Kemi Tornion ammattikorkeakoulu

PHP-verkkokauppaohjelmisto MVC-arkkitehtuurilla

Tilja Niko

Tietotekniikan koulutusohjelman opinnäytetyö
Ohjelmistotekniikka
Insinööri (AMK)

KEMI 2012

TIIVISTELMÄ

KEMI-TORNION AMMATTIKORKEAKOULU

Tekniikan yksikkö

Tekijä(t):	Niko Tilja
Opinnäytetyön nimi:	PHP-verkkokauppaohjelmisto MVC-arkkitehtuurilla
Sivuja (+liitteitä):	77 + 1 liitesivu
Päiväys:	10.4.2012

Opinnäytetyön aiheeksi valittiin PHP-verkkokauppaohjelmiston suunnittelu ja toteutus, joka on tarkoitettu julkiseen levitykseen suomeksi ja englanniksi. Suurin syy verkkokauppaohjelmiston työstämiseksi oli se, että tämän opinnäytetyön kirjoitushetkellä markkinoilla ei ole tarjolla ilmaista, täysin tuettua, suomenkielistä verkkokauppaohjelmistoa.

Ilmaisia vieraskielisiä verkkokauppaohjelmistoja on suomennettu, sekä joitakin suomalaisia maksu- ja postitustapoja on niille tarjolla, mutta ne ovat täysin yhteisöpohjaisia. Suurin ongelma yhteisön tarjoamissa ilmaisissa tuotoksissa on se, että niiden laatu vaihtelee rajusti. Niiden päivityksistäkään ei ole mitään takuuta, jolloin ne eivät välttämättä tule olemaan yhteensopivia verkkokauppaohjelmiston uudemman version kanssa.

Opinnäytetyössä suunniteltiin ja ohjelmointiin verkkokauppaohjelmiston sovelluskehys, jonka tärkeimmiksi piirteiksi asetettiin nopeus, tietoturva, sekä laajennettavuus - valmiin sovelluskehyksen päälle ohjelmointiin tyypillisimmät verkkokaupan ominaisuudet. Opinnäytetyöraportissa käsiteltiin kuitenkin enemmän valittuja tekniikoita, niiden tietoturvaa ja sitä, kuinka ohjelmiston arkkitehtuuri toimii yleisesti.

Sovelluskehyksen suunnittelu ja toteutus onnistui hyvin, minkä ansiosta ohjelmaa on helppo laajentaa erillisillä lisäosilla. Turvautumalla MVC-arkkitehtuuriin ja Dependency Injection suunnittelumalliin ohjelmiston sovelluskehys toimii luotettavasti ja takaa, että suurin osa kehittäjistä pystyy laajentamaan ohjelmistoa ongelmitta - käyttämällä tunnettuja suunnittelumalleja, ohjelmiston toiminnan ymmärtäminen on myös erittäin helppoa.

Asiasanat: PHP, SQL, JavaScript, tietoturva, sovelluskehykset.

ABSTRACT

KEMI-TORNIO UNIVERSITY OF APPLIED SCIENCES

Technology

Name:	Niko Tilja
Title:	PHP Shopping Cart Software Designed with MVC Architecture
Pages (+ appendixes):	77 + 1 appendix
Date:	10 April 2012

The objective of this study was to design and code a PHP shopping cart software, which is intended for public distribution in Finnish and in English. Designing this shopping cart software was due to the fact that, during the writing of this paper, there are no free, fully maintained, Finnish shopping cart software in the market.

Free foreign shopping cart software have been translated into Finnish, and some Finnish payment and shipping extensions are freely available for them but they are completely community based. The biggest problem with the free community based products is the fact that their quality varies drastically. There is also no guarantee that they will be updated, in which case they are not necessarily compatible with the later versions of the shopping cart software.

In this study, the framework of the shopping cart software was designed and coded, on which a big emphasis was placed on speed, security, and expandability – the most typical shopping cart features were coded on top of the finished framework. The thesis report, however, deals more with the selected technologies, their security, and how the program's architecture works in general.

The planning and implementation of the framework was a success, which allows the program to be easily expanded with different kinds of extensions. By resorting to the MVC architecture and Dependency Injection design pattern, the framework works reliably and guarantees that the majority of the developers can expand the program without any problems – by using known design patterns, understanding how the program works is also very easy.

Keywords: php, sql, javascript, security, frameworks.

SISÄLLYSLUETTELO

KÄYTETYT MERKIT JA LYHENTEET

AJAX	Asynchronous JavaScript and XML, joukko web-tekniikoita.
ASP.NET	Active Server Pages, web-sovelluskehys.
CMS	Content management system, sisällönhallintajärjestelmä.
CSS	Cascading Style Sheets, HTML:lle kehitetty tyyliohjeiden laji.
CSRF	Cross site request forgery, hyökkäys, joka käyttää uhrin istuntoa hyväksi.
DI	Dependency Injection, eli olio-ohjelmointi suunnittelumalli.
EMS	Express Mail Service, kansainvälinen pikapaketti.
HTML	Hypertext Markup Language, nettisivujen kuvauskieli.
J2EE	Java 2 Platform, Enterprise Edition, ohjelmistokehitysalusta Java-sovellusten kehitykseen.
MD5	Tiivistealgoritmi.
MVC	Model-View-Controller, joka tarkoittaa malli-näkymä-ohjain ohjelmistoarkkitehtuuria.
PCI DSS	Payment Card Industry Data Security Standard. Standardi maksukorttien käsittelyä, tallentamista ja välittämistä varten.
PHP	PHP: Hypertext Preprocessor, ohjelmointikieli.
SHA-512	Tiivistealgoritmi.
SQL	Structured Query Language, relaatiotietokantojen kyselykieli.
XML	Extensible Markup Language, merkintäkieli.
XSS	Cross site scripting, hyökkäys, jossa haitallista koodia syötetään nettisivulle.

1. JOHDANTO

Opinnäytetyön aiheeksi valittiin PHP-verkkokauppaohjelmiston suunnittelu ja toteutus, joka on tarkoitettu julkiseen levitykseen suomeksi ja englanniksi. Suurin syy verkkokauppaohjelmiston työstämiseksi oli se, että tämän opinnäytetyön kirjoitushetkellä ei ole tarjolla ilmaista, täysin tuettua, suomenkielistä verkkokauppaohjelmistoa, joka tarjoaisi vielä mukaan tyypillisimmät suomalaiset maksu- ja postitustavat.

Ilmaisia vieraskielisiä verkkokauppaohjelmistoja on suomennettu, mutta ne ovat täysin kyseisen yhteisön vapaaehtoisia tuotoksia, jolloin suomennoksen laatu voi vaihdella huomattavasti. Yhteisön tarjoamat suomalaiset maksu- ja postitustavat ovat monesti myös puutteellisia ja lisäosien yleinen tarjonta vaihtelee rajusti. Suurin ongelma kuitenkin yhteisön tarjoamissa ilmaisissa suomennoksissa ja lisäosissa on se, että niiden uusista julkaisuista ei ole mitään takuuta, jolloin käyttäjät voivat pahimmassa tapauksessa joutua käyttämään verkkokauppaohjelmiston vanhempaa versiota – vanhemmissa versiossa voi olla tyypillisesti jonkin asteisia tietoturva-ongelmia.

Opinnäytetyössä aluksi suunniteltiin ja toteutettiin verkkokauppaohjelmiston sovelluskehys, jonka tärkeimmiksi piirteiksi asetettiin nopeus, tietoturva, sekä laajennettavuus - valmiin sovelluskehyksen päälle ohjelmointiin tyypillisimmät verkkokaupan ominaisuudet. Opinnäytetyöraportissa käsitellään kuitenkin enemmän valittuja tekniikoita, niiden tietoturvaa ja sitä, kuinka ohjelmiston arkkitehtuuri toimii yleisesti.

2. KÄYTETYT TEKNIIKAT

Tässä osiossa käsitellään verkkokauppaohjelmistossa käytettyjä tekniikoita.

2.1. PHP

PHP on palvelinpuolen ohjelmointikieli, jota käytetään dynaamisten nettisivujen luomiseen. PHP:n syntaksi muistuttaa huomattavasti C- ja Perl-ohjelmointikieliä. Vaikka PHP on saanut kyseisistä ohjelmointikielistä vaikutteita, niin se on muokattu eroamaan niistä omalla tavalla, sillä PHP:n päätarkoitus on olla helppo työkalu erilaisien ongelmien ratkaisemiseksi. (Rasmus Lerdorf | PHP on Hormones 2007, hakupäivä 11.2.2012) PHP:n lyhenne tuli alun perin sanoista ”Personal Home Page”, mutta on myöhemmin, useamman muutoksen jälkeen, muutettu rekursiiviseksi akronyymiksi, joka tulee sanoista ”PHP: Hypertext Preprocessor” (History of PHP, hakupäivä 11.2.2012).

PHP oli yksi ensimmäisistä ohjelmointikielistä, jonka pystyi sulauttamaan suoraan HTML-merkintäkieleen, sen sijaan, että koodissa kutsuttaisiin ulkoista HTML-tiedostoa tai tulostettaisiin HTML:ää erillisillä käskyillä. Esimerkiksi Perl-ohjelmointikielessä HTML joudutaan tulostamaan joko rivi riviltä tai pienissä here-document lohkoissa. (History of PHP, hakupäivä 11.2.2012) PHP-koodi voidaan ajaa sellaisenaan tai sulauttaa suoraan muun sisällön joukkoon, esimerkiksi HTML-merkintäkieleen, kunhan koodi erotetaan muusta sisällöstä <?php ja ?> merkkien sisälle, jolloin PHP:n tulkitsin osaa ajaa koodin (Escaping from HTML, hakupäivä 11.2.2012).

PHP-koodi tulkitaan ja ajetaan palvelimen puolella ja lopputulos lähetetään eteenpäin, tyypillisesti nettiselaimelle, jolloin käyttäjä ei tiedä mitä koodia palvelimen puolella on ajettu (What is PHP?, hakupäivä 11.2.2012). PHP:ssä koodi tulkitaan lennosta, jolloin sitä ei tarvitse kääntää binääriksi ennen ajamista, kuten esimerkiksi C-ohjelmointikielessä - tämän ansiosta PHP:tä on helppo testata kehitys- ja testausvaiheessa, sillä päivitetyn koodin voi testata suoraan ilman ylimääräisiä

välivaiheita. PHP-koodin tulkattavuuden ansiosta PHP on tarjolla useille eri alustoille ja käyttöjärjestelmille. (What can PHP do?, hakupäivä 11.2.2012)

2.1.1. Historia

Rasmus Lerdorf alkoi kehitellä PHP:n ensimmäistä versiota vuonna 1994 työnsä ohella ja julkaisi ensimmäisen version kaikkien ladattavaksi vuonna 1995. Sillä hetkellä suurin osa koodista oli kirjoitettu Toronton yliopistolle, mutta Lerdorf sai yliopistolta luvan julkaista työkalut avoimena lähdekoodina, sillä heitä vain kiinnostivat niiden tuottamat hyödyt. (Rasmus Lerdorf | PHP on Hormones 2007, hakupäivä 11.2.2012)

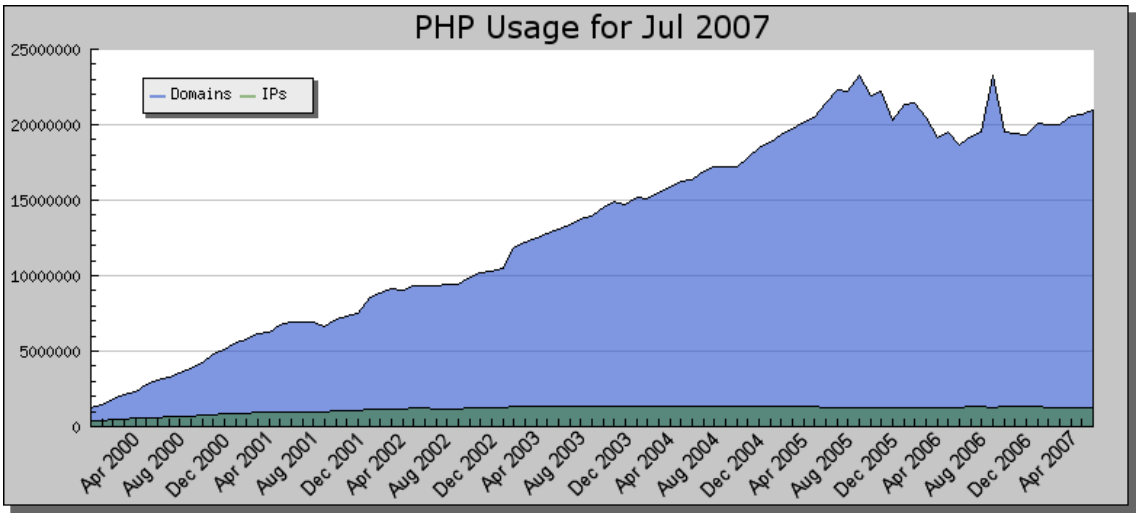
Useamman uudelleen kirjoituksen jälkeen PHP/FI 2 julkaistiin vuonna 1997 ja PHP 3 vuotta myöhemmin. PHP:n kolmas versio sisälsi helppokäyttöisemmän ja laajemman rajapinnan dynaamisten nettisivujen luomiseen kuin PHP/FI 2, joka nosti PHP:n suosion uudelle tasolle. PHP 4 julkaistiin vuonna 2000 ja se sisälsi ison määrän uusia ominaisuuksia, muun muassa tuen istunnonhallintaan. Sen ydin perustui uuteen Zend Engine -tulkkiin, jonka keskeisin ominaisuus oli sen laajennettavuus, joka antoi mahdollisuuden erilaisille lisäosille, esimerkiksi ohjelmointivirheiden jäljittämiseen, suorituskyvyn parantamiseen, sekä PHP:n laajentamiseen sisäänrakennetuilla funktioilla (PHP and Zend, hakupäivä 11.2.2012). (History of PHP, hakupäivä 11.2.2012)

PHP 5 julkaistiin vuonna 2004, minkä myötä Zend Engine päivittyi versioon 2, joka oli osittain uudelleen kirjoitettu (History of PHP, hakupäivä 11.2.2012). Uuden version suurimpia uudistuksia olivat virheiden hallinta, poikkeukset, XML-työkalut, sekä uusiksi kirjoitetut olio-ohjelmointi ja MySQL -tuet (Trachtenberg 2004, hakupäivä 11.2.2012).

