JULKAISUJÄRJESTELMÄ MARKKINOINTISIVUN TOTEUTUKSESSA

Pekka Suopellonmäki

Opinnäytetyö
Toukokuu 2011

Mediatekniikka
Tekniikan ja liikenteen ala

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖN
KUVAILULEHTI

Tekijä(t) SUOPELLONMÄKI, Pekka	Julkaisun laji Opinnäytetyö	Päivämäärä 28.4.2011
	Sivumäärä 59	Julkaisun kieli Suomi
	Luottamuksellisuus ( ) saakka	Verkkojulkaisulupa myönnetty (X)

Työn nimi
JULKAISUJÄRJESTELMÄ MARKKINOINTISIVUN TOTEUTUKSESSA

Koulutusohjelma
Mediatekniikan koulutusohjelma

Työn ohjaaja(t)
MANNINEN, Pasi

Toimeksiantaja(t)
HAUTANEN, Arto. Into-Digital Oy

Tiivistelmä
Opinnäytetyössä tarkasteltiin markkinointisivuston toteutusta ilmaisella avoimen lähdekoodin julkaisujärjestelmällä. Vaihtoehtoina olivat Wordpress, Drupal ja Joomla. Opinnäytteen toimeksiantajana toimi Into-Digital Oy, ja työ pohjautui pääasiassa yhteen Into-Digital Oy:llä tehtyyn Internet-sivustoon. Sivuston nimeä ei haluttu julki, joten opinnäytteessä pyrittiin yleisempään kuvaukseen julkaisujärjestelmän käytöstä tuotetietoja sisältäville sivustoille. Sivusto ei sisältänyt verkkokauppaa.

Sivusto rakennettiin Wordpress-julkaisujärjestelmän päälle, sillä se oli ollut paljon käytössä myös muissa Into-Digital Oy:n projekteissa ja osoittautunut nopeasti kehitettäväksi alustaksi. Julkaisujärjestelmä räätälöitiin loppuasiakkaan ylläpidettäväksi käyttäen hyväksi valmiita komponentteja ja rakentamalla hallintatyökalut tuotteiden lisäämiseen ja muokkaamiseen. Hallintatyökalujen muihin ominaisuuksiin kuului mm. tuotteiden eräajo Excel-yhteensopivasta CSV-tiedostosta, kuvien tuonti ZIP-pakettina sekä tuotetietojen lataaminen Excel-tiedostona.

Sivustolle toteutettiin AJAX-tekniikkaa hyödyntävä tuotehaku, mahdollisuus tulostaa tai lähettää tuotetiedot PDF-tiedostona sekä Google Analytics -kävijäseuranta. Google Analytics -seuranta lisättiin sivulatausten lisäksi myös AJAX-tuotehakuun, jolloin suosituimmista hakukriteereistä saatiin statistiikkaa.

Sivuston rakentaminen aloitettiin marraskuun lopulla 2010 ja julkaistiin helmi-maaliskuun vaihteessa 2011. Sivustolle toteutettuja ominaisuuksia ja niiden rakentamisesta saatuja oppeja voidaan jatkossa käyttää hyödyksi myös muilla vastaavilla sivustoilla.

Avainsanat (asiasanat)

Julkaisujärjestelmät, PHP, Javascript

Muut tiedot

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

DESCRIPTION

Author(s)	Type of publication	Date
SUOPELLONMÄKI, Pekka	Bachelor´s Thesis	28.4.2011
	Pages 59	Language Finnish
	Confidential ( ) Until	Permission for web publication (X)

Title
USING CONTENT MANAGEMENT SYSTEM FOR IMPLEMENTATION OF MARKETING WEBSITE

Degree Programme
Media Engineering

Tutor(s)
MANNINEN, Pasi

Assigned by
HAUTANEN, Arto. Into-Digital Oy

Abstract
The thesis describes the use of open source content management system (CMS) for a marketing website. Alternatives for the CMS were Wordpress, Drupal and Joomla. The project work was assigned by Into-Digital Oy and it was mainly based on one particular website project. Due to the wish of the customer, the name of the website is not revealed. For that reason, the description of CMS usage was designed to be more general and applicable to a common marketing website with product information.

The website was built on Wordpress CMS. It was tailored in order to be easy to maintain by the customer. This was achieved by using free components and by providing tools for product management. The management included ability to add and edit products with an Internet browser based interface. The features also consisted of an import/export tool which allowed the customer to download and upload new products and images in a batch. AJAX-technique was used in product search and a visitor could download selected product information in PDF-format which were dynamically created on a webserver. Google Analytics was also implemented for tracking visitors and most used search criteria.

The creation of the website was started in the end of November 2010 and it was published in late February 2011. The features implemented for the website can be used in forthcoming website projects with similar aspects.

Keywords
Content management system, PHP, Javascript

Miscellaneous

SISÄLTÖ

KUVIOT

1. TYÖN LÄHTÖKOHDAT

1.1 Tavoite

Opinnäytetyön tavoitteena oli tarkastella avoimen lähdekoodin julkaisujärjestelmien käyttöä ja valitun alustan soveltuvuutta markkinointisivuston toteutuksessa. Opinnäytetyössä tarkasteltiin sivuston sekä tuotetietokannan teknistä toteutusta eikä niinkään sen käytettävyyden tai graafisen ulkoasun toteutusta. Sivustoon kuului mm. laaja tuotetietokanta sekä hallintatyökalut tuotteiden lisäämiseen ja muokkaamiseen. Työn kuvaus perustuu pääasiassa yhteen Into-Digital Oy:llä tehtyyn markkinointisivustoon, mutta tietoperustana on käytetty myös muita sivustoja, jotka on tehty Into-Digital Oy:llä tai kouluprojekteina.

Tässä opinnäytetyössä markkinointisivulla tarkoitetaan sivustoa, jossa voi selata tuotteita, ajankohtaisia ilmoituksia ja joka sisältää tietoa yrityksestä. Lisäksi sisältö jaetaan erilaiseksi eri käyttäjille sen mukaan, onko hän yritystunnuksella kirjautunut vai ei. Sivustosta on myös suppeampi englanninkielinen versio, joka sisältää lähinnä tietoa yrityksestä ja yhteystiedot. Sivuston tarkoitus on olla enemmänkin informatiivinen eikä se sisällä verkkokauppaa, mutta tuotetiedoista on kuitenkin linkki muualla sijaitsevaan kauppaan, josta tuotteita voi tilata.

Sivusto toteutettiin kokonaan uuteen Nebulan webhotelliin, joten aiemman sivuston sisältöä ei tarvinnut teknisessä toteutuksessa ottaa huomioon.

1.2 Toimeksiantaja

Toimeksiantaja Into-Digital Oy on mainostoimistojen palvelemiseen keskittynyt digitoimisto, jonka tehtävänä on auttaa asiakkaitaan onnistumaan digitaalisissa medioissa. Sen toimenkuvaan kuuluu mm. kampanjasivujen luominen ja Internet-mainonta. Into-Digital Oy on osa viestintäkonserni Into ja Ida Oy:tä, ja toimii pääasiassa Helsingissä sekä Jyväskylässä.

2. TERMISTÖ LYHYESTI

2.1 PHP

PHP eli hypertext preprocessor on palvelinohjelmointikieli, jota käytetään dynaamisten verkkosivujen toteutukseen. PHP:n lähdekoodi on avoin ja sen jakelu perustuu omaan PHP lisenssiin. (PHP Licensing 2011.)

PHP:n nimi tulee alun perin sanoista Personal Home Pages, sillä niitä varten se oli luotu. Alkujaan se oli vain joukko makroja omien Internet-sivujen ylläpitoon. PHP:n kehitti Rasmus Lerdorf vuonna 1995. (Zandstra 2005, 20.)

PHP on tulkattava skriptikieli eli sen palvelimelle asennettu tulkki kääntää koodin ajon aikana. Se kirjoitetaan tavallisesti HTML:n yhteyteen, mutta sitä ei lähetetä suoraan asiakkaalle vaan se jäsennellään PHP-ohjelman tai -moduulin toimesta. PHP-skripti tulkataan ja suoritetaan palvelimella, ja yhdistetään HTML:n kanssa, joka lähetetään asiakkaalle. (Zandstra 2005, 20.)

PHP:llä voidaan muun muassa tehdä kyselyjä tietokantaan, luoda kuvia, lukea ja kirjoittaa tiedostoja sekä keskustella etäpalvelimien kanssa. Neljännessä versiossa PHP:stä tuli täysiverinen olio-ohjelmointikieli ja sen Zend-moottorin ansiosta PHP pärjää suorituskyvyssään esimerkiksi ASP:ille. (Zandstra 2005, 20-22.)

PHP:n nykyinen versio on 5.3. W3Techs analytiikan mukaan yli 75 % maailman Internet-palvelimista käyttää PHP:tä palvelinohjelmointikielenään. (Usage of server-side programming languages for websites 2011.)

Vaikka PHP tukee muitakin tietokantoja kuin MySQL:ää ja Windows-palvelimia, oli toteutetulla sivustolla käytetty Nebulan webhotelli Linux- ja MySQL-pohjainen. Koska yleisimmät julkaisujärjestelmät pohjautuvat PHP-kieleen, se oli luonnollinen valinta markkinointisivun toteutukseen.

2.2 Javascript

Javascript on lisenssivapaa skriptikieli, jolla luodaan interaktiivisuutta verkkosivuille. Se on ECMAScript-standardin mukainen kieli, ja se hyväksyttiin vuonna 1998. Javascript-koodia ei tarvitse kääntää erikseen, vaan tulkki suorittaa sen ajon aikana. Nimien yhtäläisyydestä huolimatta Javascriptiä ei tule sekoittaa Java-ohjelmointikieleen, sillä ne ovat periaatteellisesti erilaisia. (JavaScript Introduction 2011.) Javalla voidaan tehdä verkkosivuille itsestään toimivia appletteja, mutta Javascript upotetaan HTML:n sekaan. Javascriptin avulla Internet-sivuista saadaan toiminnaltaan sulavampia ja paljon työpöytäsovelluksia muistuttava käyttökokemus, kun näkymästä toiseen siirtymisessä ei tarvitse aina ladata sivua uudestaan.

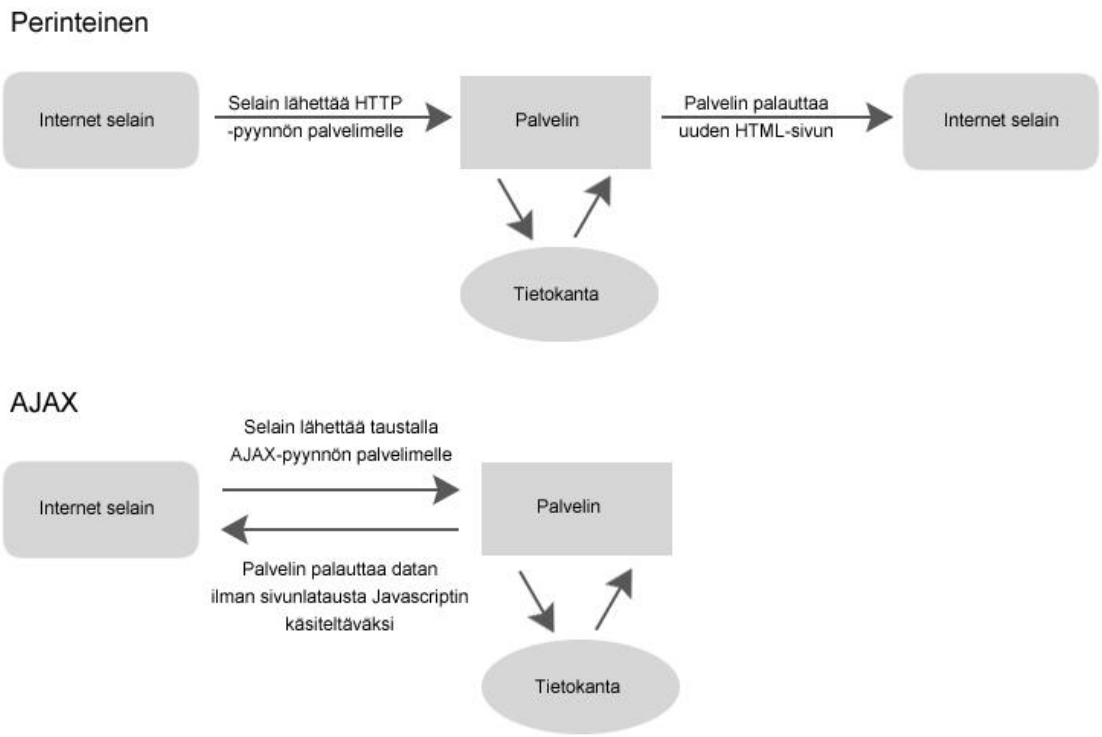
Eri Internet-selaimilla on omat Javascript-moottorinsa. Näistä tunnettuja ovat esimerkiksi Google Chrome-selaimen V8 ja Mozilla Firefoxin JägerMonkey. Javascript ajetaan käyttäjän omalla koneella, jolloin sen nopeus on hyvin paljon riippuvainen selaimen Javascript-tulkista. Eri selainten erot nopeudessa voivat olla todella suuret. Tämä korostuu erityisesti niiden vanhemmilla versioilla.

2.3 AJAX

AJAX on lyhenne sanoista Asynchronous Javascript and XML. Se on tekniikka, jota käytetään nopeiden ja dynaamisten verkkosivustojen luontiin. AJAX-tekniikan avulla käyttäjän syötteisiin voidaan reagoida asynkronisesti, jolloin palvelimelle voidaan siirtää taustalla dataa ja tuoda sitä käyttäjän selaimeen ilman, että koko sivua täytyy ladata uudestaan. (AJAX Introduction 2011.)

Selaimelta voidaan lähettää taustalla tietoa palvelimelle XMLHttp-pyyntönä, jonka jälkeen palvelin voi palauttaa vastauksen esimerkiksi XML- tai JSON-muodossa. AJAX ei ole uusi ohjelmointikieli, vaan se on uusi tapa käyttää aiempia tekniikoita. Normaalisti Internet-selain lähettää palvelimelle HTTP-pyynnön, jonka perusteella palvelin muodostaa HTML-dokumentin esimerkiksi hakemalla tietoa tietokannasta. Tämän jälkeen palvelin palauttaa selaimelle uuden HTML-sivun ladattavaksi. AJAX-menetelmän etuna on, että Javascriptin avulla tieto palautetaan olemassa olevalle

HTML-sivulle ilman, että käyttäjä joutuu lataamaan uutta sivua. Palvelimelta tuleva tieto voidaan käsitellä ja esittää Javascriptillä. Periaate on kuvattu kuviossa 1.



KUVIO 1. Asynkroninen tiedon haku AJAX-tekniikalla

2.4 JSON

JSON eli Javascript object notation on dataobjektin selkokielinen muoto. Se on tapa esittää objektin data, kuten esimerkiksi muuttujat tai taulukot ihmisen ymmärtämässä muodossa. Nimestään huolimatta JSON on todellisuudessa ohjelmointikieliriippumaton ja sitä käytetään myös muissa kielissä. (Introducing JSON 2011.)

Esimerkissä muutetaan ja tulostetaan JSON-merkkijonoksi yksinkertainen PHP:n assosiatiivinen taulukko, jossa on kaksi tietuetta käyttäen PHP:n json_encode-funktiota.

```
$taulukko = array('tuote1'=>'nimi1','tuote2'=>'nimi2');
echo json_encode($taulukko);
// tulostaa {"tuote1":"nimi1","tuote2":"nimi2"}
```

3 JULKAISUJÄRJESTELMÄN VALINTA

3.1 Mitä julkaisujärjestelmät ovat?

Vuosien ajan perinteinen tapa tehdä verkkosivuja oli rakentaa ne kokonaan pelkällä HTML-kielellä. Jos sivuille tarvitsi tehdä muutoksia, ylläpitäjät käyttivät Adobe Dreamweaverin kaltaisia ohjelmia sivuston koodin muokkaukseen jokaiselle alasivulle erikseen. Pienillä verkkosivuilla tämä ei ollut iso työ, mutta nykyään pienemmätkin verkkosivut voivat sisältää satoja ja suuremmat satoja tuhansia alasivuja. Julkaisujärjestelmien tarkoitus on auttaa hallitsemaan sisältöä jakamalla sivusto dynaamisiin osiin, jolloin esimerkiksi navigaation muutos näkyy heti kaikkialla eikä jokaista sivua tarvitse enää erikseen korjata. Sivuston sisältö, kuten tekstit ja kuvat, ovat tallennettuina tietokantaan, joten ne voidaan helpommin lisätä tai siirtää halutuille sivuille. (Severdia & Crowder. 2010)

Yksinkertaisesti määriteltynä julkaisujärjestelmä on verkkosivun sisällönhallinnan työkalu, jonka avulla eri henkilöt voivat luoda, muokata ja julkaista sisältöä verkkosivulla siten, että sisältö pysyy yhtenäisenä. Se mahdollistaa sivuston ylläpidon ja päivitykset ilman, että sisällöntuottajan on välttämättä osattava verkkotekniikoita.

