SAVONIA

OPINNÄYTETYÖ - **AMMATTIKORKEAKOULUTUTKINTO**
TEKNIIKAN JA LIIKENTEEN ALA

VERKKOKAUPPOJEN KEHITTÄMINEN

Opinnäytetyö

TEKIJÄ/T: Matti Saarinen

SAVONIA-AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Tiivistelmä

Koulutusala Tekniikan ja liikenteen ala	
Koulutusohjelma Tietotekniikan koulutusohjelma	
Työn tekijä(t) Matti Saarinen	
Työn nimi Verkkokauppojen kehittäminen	
Päiväys 6.5.2014	Sivumäärä/Liitteet 27
Ohjaaja(t) Lehtori Keijo Kuosmanen	
Toimeksiantaja/Yhteistyökumppani(t) Timo Nissinen / Carone Oy	

Tiivistelmä

Opinnäytetyön aiheena oli tehdä kaksi verkkokauppaa Carone Oy:lle. Verkkokaupoille asetettiin domainit Huonekalut.info ja toiselle Sohvat.info. Huonekalut.info korvaa vanhan Green1.fi verkkokauppan, joka myy pääasiassa huonekaluja ja sisustustarvikkeita. Kauppojen on tarkoitus toimia alussa rinnakkain ja pikkuhiljaa siirtää toiminta Huonekalut.infolle. Sohvat.info keskittyy myymään sohvia yrityksille.

Verkkokaupat toteutettiin open source -pohjaiselle Prestashop -alustalle. Verkkokauppoihin muokattiin haluttuja toimintoja, joita vanhaan Green1 -verkkokauppaan ei voitu toteuttaa. Lisäksi vanha verkkokauppa ei ole yhteensopiva uudempien PHP -versioiden kanssa, mikä lisäsi tarvetta uudelle verkkokaupalle. Koska Prestashop on tehty suurimmaksi osaksi PHP:llä, työssä käytettiin pääasiallisena kehitystyökaluna Programmer's Notepadiä, jolla tehtiin suurin osa muokkauksista, kuten lähdekoodi, scripti ja tyyli tiedostoihin.

Projektin edetessä tuli vastaan muutamia ongelmia, joten aikataulut viivästyivät. Tästä johtuen vain Huonekalut.info saatiin valmiiksi ja julkaisukuntoon. Vaikka Sohvat.infoa ei vielä valmiiksi ehditty saamaan, suuria muutoksia siihen ei kuitenkaan tarvitse tehdä, sillä siinä käytetään paljon samoja toiminnallisuuksia kuin Huonekalut.infossa, joten suurinosa muokkauksista tehdään ulkoasuun.

Avainsanat
PHP, verkkokauppa, Prestashop

SAVONIA UNIVERSITY OF APPLIED SCIENCES

THESIS
Abstract

Field of Study Technology, Communication and Transport			
Degree Programme Degree Programme in Information Technology			
Author(s) Matti Saarinen			
Title of Thesis Development of e-Commerces			
Date	6 May 2014	Pages/Appendices	27
Supervisor(s) Mr Keijo Kuosmanen, Lecturer			
Client Organisation /Partners Mr Timo Nissinen / Carone Oy			

Abstract

The purpose of this thesis was to create two e-commerce sites for Carone Oy. The domains had already been re-served for both of the sites, named Huonekalut.info and Sohvat.info. Huonekalut.info will replace the old Green1.fi e-commerce site, which mainly sells furniture and interior decorations. Huonekalut.info and Green1.fi are intended to operate in parallel and gradually transfer operation to Huonekalut.info. Sohvat.info focuses on selling sofas for businesses.

The sites were implemented on an open-source based platform named Prestashop. Sites were modifed to meet the requirements of the desired functions, which could not be implemented in Green1.fi. In addition, the old Green1.fi site is not compatible with newer versions of PHP, which increased the need for a new sites. Since Prestashop was made mostly with PHP, Programmer's notepad was chosen as the main development tool for this project. Most of the source code, script and style files were modified by Programmer's notepad.

As the project progressed few problems occurred so the schedule was delayed. Because of this only Huonekalut.info was completed and published. Although Sohvat.info is not yet published there will be no need for make major changes because both of the sites use much of the same functionalities. So most of the modifications will be made on the layout.

Keywords
PHP, e-commerce, Prestashop

SISÄLTÖ

1 JOHDANTO 5

2 KÄYTETYT TEKNIIKAT JA OHJELMISTOT 6

2.1 PHP & Smarty 6

2.2 HTML & CSS 7

2.3 JavaScript & jQuery 8

2.4 WAMP 8

2.5 Programmers Notepad & Visual Studio 9

3 ALKUTILANNE 10

3.1 Vanha verkkokauppa 10

3.2 Uusi verkkokauppa 10

4 PRESTASHOP 11

4.1 Prestashopin edut 11

4.2 Moduulit 11

4.2.1 PHP-päätiedosto 12

4.3 Teemat 15

4.4 Hakukoneoptimointi 15

5 TOTEUTUS 17

5.2 Tuotteiden tuonti tietokantaan 17

5.3 Käännökset 18

5.4 Päivitys 19

5.5 Kauppaan muokatut toiminnot 19

5.5.1 viimeistelyn valinta 19

5.5.2 Keskustelun viestit 20

5.5.3 Sähköpostit 21

5.5.4 Salasanan varmistus 23

5.5.5 Saatavuus päivämäärä 23

6 YHTEENVETO 25

LÄHTEET 26

1 JOHDANTO

Työn tavoitteena oli luoda kaksi verkkokauppaa Carone Oy:lle. Verkkokaupat luotiin Prestashop-alustalle. Yksi verkkokauppa korvaa yrityksen vanhan verkkokaupan nimeltään Green1.fi. Toinen verkkokauppa keskittyy myymään sohvia muille yrityksille.

Yrityksellä on jo olemassa verkkokauppa nimeltään Green1.f1, joka myy huonekaluja. Verkkokauppa on tehty vanhalle OsCommerce 2.2 -pohjalle.

Opinnäytetyön aihe syntyi, kun vanhaa järjestelmää käyttäessä huomattiin tarve uusille toiminnoille, joita vanhassa verkkokauppassa ei enää voitu toteuttaa. Lisäksi kyseinen OsCommercen versio ei ole yhteensopiva PHP -versioiden 5.4 ja uudempien kanssa.

Työn tavoitteena on julkaisuvalmiit verkkokaupat. Molemmat verkkokaupat asennetaan Nettihotellin webhotellipalveluun. Green1:n korvaavalle kaupalle asetetaan huonekalut.info -domain ja yrityskäyttöön tulevalle verkkokaupalle domain nimeltä sohvat.info.

2 KÄYTETYT TEKNIIKAT JA OHJELMISTOT

Tässä osiossa käydään läpi projektissa käytetyt tekniikat ja ohjelmistot. Tekniikoissa esiintyvät koodinpätkät eivät liity suoraan projektiin, vaan ovat pelkkiä esimerkkejä kyseisistä tekniikoista.

2.1 PHP & Smarty

PHP (Hypertext Prepprocessor) on suosittu yleiskäyttöinen skriptikieli, joka toimii Web-palvelinympäristöissä ja näin ollen soveltuu erityisesti dynaamisten web-sivujen luomiseen. PHP on nopea, joustava ja käytännöllinen kieli ja sitä voidaan käyttää useilla eri alustoilla ja käyttöjärjestelmillä. OsCommerce ja Prestashop on tehty PHP kielellä. (PHP 20.11.2013.)

```
1. $text = (IsVariableSomething($variable))
2.                 ? Constants::VARIABLE_IS_SOMETHING
3.                 : Constants::INVALID_VARIABLE;
4. echo $text;
```

Kuva 1. PHP esimerkki lyhennetystä if-elsestä. Tarkastetaan `$variable` muuttuja `IsVariableSomething` funktiossa ja tulostetaan lopputulos.

Smarty

Smarty on PHP-kielellä kirjoitettu mallinnejärjestelmä, joka erottaa ohjelmakoodin esittämisestä (eli PHP:n HTML/CSS:stä). Tämän ansiosta koodi on selkeää ja ulkoasua voidaan muokata ohjelmakoodiin koskematta. Prestashopin kaikki näkymät on tehty Smartyllä. (Smarty template engine 20.11.2013).