2.1.2. Miksi PHP?

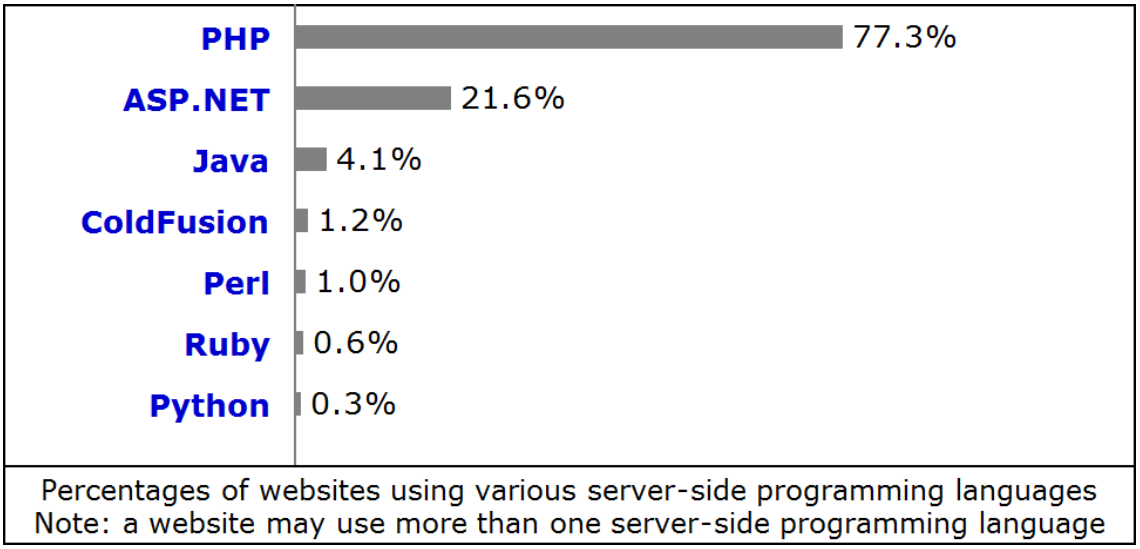
PHP on sen ensimmäisestä versiosta saakka suunniteltu web-sovelluksien tekemiseen, oli kyseessä sitten yksinkertainen lomakkeiden käsittelijä tai suuremman luokan web-sovellus. Juuri PHP:n joustavuus ja helppous ovat tehneet siitä yhden suosituimmista ohjelmointikielistä. Asiaan on myös vaikuttanut se, että PHP on ilmainen

avoimenlähdekoodin ohjelmisto, joka toimii melkein kaikissa palvelinohjelmistoissa ja käyttöjärjestelmissä, toisin kuin ASP.net, PHP:n suurin kilpailija, joka toimii virallisesti vain Microsoft Windows käyttöjärjestelmissä (ASP.NET Platform Requirements, hakupäivä 12.2.2012; The PHP License, version 3.01, hakupäivä 12.2.2012). (Baker, Reierøl & Shiftlett 2007, 4-7). Alla olevasta kuvasta näkyy, että PHP löytyi jo yli 20 miljoonasta verkkosoitteesta ja yli miljoonasta IP-osoitteesta vuonna 2005.



Kuva 1. PHP:n käyttötilastot 2000 - 2007 (PHP Usage Stats 2007, hakupäivä 12.2.2012)

PHP on tällä hetkellä ylivoimaisesti suosituin palvelinpuolen ohjelmointikieli, jolla on yli 77 % markkinaosuus, kuten kuvasta 2 huomataan. Suuren käyttäjäkunnan takia moniin ongelmiin löytyy helposti ratkaisu käyttämällä hakukoneita (Baker ym. 2007, 5-6). PHP:n suosio näkyy myös esimerkiksi Dev Shed ohjelmointi keskustelupalstassa, josta löytyy yli puolimiljoonaa viestiä liittyen pelkästään PHP:hen, joka on noin 21 % koko sivuston keskustelusta (Dev Shed Forums, hakupäivä 12.2.2012).



Kuva 2. Palvelinpuolen ohjelmointikielien käyttötilasto 2012 (Usage of server-side programming languages for websites, hakupäivä 4.3.2012)

Suuren käyttäjäkunnan ansiosta PHP:lle tehdään ylivoimaisesti eniten ilmaisia ja maksullisia web-sovelluksia (Scripts, hakupäivä 12.2.2012). Suuren web-sovellusmäärän ansiosta PHP:tä tukeville webhotelleille on luonnollisesti suuri kysyntä, jonka myötä hinnat ovat matalat suuren kilpailun vuoksi, mikä puolestaan takaa PHP web-sovelluksille enemmän käyttäjiä. Esimerkiksi Wordpress PHP web-sovellusta on ladattu yli 10 miljoonaa kertaa ja lähes 70 miljoonaa nettisivua toimii Wordpress:illä (WordPress Download Counter, hakupäivä 12.2.2012; Stats – Wordpress.com, hakupäivä 12.2.2012). (Gilmore 2010, 37-39; Baker ym. 2007, 4-7)

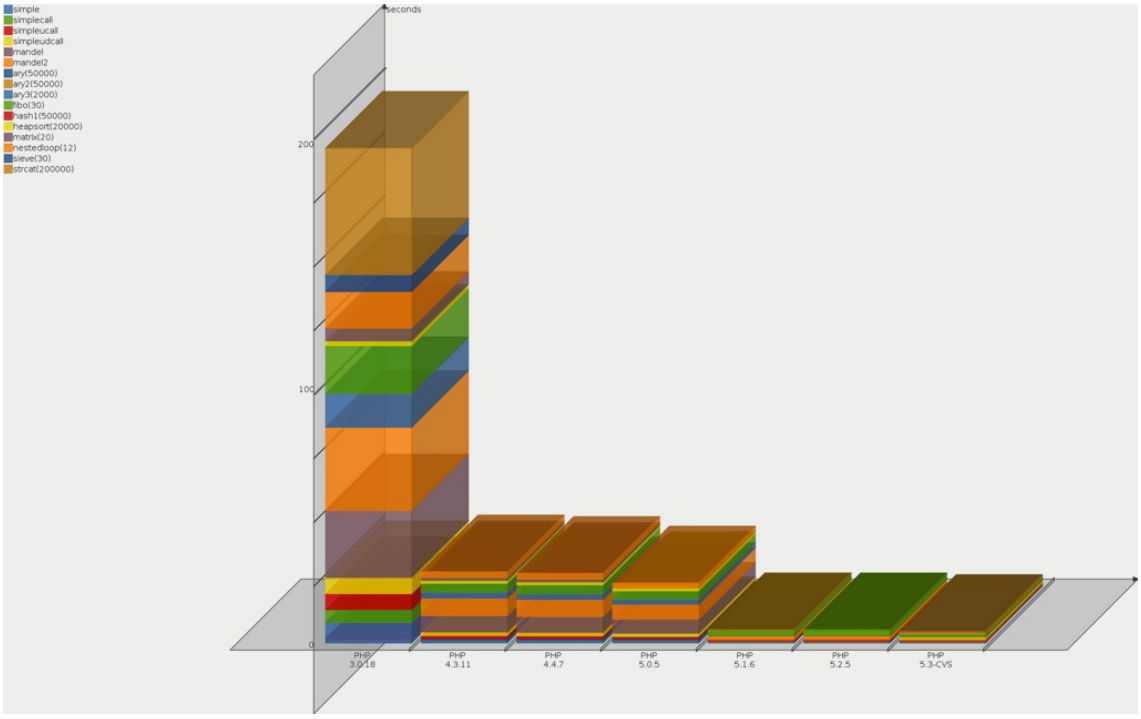
PHP:n joustavuus, suuri käyttäjäkunta ja PHP:tä tukevien webhotellien matalat hinnat olivat juurikin suurimmat syyt, miksi PHP valittiin verkkokauppaohjelmiston ohjelmointikieleksi.

2.1.3. PHP 5.3

Verkkokauppaohjelmisto tulee julkaisuvaiheessa vaatimaan PHP:n version 5.3 tai uudemman. PHP 5.3:sen yksi tärkeimmistä uudistuksista on nimiavaruudet (namespaces), joilla voidaan välttää tilanteet, jossa kaksi tai useampi luokka sisältää saman nimen. Sen sijaan, että luokkien nimiä yritetään pitää välttämättä erilaisina, nimiavaruuksien avulla kaikki luokat voidaan nimetä loogisesti ilman pelkoa siitä, että

ohjelma katuu nimien törmäyksien yhteydessä – useampia luokkia voi myös niputtaa yhteen eri nimiavaruuksiin. Nimiavaruuksilla voi myös PHP 5.3:ssa lyhentää luokkien nimiä, jotka ovat mahdollisesti liian pitkiä, jolloin niiden käyttäminen helpottuu huomattavasti, esimerkiksi jos luokkia joutuu käyttämään useasti. (Namespaces overview, hakupäivä 25.2.2012)

Uuden version myötä PHP 5.3 tuo mukanaan parannuksia, jotka nopeuttavat PHP:tä keskimäärin 10 – 30 %, riippuen sovelluksesta (Stogov 2008, hakupäivä 25.2.2012). PHP:n roskienkerääjä (garbage collector) on myös saanut uudistuksia, jolloin muistinkulutus ja mahdolliset muistivuoto -tilanteet ovat vähentyneet merkittävästi. Synteettisissä testeissä PHP 5.2 roskienkerääjä aiheutti 931 Mt:n muistivuodon, toisaalta PHP 5.3 kulutti samassa testissä vain 10 Mt muistia (Performance Considerations, hakupäivä 25.2.2012). Kuvassa 3 on esillä nopeuserot alkaen PHP 3:sta PHP 5.3 CV:een, mistä huomataan PHP 5.3:sen tuoman nopeusedun edeltäjäänsä.



Kuva 3. PHP 3 – PHP 5.3 CV nopeuserot (Bergmann 2008, hakupäivä 26.2.2012)

Uudemman version ainoa haittapuoli on sen tämän hetkinen kohtalainen saatavuus. PHP 5.3 julkaistiin virallisesti kesäkuussa 2009 ja PHP 5.2 tuki loppui joulukuussa 2010, mutta silti webhotellien siirtyminen uusimpaan PHP versioon on ollut hidasta (News

Archive - 2010, hakupäivä 25.2.2012; PHP 5 ChangeLog, hakupäivä 25.2.2012). Esimerkiksi Go Daddy, joka on yksi maailman suurimmista webhotelleja tarjoavista yrityksistä, päivitti palvelunsa tukemaan PHP 5.3:sta vasta marraskuussa 2011 (The Go Daddy Group, hakupäivä 25.2.2012; Godaddy.com Site Info, hakupäivä 25.2.2012; PHP 5.3 or bust…, hakupäivä 25.2.2012). Suurin syy siihen, miksi yritykset välttävät päivitystä PHP 5.3 versioon, on käyttäjien käyttämät web-sovellukset, jotka mahdollisesti rikkoutuisivat päivityksen myötä (Backward Incompatible Changes, hakupäivä 25.2.2012; Deprecated features in PHP 5.3.x, hakupäivä 25.2.2012). Tilanne kuitenkin paranee koko ajan, joten verkkokauppaohjelmiston julkaisuvaiheessa PHP 5.3 ei tule olemaan ongelma.

2.2. MySQL

MySQL on avoimenlähdekoodin relaatiotietokantaohjelmisto, joka toimii yli 10:ssä eri käyttöjärjestelmässä (Operating Systems Supported by MySQL Community Server, hakupäivä 18.2.2012). MySQL:än arkkitehtuuri on erilainen verrattuna muihin tietokantaohjelmiin, mutta se juuri tekee siitä samalla erittäin joustavan. Toisin kuin monet kilpailijat, MySQL tukee useita eri tallennusmoottoreita (storage engine), jotka ovat vastuussa tiedon tallennuksesta ja noudosta. Jokainen tallennusmoottori tukee erilaisia ominaisuuksia ja toimintoja, joten niillä kaikilla on omat hyvät ja huonot puolensa. (Balling, Lentz, Schwartz, Tkachenko, Zaitsev & Zawodny 2009, 1-2)

Vaikka MySQL:än arkkitehtuuri ei ole täydellinen, niin se soveltuu erinomaisesti erittäin vaativiin ympäristöihin, kuten web-sovelluksiin. MySQL-tietokannassa voidaan valita tarpeiden mukaan jokaiselle taulukolle oma tallennusmoottori, esimerkiksi istunnot voidaan tallentaa nopeaan muistipohjaiseen taulukkoon ja samalla kriittiset tiedot tallennetaan taulukkoon, jonka tallennusmoottori takaa tiedon eheyden (Chapter 13. Storage Engines, hakupäivä 18.2.2012). (Balling ym. 2009, 1-2)

2.2.1. Historia

MySQL:än ensimmäinen julkinen versio, versio 3.19, julkaistiin vuonna 1996 ruotsalaisen TcX DataKonsult AB:n toimesta. Uusia versioita julkaistiin sopivin väliajoin, mitkä toivat mukana uusia ominaisuuksia. Vuoteen 2001 mennessä MySQL oli kasvanut yhdeksi suosituimmaksi relaatiotietokantaohjelmistoksi, minkä johdosta TcX DataKonsult AB perusti MySQL AB nimisen yrityksen, joka alkoi tarjota pelkästään MySQL:ään liittyviä palveluita. Versio 4.0 julkaistiin keväällä 2003 ja sisälsi toivotun query cache -ominaisuuden, joka tallentaa suosituimmat haut ja niiden tulokset muistiin, jolloin MySQL:n nopeus parani vähintään 238 % (7.5.3. The MySQL Query Cache, hakupäivä 19.2.2012). (Gilmore 2010, 477)

Versio 4.1 julkaistiin vuonna 2004, minkä myötä MySQL nopeutui entisestään. Yksi uusimmista ominaisuuksista oli esivalmistetut lausunnot (prepared statements), joilla pystyi nopeuttamaan esimerkiksi hakuja, joita toistetaan useampaan kertaan (717.6.4. C API Prepared Statements, hakupäivä 19.2.2012). MySQL 5.0 julkaistiin vuotta myöhemmin, mikä oli noin 33 % nopeampi kuin edeltäjänsä (A Look at MySQL 5.0 Performance Benchmarks 2006, hakupäivä 19.2.2012). Versio 5.1 julkaistiin vasta vuonna 2008, mutta sitä vaivasi useat kaatumisongelmat, jotka saatiin korjattua vasta useamman kuukauden jälkeen (Widenus 2008, hakupäivä 19.2.2012).

