Hautamuistomerkkien suunnitteluohjelma

Suomen muotokivi Oy

Aki Korhonen

Opinnäytetyö

___. ___. _____ _____

Ammattikorkeakoulututkinto

SAVONIA-AMMATTIKORKEAKOULU **OPINNÄYTETYÖ**
Tiivistelmä

Koulutusala Tekniikan ja liikenteen ala	
Koulutusohjelma Tietotekniikan koulutusohjelma	
Työn tekijä(t) Aki Henrikki Korhonen	
Työn nimi Hautamuistomerkkien suunnitteluohjelma	
Päiväys 21.4.2011	Sivumäärä/Liitteet 51/0
Ohjaaja(t) Lehtori Keijo Kuosmanen, lehtori Jussi Koistinen	
Toimeksiantaja/Yhteistyökumppani(t) Suomen muotokivi / Esa Jääskeläinen	

Tiivistelmä

Tässä opinnäytetyössä rakennettiin hautamuistomerkkien suunnitteluohjelma Suomen Muotokivi Oy:lle. Sovelluksen avulla käyttäjä voi suunnitella hautamuistomerkin ulkoasun. Valmiista hautamuistomerkistä käyttäjä voi tehdä tilauksen. Tilauksesta saatavan kuvan perusteella ammattilaiset voivat suunnitella oikean hautamuistomerkin CAD-ohjelmien avulla.

Hautamuistomerkkien ulkoasun suunnittelua varten järjestelmään rakennettiin suunnitteluedeitori Adobe Flash-kehitysympäristön avulla. Suunnitteluedeitori upotettiin nettisivuille, jotka sijaitsivat asiakkaan palvelimella. Tietokannan ja suunnitteluedeitorin välillä siirreltiin dataa XML:n ja PHP:n avulla. MySQL-ohjelmistoa käytettiin tietokantana järjestelmän toteutuksessa.

Järjestelmän hallintaa varten luotiin oma web-pohjainen hallintapaneeli. Hallintapaneelin kautta pystyy hallitsemaan järjestelmän tuotekatalogeja, ylläpitäjiä ja tarkastelemaan asiakkaiden tekemiä tilauksia.

Järjestelmä saatiin valmiiksi kesällä 2011. Lopputuloksena syntyi järjestelmä, joka täytti tärkeimmät vaatimukset. Valmistumisen jälkeen järjestelmä luovutettiin asiakkaalle.

Avainsanat
Flash, PHP, MySQL

SAVONIA UNIVERSITY OF APPLIED SCIENCES **THESIS**
Abstract

Field of Study Technology, Communication and Transport			
Degree Programme Degree Programme in Information Technology			
Author(s) Aki Henrikki Korhonen			
Title of Thesis Gravestone Designer Application			
Date	21 April 2011	Pages/Appendices	51/0
Supervisor(s) Mr Keijo Kuosmanen, Lecturer, Mr Jussi Koistinen, Lecturer			
Client Organisation/Partners Suomen Muotokivi ltd / Mr Esa Jääskeläinen			

Abstract

The purpose of the thesis was to design and build a gravestone designer application for Suomen Muotokivi Ltd. The user can design the appearance of the gravestone with this application. The user can place an order with the application when the gravestone is ready. Professionals can make the gravestone with CAD software based on the image from the gravestone designer application.

Adobe Flash system was used to make the designer program. The designer application was then embedded to the web sites that were located in customer's server. Data between the database and the designer application was transferred using the XML and PHP technique. MySQL software was used to make the database.

Web based back-end was created in order to control the whole system. Admins can control the product catalogs, add admins and browse the orders made by customers.

The system was created and finished in summer 2011. The outcome was a system that fullfilled all the important requirements. After the system was complete it was delivered to the customer.

Keywords
Flash, PHP, MySQL

Alkusanat

Tämä dokumentti kertoo opinnäytetyöstä, joka toteutettiin Kuopiossa Savonia-ammattikorkeakoulua sekä Suomen Muotokivi Oy:tä varten. Työ toteutettiin kesällä 2011. Tahtoisin kiittää Suomen Muotokivi Oy:tä sekä lehtori Keijo Kuosmasta tästä opinnäytetyön suomasta mahdollisuudesta.

Kuopiossa 6.10.2011

Aki Korhonen

SISÄLTÖ

TERMIT JA LYHENTEET

Ad hoc -testaus	Tarkoittaa testausta, johon ei ole olemassa ennaltamääriteltyjä suunnitelmia. Testaus tapahtuu improvisoiden. Testauksen aikana ei suoriteta dokumentointia.
CAD-ohjelma	CAD-ohjelma on tietokoneavusteinen suunnitteluohjelma. Sovelluksen avulla voidaan luoda pohjapiirrokset tietokoneavusteisia työkaluja varten.
CSS	Tekniikka. jonka avulla voidaan keskitetysti hallinnoida websivujen ulkoasua. Tekniikan avulla voidaan määritellä miten HTML-elementit näkyvät verkkoselaimella. Määrityksien tekeminen ulkopuolisiin tiedostoihin on mahdollista.
Flash	Adobe Systems:n kehittämä kehitysympäristö. Järjestelmällä voidaan kehittää RIA-sovelluksia, animaatioita ja mainoksia web-sivuille. Sisällöntuottaminen on mahdollista sekä tietokoneille että mobiililaitteille.
FTP-ohjelma	Sovellus, jonka avulla voidaan paikalliselta koneelta siirtää dataa palvelimelle ja takaisin TCP-protokollaa hyväksi käyttäen.
Hallintaliittymä	Hallintaliittymällä tarkoitetaan PHP-pohjaisia web-sivuja. Järjestelmän avulla on mahdollista hallinnoida tuotteita ja tuotekategorioita. Sivut ovat piilotettu loppukäyttäjiltä ja ne ovat tarkoitettu vain ylläpitäjiä varten.
HTML	Kuvauskieli, jolla voidaan rakentaa staattisia web-sivuja. Sisällöntuottaminen tapahtuu merkkauselementtien avulla. Verkkoselain tulkitsee merkkauselementit ja tulostaa web-sivun selaimen näytölle.

InnoDB	Yksi tietokannan käyttämistä tallennusmuodoista. Mahdollistaa vierasavainten käytön tietokannoissa.
MySQL	SQL – kieleen pohjautuva tietokannan hallintajärjestelmä. Järjestelmä toimii palvelimella. Järjestelmä toimii useilla eri alustoilla.
NetBaron	Suomalainen internetpohjainen talouden – ja tuotannonohjausjärjestelmä.
PHP	Laajalti käytetty palvelinpuolen ohjelmointikieli. Tekniikka mahdollistaa sekä dynaamisen, että staattisten web-sivujen luonnin.
PNG	Kuvaformaatti, jonka avulla voidaan pakata kuvia häviöt tömästi. Kuvaformaatti sallii myös läpinäkyvät kuvat. Tukee 8 bittisiä ja 24 bittisiä värejä.
RIA-sovellus	Web-sovellus, joka tarjoaa samanlaisia ominaisuuksia sekä toimintoja kuten paikalliselta koneelta löytyvät ohjelmistot. Sovellukset ovat interaktiivisia ja ne voidaan upottaa web-sivustolle. Sovellukset ladataan kerralla selaimelle, eivätkä ne vaadi koko ajan sivuston uudelleenlataamista.
SQL-injektio	Tietoturva-aukko, jonka avulla voidaan tehdä vahinkoa tietokannalle. Uhka on suurin silloin, kun käyttäjän antamia syötteitä tietokantaan ei tarkisteta tietokantakyselyjen lomassa, jolloin käyttäjän antamat syötteet voivat pahimmassa tapauksessa olla muita SQL-komentoja. Tietoturva-aukon avulla voidaan esimerkiksi tyhjentää koko tietokanta tai ohittaa sisäänkirjautuminen.

Suunnitteluedeitori	Flash-pohjainen sovellus. Sovelluksen avulla käyttäjä itse voi suunnitella haluamansa hautamuistomerkin. Käyttäjän tekemästä hautamuistomerkistä on mahdollista tallentaa kuva sekä tilaustiedot palvelimelle.
XML	Merkintäkieli tiedonkuvausta varten. Tekniikkaa voidaan käyttää hyväksi esimerkiksi web-sovelluksissa ja tietojärjestelmien välisten rajapintojen toteuttamiseen.
XSS	Tietoturva-aukko, jota esiintyy www-sivuistoilla. Käyttäjä voi syöttää esimerkiksi haittakoodia lomakekenttiin ja aiheuttaa järjestelmään häiriöitä.

1 JOHDANTO

Tietotekniikan kehittymisestä huolimatta ei pienimmissä kivialan yrityksissä ole hyödynnetty tietotekniikan mahdollisuuksia parhaalla mahdollisella tavalla. Alan suunnittelijat käyttävät työssään CAD-pohjaisia suunnitteluohjelmia hautamuistomerkkejä tehdessään, mutta loppuasiakkaalle tällaiset ohjelmistot ovat usein hankalia käyttää. Erilaisten hautamuistomerkkien myyminen loppuasiakkaalle nojaa suurimmaksi osaksi valmiiden painettujen katalogien varaan. Painetut katalogit eivät anna asiakkaalle juurikaan vapautta muokata hautamuistomerkin ulkoasua omien mieltymyksien mukaan.

Tuotteina hautamuistomerkit ovat ainutlaatuisia. Niiden avulla omaiset kunnioittavat ja muistavat edesmenneitä. Hautamuistomerkeillä halutaan ilmentää vainajien persoonallisuutta ja elettyä elämää. Vainajien persoonallisuuksien esille tuominen synnyttää tarpeen luoda yksilöllisempiä hautamuistomerkkejä. Rakentamalla suunnitteluohjelma hautamuistomerkkejä varten loppuasiakasta silmällä pitäen voidaan parantaa asiakaspalvelua.

Suomen Muotokivi Oy on suomalainen kivialan yritys, jonka liikeidea perustuu hauta, – ja sisustuskiviin. Yritys tarvitsi ratkaisua, joka tarjosi helppoa ja yksinkertaista mallia hautamuistomerkkien ulkoasun suunnittelemiseen ja tilauksien tekemiseen. Tästä syntyi projekti, jossa rakennettiin hautamuistomerkkien suunnitteluohjelma yrityksen loppuasiakkaita varten. Yrityksen työntekijät voivat luoda oikean hautamuistomerkin loppuasiakkaan tilauksen pohjalta.

2 HYÖDYNNETYT TEKNIIKAT

Tässä luvussa esitellään niitä tekniikoita, joita käytettiin projektin tekemisen aikana.

2.1 Adobe Flash

Adobe Flash on Adobe Systemsin kehittämä kehitysympäristö, joka perustuu ActionScript -ohjelmointikieleen. Tätä alun perin vain animointia varten kehitettyä kehitysympäristöä on kehitetty vuosien saatossa tarjoamaan rajapinnat PHP:n ja XML:n kanssa. Nykyaikaisella ActionScript 3 -versiolla onkin mahdollista kehittää monipuolisia ja interaktiivisia web-sovelluksia (RIA -sovelluksia).

Flash on hyvin laajalle levinnyt standardi. Arviolta 95 % PC:n käyttäjistä on asentanut Flashin koneellensa. Merkittävä osa kotimaisista ja ulkomaisista internetin multimediapalveluista hyödyntää Flash-kehitysympäristöä. Näitä ovat mm. Youtube, Vimeo, Yle Areena ja Ruutu.fi. (Web Browser Plugin Market Share, 2011.)

ActionScript on oliopohjainen ohjelmointikieli, joka on kehitetty ECMAScript-ohjelmointikielen pohjalta. Tästä syystä sen syntaksit muistuttavat hyvin pitkälti JavaScript-ohjelmointikieltä. Aivan kuten JavaScriptikin, myös Flash on asiakaspuolen ohjelmointikieli (client).

Työssä hyödynnettiin Adobe Flash CS4:ää, joka puolestaan pohjautuu jo aiemmin mainittuun ActionScript 3 -versioon. Tekniikalla kehitettiin järjestelmän käyttöliittymä ja siihen liittyvä toiminnallisuus.

Kehittäminen kehitysympäristön avulla tapahtui luomalla projektitiedosto (.fla), jonne luodaan sovelluksen tarvittava toiminnallisuus. Tämän jälkeen projektitiedosto kääntyy Flash clientin ymmärtämään muotoon (.swf). ActionScript pohjainen ohjelma voi koostua yksittäisestä tai useammasta swf-tiedostosta. Tiedostojen hajauttaminen mahdollistaa monimutkaisen sovelluksen helpomman ylläpitämisen. Lopulliset swf-tiedostot ovat valmiita tiedostoja upotettavaksi web-sivustoille. (Moock C. 2007, 27.).

2.2 PHP

PHP on vapaaseen lähdekoodiin perustuva palvelinpuolen komentosarjakieli, jolla on monipuolinen tuki useille eri käyttöjärjestelmille, tietokannoille ja web-protokollille. PHP:n avoin lähdekoodi ja sen maailmalla saavuttama suosio on saanut aikaan sen, että siihen on luotu kattavat luokkakirjastot. (Sklar David 2004, 14.)

