian lisäys tapahtuu klikkaamalla ”lisää kategoria”-painiketta tuotteiden hallinta-sivulla (kuvio 3). ”Lisää kategoria” painikkeen painamisen jälkeen luodaan uusi jQWidgetsin modaalinen ikkuna, joka sisältää kategorian lisäämiseen vaaditun lomakkeen (kuvio 4). Kategorian lisäys lomakkeeseen on lisätty valindaattori, joka estää lyhyemmän kuin kolmen merkin merkkijonon syöttämisen kategoriannimeksi. Modaaliseen ikkunaan avattu lomake estää muiden objektien kohdistamisen ikkunan ollessa auki. Tällä pyritään vähentämään käyttäjän tekemiä virheitä ja lisäämään helppokäyttöisyyttä.

Ylläpitäjä voi liikuttaa kategorioita haluamiinsa alikategorioihin, joka mahdollistaa kategorioiden kätevän järjestelemisen. Kategorioiden sekä alikategorioiden määrää ei ole rajattu, vaan ne luodaan dynaamisesti.

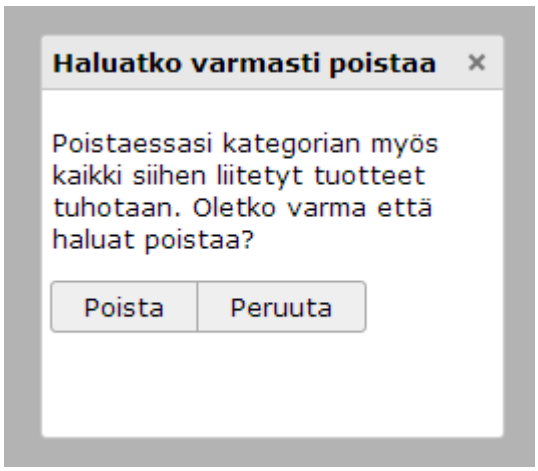
Kategoria poistetaan kategorian poisto -toiminnon avulla, jolloin kaikki kategorian alaiset tuotteet sekä mahdolliset alakategoriat tuhotaan. ”Poista kategoria” -painiketta painettaessa luodaan uusi modaalinen jQWidgets-ikkuna, jonka sisällä varmistetaan, haluaako käyttäjä todella poistaa valitun kategorian (kuvio 5). Poistaminen on mahdollista peruuttaa ”Peruuta”-painikkeen avulla sekä oikean yläkulman ruksia painamalla.



KUVIO 3. Kategoria metodit

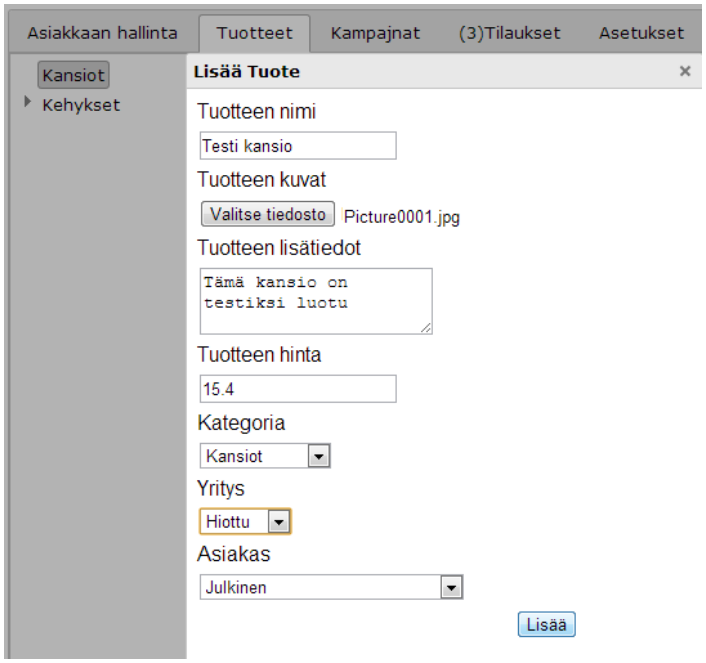


KUVIO 4. Kategorian lisäys



KUVIO 5. Kategorian poisto

Tuotteet lisätään kategorioittain. Ensiksi valitaan kategoria, johon tuote lisätään, minkä jälkeen näytetään kategoriassa sillä hetkellä olevat tuotteet, sekä tuotteiden lisäämiseen tarkoitettu toiminto. Tuotetta lisättäessä määritetään tuotteen perustiedot. Näitä ovat mm. kenelle tuote on näkyvissä, tuotteen nimi, tuotteen kuvaus sekä tuotteen kuvat (kuvio 6).
Tuote poistetaan tuotteen poisto -toiminnon avulla, jolloin tietokantataulusta shp_image poistetaan kaikki rivit, joiden iProductID on poistettavan tuotteen kanssa yhtenevä. Tietokannasta poistamisen jälkeen tuhotaan tuotteille varattu kuvakansio kategoria-id:n sekä tuote-id:n perusteella. Kuvakansion täytyy olla tyhjä, että se voidaan poistaa. Tämän takia ensiksi täytyy tyhjentää kansio poistamalla järjestyksessä kaikki tiedostot kansion sisältä.