Suuren suosion ja voittojen vuoksi Sun Microsystems osti MySQL AB:n miljardilla dollarilla vuonna 2008 (Sun to Acquire MySQL 2008, hakupäivä 19.2.2012). Oracle Corporation puolestaan osti Sun Microsystemsin vuonna 2009, mikä nykyään omistaa oikeudet MySQL:ään. (Gilmore 2010, 477)

2.2.2. Miksi MySQL?

Yksi tärkeimmistä syistä, miksi MySQL valittiin verkkokauppaohjelmiston tietokannaksi, oli sen suuri suosio. MySQL on tällä hetkellä maailman suosituin avoimenlähdekoodin relaatiotietokantaohjelmisto, jota on ladattu yli 100 miljoonaa kertaa (MySQL Market Share, hakupäivä 18.2.2012). Suuren käyttäjäkunnan takia MySQL löytyy melkein kaikista PHP:tä tukevista webhotelleista, joka takaa, että

verkkokauppaohjelmiston käyttöönotto ei tule olemaan ongelma. PostgreSQL, joka on MySQL:än suurin kilpailija, ei taas löydy jokaisesta webhotellista. Esimerkiksi DreamHost ja HostGator, jotka ovat maailman suurimpia webhotelleja, eivät tue PostgreSQL:ää heidän halvimmissa palveluissa (PostgreSQL « HostGator.com Support Portal, hakupäivä 19.2.2012; PostgreSQL - DreamHost, hakupäivä 19.2.2012; Hostgator.com Site Info, hakupäivä 19.2.2012; Dreamhost.com Site Info, hakupäivä 19.2.2012). (Gilmore 2010, 477-480)

MySQL ei sisällä yhtä paljon ominaisuuksia kuin PostgreSQL, mutta se on ensimmäisestä julkaisusta saakka suunniteltu nopeaksi ja skaalautuvaksi, jotka ovat erittäin tärkeitä kriteerejä verkkokauppasovelluksille, sillä niiden pitää olla nopeita (Gilmore 2010, 477-480). MySQL oli selvästi nopeampi kuin PostgreSQL muutama vuosi sitten, esimerkiksi sovellustestissä MySQL latasi Drupal 5.x etusivun noin 2 kertaa nopeammin kuin PostgreSQL (Benchmarking PostgreSQL vs. MySQL performance using Drupal 5.x 2007, hakupäivä 19.2.2012). Uusimmissa synteettisissä testeissä PostgreSQL on melkein yhtä nopea luku- ja kirjoitustesteissä kuin MySQL, mutta pelkissä lukutesteissä se häviää selvästi MySQL:lle (MySQL Performance: MySQL-5.4.0 and other InnoDB engines @dbSTRESS Benchmark 2009, hakupäivä 19.2.2012).

2.2.3. InnoDB

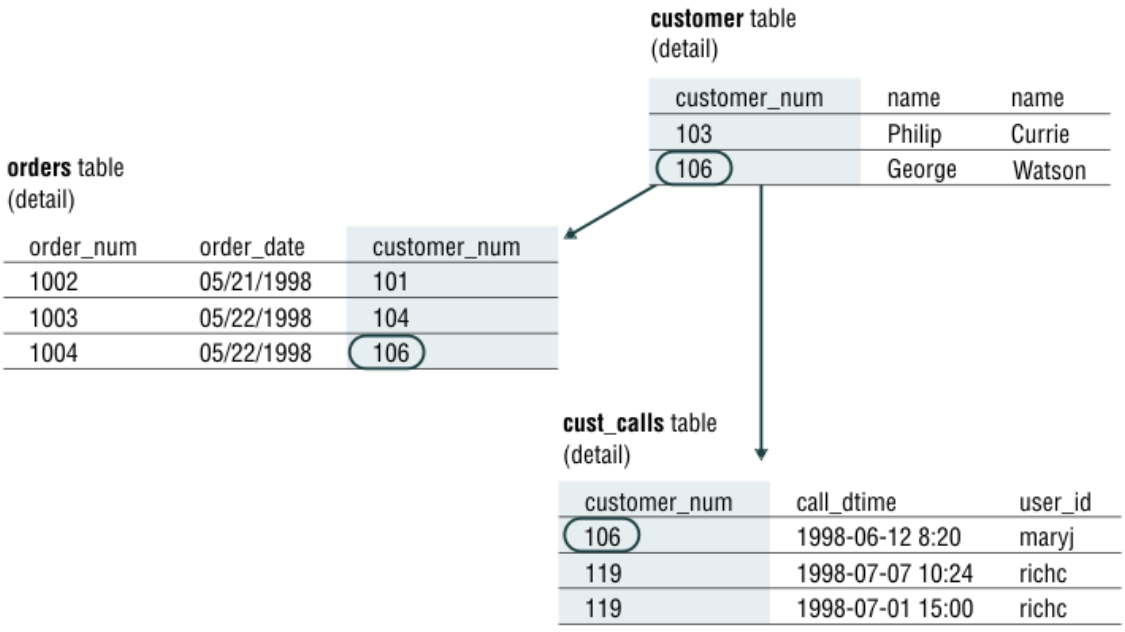
Kuten luvun 2.2 alussa mainittiin, MySQL:n arkkitehtuuri antaa mahdollisuuden valita jokaiselle taulukolle, vaatimuksien mukaan, erilaisen tallennusmoottorin. MySQL:n suosituimmat tallennusmoottorit ovat MyISAM ja InnoDB, joista InnoDB valittiin verkkokauppaohjelmiston taulukoille. (Chapter 13. Storage Engines, hakupäivä 18.2.2012).

MyISAM perustuu vanhempaan ISAM tallennusmoottorin, joka takaa, että MyISAM on tyypillisesti nopeampi taulukoiden lukemisessa kuin InnoDB, sillä se on huomattavasti yksinkertaisempi. MyISAM on myös tuetumpi, sillä se on ollut MySQL:n oletustallennusmoottori versioon 5.5.5 saakka (The MyISAM Storage Engine,

hakupäivä 19.2.2012). InnoDB tarjoaa kuitenkin huomattavia etuja, jotka ovat tärkeitä verkkokauppaohjelmistoille. (Balling ym. 2009, 2-7)

InnoDB tukee transaktioita (transactions), joka tarkoittaa sitä, että useampaa SQL-käskyä voidaan pitää yhtenä tapahtumana. Transaktiossa kaikki muutokset lisätään tietokantaan tai niitä ei lisätä ollenkaan, jolloin virhetilanteessa tietokantaan ei tule keskeneräisiä muutoksia – kyseessä on ”kaikki tai ei mitään” -periaate. Transaktiota tarvitaan tilanteissa, joissa tietoa ei voida käsitellä yhdellä kerralla, vaan tietoa pitää lukea ja kirjoittaa erikseen useampaan otteeseen, mutta samalla tiedon eheys pitää taata, jos tapahtuma keskeytyy. (Balling ym. 2009, 3-7)

InnoDB tukee myös foreign key – ominaisuutta, jolla voi yhdistää eri taulukoita yhteen, eli taata niiden viite-eheys (referential integrity). Taulukoiden foreign key:llä voidaan myös muodostaa toimiva hierarkia taulukoille, eli jos yhdestä taulukosta muokataan tai poistetaan rivi, muutos tapahtuu myös muissa yhdistetyissä taulukoissa. Esimerkiksi kuvassa 4 tilaustaulukko ja soittotaulukko on yhdistetty asiakastaulukkoon asettamalla foreign key:t viittamaan asiakastaulukon asiakasnumeroon. Jos esimerkiksi asiakas George Watson poistetaan asiakastaulukosta, myös siihen yhdistyneet taulukot poistavat asiakkaaseen liittyvät tiedot. Tällä tavoin taataan yhdenmukaisuus, eli jokaisella tilauksella ja kirjatulla soitolla pitää olla asiakas. (Balling ym. 2009, 252-253) (IBM Informix Dynamic Server v10 Information Center, hakupäivä 20.2.2012)



Kuva 4. Viite-eheys taulukoiden välillä (IBM Informix Dynamic Server v10 Information Center, hakupäivä 20.2.2012)

Normaalisti, esimerkkiä vastaavassa tilanteessa, ohjelmisto joutuisi manuaalisesti etsimään muista taulukoista asiakkaaseen liittyvät tiedot ja poistamaan ne. Foreign key:tä käyttämällä välivaihde poistuu ja samalla virheiden määrä laskee, sillä tapahtumat on automatisoitu. (Balling ym. 2009, 252-253)

Yksi tärkeimmistä syistä, miksi InnoDB on nykyään MySQL:n oletustallennusmoottori ja valittiin myös verkkokauppaohjelmistolle, on sen mahdollisuus toipua yllättävistä palvelimen kaatumisista ongelmitta. MyISAM tallennusmoottori voi korruptoida taulukoita tilanteissa, missä palvelin kaatuu, mutta MyISAM ei pysty havaitsemaan niitä - virheetkään ei välttämättä näy heti käyttäjälle, jolloin ohjelmisto voi toimia virhetilassa pitkään aikaan. (Balling ym. 2009, 19; MySQL 5.5: Storage Engine Performance Benchmark for MyISAM and InnoDB, hakupäivä 20.2.2012; 13.6.6.1. The InnoDB Recovery Process, hakupäivä 20.2.2012)

2.3. JavaScript

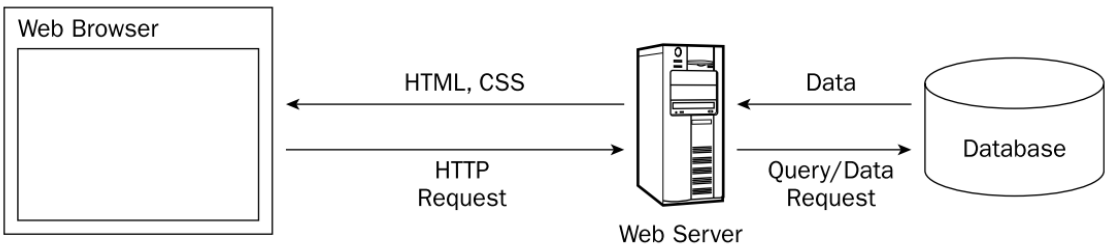
JavaScript on dynaaminen ohjelmointikieli, jota käytetään tyypillisesti nettisivuilla, jolloin niistä voidaan tehdä enemmän vuorovaikutteisempia. Koska JavaScript on dynaaminen ohjelmointikieli, sitä pystytään tulkkaamaan myös muissakin alustoissa ja ohjelmissa, joten JavaScriptiä löytyy myös esimerkiksi PDF-tiedostoista, sekä työpöytä-sovelluksista. (Flanagan 2011, 1-2)

JavaScript julkaistiin ensimmäisen kerran 1995 loppupuolella Netscape Navigator nettiselaimen mukana, mutta se oli silloin nimetty LiveScriptiksi. LiveScript nimettiin uudelleen nopeasti JavaScriptiksi, sillä Netscape oli tehnyt yhteistyösopimuksen Sun Microsystemsin kanssa, joka omisti Javan. Vaikka Javalla ja JavaScriptillä ei ole paljoakaan yhteistä, LiveScript nimettiin JavaScriptiksi, koska Java oli noussut suureen suosioon ja Netscape halusi markkinoida JavaScriptiä uutena ohjelmointikielenä. (Programming languages on the internet, hakupäivä 28.2.2012; Flanagan 2011, 2-3)

JavaScript on vuosien varrella saanut huomattavia uudistuksia ja sen suosio on ollut nopeassa kasvussa 2000-luvun alusta lähtien. JavaScript on saanut paljon vaikutteita C- ja Java-ohjelmointikieliltä, jolloin sen ohjelmointi on myös helpompaa, sillä sen syntaksi ei ole liian omaperäinen. Nykyään suurin osa moderneista nettisivuista käyttää JavaScriptiä sivuston parantamiseksi, sekä huomattava osa vaatii JavaScript tuen nettiselaimelta, sillä muuten kyseiset nettisivut eivät toimi. Vaikka JavaScriptiä pidetään nykyään itsestäänselvyytenä, niin silti keskimäärin 1 – 2 % netin käyttäjistä ei pidä JavaScriptiä päällä tai heidän nettiselain ei tue sitä (Zakas 2010, hakupäivä 28.2.2012). (Flanagan 2011, 1-3)

2.3.1. Ajax

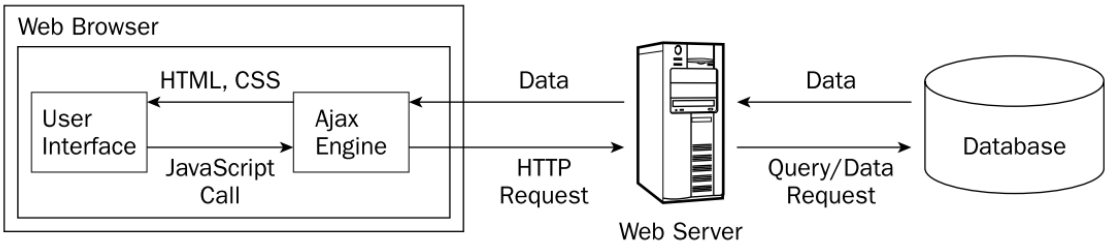
Ajax (Asynchronous JavaScript and XML) on joukko sovellustekniikoita, joiden avulla yritetään tehdä staattisesta HTTP-protokollasta vuorovaikutteisempi. Normaalisti, jos käyttäjä on vuorovaikutuksessa nettisivun sovelluksen kanssa, uuden tiedon esittäminen vaatii koko sivuston uudelleen lataamisen. (Garrett 2005, hakupäivä 28.2.2012) Kuva 5 havainnollistaa tilannetta, eli jokaista tapahtumaa kohden käyttäjä joutuu odottamaan, että sivusto latautuu uudestaan.