Komentosarjakielisyys tekee PHP:stä nopean suorittaa, koska sitä ei tarvitse erikseen palvelimella esikääntäjällä kääntää ennen tiedoston suoritusta. Tiedoston varsinainen tulkitseminen tapahtuu tiedoston suoritusvaiheessa. Skriptien suorittaminen ei myöskään vaadi paljon tehoja palvelimelta tai paikalliselta koneelta.

PHP mahdollistaa muuttujien käsittelemisen ja rajapinnat tietokantayhteyksiin. Tekniikka on suunnattu erityisesti dynaamisen sisällön tuottamiseen internetissä. Tietojen käsitteleminen PHP:llä palvelimen puolella parantaa samalla sovellusten tietoturvaa. Tekniikka piilottaa sovelluksissa käytettävän lähdekoodin käyttäjän selaimelta.

2.3 MySQL

MySQL on SQL- kyselykieleen perustuva relaatiotietokantaohjelma. MySQL tarjoaa tuen kaikille käytetyimmille käyttöjärjestelmille, ja se perustuu avoimeen lähdekoodiin. Tekniikka toimii palvelimella, ja sen avulla eri sovellukset voivat ottaa yhteyttä tietokantaan ja tallentaa dataa taulukoihin sekä noutaa tallennettua dataa takaisin. Nämä taulukot koostuvat puolestaan sarakkeista, jotka yhdessä muodostavat rivejä. Edellä mainittu järjestelmällinen tiedonjaottelu nopeuttaa tiedonkäsittelyprosessia.

2.4 XML

XML on merkintäkieli, jonka avulla voidaan kuvata dataa. Se on myös protokolla, jonka avulla voidaan välittää dataa erilaisten tietojärjestelmien kesken. Merkintäkielen kehityksen aloitti Sun Microsystems vuonna 1996. Sittemmin World Wide Web Consortium (W3C) jatkoi kyseisen merkkauskielen kehittämistä. Merkintäkieli on täysin avoin standardi. (Ray Erik 2001, 9).

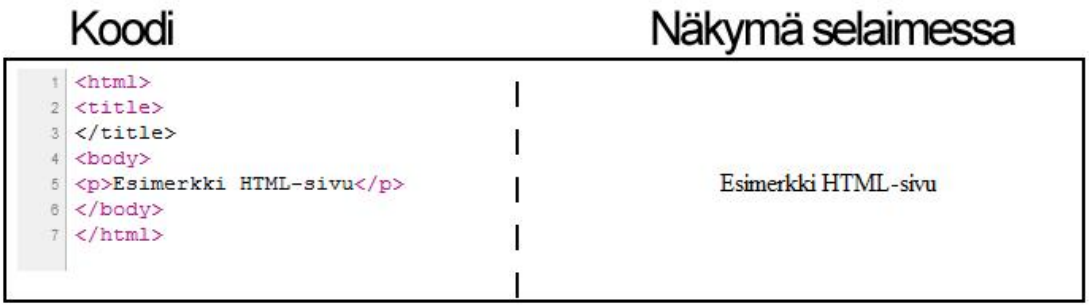
Toisin kuin HTML, XML ei keskity kuvaamaan web-sivujen ulkoasua ja rakennetta, vaan keskittyy pikemminkin tiedon rakenteen kuvaukseen, vaikka molemmat merkintäkieliä ovatkin. XML:n avulla voidaan lähettää suuria määriä dataa jäsennetyssä muodossa. Se on myös alustariippumaton. Tiedon kuvaus tapahtuu kuvaamalla dataa käyttäjän itsensä asettamien tagien avulla. Teknologia tekee mahdolliseksi sovelluskohtaisten tiedonkuvaus-skeemojen luonnin. (W3School 2011.)

2.5 WAMP

Sana WAMP koostuu sanoista Windows, Apache, MySQL ja PHP. Se on vapaaseen lähdekoodiin perustuva Windows-pohjainen kehitysympäristö, joka tarjoaa ratkaisun kehittää palvelinpohjaisia web-sivuja omalta paikalliselta koneelta. Se on kokoelma pienempiä ohjelmistokokonaisuuksia joilla voidaan ohjata palvelimen toimintaa tai vastaavasti simuloida palvelimen toimintaa paikalliselta koneelta. WAMP:n mukana tuleva phpMyAdmin–sovellus tarjoaa tehokkaan graafisen hallinnoimistyökalun MySQL-tietokantoja varten.

2.6 HTML

HTML on merkintäkieli, jonka avulla on mahdollista luoda staattisia web-sivuja verkkoselaimen ymmärtämään muotoon. Kieli koostuu kokoelmasta tageja. Tagien avulla voidaan kuvata sivuston rakennetta ja sen sisältöä. Tämä tekniikka oli toteutuksen kannalta välttämätön, sillä se sijoittuu internettiin. Tekniikan avulla voidaan luoda sivuja, joita web-selaimet ymmärtävät. Alla oleva kuva havainnollista HTML-merkintäkieltä (kuva 1).



KUVA 1. HTML havaintokuva

2.7 CSS

CSS eli *Cascading Style Sheets* on internetsivuille suunnattu tyylidokumentti, jonka avulla voidaan hallinnoida websivujen ulkoasua keskitetysti. Tekniikan kehitti Håkon Wium Lie vuonna 1994. (W3School. 1999.)

Tekniikka tarjosi tavan määritellä ulkoasun suunnittelueditorin internetsivuille. Tekniikan avulla erotetaan internetsivujen ulkoasumäärittely sisällönmäärittelystä. CSS määrittää, miten HTML-elementit sivustolla esitetään. Lisäksi tämän tekniikan avulla on tulevaisuudessa helppo tehdä muutoksia ulkoasuun.

3 MUUT HAUTAKIVIEN SUUNNITTELUOHJELMAT

Asiakkaalta saatiin spesifikaatioita sovelluksen suhteen. Näitä vaatimuksia sovelluksen suhteen olivat:

- mahdollisuus valita hautakivikategoria
- mahdollisuus säätää hautakiven kokoa ja kivilajia
- mahdollisuus lisätä koristeita
- mahdollisuus lisätä tekstiä
- reaaliaikainen hintalaskuri sovellukselle

Asiakkaalta saatujen spesifikaatioiden myötä heräsi kysymys oliko muita vastaavanlaisia järjestelmiä jo kehitetty maailmalla. Taustatyötä tehdessä kävi ilmi, että vastaavia järjestelmiä maailmalta löytyi kirjoitushetkellä vain kourallinen. Tässä luvussa keskitytään näihin olemassa oleviin ratkaisuihin tarkemmin.

Parhaiten edellä mainittuja tuntomerkkejä vastasi kotimaisen Kauhavan Kiviveistämön Velj. Mäki – yrityksen kotisivuilla oleva www-pohjainen suunnitteluohjelma. Sovellus on rakennettu siten, että asiakas itse pystyy tarvittaessa suunnittelemaan hautamuistomerkin ulkoasun perustason tietotekniikanosaamisella tai vaihtoehtoisesti valitsemaan sopivan muistomerkin valmiiden mallien pohjalta. Sovelluksen tarkastelu osoitti, että sovellus oli rakennettu JavaScriptin, Ajax-ohjelmistotekniikan, XML:n, HTML:n ja tietokannan varaan. (Hautakivisuunnittelu.fi)

Monipuolisinta ratkaisua edustaa saksalainen Schubert Comcut – ohjelmisto, joka tarjoaa ammattimaiset työkalut hautamuistomerkin suunnitteluun. Tarvittaessa ohjelman ominaisuuksia voidaan laajentaa maksullisilla lisämoduuleilla. Ratkaisu pitää sisällään kaksiulotteisen CAD-suunnitteluEditorin sekä kolmiulotteisen mallintamissovelluksen lopputuloksen esikatselua varten. Koska ohjelma vaatii monimutkaisia CAD-suunnittelutaitoja, on sillä hankala tavallisen asiakkaan suunnitella hautamuistomerkkiä, vaikkakin ohjelma on kokonaan suomennettu. (Schubert-Software).

Yhdysvaltalaiselta headstonesandmemorials.com web-sivuilta löytyi ratkaisu suunnitella hautamuistomerkki. Ratkaisu tarjoaa hyvin rajoitetun mahdollisuuden käyttäjälle muokata hautamuistomerkkiä. Käyttäjän valittavissa on kivilaji, koko, teksti ja muut grafiikat. Itse muistomerkin koristeiden ja tekstien sommitteluun näytöllä ei käyttäjällä ole vapaita käsiä. Sovellus on rakennettu Flash – suunnitteluympäristön ja tietokannan varaan. (Headstones and Memorials).

4 MÄÄRITTELY JA SUUNNITTELU

Tässä luvussa kerrotaan mitä vaatimuksia asiakas asetti projektille sekä käydään ohjelmiston osia läpi. Ennen hankkeistamissopimusta toimeksiantajalta kerättiin tietoa sovelluksen luonteesta ja toiminnallisuuksista. Tietoa näistä vaatimuksista saatiin Suomen Muotokiveltä erillisinä sähköposteina, PowerPoint – dokumentteina sekä suullisena tietona.

4.1 Tavoitteet

Asiakkaalla oli päävaatimuksena se, että palvelu pystyy tuottamaan asiakkaan haluamasta hautamuistomerkistä kuvan, jonka pohjalta on mahdollista luoda oikea hautamuistomerkki. Samalla päävaatimuksen ohessa painotettiin sitä seikkaa, että sovelluksen tulisi olla mahdollisimman helppokäyttöinen. Sovellus oli suunnattu pääasiassa Suomen Muotokivi Oy:tä varten, mutta mahdollisuus sovelluksen suuntaaminen jälleenmyyjille ei ollut pois suljettu

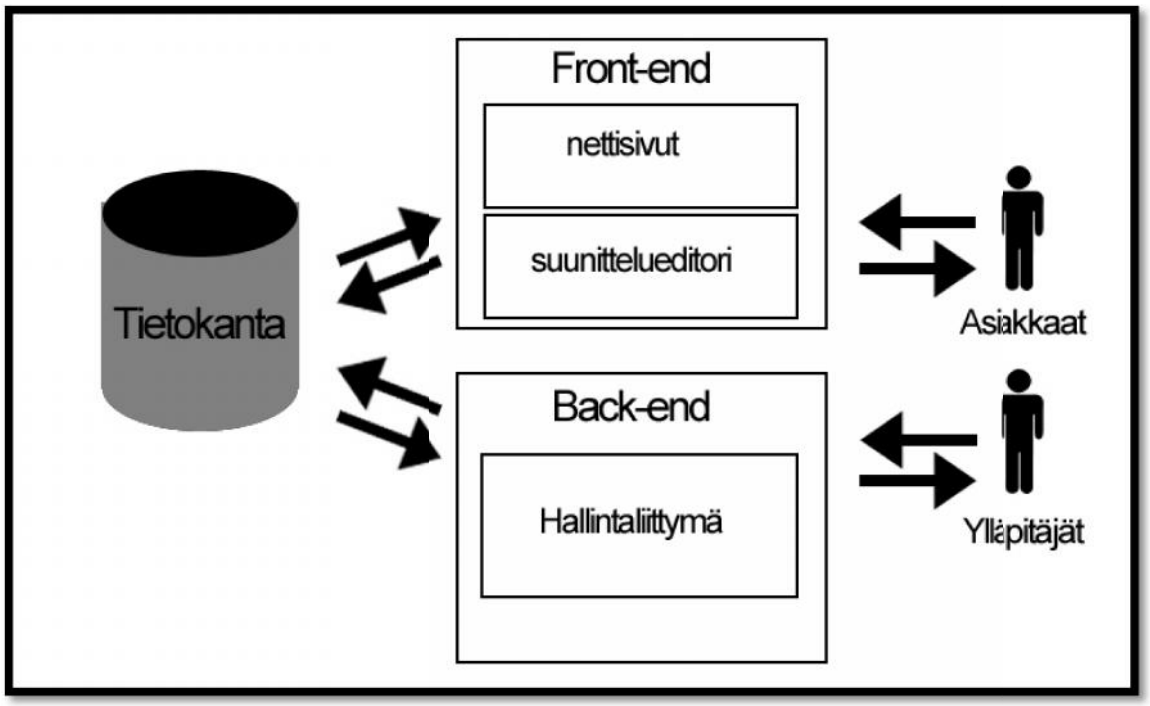
Toisarvoisia tavoitteita olivat:

- Kaikki nimikkeet, jotka löytyvät toiminnanohjauksesta tulee löytyä myös suunnitteluohjelmasta.
- Asiakasrekisterin ylläpitomahdollisuus.
- Suunnitteluohjelman tulee laskea hinta automaattisesti jälleenmyyjälle sekä laskea myyjän provisio hinnasta.
- Kuluttajalla tulee olla mahdollisuus nähdä suunnitteluohjelmassa olevan tilauksen hinta.
- Jälleenmyyjät kirjautuvat hallintapaneeliin salasanoilla.
- Sovelluksessa olevien nimikkeiden hintoja tulee pystyä muuttamaan.
- Palvelussa tulee olla valmius NetBaronin rajapinnalle.
- Suunnitellun hautamuistomerkin tulee pystyä tallentamaan ja sitä pitää pystyä muokkaamaan jälkeenpäin.