Smarty -tiedosto sisältää HTML:ää, javascriptiä ja tyyleja aivan kuten normaali .htm -tiedosto. Mutta toisin kuin .htm -tiedostot, Smarty suoritetaan palvelimen päässä PHP:n tavoin, jolloin Smartylla voidaan suorittaa PHP:llä tehtyjä funktioita, sekä käyttää Smarty -muuttujia, jolloin voidaan tehdä sivusta dynaaminen ja samaan aikaan pitää ulkoasu erillään logiikasta.

```
1. {if ($name|name_is_legit)}
2.         <p>name was legit</p>
3. {else}
4.         <p>name wasn't legit</p>
5. {/if}
```

Kuva 2. Smartyn if-else esimerkki. `$name` muuttuja tarkastetaan `name_is_legit` funktiossa.

2.2 HTML & CSS

HTML (Hyper Text Markup Language) on eräänlainen kuvauskieli. HTML dokumentti (webbisivu) koostuu HTML tageista ja tavallisesta tekstistä. (HTML 30.12.2013.)

```
<table border="1">
        <tr>
                <th> First name </th>
                <th> Last name </th>
        </tr>
        <tr>
                <td> Matti </td>
                <td> Saarinen </td>
        </tr>
</table>
```

output:

Firstname	Lastname
Matti	Saarinen

Kuva 3. HTML:llä tehty taulukko.

CSS (Cascading Style Sheets) on yksinkertainen tyylin lisäysmekanismi web-asiakirjoja varten. Yksinkertaistettuna PHP määrittää, mitä sivulla on. HTML näyttää sivun ja lopuksi CSS määrittää, miten sivu näytetään. (CSS 20.12.2013.)

```
<link rel="stylesheet" type="text/css" href="style.css">

<table border="1" id="myTable">
```

Kuva 4. CSS -tyylit voi kirjoittaa suoraan html -elementin attribuutteihin tai tyylit voi kirjoittaa omaan .css tiedostoon, johon viitataan koodissa.

style.css

```
#myTable tr td
{
        text-align:center;
}

#myTable tr td:first-child
{
        background-color:#666;
        color:#fafafa;
}
```

output

First name	Last name
Matti	Saarinen

Kuva 5. CSS:n vaikutus HTML taulukkoon

2.3 JavaScript & jQuery

Javascript on komentosarjakieli, joka toimii käyttäjän koneella, joten se ei vaadi jatkuvaa yhteyttä sivustoon. Javascriptin avulla saadaan Web-sivuille lisättyä dynaamista toiminnallisuutta. (JavaScript 20.12.2013.)

jQuery on javascriptin pieni ja kevyt kirjasto, mutta jossa kuitenkin on paljon toimintoja. Muutamia jQueryn tärkeitä ominaisuuksia ovat tapahtuman käsittelyt, animaatiot sekä ajax-toiminnot. Prestashopissa käytetään paljon jQueryä. Varsinkin ajaxia käytetään paljon, sillä sen ansiosta voidaan vaihtaa tietoa serverin välillä ja päivittää vain osaa sivun sisällöstä. Näin ollen koko sivua ei tarvitse turhaan ladata uudestaan. (jQuery 14.1.2014.)

Clientin päässä tapahtuva scripti

```
$.ajax({ url: 'Methods/JobMethods.php',
    data: {jobIdDelete: id},
    type: 'post',
    success: function(result) {
        // return       1 - succesful
        //              2 - SQL error
        //              3 - user not logged
        //              4 - wrong user
        switch (parseInt(result))
        {
        case 1:
            alert("Job Deleted successfully");
            location.reload();
            break;
        case 2:
            alert("ERROR");
            break;
        case 3:
            alert("NOT LOGGED IN");
            break;
        case 4:
            alert("Y U HAX?");
            break;
        default:
            alert("dafuq?");
            break;
        }
    }
});
```

Serverin päässä sijaitseva kutsuttava metodi

```
// summary:     Delete selected job
// $id:         id of the job that will be deleted
// return       1 - succesful
//              2 - SQL error
//              3 - user not logged
//              4 - wrong user
else if (isset($_POST['jobIdDelete']))
{
    $id = $_POST['jobIdDelete'];
    require_once("../db.inc");
    require_once("AccountMethods.php");
    $retval = -1;
    session_start();
    if (isset($_SESSION['logged']))
    {
        if(IsJobOwner($id, true) || IsAuthenticated())
        {
            $query = sprintf("Delete FROM job WHERE id='%s'", $id);
            $result = mysql_query($query);
                        .
                        .
                        .
```

Kuva 6. Esimerkki jQueryllä toteutetulla ajax -kutsusta

2.4 WAMP

WAMP (windows, apache, mysql, PHP) on web-kehitysympäristö, joka koostuu useiden ohjelmien ohjelmistokokonaisuudesta. Kokoelma muodostaa WWW-palvelimen, jonka alla voidaan ajaa dynaamisia websivuja. WAMP sisältää Apache ja MySQL avoimen lähdekoodin ohjelmat. Apache on webpalvelin ja MySQL on tietokantarajapinta. WAMP:ssa käytetään PHP, Perl ja/tai Python avoimen lähdekoodin komentosarjakieliä.

Projektit tehtiin ensin paikalliselle WAMP -serverille, jossa verkkokauppoja muokattiin halutunlaisiksi ja tämän jälkeen ne vasta asennettiin niille tarkoitetulle serverille. (WAMP 23.12.2013.)

Apache

Apache on avoimen lähdekoodin webpalvelin, joka toimii usealla eri alustalla (mm. Linux, Solaris, Digital UNIX sekä Windows). Apachea käytetään yli 60 % webpalvelimissa, eli enemmän kuin muita yhteensä. (Apache 23.12.2013.)11

MySql

MySQL on suosituin avoimen lähdekoodin SQL-tietokannan hallintajärjestelmä. Niin OsCommerce kuin Prestashopkin käyttävät MySQL:ää tietokantanaan. (MySQL 23.12.2013.)

2.5 Programmer's Notepad & Visual Studio

Programmer's Notepad on avoimen lähdekoodin tekstieditori, joka on tarkoitettu lähinnä lähdekoodien kanssa työskentelyyn. Programmer's Notepad oli tärkein opinnäytetyössä käytetty kehitystyökalu. Kaikki muokkaukset ja lisäykset PHP:hen, Smartyyn, Javascriptiin ja CSS koodeihin tehtiin Programmer's Notepadiä käyttäen. (Programmer's Notepad 23.12.2013.)

Visual Studio on kattava kokoelma työkaluja ja palveluja sovellusten kehittämistä varten. Visual Studiota käytettiin projektissa C# -ohjelman tekemiseen, jolla saatiin vanhan verkkokaupan tietokannan tuotetiedot kasattua yhteen tiedostoon tuontia varten.(Visual Studio 23.12.2013.)

3 ALKUTILANNE

3.1 Vanha verkkokauppa

Yrityksen jo olemassa oleva verkkokauppa oli tehty vanhalle OsCommerce 2.2 -pohjalle ja, koska pohjaan oli tehty paljon muutoksia, ei verkkokauppaa ole pystytty päivittämään uudempiin versioihin. Kyseinen OsCommercen versio ei ole yhteensopiva PHP 5.4 kanssa, joka on käytössä valtaosalla palveluntarjoajista. Lisäksi vanhaa järjestelmää käytettäessä on tullut tarve uusille toiminnoille, joita OsCommercessä ei ole mahdollista toteuttaa.

Tuoteominaisuuksien esittäminen oli yksi tärkeimmistä halutuista ominaisuuksista. OsCommersessa tuoteominaisuudet voi esittää vain alasvetolaatikoissa, joka ei ole kovin kätevä tapa esimerkiksi sohvien kankaiden esittämisessä. Yleisessä käytettävyydessä on myös OsCommercessa parannettavaa, varsinkin hallinnon puolella. Esimerkiksi tuotteen lisäys on tehty yhdelle sivulle ja jos tuoteominaisuuksia on paljon, sivu on varsin epäselvä.

OsCommersesta löytyy toiminto, jossa tuotteelle voi asettaa saatavuuspäivämäärän. Mutta mikäli tuotteella on useampia tuoteominaisuuksia, saatavuuspäivämäärää ei voi määrittää ominaisuuskohtaisesti.

3.2 Uusi verkkokauppa

Koska vanhalla verkkokaupalla on yhteensopivuusongelmia uusien PHP-versioiden kanssa ja verkkokaupan toiminnoilla olisi parantamisen varaa, päätettiin rakentaa kokonaan uusi verkkokauppa. Vaihtoehtoina verkkokaupan alustaksi oli uudempi OsCommerce tai Prestahop. Verkkokauppa päätettiin rakentaa Prestashopin pohjalle, koska se on modernimpi alusta kuin OsCommerce ja Prestashop tarjoaa paljon enemmän toimintoja.

Vaikka Prestashop on yleisesti ottaen paljon monipuolisempi alusta kuin OsCommerce, puuttuu siitä esimerkiksi saatavuuspäivämäärä-toiminto, joka OsCommercesta löytyy oletuksena. Kyseinen toiminto voidaan tosin rakentaa Prestashoppiin kätevästi omaan moduulin.