Kuva 5. Perinteinen web-sovellus (Fawcett, McPeak & Zakas 2007, 5)

Ajaxissa lähetetään ja vastaanotetaan tietoa palvelimelta asynkronisesti ilman, että se häiritsisi käyttäjää, sillä kaikki tapahtuu taustalla. Tämän avulla voidaan päivittää osa nettisivun sisällöstä ilman, että koko sivu ladattaisiin tai päivitettäisiin uudelleen, kun käyttäjä tekee erilaisia pyyntöjä. Ajaxin avulla nettisivujen toiminnot pääsevät lähemmäs tavanomaisia tietokoneohjelmia, mikä luonnollisesti parantaa nettisivujen käytettävyyttä käyttäjän kannalta. Ajax web-sovellukset ovat periaatteessa omia kokonaisuuksia, jotka pyörivät käyttäjän nettiselaimessa, kuten kuvasta 6 huomataan.

(Ajax: A New Approach to Web Applications, hakupäivä 28.2.2012; Fawcett ym. 2007, 5)



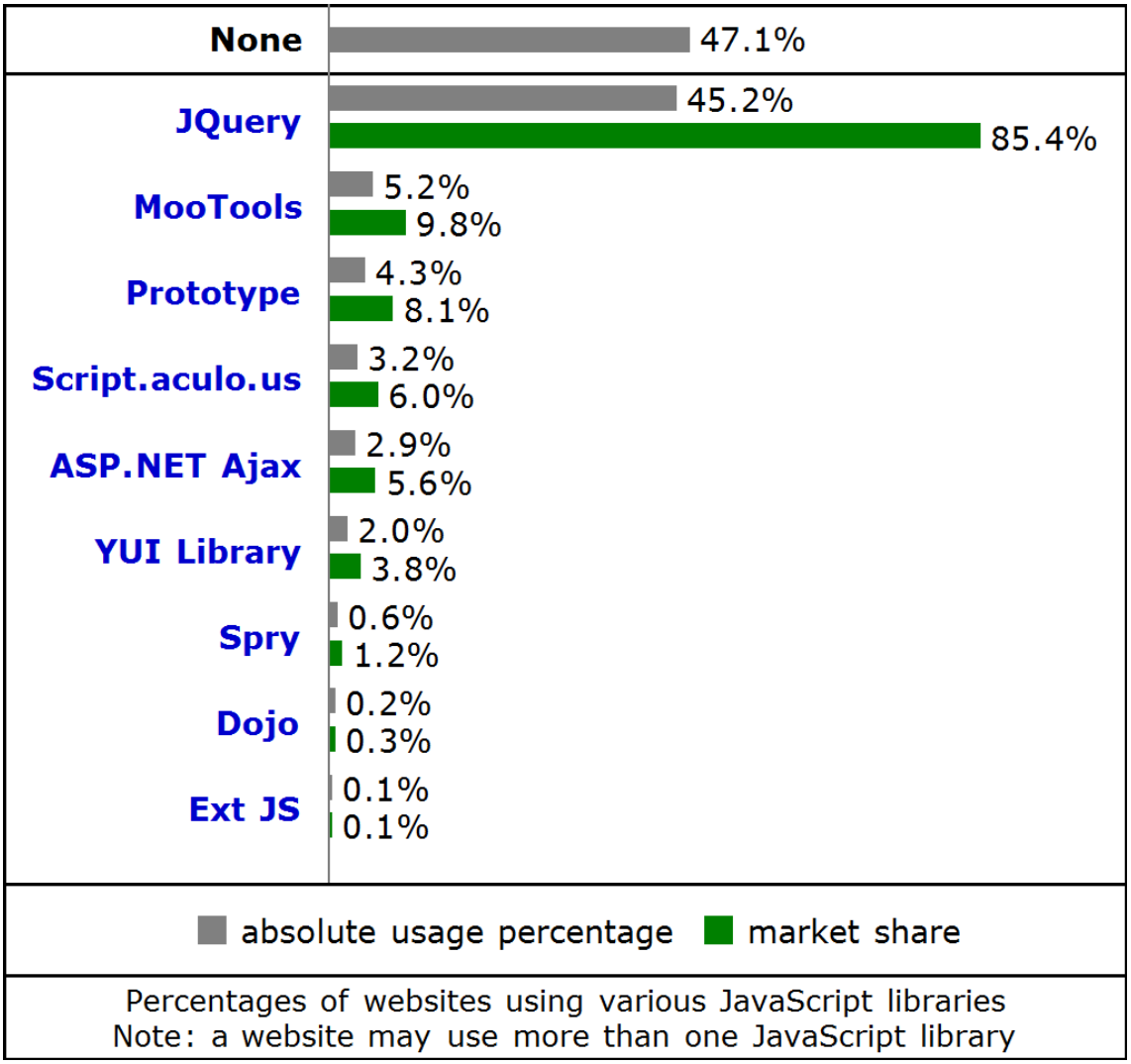
Kuva 6. Ajax web-sovellus (Fawcett ym. 2007, 5)

2.3.2. jQuery-kirjasto

jQuery on ilmainen avoimenlähdekoodin JavaScript-kirjasto, jonka tarkoituksena on yksinkertaistaa HTML dokumenttien, CSS:än (Cascading Style Sheets) ja tapahtumien käsittelyä, sekä yleensä helpottaa JavaScriptin käyttöä. jQuery tarjoaa helpon, mutta kehittyneen rajapinnan, jolloin websovelluksien kehittäjien ei tarvitse tuhlata aikaa monimutkaisimpien JavaScript-funktioiden koodaamiseen. Sen sijaan kehittäjille jää enemmän aikaa keskittyä olennaiseen, eli HTML:än ja CSS:än käsittelyyn. (jQuery: The Write Less, Do More, JavaScript Library, hakupäivä 29.2.2012; Bibeault & Katz 2008, 2-3)

Web-sovelluksien kehittäjät hyötyvät myös jQuery-kirjaston laajasta yhteensopivuudesta, sillä se toimii melkein kaikissa 2000-luvun nettiselaimissa. Normaalisti eri nettiselaimien JavaScript-tulkeissa on eroja, mikä aiheuttaa erilaisia yhteensopivuus ongelmia, mutta jQuery takaa, että kaikki sen ominaisuudet toimivat samalla tavalla kaikissa tuetuissa nettiselaimissa. (Bibeault ym. 2008, 2-3; Flanagan 2010, 1-4; jQuery: The Write Less, Do More, JavaScript Library, hakupäivä 29.2.2012)

Alla esitetystä kuvasta huomataan, että jQuery on tällä hetkellä ylivoimaisesti suosituin JavaScript-kirjasto, joten sen valitseminen verkkokauppaohjelmistolle on itsestäänselvyys, mutta luonnollisesti myös korkea yhteensopivuus eri nettiselaimien kanssa vaikutti asiaan. Tällä hetkellä yli 24 miljoonaa nettisivua käyttää jQuery-kirjastoa hyödyksi nettisivuillaan (jQuery Usage Statistics, hakupäivä 4.3.2012).



Kuva 7. JavaScript-kirjastojen käyttötilastot (Usage of JavaScript libraries for websites, hakupäivä 4.3.2012)

3. YLEISET TIETOTURVAUHAT

Palvelinpuolen ohjelmointikielien ansiosta dynaamiset nettisivut ja web-sovellukset ovat arkea jokaiselle Internetin käyttäjälle, oli kyseessä sitten blogit, pankkien palvelut tai sosiaaliset mediat. Internetin suuren käyttäjäkunnan takia joukosta löytyy myös henkilöitä, jotka haluavat huvin vuoksi tai muuten vain mielenkiinnosta kokeilla nettisivujen ja web-sovelluksien tietoturvarajoja. Joukkoon pesiytyy myös ääripään rikollisia, jotka haluat hyötyä ohjelmistojen tietoturva-aukoista rahallisesti.

Vaikka PHP, sekä muut ohjelmointikielet, ovat alustana turvallisia, se ei kuitenkaan tarkoita, että ohjelmistot olisivat turvassa hyökkäyksiltä. Tyypillisin syy nykypäivänä tietoturva-aukon syntymiseen on liiallinen luottamus käyttäjien lähettämään tietoon, sekä yleinen huolimattomuus. Ohjelmistot tarvitsevat tietoa käyttäjiltä pystyäkseen näyttämään tietoa takaisin, mutta joissakin tapauksissa käyttäjän syötettä ei tarkisteta, mikä johtaa siihen, että ulkopuolinen voi päästä tekemään jotakin sellaista, mitä hänen ei normaalisti pitäisi päästä tekemään. (Myer, Snyder & Southwell 2010, 3-12)

PHP on turvallinen, mutta sillä on silti ylivoimaisesti eniten web-sovelluksia, joihin löytyy vakavia tietoturva-aukkoja. Esimerkiksi Exploit Database sivustosta löytyy melkein 10000 tietoturva-aukko ilmoitusta pelkästään suosituimmista PHP web-sovelluksista (Search « Exploits Database by Offensive Security, hakupäivä 3.4.2012). Syy ei ole kuitenkaan itse PHP:ssä, vaan enemmänkin kehittäjissä, jotka eivät ota syötteiden tarkistusta huomioon, sekä yleensä kirjoittavat epäturvallista koodia. (Shiflett 2006, vii-viii)

Nykyään melkein jokaiselle palvelinpuolen ohjelmointikielelle löytyy tietoa, kuinka yleisimpiä tietoturva-aukkoja voidaan estää, mutta silti pieniä että isoja nettisivuja ja palveluita hakkeroidaan vähän väliä (Martin 2011, hakupäivä 4.3.2012). Monien tapauksien kohdalla ongelmaa pahentaa se, että hakkereihin ei ollut varauduttu, joten esimerkiksi käyttäjien salasanoja ei suojattu millään tavalla, vaan ne olivat tietokannassa pelkkänä tekstinä. Suurin osa ihmisistä käyttää samaa salasanaa useisiin palveluihin, joten niiden vuotaminen rikollisien käsiin voi tulla kalliiksi – sekä

yrityksille, että käyttäjille (Password Security 2011, hakupäivä 3.4.2012). (Myer ym. 2010, 8-12)

Alla olevissa alaotsikossa käydään läpi yleisimmät uhat, jotka tyypillinen web-sovellus voi kohdata.

3.1. SQL-injektio

SQL-injektiolla (SQL injection) tarkoitetaan, että tietokannan SQL-lauseeseen (SQL query) lisätään käyttäjältä tietoa, esimerkiksi tuotteen etsimiseen, mutta käyttäjän syötettä ei tarkisteta, minkä kautta käyttäjä voi muokkaa SQL-lauseen muotoa. Koska käyttäjä voi vaikuttaa SQL-lauseeseen, yksinkertainenkin SQL-lause voi muuttua erittäin haitalliseksi ilman syötteiden tarkistuksia, esimerkiksi käyttäjä voi tyhjentää koko tietokannan tai tarkastella arkaluontoisia tietoja. SQL-injektiot ovat tällä hetkellä ylivoimaisesti suurin tietoturva-aukko web-sovelluksissa (OWASP Top 10 Most Critical Web Application Security Risks 2010, hakupäivä 4.3.2012). (Myer ym. 2010, 33)

Normaali SQL-lause voi näyttää esimerkiksi seuraavalta:

```
SELECT * FROM tuotteet
```

Kyseinen SQL-lause hakee kaikki tuotteet tietokannan tuotetaulukosta ja tässä tapauksessa ulkopuolinen käyttäjä ei pysty vaikuttamaan hakuun. Jos tarvitaan vain tiettyjä tuotteita tietokannasta, esimerkiksi Microsoftin valmistamia, SQL-lause voidaan muokata seuraavaksi:

```
SELECT * FROM tuotteet WHERE tuoteen_valmistaja = ’Microsoft’
```

Jos halutaan, että web-sovelluksen käyttäjä voi etsiä esimerkiksi tuotteita valmistajan mukaan, mikä on erittäin tyypillistä, lisätään käyttäjän syötteen SQL-lauseeseen:

```
SELECT * FROM tuotteet WHERE tuoteen_valmistaja = ’KÄYTTÄJÄN_SYÖTE’
```

Ongelmaksi edellisessä esimerkissä muodostuu se, että jos syötettä ei millään tavalla tarkisteta, käyttäjä voisi hakea esimerkiksi seuraavaa valmistajaa: Microsoft'; DROP TABLE tuotteet; --, joka muodostaisi SQL-lauseen seuraavaksi:

SELECT * FROM tuotteet WHERE tuoteen_valmistaja = 'Microsoft'; DROP TABLE tuotteet; --'

Käyttäjän vaikutuksesta SQL-lause muodostui nyt yhden käskyn sijaan kahdesta eri käskystä. Tietokanta tässä tapauksessa hakee ensiksi kaikki Microsoftin tuotteet, mutta tämän jälkeen, koska SQL-lausetta muokattiin, tietokantaa käsketään poistamaan tuotteet-taulukko tietokannasta.

Tyypillisin tapa estää SQL-injektiot esimerkiksi PHP:ssä on käyttää mysql_real_escape_string-funktiota, joka automaattisesti lisää koodinvaihtomerkin, eli kenoviivan, seuraaviin merkkeihin: \x00, \n, \r, \, ', " ja \x1a. Tämän avulla käyttäjän syöte ei voi vaikuttaa SQL-lauseen rakenteeseen, koska kenoviivan ansiosta tietokanta ymmärtää, että ', " ja ; -merkit ovat syötteitä, jotka eivät ole osa SQL-lauseen rakennetta. (Myer ym. 2010, 37-39)

Vaikka syötteet varmistetaan turvallisiksi, esimerkiksi mysql_real_escape_string -funktiolla, niin se ei tarkoita, että ohjelmisto olisi vielä turvassa. Esimerkiksi jos käyttäjän syöte lisätään tietokantaan, niin se voi avata tietoturva-aukon toisen asteen SQL-injektiolle. Kyseisessä hyökkäyksessä tietokantaan syötetään haitallista tietoa, joka sitten myöhemmässä vaiheessa noudetaan tietokannasta ja käytetään SQL-lauseessa, mitä kautta toisen asteen SQL-injektio tapahtuisi. Kyseinen SQL-injektio on harvinaisempi, mutta silti mahdollista, sillä tietokannassa oleva tieto luokitellaan yleensä turvalliseksi, mitä myöten siihen ei monesti kohdisteta tarkistuksia. (Mackay 2005, hakupäivä 4.3.2012)

Toisen asteen hyökkäys voisi tapahtua esimerkiksi verkkokaupassa, joka antaa käyttäjän tallentaa suosikki tuotteiden nimiä omalle tilille, mitkä ohjelmisto sitten käy läpi kerran päivässä ja ilmoittelee saatavuudesta sähköpostilla. Käyttäjän suosikki tuotteen nimi voisi olla esimerkiksi '; DROP TABLE tilaukset --, mutta koska hyökkäykseen on

varauduttu, se ei vaikuta syötteen tallennus hetkellä. Myöhemmässä vaiheessa, kun ohjelmisto käy läpi asiakkaiden suosikki tuotteiden nimiä tietokannasta, tapahtuu toisen asteen SQL-injektio. Tässä esimerkissä tilaukset-taulukko poistettaisiin tietokannasta kokonaan, koska tuotteiden nimiä ei tarkistettu huolimattomuuden takia.