KUVIO 6. Tuotteen lisääminen

6.2.3 Kampanjoiden ylläpito

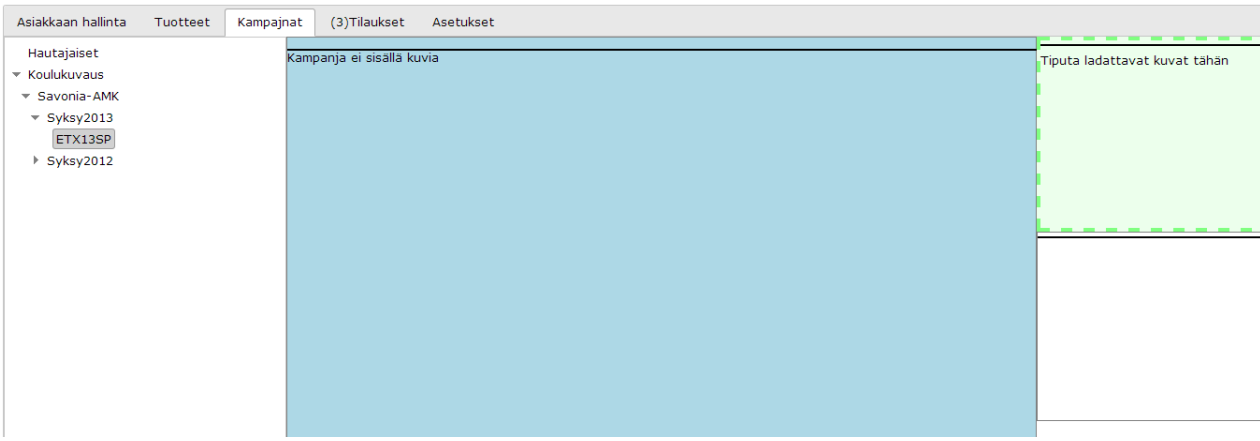
Kampanjat ovat esimerkiksi yksittäisiä kuvauskeikkoja, kuten hääkuvaukset. Kampanjoihin tallennetaan kaikki kampanjaan liittyvät kuvat. Kampanjoiden ylläpito onnistuu niille tarkoitetusta kampanjoiden hallintapaneelista. Kampanjoita voidaan luoda ja poistaa samaan tapaan kuin kategorioitakin. Kampanjaan voidaan lisätä kuvia. Jos kampanja poistetaan, tuhotaan samalla kaikki kampanjaan liitetyt kuvat.

Kampanja luodaan ”Luo Uusi” -toiminnon avulla. ”Luo Uusi” -painiketta painettaessa luodaan uusi modaalinen jQWidgets-ikkunakomponentti, jonka sisälle tulostetaan kampanjan luontiin tarvittava lomake. Kampanjaa luotaessa ylläpitäjä syöttää kampanjan nimen sekä valitsee kohdeasiakkaan alasvetovalikosta.

Ennen kuin kuvia voidaan lisätä, täytyy ylläpitäjän valita kampanja, johon kuvat lisätään.
Kuvien näkyvyys määräytyy kampanjan näkyvyyden perusteella (kenelle se on suunnattu). Kuvia voi poistaa yksitellen ja kuviin voi liittää ylläpitäjän suosituksen. Kuvat poistetaan samaan tapaan kuin tuotteetkin: Ensiksi painetaan hiiren oikealla painikkeella kuvan kohdalla ja painetaan avautuvasta valikosta poistotoimintoa. Kuvan poisto varmistetaan vielä modaalisella varmistusikkunalla.

Suosituksien tehtävä on ilmoittaa asiakkaalle ylläpitäjän mielestä parhaiten onnistuneet kuvat. Samaan tapaan kuin tuotteiden suosittelun myös kuvien suosittelun tarkoituksena on auttaa asiakasta päättämään, mitä kannattaa tilata. Asiakkaan ei ole pakko tilata yhtään ylläpitäjän suosittelemaa kuvaa, vaan asiakkaalle annetaan täysi vapaus valita kaikista tuotteista ja kuvista itselleen parhaiten sopivimmat.

Kuvio 7 on esimerkki ylläpitäjän kampanjoidenhallinta-sivusta. Vaaleansinisellä pohjalla näkyy kampanjan kuvat thumbnaileina. Jos kampanja ei sisällä kuvia sinisellä pohjalla lukee ”Kampanja ei sisällä kuvia” (kuvio 7).



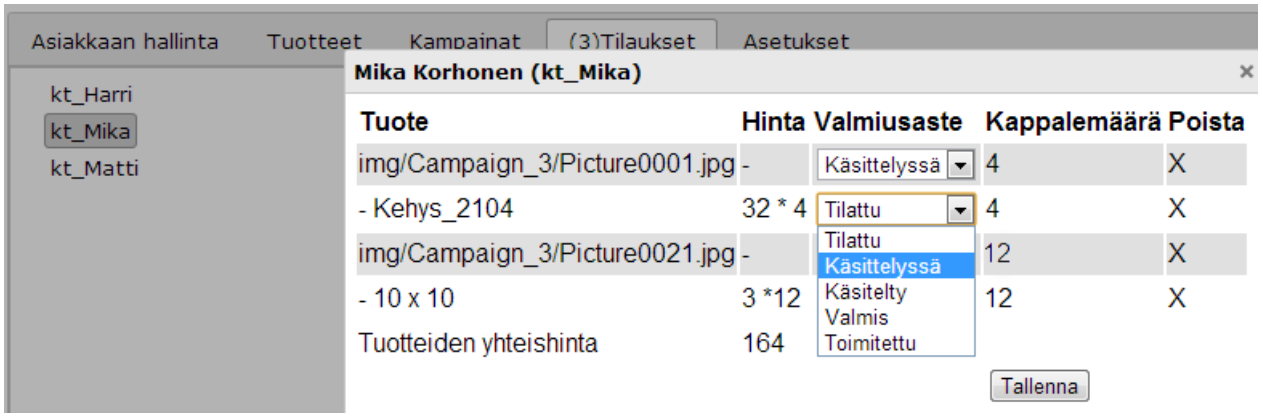
KUVIO 7. Kampanjat-näkymä

6.2.4 Tilausten ylläpito

Tilauksien ylläpidolle on oma välilehtensä. Välilehteen päivitetään sulkujen sisään käsittelemättömien tilausten määrä kokonaislukuna, esimerkiksi ”(4) Tilaukset”. Tilausten ylläpitosivulla ylläpitäjä voi muuntaa asiakkaiden tilaamien tuotteiden valmiusastetta.