Julkaisujärjestelmää voisi verrata kirjastoon. Siinä missä kirjasto säilyttää kirjoja ja pitää ne järjestyksessä, julkaisujärjestelmä säilyttää verkkosivuston tekstit, kuvat ja muun sisällön järjestyksessä ja helposti saatavilla. (What is Joomla! 2011.)

Erilaisia julkaisujärjestelmiä verkkosivujen hallintaan on todella monia. Avoimen lähdekoodin alustat, kuten esimerkiksi Wordpress, Drupal ja Joomla, perustuvat GPL eli GNU General Public -lisenssiin ja ovat siten ilmaisia käyttää. Ilmaisuutensa, laajennettavuuden ja niiden ympärille muodostuneiden yhteisöjen tuen takia niiden suosio on kasvanut suureksi. Ne ovat hyvin suositeltavia pientenkin sivustojen rakentamiseen, sillä niiden asennus ei vie kauan, mutta ylläpidon kannalta edut ovat merkittävät: päivitykset voidaan tehdä selaimella hallintapaneelista eikä ftp-yhteyttä tarvitse avata.

Erityisesti mainosalalla erilaiset kampanjasivut saatetaan rakentaa hyvin lyhyellä varoitusajalla ja vaikka niiden elinikä saattaa olla lyhyt, sisällön päivitystahti on nopea. Tämän vuoksi pelkät staattiset sivut eivät riitä, koska silloin työmäärä kasvaa suhteettoman suureksi. Pienikin tekstimuutos vaatii korjauksen tekemisen suoraan sivuston tiedostoihin, ftp-yhteyden avaamisen ja edellisen tiedoston korvaamisen. Lisäksi jos sisältö on sekaisin toimintalogiikan kanssa, on olemassa riski, että pelkkä sisältötekstin huolimaton muokkaus saattaa rikkoa sivuston toimivuuden. Muokkaaja saattaa esimerkiksi vahingossa ja huomaamattaan muokata ohjelmakoodista jonkin oleellisen asian; pelkkä puolipisteen poistaminen PHP-koodirivin lopusta aiheuttaa virheen.

Avoimen lähdekoodin Joomla, Drupal ja erityisesti Wordpress ovat nopeita asentaa ja antavat heti perustason sivurakenteen ja työkaluja sisällön hallintaan. Näin aikaa jää enemmän itse sivuston ulkoasun ja toimintojen räätälöintiin. Lisäksi ohjelmalogiikka on eriytetty sisällöstä, jolloin sisällönmuokkaus ei riko palvelinohjelmointia.

Markkinointisivu voi olla osa suurempaa kokonaisuutta, joka on rakennettu laajemman kaupallisen sisällönhallintajärjestelmän, kuten Typo3:n päälle, mutta tässä tapauksessa sivusto rakennettiin tyhjästä Nebulan webhotelliin, joten sisällönhallintaratkaisu perustui nopeasti käyttöönotettaviin ja ilmaisiin alustoihin.

Luvuissa 3.2-3.4 esitellään kolme yleisintä julkaisujärjestelmävaihtoehtoa, kun markkinointisivusto rakennetaan alusta. Näistä kaikki vaativat palvelimelta PHP tuen sekä MySQL-tietokannan.

3.2 Wordpress

Wordpressin kehitys alkoi vuonna 2003. Se on alun perin luotu blogien ylläpitämiseen ja rakentui pitkään sen ympärille. Avoimen lähdekoodin, helpon laajennettavuuden ja yhteisössä saamansa suosion myötä se on kuitenkin kasvanut täysimittaiseksi julkaisujärjestelmäksi, joka soveltuu blogien lisäksi hyvin monimuotoisten sivustojen hallintaan. Nykyään sille on saatavilla tuhansia laajennuksia ja se on käytössä miljoonilla sivustoilla. (About Wordpress 2011.)

W3Techs-tilastojen mukaan Wordpress on suosituin julkaisujärjestelmä. Kaikista W3Techs-sivuston tarkkailemista julkaisujärjestelmistä Wordpressin osuus on peräti 55 %. Toiseksi yleisimmän Joomlan osuus on n. 11 %. (Usage of content management systems for websites 2011.) W3Techs analysoi noin miljoonaa suosituinta verkkosivua päivittäin (W3Techs Technologies Overview 2011).

Wordpressiä käyttävät mm. musiikkipalvelu Spotify, uutistoimistot Reuters ja CBS toimittajablogeissaan sekä teknologiasivusto Techcrunch (Wordpress showcase 2011). Suomalaisista sivustoista Wordpressiä käyttävät joko koko sivustona tai osana sitä mm. MTV3:n blogit, Tietokone-lehden sekä Taloussanomien blogit sekä Kamera- ja Stara-lehti (Wordpress näyteikkuna 2011).

Perusasennuksella Wordpress jakaa sisällön karkeasti kahteen eri osa-alueeseen: artikkelit sekä sivut. Artikkelit (eng. posts) ovat dynaamisia viestejä, joita yleensä käytetään sivuston uutisten, blogiviestien ym. usein päivittyvien sisältöjen esittämiseen. Ne voidaan jaotella kategorioin ja avainsanoin, jolloin ne taipuvat hyvin monipuolisesti jopa tuotteiden esittämiseen. Sivut (eng. pages) ovat staattisempia ja niillä esitetään harvemmin muuttuvaa sisältöä, kuten yhteystiedot. Sivuille voidaan luoda omia sivupohjia ja ne voidaan järjestellä hierarkkisesti ylä- ja alasivuiksi.

Wordpressin uusin 3.0 versio toi ns. mukautetut tietotyypit, joiden avulla sisältö voidaan jakaa paremmin esimerkiksi blogiviesteihin, uutisiin tai tuotteisiin ilman aiempaa säätämistä artikkeleiden kategorioilla. Tämä muutti Wordpressin mielikuvaa blogialustasta täysimittaiseksi sisällönhallintajärjestelmäksi.

3.3 Joomla!

”Joomla! on ilmainen, monipuolinen ja pelkällä Internet-selaimella käytettävä julkaisujärjestelmä, jonka avulla yksityishenkilöt, yritykset tai yhteisöt voivat lisätä ja päivittää Internet-sivujen sisältöä itsenäisesti ilman mitään ulkopuolista apua.” (Mikä on Joomla!? 2005.)

Joomlan ensimmäinen 1.0 -versio ilmestyi vuonna 2005, mutta se pohjautuu vanhempaan Mambo-julkaisujärjestelmään. Mambo oli tarkoitettu yhtiön Miro Corpora-

tion of Australia sisäiseksi sisällönhallintajärjestelmäksi, ja sen lähdekoodista tuli avointa vuonna 2001. Mambo sai maailmanlaajuista suosiota, ja sen ympärille kehittyi yhteisö, joka tuotti teemoja ja laajennuksia sekä kaupallisesti että ilmaiseksi. Kuitenkin vuonna 2005 Mambon kehittäjien ja kehityksen ohjaukseen luodun voittoa tavoittelemattoman järjestön erimielisyyksien vuoksi kehittäjät päättivät hylätä Mambo-projektin ja luoda sen pohjalta uuden julkaisujärjestelmän. Ensimmäisen vuoden aikana Joomla ladattiin yli 2.5 miljoonaa kertaa. (Rahmel 2009, 1-2)

Joomlan nykyinen versio on 1.6.

3.4 Drupal

Drupal on avoimen lähdekoodin sisällönhallintajärjestelmä, joka pohjautuu MySql-tietokantaan ja PHP-ohjelmointikieleen. Sitä käyttävät mm. Uusi Suomi, Nokia Siemens Networks ja Nelonen. (Mikä on Drupal? 2011.)

Drupal syntyi puoliksi vahingossa, kun Dries Buytaert opiskeli tietojenkäsittelyä Belgiassa Ghentin yliopistossa ennen vuosituhannen vaihdetta. Muutamien kollegoidensa kanssa hänellä oli langaton verkko, johon hän päätti rakentaa foorumin yhteydenpitoa varten. (Drupal history as seen by Dries) Driesin valmistumisen jälkeen ryhmä päätti siirtää foorumin Internetiin, jotta yhteydenpito jatkuisi opiskeluiden jälkeenkin. Sivuston nimeksi piti tulla dorp.org, joka tarkoittaa hollanniksi pientä kylää. Kirjoitusvirheen takia osoitteeksi tuli kuitenkin drop.org, jonka Dries päätti hyväksyä. Keskustelu sivustolla kääntyi verkkotekniikoihin ja ideoihin, joita myös kokeiltiin sivustolla. (About Drupal 2011.)

Vuosituhannen vaihteen jälkeen kiinnostus drop.org sivustolla käytettyä foorumia kohtaan kasvoi ja uusia ideoita tulvi. Vuonna 2001 Dries julkaisi drop.org:ssa käytetyn koodin nimellä Drupal, jotta useammat foorumin jäsenet voisivat kokeilla eri ideoita, ja laajentaa sitä. Näin hänelle itselleen jäisi enemmän aikaa muuhun kehitystyöhön kuin uusien ominaisuuksien lisäämiseen. Lähdekoodista tuli avointa. Jeremy Andrews kirjoitti artikkeleita ja ohjeita Drupalista, ja yhteisö jatkoi kasvuaan. (Drupal history as seen by Dries.)

Drupal on rakenteeltaan hyvin modulaarinen. Sillä tehdyt sivustot koostuvat eri rakennuspalikoista, jotka hoitavat tietyn toiminnon. Moduulit voivat olla suurempia tai pienempiä, kuten esimerkiksi kuvagalleria tai ostoskori. Kehittäjälle lähes kaikki toiminnot ovat muokattavissa ja moduuleita löytyy valmiina tuhansia. Useimmat saatavilla olevista moduuleista ovat Drupal-yhteisön tekemiä ja siten käytettävissä ilmaiseksi kehittäjille. (Byron, Berry, Haug, Eaton, Walker & Robbins. 2009, 1-2)

Drupalin nykyinen versio on 7.0.

3.5 Miksi Wordpress?

Jokainen näistä kolmesta suosituimmasta julkaisujärjestelmästä olisi ollut soveltuva toteutetun markkinointisivuston alustaksi. Ne ovat olleet olemassa jo vuosia, niiden kehitys on ollut tasaista ja jokaisella on oma kehittäjäyhteisönsä, joka paitsi luo uusia laajennuksia ja ehdottaa uusia ominaisuuksia, raportoi myös tietoturvaongelmista. Monilla kehittäjillä on oma suosikkinsa ja siitä toiseen vaihtaminen ei ole välttämättä helppoa. Jokaisella kun on oma tapansa toteuttaa laajennukset, tietokantarakenne ja esimerkiksi käyttäjien tai käyttäjäryhmien jako.

Wordpressin suurimpia etuja on sen koko ja suosio. Se on kooltaan pieni, kevyt ja todella helppo asentaa, mutta toisin kuin Drupal, se sisältää valmiiksi hyvät ja helppokäyttöiset työkalut sivujen, artikkeleiden ja käyttäjien hallintaan. Lisäksi erikoisemmat toiminnot voidaan toteuttaa sivumallien tai laajennusten avulla. Laajennuksia löytyy valmiina tuhansia. Wordpress ei kuitenkaan ole missään nimessä täydellinen julkaisujärjestelmä. Blogitausta näkyy sisällönhallinnan jaottelussa, ja vaikka artikkeleita voikin kategorioiden avulla käyttää hyväksi esimerkiksi nostojen hallintaan, se on aina hieman viritelmän oloinen ratkaisu. Wordpress 3.0 on parantanut tätä tilannetta tuomalla itse luodut sisältötyypit.

Joomla on kolmikon järein julkaisujärjestelmä, mutta suuren koon ja henkilökohtaisen kehittämiskokemuksen puutteen vuoksi se olisi hidastanut sivuston valmistumista. Joomlaa ei harkittu yhtä vakavasti kuin Wordpress- tai Drupal-julkaisujärjestelmää, sillä asiakas suosi Wordpressiä.

Drupal on perusasennukseltaan hyvin insinöörimäisen pelkistetty, ja sama koskee sen sisällönhallintaa; ylläpidon työkalut ovat hyvin karsittuja. Ilman laajennuksia se tarjoaa rungon sivustolle, mutta ei sovellu loppuasiakkaan ylläpidettäväksi. Tämä on samalla Drupalin vahvuus ja heikkous. Vahvuutena se on täysin muokattavissa mieleiseksi ja taipuu käytännössä mihin vain, mutta se vaatii aikaa. Drupal on myös hyvin nopea asentaa. Keveytensä ja laajennettavuutensa puolesta se olisi voinut olla soveltuva vaihtoehto sivuston toteutukselle, mutta aiemmat positiiviset kokemukset Wordpress-sisällönhallintajärjestelmästä sekä asiakkaan toive käänsivät valinnan siihen. Kun julkaisujärjestelmän käyttö oli ennestään tuttu, perustoteutus oli nopeammin valmis ja aikaa oli enemmän tuotehaun sekä -hallinnan tapaisiin räätälöinteihin.

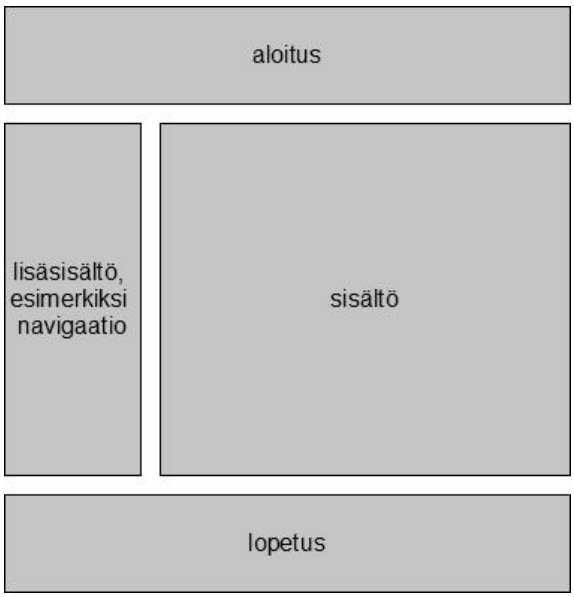
4. SIVUSTON RAKENNE

4.1 Teema

Wordpress-sivustojen näkyvä julkinen osuus tehdään teemoilla. Teema on ulkoasu ja toiminnallisuus, jonka käyttäjä näkee. Se on kokoelma tiedostoja, jotka yhdessä muodostavat toimivan sivun. (Using themes 2011.) Teeman ytimenä on ns. silmukka, joka hakee tarvittavat tiedot ja esittää jokaisen artikkelin halutulla tavalla. (The loop 2011.)

4.1.1 Miten teema toimii

Internet-sivut voidaan useimmiten jakaa kolmeen osaan, sillä yleensä eri sivujen välillä vaihtuu vain sisältöteksti sivun alku- ja loppuosioiden pysyessä samana. Näin ollen osat olisivat siis alku, sisältö ja loppu (ks. kuvio 2).



KUVIO 2. Sivuston jako osiin.

Wordpress-teemat noudattavat useimmiten tätä periaatetta, ja ne koostuvat yleensä vähintään seuraavista tiedostoista:

- header.php

- index.php
- footer.php
- functions.php
- style.css.

Wordpress ei kuitenkaan estä käyttämästä esimerkiksi vain yhtä index.php-tiedostoa, mutta se ei ole järkevää: sivun eri osiot ja toiminnot on parempi jakaa omiin osa-alueisiin, jolloin niiden hallinta on helpompaa. Functions.php sisältää teeman erikoisemmat ominaisuudet, kuten AJAX-funktiot.

Wordpress ei aseta mitään rajoituksia siihen, miten teema jaetaan. Edellä mainitut tiedostot ovat järkevä minimi, mutta näiden lisäksi oleellisia tiedostoja ovat mm. single.php, page.php, comments.php sekä tarvittaessa sidebar.php. Single.php on sivumalli yksittäiselle artikkelille ja page.php vastaavasti yksittäiselle sivulle. Koska Wordpress on aikoinaan rakennettu blogialustaksi, kommenteilla on erityinen asema Wordpress-julkaisujärjestelmässä. Comments.php on yleinen kommenttipohja niille artikkeleille ja sivuille, joissa kommentit ovat sallittuna. Joillekin sivuille halutaan usein erillinen sivupalkki, jossa on esimerkiksi sivuston navigaatio tai vimpaimia (Wordpressin käyttämä suomennos widget = vimpain), kuten kalenteri tai lista uusimmista kommenteista, ja sidebar.php on tarkoitettu tätä varten. Huomioitavaa on, että näiden tiedostojen nimet on määrätty ja Wordpress osaa automaattisesti etsiä niitä teeman sisältä. Esimerkiksi funktio get_sidebar() etsii nimenomaan sidebar.php-tiedostoa. Jos tätä tiedostoa ei löydy, funktio käyttää Wordpressin oletussivupohjaa sivupalkin esittämiseen.