(Jääskeläinen 2010, s.6)

4.2 Järjestelmän rakenne

Koska asiakkaalla ei ollut vaatimuksia järjestelmän rakenteen suhteen, päätettiin järjestelmä jakaa kolmeen eri osakokonaisuuteen. Ensimmäisen kokonaisuuden muodostaa tietokanta, joka pitää tallessa järjestelmän tarvitsemaa dataa. Toisen kokonaisuuden muodostaa ylläpitäjiä varten rakennettu hallintapaneeli. Hallintapaneelin kautta hallinnoidaan järjestelmän asetuksia. Hallintapaneeli on web-sivusto kokonaisuus, joka on piilotettu loppuasiakkailta. Viimeisen kokonaisuuden muodostaa palvelun julkisivu joka on luotu loppuasiakkaita varten. Julkisivu koostuu web-sivuista, jonka avulla voidaan hallinnoida asiakastietoja ja tehdä tilauksia. Itse suunnitteluedi tori on upotettu näille julkisivun muodostamille sivustoille. Alla oleva kuva havainnollistaa järjestelmän osakokonaisuuksia (kuva 2).



KUVA 2. Kuvaus järjestelmästä

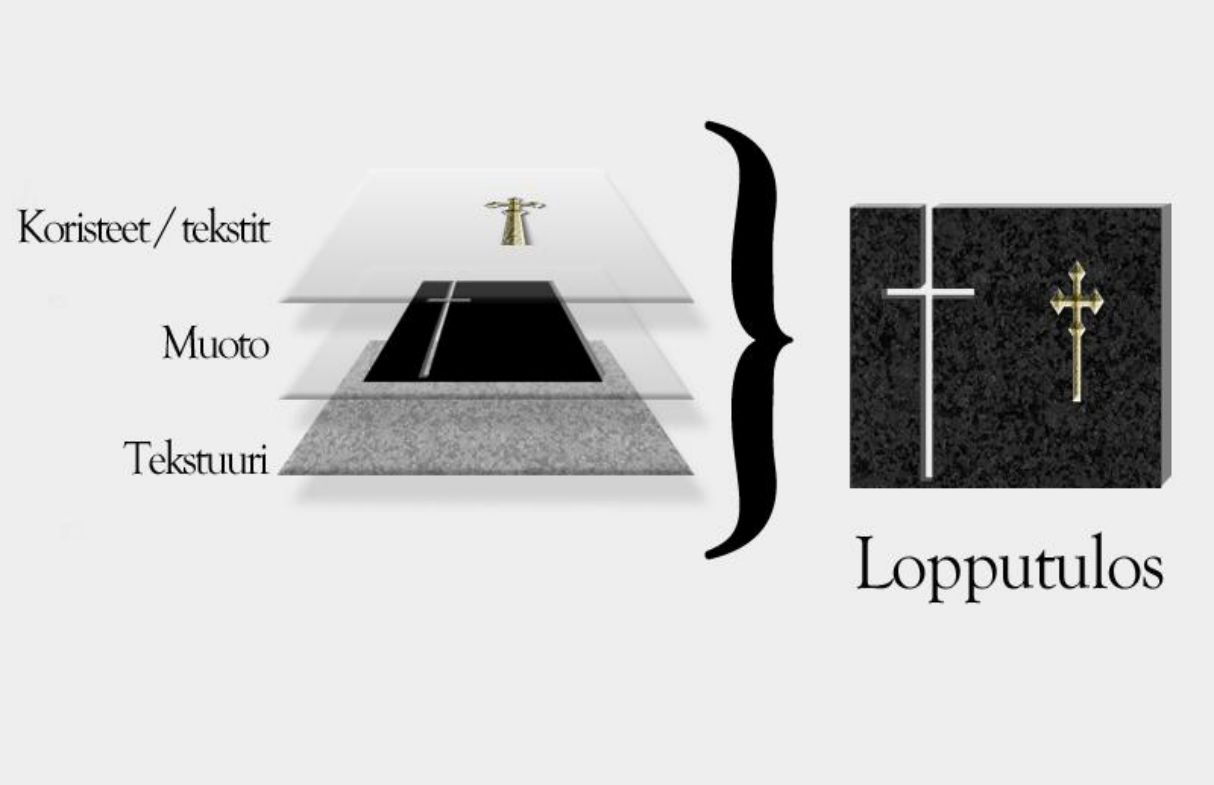
Asiakkaalta ei ollut myöskään reunaehtoja sille, millä tekniikoilla palvelu tulisi toteuttaa. Tämä antoi itse tekijälle vapauden valita projektiin sopivat tekniikat. Lopputuloksena päädyttiin valitsemaan työkaluiksi Adoben Flash kehitysympäristö suunnitteluedi torin varten (client), PHP-komentokieltä ja MySQL – relaatiotietokantaohjelmistoa palvelinpuolen ohjelmointiin (server).

4.2.1 Suunnittelueditori

Potentiaalisia työkaluja suunnittelueditorin tekemiseen oli JavaScript + AJAX, Java tai .NET –ympäristöt, mutta lopulta valittiin Flash -kehitysympäristö tämän toiminnallisuuden toteuttamista varten. Flashin vahvimpina etuina kilpailijoihin nähden katsottiin olevan tasohallinta sekä se, että Flash-liitäntä löytyy suurimmalta osalta tietokoneen käyttäjistä koneelta. Flashin ActionScript 3 tarjoaa lisäksi tuen kaksisuuntaiseen kommunikaation palvelimen ja asiakkaan välillä. Tämä mahdollistaa dynaamisten web-pohjaisen sovellusten rakentamisen.

Sovelluksen tiedettiin sijoittuvan internetiin, joten raskas kolmiulotteinen suunnittelueditori ei ollut järkevä ratkaisu, koska käsiteltävän datan määrä olisi ollut tällöin suurempi. Lisäksi aloituspalaverissa arvioitiin, ettei kolmiulotteisuus tarjoaisi loppuasiakkaalle mitään lisäarvoa. Näistä syistä suunnittelueditori pohjautuu kaksiulotteisten kuvien mallintamiseen. Muistomerkin kuvaaminen editorissa muodostuu muistomerkin muodosta, pintakuviosta eli tekstuurista, muistokiven koosta, muistokirjoituksista sekä koristeista.

Flashin tasohallinta mahdollistaa muistomerkin luomisessa sommitella eri sovelluksen tasoja keskenään siten, että voidaan muodostaa uusia kuvaefektejä. Esimerkiksi muistomerkin muoto on kuvattuna yhdellä kuvalla alemmalla tasolla. Tämä muoto voi olla vaikkapa tavallinen kuva. Muoto on vain mustavalkoinen silhuettikuvaus muistomerkin muodosta. Tämän muodon ylemmällä tasolla on toinen kuva, jolla kuvastetaan muistomerkin kivilajia. Ylemmän tason tyylejä määrittelemällä saadaan aikaan kuva, jossa näkyy muistomerkin muoto ja kivilaji yhtenä kuvana. Tällöin syntyy illuusio kuvasta, jolla on sekä muoto että tekstuuri (kuva 3). Tästä saavutetaan se hyöty, ettei tarvitse luoda suuria määriä erilaisia kuvavariaatioita muistomerkeistä erilaisine muotoineen ja tekstuureineen.



KUVA 3. Havaintokuva Flashin tasohallinnasta

Asiakkaalla oli määrityksiä suunnitteluedeitorin toiminnallisuuden osalta. Sovellukseen tuli pystyä valitsemaan valmis kivimalli katalogista. Hautakivien kokoa, muotoa ja kivilajia tuli pystyä sovelluksessa muuttamaan. Tekstien osalta tuli olla mahdollisuus vaihtaa fonttia, kokoa, väriä ja leveyttä. Fonttien tuli vastata samoja fontteja, mitkä olivat Suomen Muotokivellä käytössä heidän leikkuriohjelmissaan. Kaikkien koristeiden tuli olla samoja, jotka löytyvät jälleenmyyntihinnastosta. (Jääskeläinen 2010, s.4-5)

Suunnitteluohjelman tilauskaavakkeella tuli löytyä seuraavat tiedot:

- Jälleenmyyjän nimi
- Yhteyshenkilö
- Osoite, postinumero
- Sähköposti
- Hautausmaan nimi
- Hautausmaan paikkakunta
- Lisätietokenttä

(Jääskeläinen 2010, s. 7)

4.2.2 Hallintaliittymä

Koska asiakas toivoi sovellukselta joustavuutta ylläpidettävyydeltä, valittiin dynaamista hallintaliittymän rakentamista varten PHP -komentokieli sekä MySQL-tietokanta. Tekniikoiden avulla päätettiin luoda web-pohjaiset sivut, jotka oli määrä luoda asiakkaan palvelimelle.

Hallintaliittymän avulla voidaan hallinnoida järjestelmällisesti

- tuotteita
- hintoja
- ylläpitäjiä
- tilauksia.

Hallintaliittymän tietokantaa varten oli olemassa useita potentiaalisia vaihtoehtoja. Tietokannan valinnan osalta päädyttiin lopulta MySQL–relaatiotietokantaan, koska se on helppokäyttöinen, vakaa ja mahdollisissa ongelmatilanteissa siihen löytää apua hyvän dokumentaation ansiosta.

Tietokannan taulukot asetettiin käyttämään InnoDB-moottoria, koska se mahdollisti vierasavainten käyttämisen taulujen suhteiden kuvauksessa. Tosielämän tarpeita palvelevat tietokantataulut hahmoteltiin käsitemallien avulla. Keskeisimmistä käsitteistä luotiin omat taulut, minkä jälkeen lähdettiin tarkentamaan taulujen sisältöä. Taulut pyrittiin optimoimaan siten, ettei niihin syntynyt toistuvia tietueita. Kun alustavat taulut oli saatu hahmoteltua tarkistettiin, palvelivatko ne kaikkia asiakkaan tarpeita.

5 TOTEUTUS

Tässä luvussa perehdytään projektin toteutukseen jokaisen projektin osa-alueelta. Luvussa selitetään, miten ratkaisuihin päädyttiin ja hahmotellaan järjestelmän toimintaa.

5.1 Prototyyppien kehitys

Kun määritysvaiheen spesifikaatiot oli saatu, alettiin hahmotella sovelluksen ulkoasua ja toiminnallisuutta tekemällä erilaisia prototyyppejä. Prototyyppejä tekemällä saatiin samalla arvokasta kokemusta toteutuksen hahmottelua varten. Nämä prototyypit olivat kooltaan pieniä sovelluksia, joissa keskityttiin vain tiettyihin toiminnallisuuksiin. Prototyyppejä tekemällä pyrittiin kartoittamaan mahdollisia ongelmakohtia toteutuksessa.

Flashin avulla prototyyppejä tehtäessä keskityttiin sovelluksen kannalta olennaisimpiin ominaisuuksiin kuten tiedon siirtoon Flashista PHP:n ja kuvien luomiseen. PHP:lla rakennettiin alustava hallintapaneeli ylläpitäjälle ja MySQL:llä hallintapaneelin tarvitsemat taulut.

5.2 Suunnittelueditorin kehitys

Suunnitteluohjelma on Flash-tekniikan päälle rakennettu suunnittelueditori, joka upotetaan PHP-pohjaisille web-sivustolle. Suunnittelueditori käyttää hyväkseen MySQL-tietokantaa. Tässä luvussa kerrotaan suunnittelueditorin osa-alueista.

5.2.1 Ensitoimenpiteet

Prototyyppien valmistuttua lähdettiin tekemään varsinaista sovellusta. Flashilla luotiin suunnitteluohjelmalle oma projektitiedosto. Projektitiedostoon tehtiin ensimmäisenä käyttöliittymä. Käyttöliittymän päävalikkoja ovat kiviasetukset, tekstiasetukset ja koristeasetukset. Nämä valikot järjestettiin suunnittelueditorissa limittäin päällekkäin, jotta itse suunnittelutyöpöydälle jäi mahdollisimman paljon tilaa. Käyttäjän valittua haluamansa valikon piilotettiin muut valikot valitun valikon taakse (kuva 4). Suunnittelueditorin haluttiin olevan sopiva suurimpiin osiin tietokoneiden monitoreista, joten sen koko määriteltiin 1024 pikseliä leveäksi ja 800 pikseliä korkeaksi.

Ylläpitäjää varten jouduttiin luomaan toinen vastaavanlainen suunnitteluedi-tori, koska asiakas halusi järjestelmään mahdollisuuden suunnitella valmiita muistomerkkipohjia loppuasiakkaita varten. Käytännössä tämä tarkoitti sitä, että tämä ylläpitäjän editori oli kopio alkuperäisestä loppuasiakkaan editorista, johon oli tehty muutoksia mallipohjia varten. Ylläpitäjän editorista tallennetaan mallipohjan tiedot tietokantaan, josta ne voidaan tarvittaessa hakea ja tulostaa uudestaan suunnitteluedi-torin ruudulle. Ylläpitäjän suunnitteluedi-torissa hinnoittelu on kytketty pois, jotta ylläpitäjä voi syöttää mallipohjalleen halutun tarjoushinnan.

5.2.2 Käyttöliittymä

Kun kolmiulotteista suunnitteluedi-toria ei rakennettu, päädyttiin siihen, että päätetään kuvata muistomerkki kavaljeeriperspektiivissä. Kavaljeeriperspektiivissä muistomerkin kuvaus tapahtuu kolmella erisuuntaisella viivalla: 45 asteen viistoilla syvyysviivoilla (z-akseli) sekä suorilla vaaka (x-akseli) ja pystyviivoilla (y-akseli). Perspektiivi antaa vaikutelman kolmiulotteisuudesta sekä kuvat on helppo rakentaa tässä perspektiivissä. Alla oleva kuva havainnollistaa käyttöjärjestelmän ulkoasua (kuva 4).