Tilauksia selataan tilauksien ylläpitosivustolla listattujen asiakkaiden nimen perusteella. Ylläpitäjän täytyy valita asiakkaan tekemä tilaus, jota hän haluaa käsitellä. Ylläpitäjän valittua haluamansa tilauksen luodaan uusi modaalinen ikkuna jQWidgets-käyttöliittymäkomponentin avulla. Ikkunan sisään listataan allekkain kaikki asiakkaan tekemät tilaukset, joihin ylläpitäjä voi muuttaa valmiusastetta jokaiseen tilattuun tuotteeseen erikseen. Tilausten valmiusaste päivitetään tietokantaan tallenna-painiketta painettaessa, samalla luotu ikkuna suljetaan. Tilauksista näkyy jokaisen tilatun tuotteen hinta sekä yhteishinta (kuvio 8).
Kun tilaukset on kuitattu valmiiksi, lähetetään asiakkaalle tieto valmistuneista tuotteista.

Ylläpitäjä voi halutessaan poistaa tilauksia. Tarve tilausten poistamiseen voi olla esimerkiksi silloin, kun ylläpitäjällä ei ole varastossa asiakkaan haluamaa tuotetta. Tällä hetkellä ylläpitäjän vastuulle jää ilmoittaminen tuotteiden poistamisesta tilauksesta tai mahdollisista viivästyksistä toimituksessa.



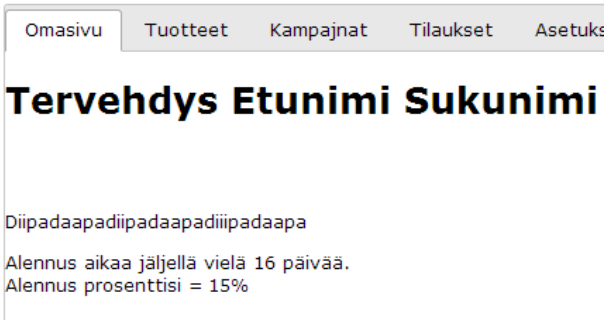
KUVIO 8. Tilausten ylläpito

6.3 Asiakkaan käyttöliittymä

Jokaisella asiakkaalla on oma henkilökohtainen ”Omasivu”, jolla näkyy ylläpitäjän määrittämä tervetuloteksti. Tervetuloteksti sisältää mahdollisten tarjousten voimassaoloajan sekä tarjouksen prosentuaalisen suuruuden (kuvio 9).

Asiakkaan ”Omasivun” tarkoitus on aktivoida asiakas tuntemaan itsensä henkilökohtaisesti huomioiduksi. Asiakkaan henkilökohtaisella huomioimisella pyritään lisäämään asiakkaan käyttömukavuutta. Omasivu voi sisältää tekstin sijasta html-koodia, joka mahdollistaa sivun personoimisen kohdehenkilölle.

Asiakkaan käyttöliittymään kuuluu oman sivun lisäksi erilliset sivut kampanjoille, tuotteille sekä tilauksille.



KUVIO 9. Tervehdysteksti

6.3.1 Kampanjat-sivu

Asiakkaan siirryttyä kampanjat-sivulle näytetään hänelle kaikki hänelle määritetyt avoimet kampanjat. Asiakkaan valittua kampanja näytetään hänelle kaikki kampanjaan kuuluvat kuvat, joista ensimmäisenä listataan valokuvaajan suosittelemat kuvat. Suositellut kuvat näytetään vihreällä kehyksellä korostettuna. Asiakkaan ei ole kuitenkaan pakko valita ostoskoriinsa suositeltuja kuvia. Lopullinen kuvien valinta jätetään asiakkaan omaksi päätökseksi.

Asiakkaan ostoskoriin siirtämiä kuvia voidaan myös poistaa ostoskorista. Ostoskori päivittää hinnan sekä ostoskorin sisällön jokaisen ostoskoriin kohdistuneen tapahtuman seurauksena. Asiakas voi poistaa kuvia ostoskorista yksitellen. Ostoskorissa näkyvä kappalemäärä pienenee yhdellä ja uusi hinta lasketaan automaattisesti ostoskorin alaosaan ”Total : € 13.20” (kuvio 10).

Sovellus ei sisällä maksumoduulien rajapintoja, jotka mahdollistaisivat tuotteiden maksamisen suoraan valokuvaajan tilille. Sovellukseen mahdollisesti lisätään ominaisuus myös suoriin maksutapahtumiin, kun moottoria aletaan integroida Hiotun julkaisujärjestelmään, Mugiin.

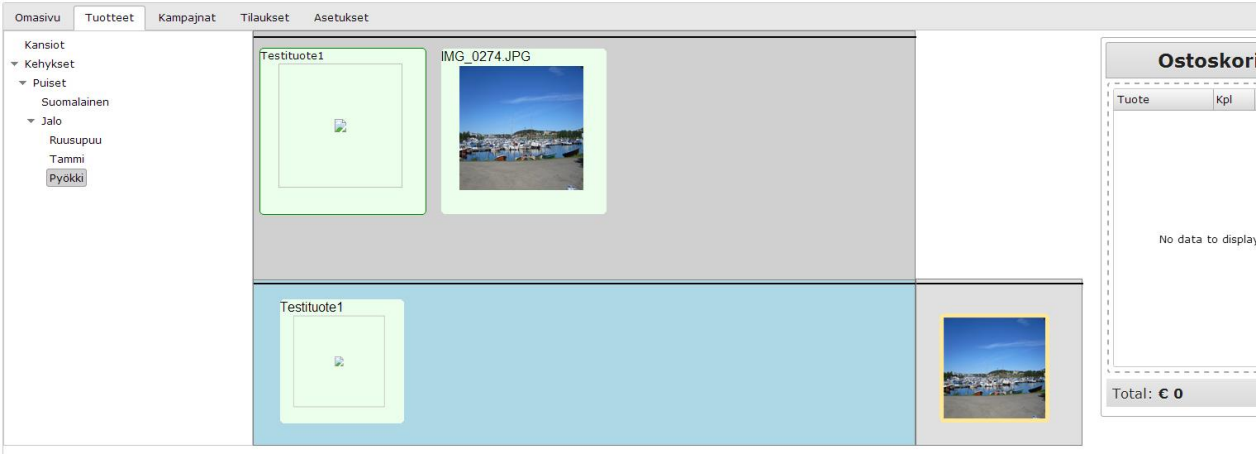
KUVIO 10. Asiakaspaneelin kampanjat-sivu