Page.php-sivumalli toimii teeman oletusmallina sivuille. Tämän lisäksi kehittäjä voi tarvittaessa luoda oman sivupohjan vaikka jokaiselle eri sivulle. Sivupohja on yksittäinen PHP-tiedosto, jonka alussa sen nimi on määriteltynä kommenteissa.

```
<?php
/* Template Name: Tuote */
?>
```

Tämän jälkeen luotu sivupohja ilmestyy pudotusvalikkoon sivunmuokkausnäkymään.

4.1.2 Teema toteutetulla sivustolla

Rakennettu sivusto toteutettiin Wordpress-julkaisujärjestelmällä. Sivuston julkinen sisältö jaoteltiin seuraavasti:

- Kirjautumissivu yritysasiakkaille
- Erillinen etusivu kirjautuneille, jolla voidaan nostaa esille yksi ajankohtainen artikkeli sekä kaksi tuotetta
- Tuotehaku, joka toimii kirjautumattomilla myös etusivuna
- Ajankohtaista
- Yritysinfo
- Yhteystiedot
- Sisältö kirjautumattomille sekä englanninkielisille käyttäjille.

Perusasennuksen jälkeen luotiin sivupohjat eri osioille sekä kirjautumissivu. Kirjautuminen tallennetaan PHP:n avulla sessiomuuttujaan ja evästeeseen, jolloin käyttäjän ei tarvitse joka kerralla kirjautua uudestaan. Sessiomuuttujan avulla sisällöstä näytetään eri osuudet eri käyttäjäryhmille yksinkertaisella if-else-komentorakenteella.

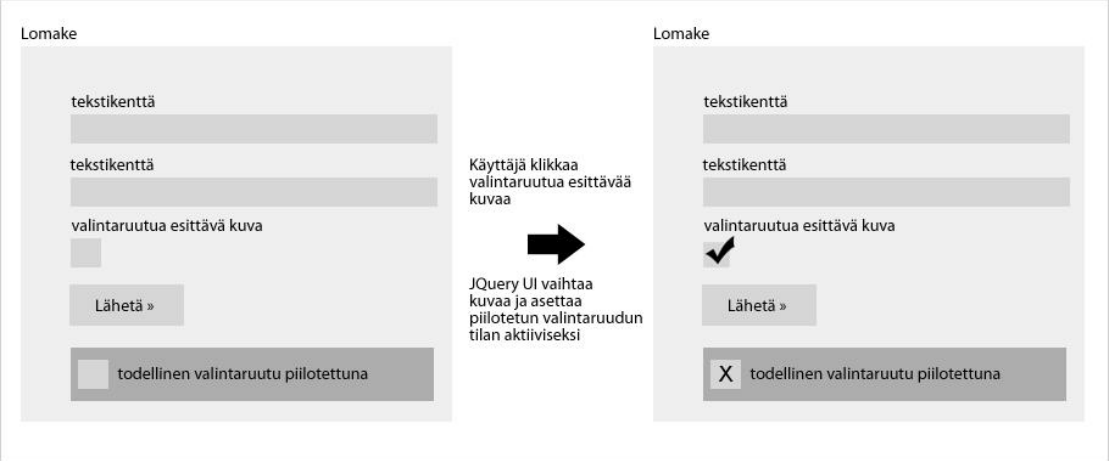
```
If (!empty ($_SESSION['userLevel']) &&
    $_SESSION['userLevel'] == ”loggedUser” ) {
    // sisältö kirjautuneelle käyttäjälle
} else {
    // normaali sisältö
}
```

Käyttäjän tietoja ei näytetä sivuilla eikä käyttäjän nimissä voi tehdä esimerkiksi tilauksia. Kirjautuneille näytetään vain useampia tuotteita ja tarkempia tietoja, kuten tukkuhinta, joten sessio-pohjainen tarkistus oli tietoturvan kannalta riittävä.

Erillisiä alasivuja oli siis suhteellisen vähän, mutta useimmille niistä toteutettiin erilliset sivupohjat, sillä jokaisella niistä oli erilaisia toiminnallisuuksia pelkän tekstin

sijaan. Kommentointimahdollisuutta ei ole, joten niiden räätälöinti voitiin jättää huomiotta.

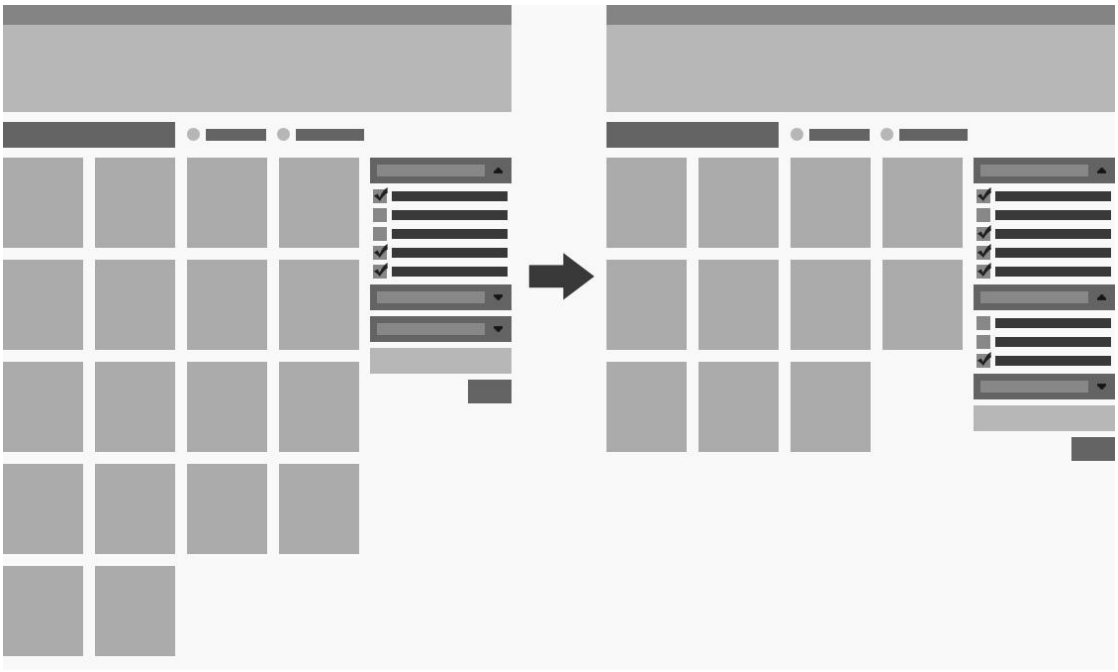
Sivuston lomakkeisiin haluttiin tavallisista poikkeavat valintaruudut, jotka sopivat paremmin muuhun ulkoasuun. Ongelmana kuitenkin oli että eri Internet-selaimet näyttävät HTML-lomakkeissa käytetyt tietueet eri tavalla ja esimerkiksi valintaruutuja ei voi tyylitellä selainriippumattomasti mitenkään pelkillä CSS-määrittelyillä. Tähän tarkoitukseen JQuery UI-kirjasto on kätevä: se on JQueryn päälle rakennettu käyttöliittymäkomponenttien kirjasto, jonka tarkoituksena on nopeuttaa käyttöliittymän rakentamista erityisesti AJAX-toteutuksissa. Komponenttien ulkoasu on hyvin muokattavissa, ja siksi se soveltuukin selainriippumattomien lomakkeiden tekemiseen. Periaatteeltaan komponenttien muokkaus on yksinkertainen: varsinainen komponentti piilotetaan ja sen tilalla näytetään kuva. Kun tila muuttuu, esimerkiksi valintaruutua esittävää kuvaa klikkaamalla, vaihdetaan piilotetun todellisen komponentin tila Javascriptillä ja samalla vaihdetaan myös valintaruutua esittävä kuva vastaamaan uutta tilaa (ks. kuvio 3). Näin käyttäjä näkee muokatun valintaruudun, mutta koska valinta on sidottu lomakkeeseen, valinta lähtee oikein lomakkeen lähetyksen mukana. Toimenpide on helppo toteuttaa käsin Javascriptillä ja varsinkin JQuery kirjaston avulla, mutta JQuery UI automatisoi prosessin ja säästää näin aikaa.



KUVIO 3. JQuery UI:n tapa luoda tyyliteltyjä lomakkeita

4.2 Tuotehaku

Tuotehaku oli yksi sivuston keskeisimmistä osa-alueista. Tietokantaan syötettiin satoja tuotteita ja haun haluttiin olevan reaaliaikainen siten, että käyttäjän muuttaessa hakukriteeriä, hakutulos tulee suoraan näkyviin ilman sivun päivitystä. Esimerkiksi valmistusmaan vaihtaminen pudotusvalikosta suorittaa uuden haun heti, eikä käyttäjän tarvitse painaa erillistä hae-nappia. Tämän vuoksi haut suoritettiin Javascriptillä ja AJAX -menetelmällä. Lisäksi hakukriteerit jaoteltiin ryhmittäin Javascriptin avulla siten, että tarpeettomat kriteerit voitiin piilottaa näkyvistä. Sivuston tuotehaun rakenne on kuvattu kuviossa 4.



KUVIO 4. Sivuston tuotehaun rakenne

Wordpress sisältää Javascript JQuery-kirjaston, ja sen tapa käsitellä AJAX-kutsut pohjautuu pitkälti JQueryn vastaavaan. Käytännössä suurin ero on siinä, että palvelinpuolen PHP-funktiot on esiteltävä ennen kuin niitä voi käyttää. JQueryn implementaatiosta johtuen Wordpress ei kuitenkaan salli AJAX-kutsuja muualla kuin Wordpressin hallintapaneelissa. Sivuston julkisella puolella AJAX on otettava käyttöön enque_script-metodilla. Mikään ei kuitenkaan estä kehittäjää asentamasta JQuery-kirjastoa sen normaalilla tavalla, mutta saman kirjaston käyttö kahteen kertaan on turhaa lisäladattavaa käyttäjälle ja hidastaa näin sivun latautumista.

Teemaan liittyvät PHP-funktiot on hyvä pitää functions.php-tiedostossa, jolloin ne ovat selkeästi yhdessä paikassa ja toimivat kaikkialla teemassa.

Koska haku toteutettiin asynkronisesti ilman sivunlatauksia, käyttäjän selaimen sivuhistoriaan ei tallentunut haun tuloksia tai valittuja hakukriteereitä. Hakuprosessissa tämä ei ollut ongelma, mutta kun käyttäjä valitsi tuotteen ja palasi takaisin hakuun jatkaen siitä mihin oli jäänyt, haussa ei ollutkaan enää samoja hakukriteereitä ja hakeminen piti aloittaa alusta. Tämä ongelma ratkaistiin tallentamalla valitut hakukriteerit ja -sanat jokaisen haun yhteydessä sessiomuuttujaan. Kun käyttäjä muuttaa hakukriteeriä ja Javascript lähettää tiedon asynkronisesti palvelimelle, tallennetaan hakukriteerit taulukoksi $_SESSION['haku']-muuttujaan ennen haun prosessointia.

```
$kriteerit = "";
foreach ($_POST as $nimi=>$arvo) {
     if ($arvo != "") {
          if (is_array($arvo)) {
               foreach ($arvo as $a)
                    $kriteerit.= "&".$nimi."%5B%5D=".$a;
          } else {
               $kriteerit.= "&".$nimi."=".$arvo;
          }
     }
}
if ($_POST['tyhjenna']) {
     unset($_SESSION['haku']);
     $_SESSION['haku'] = "&action=haeTuotteet_ajax&tyhjenna=1";
} else
     $_SESSION['haku'] = $ kriteerit;
```

Yksittäisen tuotteen näkymässä asiakas halusi joidenkin listattujen ominaisuuksien ja valmistajan nimen toimivan linkkinä tuotehakuun, jossa olisi esitäytettynä ko. valmistaja tai ominaisuus. Käyttäjä voisi näin nopeasti etsiä esimerkiksi saman valmistajan muut tuotteet. Tämä ratkaistiin lisäämällä tuotesivulle haluttuihin linkkeihin hakuehto PHP:n $_GET-muuttujana. Esimerkiksi valmistajan kohdalla linkkinä olisi ”/tuotehaku/?valmistaja=nimi”. Tuotteella saattoi kuitenkin olla esimerkiksi useita raaka-aineita. Ne piti siis tuoda taulukkona, jotta tuotehaku osaa käsitellä ne erikseen. Esimerkiksi ”/tuotehaku/?raaka-aine[]=alumiini&raaka-aine[]=teräs”.

Kun tuotehakusivu aukeaa, täytyi alussa tehdä kaksi tarkistusta. Ensimmäiseksi tarkistettiin, onko osoiterivillä annettu sallittuja hakuehtoja $_GET-muuttujina. $_GET-muuttujista parseroitiin vain sallitut sanat, jotta hakualgoritmille ei voi syöttää mitä tahansa ja näin tehdä injektiohyökkäystä. Jos valideja hakukriteereitä löytyi, suoritettiin haku niiden perusteella. Jos ei, tarkistettiin onko käyttäjä tehnyt aiempaa hakua tutkimalla sessiomuuttujaa, jolloin käyttäjä saa aiemman haun tulokset. Jos tätäkään ei löytynyt, näytettiin oletushaku, joka oli uutuustuotteet. Tarkistuksen toiminta on kuvattu tarkemmin liitteessä 1.

Kun hakukriteerit olivat tallennettuna sessioon, suoritettiin itse haku. Jokainen hakukriteerimahdollisuus käytiin läpi ja niistä tarkistettiin käyttäjän asettama rajaus. Näistä koottiin itse tietokantakysely, jonka tulokset palautetaan JSON-objektina takaisin käyttäjän selaimeen. Mahdollisista rajauksista osa oli ”AND” ja osa ”OR” -tyyppisiä ehtoja. Esimerkiksi jos käyttäjä on valinnut valmistusmaan, se tuli kyselyyn ”AND” -ehtona, jolloin vain kyseisen maan tuotteet tulivat valituksi. Toisaalta vapaasanahaun haluttiin toimivan niin, että siihen voi kirjoittaa esimerkiksi useita tuotenumeroita, jotka kaikki tuottavat tuloksen. Näin ollen jokainen hakusana oli ”OR” -tyyppinen ehto kyselyssä. Vapaasanahaun toteutusta jouduttiin muokkaamaan moneen kertaan ja sen toimintaperiaate muuttui välillä kokonaan, kun muutosten jälkeen aiemmat haut eivät enää tuottaneet haluttuja tuloksia. Aluksi sanahaku toimi säännöllisillä lausekkeilla (eng. regular expressions), mutta MySQL:n CONCAT toiminto osoittautui huomattavasti selkeämmäksi ja helpommaksi. Siinä yhdistetään sarakkeet, joista sanahaun halutaan hakevan ja tarkistetaan löytyykö hakusanaa tästä yhdistetystä merkkijonosta. Kyselyn periaate on esitetty alapuolella. Hakusana tulee tarkistaa injektioiden varalta.

```
SELECT tuote.tuoteID, tuote.nimi, kategoria.nimi AS kategoria,
FROM tuote
LEFT JOIN kategoria ON tuote.kategoria = kategoria.kategoriaID
WHERE CONCAT(tuote.nimi,' ',kategoria .nimi,' ',tuote.tuotenro)
LIKE '%hakusana%'
```

Hakupyyntö lähetettiin JQueryn post-metodilla, ja tuotetiedot palautettiin JSON-muotoisena merkkijonona takaisin käyttäjän selaimeen, josta Javascript voi lukea ne käyttäen JQueryn each-metodia.

```
$.post('<?php echo admin_url('admin-ajax.php'); ?>',
    data,
    function(response) {
   $.each(response,function(key, value) {
      if (value.kuva != "")
         var kuvatagi = '<img src="<?php
bloginfo('wpurl');?>/tuotekuvat/pieni/'+value.kuva+'"
class="product-image" />';
      else
         var kuvatagi = '';

   $('#tulokset').append(<a
href="/tuotteet/?tuote='+value.tuoteID+'" class="product-item"
title="'+value.nimi+'" id="product-
'+value.tuoteID+'">+kuvatagi+'<div class="product-
text"><h2>'+value.kategoria+'</h2><h3>'+value.nimi+'</h3></div>
</a>');
   });
}, "json");
```

MySQL-kyselyn rakentaminen, suorittaminen ja tietojen palautus palvelimella on esitetty liitteessä 2.