Verkkokauppoja varten oli varattu kaksi domainia; huonekalut.info ja sohvat.info, joista huonekalut.info korvaa vanhan green1.fi verkkokaupan ja sohvat.info domainissa oleva verkkokauppa on yrityksiä varten.

4 PRESTASHOP

4.1 Prestashopin edut

Prestashop on ilmainen opensource -pohjainen verkkokauppa. Prestashop on tehty PHP:llä ja se perustuu Smarty -mallinnejärjestelmään. Tietokantana toimii MySQL. Prestashop on käytössä yli 165 000 kaupassa maailmanlaajuisesti.

Prestashop on moderni ja monipuolinen alusta, joka on rakennettu MVC -mallia noudattaen. MVC-malli (Model-View-Controller) eli suomennettuna malli-näkymä-käsittelijä on ohjelmistoarkkitehtuurimalli, jonka tarkoituksena on pitää käyttöliittymä (näkymä) erillään ohjelman logiikasta.

Prestashop käyttää teemoja ja moduuleita. Moduulien ansiosta kauppaa voi laajentaa ja muokkailla joko valmiita mooduuleita käyttämällä tai rakentamalla omia. Ulkoasu taas on vapaasti muokattavissa teemojen ansiosta. Koska muutokset tehdään moduuleja ja teemoja käyttäen, voidaan verkkokauppa päivittää uudempiin versioihin helposti, vaikka oletuskauppaa oltaisiin muokattu vahvasti. (Prestashop, Prestashop 1.5 documentation 20.1.2014.)

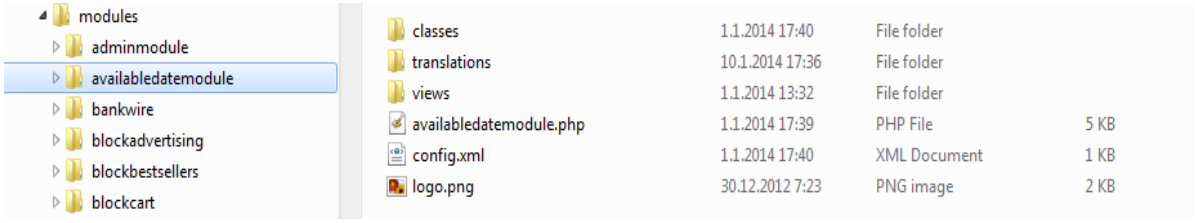
4.2 Moduulit

Prestashop painottuu moduulien käyttöön. Siinä on paljon hyviä puolia. Koska moduulit toimivat itsenäisesti, mm. Prestashopin päivitys uudempiin versioihin on varsin helppo toimenpide. Moduulien ansiosta voidaan kauppaa laajentaa lisäämällä siihen haluttuja moduuleja, jotka toimivat itsenäisesti.

Moduulit sijaitsevat Prestashopin modules -hakemistossa. Kyseiseen hakemistoon luodaan jokaiselle moduulille oma juurihakemisto, joka nimetään moduulin mukaan ja se pitää sisällään moduulin tiedostot. Ainoa pakollinen tiedosto moduulissa on PHP-päätiedosto, joka on nimetty moduulin mukaan.

Moduulin juurihakemistossa sijaitsee myös kaikki moduulin metodit/luokat, jotka esimerkiksi hoitavat yhteydet tietokantaan. Hakemistossa on yleensä myös .tpl -tiedostoja, jotka määrittävät moduulin ulkoasun eli teeman. Mikäli moduulilla on näytettävää tekstiä, Prestashop luo kielitiedostot moduulin translations -alihakemistoon. Kielitiedoston avulla moduulin kääntäminen eri kielille on mahdollista ilman, että kajotaan lähdetiedostoon.

Kieli- ja .tpl -tiedostot voidaan halutessa myös sijoittaa prestashopin teeman hakemiston `/themes/modules` –kansioon, joka on saman niminen kuin moduuli. Mikäli Prestashopilla on useampia teemoja käytössä, voidaan moduulin ulkoasu asettaa jokaisen kaupan teeman mukaisesti.



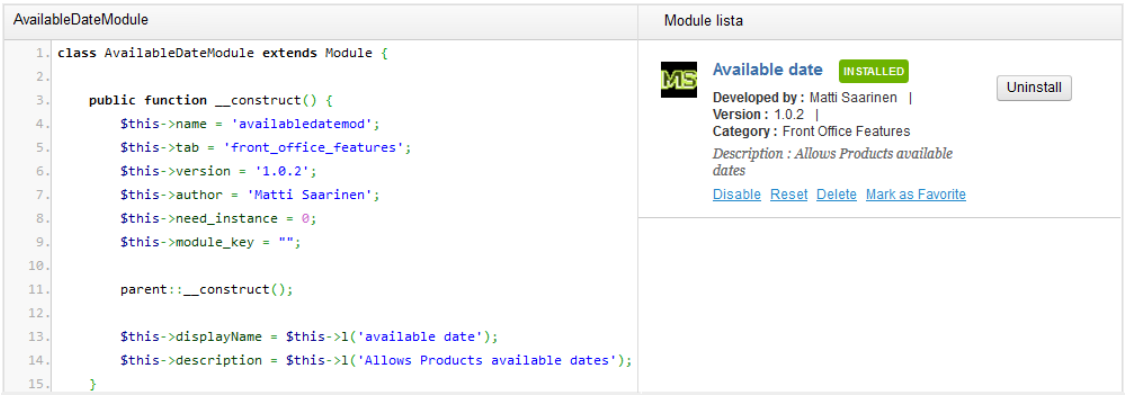
Kuva 7. Availabledatemodulen sisältö

4.2.1 PHP-päätiedosto

Moduulin PHP-päätiedosto voidaan rinnastaa MVC-mallin käsittelijään. Päätiedosto käyttää muita moduulin tiedostoja. PHP-päätiedostossa on aina vähintään construct-, install- ja uninstall-funktiot, sekä koukku-funktioita.

Construct

Construct-funktiossa asetetaan moduulin perustiedot, mm. moduulin nimi, versio ja lyhyt kuvaus.



Kuva 8. PHP-päätiedostoon asetaan ensin moduulin tiedot

Install

Install-funktio kutsutaan silloin, kun moduuli halutaan asentaa. Mikäli funktio käyttää tietokantoja, määritetään tarpeelliset taulut ja sen kentät asennuksen yhteydessä. Lisäksi install-funktiossa määritetään koukut, joita moduuli käyttää.

```php
public function install() {
    $sql = array();

    $sql[] = 'CREATE TABLE IF NOT EXISTS `'._DB_PREFIX_.'AvailableDatemod` (
                `id_AvailableDatemodule` int(10) unsigned NOT NULL AUTO_INCREMENT,
                `id_product` INT( 11 ) UNSIGNED NOT NULL,
                `textarea` TEXT NOT NULL,
                `textattr` TEXT NOT NULL,
                PRIMARY KEY (`id_AvailableDatemodule`),
                UNIQUE  `AvailableDate_UNIQ` (  `id_product` )
              ) ENGINE='._MYSQL_ENGINE_.' DEFAULT CHARSET=utf8';

    if (!parent::install() OR
        !$this->registerHook('displayAdminProductsExtra') OR
        !$this->registerHook('actionProductUpdate') OR
        !$this->registerHook('displayProductButtons') OR
        !$this->registerHook('productOutOfStock') OR
        !Configuration::updateValue('sample_module_textarea', '') OR
        !$this->runSql($sql)
    ) {
        return FALSE;
    }

    return TRUE;
}
```

Kuva 9. Kyseisessä install -funktiossa luodaan tietokantaan 'AvailableDatemod' -taulu ja sille kentät, sekä rekisteröidään koukut, joihin moduuli voidaan kiinnittää

Uninstall

Uninstall-funktiossa koodi suoritetaan moduulin poiston yhteydessä, mm. moduulia varten asennettu tietokannan taulu poistetaan uninstall-moduulissa.

```php
public function uninstall() {
    $sql = array();

    $sql[] = 'DROP TABLE IF EXISTS `'._DB_PREFIX_.'AvailableDatemod`';
    if (!parent::uninstall() OR
        !$this->runSql($sql)
    ) {
        return FALSE;
    }

    return TRUE;
}
```

Kuva 10. Funktiossa poistetaan tietokannasta 'AvailableDatemod' -taulu.