6.3.2 Tuotteet-sivu

Asiakkaan siirtyessä tuotteet-sivulle hänelle avautuu näkyviin tarkoitettujen tuotteiden kategoriat. Kategoriat listautuvat siinä järjestyksessä, johon ylläpitäjä on ne asetellut. Myöhemmin esiteltävästä tietokantarakenteesta selviää, ettei kategorian tietoja sisältävä taulu itsessään kerro, mitkä kategoriat asiakkaalle ovat näkyvissä. Näkyvyys määräytyy kategoriaan tallennettujen tuotteiden iClientID:n perusteella. Mikäli iClientID on sama kuin sisäänkirjautuneen asiakkaan tai iClientID on merkitty julkiseksi, näkyvät kaikki kyseiseen kategoriaan iCategoryID kuuluvat edellä mainitut ehdot täyttävät tuotteet.

Asiakkaan valittua kategorian, näytetään kaikki kategoriaan kuuluvat tuotteet niille tarkoitetussa sivun osassa (kuvio 11). Tuotteita voi vedellä suoraan ostoskoriin tai esimerkiksi yhdistellä toisten tuotteiden kanssa. Asiakkaalla on siis mahdollisuus yhdistää esimerkiksi ”Kehykset”-tuotekategorian ”puu kehykset”-tuote, ja lisätä sen päälle vaikka joku hänelle suunnatuista kuvista. Valmis tuote prosessoidaan näyttäytymään asiakkaalle, kuin se olisi kehystetty kuva. Valmiin yhdistelmätuotteen voi tämän jälkeen lisätä ostoskoriin. Html 5 -kielen mukanaan tuomaa DragDrop-ominaisuutta pyrittiin käyttämään mahdollisimman sulavasti, sillä sovellusta on tarkoitus pystyä käyttämään myös mobiililaitteilla. DragDrop-ominaisuudella on pyritty vähentämään comboboxien, radio-, sekä checkboxien määrää. Edellä mainitut komponentit on nimittäin koettu hankalahkoiksi käyttää mobiilimaailmassa.

Ylläpitäjä on voinut lisätä suosituksensa myös tuotteisiin, ja nämä tuotteet listautuvat ensimmäisenä ja korostetusti. Korostetun tuotteen tunnistaa tuotetta ympäröivästä tummanvihreästä reunuksesta. Korostuksen tehokeinot saattavat muuttua tulevien käyttöliittymäkomponenttien päivityksen yhteydessä.

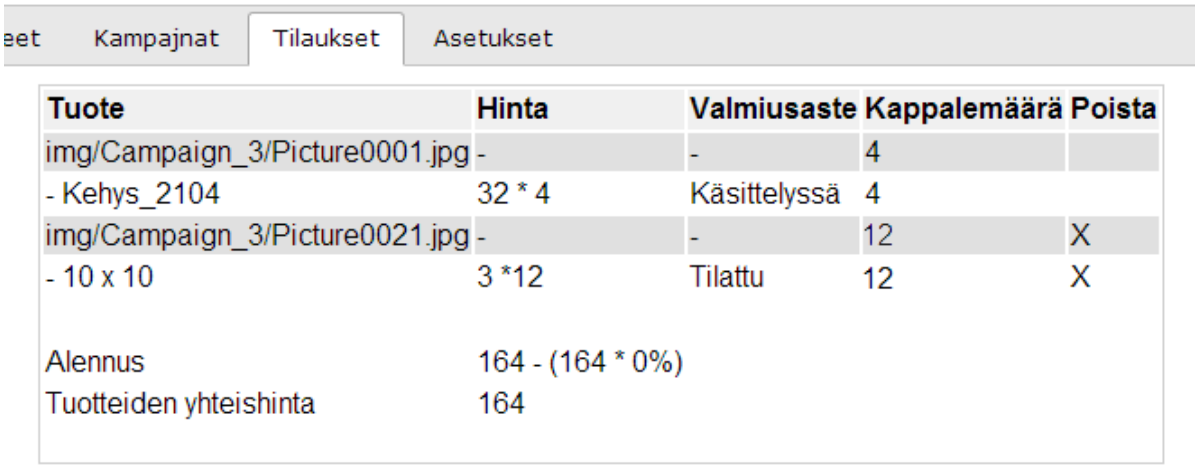


KUVIO 11. Asiakkaan tuotteet-sivu

6.3.3 Tilaukset-sivu

Asiakas voi selata tilauksiensa etenemistä tilaukset-sivulla. Tilaukset on jaettu listaksi, joista näkyy tilattujen tuotteiden kokonais-, sekä yksikköhinnat. Kokonaishintaan on laskettu erikseen tilaukseen liittyvät alennukset (kuvio 12).
Myös tuotekohtainen valmiusaste on asiakkaalle näkyvissä. Kun asiakkaan tilaamat tuotteet ovat valmiita noudettaviksi, lähetetään hänelle siitä ilmoitus.