3.2. Tiedostot

PHP:ssä tiedostojen lataaminen on tehty erittäin helpoksi. Tätä kautta voi myös syntyä tietoturva-aukkoja, jos esimerkiksi käyttäjän syötettä käytetään tarkistamatta tiedostojen lataamisessa. Tyypillisin tietoturva-aukko löytyy esimerkiksi ulkoasuun liittyvissä tietojenkäsittelyssä, jossa ladataan tietty tyyli sivustolle osoiterivin avulla:

```
include("/tiedostot/{$_GET['tyyli']}.php");
```

Ongelmana tässä on, että hyökkääjä voisi muokata tiedosto pyyntöä, jolloin pahimmassa tapauksessa arkaluontoista tietoa voitaisiin tulostaa sivulle. Edellisen esimerkin tiedoston latausta voitaisiin muuttaa helposti:

www.esimerkki.com/index.php?tyyli=TEKSTI_TÄHÄN

Osoiteriviä muuttamalla esimerkiksi ”index.php?tyyli=../asetukset” muotoon kertoisi PHP:lle ladata asetukset.php -tiedoston yhtä kansiota alempaa, mikä tarkoittaisi samaa kuin:

```
include("asetukset.php");
```

Koska PHP käyttää pohjimmiltaan C-ohjelmointikielen tasoisia funktioita tiedostojen käsittelyssä, se lisää tietoturvariskejä joissain tapauksissa. Esimerkiksi null-merkki C-ohjelmointikielessä kertoo merkkijonossa, että teksti loppuu siihen, mutta PHP:ssä sillä ei ole mitään vaikutusta merkkijonossa. Jos null-merkki löytyy PHP:n merkkijonosta, kun tiedostoa ladataan, niin tiedoston nimi katkaistaan null-merkkiin konekielen tasolla, vaikka näin ei pitäisi normaalisti PHP.ssä tapahtua. (Shiflett 2006, 54) Käyttämällä tätä tietoa hyväksi, hyökkääjä voi kutsua melkein mitä tahansa tiedostoa, jos syötettä ei tarkisteta:

www.esimerkki.com/index.php?tyyli=../etc/passwd%00

Yllä olevassa esimerkissä include-funktion sisältö muuttui seuraavaksi:

```
include("/etc/passwd");
```

Vaikka alkuperäisessä include-funktiossa hyväksyttiin vain .php -loppuisia tiedostoja, niin null-merkki poisti sen, kun tiedostoa käsiteltiin. Pyyntö oli aluksi ”/tiedostot/../etc/passwd%00.php” -tiedostolle, joka muuttui ”/etc/passwd%00.php” pyynnöksi ”../” ansiosta ja lopuksi ”/etc/passwd”, koska merkkijonosta löytyi null-merkki, joka poisti ”.php” osuuden.

3.3. Cross site scripting

Cross site scripting (XSS) on yksi vanhimmista ja tunnetuimmista web-pohjaisista hyökkäyksistä. Toisin kuin SQL-injektiossa, XSS-hyökkäyksissä yleensä syötetään haitallista JavaScript-koodia nettisivustolle. Ideana hyökkäyksessä on käyttää hyväksi nettiselaimen ja käyttäjän luottamusta sivustoon, minkä kautta haitallisella koodilla voidaan esimerkiksi ohjata nettiselainta, varastaa evästeitä, muokata sivuston linkkejä ohjaamaan muualle, sekä pahimmassa tapauksessa huijataan käyttäjää luovuttamaan henkilökohtaisia tietoja, jotka lähetetään hyökkääjän palvelimelle. (Myer ym. 2010, 45-47)

XSS on ongelma kaikille web-sovelluksille, eikä kohdistu pelkästään PHP web-sovelluksiin. Periaatteessa kaikki ohjelmat, jotka tulostavat käyttäjien syötteitä, ovat riskissä hyökkäykselle, oli kyseessä blogi, keskustelupalsta tai vaikkapa web-sähköposti. Hyökkäykset voivat tapahtua joko väliaikaisesti osoiterivin kautta tai suoraan sivustolla, missä tieto on tallennettu pysyvästi. (Shiflett 2006, 23-24)

Osoiterivinkautta tehdyllä XSS-hyökkäyksellä JavaScript-koodi on lisätty hakuparametreihin, jotka tulostetaan sivustossa. Tyypillinen esimerkki voisi olla sivuston hakukone, joka näyttää käyttäjän syöttämän hakusanan sivustossa. Jos sivusto

ei tarkista syötettä, XSS-hyökkäys on mahdollista. Tämän tyyppinen XSS-hyökäys on helpompi havaita, koska sivustoon lisättävä JavaScript-koodi on sivuston osoitteessa esillä. (Myer ym. 2010, 46-47) Esimerkiksi seuraavaan osoitteeseen on lisätty JavaScript alert-ikkuna, jossa lukee ”XSS-hyökkäys”:

www.esimerkki.com/?hae=%3Cscript%3Ealert%28%27XSS-hy%C3%B6kk%C3%A4ys%27%29%3B%3C%2Fscript%3E

Itse sivustoihin tallennetut hyökkäykset, esimerkiksi uutisien kommentteihin, ovat vakavampia, sillä ne eivät yleensä näy käyttäjälle. Koska käyttäjä monesti luottaa sivustoon, pienet virheet, jotka mahdollisesti jäävät XSS-hyökkäyksestä sivustolle, eivät herätä epäilyksiä. (Myer ym. 2010, 46-47)

3.4. Istunnon kaappaus

Normaalisti HTTP-protokollassa tietoa voidaan siirtää yhdestä pyynnöstä toiseen, mutta pidemmän ajan tapahtumia ei voida varastoida. Tätä varten kehiteltiin evästeet (cookies), joihin palvelin voi tallentaa tietoa, jotka puolestaan käyttäjän nettiselain tallentaa tietokoneelle ja palauttaa jokaisen pyynnön yhteydessä. Evästeiden avulla voidaan luoda yksilöllinen istunto jokaiselle käyttäjälle, mitä kautta palvelin pystyy tunnistamaan käyttäjän. (Myer ym. 2010, 93-94)

Esimerkiksi PHP:ssä sisäänrakennettu istunnonhallinta asettaa käyttäjälle PHPSESSID nimisen evästeen, jonka arvo on satunnainen 32 merkkiä pitkä teksti. Kun käyttäjän pyyntö tulee sisään sivustolle, jolla on istunto, palvelin tunnistaa käyttäjän evästeen avulla – jos käyttäjällä ei ole aiempaa istuntoa, sellainen luodaan. Samalla kun käyttäjä on tunnistettu, yleensä kiintolevyltä ladataan PHPSESSID:tä vastaava istunto web-sovellukselle, mikä sisältää esimerkiksi asiakkaan ostoskorissa olevat tuotteet. (Shiflett 2006, 40-50; Myer ym. 2010, 93-94)

Koska istuntojen tunnisteet ovat tallessa evästeissä, ne on mahdollista varastaa, minkä avulla hyökkääjä voi esimerkiksi esittää toista käyttäjää. Edellisessä kappaleessa mainittu XSS-hyökkäys on tyypillisin tapa varastaa käyttäjien istuntoja, sillä

JavaScriptillä voi pyytä käyttäjän nettiselaimelta evästeitä. Isoimmissa verkoissa tai langattomissa verkoissa, joita ei ole suojattu, on mahdollista ”salakuunnella” käyttäjien verkkoliikennettä ja etsiä esimerkiksi PHPSESSID nimisiä evästeitä, joilla voisi päästä toisen henkilön web-sähköpostiin käsiksi. (Shiflett 2006, 40-50)

3.5. Cross site request forgery

Cross-site request forgery (CSRF) on hyökkäys, jossa hyökkääjä lähettää HTTP-pyynnön uhrin nettiselaimella, joten CSRF-hyökkäystä on melkein mahdoton erottaa normaalista pyynnöstä. Hyökkäyksen voi suorittaa monella tavalla, mutta tyypillisintä on käyttää HTML:än kuva-elementtiä, jolloin käyttäjä ei huomaa mitään. CSRF-hyökkäyksessä oletetaan monesti, että uhrilla on istunto, jonka avulla pyyntö voidaan sitten suorittaa kohde sivustossa – myös kohteena olevassa sivustossa pitää olla jonkin tyyppinen haavoittuvuus, jota voidaan CSRF-hyökkäyksellä käyttää hyväksi. (Shiflett 2006, 24-28) Esimerkiksi CSRF-hyökkäys voisi tapahtua seuraavalla HTML kuva-elementillä:

<img src=”http://www.esimerkki.com/lisää-ostoskoriin.php?tuote=50”>

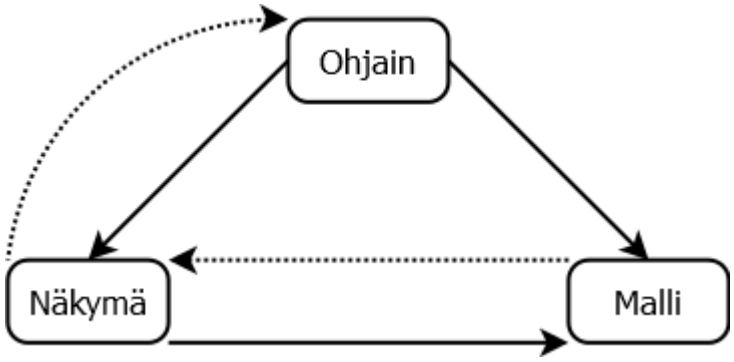
Jos uhri tulee sivulle, nettiselain lataa kuvan osoitteen, joten hänelle tässä tapauksessa lisätään esimerkki.com verkkokaupassa ostokoriin tuote numero 50. Hyökkäys voisi olla myös pahempi, esimerkiksi jos kohde sivusto antaa lisätä uusia ylläpitäjiä osoiterivin kautta:

www.esimerkki.com/ylläpitö/lisää.php?nimi=hyökkääjä&salasana=jotain

Jos jokin ylläpitäjistä olisi kirjautunut sisään, hyökkääjän tarvitsisi vain houkutella uhri sivustolle, jossa yllä oleva osoite ladataan. Tässä tapauksessa uusi ylläpitäjä nimeltä ”hyökkääjä” luotaisiin järjestelmään, koska web-sovellus luuli, että pyyntö tuli oikealta ylläpitäjältä. CSRF-hyökkäykset ovat myös pahentuneet vuosien varrella JavaScriptin ansiosta, koska sillä voidaan luoda erittäin monimutkaisia pyyntöjä, mutta uhri ei tiedä tapahtumista mitään (Myer ym. 2010, 49-50).

4. MVC-ARKKITEHTUURI

MVC-arkkitehtuurilla (model–view–controller) tarkoitetaan malli-näkymä-ohjain suunnittelumallia. Suunnittelumallin ideana on erottaa sovellus kolmeen loogiseen osaan, kuten kuvassa 8 on esitetty, minkä ansiosta suunnittelu ja ylläpito yksinkertaistuvat. Malli ja näkymä eivät ole riippuvaisia toisistaan, sama koskee myös näkymää ja ohjainta. (Baker ym. 2007, 340)



Kuva 8. MVC suunnittelumalli

Malli osuus on vastuussa business logiikasta (business logic / domain logic), eli on yhteydessä tietokantaan, joka sisältää kaiken tiedon. Malli kapseloi tiedonhaun ja -tallentamisen uudelleenkäytettäviksi luokiksi – ideana tässä on koodin uudelleenkäytettävyys. (McArthur 2008, 201)

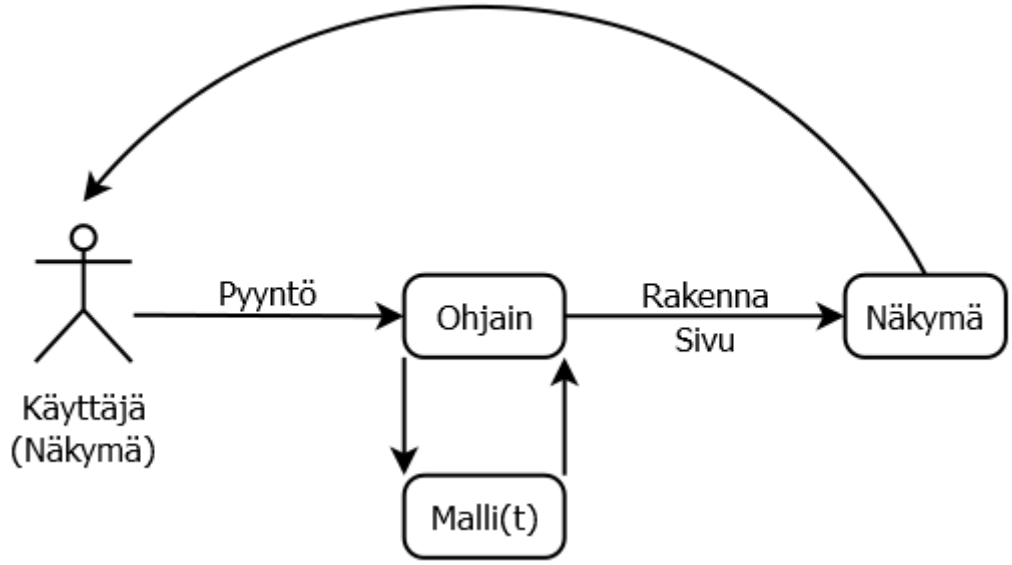
Näkymä edustaa nimensä mukaan ohjelmiston ulkoasua, oli kyseessä sitten työpöytäsovellus tai web-sovellus. Näkymä saa tarvittavat tiedot ohjaimelta, joka on puolestaan hakenut tarvittavat tiedot mallilta. Näkymässä ei yleensä ole paljon logiikkaa, sillä sen tarkoitus on vain esittää tietoa käyttäjälle. (McArthur 2008, 201)