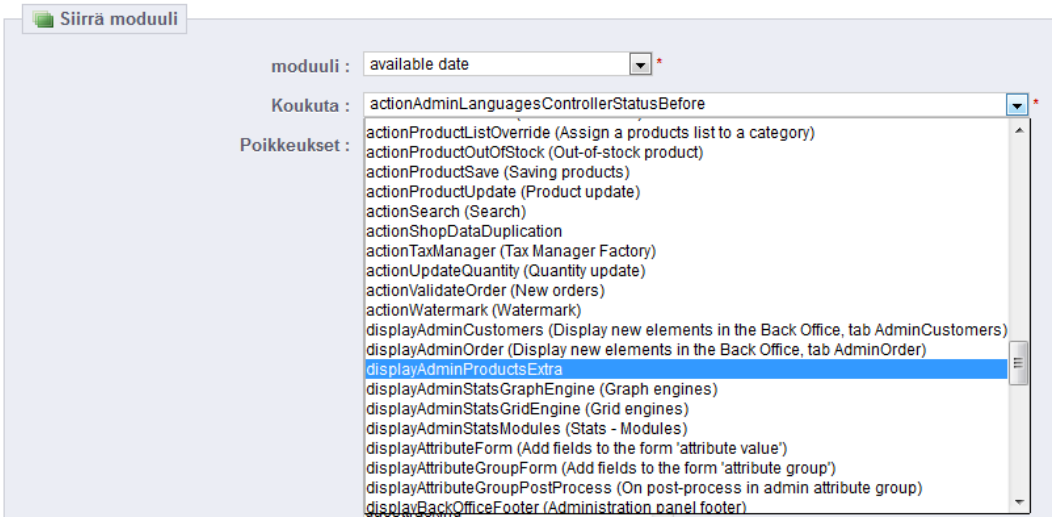
Koukut

Prestashopin koukut ovat eräänlaisia kiinnityspisteitä, joilla määritetään missä moduuli näytetään tai milloin sitä kutsutaan. Koukkuja on kahdenlaisia: display ja action -koukkuja. Display-koukut määrittävät sijainnin sivustolla, jossa moduuli näytetään. Action-koukut ovat sen sijaan eräänlaisia eventtejä, jotka määrittävät milloin moduulia kutsutaan. Esimerkiksi `ActionProductUpdate` -koukkua kutsutaan silloin, kun tuote päivitetään.

```
101     public function hookDisplayAdminProductsExtra($params) {
102         $id_product = Tools::getValue('id_product');
103         $sampleObj = AvailableDate::loadByIdProduct($id_product);
104         if(!empty($sampleObj) && isset($sampleObj->id)){
105             $this->context->smarty->assign(array(
106                 'available_text' => $sampleObj->textarea,
107                 'text_attr' => $sampleObj->textattr,
108             ));
109         }
110 
111         return $this->display(__FILE__, 'views/admin/frontview.tpl');
112     }
```

Kuva 11. Oheisessa funktiossa määritetään displayProductButtons -koukku, sekä smarty muuttujat.

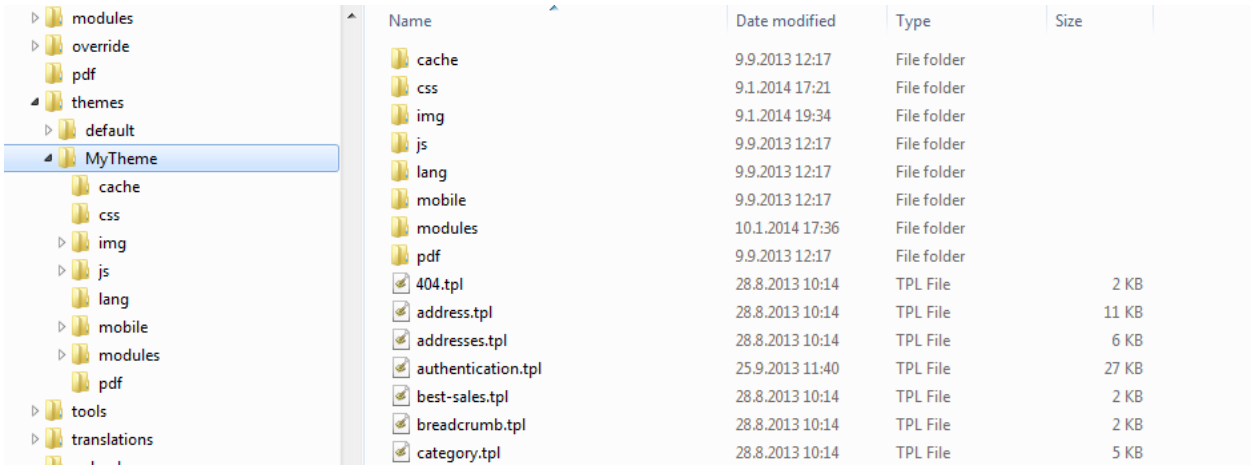
Halutut koukut määritetään PHP-päätiedostossa. Koukun sisällä voidaan luoda sitten Smarty-muuttujia, joita voidaan käyttää moduulin .tpl -tiedostossa. Koukkuja voidaan määritellä useampi kappale, sillä Prestashopin hallinnossa valitaan haluttu koukku, jota voidaan vaihtaa milloin vain, kunhan valittu koukku on määritetty moduulissa.



Kuva 12. Kuvassa näkyy lista koukuista, joihin valittu moduuli voidaan asettaa.

4.3 Teemat

Teema määrittää verkkokaupan ulkoasun. Teemoja voi olla useampia, joista yksi on aina päällä. Prestashopissa on omat teemat kaupan sekä hallinnon puolelle. Kaupan teemat sijaitsevat `themes` -hakemistossa ja hallinnon teemat `admin/themes` -hakemistossa.



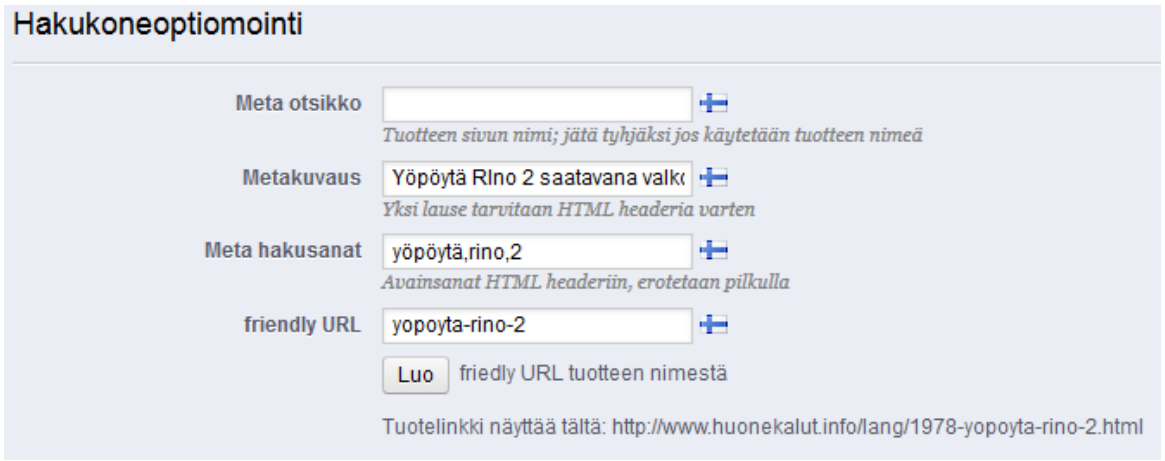
Kuva 13. Yleiskatsaus teeman rakenteesta.

Teema pitää sisälllään Smarty-tiedostoja (.tpl), kuvatiedostoja, tyylitiedostoja, sekä javascript-tiedostoja.

Koska kaikki näkymät toteutetaan teemojen avulla, voidaan ulkoasua muokata koskematta kaupan logiikkaan. Näin ollen kauppaa voidaan helposti muokata halutun näköiseksi ja se voidaan päivittää uudempiin versioihin ongelmitta.

4.4 Hakukoneoptimointi

Prestashop optimoi sivuston ja varmistaa, että suuret hakukoneet indeksoivat myymälän. Optimointi tapahtuu metakuvauksilla ja avainsanoilla, sekä tageilla. Kyseiset tiedot voidaan asettaa jokaiselle tuotteelle erikseen.