Asiakkaalla on mahdollisuus poistaa tilaamiaan tuotteita, mikäli niiden valmiusaste on ”Tilattu”. Ominaisuudella pyritään vähentämään ylläpitäjän tilausten hallintaan käytettäviä resurrseja. Resurssien vähenemisellä tarkoitetaan tässätapauksessa aikaa, joka ylläpitäjällä kuluu kun asiakas ottaa yhteyden ylläpitoon ja haluaakin poistaa tuotteitaan tilauksesta.



KUVIO 12. Asiakkaan tilaukset

6.4 Tietokanta

Valokuvaajan työkalupakki vaatii toimiakseen tietokannan. Tietokanta sijaitsee palvelimella ja toimii tiedon tallennuspaikkana. Tässä opinnäytetyössä käytettiin MySQL-tietokantaa verkkokaupan tarvitsemien tietojen tallentamiseen. Tallennettavia tietoja ovat esimerkiksi asiakkaan yhteystiedot.

Tietokantataulujen nimeämisessä käytettiin selkeyden takia etuliitteitä, kuten com_ (common), shp_ (shop). Etuliitteillä pyritään parantamaan tietokannan hallittavuutta, sekä tuomaan selkeyttä ohjelmiston jatkokehittäjille. Tietokannan taulujen muuttujissa käytettiin Hiotulle vakiintuneita unkarilaisia notaatioita. Nimeämisistä hyvänä esimerkkinä toimii com_client-taulu, joka pitää sisällään asiakkaan yhteystiedot. Com_client-taulussa, kuten kaikissa muissakin tietokantatauluissa muuttujat on nimetty unkarilaisten notaatioiden mukaan, tämä tarkoittaa sitä että kokonaislukua sisällään pitävä pääavainmuuttuja – asiakas id – on nimetty iClientID. Kyseisen muuttujan pienellä kirjoitetusta etuliitteestä käy heti ensisilmäyksellä ilmi että kyseessä on integer- eli kokonaisluku-muuttuja. Vastaava esimerkki on merkkijonon sisältävässä muuttujassa strEmail, jossa str-etuliite tarkoittaa merkkijono muuttujaa (string).

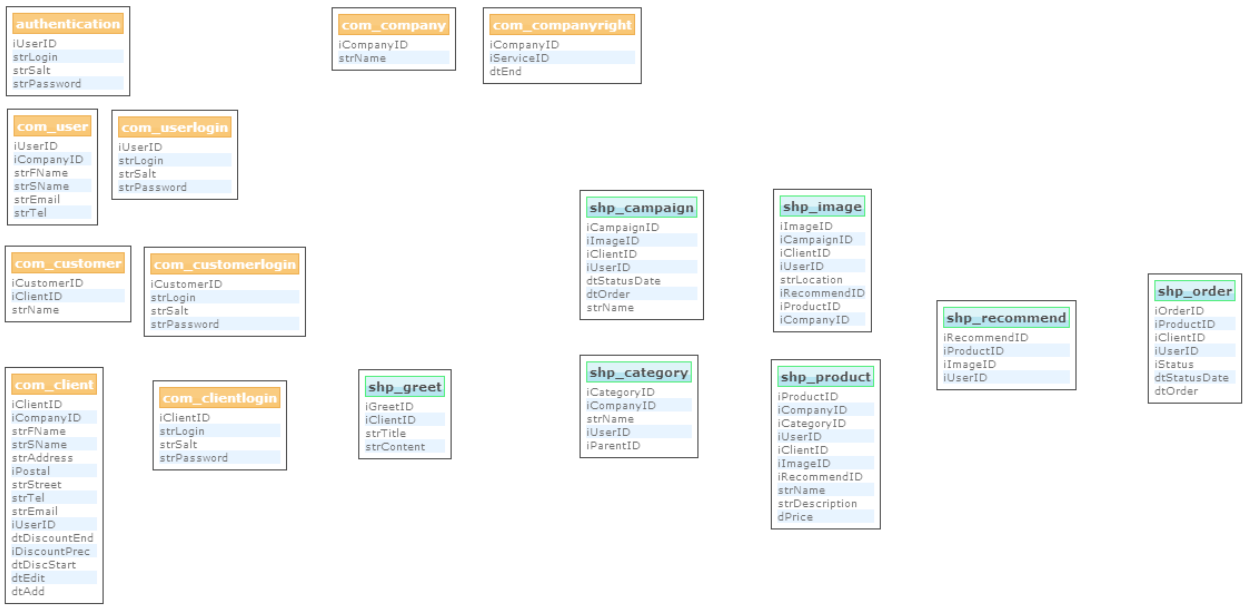
Seuraava käsitekaavio-kuva (kuvio 13) osoittaa tietokannan rakenteen. Oranssilla pohjalla olevat taulut ovat com_ -etuliitteisiä yhteisiä tauluja, joihin lukeutuvat ylläpitäjän com_user, yrityksen com_company ja asiakkaan com_client-taulut. Com_customer-taulu on lisätty tulevien ohjelmistopäivitysten takia, sillä sovellukseen ollaan myöhemmin lisäämässä vielä neljäs sisäänkirjautumismahdollisuus. Kuvasta käy ilmi että sisäänkirjautumistiedoille on oma tietokantataulu jokaista sisäänkirjautumismahdollisuutta varten.

Shp_ etuliitteen taulut on kuvaan väritetty sinivihreälle pohjalle. Shp_ etuliitteen taulut sisältävät verkkokaupan toimintojen tarvitsemia talletettavia tietoja, kuten esimerkiksi shp_category. Shp_category sisältää verkkokaupan toimintaan olennaisesti liittyvät kategorian tiedot, kuten pääavaimen iCategoryID, sekä muut kategorian tarvitsemat tiedot. Kategorian nimi tallennetaan strName-kenttään, ja mikäli kyseinen kategoria on jonkun muun kategorian alakategoria, on iParentID-kentässä jonkun jo olemassa olevan kategorian iCategoryID kokonaislukuarvo.