KUVA 4. Käyttöliittymän yleisnäkymä

Kuvasta 4 käy ilmi valikoiden sijoittelu muistomerkkidesignerilla. Yllä olevilta välilehdiltä on mahdollista säädellä muistomerkin tekstejä, kivimallia sekä koristeita. Oikealla yläkulmassa on havaittavissa reaaliaikainen hintalaskuri, joka laskee muistomerkin hinnan. Keskellä kuvaa näkyy itse rakennettu muistomerkki koristeineen ja teksteineen. Koristeiden ja tekstien sijaintia on mahdollista siirrellä kursorilla.

Kuvan alalaidassa on nähtävissä kivet-välilehti, josta käsin voidaan säätää muistokiven mallia, kokoa, kivilajia, reunakiillotusta sekä aluskiviä. Välilehden vasemmassa alakulmassa on havaittavissa skaalauskuva, josta näkyy koko muistomerkin koko suhteessa ihmiseen. Skaalauskuva päivittyy muistokiven koon muutosten yhteydessä.

Tekstit-välilehdellä on mahdollista hallinnoida hautamuistomerkkiin liittyviä muistokirjoituksia. Kirjoitusten lisäksi valikosta voidaan säätää fontin lisäksi kokoa, väliä ja väriä. Fontista on mahdollista tehdä kaiverrettava tai pronssinen kohokirjain.

Tekstien puolesta asiakkaalla oli toiveena, että metalliset kohoteksteihin saataisiin kolmiulotteista vaikutelmaa. Tästä syystä metallisille teksteille asetetaan suunnittelueditorissa filttereitä, jotka antavat niille haluttua kohovaikutelmaa. Alla olevassa kuvassa oleva koodi havainnollistaa, kuinka Flash – ympäristössä on mahdollista asettaa teksteille efektejä (kuva 5).

```
o.getChildAt(0).setTextFormat(myFormat);

//haetaan sopivat filtterit sopivaa efektiä varten
var filt:GlowFilter = new GlowFilter;
var filt_varjo:DropShadowFilter = new DropShadowFilter;


//asetetaan filttereille sopivat arvot
filt.color = 0xC96333;
filt.blurX = 1;
filt.blurY = 2;
filt.alpha = 0.5;
filt_varjo.blurX = 4;
filt_varjo.blurY = -2;
filt_varjo.alpha = 1;
filt_varjo.strength = 255;
filt_varjo.angle = 5;
myFormat.color = 0xC96333;

//asetetaan teksteille efektit
o.getChildAt(0).filters = [new BevelFilter(2.5, 45), filt_varjo, filt];
```

KUVA 5. Efektin asettaminen teksteille

Koristeet-välilehdellä voidaan hallinnoida hautamuistomerkin koristeita. Käyttäjä valitsee valikosta haluamansa kategorian, jonka jälkeen järjestelmä listaa kategorian alla olevat koristeet viereiseen valikkoon. Valikosta on näkyvillä kuva ja tuotteen nimi. Valikosta käyttäjä voi klikata haluamansa koristeen, jonka jälkeen järjestelmä lisää sen tilaukseen ja tulostaa sen suunnitteluedittorin kuvaruudulle.

Koska käyttöliittymässä oli suuri määrä erilaisia komponentteja, oli pakko nimetä ne selkeästi. Nimien haluttiin paljastavan niiden sijainnin ja funktion projektitiedostossa. Tästä syystä nimien eteen syötettiin tieto siitä minkä valikon alla ne sijaitsivat sekä tieto siitä mistä funktiosta ne vastasivat. Tämä johti pitkiin komponenttinimiin, mutta muutoin nimeäminen selkeytti järjestelmän ylläpitoa.

Tekstien ja koristeiden koko suhteutetaan hautakiven kokoon, jotta suunnitteluedito-rissa tehdyn luonnoksen mittasuhteet vastaa tosielämän lopputulosta. Alla olevan kuvan koodi havainnollistaa skaalauksen kulkua (kuva 6).

```
//lasketaan kerroin

var kerroin:int = globaali_civen_leveys / alamenu_kivet.alamenu_kivet_leveys.value;

//lasketaan suhteellinen leveys
bm.width = currentKategoria[j].Leveys * kerroin;

//skaalataan korkeus sopusuhtaiseksi leveyteen nähden
bm.scaleY = bm.scaleX;
```

KUVA 6. Skaalausta suhdeluvun avulla

Koodi skaalaa kertoimen avulla, joka on suhdeluku. Koristeen leveys editorin kuva-ruudussa määräytyy lopulta sen fyysisen koon ja suhdeluvun kertoimen tulona. Lo-puksi korkeus skaalataan, jotta leveyden ja korkeuden väliset mittasuhteet eivät vää-ristyisi.

5.2.3 Tuotteiden tuominen suunnitteluedi toriin

Suunnitteluedi torin käynnistyessä ladataan hautakivien muodoista sekä koristeista kategoriatiedot. Kivilajeja ei ole millään tavalla kategorioitu, joten niiden osalta tiedot voidaan suoraan hakea tekstuurit.xml – tiedostosta. Suunnitteluedi tori ottaa yhteyttä seuraaviin tiedostoihin:

- flash_haeKivikategoriat.php
- flash_haeKoristekategoriat.php

Molemmat tiedostot palauttavat tiedot käytettävissä olevista kategorioista, jotka lada-taan talteen suunnitteluedi torin kategoriavalikkoihin. KUVA 7 havainnollistaa tietojen tulostusta. Kivikategorioiden osalta saadaan vielä tarkempaa tietoa jokaisesta kate-goriasta seuraavasti:

- kategorian nimi
- onko aluskivi käytettävissä
- onko tassukivet käytettävissä
- maksimi korkeus
- minimi korkeus
- maksimi leveys
- minimi leveys

```php
<?php
    //määritellään XML-julistus
    header('Content-Type: text/xml');
    print '<?xml version="1.0" encoding="UTF-8" standalone="yes"?>';


    //haetaan tiedot tietokannasta
    include("../config.php");
    $dblink = mysql_connect($_SESSION['host'], $_SESSION['db_user'], $_SESSION['db_user_pw']);
    mysql_select_db($_SESSION['db'], $dblink);

    $sql = "SELECT * FROM katalogi_koriste_kategoria";
    $result = mysql_query($sql);
    $kategoriat = array();

    if( $result )
    {
        while($row = mysql_fetch_array($result))
        {
            $kategoriat[] = $row['kategoria'];
        }
    }

    //rakennetaan dokumentti
    $xml_output = "<root>\n";
    for($i = 0; $i < sizeof($kategoriat) ; $i++)
    {
            $xml_output .= "\t\t<kategoria>";
                $xml_output .= "\t\t\t\t<nimi>". $kategoriat[$i]. '</nimi>\n";
            $xml_output .= "</kategoria>\n";

    }
    $xml_output .= "</root>";

    //tulostetaan dokumentti
    echo $xml_output;
?>
```

KUVA 7. Koristekategorioiden tulostusta

Käyttäjän valitessaan halutun kategorian suunnitteluediteri lataa valitun kategoriakohtaisen xml-tiedoston palvelimelta. Näistä tiedostoista käy ilmi tuotenimikkeiden kuvapolut palvelimella, nimi, mahdollinen tyyppi, koko ja hinta. Tämän jälkeen tuotteet voidaan kuvapolkujen avulla listata näytille suunnitteluediteriin valikkoihin. Käyttäjän valittua uusi tuotenimike valikosta, suunnitteluediteri lataa kuvapolun avulla uuden tuotteen näytille työpöydälle ja laskee suunnitellulle hautamuistomerkille uuden hinnan. Alla oleva kuva havainnollistaa tuotteiden lataamista suunnitteluediteriin (KUVA 8).

```
//asetetaan kuuntelija kivilajeja varten:
alamenu_kivet.alamenu_kivet_lajit.addEventListener(Event.CHANGE, vaihdaKivilaji);

function vaihdaKivilaji(e:Event) :void
{

    var image1:Loader = new Loader();

    //noudetaan kuva
    image1.load(new URLRequest(e.target.selectedItem.source));

    //poistetaan entinen kuva
    tekstuuri.removeChildAt(0)

    //ladataan tilalle uusi kuva
    tekstuuri.addChild(image1)
    tekstuuri.x = 0;
    tekstuuri.y = 0;

}
```

KUVA 8. Kivilajin lataaminen suunnitteluedittoriin

5.2.4 Hautakiven mitoitus

Hautakiven mitoitus alkaa kun käyttäjä valitsee ensin haluamansa hautamuistomerkin muodon kivikategorioista. Suunnitteluedittori lataa muodon talteen hautamuistomerkin muodoista vastaavaan MovieClip-olioon. Tämän jälkeen muoto hautamuistomerkistä latautuu suunnitteluedittorin näytölle.

Kun muoto ruudulle on latautunut, voidaan säätää kiven koko sopivaksi. Tämä tapahtuu säätämällä Kivet-välilehdessä löytyviä leveys ja korkeus-liukusäätimiä. Liukusäätimien avulla MovieClip-oliossa tallessa oleva kuva säädetään alla olevan kuvan mukaisesti (kuva 9). Kun korkeus ja leveys hautakivelle on säädetty, aktivoituvat muut valikkovälilehdet ja käyttäjä voi esimerkiksi lisätä hautakivelle tekstiä sekä koristeita.

```
//asetetaan kuuntelija korkeus-liukusäätimelle:
alamenu_kivet.alamenu_kivet_korkeus.addEventListener(Event.CHANGE, vaihdaKivenKorkeutta);

function vaihdaKivenKorkeutta ( e:Event): void
{
        //asetetaan korkeus:
        kivi_muoto.height = 5 * e.target.value;

        //päivitetään tulos näytölle:
        alamenu_kivet.label_kivet_korkeus.text = e.target.value + ' cm";

        //keskitetään:
        kivi_muoto.y = editointialue.y - (editointialue.y / 2) - e.target.value;
        hautakivi_pohja.y  = kivi_muoto.y + kivi_muoto.height + (hautakivi_pohja.height / 2 );

        if( ala...

        //alapalkki ja pohja kuosiin:
        var ar...
        alapalkki1.y = hautakivi_pohja.y - (hautakivi_pohja.height/2);
        alapalkki2.y = hautakivi_pohja.y - (hautakivi_pohja.height/2);

        if(alam...

        SkaalaaKaikkiKoristeet(); //skaalataan koristeet suhteessa kiveen
}
```

KUVA 9. Hautakiven koon säätäminen

5.2.5 Ostoskori

Käyttäjän valitsemia tuotteita hallitsemaan rakennettiin erillinen ostoskoriluokka. Lisättyjä tuotteita hallitaan lisäämällä, poistamalla tai muokkaamalla tietoja. Luokka implementoidaan suunnitteluedittoriin käynnistyksen yhteydessä.

Luokka pitää sisällään sisennettyjä taulukoita tekstejä, koristeita ja hautakiveä varten. Sisennetyt taulukot mahdollistavat struktuurimaisen tiedon tallentamisen jäsennetyssä muodossa. Luokkaan tuodaan myös viittaus suunnitteluedittorissa sijaitsevasta tekstikentästä, joka näyttää tilauksen hintaa. Päivitettäessä hintaa tieto uudesta hinnasta välitetään takaisin käyttöliittymän tekstikenttään. Alhaalla olevassa kuvassa oleva koodi havainnollistaa tätä hinnan päivitystä (kuva 10).

```
public function paivitaHinta():void
{
        var hinta:Number = new Number();

        //lasketaan koristeiden yhteishinta

        var n:Number = new Number();
        n = 0;

        for each( var i in _koristeet )
        {
                var s:String = i.Hinta;
                var n2:Number = new Number(s);
                n += n2;
        }

        hinta = _hinta_tarjous + n + _hinta_kivi_muutos;


        //lasketaan tekstin yhteishinta
        hinta += _hinta_tekstit;

        //ynnätään yhteen muita optioita

        if( _aluskivi == false )
        {
                hinta -= _hinta_aluskivialennus
        }

        hinta -=  _hinta_reunakivikiillotus;
        hinta += _hinta_kivilaji_muutos;
        hinta -= _hinta_koriste_muutos;


        //päivitetään hinta suunnitteluEditoriin
        _hinta.text = hinta.toString();
}
```

KUVA 10. Ostoskorin hinnan päivitysfunktio

Hinnoittelun monimuotoisuuden vuoksi, jokaisen tuotenimikkeen tiedot tulee tarkistaa ennen hinnan laskemista. Hautakiveä hinnoitettaessa täytyy ottaa huomioon, onko sen oletuskokoa muutettu ja kuinka paljon koko on muuttunut. Mikäli hautakivi ei ole oletuskokoa, lisätään siihen lisämaksua sen mukaan, paljonko sen kokoa on muutettu. Myös kivilaji vaikuttaa hautakiven hintaan. Tekstien hintaan vaikuttavat tekijät ovat kategoria, koko ja tekstin pituus. Koristeiden hintaan vaikuttaa lähinnä tilanne, jossa valmiin mallipohjan pronssinen koriste muutetaan kaiverretuksi. Tällöin asiakkaalle annetaan alennusta. Käyttäjä saa myös alennusta, mikäli hän jättää pois hautamuistomerkille tarkoitetun aluskiven. Lisämaksua veloitetaan hautamuistomerkin reunakiil-