Ohjaimen tarkoitus on liimata kaikki asiat yhteen ja olla ohjelmiston logiikka. Ohjain kerää tietoa käyttäjältä, sekä hakee lisätietoa mallista ja joissain tapauksissa myös tallentaa tietoa sinne. Kun tarvittava tieto on prosessoitu, se lähetetään näkymälle, joka näyttää uuden tiedon käyttäjälle. (McArthur 2008, 201)

Erotuksen myötä osien vaihdettavuus kasvaa, eli pitämällä osat erillään, niitä voidaan tarpeiden mukaan vaihtaa. Esimerkiksi jos web-sovellus halutaan toimivan pelkästään

mobiililaitteiden kanssa, niin ainoastaan näkymä komponentti täytyy muuttaa sellaiseksi, joka soveltuu mobiililaitteille. Koodin uudelleenkäyttö tulee myös nopeasti esille, sillä malleja ei tarvitse yleensä muuttaa, vaikka ohjain tai näkymä vaihtuisi. Mallit sisältävät yleensä suurimman osan tiedosta, jolloin niitä käytetään uudelleen ja uudelleen. Ohjelmistoa on myös helpompi ylläpitää, sillä kaikki asiat ovat pienemmissä osissa, jotka kommunikoivat toisiensa kanssa – toisin kuin perinteisemmissä ohjelmissa, joissa on muutamia isoja tiedostoja täynnä koodia. (Lecky-Thompson, Myer & Nowicki 2009, 306-307)

Web-sovelluksissa MVC-arkkitehtuuri toimii osittain eri tavalla kuin työpöytäsovelluksissa ja on sitä myöten yksinkertaisempi, kuten alla olevassa kuvassa esitetään. Mallit ovat luokkia, joiden eri metodeja kutsutaan, kun tietoa halutaan hakea tai tallentaa – malleja käytetään myös monesti tärkeiden tietojen käsittelyyn. Toisin kuin työpöytäsovelluksissa, web-sovelluksissa näkymää esittää HTML-dokumentti, joka palautetaan käyttäjälle. Ohjain taas ottaa käyttäjän GET- ja POST-pyynnöt vastaan, käsittelee ne, kutsuu mallien metodeja tarvittaessa, lähettää tiedot näkymälle ja lopuksi lähettää viimeistellyn HTML-dokumentin käyttäjälle. (Lecky-Thompson ym. 2009, 308-309)



Kuva 9. Tyypillinen web-pohjainen MVC-arkkitehtuuri

5. DEPENDENCY INJECTION

Dependency Injection (DI) on olio-ohjelmointi suunnittelumalli, jonka avulla yritetään vähentää luokkien riippuvuutta toisistaan. Tyypillisesti luokka itse päättää mistä luokista se on riippuvainen, mutta DI:ssä luokan konstruktori (constructor) kertoo mistä luokista se on riippuvainen, mitä kautta sille annetaan valitut oliot. (Prasanna 2009, 4-5) Alla olevassa esimerkissä on tyypillinen luokka, joka tarvitsee tietokanta-luokan, joten se luodaan nykyisen luokan metodin sisällä:

```
class Esimerkki
{
    public function haeMetodi()
    {
        $tietokanta = new MySQL();

        // käytetään tietokanta-luokkaa
    }
}
```

Jos halutaan käyttää MySQL:n sijaan esimerkiksi PostgreSQL:ää, kaikki luokat joudutaan muokkaamaan, missä käytetään nykyistä tietokanta-luokkaa, kuten äskeisessä esimerkissä. DI:n ansiosta sellaiset tapaukset voidaan välttää, kuten alla olevassa esimerkissä:

```
class Esimerkki
{
    private $tietokanta;

    public function __construct(Tietokanta $tietokanta)
    {
        $this->tietokanta = $tietokanta;
    }

    public function haeMetodi()
    {
        // käytetään tietokanta-luokkaa
    }
}
```

Esimerkissä syötetään luokalle olio konstruktorissa, joka on perinyt tietokanta-luokan – itse olio voi olla esimerkiksi MySQL tai PostgreSQL -pohjainen. Tulevaisuudessa, jos halutaan vaihtaa tietokantaa, vaihdetaan vain alkuperäinen olion, joka syötetään

luokille. Kummassakin alhaalla olevassa esimerkissä syötetty tietokanta periytyy tietokanta-luokasta.

```
$esimerkki = new Esimerkki(new MySQL());

$esimerkki = new Esimerkki(new PostgreSQL());
```

DI toimii myös hyvin luokkien automaattisissa testauksissa, kunhan kaikki luokat käyttävät sitä, joten testausvaiheessa testatuille luokille voidaan syöttää erilaisia olioita tarpeiden mukaan. Normaalisti testatut luokat jouduttaisiin muokkaamaan testiä varten, koska käytetyt luokat, luokan sisällä, olisivat esimerkiksi yhteydessä oikeaan tietokantaan, mitä luonnollisesti halutaan välttää testauksessa. Sen sijaan, että luokka ottaisi yhteyden oikeaan tietokantaan, luokalle syötetään DI:ssä tekotietokanta-luokan olio. Tekotietokanta tässä tapauksessa kuvastaisi oikeata tietokantaa, jolloin testin tulokset olisivat luotettavia, mutta tapahtuisivat suljetussa ympäristössä halutulla tavalla. (Prasanna 2009, 112-116)

6. OHJELMISTON ARKKITEHTUURI

Verkkokauppaohjelmistoon suunniteltiin oma sovelluskehys sen sijaan, että käyttöön otettaisiin jokin tunnettu PHP-sovelluskehys. Etuna valmiissa sovelluskehyksissä olisi luonnollisesti se, että ne ovat erittäin testattuja ja suoraan käyttövalmiita, jolloin suurin osa työstä kohdistuisi itse verkkokauppaohjelmiston kehitykseen. Ikävä kyllä melkein kaikissa ”jokapaikanhöylä” PHP-sovelluskehyksissä on sama ongelma, eli ne ovat hitaita ja kuluttavat paljon muistia, joten niistä ei monikaan sovellu halvempiin webhotelleihin (PHP Framework benchmark: Zend, CodeIgniter & CakePHP 2009, hakupäivä 1.4.2012; PHP MVC Framework Performance – Part 1 2010, hakupäivä 1.4.2012; PHP framework comparison benchmarks 2009, hakupäivä 1.4.2012).

Verkkokauppaohjelmiston yksi tavoitteista on toimia nopeasti, jolloin verkkokaupan omistaja ei joudu ongelmiin webhotellinsa kanssa – suurin osa käyttäjistä ei varmaankaan tajua, että heidän 2,95 €/kk webhotelli on erittäin tarkka kuinka resursseja palvelimella käytetään. Joissain isoimmissa webhotelleissa toimii 1000 – 2000 käyttäjätiliä yhdellä palvelinkoneella, joista jokainen tili voi pyörittää useampaa nettisivua, jolloin resurssit voivat olla tiukalla (Shared hosting – WHTwiki, hakupäivä 31.3.2012).

Jos verkkokauppaohjelmisto olisi tehty yhdelle asiakkaalle, jolla olisi vara parempaan webhotelliin, tai vaikka omaan palvelimeen, ohjelmistossa olisi luonnollisesti käytetty valmista PHP-sovelluskehystä. Ohjelmisto tulee kuitenkin olemaan ilmaiseksi ladattavissa, jolloin oletuksena kuitenkin on, että suurin osa käyttäjistä on halvemman luokan webhotelleissa, joissa on tiukat säännöt prosessorin ja muistikulutuksen kanssa. Esimerkiksi monet webhotellit estävät PHP-tiedostoja pyörimästä, jos ne kuluttavat X verran resursseja muutaman kymmenen sekunnin ajan tai yhteensä päivän aikana – pahimmassa tapauksessa tili suljetaan kokonaan (CPU minutes, hakupäivä 31.3.2012; CPU Resource Usage, hakupäivä 31.3.2012).

Muistinkulutus on asetettu ohjelman sovelluskehyksessä maksimissaan 8 megatavuun, mutta ylläpito ohjaimissa, joissa käsitellään kuvia, muistinkulutus voi kasvaa sen yli, mutta se riippuu täysin ylläpitäjän lataamista kuvista. Koska sovelluskehyksen

muistinkulutus on asetettu vain 8 megatavuun, se on erittäin alhainen verrattuna muihin sovelluskehyksiin ja huomattavasti nopeampi. Esimerkiksi kilpaileva verkkokauppaohjelmisto Magento käyttää Zend Framework sovelluskehystä, minkä takia se vaatii minimissään 256 megatavua muista, mutta sivustolla suositellaan jopa 512 megatavua (Magento - System Requirements, hakupäivä 31.3.2012).

Omassa sovelluskehyksessä on kuitenkin etunsa, sillä se on suunniteltu vain yhteen tehtävään, jolloin ylimääräisiä rihkama-luokkia ei ole esillä sekoittamassa pakkaa, vaan ainoastaan tarpeelliset apuluokat ovat esillä. Luonnollisesti, koska sovelluskehyksessä käytetään MVC-arkkitehtuuria, kuka tahansa kehittäjä, joka on käyttänyt MVC-pohjaisia sovelluksia, voi lyhyen tutustumisen jälkeen ymmärtää suurimman osan verkkokauppaohjelmiston toimivuudesta.

Verkkokauppaohjelmisto on melkein 100 %:sti oliopohjainen, ainoastaan etuohjain on proseduraalisesti ohjelmoitu, sillä sen tehtävä on esimerkiksi luoda oliot ohjelmistolle ja lähettää ne käytettäväksi ohjaimille. Ohjelmiston tiedostot on jaettu siten, että yksi luokka sijoitetaan yhteen tiedostoon. Tällä tavoin vain tarvittavat luokat ladataan ja ohjelma toimii huomattavasti nopeammin kuin ohjelmissa, joissa yksi tiedosto sisältää melkein 1 / 5 ohjelman funktioista, mutta tarvitsee vain murto-osan niistä.

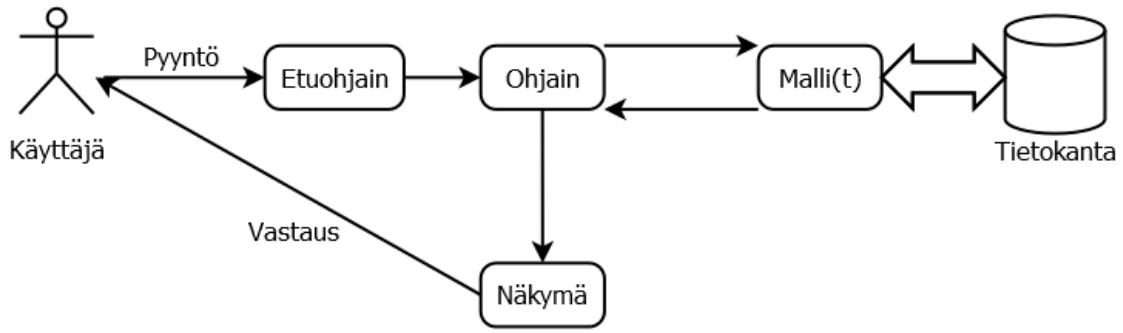
Sovelluskehyksessä käytettiin PHP 5.3:sen tarjoamia nimiavaruuksia, jolloin vältetään luokkien nimien yhteentörmäykset. Kaikki luokkien nimiavaruudet ovat nimetty tiedostohierarkian mukaan, jolloin niitä voidaan käyttää spl_autoload ominaisuuden kanssa. Spl_autoload ominaisuuteen voi rekisteröidä oman funktion tai luokan metodin, joka lataa automaattisesti luokan tiedoston, jos sitä ei ole aikaisemmin ladattu. (PHP: Autoloading Classes – Manual, hakupäivä 31.3.2012) Kutsumalla luokkaa ”Esimerkki”, joka on rekisteröity ”Tiedosto\Hierarkia” nimiavaruuteen, antaisi luokalle lopulliseksi nimiavaruudeksi ”Tiedosto\Hierarkia\Esimerkki”.

Rekisteröity spl_autoloader voisi ladata esimerkki luokan tiedoston seuraavasta polusta:

```
/tiedosto/hierarkia/esimerkki.php
```

Jolloin tilanteissa, jossa luokkaa ei ole ladattu, ohjelma osaisi ladata luokan automaattisesti, eikä ohjelma kaadu.

Kuten kaikki modernit MVC-arkkitehtuuriin pohjautuvat web-sovelluskehykset, verkkokauppaohjelmiston sovelluskehys on ottanut vaikutteita Sun Microsystemsin ja Apache Strutsin ”Model 2” suunnittelumallista. (The Model-View-Controller Design Pattern: 'Model 2' JSP Development, hakupäivä 31.3.2012) Erona kuitenkin siihen on, että verkkokauppaohjelmiston sovelluskehyksessä ainoastaan ohjain vaaditaan sivun luomiseen, mallit ja näkymät ovat vapaaehtoisia – eli tilanteissa, jossa näkymää ei tarvita, ohjain palauttaa vastauksen. Alla oleva kuva esittää tyypillistä pyyntöä verkkokauppaohjelmistoon, jossa kaikki osat ovat käytössä:

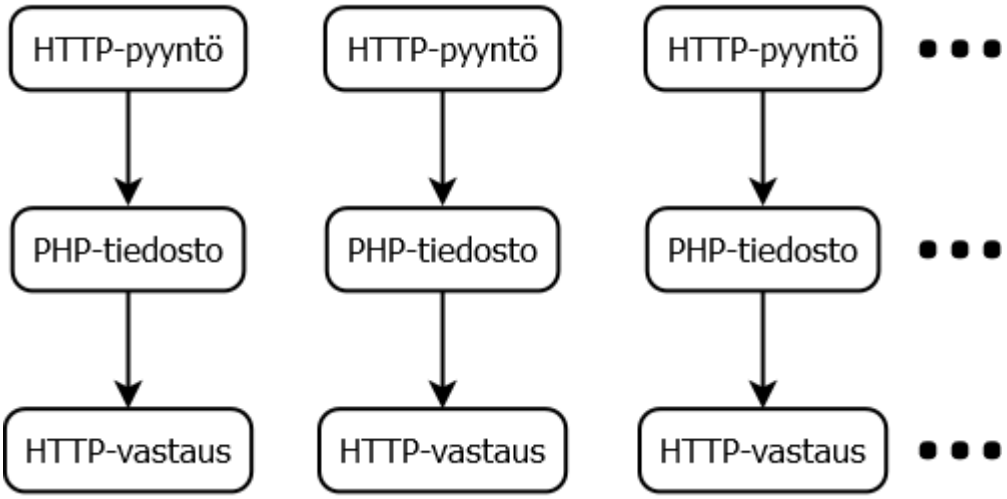


Kuva 10. Sovelluskehyksen MVC-hierarkia

Alla olevissa osioissa käsitellään verkkokauppaohjelmiston arkkitehtuuria yleisesti.