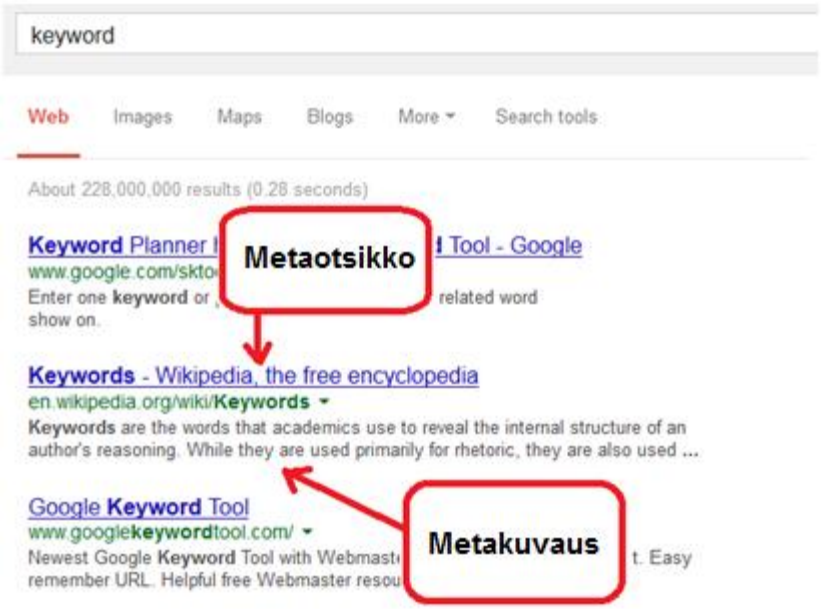


Kuva 14. Prestashopin SEO - hakukoneoptimointi

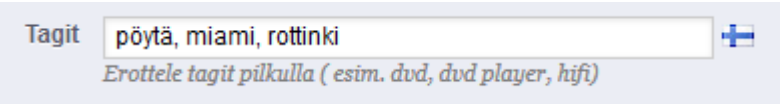
```
<meta name="description" content="Techniques used in this project">
<meta name="keywords" content="HTML,CSS,JavaScript,php,Smarty,jQuery">
```

Kuva 15. Esimerkki metaheaderin metatageista.

Metaotsikko ja -kuvaus ovat tärkeitä hakukoneiden indeksointia varten, mutta metahakusanat alkavat olla jo historiaa. Ainoa varteenotettava hakukone, joka vielä käyttää metahakusanoja on Microsoftin Bing. (Meta Tags 20.1.2014.)



Kuva 16. Metatagien havainnollistaminen



Kuva 17. Tuotteen tagit helpottavat tuotteiden löytämistä kaupan haku-toiminnolla.

Prestashop luo jokaiselle tuotteelle oman URL:n. Näin vältytään ikäviltä duplikaateilta, jotka häiritsevät hakukoneiden indeksointia.

5 TOTEUTUS

Työ toteutettiin etätyönä ja se aloitettiin ensin tutustumalla eri verkkokaupparatkaisuihin (OsCommerce ja Prestashop). Sen jälkeen kun Prestashop valittiin tulevaksi pohjaksi asennettiin se ensin paikalliselle WAMP serverille, jotta siihen pystyi helposti tutustumaan ja kokeilemaan muutoksia.

Kun tarpeelliset toiminnot ja alustava ulkoasu oli tehty, asennettiin uudet prestashop-asennukset oikeille palvelimille ja siirrettiin muokatut tiedostot paikalliselta WAMP-serveriltä Nettihotellin palvelimille. Kaikki muokkaukset, teemat ja moduulit tehtiin ensin paikallisella palvelimella, sekä testattiin ne, tämän jälkeen ne siirrettiin oikeille palvelimille ja testattiin vielä, että ne varmasti toimivat.

Työn edetessä aina välillä järjestettiin tapaaminen työnantajan kanssa ja tarkasteltiin missä mennään, mitä uusia toimintoja tarvitaan, sekä tarvitseeko joitain vanhoja toimintoja muokata.

5.1 Tuotteiden tuonti tietokantaan

Prestashopiin on rakennettu tuonti-toiminto, mutta koska vanha verkkokauppa on tehty OsCommerce-pohjalle, sitä ei voi suoraan käyttää. Tätä varten tehtiin C#-ohjelma, joka muokkaa OsCommercen tauluista yhtenäisen tuotteet-tiedoston, joka syötetään Exceliin ja tallennetaan .CSV-muotoon, jota voidaan käyttää Prestashopin tuonti-toiminnossa.

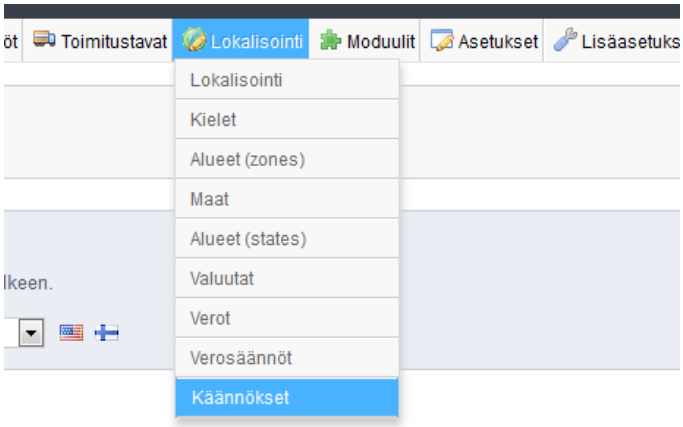
```
44          TextWriter tw = new StreamWriter(FILE_NAME, false, Encoding.GetEncoding(ENCODING));
45          string product = string.Empty;
46          string id = string.Empty;
47
48          for (int i = 1; i < tmpID.Count(); i++ )
49          {
50              id = tmpID[i].FirstOrDefault();
51              product += (tmpID.GetStatus(id) + NEW_LINE);        // ENABLE
52              product += (tmpDesc.GetName(id) + NEW_LINE);        // NAME
53              product += (tmpCat.GetGategory(id) + NEW_LINE);     // CATEGORIES
54              product += (tmpID.GetPrice(id) + NEW_LINE);         // PRICE
55              product += TAX_RULE;                                // TAX RULE
56              product += (tmpID.GetReference(id) + NEW_LINE);     // REFERENCE
57              product += (tmpID.GetQuantity(id) + NEW_LINE);      // QUANTITY
58              product += (tmpDesc.GetDescription(id) + NEW_LINE); // DESCRIPTION
59              product += NEW_LINE;
60
61              tw.WriteLine(product);
62              product = string.Empty;
63          }
64          tw.Close();
65          Console.WriteLine(DONE);
```

Kuva 18. Näytekoodi tiedostoon kirjoittamisesta, C#:lla tehdystä parserista.

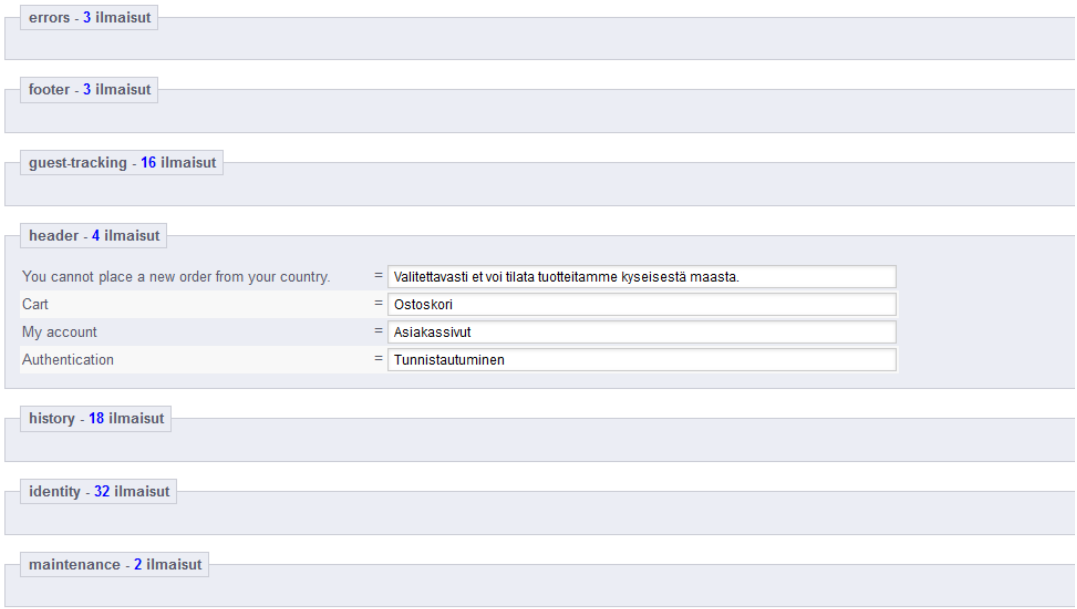
Koska Prestashop on rakenteellisesti hyvin erilainen OsCommersesta, kaikkia tietoja ei valitettavasti saatu vietyä ohjelman avulla. Iso osa tiedoista piti lisätä manuaalisesti jälkeenpäin, jonka takia projektin aikataulu viivästyi.