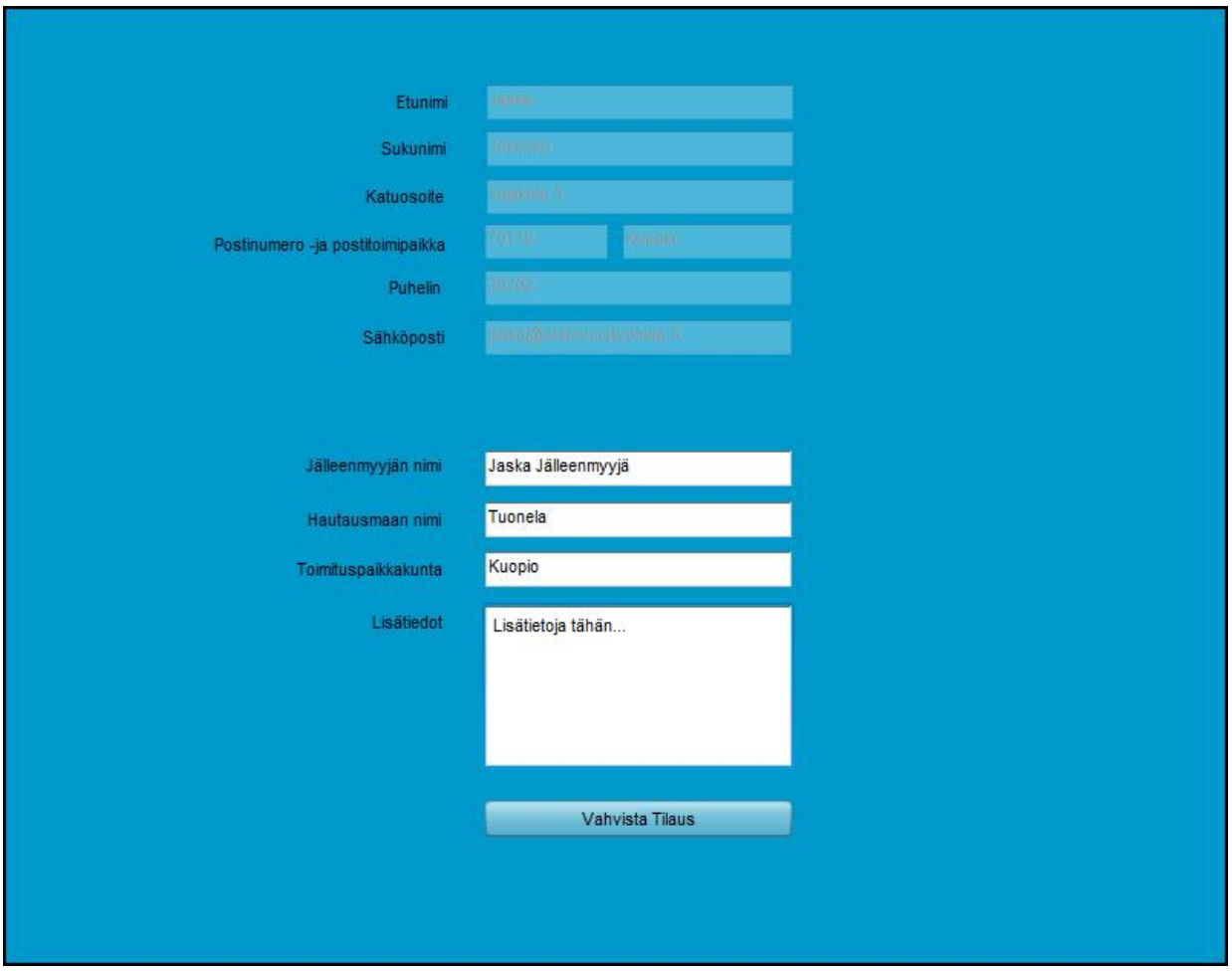
lotuksesta. Muutoin koristeiden hinnat lasketaan yhteen aivan normaalisti. Suunnittelueditori lataa edellä mainitut hintatiedot palvelimelta ja lähettää saamansa datan ostoskoriluokalle.

Ostoskoriluokka rakentaa tilausvaiheessa valituista tuotteista xml-tiedostot, jotka voidaan lähettää palvelimelle. Ylläpitäjän suunnitteluedittorilla puolestaan luokkaa käytetään luomaan xml-tiedostot pohjamalleja varten.

5.2.6 Tilauksen tekeminen

Käyttäjä kirjautuu tunnuksillaan palveluun web-sivuilla. Sivustolla hän voi valita, luoko hän täysin uuden tilauksen vai luoko hän tilauksen valmiin ylläpitäjän tekemän mallin pohjalta, painamalla sivuston valikolla näkyviä nappeja. Päätettyään minkälaisen tilauksen hän luo, käyttäjä ohjataan valikkoon, jossa hänen tulee valita jälleenmyyjä. Kun käyttäjä on valinnut jälleenmyyjän, tietää sovellus laskea provisio-osuuden myyjälle tilauksen päätyttyä. Tämän jälkeen käyttäjä ohjataan suunnitteluedittoriin.

Kun käyttäjä on saanut valmiiksi haluamansa muistomerkin, ohjataan käyttäjä tilausikkunaan syöttämään tilaukseen tarvittavat tiedot. Alla olevasta kuvasta käy ilmi tilauslomakkeen ulkonäkö ja tiedot (kuva 11).



KUVA 11. Tilauslomakkeen yleisnäkymä. Lomakkeesta käy ilmi tilauksessa tarvittavat tiedot

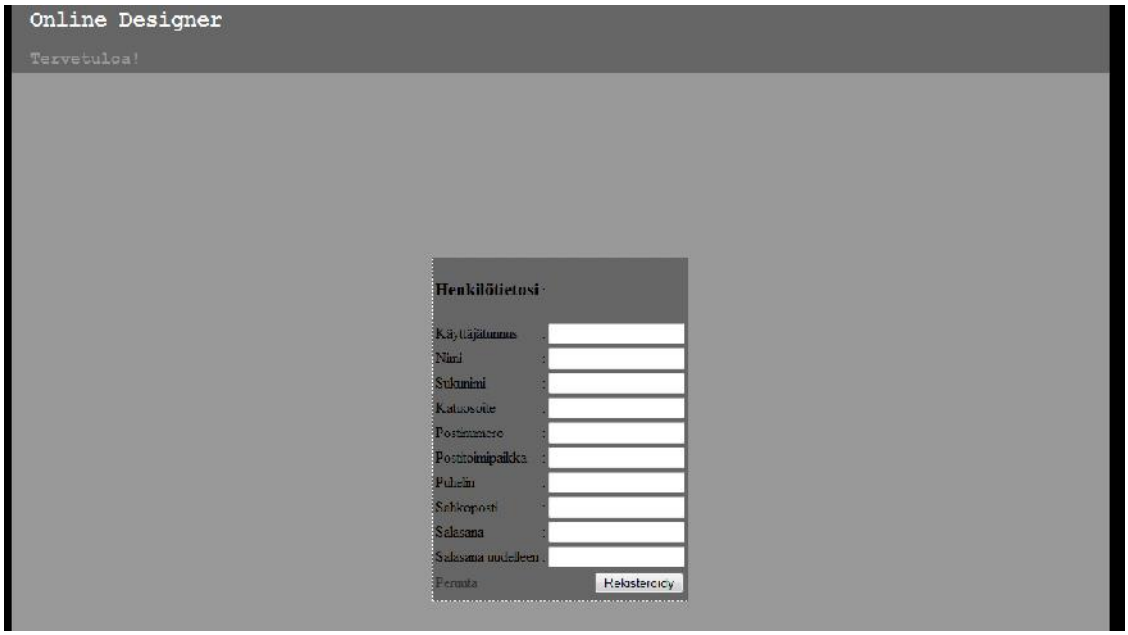
Yllä olevasta kuvasta voi nähdä, kuinka järjestelmä on jo esitäyttänyt asiakkaan henkilötiedot kuvan yläosassa. Järjestelmä hakee tietokannasta asiakkaan tiedot, jotka hän on sinne rekisteröinyt. Asiakkaan täytettäväksi jäävät vain jälleenmyyjän nimi, hautausmaan nimi, toimituspaikkakunta ja mahdolliset lisätiedot. Tilaus sinetöidään painamalla ”Vahvista tilaus” -nappia. Tilauksen tallennuksen yhteydessä Flash luo editorissa olevasta muistomerkistä yksilöidyn kuvan ja lähettää sen palvelimelle, jonne se tallennetaan.

Kaikista käyttäjän valitsemista tuotteista, asiakas– ja tilaustiedoista muodostetaan xml-tiedosto, joka lähetetään flash_teeTilaus.php–tiedostolle. Palvelimen päässä tilaustiedoista tallennetaan tiedot tietokantaan.

Muistomerkkidesignerin valmistuttua käännettiin projektitiedosto swf-tiedostomuotoon ja upotettiin PHP-pohjaisille web-sivuille. Sivustolla käyttäjän tulee rekisteröityä palveluun päästäkseen käsiksi muistomerkkidesigneriin.

5.2.7 Asiakkaan rekisteröiminen järjestelmään

Rekisteröityminen palveluun tapahtuu palvelun web-sivustolla. Painamalla Rekisteröidy käyttäjäksi –linkkiä ohjataan käyttäjä rekisteröitymislomakkeelle, jonne hän syöttää henkilötietonsa (kuva 12).



KUVA 12. Rekisteröimislomake palveluun

Rekisteröintivaiheessa asiakkaan antamat syötteet tarkastetaan mysql_real_escape_string – ja strip_tags-funktioiden avuilla ennen tietokantaan tallentamista mahdollisten SQL-injektioiden ja XSS -tietoturva-aukkojen varalta.

Rekisteröityessä käyttäjä syöttää järjestelmään henkilötietonsa, käyttäjätunnuksen ja salasanansa. Rekisteröinnin yhteydessä järjestelmä tarkistaa, onko olemassa identtistä nimimerkkiä jo entuudestaan. Rekisteröintiprosessi peruuntuu, mikäli järjestelmästä löytyy toinen identtinen nimimerkki. Mikäli numeerisiin kenttiin on syötetty aakkosia tai aakkosellisiin kenttiin on syötetty numeerista dataa, keskeytetään rekisteröityminen ja käyttäjälle annetaan ilmoitus vääränlaisen tiedon syöttämisestä kenttiin. Onnistuneessa rekisteröintitapahtumassa käyttäjän antama salasana suojataan lopuksi md5-algoritmilla, etteivät ne päädy tietokantaan selväsanaisena.

Tallennettuja henkilötietoja käytetään sovelluksessa myöhemmin hyväksi siten, että suunnitteluеditori lataa asiakkaan henkilötiedot valmiiksi tilauslomakkeelle eikä asiakkaan itse tarvitse syöttää tietojaan joka tilauksen yhteydessä.

5.2.8 Kuvan luominen ja tallennus palvelimelle

Suunnitteluеditorin käynnistyessä luodaan ensin kuvalle nimi, jotta useampien tilauksien kuvat voidaan yksilöidä, jotta ne eivät tallennu toistensa päälle. Mikäli kuitenkin muokataan vanhaa tilausta, kuva tallennetaan vanhan kuvan päälle. Kuvanimen luonti tapahtuu suunnitteluеditorin kutsuessa flash_luoKuvatunnus.php – tiedostoa.

PHP:n avulla tarkistetaan, onko käyttäjä valinnut muokattavaksi vanhan tilauksen vai onko tilaus kokonaan uusi. Mikäli käyttäjä muokkaa vanhaa tilausta, on kytketty session-muuttuja, johon on tallennettu muokattavan tilauksen indeksinumero tietokantataulusta. Tällöin kuvanimi voidaan hakea suoraan tietokantataulusta ja palauttaa takaisin suunnitteluеditoriin.

Käyttäjän tehdessä uutta tilausta ei session-muuttujaan ole tallennettu mitään dataa. Tällöin kuvanimi muodostetaan asiakkaan tunnusnumeron ja tilausten maksimiarvon avulla. Luotu kuvanimi palautetaan myös suunnitteluеditoriin.

Kuvanimi noudattaa muotoa:

```
asiakas[asiakasnumero]tilaus[maksimi-arvo].png
```

Kuvista tehdään png-kuvia, koska niitä luodessa ei synny kuvahäviötä. Kuvista on myös tarvittaessa mahdollisuus tehdä jpg:ta, mutta näissä kuvalaatu on huonompi. On tärkeää, että henkilötiedot voidaan lukea lopullisesta tilauskuvasta.

Kun kuvanimi on tilaukselle luotu ja asiakas on päättänyt tehdä tilauksen, on aika luoda varsinainen kuva tilauksesta. Luodaan ActionScript 3 BitmapData-luokan avulla uusi instanssi, joka ottaa parametreikseen tiedot kuvan dimensioista. Tämän jälkeen täytyy tallentaa editorissa näkyvän tilauksen tiedot instanssin pikselitaulukkoon. Muunnetaan instanssin pikselitaulukko bittitaulukoksi käyttämällä avuksi PNGEditor-

luokkaa. Bittitaulukko voidaan myöhemmin muuntaa png -kuvaksi palvelimen päässä. Tämä tapahtuu, kun suunnitteluedi tori kutsuu upload.php–tiedostoa, jossa bittitaulukosta luodaan png-kuva ja tieto kuvapolusta tallennetaan tietokantaan.