4.3 Tuotekortti ja muistilista

Tuotekortti oli toteutetun sivuston näkymä, jossa näytetään yksittäisen tuotteen tarkat tiedot. Tuotteen tiedoista näytettin kaikki vain kirjautuneille käyttäjille. Valitun tuotteen numero luettiin $_GET-parametrina osoiteriviltä periaatteella ?tuote=123, ja sitä käytettiin tuotetietojen hakuun tietokannasta. $_GET-parametrit ovat äärimmäisen helposti muokattavissa, joten ne pitää aina tarkistaa SQL-injektioiden varalta. Wordpress-julkaisujärjestelmässä on oma wpdb-niminen luokka, joka helpottaa MySQL-kyselyiden tekemistä. Tästä luokasta löytyy prepare-metodi, jolla injektioiden estäminen on nopeasti toteutettavissa. Prepare-metodin toimintaperiaatteena on, että muuttujia ei lisätä SQL-kyselyyn sellaisenaan, vaan niiden tilalle laitetaan erikoismerkit samaan tapaan kuin esimerkiksi PHP:n sprintf-funktiolla. Nämä erikoismerkit korvataan todellisilla muuttujilla vasta kun ne on tarkastettu. Käytännössä muuttujat pitää ajaa PHP:n mysql_real_escape_string-funktion läpi, mutta Wordpressin prepare-metodi tekee sen valmiiksi.

```
global $wpdb;
$tuote = $wpdb->get_row($wpdb->prepare(
         "SELECT tuote.tuoteID,
```

```
                tuote.nimi,
                tuote.teksti,
                tuote.valmistaja,
                tuote.hinta,
                kategoria.nimi AS kategoria
        FROM   tuote
        LEFT JOIN kategoria
        ON tuote.kategoria = kategoria.kategoriaID
        WHERE tuoteID = %s ", $_GET['tuote']
        )
    );
```

Sivustolle ei tullut verkkokauppaa, mutta asiakas halusi, että käyttäjä voi lisätä tuotteita muistilistaan, jonka voi edelleen lähettää sähköpostilla itselleen tai ostoista vastaavalle henkilölle. Tämä toteutettiin lisäämällä tuotekorttiin linkki ”Lisää muistilistaan”, joka tallentaa tuotteen numeron ja nimen sessiomuuttujan taulukkoon. Sessiomuuttujia voidaan muokata AJAX:lla, koska kyseinen tieto on tallennettuna palvelimella, ei käyttäjän koneella. Jos tuote on jo muistilistalla, käyttäjälle näytettiin linkki ”Poista muistilistalta”, joka vastaavasti poistaa tuotteen listalta. Lisäämis- ja poistamisfunktiot sijaitsevat muiden AJAX-funktioiden tapaan teeman functions.php-tiedostossa.

Kun käyttäjä lisää tuotteen muistilistaansa, luodaan taustalla kyseisestä tuotekortista PDF-tiedosto palvelimelle. PDF luodaan samaan tapaan myös silloin, kun käyttäjä painaa ”Lataa PDF”-linkkiä, mutta tällöin tiedosto syötetään suoraan käyttäjän selaimelle, eikä sitä tallenneta väliaikaisesti palvelimelle. Käyttäjälle toimenpide näkyy vain tiedoston latauksena. Palvelimelle tallennettaessa tiedoston nimeksi annetaan tuotteen numero, joka on tallennettuna myös muistilistan sessiomuuttujaan. Kun käyttäjä päättää lähettää muistilistan sähköpostilla, voidaan sähköpostiin liitetiedostoiksi lisätä PDF:t tuotenumeron perusteella. Sähköposti lähetettiin yksinkertaisella lomakkeella ja Wordpress-julkaisujärjestelmän omalla wp-mail-funktiolla, jolle annettiin header-tiedoissa mm. lähettäjän nimi ja liitetiedostojen osoitteet taulukkona.

```
$headers = 'From: '.$_POST['nimimerkki'].'
<'.$_POST['lahettajan_email'].'>' . "\r\n\\
Content-Type: text/plain; charset=ISO-8859-15; delsp=yes; for-
mat=flowed";
$attachments = array();
foreach ($_SESSION['muistilista'] as $item) {
   $filename = $wpdb->get_var( $wpdb->prepare("SELECT tuotenro
```

```
FROM tuote WHERE tuoteID = %s",$item)).".pdf";
   $attachments[] = "/var/www/.../pdf-tmp/".$filename;
}

wp_mail( $_POST['email'], 'Otsikko', utf8_decode($msg), $head-
ers, $attachments );
```

PDF:n luomiseen käytettiin ilmaista DOMPDF-luokkaa, joka osaa luoda annetusta HTML-dokumentista PDF-tiedoston. Sen käyttö oli varsin yksinkertaista, vaikkakin merkistökoodauksen kanssa oli ongelmia: oletuksena DOMPDF käytti ANSI-merkistöä, mutta dokumentin haluttiin olevan UTF-8. Ongelmat ratkesivat muuttamalla tulostettava HTML-dokumentti UTF-8 merkistöön PHP:n avulla.

```
require_once("../dompdf_config.inc.php");
$filename = $wpdb->get_var( $wpdb->prepare("SELECT tuotenro
FROM tuote WHERE tuoteID = %s",$tuote)).".pdf";

// generatePdf sisältää tuotekortin HTML:n $html-muuttujassa
require_once("../generatePdf.php");
$dompdf = new DOMPDF();
$html = mb_convert_encoding($html, 'HTML-ENTITIES', 'UTF-8');
$dompdf->load_html($html);
$dompdf->set_paper("a4", "portrait");
$dompdf->render();
$file = $dompdf->output();
file_put_contents("../pdf-tmp/".$filename, $file);
```

DOMPDF taittoi tuotekortin pääsääntöisesti hyvin vastaamaan oikeaa ulkoasua, mutta joiltain osin korjauksia jouduttiin tekemään HTML-rakenteeseen sekä CSS-tyylimäärityksiin. Lisäksi DOMPDF katkoi merkkijonoja vääristä kohdista ja asetti osan sanoista päällekkäin. Merkkijono-ongelmat korjaantuivat PHP:n wordwrap-funktiolla, joka pakottaa rivinvaihdon tietyn merkkimäärän välein.

Myös muita PDF-muuntimia, kuten TCPDF-luokkaa kokeiltiin, mutta DOMPDF antoi heti lähimmäksi oikean näköisen tuloksen ja sen käyttö oli yksinkertaista.

4.4 Nostotuotteet

Kirjautuneille käyttäjille haluttiin näyttää erilainen etusivu, jossa voidaan nostaa esille ajankohtainen tapahtuma sekä kaksi siihen mahdollisesti liittyvää tuotetta. Ajankohtainen tapahtuma valitaan sivuston ajankohtaista-osiosta. Tällaisia etusivuja

voidaan luoda vaikka vuoden jokaiselle päivälle erilainen, mutta käytännössä sen on tarkoitus muuttua muutaman viikon välein. Käyttäjä voi selata näitä eri viikkojen nostoja taaksepäin, mutta tulevaisuuden nostot ovat piilotettuja, kunnes niiden ajastettu aika on ohitettu.

Kirjautuneiden käyttäjien etusivuja varten luotiin erillinen sivupohja ja sen sisällönhallintaan erillinen työkalu, sillä Wordpressin sivujen hallinta ei sellaisenaan soveltunut riittävän hyvin. Sisällönhallinnan työkalua käsitellään enemmän luvussa 6.8.

4.5 Sisällön rajaus eri käyttäjätasoille

Sivuston sisältö jaettiin eri käyttäjäryhmille: yritysasiakkaat, englanninkieliset ja muut. Wordpress-julkaisujärjestelmän suurimpia heikkouksia on huono natiivi tuki eri kieliversioille. Tämän sivuston kohdalla ongelma ei kuitenkaan ollut suuri, sillä englanninkielinen osuus oli muuta suppeampi koostuen lähinnä yritysinfosta ja yhteystiedoista. Tämä osuus voitiin tehdä vain lisäämällä hallintapaneelista omat sivupohjat englanninkielisille sivuille. Kirjautumissivulta käyttäjä ohjattiin englanninkieliseen yritysinfoon. Ns. header- ja footer-osioihin, joissa oli navigaatio, piti kuitenkin myös lisätä tarkistus kieliversiosta. Tarkistus voitiin tehdä yksinkertaisella if-else-lauseella, sillä kieliversio tallennettiin sessiomuuttujaan kirjautumissivulla. Pääsivut kovakoodattiin navigaatioon käyttäen niiden id-tunnistetta wp_list_pages-funktiossa.

```
<ul id="menu">
<?php
if ($_SESSION['userLevel'] == ”loggedUser”)
   wp_list_pages('include=5,7,11,13&title_li=');
else if ($_SESSION['userLevel'] == "en")
   wp_list_pages('include=91,204&title_li=');
else
   wp_list_pages('include=5,11,13&title_li=');
?>
</ul>
```

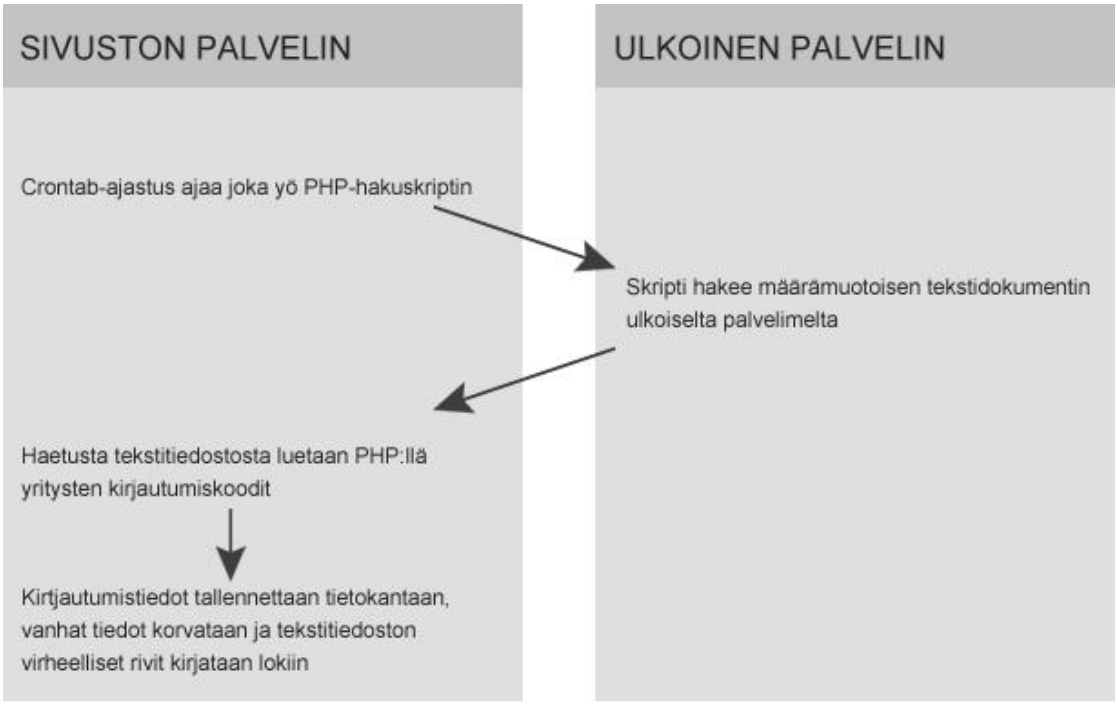
Kirjautuneiden ja tavallisten kävijöiden sisältö tuli samoista sivupohjista, mutta niiden joukkoon oli sijoitettu if-else-rakenteita, joiden perusteella sisältöä joko

näytettiin tai piilotettin. Vaihtoehtona olisi ollut luoda omat sivupohjat eri käyttäjäryhmille, mutta erot olivat lopulta melko pieniä ja muutosten ylläpito oli helpompaa, kun ne tarvitsi tehdä vain yhteen tiedostoon.

Jokaiselle tuotteelle lisättiin tieto näkyvyydestä eri käyttäjätasoille. Haun yhteydessä käyttäjän taso lähetettiin palvelimelle muiden hakukriteerien kanssa, joten se voitiin ottaa huomioon hakukyselyä rakennettaessa.

4.6 Kirjautumistietojen ajastettu haku

Yritysasiakkaat voivat kirjautua sivustolle Valvira-lupanumerolla, jolloin heille näytetään mm. enemmän tuotteita ja ylimääräistä tietoa niistä. Jotta kirjautumistiedot olisivat ajan tasalla, palvelimelle tehtiin skripti, joka käy joka yö hakemassa Valviran sivuilta lupanumerot ja syöttää ne tietokantaan. Toimenpide on kuvattu lyhyesti kuviossa 5.



KUVIO 5. Kirjautumistietojen haku ulkoiselta palvelimelta

Nebulan webhotel toimii Linux-käyttöjärjestelmällä, joten ajastus toteutettiin Linuxista löytyvällä cron-ajastuspalvelulla. Sama toiminto löytyy myös kaikista Unix-

pohjaisista käyttöjärjestelmistä. Crontab on ohjelma, joka tarkistaa minuutin välein cronille asetetut ajastettavat komennot ja suorittaa ne.

Lupanumerot löytyvät Valviran sivuilta yksinkertaisena tekstitiedostona, joten varsinainen lupanumero ja siihen liittyvä yritysinfo piti parseroida sieltä rivi riviltä ennen tietojen syöttämistä tietokantaan. Kun PHP:llä luetaan tekstitiedosto ulkopuoliselta palvelimelta esimerkiksi fopen -komennolla, palvelimella pitää olla sallittuna allow_url_fopen niminen asetus. Nebulalla tämä ei ole oletuksena päällä, joten se pitää antaa ajettavalle skriptille erikseen.

```
5 2 * * * php -d allow_url_fopen=1 /var/.../getCodes.php
```

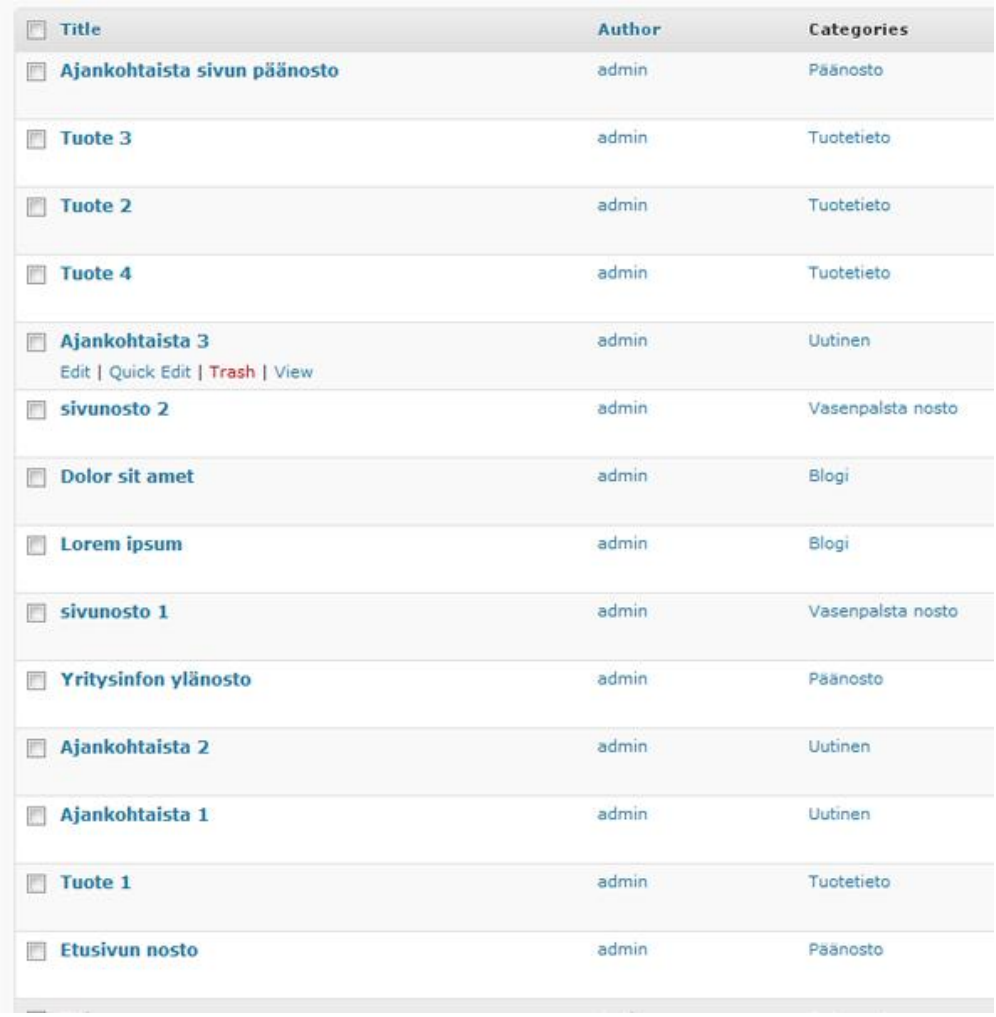
Kun PHP:llä aloitettiin tekstitiedoston parserointi, ensimmäisenä tarkistettiin löytyikö kyseistä tiedostoa ollenkaan. Tähän tarkoitukseen toimii hyvin HTTP-tilakoodien lukeminen. Kun PHP:n fopen-komennolle annetaan tiedosto, voidaan sen HTTP-pyynnöstä lukea pyynnön tila (eng. status code). Esimerkiksi useimmille tutuin tilakoodi 404 tarkoittaa, että pyydettyä tiedostoa ei löydy (eng. not found). 403 taas tarkoittaa, että kyseiseen tiedostoon ei ole oikeuksia (eng. forbidden). Näitä koodeja on kymmeniä erilaisia, mutta tiedoston löytymiseen riittää koodi 200, joka yleisesti tarkoittaa ”OK”. Toisin sanoen tiedosto on saatavilla, jos sen hakemiseen käytetyn HTTP-pyynnön tila on 200. Luetun tekstitiedoston parserointi koodina on esitetty tarkemmin liitteessä 3.