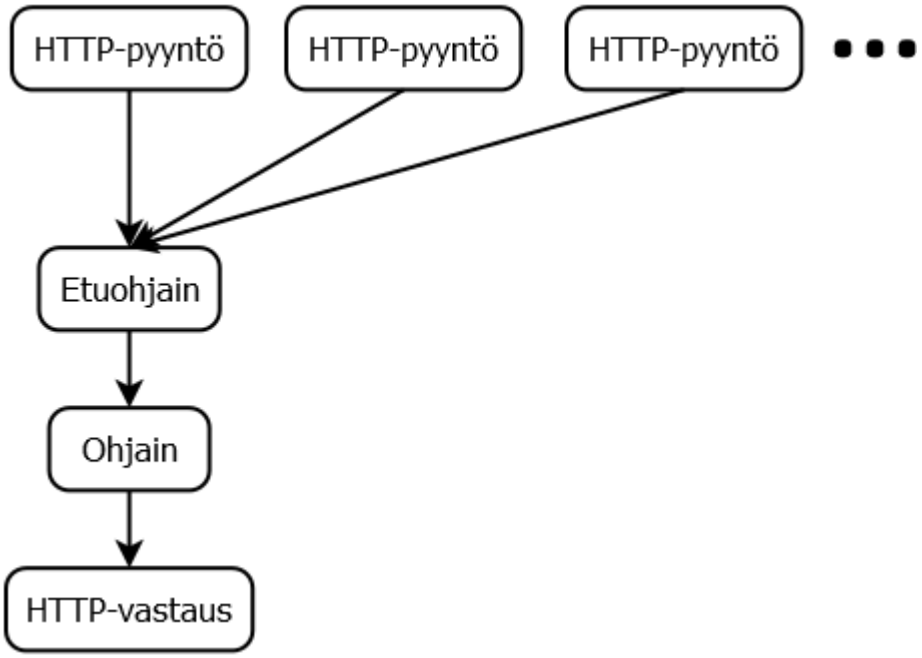
6.1. Etuohjain

Perinteinen PHP web-sovellus koostuu useista PHP-tiedostoista, joiden tehtävä on luonnollisesti vastata takaisin käyttäjien pyyntöihin. Yleensä kyseisessä suunnittelumallissa jokainen sivu on erillinen PHP-tiedosto, joissa ikävä kyllä toistetaan koodia, mitä kautta niiden ylläpitäminen muuttuu myös hankalammaksi (Zandstra 2010, 235). Alla olevassa kuvassa on esillä kyseinen suunnittelumalli.



Kuva 11. Perinteinen PHP web-sovellus

Web-pohjaisissa MVC-arkkitehtuureissa käytetään yleensä etuohjainta, jonka tehtävänä on vastaanottaa kaikki pyynnöt. Etuohjaimen ansiosta koodia ei tarvitse toistaa, sillä kaikki HTTP-pyynnöt tulevat yhteen tiedostoon. Verkkokauppaohjelmiston etuohjain idea perustuu, kuten kymmenissä tuhansissa muissakin web-sovelluksissa, Sun Microsystemsin Core J2EE Patterns etuohjain malliin, jonka avulla pyydetty ohjain ladataan. (Zandstra 2010, 235-236) Kuvassa 12 on esillä Core J2EE Patternsin keskeinen idea, joka on sovellettu verkkokauppaohjelmistoon.



Kuva 12. Core J2EE Patternsiin perustuva etuohjain

Kun HTTP-pyyntö tulee sisään etuohjaimeen, eli index.php -tiedostoon, ohjelma ensiksi tarkistaa onko PHP versio 5.3 tai uudempi asennettu. Samalla tarkistetaan onko tarvittavat lisäosat, kuten PDO-luokka, käytettävissä. Jos PHP versio on liian pieni tai lisäosia puuttuu, käyttäjälle ilmoitetaan ystävällisesti korjaamaan viat, eli päivittämään PHP tai asentamaan lisäosat. Syy tarkistuksiin johtuu siitä, että käyttäjä saattaa muuttaa uuteen webhotelliin, josta puuttuu tarvittavia lisäosia tai PHP versio on liian vanha.

Etuohjain sisältää vain PHP 5.2 ja siitä alaspäin yhteensopivaa koodia, jolloin edellä mainitut virheet voidaan ilmoittaa ilman kaatumista. Tarkistuksien jälkeen ladataan toinen PHP-tiedosto, joka sisältää PHP 5.3 ominaisuuksia, esimerkiksi nimiavaruuksia. Kyseisessä tiedostossa ladataan kaikki tarvittavat apuluokat, joita käytetään pitkin ohjelmistoa. Etuohjaimessa reititin tarkistaa, minkä ohjaimen etuohjain muodostaa olioksi ja mitä metodia kyseisestä oliosta kutsustaan.

Koska ohjelmisto käyttää Dependency Injection suunnittelumallia luokkien riippuvuuksien hoitamiseksi, kaikkien apuluokkien siirtäminen ohjaimelle konstruktorin kautta on ongelma. Web-pohjaisissa MVC-arkkitehtuureissa on tyypillisesti otettu käyttöön säiliö (container), jolla kaikki DI riippuvuudet siirretään ohjaimelle. Etuohjain lataa kaikki luokkien oliot säiliöön ja siirtää säiliön ohjaimen konstruktoriin. Kun ohjain on lopulta palauttanut syötteen takaisin, se lähetetään käyttäjälle.

6.2. Reititin

Sovelluksen reititintä käytetään etuohjaimessa tutkimaan käyttäjän pyydettyä osoitetta, jonka avulla oikea ohjain ladataan ja sen metodia kutsutaan – pyynnössä voi olla myös parametreja, joita ohjain voi käyttää ylimääräisenä syötteenä. Etuohjaimen reitittimen ymmärrettäväksi osoite muodoksi valittiin seuraava hakukoneystävällinen tapa:

index.php?/ohjain/metodi/parametrit/

Koska ohjaimet ovat PHP-tiedostoja, joita ladataan kiintolevyltä, reititin ohjelmoitiin myös ymmärtämään ohjaimien kohdalla alikansioita, joten pyyntö voi olla myös seuraavanlainen:

index.php?/alikansio1/alikansio2/…jne/ohjain/metodi/parametrit/

Käyttäjän syöttämä query string saadaan $_SERVER['QUERY_STRING'] super globaalista ohjelman alkuvaiheessa, mikä puolestaan siirretään erilliseen luokkaan, jota kautta sitä käytetään. Hakukoneoptimoinnin kannalta reititin asetettiin vaatimaan, että query string alkaa kauttaviivalla ja on pidempi kuin 2 merkkiä, muuten ”sivua ei löydy” -ohjain ladataan. Query string jaetaan kauttaviivalla taulukkoon, jolloin jokaista osaa voidaan käsitellä erikseen. Esimerkiksi pyyntö

index.php?/alikansio/ohjain/metodi/

Muodostuu seuraavaksi taulukoksi, jossa ensimmäinen tyhjä solu jätetään huomioimatta:

```
array
  0 => string '' (length=0)
  1 => string 'alikansio' (length=9)
  2 => string 'ohjain' (length=6)
  3 => string 'metodi' (length=6)
  4 => string '' (length=0)
```

Reitin käy läpi taulukon jokaisen solun, kuten liitteen 1 vuokaaviossa on esitetty, yksinkertaisella for-silmukalla. Reititin yrittää aluksi etsiä ohjainta, mutta jos taulukon solu sisältää muita merkkejä kuin a-z tai 0-9, käsittely lopetetaan ja ”sivua ei löydy” -ohjain ladataan. Syy ankaraan tarkasteluun liittyy tietoturvaan, jota käsitellään enemmän myöhemmässä vaiheessa.

Reititin aloittaa taulukon ensimmäisestä solusta ja olettaa, että se on ohjain. Jos ohjain-kansion juuresta ei löydy solun nimistä ohjaintiedostoa, tarkistetaan, löytyykö samanniminen alikansio johon voi siirtyä, minkä jälkeen siirrytään taulukon seuraavaan soluun ja aloitetaan alusta. Jos ohjainta ja alikansiota ei löydy, niin taulukon käsittely lopetetaan.

Heti kun ohjain on löytynyt, vaikka useamman alikansion sisältä, siirrytään taulukon seuraavaan soluun, joka oletetaan olevan ohjaimen metodi. PHP:n is_callable-funktiolla

voidaan tarkistaa, onko ohjain, eli luokka, ja sen metodi kutsuttavia. Jos ohjain ja luokka ovat kutsuttavia, ne asetetaan myöhempää käyttöä varten, missä ohjaimesta luodaan olio ja sen metodi kutsutaan. Jos taulukkoon jää jäljelle soluja, ne lisätään parametreiksi ohjaimelle.

6.3. Ohjaimet

Etuohjain täydentää MVC-arkkitehtuuria luomalla olion ohjaimesta (luokasta) ja kutsumalla sen metodia, jonka tehtävä on rakentaa sivusto. Pelkkänä luokkana ohjain ei pysty tekemään mitään, sillä se tarvitsee erilaisia funktioita pystyäkseen kutsumaan muita ohjaimia ja yleensä työntämään tiedot näkymälle. Tässä kohtaa verkkokauppaohjelmiston sovelluskehys astuu kuvioon.

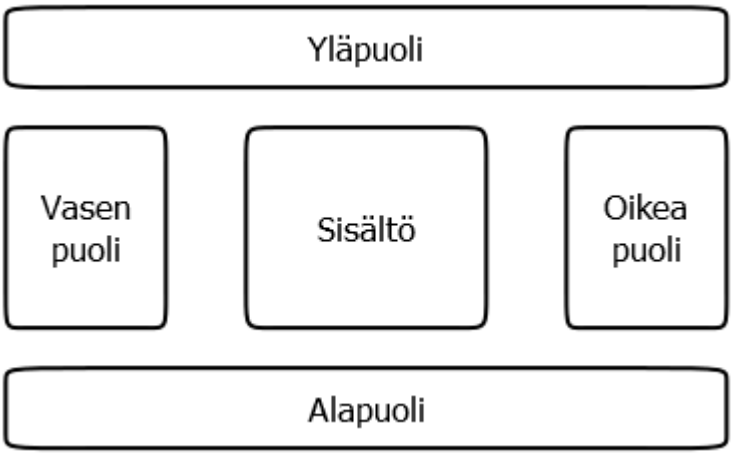
Ohjelmistossa on abstrakti ohjain luokka nimeltä Controller, joka sisältää erilaisia metodeja, joita sivuston ohjaimien täytyy käyttää. Ideana abstrakti luokassa on se, että kaikki ohjaimet periytyvät siitä, jolloin ne saavat metodit käyttöön. Kun ohjain periytyy sitä, se pystyy kutsumaan muita ohjaimia, asettamaan näkymän, sekä myös lähettämään tiedot näkymälle ja palauttamaan kyseisen HTML-dokumentin. Abstrakti luokan yksi tärkeimmistä tehtävistä on vastaanottaa Dependency Injection säiliö, joka sisältää kaikki ohjelman apuluokat, ja sen jälkeen purkaa ne luokkaan attribuuteiksi, jolloin niitä voidaan kutsua.

Normaalisti kaikkiin ohjaimiin jouduttaisiin kopioimaan kaikki metodit, jotta niitä voitaisiin käyttää, mutta periytymällä abstraktista luokasta, niin ei tarvitse tehdä. Esimerkiksi jos ”index.php?/testi/sivu/” osoite haluttaisiin toimivan, testi.php niminen tiedosto luotaisiin ja se sisältäisi seuraavan koodin, jossa perittäisiin abstrakti luokka:

```
<?php

namespace mvCart\MVC\Controllers;

class Testi extends \mvCart\Core\Classes\Controller
{
    public function sivu()
    {
        // koodia sivun luomiseen
```

```
    }
}

?>
```

Ohjain perii abstrakti luokan, joten ylimääräistä koodia ei tarvitse kirjoittaa, koska kaikki apuluokat ja metodit ovat käytettävissä kulissien takana. Verkkokauppaohjelmistossa koodin uudelleenkäyttö on myös parannettu ohjaimien ja näkymien välillä. Perinteisessä MVC-sovelluskehyksessä sivuston ulkoasu jouduttaisiin jakamaan viiteen osaan, kuten alla olevassa kuvassa on esitetty.



Kuva 13. Perinteinen sivuston ulkoasu

Jotta ulkoasu saataisiin toimimaan oikealla tavalla, jokaisessa ohjaimen näkymässä pitäisi olla neljä muuttujaa, jotka tulostaisivat yläpuolen, vasemman puolen, oikean puolen ja alapuolen. Jokainen muuttuja sisältäisi HTML-dokumentin vastaavasta ohjaimesta.

Ikävä kyllä kyseinen menettely ei ole koodin uudelleenkäytön kannalta järkevää, joten verkkokauppaohjelmistossa asia hoidetaan eri tavalla. Ohjelmassa on MainTemplate niminen ohjain, joka hoitaa yllä olevan kuvan tapaisen HTML-dokumentin tulostuksen kutsumalla yläpuolen, vasemman puolen, oikean puolen ja alapuolen ohjaimet erikseen. Ideana on, että käyttäjän kutsumat ohjaimet, jotka tulostavat kuvan 13 tyyppisen sivun, käyttäisivät MainTemplate ohjainta, sen sijaan, että toistaisivat itse koodia. MainTemplate ohjain astuu voimaan, kun abstrakti luokan metodia kutsutaan ohjaimessa:

```
$this-> useMainTemplate(true);
```

Tämän jälkeen, kun ohjain siirtää tiedot näkymään, lopullinen tieto lähetetään MainTemplate ohjaimelle, joka asettaa käyttäjän kutsuman ohjaimen HTML-dokumentin kuvan 13 ”Sisältö” osaan.