Asiakkaan palvelimelle jouduttiin lopuksi tekemään kansioasetuksia kuvaluontia varten. Tilauskuvista vastaavaan kansioon annettiin kirjoitusoikeudet, jotta siirto palvelimelle mahdollistui. Kansioasetuksien teko tapahtui FTP-ohjelmalla.

5.3 Hallintaliittymän kehitys

Hallintaliittymä on PHP– ja MySQL–tekniikoiden pohjalle rakennettu ylläpitojärjestelmä. Uusien kuvioiden, muotokivien ja kivilajien lataaminen suunnitteluedi toriin tapahtuu täältä. Hallintaliittymä on jaettu käyttäjähallintaosioon ja tuotehallintaosioon.

5.3.1 Ensitoimenpiteet

Projektin alkuvaiheessa ei palvelin– ja tietokantatunnuksia ollut jaettu. Tästä syystä hallintaliittymän kehitys täytyi aloittaa tekemään paikalliselta koneelta käsin. Tätä varten täytyi ensin asentaa WAMP–ohjelmisto paikalliselle koneelle. WAMP:n www-kansioon luotiin oma projektikansio muotokividesigneria varten, jonne alettiin luoda tiedostoja hallintaliittymää varten.

Ylläpidettävyyden vuoksi kaikkia tiedostoja ei tallennettu samaan kansioon, vaan tiedostot jaoteltiin toiminnallisuuden mukaan omiin alikansioihinsa. Alikansiot, jotka pitivät sisällään sovelluksen toiminnan kannalta keskeisimpiä tiedostoja, suojattiin index.php–tiedostolla. Index.php on tiedosto, joka kertoo pääsyn olevan evätty. Tämä toimenpide vaikeuttaa ulkopuolisija henkilöitä näkemästä kansion hakemistorakennetta.

Web-sivut rakennettiin käyttämään konfigurointitiedostoja (*config.php*) tulevaisuuden varalle mahdollisia palvelintilan muutoksia silmällä pitäen. Nämä tiedostot pitävät sisällään tietokannan käyttäjätunnukset sekä palvelinkohtaisia hakemistopolkuja session-muuttujissa, joita itse sovellus käyttää hyväkseen. Palvelintilojen muutosten yhteydessä voidaan täten säätää hakemistopolkuja keskitetysti.

Järjestelmän sivujen lataamisen dynaamisuudesta vastaa pääkansion juuressa sijaitseva index.php -tiedosto, joka ohjaa käyttäjän oikealle sivulle. Navigoiminen sivujen välillä tapahtuu seuraavan koodin keinoin (kuva 13).

```
// Hakemisto :  www/suomenmuotokivi/index.php


include("hallinta/luokat/content.php");
$content = new Content();
$sisalto = "";
if( isset($_GET['sivu']))
{
        $sisalto = "hallinta/sisalto/".$_GET['sivu'].".php";
}
…
<div id="main">
        <?php
                //sivun tulostus:
                $content->getPage($sisalto);
        ?>
</div> <!-- end of main -->
```

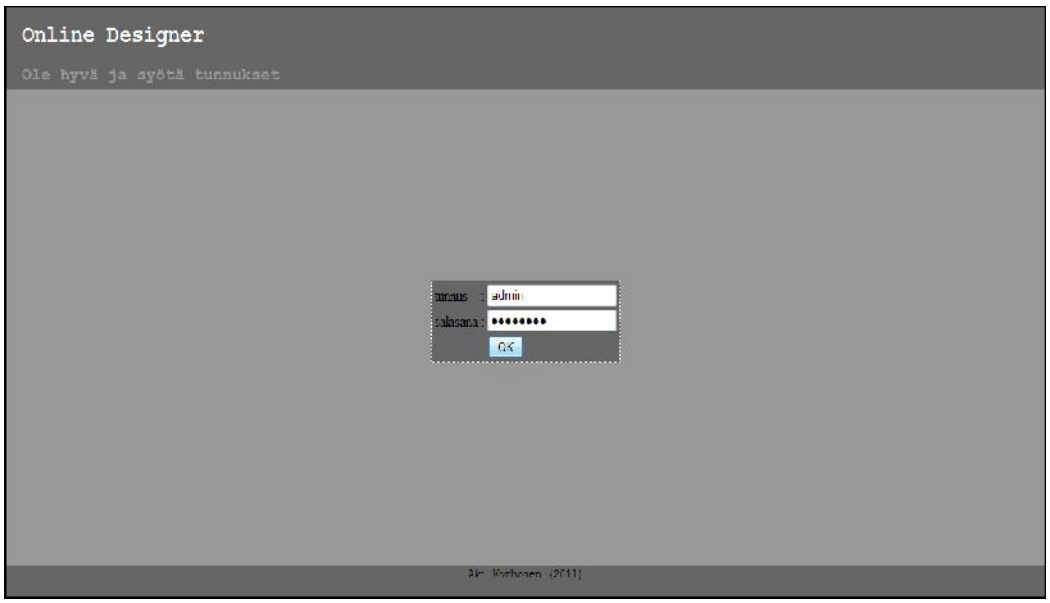
KUVA 13. Sisällöntulostus sivustolle keskitetysti

Yllä oleva koodi hakee *Content* nimisen luokan alikansiosta ja luo siitä olion. Tämän jälkeen tarkistetaan onko GET -parametreihin annettu sivutunnusta. Mikäli parametreissa on annettu sivutunnus, tallennetaan se *sisalto* nimiseen muuttujaan. Lopuksi *content*-olio hakee sivun sisällön getPage -funktiolla ja tulostaa sen näytölle. Mikäli annettu parametri on väärä tai tyhjä, ohjataan käyttäjä takaisin etusivulle.

5.3.2 Hallintaliittymä

Ylläpitäjän hallintaliittymä on turvallisuussyistä piilotettu muilta käyttäjiltä siten, että sinne pääsee suoraan vain URL:in kautta. Järjestelmä kysyy sisäänkirjautumisen yhteydessä käyttäjältä käyttäjätunnuksen ja salasanan (kuva 14).

Käyttäjän antamat syötteet sisäänkirjautumisen yhteydessä siivotaan tietoturvalliseen muotoon mysql_real_escape_string-funktion avulla ennen tietokantakyselyjen tekemistä.



KUVA 14. Hallintaliittymän sisäänkirjautuminen

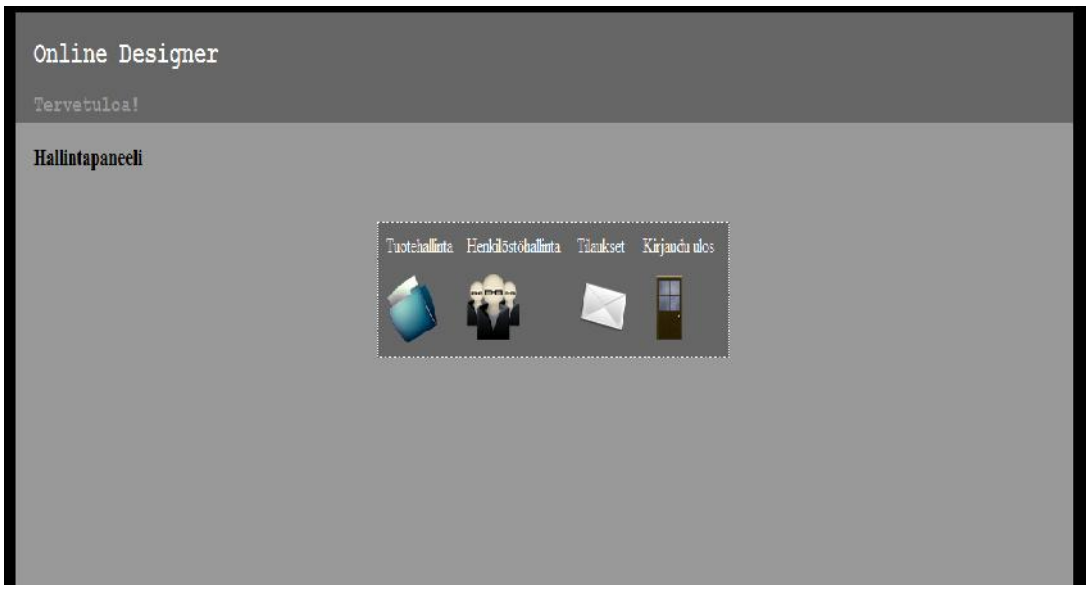
Alhaalla esimerkkikuva koodista, jossa tarkastetaan käyttäjä (kuva 15). Oikeat tunnukset syötettyään ohjataan käyttäjä hallintapaneelin etusivulle ja tieto kirjautumisesta tallennetaan session-muuttujaan.

Sisäänkirjautumisen kiertäminen selaimen osoitepalkin kautta on estetty. Ohitusyritys ohjataan automaattisesti sisäänkirjautumis-ikkunaan header() -funktion avulla, niin kauan kunnes oikeat tunnukset on syötetty. Järjestelmä todentaa sisäänkirjautumisen edellä mainitun session-muuttujan avulla. Vastaavasti uloskirjautumisen yhteydessä järjestelmä poistaa session-taulukosta tiedot sisäänkirjautumisesta. Tietojen poisto tapahtuu session_destroy() –funktiolla.

```
if( isset($_POST['laheta']) )
{


        //riisutaan sql-injektioiden varalta

        $tunnus = mysql_real_escape_string($_POST['tunnus']);
        $salasana = mysql_real_escape_string($_POST['salasana']);

        $tarkastus = $tietokanta->LoginQuery('SELECT id FROM od_admin WHERE tunnus = '".$tunnus."' AND salasana = '".$salasana."'");

        if(is_numeric($tarkastus) )
        {


                $_SESSION['kirjautunut'] = 'y';
                header('location: index.php');


        }


}
```

KUVA 15. Sisäänkirjautumisen tarkastus

Ulkoasu hallintapaneelille luotiin PHP -tiedostoihin, jossa käytettiin apuna HTML-merkkauskieltä ulkoasun rakenteen kuvauksessa. Ulkoasu koostuu div-tageista, joille annettiin omat yksilöidyt tunnukset ulkoasun muokkausta varten. Näillä tunnuksille luotiin vastinparit CSS-tiedostoon, johon lopulta annettiin yksityiskohtaiset tyylimäärittelyt ulkoasun osalta. Web-sivujen ulkoasusta ei ollut saatu määrityksiä asiakkaalta ja tästä syystä niiden ulkoasuun ei panostettu niin paljoa kuin esimerkiksi käyttöliittymään tai toiminnallisuuteen. Sivuista kuitenkin pyrittiin tekemään helposti muunneltavat tulevaisuutta ajatellen, käyttämällä selkeää merkintäkieltä CSS-tekniikan avulla ja pitämällä ulkoasua mahdollisimman minimalistisena. Onnistuneen sisäänkirjautumisen myötä käyttäjä ohjataan etusivulle (kuva 16).



KUVA 16. Ylläpitäjän etusivu

5.3.3 Tuotehallintaosio

Tuotehallintaosiosta ylläpitäjä voi lisätä, muokata ja poistaa tuotteita suunnitteluohjelmasta. Tuotteet on jaoteltu hautakiviin, kivilajeihin ja koristeisiin. Lisäykset ja muokkaukset tuottavat dataa, joka tallennetaan tietokantaan ylös.

Hautakiville ja koristeille on mahdollista luoda kategorioita, joihin voidaan jaotella tuotteita. Kategorioiden pohjalta luodaan xml-tiedostoja aina kun niihin lisätään tai poistetaan tuotteita. Suunnitteluediteri lataa näitä xml-tiedostoja aina kun käyttäjä valitsee uuden kategorian valitakseen tuotteita.

Asiakkaalta saatu jälleenmyyntihinnasto piti sisällään mainintoja lisähinnoista ja tuotteiden muutoksista. Tästä syystä järjestelmään jouduttiin työstämään erilliset osiot näitä muutoksia varten, jossa on mahdollista säätää erinäisiä hintoja ja alennuksia. Kun uudet lisäykset tietokantaan on tallennettu, rakennetaan niistä tiedoista uudet XML-tiedostot, joita itse suunnitteluediteri käyttää hyväkseen.

Itse tuotehallinnan lisäksi tässä osiossa ylläpitäjän on mahdollista luoda valmiita mallipohjia suunnitteluediteria varten. Mallipohjia luodessa käytetään ylläpitäjän suunnitteluediteria, josta tiedot tallennetaan tietokantaan.

Kun suunnitteluediteri oli upotettu sivustolle, luotiin asiakkaan antamasta tuotekuvaston pohjalta kuvat tuotteista kuvankäsittelyohjelmalla. Tämän jälkeen tuotteiden tiedot ja kuvat syötettiin järjestelmään.