5.2 Käännökset

Prestashopilla on paljon valmiita kielipaketteja, mutta moni niistä ei ole täysin valmis. Suomen kielipakettista oli käännetty n. 70%. Lisäksi osa käännöksistä oli virheellisesti käännetty. Prestashopissa käännöksiä pystyy muokkaamaan sekä lisäämään kätevästi hallintapaneelin käännökset-osiossa.



Kuva 19. Hallintapaneelin käännökset valikko



Kuva 20. Käännökset on jaoteltu moduuleittain, jotta käännösten löytäminen olisi helpompaa ja loogisempaa.

5.3 Päivitys

Ulkoasua varten kaupoille tehtiin omat teemat. Suurinosa logiikkaan tehdyistä muokkauksista toteutettiin tekemällä oma moduuli tai käyttämällä override-toimintoa luokka- ja ohjain -tiedostoja muokatessa. Muutama toiminto piti tehdä olemassa oleviin moduuleihin, jonka vuoksi päivittämisen yhteydessä kyseiset muokkaukset täytyy tehdä uudestaan.

Pretashop 1-click-upgrade on Prestashopin oma moduuli, jonka avulla kaupan voi päivittää uusimpaan versioon. Ennen päivittämistä on kuitenkin hyvä ottaa varmuuskopiot tiedostoista, sekä tietokannasta, sillä jos ongelmia ilmenee, voidaan aiempi versio palauttaa. 1-click-upgrade-moduuli kirjoittaa uudet tiedostot vanhojen päälle, jolloin niihin tehdyt muutokset häviävät. Tämän vuoksi on tärkeää pyrkiä tekemään muokkaukset omiin moduuleihin ja teemoihin.

1-click-upgrade moduulia käytettiin kahdesti verkkokaupan kehittämisen aikana. Ensimmäinen päivitys oli varsin pieni ja päivittäminen kävi varsin helposti. Toinen päivitys oli suuri, jossa Prestashop päivitettiin 1.5 versiosta versioon 1.6. Päivityksen myötä teemoista tehtiin responsiivisia, eli sama teema soveltuu pc:lle, sekä mobiilaitteille. Aikaisemmin teemoista oli oma versio molemmille. Tämän vuoksi vanhat teemat eivät toimineet uudessa päivityksessä. Asia korjattiin rakentamalla vanha ulkoasu uuden teeman pohjalle.

5.4 Kauppaan muokatut toiminnot

Mikäli Prestashoppiin tarvitsee tehdä muutoksia, ihanteellisin tilanne olisi, jos muutokset tehtäisiin pelkästään teemoja ja moduuleita hyödyntäen. Aina tämä ei kuitenkaan ole mahdollista, vaan muutoksia täytyy tehdä Prestashopin luokka- ja käsittelijä tiedostoihin. Onneksi Prestashopissa on valmiina override-toiminnot tätä varten. Muokatut tiedostot laitetaan override-kansioon alkuperäisiä tiedostoja vastaaviin paikkoihin. Tällöin Prestahop tunnistaa, mitkä tiedostot tulee ohittaa ja käyttää override-hakemiston sisällä olevia tiedostoja. Sivuja kehittäessä suurin osa muutoksista pystyttiin toteutttamaan kyseisillä menetelmillä.

Koska Prestashop on laajin avoimen lähdekoodin verkkokauppa, monelle muullekin on tullut vastaan samanlaisia ongelmia ja haluttuja toimintoja, joten ratkaisuja ja valmiita moduuleita on paljon saatavilla.

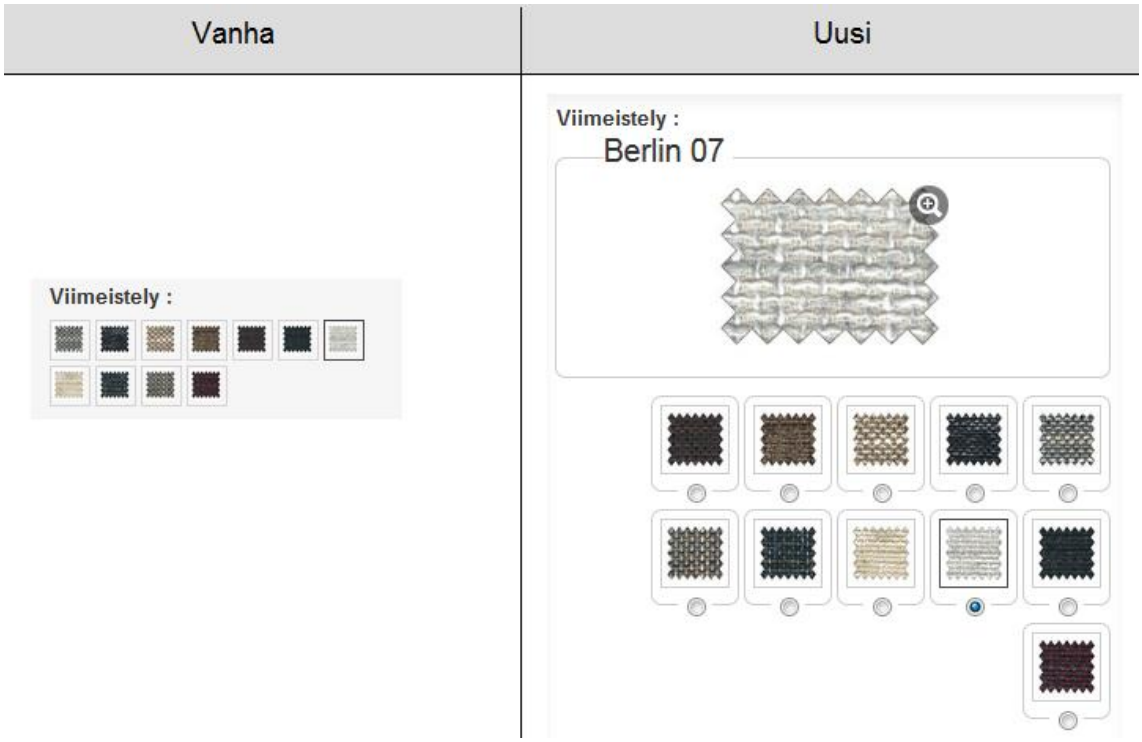
Sivustoille on tehty paljon pieniä muutoksia, joista suurin osa on tehty ulkoasuun. Seuraavissa kohdissa on muutamia esimerkkejä erilaisista muutoksista ja miten ne on toteutettu.

5.4.1 viimeistelyn valinta

Prestashopissa tuoteattribuuteilla on kolme esitystapaa: radiobuttonit, alasvetovalikot tai värit. Radiobuttonit ja alasvetolaatikot ovat hyviä sellaisenaan, mutta väriattribuuteissa oli parannettavan varaa. Tuotteen väriattribuuteiksi pystyy valitsemaan joko värin tai kuvan, jotka näytetään

tuotesivulla. Huonekalut.infon ja sohvat.infon tapauksessa sohvien kangas- ja nahkavaihtoehdot näytetään väriattribuuteilla.

Oletuksena väriattribuutit näkyvät hyvin pienellä, joten kuvista ei saa kunnolla selvää. Näin ollen attribuutit siirrettiin paremmin esille ja niiden visualisointia muokattiin käyttäjä-ystävällisemmäksi.

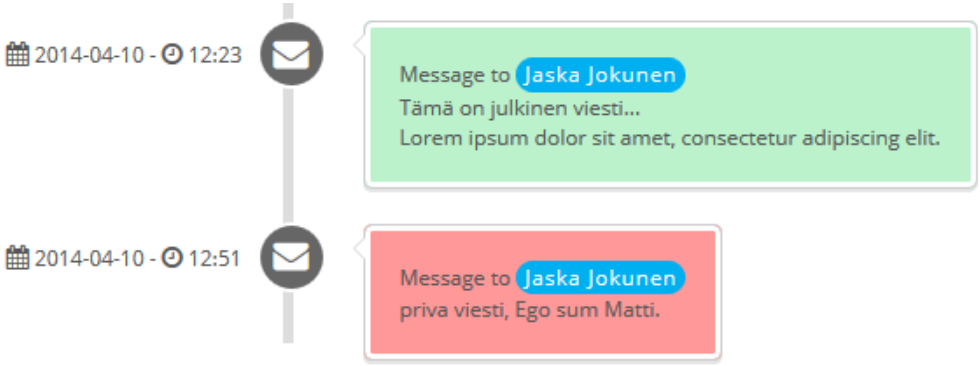


Kuva 21. Väri attribuutin paranneltu ulkoasu.