Jokaisesta lupanumeroiden hausta pidetään kirjaa sitä varten luodulla tietokannan taululla. Tauluun kirjataan joka yö haussa mahdollisesti tapahtuneet virheet sekä lisättyjen lupanumeroiden määrä. Jos kirjautumisten yhteydessä ilmenee ongelmia, voidaan tätä lokia käyttää apuna ongelman selvittämisessä.

5 TUOTETIETOKANTA

Kun halutaan tallentaa tuotetietoja Wordpress-julkaisujärjestelmään, se voidaan toteuttaa ilman omia tietokantatauluja käyttäen hyväksi Wordpressin sisältötyyppejä. Tällöin datan integraatio Wordpressiin on saumaton. Vaihtoehtona on luoda tietokantaan taulut tuotteille ja rakentaa Wordpressin hallintapaneeliin omat työkalut näiden tietojen muokkaamiseen, jolloin kehittäjällä on täysi kontrolli tietojen relaatioihin ja hallinnan ominaisuuksiin.

Wordpress 3.0 esitteli uuden mukautetun tietotyypin periaatteen, jolla sivuston eri sisältötyyppejä voidaan jakaa selkeämpiin kokonaisuuksiin. Normaalisti Wordpress käyttää vain kahta sisältötyyppiä: artikkelit ja sivut. Perinteisesti näitä kahta on käytetty varsin luovasti, jotta hallintapaneelista saataisiin muokattua esimerkiksi sivuston nostoja tai tuotteita. Keinoina on esimerkiksi käyttää sivujen hierarkiaa tai jakaa sisältötyypit artikkeleiden kategorioilla. Kategoriajako on sinänsä toimiva ja periaatteessa looginen ratkaisu, mutta käytännössä sekava hallita, koska artikkelit ovat kuitenkin samassa listassa. Asiaa voisi verrata tiedostokansioon, jossa on eri tiedostotyyppejä sekaisin. Kuviossa 6 näkyy esimerkki artikkelilistauksesta, jossa artikkeleita on käytetty eri sisällön esittämiseen jakamalla ne kategorioittain.

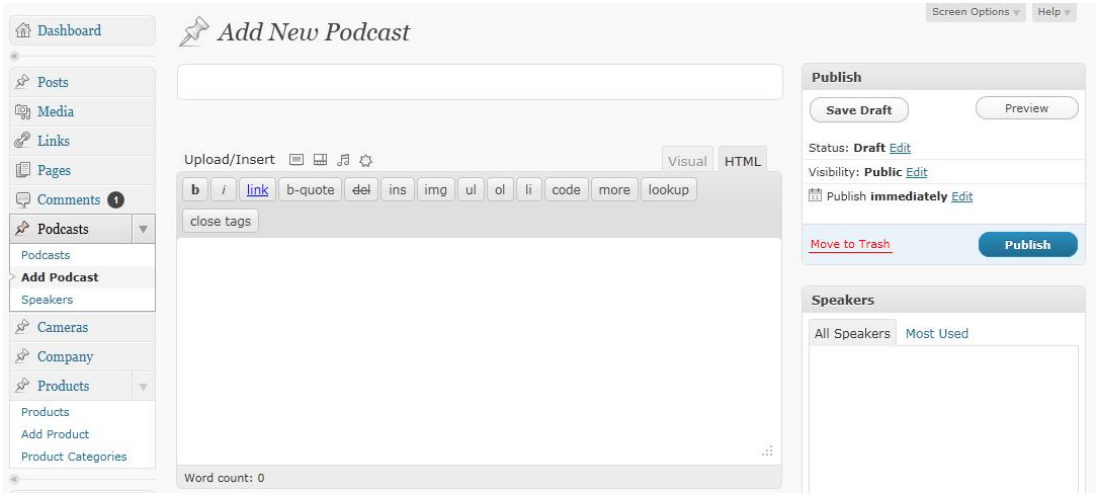


KUVIO 6. Artikkelikategorioiden käyttö sisällön jaotteluun

Mukautetun tietotyypin avulla sisältöä voidaan sivustokohtaisesti jakaa loogisempiin kokonaisuuksiin, kuten esimerkiksi videoihin, palveluihin tai markkinointisivun tapauksessa tuotteisiin. Etuna on, että tieto on säilöttynä samalla tavalla kuin muutkin Wordpressin sivut ja tiedon hallintanäkymä on oletuksena hyvin samanlainen. Lisäksi näissä voidaan käyttää hyväksi samoja meta-tietoja, kuten jaottelua kategorioittain tai avainsanoin. Sisällön hakutoiminto toimii suoraan myös mukautettuihin tietotyyppeihin. Huonona puolena on, että kehittäjällä ei ole täydellistä kontrollia tietotyyppien relaatioihin. Esimerkiksi tuotteella voi olla yksi valmistusmaa ja useita valmistusmateriaaleja. Toisaalta monella tuotteella voi olla sama valmistusmaa ja joitain samoja, mutta ei välttämättä kaikkia valmistusmateriaaleja. Valmistusmaa on esimerkki ns. yksi moneen-relaatiosta ja valmistusmateriaali ns. moni moneen relaatiosta. Tuotetietotyyppi voidaan kyllä jakaa erilaisiin kategorioihin ja nimetä ne maan mukaan, sekä antaa erilaisia

avainsanoja, jotka taas voidaan nimetä valmistusmateriaalien mukaan, ja näin luoda tuotteille yhteisiä relaatioita muihin tietueisiin. Kategoria- ja avainsanapohjainen jaottelu on käytössä Wordpressin artikkeleissa ja on nopea ottaa käyttöön myös mukautettuihin tietotyyppeihin. Jos relaatioiden yhteyteen pitää kuitenkin tallentaa muutakin tietoa, esimerkiksi valmistusmateriaalin prosenttiosuus, avainsanat eivät siihen taivu, sillä silloin jokaiselle prosentille pitäisi määrittää oma avainsanansa. Esimerkiksi alumiini 50 %, alumiini 77 % jne. Yksi moneen-relaatiot kuitenkin onnistuvat kategoriajaottelulla mainiosti.

Mukautettu tietotyyppi on siis pääasiassa työkalu sisällön loogiseen jaotteluun ja se helpottaa sisällön tuottamista niille, jotka eivät tunne sivuston toimintalogiikkaa. Omat sisältötyypit näkyvät hallintapaneelin vasemmassa laidassa (ks. kuvio 7). Ilman jaottelua helpottavia laajennuksia, niiden luominen vaatii kuitenkin hieman enemmän työtä verrattuna siihen, että vain jakaisi sisällön eri artikkelikategorioihin tai alasivuihin. Mukautetun sisältötyypin esittäminen teemassa sivuston julkisella puolella ei kuitenkaan juuri eroa tavallisista sivuista tai artikkeleista, jos ne on vain suunniteltu hyvin etukäteen. Sisältötyyppien luomiseen löytyy valmiina hyviä laajennuksia, jotka helpottavat niiden luomista huomattavasti.



KUVIO 7. Näkymä hallintapaneelista, kun sivustolle on lisätty omia sisältötyyppejä.

Vaihtoehtona, johon myös toteutetulla sivustolla päädyttiin, on luoda tietokantaan tuotetiedoille omat taulut. Tällöin tiedot ovat erillään itse julkaisujärjestelmästä ja yhteys siihen on laajennus, joka on itse toteutettava. Laajennuksella luotiin omat hallintasivut ja -työkalut tuotetietojen muokkaukseen Wordpress-

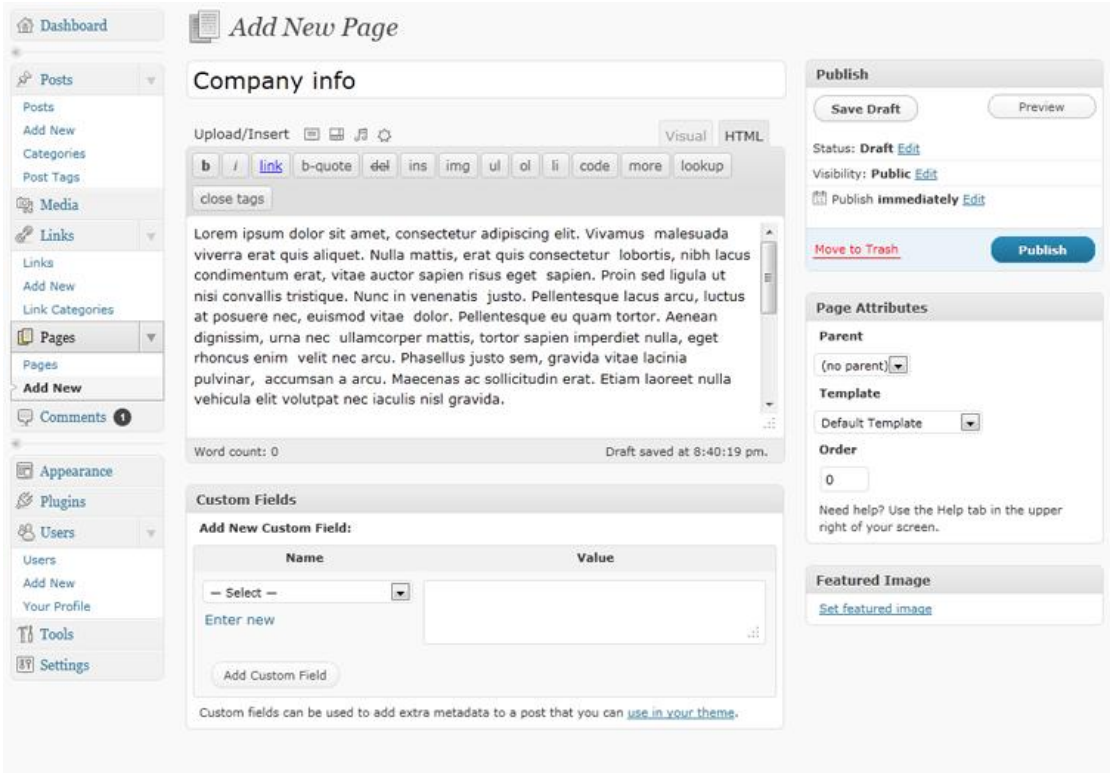
julkaisujärjestelmän ylläpitäjille. Tuotetiedot ovat oma kokonaisuutensa ja ne voidaan helposti siirtää sivustolta toiselle, jos julkaisujärjestelmää joskus vaihdetaan. Lisäksi rakenteen suunnittelussa oli vapaat kädet järjestää relaatiot parhaaksi katsomalla tavalla.

Oman laajennuksen toteutus vie luonnollisesti enemmän aikaa ja tuotteilla oli eri ominaisuuksia yhteensä noin 35. Näistä kaksi tuli ns. moni moneen-relaatioilla, ja kahdeksan yksi moneen-relaatioilla muista tauluista. Tämä tarkoitti sitä, että taulujen relaatiot tuli suunnitella hyvin ja viiteavaimet luoda oikein. Hyvin luotu viite-eheys kuitenkin helpottaa myöhempiä muokkauksia, kun yhden kategorian nimen vaihto tai poisto siirtyy automaattisesti sitä käyttäviin tuotteisiin.

6 SISÄLLÖNHALLINTA

6.1 Perustason sisällönhallinta

Wordpress sisältää perusasennuksella melko kattavan hallintapaneelin sisällön ja käyttäjien muokkaamiseen. Sisältötekstien syöttöön on ns. WYSIWYG-editori ("what you see is what you get"), jonka avulla sisällöntuottaja voi ilman HTML-tietämystä tehdä perustason tekstin muokkausta. Hän voi esimerkiksi lihavoida ja kursivoida tekstiä tai lisätä linkkejä sekä kuvia. Kuviossa 8 näkyy perusnäkymä Wordpressin tekstieditorista. Edistyneemmät kehittäjät voivat käyttää HTML:ää sisällön muokkaamiseen monipuolisemmin. Sivuja voi piilottaa asettamalla niiden tilaksi luonnos (eng. draft). Avustajatason (eng. contributor) käyttäjä ei voi itse julkaista kirjoituksiaan, vaan ne pitää hyväksyttää ylemmän tason käyttäjällä. Sivut voidaan järjestää hierarkkisesti ylä- ja alasivuihin.



KUVIO 8. Sivunmuokkausnäkymä Wordpress-julkaisujärjestelmässä

Artikkeleiden hallinta on hyvin samanlainen kuin sivuilla. Erona sivuihin artikkeleille voidaan antaa avainsanoja ja jakaa eri kategorioihin, mutta niitä ei voi järjestää hierarkkisesti. Lisäksi niiden julkaisu voidaan ajastaa tulevaisuuteen, jolloin ne voidaan kirjoittaa valmiiksi etukäteen. Artikkelit soveltuvat hyvin blogiviesteihin tai esimerkiksi uutispalstan ylläpitoon.

Wordpress jakaa käyttäjätasot eri roolin mukaan. Näitä rooleja on kuusi, mutta käytännössä useimmissa tapauksissa kuitenkin vain viisi, sillä ns. Super Admin-rooli on tilanteisiin, jossa Wordpress asennuksia on samaan aikaan useampia. Super Adminilla on täydet oikeudet kaikkiin asennuksiin, mutta tavallisella Admin-roolilla vain yksittäiseen blogiin. Alimman tason käyttäjä pystyy ainoastaan muuttamaan omia tietojaan, joten se soveltuu hyvin sivustoihin, jotka vaativat rekisteröinnin. Puhtaalla asennuksella käyttäjien tietojen ja tason hallinta sujuu hyvin, mutta tarkempi oikeuksien määrittely ei Ilman laajennuksia onnistu. Ylläpitäjä ei voi antaa erikoisoikeuksia yksittäiselle käyttäjälle, vaan oikeudet ovat aina sidottuja tiettyihin rooleihin. Esimerkiksi jos haluaisi antaa yhdelle avustajalle oikeuden ladata kuvia mediakirjastoon, se ei onnistu antamatta samaa oikeutta myös muille avustajille. Tähän on onneksi laajennuksia, joilla ongelman voi ratkaista.

Wordpressin sisällönhallinta on perusasennuksella kohtalaisen hyvä useimpiin tilanteisiin ja se on laajennettavissa monilla valmiilla laajennuksilla.

6.2 Palautelomakkeet

Sivuston ainoa osio, jossa käytettiin yhteisön luomia laajennuksia, oli palautelomakkeet. Contact form 7 on Takayuki Miyoshin kirjoittama kevyt, mutta joustava työkalu palautelomakkeiden luomiseen ja hallintaan Wordpress-julkaisujärjestelmässä. Sen etuna oli nopea käyttöönotto ja helppo hallinta useammalle eri palautelomakkeelle. Näin eri käyttäjänäkymiin saatiin helposti omat lomakkeet, joissa voidaan käyttää eri tietoja, ja jotka voidaan ohjata eri vastaanottajille. Käytännössä lomakkeeseen lisätään hallinnasta halutut syötekentät valikoimasta tuettuja kenttiä, jonka jälkeen se liitetään halutulle sivulle. Lisätyötä toi lomakkeen ja virheilmoitusten tyylittely muuhun ulkoasuun sopivaksi.

Laajennusten etsiminen ja asentaminen on helppoa. Jos palvelimen käyttäjällä on tarvittavat oikeudet tiedostojen hallintaan, voi laajennuksia etsiä ja asentaa suoraan Wordpress-asennuksen hallinnasta ilman ftp-yhteyttä. Hallinnassa näkyvät asennettavat laajennukset tulevat wordpress.org sivulta, mutta niitä voi ladata muualtakin. Tällöin ne on kuitenkin siirrettävä palvelimelle käsin ftp-yhteyden yli.