Tietokannan käsitekaaviosta (kuvio 13), ERD:sta (Entity-relatioinship diagram) käy ilmi, ettei tietokanta sisällä ainuttakaan fyysistä taulujenvälistä suhdetta, aikaisempien määrityksien takia. Osasyynä tietokantarelaatioiden puutteisiin oli tietokannan suunnitteluvaiheessa käytetyn XAMPP-sovelluksen mukana asentuneen phpMyAdmin-järjestelmän hankalaan relaatioiden luomiseen. Myöhemmin tietokantataulujen väliset yhteydet olisivat olleet helpohkot rakentaa SQLyog-sovelluksella. Taulujen välisiä suhteita ei kuitenkaan nähty tarpeellisena toteuttaa vielä opinnäytetyön aikana, sillä SQLyogin lisenssit saapuivat vasta siinävaiheessa, kun taulujen väliset liitokset oli hoidettu SQL-kyselyillä. Taulujen välisten yhteyksien tarve näkyy kuvaan (Kuva 13) järjesteltyjen tietokantataulujen yhdenmukaisista nimistä. Esimerkiksi com_client taulun sekä sisäänkirjautumistiedoista vastaavan com_clientlogin taulun iClientID-kentät viittaavat ohjelmiston puolella tehdyissä kyselyissä toi-

siinsa. Jälkikäteen tarkasteltuna huomaa, että jokainen com_*login-taulu tulisi sisältää myös ihan oman pääavaimensa eikä vierasavaimena toimivia i*ID-kenttiä tulisi käyttää taulun pääavaimena.

Tietokannan jokaisen taulun merkitystä ei tässä dokumentissa erikseen selvitetä, sillä niiden tarkoitusta ohjelmiston toimintaan ei tahdota kertoa julkisessa opinnäytetyössä.



KUVIO 13. Tietokantakäsitekaavio

7 TESTAUS

Sovelluksen testaamisen tavoitteena oli löytää mahdollisimman paljon ohjelmointivirheitä sekä tarkistaa suunnittelussa määrättyjen vaatimusten toimivuus. Sovelluksen testaamisesta kirjoitetaan myöhemmin valmistuvan kaupallisen sovelluksen järjestelmätestauksen yhteydessä testaussuunnitelma sekä testausraportti.

Alun perin ohjelmiston kehitykseen suunniteltiin testivetoista ohjelmistokehitystä, joka on todettu toimivan erityisen hyvin web-ohjelmointiympäristössä. Testivetoinen ohjelmistokehitys tunnetaan paremmin lyhenteestä TDD (Test Driven Developement). TDD:n ideana on luoda testitapaukset ennen testattavan kohteen määrittelemistä (Ville Karjalainen 2012, 8). Menetelmä säästää aikaa ohjelmiston suunnitteluvaiheelta, joka nopeuttaa toteutuksen aloittamista.
Ideasta kuitenkin luovuttiin, sillä toisin kuin TDD:ssä tässä projektissa ohjelmiston suunnitteluun haluttiin nähdä vaivaa ohjelmistokehityksen alkuvaiheessa.

Sovelluksen testaus toteutettiin yksikkötestauksella sovelluksen ohjelmoimisen yhteydessä. Tämän jälkeen testattiin sovelluksen valmiit osat.
Valmiita osia varten suoritettiin erillinen testaus suunnitteluvaiheessa tehtyjen käyttötapauskaavioiden pohjalta. Testitapauksia ajettiin läpi vaihe kerrallaan PhpED-sovelluskehittimen debuggertoiminnolla, mikä mahdollisti muuttujien arvojen tarkastelun ohjelman ajon aikana. Löytyneitä virheitä pyrittiin korjaamaan heti. PhpED ei mahdollista JavaScript-kielen debuggaamista. JavaScriptin testaamiseen käytettiin apuna Google Chrome -selaimen JavaScript-debuggeria.
Ohjelma testattiin siis pienissä osissa, eikä moduulien välille muodostunut suuria integrointeja. Sovelluksen osien integrointi toteutetaan myöhemmin käyttöliittymäkomponentteja käsittelevän kantaluokan valmistuttua. Integroimisen jälkeen toteutetaan myös kattavampi integrointi ja järjestelmätestaus, jotka nähtiin aiheettomiksi sovelluksen ollessa vielä kehitteillä.

Käyttöliittymän komponenttien testaamista lykättiin opinnäytetyön aikana, sillä niihin tehtiin muutoksia myös testauksen aikana. jQWidgetsistä riippumattomat käyttöliittymän osat testattiin eri selaimilla, jotta varmistuttiin niiden toimivuudesta. Ohjelma testataan tarkemmin tulevaisuudessa käyttöliittymäkomponentin vakiinnuttua.

Opinnäytetyötä koskevan osuuden lopputestauksessa testattiin ohjelmistoon valmistuneet osat ja niistä löytyneistä virheistä kirjoitettiin lista. Aikaan saadun listan avulla löytyneitä virheitä korjataan sovelluksen muiden osien toteutuksen yhteydessä.

8 TYÖN ARVIOINTI

Tavoitteena työssä oli suunnitella ja toteuttaa Hiottu Oy:n asiakkaille (valokuvaajille) sovellus myynnin tukemiseksi. Sovelluksen käytettävyyttä pyrittiin saamaan helpoksi ja nopeaksi, minkä vuoksi käyttöliittymän komponentit päätettiin toteuttaa jQWidgets JavaScript -kirjastolla. jQWidgets-kirjasto koettiin kuitenkin sisältävän paljon virheitä, minkä vuoksi kirjaston versionumero päivittyi tiuhaa tahtia. Muutosten takia sovelluksen käyttöliittymän lopullista ilmettä päätettiin lykätä myöhemmäksi ja jatkaa sovelluksen moottorin kehittämisellä.