5.3.4 Henkilöstöhallintaosio

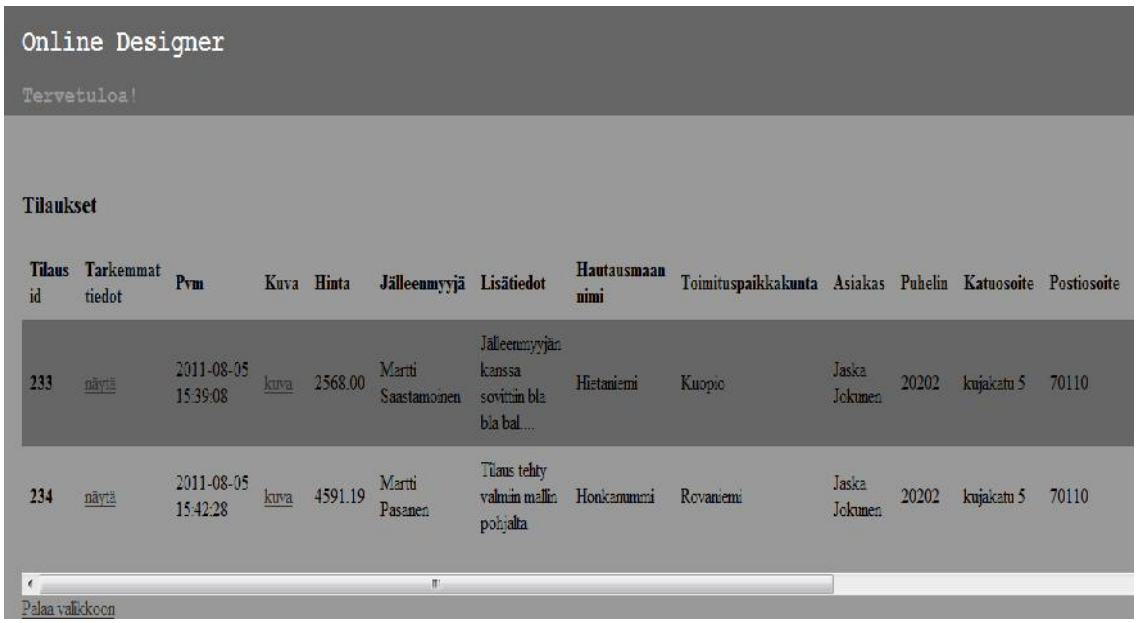
Henkilöstöhallinnassa on mahdollista lisätä järjestelmään uusia ylläpitäjiä, muokata omia ylläpitäjän tietoja sekä säätää tilauksista saatavaa provisiota. Omien tunnusten poistamista ei ole myöskään unohdettu.

Lisättäessä ylläpitäjiä järjestelmään täytyy käyttäjän täyttää lomake, jossa kysytään tunnus, salasana, etu – sekä sukunimi. Uudet ylläpitäjän tiedot tallennetaan admin-tauluun tietokantaan, kun annetut syötteet on syötetty hyväksyttävästi. Muokatessa ylläpitäjätietoja käyttäjän on mahdollista muuttaa salasanaa tai omia nimitietoja.

Säädä provisio-valikosta aukeaa näkymä, johon käyttäjä voi syöttää uuden järjestelmän käyttämän provisioprosentin. Numeerinen provisioprosentti syötetään lomakkeelle, jolloin hyväksytysti syötetty arvo tallentuu provisio-tauluun.

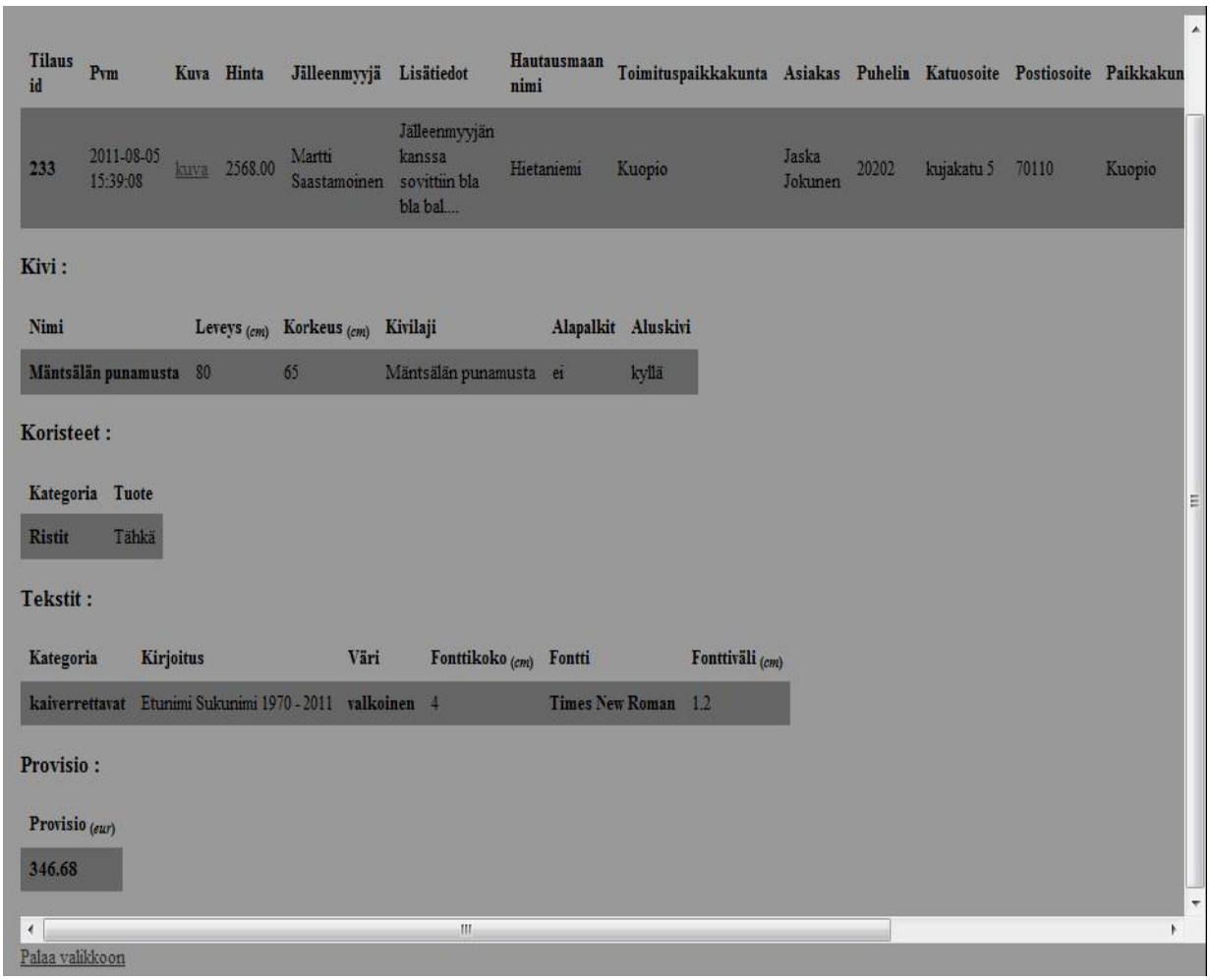
5.3.5 Tilausseuranta

Mikäli ylläpitäjä haluaa seurata tilauksia, hän menee tilausseurantaosioon. Ensin osio listaa kaikki tulleet tilaukset taulukkoon (kuva 17). Taulukkoon on listattu vain tilaus – ja henkilötiedot. Listasta on mahdollista tarkastella kuvaa asiakkaan tekemästä tilauksesta painamalla kuva -linkkiä. Tällöin avautuu uusi selainikkuna johon tulostuu kyseinen kuva.



KUVA 17. Kaikkien tilausten listaus

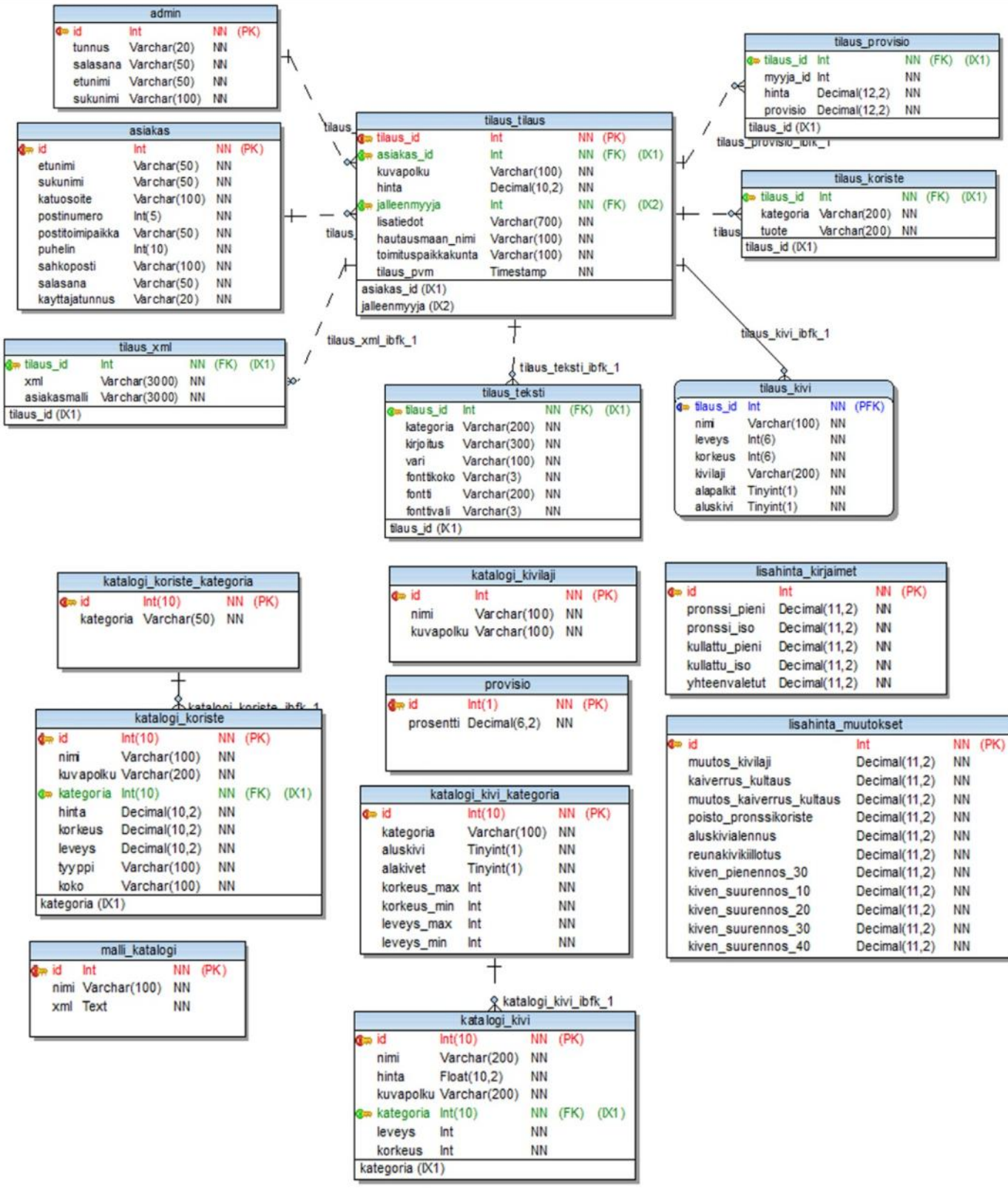
Taulukosta on mahdollista katsoa kunkin tilauksen tarkempia tietoja painamalla ”näytä” – linkkiä halutusta tilauksesta (kuva 18). Yläriviltä näkee yleiset tilaustiedot kuten edellisestäkin näkymästä. Tilaustietojen alta voi nähdä hautakiveen liittyvät tiedot kuten koon, kivilajin sekä aluskivioptiot. Hautakivitietojen alle listataan tiedot koristeista. Koristeiden tiedoista saadaan selville, mitkä koristeet on valittu hautamuistomerkkiin. Koristeiden alle on listattu teksteihin liittyvät tiedot, kuten kategoria, kirjoitus, väri, fonttikoko, fontti ja fonttiväli. Lopuksi alimmaiseksi on listattu myyjän tilauksesta saatava provisio-osuus.



KUVA 18. Tarkka listaus tilauksesta

5.4 Tietokannan kehitys

WAMP:n asennuksen jälkeen luotiin phpMyAdminilla uusi tietokanta projektia varten. Tietokannalle luotiin omat käyttäjätunnuksensa. Asiakkaan tarpeisiin pohjautuen tehtiin tietokantaan sopivat taulut. Muistomerkkidesignerin tietokanta pitää sisällään 17 taulun kokoelman. Koska järjestelmän arvioitiin tulevan matalan rasituksen käyttöön, ei nähty tarpeelliseksi toteuttaa tietokantaa kolmanteen normaalimuotoon. Alla oleva kuva havainnollistaa tarkemmin tietokannan rakennetta (kuva 19).



KUVA 19. Tietokantakuvaus

Järjestelmän tilauksista vastaa tilaus_tilaus-taulu, johon on yhteydessä tilaus_teksti-, tilaus_kivi-, tilaus_koriste-, tilaus_provisio-, tilaus_xml-, admin- ja asiakastaulu. Näihin tauluihin tallentuu asiakkaan henkilötiedot sekä tiedot tilatuista tuotteista.