6.3 Tuotetietojen hallinta

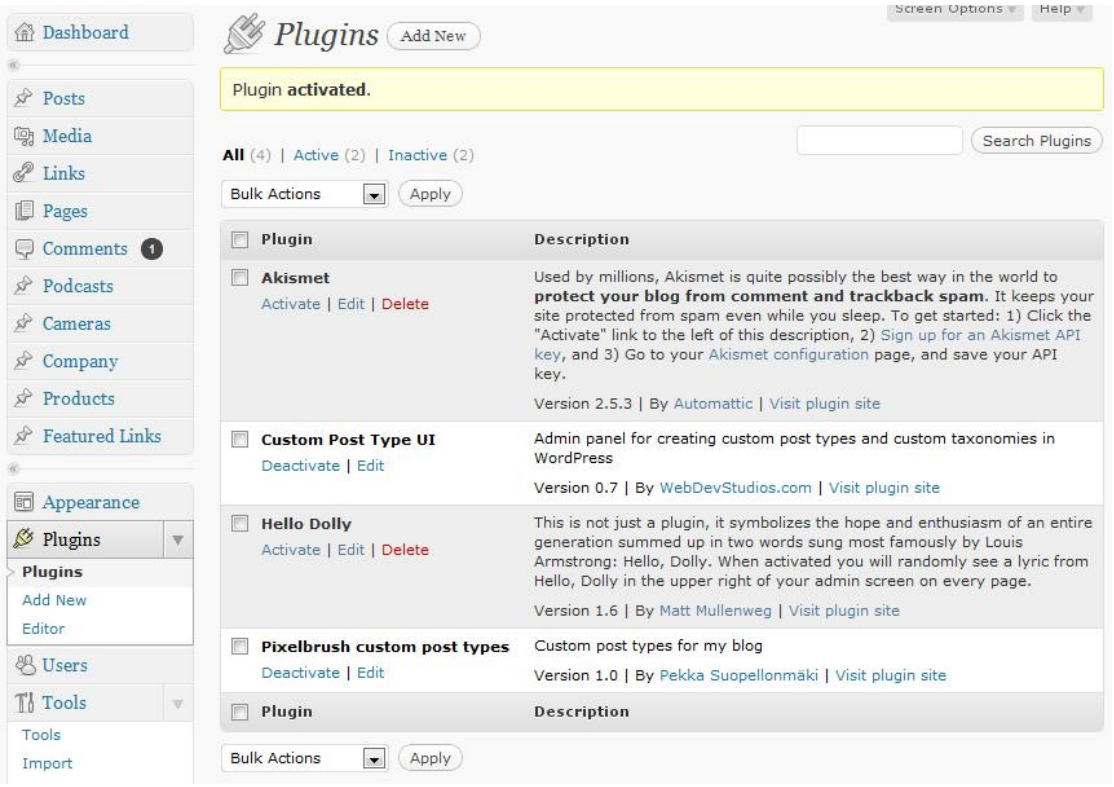
6.3.1 Hallintasivujen luonti

Sivuston tuotteet päätettiin tallentaa omiin tietokannan tauluihin, joten niiden muokkaamiseen oli luotava omat työkalut. Wordpress-hallintapaneeliin voi lisätä omia sivuja, joiden näkyminen voidaan rajata vain tietyn tasoisille käyttäjille. Näille sivuille kehittäjä voi käytännössä vapaasti tehdä lomakepohjaisia tietokannanhallintasivuja käyttäen hyväksi mm. Wordpress-julkaisujärjestelmän omaa tietokantaluokkaa. Hallintasivu rakennetaan käytännössä laajennuksena.

Laajennukset luodaan Wordpress-asennuksen wp-content/plugins-kansioon. Laajennus voi olla pieni, vain yhden tiedoston kokoinen, jolloin se voidaan sijoittaa plugins-kansion juureen. Laajemmille toteutuksille kannattaa luoda oma kansiorakenne, sillä Wordpress osaa tutkia alikansiot laajennoksia etsiessään. Laajennos tunnistetaan PHP-tiedoston alussa olevasta kommentista.

```
<?php
/*
Plugin Name: Tuotteiden hallinta
Plugin URI:
Description: Lorem ipsum dolor sit amet
Author: Pekka Suopellonmäki
*/
?>
```

Tällöin laajennus näkyy Wordpress-hallintapaneelin laajennukset-sivulla (eng. plugins), josta sen voi aktivoida (ks. kuvio 9)



KUVIO 9. Wordpress-laajennukset hallintapaneelissa

Hallintasivu esitellään add_menu_page ja add_submenu_page-funktioilla. Funktiolle annetaan parametreina sivun nimi ja otsikko, käyttäjän vähimmäisoikeustaso, sivun koodiystävällinen nimi sekä funktion tai tiedoston nimi, josta sivun sisältö haetaan. Alasivulle annetaan ensimmäisenä parametrina yläsivun nimi.

```
add_menu_page('Tuotehallinta', 'Tuotehallinta', 'read',
    'tuotehallinta', 'productHandler');
add_submenu_page( 'tuotehallinta', 'Maat', 'Maat', 'read',
'maat', 'maatHandler');
```

Yllä olevassa esimerkissä luodaan ylätason sivu Tuotehallinta, ja sille alasivu Maat. Kun käyttäjä valitsee hallintapaneelissa tuotehallinnan, suoritetaan funktio productHandler. Sivustolla käytettiin tapaa, jossa kaikki alasivut luotiin suorittamalla ne niille määrätyistä funktioista, eikä suoraan tiedostoista. Tällä tavalla laajennuksen päätiedostosta voitiin tehdä kaikkia alasivuja koskevia ratkaisuja. Voitiin esimerkiksi liittää sama tyylitiedosto kaikille alasivuille wp_register_style- ja wp_enqueue_style-funktioilla.

```
wp_register_style('tuotehallintaTyylit',
    WP_PLUGIN_URL . '/tuotehallinta/style.css');
```

```
wp_enqueue_style('tuotehallintaTyylit');

function tuotteetHandler() {
   global $wpdb;
   echo '<div class="wrap tuotehallinta">';
   // muokkaus tai uusi tuote
   if (isset($_GET['tuote'])){
      require_once('tuotemuokkaus.php');
   } else {
      require_once('tuotelistaus.php');
   }
   echo '</div>';
}
```

6.3.2 Yksittäisen tuotteen hallinta

Yksittäisen tuotteen hallinta oli työmääräisesti tuotehaun kanssa suuritöisimpiä. Tuotehallintaan tehtiin samanlainen AJAX-pohjainen haku pienemmässä mittakaavassa kuin sivuston julkiselle puolelle, sillä muokattavan tuotteen etsiminen sivutetulta listalta olisi ollut melko hidasta. Toisaalta kaikkia tuotteita ei ole järkeä listata sivulle, jota ylläpitäjä joutuu avaamaan usein, sillä se tuottaa turhaa kuormaa palvelimelle ja hidastaa sivun latautumista. Hakua varten tehtiin sanahakukenttä ja pudotusvalikko, josta voi valita kategorian. Toisin kuin julkisella puolella, hallinnassa haku suoritetaan vasta Hae-nappia painamalla, eikä heti kun ylläpitäjä valitsee toisen kategorian. Sanahaku asetettiin hakemaan tuotteiden nimiä ja tuotenumeroita.

Tuotteilla oli todella monia ominaisuuksia ja relaatioita eri tauluihin, kuten luvussa 5 mainittiin. Tämä vaati tarkkuutta ja järjestelmällisyyttä syötteiden tarkistuksessa, mutta virheiltä ei silti vältytty. Erityisesti tuotteiden raaka-aineiden kanssa jouduttiin tekemään paljon työtä, sillä myös raaka-aineen prosenttiosuus tuli tallentaa.

Ennen lomakkeen tulostusta tuotteen tiedot haetaan $_GET-parametrina annetun id-numeron perusteella. Jos parametri on tyhjä, tulkitaan lomake uuden tuotteen lisäykseksi. Näin voitiin käyttää samaa lomaketta sekä uuden tuotteen lisäykseen että muokkaamiseen. Alla olevan esimerkin mukaisesti kaikki tietueet tallennettiin omaan muuttujaan, jonka arvoa muutettiin sen mukaan, liittyykö se muokattavaan tuotteeseen tai, jos lomake on lähetetty, tallennetaan siihen muuttunut arvo. Näin lomakkeen kentissä oli aina oikea arvo riippumatta siitä, oliko tuote uusi, muokattu tai jos virheen takia lomakkeen tietoja ei lähetyksessä tallennettu.

```
$kategoriat = $wpdb->get_results("SELECT * FROM kategoria");
$tuote = $wpdb->get_row( $wpdb->prepare("SELECT * FROM tuote
WHERE tuoteID = %s" , $_GET['tuote']) );
if ($tuote) {
   // Tuotetta muokataan
   $nimi = $tuote->nimi;
   $kategoria = $tuote->kategoria;
} else {
   // Tuotetta ei ennestään löydy
   $nimi = "";
   $kategoria = "";
}
if (isset($_POST['tallenna'])) {
   $nimi = $_POST['nimi'];
   $kategoria = $_POST['kategoria'];
}

<input type="text" name="nimi" value="<?php echo $nimi; ?>" />
<select id="kategoria" name="kategoria">
<?php foreach ($kategoriat as $k)
   echo '<option value="'.$k->id.'" '.($kategoria == $k->id ?
'selected="selected"' : '' ).' >'.$k->nimi.'</option>'; ?>
</select>
```

Tuotteen hallinnassa käytettiin paljon Javascriptiä hyödyksi. Esimerkiksi tuotteen poistaminen havainnollistettiin ylläpitäjälle hyvin selkeästi, jotta hän ei tee sitä huomaamattaan. Tuotteen poistamiseksi ylläpitäjän on ensin aktivoitava valintaruutu, jonka jälkeen Poista-nappi korostetaan punaisella. Näin välitettiin osaltaan tieto siitä, että toimenpidettä ei voi peruuttaa. (ks. kuvio 10).

KUVIO 10. Tuotteen poistaminen hallintapaneelissa

Lisäksi mm. maiden alueet päivittyivät AJAX-periaatteella aina, jos ylläpitäjä vaihtoi tuotteen valmistusmaata pudotusvalikosta. Lisäksi tuotteen tietyt ominaisuudet piilotettiin, jos ne eivät kuuluneet tuotteelle valittuun kategoriaan.

6.3.3 Relaatiot muihin tietoihin

Kuten yleensä tuotelistauksissa ja -tietokannoissa, eri tuotteilla on monia yhdistäviä ominaisuuksia, kuten esimerkiksi valmistaja, kategoria tai vaikka vain väri. Samojen tietojen tallentaminen jokaiselle tuotteelle erikseen on paitsi turhaa myös helposti virheitä aiheuttavaa. Kirjoitusvirhe aiheuttaa virhetulkintoja, kun osalla tuotteista valmistajana on Yritys ja toisilla Yrritys. Relaatiotietokantaa, kuten esimerkiksi MySQL:ää, ei pidä ajatella pelkkänä tietovarastona vaan kannattaa käyttää hyväksi mahdollisuutta linkittää tietoja yhteen.

Valmistusmaille, kategorioille ja muille toisista tauluista tuleville tiedoille luotiin omat yksinkertaiset hallintasivut niiden muokkaamiseen. Kategorioita on suhteessa selkeästi vähemmän kuin itse tuotteita, joten ne voitiin yksinkertaisesti listata. Niistä klikkaamalla voi kyseisen kategorian tietoja muokata. Koska kategoria on viiteavaimella liitetty tuotteeseen, kategorian nimen muutos näkyy kaikissa tuotteissa, johon se on määritelty. Viiteavaimeen on määritelty ON UPDATE-tapahtumaan asetus CASCADE, ja ON DELETE-tapahtumaan SET NULL, jolloin kategorian muutokset periytyvät sitä käyttäviin tuotteisiin. Jos kategoria poistetaan, asetetaan sitä käyttävien tuotteiden kategoriaksi NULL.

6.3.4 Tuotetietojen tulostus excel-tiedostoon

Tietokannan tietojen tulostaminen sivuston hallintapaneelista excel-tiedostoon on verrattain yksinkertaista. Palvelimelta palautetaan tiedosto, jonka otsikkotiedoissa käyttäjän selaimelle kerrotaan kyseessä olevan excel-tiedosto, jolloin selain tarjoaa käyttäjälle tiedoston latausta. Tiedoston nimeen on hyvä antaa jokin yksilöllinen tunniste, jotta sen erottaa aiemmista latauksista. Esimerkiksi päivä ja kellonaika auttavat pitämään tiedostot järjestyksessä.

```
<?php
     $file = Tuotelistaus.'_'.date("Y-m-d_H-i").'.xls';
     header("Content-type: application/ms-excel");
```

```
    header("Content-Disposition: attachment; filename=$file")
?>
```

Tämän jälkeen tarvitsi suorittaa tietokannan kysely ja lopuksi tulostaa tiedot excelin ymmärtämässä muodossa. Solusta toiseen siirrytään sarkaimella ja rivinvaihto tietojen tulostuksessa näkyy rivinvaihtona myös excel-tiedostossa. Selkeyden vuoksi tiedot tulostettiin yhtenä merkkijonona lopussa, jolloin sen merkistökoodaukseen voitiin vaikuttaa. Näin koko excel-tiedosto voitiin tulostaa utf-8 merkistökoodauksella.

Hallintapaneliin lisättiin myös mahdollisuus ladata tuotteet CSV-tiedostona. CSV on lyhenne sanoista comma separated values, ja se on tekstitiedosto, jossa tietueet on jaoteltu jollain erikoismerkillä, usein puolipisteellä. Taulukkolaskentaohjelmat, kuten MS Excel ja Open Office Calc, osaavat näyttää tietueet soluittain, jolloin tieto on helposti muokattavissa. CSV:n latausmahdollisuus lisättiin hallintapaneelin siksi, että tuotteiden eräajoa suoritetaan CSV-tiedostosta, ja ylläpitäjä voi ladata esimerkkitiedoston. Vaihtoehtoisesti ylläpitäjä voi ladata kaikki tuotetiedot yhdessä tiedostossa, tehdä muutokset ja ladata tiedoston takaisin palvelimelle. Eräajon toteutus on selitetty luvussa 6.3.4.

CSV-tiedostossa tuotteiden pidemmät tietueet, kuten kuvausteksti, piti ottaa huomioon tietojen tulostuksessa. Koska CSV-tiedosto luetaan riveittäin, kuvaustekstin rivinvaihdot aiheuttavat ongelman, sillä jokainen uusi rivi tulkitaan omaksi tuotteeksi. CSV-tiedostossa voi käyttää useampia rivejä, mutta teksti pitää ympäröidä lainausmerkeillä. Tämä korjasi ongelman, kunnes huomattiin, että osa teksteistä sisältää lainausmerkkejä, jotka taas sekoittivat CSV-tiedoston. Tämä korjattiin korvaamalla tekstin lainausmerkit kaksinkertaisilla lainausmerkeillä, sillä ne tulkitaan tavallisiksi lainausmerkeiksi.

Sekä excel, että CSV-tiedostoon viiteavaimet tallennettiin selkokielisinä. Toisin sanoen valmistusmaan kohdalle ei tiedostoon tule viitenumeroa maat-tauluun, vaan selkeästi maan nimi, jotta tiedosto olisi selkeämpi lukea.

6.3.5 Eräajo

Tuotteita sivustolla on satoja ja niihin saattaa tulla muutoksia kausittain. Jos muokattavia tuotteita on kerralla useita, niiden yksittäinen muuttaminen on hidasta. Tämän vuoksi hallintapaneeliin haluttiin mahdollisuus tuoda uusien tai muuttuneiden tuotteiden tiedot suoraan excel-muotoisesta dokumentista.

Excelillä voidaan tallentaa taulukko määrämuotoiseksi CSV-tekstitiedostoksi, jossa sarakkeet on eroteltu määrätyllä erotinmerkillä. Tällainen tiedosto aukeaa selkeästi taulukkomuotoisena taulukkolaskentaohjelmissa, jolloin ylläpitäjä voi muokata sitä. Toisaalta se on hyvin määrämuotoinen tekstitiedosto, joka voidaan rivi kerrallaan parseroida palvelimella ja tallentaa tiedot tietokantaan.