Ohjelman käyttämisen nopeudesta huolehti AJAX-rajapinta, joka mahdollisti sivuston keskustelun palvelimen kanssa päivittäen vain halutun osan sivusta. Tämä toi mukanaan sivuston käytettävyyteen interaktiivisuutta ja lisäsi huomattavasti sovelluksen käyttömukavuutta. Rajapinnan opettelemiseen käytettiin aikaa työn ohjelmointivaiheessa.

Sovelluksesta haluttiin mahdollisimman modulaarinen. Tämän toteuttamiseksi noudatettiin Hiotun ohjelmointimetodeille tyypillistä kerrostettua MVC-mallia. MVC-mallin käyttäminen vei aikaa, sillä vastaavaa ohjelmointimenetelmää oli käytetty vain yhdellä suunnittelumallien kurssilla. Menetelmän käyttäminen sovelluksessa mahdollistaa aikaansaadun verkkokauppamoottorin käyttämisen myös esimerkiksi Hiotun julkaisujärjestelmässä.

Kokonaisuutena projekti onnistui hyvin, vaikkei aivan kaikkea suunnitelmiin kirjoitettuja toimintoja ehdittykään rakentamaan loppuun. Toimeksiantaja oli tyytyväinen työssä aikaansaatuun sovellukseen sekä suunnitteludokumentaatioon. Ohjelma sisältää vielä usean kuukauden verran korjailtavaa käyttöliittymän toiminnoissa, joiden käyttöönottamista viivästytti ennalta määrätyn jQWidgets-käyttöliittymäkomponenttien nopea versiopäivitystahti.

Opinnäytetyö oli erittäin haastava ja tarjosi mahdollisuuden toteuttaa sovellusta ammattimaisin menetelmin kehittyneillä ohjelmistoilla. Sovelluksen kehittäminen tarjosi tilaisuuden oppia käyttämään entuudestaan tuttua web-ohjelmointia laajemmin kuin aikaisemmissa työtehtävissa.

Opinnäytetyönä valmistuneen sovelluksen käyttöönottamista viivästytetään vielä, sillä siihen lisätään toimintoja ja parannetaan käytettävyyttä entuudestaan. Tarkoituksena on saada aikaiseksi täysin kaupallinen sovellus vuoden 2013 aikana.

9 POHDINTA

Jos aloittaisin projektin alusta, pyytäisin dokumentaatiot ohjelmointimetodeista sekä käytettävistä rajapinnoista heti aiheen saamisen jälkeen. Näin olisi jäänyt enemmän aikaa perehtyä käytettäviin ohjelmointi- ja nimeämisperiaatteisiin eikä opinnäytetyöhön olisi tarvinnut käyttää niin monta tuntia vuorokaudesta. Asioihin perehtyminen etukäteen olisi myös mahdollistanut tarkemman luokkarakenteen suunnittelemisen jo suunnitteluvaiheessa. Projektissa jäi harmittamaan se, etteivät kaikki käyttöliittymän komponentit ehtineet valmistua täysin toimiviksi. Tämä oli kyllä jo ennalta arvattavissa, sillä itse ohjelmointiosuudelle jäi aikaa hieman reilu kuukausi (sisältäen ohjelmointimenetelmien ja rajapintojen opettelemisen).

Vaikeimmaksi osaksi opinnäytetyötä muodostui entuudestaan vain nimeltä tuttu AJAX-rajapinta yhdistettynä JSON-tiedonsiirtomuotoon. AJAXin käyttäminen ja toiminta vaikutti aluksi erittäin mutkikkaalta yhdistettynä MVC-arkkitehtuurin mukanaan tuomiin luokkajakoihin. AJAXin toimintaan perehtyminen vei aikaa ja tiedon siirtyminen piti tutkia debuggaamalla rivi kerrallaan PHP-debuggerilla sekä Google Chrome -selaimen JavaScript debuggerin avulla.
Jälkeenpäin jäi harmittamaan, miten yksinkertainen metodi onnistuikaan vaikuttamaan niin monimutkaisen vaikealta.

Hiotun tarjoama opinnäytetyöprojekti tarjosi tilaisuuden opetella ohjelmistokehitystä ammattimaisin menetelmin. Kommentoinnin ja muuttujien nimeämisen tärkeys tuli erinomaisesti esiin työtä ohjelmoitaessa. Uskon, että sovellusta on erittäin helppo jatkokehittää tarkkojen ohjelmointisääntöjen ansiosta. Sain Hiotun toimitusjohtaja Kari Lapinlammelta koulutusta MVC-arkkitehtuurin toteuttamisesta sekä JavaScriptin debuggaamisesta selaimella. Hän myös ohjeisti sovelluksen muuttujien nimeämisessä. Opinnäytetyö perehdytti hyvin Hiotun ohjelmistonkehitysmenetelmiin, joita pidän web-ohjelmointiympäristöön erittäin kehittyneinä.

Vielä ohjelmointi vaiheessakin sovellukseen kehiteltiin jatkuvasti uusia toimintoja, joita sovellukseen ei vielä ehditty lisätä. Toimeksiantajan kanssa sovittiin, että jatkan *Valokuvaajan työkalupakki* -projektin loppuun opinnäytetyön jälkeen. Sovelluksessa käytetyt ohjelmointimenetelmät mahdollistavat verkkokauppa-sovellusmoottorin lisäämisen Hiotun julkaisujärjestelmään, Mugiin.