Tilaus_xml-tauluun tallennetaan suunnittelueditorista tulevan tilauksen tiedot xml-muotoon, jotta sen avulla voidaan tarvittaessa ladata asiakkaan tilaus uudestaan suunnittelueditoriin muokattavaksi. Vastaavasti malli_katalogi-tauluun tallennetaan ylläpitäjän tekemät valmiit muotokivipohjat, jotka voidaan myös ladata suunnittelueditoriin. Ylläpitäjän malli poikkeaa asiakkaan mallista siten, että ylläpitäjä voi hinnoitella koko valmiin mallipohjan haluamallaan tavalla.

Jälleenmyyjän saama provisioprosentti on tallennettu provisio-tauluun. Taulusta haetaan provision prosentuaalinen suuruus, jonka avulla voidaan laskea tilauksen loppusummasta jälleenmyyjän saama osuus.

Ylläpitäjän hallintapaneelissa tuottama katalogidata tallentuu katalogi_koriste-, katalogi_kivilaji– ja katalogi_kivi-taulukoihin. Koska oli tarve jaotella koristeita ja muistokiviä omiin kategorioihinsa, luotiin näille omat taulunsa, jonne tieto kunkin koristeen tai muistokiven kategoriasta tallentui.

Huomioitavaa tietokantatauluissa on se, että kaikkia tauluja ei ole yhdistetty vierasavaimilla toisiinsa. Tämä johtuu siitä, että suunnittelueditori käyttää näitä irrallisia tauluja hyväkseen. Suunnittelueditorin lataamat tiedot siirtyvät lopulta kuitenkin tilaustietoihin.

Testiympäristön tietokannan valmistuttua luotiin tietokannasta WAMP:n graafisen hallintaliittymän avulla sql-luontitiedosto, jonka avulla voitiin siirtää valmis tietokanta asiakkaan palvelimelle. Asiakkaan palvelimelle luotiin tätä tietokantaa varten oma käyttäjänsä ja määriteltiin salasana.

5.5 Ongelmat

Projektin aikana ongelmaksi koitui Flash-kehitysympäristön ominaisuus tallentaa ulkopuolista dataa välimuistiin. Kehitysvaiheessa tämä ilmeni siten, että päivitettäessä ulkopuolisia XML-dokumentteja eivät muutokset tulleet näkyviin ilman, että välimuistia olisi tyhjennetty. Localhostilla testattaessa huomattiin, että edes välimuistin tyhjennys ei riittänyt, vaan ongelma täytyi ohittaa antamalla XML-tiedostolle parametrejä. Joka kerta ikään kuin uusi XML-tiedosto ladaan Flashia varten parametreja hyväksi käyttäen. Alla oleva koodinpätkä havainnollistaa tätä ratkaisua. Esimerkkikoodissa estetään XML-tiedoston lataamista välimuistiin. Muuttujan *_tiedosto* perään laitetaan parametriksi aikaleima, joka on jokaisella kerralla erilainen.

```
var _tiedosto:String =
"http://localhost/suomenmuotokivi/"+alamenu_koristeet.alamenu_k
oriste_kategoria.selectedItem.label+".xml?"+ new
Date().getTime();


var xmlLoader_kategoria:URLLoader = new URLLoader();

var xmlData_kuviot:XML = new XML();

xmlLoader_kategoria.addEventListener(Event.COMPLETE, LataaKate-
goria);


//suoritetaan lataus:

xmlLoader_kategoria.load(new URLRequest(_tiedosto));
```

6 TESTAUS

Tässä luvussa esitellään järjestelmän testausta. Luku on jaettu osiin suunnitteluedi-torin, hallintaliittymän ja yleisen järjestelmätestauksen kesken. Testauksia suoritti tekijän lisäksi asiakas.

6.1 Suunnitteluedi-torin testaus

Suunnitteluedi-torin testaus pohjautui asiakkaalta saatuun vaatimusmäärittelyyn, jossa oli sovelluksen suhteen lista määrityksiä (katso sivu 21). Tämä tiedosto käytiin läpi kohta kohdalta ja katsottiin, toimiko sovellus määrityksien vaatimalla tavalla. Suomen Muotokiveltä saatu vaatimusmäärittelydokumentaatio helpotti merkittävästi suunnitteluedi-torin testausta, koska sen avulla löydettiin sellaisiakin virheitä, joita ei moduulitestauksessa löydetty. Moduulitestaus suoritettiin testiympäristössä.

Jokainen käyttöliittymän komponentti käytiin yksittäin läpi ja tarkastettiin niiden toimivuus. Virhetilanteissa korjattiin ilmennyt virhe ja toistettiin testaus. Käyttöliittymän testauksessa pyrittiin karsimaan loogiset käyttöliittymävirheet. Nämä virheet pyrittiin estämään käyttäjän toimintaa ohjaamalla niin, ettei virhetilanteita pääse sattumaan. Esimerkiksi käyttäjää ei päästetä valitsemaan koristeita ennen kuin itse hautakivi on valittu. Alustavan käyttöliittymän valmistuttua suunnitteluedi-torin graafista ulkoasua testattiin asiakkaan tiloissa. Asiakkaalta saatiin näin suoraa palautetta ulkoasusta liittyen. Palautteen pohjalta oli mahdollista toteuttaa asiakkaan toivomat muutokset sovelluksen ulkoasuun.

Käyttäjän rekisteröinnissä tarkastettiin rekisteröintilomakkeeseen syötetyt arvot mahdollisten virheiden tai haittakoodin varalta. Kenttien arvot tarkastettiin sen mukaan, tuliko niiden olla numeerisia tai aakkosellisia. Virhetilanteissa käyttäjälle ilmoitetaan virheistä ja pyydetään käyttäjää antamaan arvot uudelleen. Testauksen aikana testattiin oikeita arvoja, vääriä arvoja sekä äärimmäisiä arvoja.

6.2 Hallintaliittymän testaus

Hallintaliittymään ei ollut tarjolla valmista asiakkaan vaatimusmäärittelydokumenttia eikä asiakkaalla ollut vaatimuksia testausdokumentoinnista. Tästä syystä testaus

tapahtui Ad-hoc-menetelmin. Testaus koostui toiminnallisuuden testaamisesta aina sitä mukaa kuin toteutus eteni. Testausvaiheessa pääpaino keskittyi siihen, pystyikö dataa lisäämään, muokkaamaan ja poistamaan järjestelmän sisällä.

Kun datan hallinta järjestelmän sisällä oli tarkistettu, siirryttiin tarkistamaan käyttäjän antamia syötteitä eri lomakkeilla. Käyttäjän syötteitä rajattiin lomakekomponenttien avulla. Esimerkiksi alasvetovalikoin, radiopainikkein ja valintaruuduin aina, kun tilanne salli. Käyttäjän antamat syötteet tekstikenttiin siivottiin SQL-injektioiden varalta ja tarkistettiin, olivatko ne oikeaa muotoa.

Hallintaliittymän ulkoasua rakennettaessa testattiin sen toimivuus useissa eri internetselaimissa. Mahdollisia virheitä koodissa etsittiin kytkemällä PHP:sa kaikki mahdolliset virheilmoitukset päälle funktiolla error_reporting(E_ALL).

6.3 Yleinen järjestelmätestaus

Yleinen järjestelmätestaus suoritettiin ensin testiympäristössä, kun suunnitteluedi tori ja hallintaliittymä oli yhdistetty toisiinsa upottamalla. Samalla testattiin sivuston navigoinnin käytettävyyttä. Navigointi tarkistettiin siten, että linkit sivuille olivat merkitty selvästi sekä tarkistettiin, että ne ohjasivat oikeaan osoitteeseen.

Ensimmäisenä integraatiotestauksessa tarkistettiin suunnitteluedi torin osalta, tallentuivatko tilaukset tietokantaan ja pysyikö sovelluksessa syntyvä data samana. Vastaavasti integraatio varmistettiin myös siten, että datan nouto tietokannasta suunnitteluedi toriin toimi. Seuraavaksi tarkistettiin, että suunnitteluedi torista oli mahdollista muodostaa tilauskuva. Hallintaliittymästä tarkastettiin, että data tallentui tietokantaan, sekä testattiin xml-tiedostojen luonti.

Kun kaikki tunnukset palvelimelle ja tietokantaan oli saatu, asennettiin sovellus asiakkaan palvelimelle ja suoritettiin uusi testauskierros todellisessa ympäristössä. Suunnitteluedi torin ja hallintapaneelin hakemistopolut jouduttiin päivittämään palvelimelle sopiviksi. Uudella testauksella pyrittiin varmistamaan sovelluksen toimivuus ennen sen luovuttamista asiakkaalle. Tämän jälkeen sovelluksen asennuksesta lähetettiin ilmoitus asiakkaalle, jonka jälkeen hän saattoi vielä beta-testailla palvelua ja jättää ilmoituksia mahdollisista ohjelmistovirheistä kehittäjälle.

Projektin kannalta testaus osoittautui hyödylliseksi. Useita ohjelmistovirheitä saatiin karsittua ennen sovelluksen luovuttamista asiakkaalle. Ohjelmistovirheitä olisi voitu vähentää, mikäli projektia olisi ollut tekemässä useampi henkilö, jolloin koodia olisi voinut katselmoida projektin edetessä tai ohjelmointivaihe olisi voitu toteuttaa parioh-jelmointina.

7 YHTEENVETO

Projekti tarjosi paljon uutta opittavaa ja vahvisti omaa ohjelmointitaitojani, sillä jouduin tekemään sen kokonaan yksin. Uutena asiana itselleni tuli rajapintojen toteuttaminen Flash-kehitysympäristön ja PHP:n välillä. Koulussa aiemmin opitut web-ohjelmointitaidot vahvistuivat projektin aikana entisestään.

Se että projektia tekemässä oli vain yksi henkilö, ei ollut asiakkaan kannalta paras mahdollinen tilanne, sillä ongelmatilanteissa on haastavaa saada asiantuntija-apua. Toisaalta asiakkaan oli helpompi hallita projektia, koska oli vain yksi vastuuhenkilö. Projektin aikana eniten kokemusta tuli RIA-sovelluksien tekemisestä ja yleisesti asiakaslähtöisestä projektityöskentelystä.

Hankalinta projektissa oli saada aikataulu pysymään kasassa tietokantatunnuksien puutteesta huolimatta. Oli mahdotonta asentaa sovellusta asiakkaan palvelimelle, koska sovellus nojasi vahvasti tietokantaan. Tätä puutetta paikattiin sillä, että sovellusta kehitettiin paikalliselta koneelta niin pitkälle kuin mahdollista.

Tämän kokoluokan projekteja en ollut aikaisemmin tehnyt. Perusteet projektin johtamiselle ja toteuttamiselle oli saatu koulun järjestämiltä kursseilta, mutta siitä huolimatta oli haastavaa pitää projekti hallinnassa. Projekti sijoittui hankalahkoon ajankohtaan siinä mielessä, että asiakkaalla oli kesällä menneillään sesonkiaika sekä ohjaavilla opettajilla oli menneillään lakisääteinen loma. Tämä hankaloitti hieman yhteydenpitoa sidosryhmien kesken.

Tärkeimmät annetuista tavoitteista saavutettiin. Prosessin aikana syntyi myös uusia vaatimuksia, jotka nekin pystyttiin toteuttamaan. Lopputuotoksena saatiin aikaan sovellus, jossa oli mahdollista suunnitella hautamuistomerkin ulkoasu suunnitteluedi-torilla ja tilauksesta pystyttiin tuottamaan kuva malliksi hautamuistomerkkien tekijöille. Ylläpitäjän oli mahdollista hallinnoida suunnitteluedi-torissa näkyviä tuotenimikkeitä ja hinnastoja palvelimella sijaitsevan ylläpitopaneelin kautta.

LÄHTEET

Headstones and Memorials. 2011.[verkkodokumentti] [viitattu 9.9.2011]. Saatavissa: http://www.headstonesandmemorials.com/

Kauhavan Kiviveistämö. 2011. [verkkodokumentti] [viitattu 9.9.2011]. Saatavissa: http://www.hautakivisuunnittelu.fi/

Jääskeläinen, E. 2010. Hautamuistomerkkien suunnitteluohjelma. Kuopio: Suomen Muotokivi Oy.

Moock, C. 2007. *Essential ActionScript 3.0*. New York: O'Reilly Media.

Ray, E. 2001. *Learning XML*. New York: O'Reilly Media.

Schubert-Software. 2011. [verkkodokumentti] [viitattu 9.9.2011]. Saatavissa: http://www.schubert-software.de

Sklar, D. 2004. *Learning PHP 5*. New York: O'Reilly Media.

Stat Owl. 2011. *Web Browser Plugin Market Share* [verkkodokumentti]. [viitattu 13.6.2011]. Saatavissa: http://statowl.com/plugin_overview.php

W3School. 2011. *Introduction to XML* [verkkodokumentti]. [viitattu 20.6.2011]. Saatavissa: http://www.w3schools.com/xml/xml_whatis.asp

World Wide Web Consortium. 2011. *A* History *of HTML* [verkkodokumentti]. [viitattu 22.6.2011]. Saatavissa: http://www.w3.org/People/Raggett/book4/ch02.html

World Wide Web Consortium. 1999. The CSS Saga [verkkodokumentti]. [viitattu 23.9.2011]. Saatavissa:

http://www.w3.org/Style/LieBos2e/history/