Muutokset toteutettiin muokkaamalla pelkästään teemaa eli tarkennettuna Smarty-, tyyli-, sekä javascript tiedostoja. kuvista tetiin isompia ja niihin lisättiin radiobuttonit jotta valittu attribuutti erottuu selvemmin. Lisäksi kuvien yläpuolelle lisättiin suurennettu kuva valitusta attribuutista, jonka voi klikkaamalla suurentaa popup-ikkunaan.

5.4.2 Keskustelun viestit

Hallintopaneelissa tilausta tarkastellessa on mahdollista laittaa tilausta koskeva viesti ja määritellä näkyykö viesti asiakkaalle vai pelkästään työntekijöille. Viestiketjua selatessa ei kuitenkaan käy selville, mitkä viestit näkyvät asiakkaille ja mitkä eivät. Asia korjattiin asettamalla viesteille taustavärit, jotka osoittavat ovatko viestit yksityisiä vai julkisia.



Kuva 22. Taustavärin avulla viestien julkisuus käy helposti selville.

Message-luokalla on jo olemassa private-muuttuja, joka määrittää onko viesti julkinen vai yksityinen. Muuttujaa ei voida sellaisenaan käyttää smarty-tiedostoissa, vaan se täytyy ensin asettaa smarty-muuttujaan luokkaa hoitavassa käsittelijässä, ennen kuin sitä voidaan käyttää smarty -tiedostoissa.

```
663     public function getTimeline($messages, $id_order)
664     {
665         $timeline = array();
666         foreach ($messages as $message)
667         {
668             $content = $this->l('Message to: ').' <span c
669
670             $timeline[$message['date_add']][] = array(
671                 'arrow' => 'left',
672                 'background_color' => '',
673                 'icon' => 'icon-envelope',
674                 'content' => $content,
675                 'date' => $message['date_add'],
676                 'private' => $message['private'],
677             );
678         }
```

Kuva 23. Message luokan private muuttuja asetetaan $timeline-ryhmään, jotta sitä voidaan käyttää smarty-tiedostoissa.

```
<div class="{if ($timeline_item.private==1)}private-message{else}public-message{/if}
```

Kuva 24. Div -elementin luokka määräytyy sen mukaan onko viesti julkinen vai ei.

5.4.3 Sähköpostit

Prestashopilla on sähköposti viestit jaettu kahteen eri paikkaan. Suurin osa sähköposteista on toteutettu kauppaan ja osa kuuluu Mailalerts-moduuliin. Yleisesti ottaen sähköpostien rakenteet ovat hyvät, mutta asiakasviesteissä sekä uuden tilauksen ilmoitusviestissä oli parannettavan varaa.

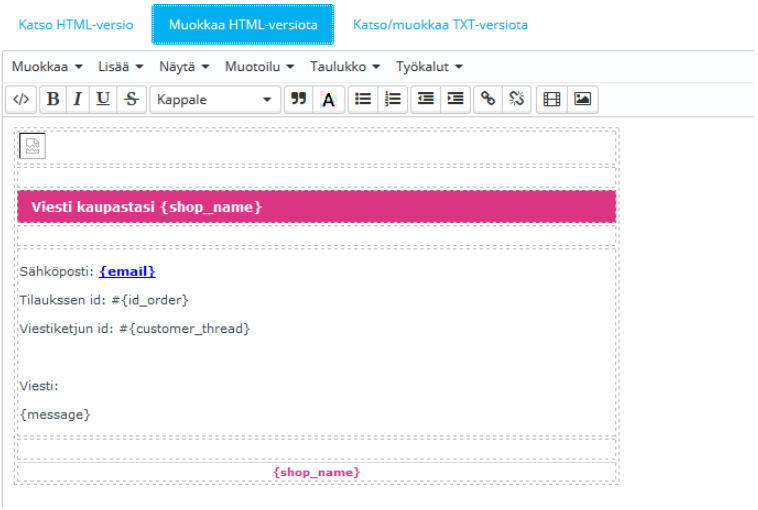
Asiakasviestit

Asiakasviesteissä oli lähettäjä ja itse viesti, mutta tilausta ja viestiketjua, johon viesti kuului ei oltu mainittu ollenkaan. Näin ollen viestin löytäminen oikeasta viestiketjusta oli hankalaa.
Jotta viesteihin pystytään lisäämään tilauksen ja viestiketjun id, täytyy kyseiset arvot asettaa käsittelijöissä muuttujiin, jotta ne voidaan sisällyttää itse sähköpostiviestiin. Asiakasviestit hoitaa kaupan oma Mail.php-luokka, sekä joukko käsittelijöitä.

```
161                 $var_list = array(
162                     '{order_name}' => '-',
163                     '{attached_file}' => '-',
164                     '{message}' => Tools::nl2br(stripslashes($message)),
165                     '{order_id}' => Tools::getValue('id_order'),
166                     '{customer_thread}' => $id_customer_thread,
167                     '{email}' =>  $from,
168                     '{product_name}' => '',
169                 );
```

Kuva 25. Käsittelijöissä esitellään muuttujat, joita halutaan käyttää sähköposteissa.

Sähköpostin ulkoasun ja tekstit voi määrittää hallintopaneelin sähköpostipohjien käännökset-osiossa. Sähköposteista on olemassa html-muodot, sekä tekstitiedosto-muodot vanhempia sähköposteja varten.



Kuva 26. Sähköpostiviestin pohja, johon on lisätty tilauksen- ja viestiketjun id:t.

Tilauksen ilmoitusviesti

Uuden tilauksen ilmoitusviesti tapahtuu Mailalerts-moduulin kautta, joka myös noudattaa MVC -mallia. Moduulin PHP-päätiedostossa esitellään muuttujat, joita halutaan käyttää sähköposti-viesteissä.

```
340             // custom items
341             $items_table_custom .=
342                 '<tr style="background-color:'.($key % 2 ? '#DDE2E6' : '#EBECEE').';">
343                     <td style="padding:0.6em 0.4em;">
344                         <strong>'
345                             .$product['product_name'].(isset($product['attributes_small']) ?
346                         '</strong>
347                     </td>
348                     <td style="padding:0.6em 0.4em; text-align:center;">'.(int)$product['pro
349                 </tr>';
```

Kuva 27. Uusi $items_table_custom muuttuja.

Kun uusi muuttuja on tehty ja siitä on muodostettu smarty muuttuja, voidaan sitä käyttää sähköpostiviestissä. Mailalerts -moduulin sähköpostiviestit voi tehdä samassa paikassa kuin muutkin sähköpostiviestit, eli hallintopaneelin sähköpostipohjien käännökset -osiossa.

```
406                 '{items_custom}' => $items_table_custom,
```

Kuva 28. Luodusta muuttujasta tehdään smarty versio.

Salasanan varmistus

Prestashopin rekisteröinti lomakkeessa ei ole ollenkaan salasanan varmistus -kenttää. Kenttä saatiin helposti luotua, lisäämällä puuttuva kenttä smarty-tiedostoihin, sekä tarkistus AuthController.php-käsittelijään.

Asiakastilin pystyy luomaan kahdessa eri paikassa; kirjautumissivulla tai ostoskorin sivuilla. Ostoskorin smarty-tiedostossa order-opc-new-account.tpl täytyy muistaa kenttä lisätä kahteen eri paikkaan, sillä Pretashopilla on ostoskorista kahta eri tyyppiä, perinteisempi tyyli, jossa jokainen ostoskorin kohta on omalla sivullaan ja käytännöllisempi yhden sivun ostoskori -tyyli.

Authcontroller.php-käsittelijässä tarkastetaan, että salasanakentät ovat samoja. Presashopissa on Tools hakemisto, joka pitää sisällään mm. Smarty-funktiot, Json-funktiot ja liuta erillaisia apufunktioita. Toolsin avulla voidaan esimerkiksi ottaa lomakkeiden kenttien tiedot ja näyttää virheviestit kätevästi.

```
374         // Checked the user address in case he changed his email address
375         if (Validate::isEmail($email = Tools::getValue('email')) && !empty($email))
376             if (Customer::customerExists($email))
377                 $this->errors[] = Tools::displayError('An account using this email address has already been registered.', false);
378         // Check if password matches confirmation
379         if (Tools::getValue('passwd') != Tools::getValue('confirm_passwd'))
380             $this->errors[] = Tools::displayError('Password confirmation does not match password !', false);
381         // Preparing customer
382         $customer = new Customer();
```

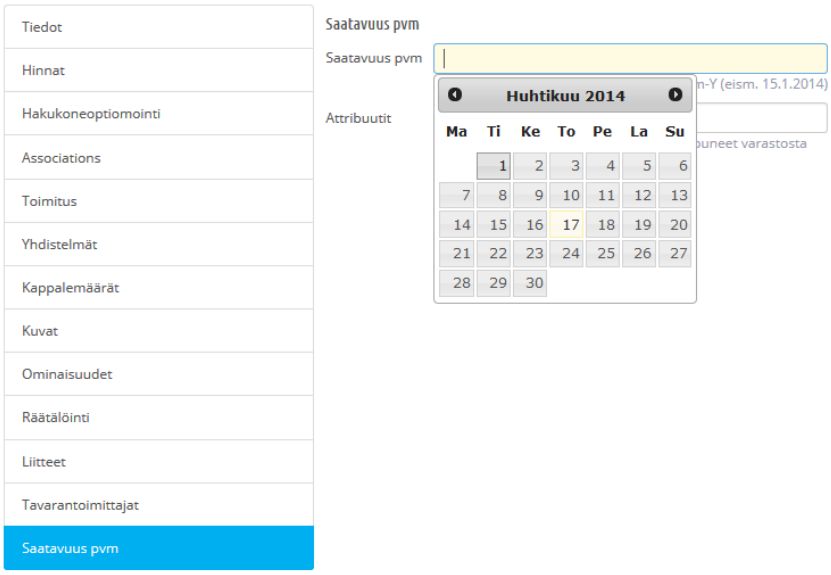
Kuva 29. Authcontroller.php-käsittelijä.