CSV-tiedoston tuonti palvelimelle tehtiin kaksiosaisena. Kun ylläpitäjä tuo tiedoston palvelimelle tavallisella HTML:n file input-elementillä, tarkistetaan ensiksi tiedoston oikeellisuus. CSV-tiedoston lukemiseen PHP sisältää hyvän fgetcsv-funktion, jolla rivien lukeminen oli helppoa.

```
<?php
$handle = fopen($_FILES['tiedosto']['tmp_name'], 'r');
$count = 0;
while ($row = fgetcsv($handle,0,";")) {
   // rivin käsittely, $row sisältää puolipisteellä erotellut
   // tietueet taulukkona
   $count++;
}
?>
```

Rivin tiedot tarkistettiin yksitellen tavallisilla if-else-rakenteilla, ja mahdolliset virheet tallennettiin listaan, joka lopulta näytettiin käyttäjälle. Rivin tietuejärjestys tallennettiin aputaulukkoon, jossa avainsana vastasi CSV-tiedoston järjestysnumeroa.

```
$csv = array('stock_price', 'category' … );
$csv = array_flip($csv);
```

Tällöin rivin tietojen tarkistuksessa voitiin käyttää selkokielisempää nimeä, kun rivin tietyn sarakkeen tieto haettiin alla olevan esimerkin mukaisesti. Jos olisi käytetty vain numeroa, kaikki tarkistukset olisi pitänyt muuttaa järjestyksen muuttuessa.

```
if (!is_numeric(str_replace(",",".",
                            $row[$csv['stock_price']])))
   $csv_virheet .= "Tukkuhinta ei ole luku tai se puuttuu (rivi
".$count.")<br/>";
```

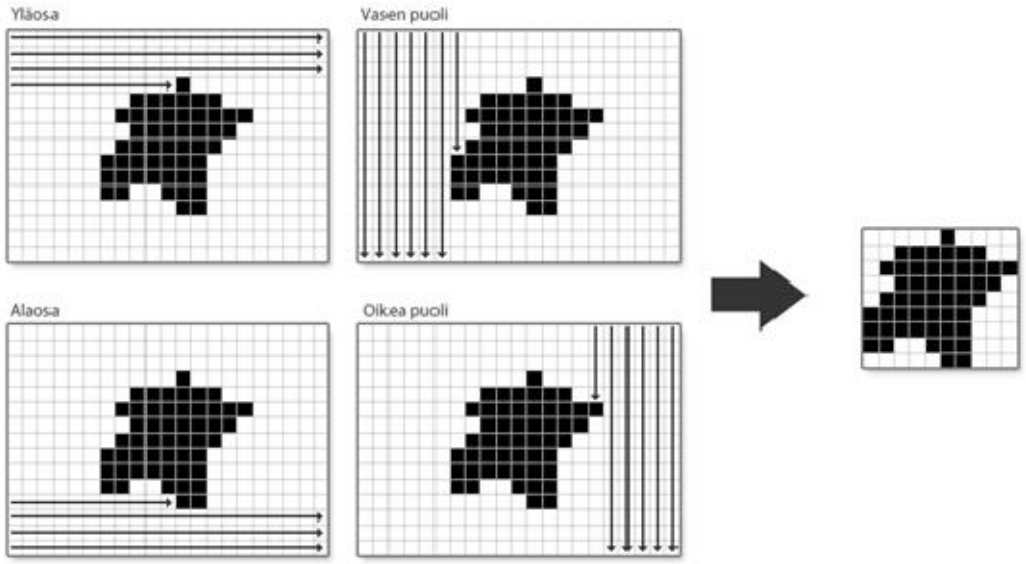
Tarkistuksen yhteydessä CSV-tiedosto tallennettiin väliaikaiseen hakemistoon, josta tietojen tallentaminen suoritettiin, jos se läpäisi virheiden tarkistuksen. Tietojen tallennus suoritettiin samanlaisella while-rakenteella ja fgetcsv-funktiolla kuin tarkistus.

6.3.6 Kuvien tuonti zip-pakettina

Samaan tapaan kuin tuotetietojen eräajo, voidaan tuotekuviakin vaihtaa useampia kerrallaan. Tämä tapahtuu lisäämällä halutut uudet kuvat zip-pakettiin, jonka voi Wordpress-hallintapaneelin kautta lähettää palvelimelle, jossa skripti avaa ja purkaa tiedostot oikeisiin kansioihin. Samalla kuvista luodaan myös pienemmät ns. thumbnails-kuvat.

PHP kykenee käsittelemään zip-tiedostoja monipuolisesti. Zip_open-funktiolla tiedosto avataan palvelimella ja sen sisältö voidaan käydä läpi while-silmukalla. Tässä tapauksessa zip-paketista etsittiin vain jpg-kuvia tarkastamalla tiedostopäätteitä: jokainen kuvatiedosto avattiin väliaikaiseen kansioon, josta siitä tehtiin eri kokoversiot. Kun oikeat kuvatiedostot on tallennettu oikeisiin sijanteihin, poistetaan väliaikainen kuva PHP:n unlink-funktiolla. Zip-paketin avaaminen PHP:llä on esitetty liitteessä 4. Erikoisuutena kuvista myös leikattiin dynaamisesti ylimääräinen valkoinen tyhjä tila itse tuotteen ympäriltä, sillä asiakkaalta tulevat tuotekuvat sisälsivät usein isot valkoiset reunat. Tähän löytyi Internetistä paljon erilaisia toteutuksia, ja vaihtoehdoista päädyttiin periaatteeseen, jossa PHP:n avulla tutkitaan pikseli kerrallaan, milloin väriarvo on jotain muuta kuin valkoinen. Tällä tavalla saadaan selville tuotteen muodostama suorakulmainen alue, jota voidaan käyttää luodessa uutta kuvaa alkuperäisestä. Imagecreatetruecolor-funktiolle voidaan antaa parametrina alkuperäinen kuva sekä rajat, joista kuvadata kopioidaan. Imagecreatetruecolor-funktio kuitenkin vaati, että palvelimelle on asennettu PHP:n GD-kirjasto. Useimmilla palvelintarjoajilla tämä on valmiiksi päällä ja niin myös Nebulalla. Kuvan ylimääräisen valkoisen alueen leikkaaminen on esitetty PHP-koodina liitteessä 5. Kuviossa 11 näkyy valkoisen alueen leikkaamisen periaate.

Kuvadata skannataan kaikista suunnista tutkimalla pikselin väriarvoa, jotta todellisen kuvan rajat saadaan selville.



KUVIO 11. Ylimääräisen tyhjän alueen poistaminen kuvasta

6.4 Nostotuotteet

Etusivun hallintaa varten tehtiin hyvin samanlainen sivu kuin tuotteillekin. Sivulta voitiin luoda ja muokata kirjautuneille käyttäjille näkyviä etusivukokonaisuuksia. Koska erilaisia etusivuja oli alun perin tarkoitus näyttää ja selata vain viikoittain, asetettiin hallintaan etusivun näkymisajankohdaksi viikkonumero päivämäärän sijaan. Tietokantaan alkamisaika merkittiin merkkijonona, jonka alkuosan muodosti vuosiluku ja toisen osan viikkonumero, eli esimerkiksi ”2011_23”. Tällöin kaikki etusivukokonaisuudet voitiin järjestää tämän sarakkeen mukaan oikeaan järjestykseen.

Myöhemmin, lähellä sivuston julkaisua, asiakas kuitenkin toivoi, että etusivut voisivat vaihtua useammin. Samalle viikolle piti siis pystyä lisäämään useita nostoja, jolloin hallintasivua jouduttiin muokkaamaan. Koko systeemin muuttaminen siten, että tietokantaan merkittäisiin aikaleima, olisi ollut isompi työ, sillä silloin olisi pitänyt muuttaa myös logiikka, jonka mukaan kirjautunut käyttäjä pystyy selaamaan viikkoja taaksepäin. Ongelma ratkesi kuitenkin melko vähällä vaivalla lisäämällä hallintaan

pudotusvalikko, josta sisällöntuottaja voi lisätä etusivun ajankohtaan myös viikonpäivän. Valittu viikonpäivä muutettiin yksinkertaisesti numeroksi, joka lisättiin alkamisajan loppuosaan, esimerkiksi ”2011_23_3”, joka tarkoittaa, että etusivu tulee voimaan keskiviikkona viikolla 23. Näin julkisella sivulla käytettyä logiikkaa ei tarvinnut juuri muuttaa ja käytännössä ainoa muutos oli päivämäärä lisääminen Edellinen- ja Seuraava-linkkeihin. Ominaisuuden muuttaminen ei siirtänyt sivuston julkaisuaikataulua.

7 KÄVIJÄSEURANTA

Markkinointisivuilla kävijänseuranta on tärkeää. Yritystä usein kiinnostaa, ketkä sivustolla käyvät ja mitä tuotteita he katsovat. Tietoa voidaan käyttää hyväksi markkinoinnissa ja sivuston parantamisessa. Lisäksi sivuston on hyvä näkyä eri hakukoneilla. Wordpress-asetuksissa on kohta, joka sallii hakukonerobottien tutkia sivustoa. Tämä on kehitysvaiheessa hyvä olla pois päältä, jotta keskeneräiselle sivustolle ei tule vierailijoita. Julkaisun yhteydessä sivuston näkyminen avattiin hakukoneille.

Koska yritysasiakkaat voivat kirjautua sivuille, ensimmäinen seuranta oli loogisinta laittaa sen yhteyteen: yritysasiakkaan kirjautuessa tallennetaan tietokannan lokiin asiakasnumeron lisäksi vain aikaleima. Tällä on hyvä tarkistaa nopeasti, ketä on kirjautunut ja milloin.

Varsinaista käyttäjäseurantaa varten sivustolle asennettiin Googlen Analytics. Se on Googlen tarjoama ilmainen selaimella toimiva työkalu, jolla kävijöiden liikkeistä sivustolla saadaan tietoa tarkempaa analyysia varten. Sen avulla nähdään, kuinka kauan kävijät pysyvät tietyillä sivuilla ja mitä tuotteita he katsovat.

Google Analayticsin asentaminen oli helppoa: sivustolle luotiin profiili Googlen Analytics palveluun, joka antoi sivuston head-elementtiin kopioitavan koodin. Tämä hoiti suurimman osan käyttäjän seurannasta, sillä se lähetti jokaisen sivunlatauksen yhteydessä tiedon Analytics-palvelimelle. Kirjautumisen yhteydessä vierailijan käyttäjätaso tallennettiin sessiomuuttujaan, kuten luvussa 4.1.2. mainittiin. Tämä tieto lisättiin Analytics-seurantaan muuttujana, jolloin voitiin verrata kirjautuneiden käyttäjien, vierailijoiden sekä englanninkielisten käyttäjien liikkeitä erikseen. Alla on esimerkki Googlen Analytics-seurannan asentamisesta ja kävijätason liittämisestä siihen.

```
<script type="text/javascript">
  var _gaq = _gaq || [];
  _gaq.push(['_setAccount', 'UA-TUNNUS']);
  <?php if ($_SESSION['userLevel'] == "loggedUser") { ?>
  _gaq.push(['_setCustomVar',1,'User Type','loggedUser',2]);
```

```
  <?php } else if ($_SESSION['userLevel'] == "en") { ?>
  _gaq.push(['_setCustomVar',1,'User Type','English', 2 ]);
  <?php } else { ?>
  _gaq.push(['_setCustomVar',1,'User Type', 'Visitor', 2 ]);
  <?php } ?>

  _gaq.push(['_trackPageview']);

  (function() {
    var ga = document.createElement('script');
    ga.type = 'text/javascript'; ga.async = true;
    ga.src = ('https:' == document.location.protocol ?
'https://ssl' : 'http://www') + '.google-analytics.com/ga.js';
    var s = document.getElementsByTagName('script')[0];
    s.parentNode.insertBefore(ga, s);
  })();
</script>
```

Tuotehaun yhteydessä ei tehdä sivunlatausta (ks. luku 4.2). Asiakas kuitenkin toivoi, että myös suoritetuista hauista saataisiin tilastoja. Googlen Analytics toimii Javascriptillä, joten sillä voitiin suorittaa seurantaa myös lennosta. Koska haku suoritetaan aina, kun käyttäjä valitsee pudotusvalikosta hakukriteerille uuden arvon, oli järkevämpää lähettää Analyticsille vain kyseisen hakukriteerin muutos. Jos jokaisesta hausta lähetettäisiin kaikki hakukriteerit, se vääristäisi tilastoja, sillä käyttäjä voi muuttaa vain yhtä kriteeriä kerrallaan. Näin aiemmat valinnat tulisivat Analyticsiin suhteessa useamman kerran. Lisäksi hakukriteerien tallentaminen helposti tilastoitavaksi olisi ollut hankalampaa.

```
jQuery('.rajaus_maa').change( function () {
    _gaq.push(['_trackPageview',
      '/search/maa/+$(this).children('option:selected').text()
    ]);
    haeTuotteet();
});
```

8 POHDINTA

Opinnäytetyössä tarkasteltiin markkinointisivuston toteutusta ilmaisella avoimen lähdekoodin julkaisujärjestelmällä. Vaihtoehtoina olivat Wordpress, Drupal ja Joomla. Opinnäytteen toimeksiantajana toimi Into-Digital Oy, ja työ pohjautui pääasiassa yhteen Into-Digital Oy:llä tehtyyn Internet-sivustoon. Sivuston nimeä ei haluttu julki, joten opinnäytteessä pyrittiin yleisempään kuvaukseen julkaisujärjestelmän käytöstä tuotetietoja sisältäville sivustoille.

Alustaksi valikoitui Wordpress, sillä se oli paljon käytössä Into-Digital Oy:n toteuttamissa sivustoissa. Olennaisin vaihtoehto olisi ollut Drupal, joka on käytössä monissa isoissa sivustoissa. Drupal oli kuitenkin melko tuntematon julkaisujärjestelmä Into-Digital Oy:llä, ja sivuston kehittäminen sillä olisi vaatinut todennäköisesti enemmän aikaa. Wordpress-julkaisujärjestelmä on osoittautunut muutamista puutteistaan huolimatta nopeasti kehitettäväksi alustaksi. Se on nopea asentaa ja laajennuksien kehittäminen on helppoa.

Sivuston tuotteet päätettiin tallentaa erillisiin tauluihin tietokannassa, ja tämä vaati erillisten työkalujen luonnin niiden ylläpitoon. Näin kuitenkin varmistuttiin siitä, että tuotetietojen relaatioista saatiin juuri halutunlaiset. Vaihtoehtona olisi ollut käyttää Wordpress-julkaisujärjestelmän omia tietotyyppejä. Tuotetietojen ylläpito oli toinen sivuston isoista ominaisuuksista. Julkaisujärjestelmän hallintapaneeliin salasanan taakse luoduilla työkaluilla ylläpitäjät pystyivät lisäämään, muokkaamaan ja poistamaan tuotteita, sekä hallitsemaan myös tuotteisiin liittyviä kategorioita ja valmistusmaita. Tuotteille tehtiin myös vienti- ja tuontitoiminnot Excel-yhteensopivalla CSV-tiedostoformaatilla. Myös tuotekuvat oli mahdollista tuoda palvelimelle yhtenä keskitettynä zip-tiedostona.

Sivuston julkisen puolen merkittävin osio oli tuotehaku, joka päätettiin tehdä dynaamisena AJAX-toteutuksena. Käyttäjä pystyi valitsemaan hakukriteerit pudotusvalikoista, valintaruuduista ja tekstihakukentistä. Kun hakukriteeri muuttui, lähetettiin valinnat Javascriptillä palvelimelle, jossa haku suoritettiin perustuen näihin kriteereihin sekä käyttäjän tasoon, sillä suurin osa tuotteista näkyi vain

kirjautuneille käyttäjille. Tuotetiedot palautettiin universaalissa JSON-muodossa, joka voitiin purkaa Javascriptillä käyttäjän selaimessa, ja näin esittää uudet tulokset ilman sivunlatausta. Sivuston julkiselle puolelle lisättiin myöhemmin myös mahdollisuus ladata tuotekortti PDF-dokumenttina.

Sivustolle asennettiin Googlen Analytics-työkalu kävijästatistiikan keräämiseen. Analyticsiin kerätään tietoja mm. suosituimmista tuotehauista, katsotuimmista tuotteista ja sivuista. Tietoa voidaan hyödyntää sivuston parantamisessa, markkinoinnissa tai esimerkiksi, kun tutkitaan, miten hyvin sivusto on vaikuttanut myyntiin.

Wordpress-julkaisujärjestelmä osoittautui hyväksi valinnaksi ja vain harva ongelma johtui julkaisujärjestelmästä. Ainoastaan Wordpressin tietokantaluokan tapa käsitellä NULL-arvoja INSERT sekä UPDATE-käskyissä aiheutti tilanteen, jossa toimintalogiikkaa ja tietokannan taulun rakennetta jouduttiin muuttamaan. Tämä on tunnettu virhe, jonka korjaantumista toivottiin projektin aikana, mutta näin ei käynyt.

Ongelmatyypeistä merkistökoodaus aiheutti eniten virheitä ennen ja jälkeen sivuston julkaisun helmi-maaliskuun vaihteessa 2011. Monissa tilanteissa merkistökoodausta piti lennosta vaihtaa UTF-8:ksi tai siitä pois PHP:n avulla, koska suomenkielen erikoismerkit näkyivät väärin. Merkistön kanssa oli ongelmia mm. sähköpostin lähetyksessä, PDF-dokumentin luonnissa ja tuotetietojen tuonnissa sekä viennissä CSV-formaatissa.

Tehty työ opetti paljon mm. tietokannan sekä AJAX:in käyttöä tuotetietojen tallentamisessa ja dynaamisen tiedon latauksessa. Sivustolle toteutettuja ominaisuuksia ja niiden rakentamisesta saatuja oppeja voidaan käyttää hyödyksi myös muilla vastaavilla sivustoilla.

LÄHTEET

About Drupal 2011. Drupal-julkaisujärjestelmän historiaa Drupalin verkkosivuilla. Viitattu 19.3.2011. http://drupal.org/about/history

About Wordpress. 2011. Julkaisujärjestelmän esittely Wordpressin omilla sivuilla. Viitattu 20.2.2011. http://wordpress.org/about/

AJAX Introduction. 2011. Artikkeli AJAX-tekniikasta w3schools.com sivustolla. Viitattu 16.2.2011. http://www.w3schools.com/ajax/ajax_intro.asp

Byron, Berry, Haug, Eaton, Walker & Robbins. 2009. Using Drupal. 2.p. O'Reilly Media, Inc.