10 LÄHTEET

PHP.NET.2013.PHP Manual [verkkodokumentti].[viitattu 27.2.2013]. Saatavissa:
http://www.php.net/manual/en/
PhpED [verkkosivu]. [viitattu 27.2.2013]. Saatavissa:
http://www.nusphere.com/index.htm
GanttProject [verkkosivu].[viitattu 27.2.2013]. Saatavissa:
http://www.ganttproject.biz/
Sdedit [verkkosivu]. [viitattu 27.2.2013]. Saatavissa:
http://sdedit.sourceforge.net/index.html
VssConnect Client [verkkosivu]. [viitattu 27.2.2013]. Saatavissa:
http://www.vssconnect.com/index.html
Xampp [verkkosivu].[viitattu 27.2.2013]. Saatavissa:
http://www.apachefriends.org/en/xampp.html
Ajax [verkkosivu].[viitattu 15.3.2013].Saatavissa:
http://www.w3schools.com/ajax/ajax_intro.asp
JSON [verkkosivu].[viitattu 15.3.2013].Saatavissa:
http://www.w3schools.com/json/default.asp
Karjalainen, V. 2012. Testivetoinen web-sovelluskehitys, 8-9. [Pro gradu].[viitattu 15.4.2013].
Saatavissa:
http://www.cs.helsinki.fi/u/paakki/Semik12-Karjalainen.pdf
Pitt, Chris 2012. Pro PHP MVC, 10. [ekirja]: Safari Books Online. [viitattu 18.4.2013].

Opinnäytetyön aikataulutus

27.2.2013

Hiottu

http://

Project managers:

Dates:	18.2.2013 - 26.4.2013
Complete:	1%
Tasks:	14
People:	0

Tähän aikataulutukseen tulee insinöörityössä suoritettavan sovelluksen kaikki vaiheet projektin suunnittelusta, insinöörityön kirjoittamisesta, ja itse sovelluksen toteuttamisesta.

Opinnäytetyön aikataulutus

27.2.2013

2

Tasks

Nimi	Aloituspäivä	Päättymispäivä
Aikataulun laatiminen	18.2.2013	22.2.2013
Tällä viikolla tulee toteuttaa aikataulutus. Skypekeskustelun myötä Karin kanssa. Sekä laatia tiivistelmä tulevasta työstä.		
Tiivistelmän kirjoittaminen	18.2.2013	22.2.2013
Tiivistelmää kirjoitetaan sitämukaa, kun aikataulua saadaan kuntoon.		
Projektin lopullinen suunnittelu	25.2.2013	28.2.2013
Projektin suunnitteleminen.		
Usecasen suunnittelu, ennen piirtämistä	25.2.2013	26.2.2013
UseCase-piirtäminen	27.2.2013	28.2.2013
Vaatimusten määritteleminen	1.3.2013	5.3.2013
Määritellään projektin toiminnallisuus, ja vaatimukset, jotka tulee toteutua ennen insinöörityö seminaaria.		
Työn ohjelmointi	6.3.2013	10.4.2013
Ohjelmointi vaiheen aikana testataan sitämukaa, mitä tehdään.		
Tietokannan rakentaminen	6.3.2013	8.3.2013
Tällä pätkällä tulee saada valmiiksi ennalta suunniteltu tietokanta. käyttäjä kategoria tuote ostoskori		
Sisäänkirjautuminen ja asiakkaan liittymä	11.3.2013	22.3.2013
Sisäänkirjautuminen asiakkaan liittymä UI: Omasivu Tuotteet Tilaukset		
Sisäänkirjautuneen ylläpitäjän liittymä	25.3.2013	10.4.2013
Sisäänkirjautuneen ylläpitäjän UI: Kategoriat Asiakkaat Tilaukset		
Lopputestaus	11.4.2013	24.4.2013
Testaus raportti ja bugien korjaus.		

Opinnäytetyön aikataulutus

27.2.2013

Tasks

3

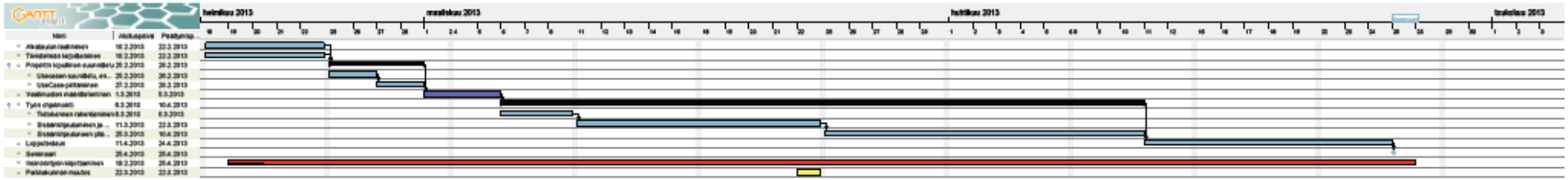
Nimi	Aloituspäivä	Päättymispäivä
Seminaari	25.4.2013	24.4.2013
DEADLINE! Tähän mennessä tulee olla 90% työstä tehty. Ja seminaari powerpoint suuniteltu ja tehty. Tänä kyseisenä päivänä on myös seminaarin esittäminen ja kypsyyskoe. Ohjelman lopputestaus ja testausraportti tehty.		
Insinöörityön kirjoittaminen	19.2.2013	25.4.2013
Opinnäytetyön kirjoittaminen. Opinnäytetyöstä palautetaan jokatoinen viikko päivitetty versio opettajalle ja projekin johtohenkilöstölle Hiotulle.		
Paikkakunnan muutos	22.3.2013	22.3.2013
Muutan kuopiosta ouluun		

Opinnäytetyön aikataulutus

27.2.2013

Gantt Kaavio

4



Opinnäytetyön aikataulutus

27.2.2013

Resurssikaavio

5