5.4.4 Saatavuus päivämäärä

Vanhassa OsCommercessa on saatavuuspäivämäärä-toiminto, jolla voidaan tuotesivulla näyttää päivämäärä, milloin tuotetta on saatavila. Kyseinen toiminto puuttuu kokonaan Prestashopissa. Jotta tämä toiminto voidaan toteuttaa Prestahopissa, tehtiin sitä varten oma moduuli.

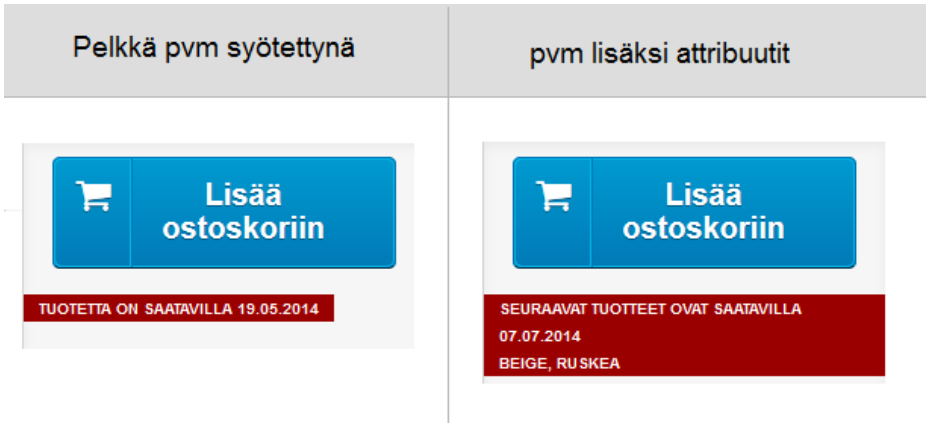
Prestashopin sivuilla on hyvät tutoriaalit moduulin tekoon. Moduuli rakennettiin valmiille moduuli pohjalle, johon muokattiin haluttuja vaatimuksia vastaavat toiminnot. Moduulilla on tietokannassa oma taulu, johon tallennetaan puuttuvien tuotteiden päivämäärät ja attribuutit.

Moduuli on kiinnitetty kolmeen eri koukkuun: actionProductUpdate, displayAdminProductsExtra, sekä displayProductButtons. ActionProductUpdate kutsutaan silloin, kun tuote lisätään tai päivitetään. Mikäli tuotteelle on asetettu saatavuuspäivämäärä, syötetään annetut tiedot tietokantaan kyseisessä action-koukussa. DisplayAdminProductsExtra on koukku, joka määrittää missä moduuli näytetään hallinnnon puolella. DisplayProductButtons koukku määrittää, missä kohti moduuli näytetään kaupan puolella. Kyseinen koukku kutsutaan tuotesivulla lisää ostoskoriin -napin jälkeen.



Kuva 30. Hallintopaneelin tuotesivu.

Kun tuotesivu avataan kaupan puolella, tarkastetaan moduulissa onko tuotteelle asetettu päivämäärää. Näytettävä teksti määräytyy sen mukaan, koskeeko päivämäärä kaikkia tuotteen attribuutteja vai vain osaa.



Kuva 31. Kaupan puolella näkyvät tekstit.

6 YHTEENVETO

Projektin tavoitteena oli luoda kaksi julkaisuvalmista verkkokauppaa. Projekti oli tekijälle ensimmäinen, jota ei tehty alusta loppuun, vaan se rakennettiin valmiin ohjelmiston ympärille. Suurin osa käytetyistä tekniikoista oli ennestään tuttuja Smartyä lukuun ottamatta.

Projektissa tuli vastaan muutamia ongelmia. Isoin ongelma oli projektin viivästyminen. Lisäksi aikataulut tuli arvioitua huonosti. Tuotteiden korjaaminen Huonekalut.infoon vei paljon arvioitua kauemmin, sillä tuotteita oli paljon. Vaikka tuotteiden siirtoa varten tehtiin parseri, joka siirti perustiedot, oli verkkokauppojen erot niin suuret, että kaikkia tietoja ei saatu vietyä. Tämän takia osa tiedoista piti korjata manuaalisesti. Käännösten tekeminen viivästytti myös projektia. Prestashopilla oli olemassa suomenkieliset käännökset, mutta ne eivät olleet kokonaan valmiit ja iso osa käännöksistä oli väärin, joten kaikki suomennokset piti käydä läpi.

Prestashopin 1.6 päivityksen kanssa ilmeni myös ongelmia. 1.6 päivityksen myötä teemojen oli tarkoitus olla responsiivisia, minkä vuoksi vanhat teemat eivät toimineet. Asia korjattiin rakentamalla vanha ulkoasu uuden teeman pohjalle. Ajallisesti päivitys ei aikatauluja juurikaan pidentänyt, vaikka ylimääräistä työtä se lisäsikin.

Viivästyksestä huolimatta Huonekalut.info saatiin valmiiksi ja julkaisukuntoon. Sohvat.info ei vielä ole julkaisukunnossa. Koska siinä käytetään paljon samoja toiminnallisuuksia kuin Huonekalut.infossa, lähinnä vain ulkoasuun tehdään muutoksia.

Parserin tekemiseen olisi pitänyt käyttää paljon enemmän aikaa, jotta tuotetietoja olisi voitu viedä lisää automaattisesti, Tämä olisi vähentänyt ylimääräistä työtä ja nopeuttanut projektia.

LÄHTEET

1. PHP. [Verkkoaineistot]. [20.11.2013]. Saatavissa:
   http://php.net/
   http://en.wikipedia.org/wiki/PHP

2. Smarty template engine. [Verkkoaineisto]. [20.11.2013]. Saatavissa:
   http://www.smarty.net/about_smarty

3. HTML [Verkkoaineisto]. [30.12.2013]. Saatavissa:
   http://www.w3schools.com/html/html_intro.asp

4. CSS [Verkkoaineisto]. [20.12.2013]. Saatavissa:
   http://www.w3.org/Style/CSS/

5. JavaScript [Verkkoaineisto]. [20.12.2013]. Saatavissa:
   http://javascript.about.com/od/reference/p/javascript.htm

6. jQuery [Verkkoaineisto]. [14.1.2014]. Saatavissa:
   http://jquery.com/

7. Programmer's Notepad [Verkkoaineisto]. [23.12.2013]. Saatavissa:
   http://en.wikipedia.org/wiki/Programmer's_Notepad

8. MySQL. The world's most popular open source database [Verkkoaineisto]. [23.12.2013]. Saatavissa:
   http://www.mysql.com/products/workbench/

9. Visual Studio [Verkkoaineisto]. [23.12.2013]. Saatavissa:
   http://www.visualstudio.com/

10. WAMP [Verkkoaineistot]. [23.12.2013]. Saatavissa:
    http://www.wampserver.com/en/
    http://fi.wikipedia.org/wiki/WAMP

11. Apache [Verkkoaineisto]. [23.12.2013]. Saatavissa:
    http://searchsoa.techtarget.com/definition/Apache

12. MySQL [Verkkoaineisto]. [23.12.2013]. Saatavissa:
    http://dev.mysql.com/doc/refman/4.1/en/what-is-mysql.html

13. Meta Tags [Verkkoaineisto]. [20.1.2014]. Saatavissa: http://searchenginewatch.com/article/2067564/How-To-Use-HTML-Meta-Tags

14. Prestashop, Prestashop 1.5 documentation [Verkkoaineisto]. [20.1.2014]. Saatavissa: http://doc.prestashop.com/dashboard.action