Drupal history as seen by Dries. 2011. Drupal-julkaisujärjestelmän historiaa Drupalin verkkosivuilla. Viitattu 19.3.2011. http://drupal.org/node/297669

Introducing JSON. 2011. JSON-formaatin esittely sen omalla sivustolla. Viitattu 4.3.2011. http://www.json.org/

JavaScript Introduction. 2011. Artikkeli JavaScript-ohjelmointikielestä w3schools.com sivustolla. Viitattu 16.2.2011. http://www.w3schools.com/js/js_intro.asp

Mikä on Joomla!? 2005. Artikkeli Joomlan suomenkielisillä sivuilla. Viitattu 4.3.2011. http://www.joomlaportal.fi/content/view/93/39/

Mikä on Drupal? 2011. Drupalin suomenkieliset sivut. Viitattu 19.3.2011. http://drupal.fi/fi/drupal-suomi

PHP Licensing. 2011. PHP-ohjelmointikielen lisenssiehtojen esittely lyhyesti. Viitattu 4.3.2011. http://www.php.net/license/

Rahmel, D. 2009. Joomla! 2.p. USA.

Severdia, R. & Crowder, K. 2010. Using Joomla 2.p. O'Reilly Media, Inc.

The loop. 2011. Ohjeistus Wordpress-julkaisujärjestelmän teeman käytöstä. Viitattu 4.2.2011. http://codex.wordpress.org/The_Loop

Usage of content management systems for websites. 2011. W3Techs tarjoamia tilastoja eri julkaisujärjestelmien käytöstä. Viitattu 26.2.2011. http://w3techs.com/technologies/overview/content_management/all

Usage of server-side programming languages for websites. 2011. W3Techs tarjoamia tilastoja eri palvelinohjelmointikielien käytöstä. Viitattu 26.2.2011. http://w3techs.com/technologies/overview/programming_language/all

Using themes 2011. Artikkeli Wordpress teemojen käytöstä. Viitattu 4.2.2011. http://codex.wordpress.org/Using_Themes

W3Techs Technologies Overview. 2011. Listaus W3Techs tilastointiperiaatteista. Viitattu 26.2.2011. http://w3techs.com/technologies

What is Joomla! 2011. Joomla-julkaisujärjestelmän esittely Joomlan sivustolla. Viitattu 24.2.2011. http://www.joomla.org/about-joomla.html

Wordpress näyteikkuna. 2011. Wordpress suomen etusivu, jossa esimerkkejä suomalaisista sivustoista, jotka käyttävät Wordpress julkaisujärjestelmää. Viitattu 20.2.2011.

Wordpress showcase 2011. Lista sivustoista, jotka käyttävät Wordpress-julkaisujärjestelmää. Viitattu 20.2.2011. http://wordpress.org/showcase/http://fi.wordpress.org/

Zandstra, M. 2005. PHP 3.p. Edita Prima Oy, Helsinki.

LIITTEET

Liite 1. Oletushaun suoritus PHP:llä ja Javascriptillä

```
// function($) mahdollistaa $-merkin käytön funktion sisällä
// $-merkki on lyhenne jQuery sanalle
jQuery(document).ready(function($){
    <?php
    $sallitutMuuttujat = array('sopivuus','kategoria','maa','hinta_min','hinta_max','vapaahaku');
    if (!empty($_GET)) {
        i$hakukriteerit = "";
        foreach ($_GET as $nimi=>$arvo) {
            if ($arvo != "" && in_array($nimi, $sallitutMuuttujat)) {
                if (is_array($arvo)) {
                    foreach ($arvo as $a)
                        $hakukriteerit .= "&".$nimi."[]=".$a;
                } else {
                    $hakukriteerit .= "&".$nimi."=".$arvo;
                }
            }
        }
    ?>
    // haeTuotteet() on javascript funktio, joka ottaa parametrinä sivunumeron, ajaxilla
    // palvelimelle menevän datan sekä merkkijonon, jolla tulosten käsittelyssä
    // tunnistetaan minkälainen haku oli.
    // Sekä GET että SESSION tyyppinen haku käsitellään
    // samalla tavalla ja siksi viimeinen parametri on tässä ”sessioHaku”
    haeTuotteet(sivu, '&action=haeTuotteet_ajax<?php echo $hakukriteerit;?>&taso=<?php echo
$kayttajaTaso; ?>&order=nimi&sivu='+sivu, "sessioHaku");
    <?php
    } else if ( !empty($_SESSION['haku']) ) {
        $sessioMuuttujat = array();
        parse_str($_SESSION['haku'], $sessioMuuttujat);
    ?>
        sivu = '<?php echo (isset($sessioMuuttujat['sivu'])) ? $sessioMuuttujat['sivu'] : 1; ?>';
```

```
        haeTuotteet(sivu, '<?php echo $_SESSION['haku'];?>&taso=<?php echo $kayttajaTaso; ?>',
"sessioHaku");
    <?php
    } else {
    ?>
    haeTuotteet(sivu, '&action=haeTuotteet_ajax&uutuudet=1&taso=<?php echo $kayttajaTaso;
?>', 'vainUutuudet');
    <?php
    }
    ?>
}
```

Liite 2. Tuotehaun MySQL-kyselyn rakentaminen ja tulosten palautus

```
<?php
// esitellään ajax funktio käytettäväksi sivuston hallinnassa ja julkisella puolella
add_action('wp_ajax_nopriv_haeTuotteet_ajax', 'haeTuotteet_ajax');
add_action('wp_ajax_haeTuotteet_ajax', 'haeTuotteet_ajax');

function haeTuotteet_ajax() {
   global $wpdb;      // Käytetään wordpressin omaa wpdb-luokkaa
   $prepared = array();       // sanahaun sanat tarkistetaan injektioiden varalta
   $tuotteitaPerSivu = 30;

   // kyselyn alkuosa
   $sql = "SELECT tuote.tuoteID, tuote.nimi, maa.nimi, kategoria.nimi AS kategoria,
            FROM tuote
            LEFT JOIN kategoria
            ON tuote.kategoria = kategoria.kategoriaID
            LEFT JOIN maa
            ON tuote.maa = maa.maaID ";

   // vapaasanahaku
   $sanaEhto = "";
   if (!empty($_POST['vapaahaku']) ) {
      foreach (explode(" ", $_POST['vapaahaku']) as $hakusana) {
         if (strlen($hakusana) > 3) {
            $sanaEhto .= ($sanaEhto != "") ? " OR " :  "";
            // prosenttimerkin jälkeinen merkki tulostetaan sellaisenaan, %%->%
            $sanaEhto .= "CONCAT(tuote.nimi,' ',maa.nimi,' ',tuote.tuotenro) LIKE '%%%s%%' ";
            $prepared[] = $hakusana;
         }
      }
   }

   // AND -ehdot
   $andEhto = "";
   if (isset($_POST['uutuudet']) && @$_POST['uutuudet'] != 0) {
```

```
    $andEhto .= "tuote.uutuus = 1";
  }
  if (!empty($_POST['maa']) ) {
    $andEhto .= ($andEhto != "") ? " AND " :  "";
    $andEhto .= ”tuote.maa = ' ".$_POST['maa']." ' ";
  }

  // moni-moneen ominaisuus, esimerkiksi tuotteen soveltuvuus eri asioihin
  $sopivuusEhto ="";
  if (!empty($_POST['sopivuus']) && is_array($_POST['sopivuus'])) {
    for($i = 0; $i<count($_POST['sopivuus']); $i++) {
      $sopivuusEhto .= ($ sopivuusEhto != "") ? " OR " :  "";
      $sopivuusEhto="sopivuus.sopivuusID=" . $_POST['sopivuus'][$i] ;
    }
    $sql .= " RIGHT JOIN (
                SELECT sopivuus.sopivuusID, sopivuus.nimi,  tuote_sopivuus.tuoteID AS sopID
FROM tuote_sopivuus
                LEFT JOIN sopivuus
                ON sopivuus.sopivuusID = tuote_sopivuus.sopivuusID
                WHERE ".$sopivuusEhto.") AS sopivuus
              ON tuote.tuoteID = sopivuus.sopID";
  }

  // WHERE lauseen kokoaminen
  if ($andEhto != "" || $sanaEhto != "")
    $sql .= " WHERE ";
  $sql .= $andEhto;
  if ($sanaEhto != "")
    $sql .= ($andEhto != "") ? " AND  (".$sanaEhto.")" :  $sanaEhto;

  // GROUP BY
  $sql .= " GROUP BY tuote.tuoteID HAVING tuote.tuoteID IS NOT NULL";

  // ORDER BY tulee radiobutton valinnasta, jossa vaihtoehdot hinta ja nimi
  $orderBy = ($_POST['order'] == 'hinta' ) ? 'tuote.hinta' : 'tuote.nimi' ;
  $sql .= " ORDER BY ".$orderBy." ";
```

```
    // LIMIT sivutusta varten
    $limit = max($_POST['sivu']-1, 0)*$tuotteitaPerSivu.",".$tuotteitaPerSivu;
    $sql .= " LIMIT ".$limit;

    // kyselylause on valmis, seuraavaksi suoritetaan itse haku
    if (count($prepared)>0)
        $tuotteet = $wpdb->get_results( $wpdb->prepare( $sql, $prepared ) );
    else
        $tuotteet = $wpdb->get_results( $wpdb->prepare( $sql ) );

    // lähetetään tulokset json objektina käyttäjän selaimeen, jossa se käsitellään javascriptillä
    echo json_encode( $tuotteet );

    // wordpress vaatii, että ajax-funktiot lopetetaan die() tai exit() -komennolla,
    // muuten palautetaan -1.
    // -1 palautetaan myös silloin, kun skriptiä yritetään ajaa Wordpressin ulkopuolelta ilman
    // oikeuksia
    die();
}
?>
```

Liite 3. Ulkoisen tekstitiedoston parserointi

```
<?php
ini_set('auto_detect_line_endings', TRUE);
$file = "http://...";
$handle = fopen($file, "r");              // avataan tiedosto vain lukemista varten (”read”)
$headers = get_headers($file, 1);     // tiedoston otsikkotietoja voidaan käyttää virheiden varalta

$codesArr = array();
$rowCount = 0;
$errors = array();
if($headers[0] == "HTTP/1.1 200 OK") {
    // silmukka käy tiedoston läpi ja tallentaa rivin merkit kolmeen eri tietueeseen.
   // Tiedoston ensimmäiset 22 merkkiä sisältävät yritystietoa, seuraavat 9 varsinaisen
   // kirjautumiskoodin ja loput ylimääräistä tietoa yrityksestä
    while (!feof($handle)) {
      $codesArr[] = array('clientnro'=>fgets($handle, 22), 'code'=>fgets($handle, 9),
'companyData' => fgets($handle, 8192));
    }
    fclose($handle);
} else {
    $insertQuery = "INSERT INTO insertLog (row_amount,errors) VALUES (0,'File is empty')";
    $result = mysql_query($insertQuery) or die( "Error_code_sql_log");
    die("File is empty");
}
// Jos tekstitiedoston lukeminen onnistui, poistetaan vanhat tiedot tietokannasta (truncate) ja
// lisätään uudet. Virheet kirjataan lokiin.
if (count($codesArr) > 0) {
    $truncateQuery = "TRUNCATE loginCodes";
    $result = mysql_query($truncateQuery) or die( "Error_code_sql_truncate");
    foreach ($codesArr as $row) {
        if ($row['code'] != "") {
            $row['code'] = mysql_real_escape_string($row['code']);
            $row['clientnro'] = mysql_real_escape_string($row['clientnro']);
            $row['companyData'] = mysql_real_escape_string($row['companyData']);
```

```
            $insertQuery = "INSERT INTO loginCodes(clientnro,code,company) VALUES
('".$row['clientnro']."','".$row['code']."','".$row['companyData']."')";
            $result = mysql_query($insertQuery);
            if (!$result)
                $errors[] = "Insert failed on row ".$rowCount;
        } else {
                $errors[] = "Empty set on row ".$rowCount;
        }
        $rowCount++;
    }
    $insertQuery = "INSERT INTO insertLog (row_amount,errors,error_row_amount) VALUES
(".$rowCount.",'".implode(',', $errors)."',".count($errors).")";
    $result = mysql_query($insertQuery) or die( "Error_code_sql_log");
    die("codes successfully saved");
} else {
    $insertQuery = "INSERT INTO insertLog (row_amount,errors) VALUES (0,'No rows found')";
    $result = mysql_query($insertQuery) or die( "Error_code_sql_log");
    die("No rows");
}
?>
```

Liite 4. ZIP-tiedoston purkaminen palvelimella

```
<?php
// aluksi tarkistetaan, että tuotu tiedosto on zip-paketti
$zip_file_errors = array();
if (isset($_POST['tuoZip'])) {
    if (file_exists($_FILES['zip_tiedosto']['tmp_name'])) {
        $type = explode("/", $_FILES['zip_tiedosto']['type']);
        $ext = substr($_FILES['zip_tiedosto']['name'], strrpos($_FILES['zip_tiedosto']['name'], '.') +
1);
        if ($type[1] != "x-zip" || $ext != "zip" )
            $zip_file_errors['tiedosto'] = "Tiedosto ei ole .zip";
    } else {
        $zip_file_errors['tiedosto'] = "Tiedosto puuttuu";
    }

    if (count($zip_file_errors)<1) {
        $zip = zip_open($_FILES['zip_tiedosto']['tmp_name']);
        if ($zip) {
            $count = 0;
            // silmukka käy läpi jokaisen zip-paketin sisältämän tiedoston ja käsittelee vain jpg-kuvat
            // tallentamalla ne väliaikaiseen kansioon. Tämän jälkeen niistä luodaan eri kokoversiot
            // ja lopulta väliaikainen tiedosto poistetaan
            while ($zip_entry = zip_read($zip)) {
                if( !strstr(zip_entry_name($zip_entry), "/") ) {
                    $filename = zip_entry_name($zip_entry);
                } else {
                    $filename = substr(strrchr(zip_entry_name($zip_entry), "/"), 1 );
                }
                $ext = substr($filename, strrpos($filename, '.') + 1);
                if ($ext == "jpg") {
                    $fp = fopen("../zip-tmp/".$filename, "w");
                    if (zip_entry_open($zip, $zip_entry, "r")) {
                        $buf = zip_entry_read($zip_entry, zip_entry_filesize($zip_entry));
                        fwrite($fp,$buf);
                        zip_entry_close($zip_entry);
```

```
                    fclose($fp);

                    trimImage("../zip-tmp/".$filename,0xFFFFFF);
                    // muut kuvankäsittelyn toimenpiteet, kuten eri kokoversioiden luonti
                    unlink("../zip-tmp/".$filename);
                    $count++;
                }
            }
        }
        zip_close($zip);
        $zip_file_successMsg = '<p id="successMsg">Lisätty: '.$count.' kuvaa</p>";
    } else {
        $zip_file_errors['tiedosto'] = "Tuntematon virhe";
    }
} else {
    $zip_file_errorsMsg = '<p class="error">';
    foreach ($zip_file_errors as $v)
        $zip_file_errorsMsg .= '- '.$v.'<br/>';
    $zip_file_errorsMsg .= '</p>';
}
}
?>
```

Liite 5. Kuvan reunojen poistaminen käyttäen PHP:n GD-kirjastoa

```
<?php
function trimImage($imgOrig, $trimColor) {
    $img = imagecreatefromjpeg($imgOrig);

    // Ylälaita
    for($bd_yla = 0; $bd_yla < imagesy($img); ++$bd_yla)
        for($x = 0; $x < imagesx($img); ++$x)
            if(imagecolorat($img, $x, $bd_yla) != $trimColor)
                break 2;

    // Alalaita
    for($bd_ala = 0; $bd_ala < imagesy($img); ++$bd_ala)
        for($x = 0; $x < imagesx($img); ++$x)
            if(imagecolorat($img, $x, imagesy($img) - $bd_ala-1) != $trimColor)
                break 2;

    // Vasen laita
    for( $bd_vas = 0; $bd_vas < imagesx($img); ++$bd_vas)
        for($y = 0; $y < imagesy($img); ++$y)
            if(imagecolorat($img, $bd_vas, $y) != $trimColor)
                break 2;

    // Oikea laita
    for($bd_oik = 0; $bd_oik < imagesx($img); ++$bd_oik)
        for($y = 0; $y < imagesy($img); ++$y)
            if(imagecolorat($img, imagesx($img) - $bd_oik-1, $y) != $trimColor)
                break 2;

    // kopioidaan alkuperäisen kuvan data ilman reunoja
    $newimg = imagecreatetruecolor(imagesx($img)-($bd_vas+$bd_oik), imagesy($img)-
($bd_yla+$bd_ala));

    imagecopy($newimg, $img, 0, 0, $bd_vas, $bd_yla, imagesx($newimg), imagesy($newimg));
```

```
    // korvataan alkuperäinen kuva
    imagejpeg($newimg,$imgOrig);
    imagedestroy($newimg);
    imagedestroy($img);
}
?>
```