Ylläpitopuolen ohjaimet

Kun ohjelman ylläpito ohjaimia kutsutaan, käyttäjän istunto pitää varmistaa, jotta kyseessä on oikeasti ylläpitäjä, eikä hyökkääjä. Normaalisti reititin tarkistaisi osoitteen ja huomaisi, että ylläpitoon liittyvä ohjain on kutsuttu, jolloin käyttäjän istunto pitää tarkistaa. Ikävä kyllä, jos hyökkääjä pystyy jollain tapaa huijaamaan reititintä, hän pystyisi tekemään melkein mitä tahansa.

Sen sijaan, että reitittimessä tarkistettaisiin, että onko kutsuttu ohjain ylläpitoon liittyvä, kaikki ylläpito ohjaimet perivät AdminController nimisen abstrakti luokan, joka puolestaan perii Controller abstrakti luokan. Kyseisessä luokassa konstruktori tarkistaa, onko käyttäjän istunnolla oikeudet kutsua ohjainta. Hyökkääjä ei pysty millään tavalla ohittamaan tarkistusta, koska ohjaimen metodeja ei pysty kutsumaan ilman, että siitä luotaisiin olio – jos ohjaimesta luodaan olio, AdminController luokan kontruktorin tarkistus käydään pakosta läpi.

Asiakaspuolen ohjaimet

Sivustoissa, joissa asiakkaan pitää olla kirjautuneena, sovelletaan samaa tekniikkaa kuin ylläpitopuolen ohjaimissa. Kyseisissä ohjaimissa periydytään AccountController abstraktista luokasta, joka puolestaan perii Controller abstrakti luokan. Asiakas sivustoissa ei tarvita yhtä paljon tietoturvaa kuin ylläpitopuolella, mutta samalla menetelmällä koodia ei tarvitse toistaa ja aina varmistetaan, että asiakas on kirjautunut sisään tarvittavissa sivustoissa.

6.4. Mallit

Normaalisti MVC-pohjaisissa ohjelmissa mallit pitää ladata erikseen, jotta niiden metodeja voidaan kutsua. Verkkokauppaohjelmiston mallien käyttö tehtiin kuitenkin helpommaksi, eli ne ladataan automaattisesti, jos niitä ei ole aiemmin ladattu. Kyseinen menetelmä kuitenkin vaatii jonkinlaisen pelisäännön, jotta mallien lataus onnistuisi aina ilman ongelmia. Mallien nimet pitää ohjelmassa muodostaa CamelCase kirjoitustyylin mukaan, jossa nimen KaikkiSanatOvatYhdessäJaAlkavatIsoillaAlkukirjaimilla. CamelCase kirjoitustyylin avulla mallin nimi voidaan jakaa osiin käyttämällä isokirjaimia, missä jokainen sana kuvastaa alikansiosta ja viimeinen sana mallin tiedoston nimeä.

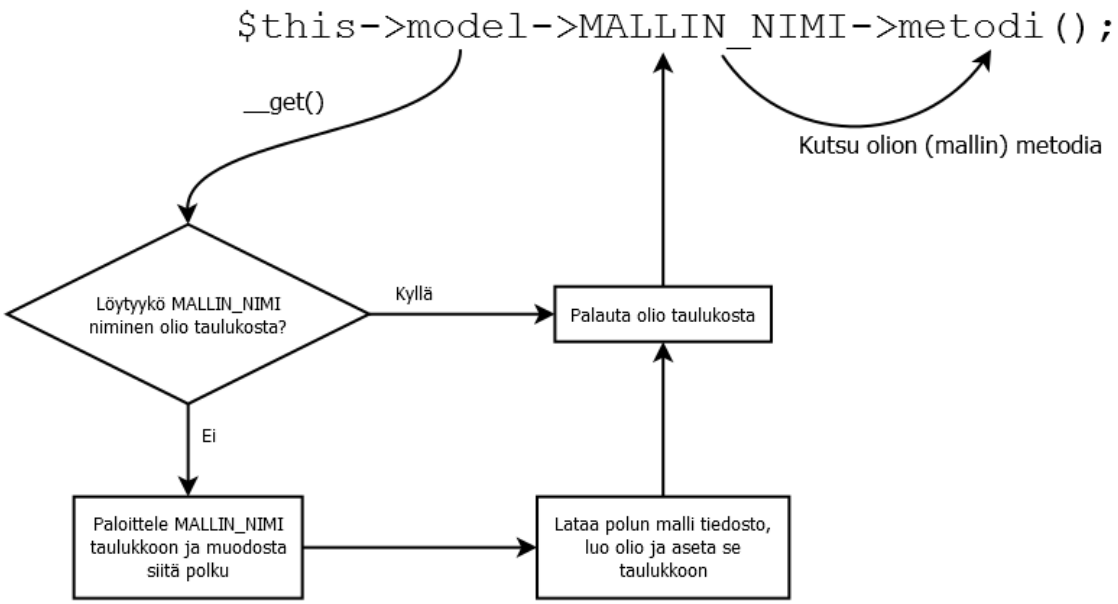
Esimerkiksi malli TämäOnEsimerkki luokiteltaisiin seuraavaksi poluksi:

```
/tämä/on/esimerkki.php
```

Menetelmässä käytetään myös PHP:n taikametodeja (magic methods), joiden avulla voidaan dynaamisesti luoda uusia attribuutteja ja metodeja, joita ei ole olemassa (PHP: Overloading - Manual, hakupäivä 29.3.2012). Ohjelmaan luotiin apuluokka, joka sisältää PHP:n __get() taikametodin, jota voidaan abstrakti ohjain luokan avulla kutsua attribuuttina:

```
$this->model->MALLIN_NIMI->metodi();
```

Ideana on, että model attribuutin jälkeen kutsutun mallin nimi puuttuu, jolloin PHP yrittää viimeisenä oljenkortena lähettää mallin nimen __get() taikametodille. Kyseisen metodin sisällä mallin nimi pilkotaan taulukkoon isojen kirjaimien perusteella ja muunnetaan ladattavaksi tiedostoksi, minkä jälkeen mallista luodaan olio. Luonnollisesti taikametodiin tehtiin optimointeja, eli jos pyydetyn mallin oliota ei löydy taulukosta, sitä ei ole ladattu, joten se ladataan ja olio siirretään taulukkoon seuraavaa kertaa varten. Mallin olio luonnollisesti palautetaan, jolloin sen metodia voidaan kutsua. Alla olevassa kuvassa on asia selitetty paremmin.



Kuva 14. Mallien metodien käyttö

Kutsumalla seuraavaa mallia:

```
$this->model->Opinnäytetyö->palautaTyö();
```

Lataisi opinnäytetyö.php mallitiedoston, josta luotaisiin olio ja sen metodia, palautaTyö(), kutsuttaisiin.

6.5. Näkymät

Näkymät verkkokauppaohjelmistossa toimivat MVC-arkkitehtuurin mukaisesti, mutta ovat työntö-pohjaisia (push-based). Näkymät eivät sisällä melkein mitään logiikkaa, joten niiden ainoa tehtävä on tulostaa muuttujien arvot, jotka on lähetetty ohjaimen kautta niille. Tiedonsiirto ohjaimen ja näkymän välillä on yksinkertaista: ohjaimella on view niminen taulukko attribuutti käytössä, mihin asetetaan tietoa. Tieto asetetaan taulukkoon seuraavasti:

```
$this->view['muuttujan_nimi_näkymään'] = 'teksti tähän';
```

View attribuutti taulukon nimi muutetaan myöhemmässä vaiheessa PHP:n extract-funktiolla oikeiksi muuttujiksi, jotka annetaan näkymälle. Esimerkiksi jos näkymä sisältää seuraavan HTML-dokumentin:

```
<div>
    <p>
        <?php echo $esimerkki_tulostus; ?>
    </p>
</div>
```

Muuttujaan $esimerkki_tulostus voidaan lisätä ohjaimessa tietoa seuraavasti:

```
$this->view['esimerkki_tulostus'] = 'teksti tähän';
```

Joka muuttaisi näkymän HTML-dokumentin seuraavaksi:

```
<div>
    <p>
        teksti tähän
    </p>
</div>
```

PHP:lle on myös tarjolla useita näkymä-moottoreita (template engine), esimerkiksi Smarty Template Engine on erittäin suosittu ja käytetään useassa PHP web-sovelluksessa. Ikävä kyllä Smartyn tyylisissä näkymä-moottoreissa ongelmaksi kuitenkin muodostuu niiden resurssien käyttö. Tilanteessa, jos verrattiin Smartyä ja vastaavaa ”raakaa” PHP-koodia, Smarty oli keskimäärin 15–20 kertaa hitaampi kuin raaka PHP, sekä myös kulutti yli 15 kertaa enemmän muistia (Rain TPL - Speed Test, hakupäivä 29.3.2012).

Tämä on yksi suurimmista syistä, miksi Smartyn tyylistä näkymä-moottoria ei otettu käyttöön. Näkymä-moottorien käyttöönotto aiheuttaa myös ongelmia henkilöille, jotka eivät ole aikaisemmin käyttäneet sellaista. He voivat sitä kautta välttää verkkokauppaohjelman käyttöönottoa, koska he eivät pysty luomaan omia teemoja nopeasti ja helposti. PHP:n syntaksi on kuitenkin yksinkertainen, joten henkilöt, jotka ovat käyttäneet sitä vähänkin, pystyvät luomaan oman teemansa verkkokauppaan.

6.6. Lisäosat

Lisäosien tukeminen on verkkokauppaohjelmiston yksi tärkeimmistä ominaisuuksista, sillä erilaisia maksu- ja postitustapoja on tuhansia pitkin maailmaa, mutta niitä kaikkia ei voida tukea suoraan ohjelmistossa. Ideana on, että sovelluksen kehittäjät voivat luoda omat lisäosat vaivatta ja helposti. Verkkokauppaohjelmisto tukee opinnäytetyön valmistumisen yhteydessä maksu-, postitus-, sekä ”widget” lisäosia. Widgeteillä tässä tapauksessa viitataan sivuston ulkoasun komponentteihin, jotka tunnetaan myös moduuleina ja blockeina, joita ylläpitäjä voi järjestää halutulla tavalla.

Kaikki lisäosat on jaettu erikseen extensions nimiseen kansioon, jossa maksutavat ovat payments-kansiossa, postitustavat shipping-kansiossa ja widgetit widget-kansiossa. Jokainen lisäosa pitää laittaa sille kuuluvaan alikansioon, jotta ohjelma osaa käyttää sitä, esimerkiksi kaikki maksutavat ovat luonnollisesti payments-alikansiossa tai muuten verkkokauppaohjelmisto ei löydä niitä.

Lisäosa pitää laittaa lisäosan nimiseen alikansioon, joka saa sisältää vain a-z ja 0-9 merkkejä. Lisäosa kansion sisällä olevat ohjain ja malli ovat nimetty seuraavasti:

```
LISÄOSAN_NIMI-controller.php
LISÄOSAN_NIMI-model.php
```

Kaikki lisäosan näkymät löytyvät erikseen view nimisestä kansiosta. Kaikilla lisäosilla on käytössä samat mallit ja näkymät kuin normaaleilla ohjaimilla, mutta ne voivat myös kutsua omia malleja lisäämällä PaymentExtension, ShippingExtension tai WidgetExtension sanan mallin eteen. Esimerkiksi maksutavalla, jonka nimi on PayPal, on sijoitettu payments kansioon, jossa se sijaitsee paypal nimisessä alikansiossa. PayPal lisäosa pystyy kutsumaan omaa malliaan seuraavasti:

```
$this->model->PayPalPaymentExtension->metodi();
```

Kyseinen kutsu lataa mallin seuraavasta polusta:

```
/extensions/payments/paypal/paypal-model.php
```

MVC-arkkitehtuurin ansiosta ohjelma on myös yleensä laajennettavissa, eli jos verkkokaupalle halutaan luoda uusi sivu, se on mahdollista luomalla sille ohjain. Ohjelmiston reititin on dynaaminen, joten jos ohjainkansioon lisätään uusi ohjain, se voidaan ladata automaattisesti kutsumalla sitä osoiterivissä ilman, että mitään asetuksia pitää muuttaa tai lisätä.

6.7. Unicode-tuki

PHP:ssä ei ole sisäänrakennettua tukea Unicode-merkistölle, vaan oletuksena on, että kaikki merkit ovat yhden tavun pituisia (PHP: Strings - Manual, hakupäivä 22.3.2012). Tämä aiheuttaa ongelman ulkomaankielissä, joissa merkkien esittämiseen tarvitaan kaksi tavua tai enemmän. PHP:lle on olemassa Multibyte String lisäosa, joka tukee Unicode-merkistöä, joten kyseistä lisäosaa käytetään verkkokauppaohjelmistossa (PHP: Introduction - Manual, hakupäivä 22.3.2012).

Internetin suosituin Unicode-merkistökoodaus on UTF-8, joten se otettiin käyttöön ohjelman sisäisenä merkistökoodauksena, sekä myös HTML-dokumenteissa. UTF-8:ssa normaalit ASCII-merkit, 0-127, ovat samat kuin ennen ja vievät vain yhden tavun tilaa, sen jälkeen kaikki merkit vievät 2-6 tavua tilaa, riippuen merkistä (Spolsky 2003, hakupäivä 22.3.2012). Kun verkkokauppaohjelmisto vastaanottaa syötteen käyttäjältä, sen merkistökoodaus tarkistetaan. Jos merkistökoodaus ei ole UTF-8, merkistökoodaus tunnistetaan Multibyte String lisäosalla ja sen jälkeen muunnetaan iconv lisäosalla UTF-8:ksi.

MySQL:ässä on useita Unicode-merkistökoodauksia, joista suosituimmat ovat utf8_general_ci ja utf8_unicode_ci. Verkkokauppaohjelmiston taulukoille valittiin utf8_unicode_ci merkistökoodaukseksi, koska se on huomattavasti tarkempi merkistöjen kanssa. Vaikka utf8_general_ci on nopeampi, sen tarkkuus ei ole yhtä hyvä merkistöjen vertailussa. Esimerkiksi Œ -merkki on yhtä kuin OE, mutta utf8_general_ci luokittelisi sen pelkäksi E:ksi. Ainoa ongelma kummassakin merkistökoodauksessa on se, että ne vain tukevat UTF-8 merkkejä kolmeen tavuun asti. (MySQL 5.1 Reference Manual :: 10.1.13.1 Unicode Character Sets, hakupäivä 22.3.2012)

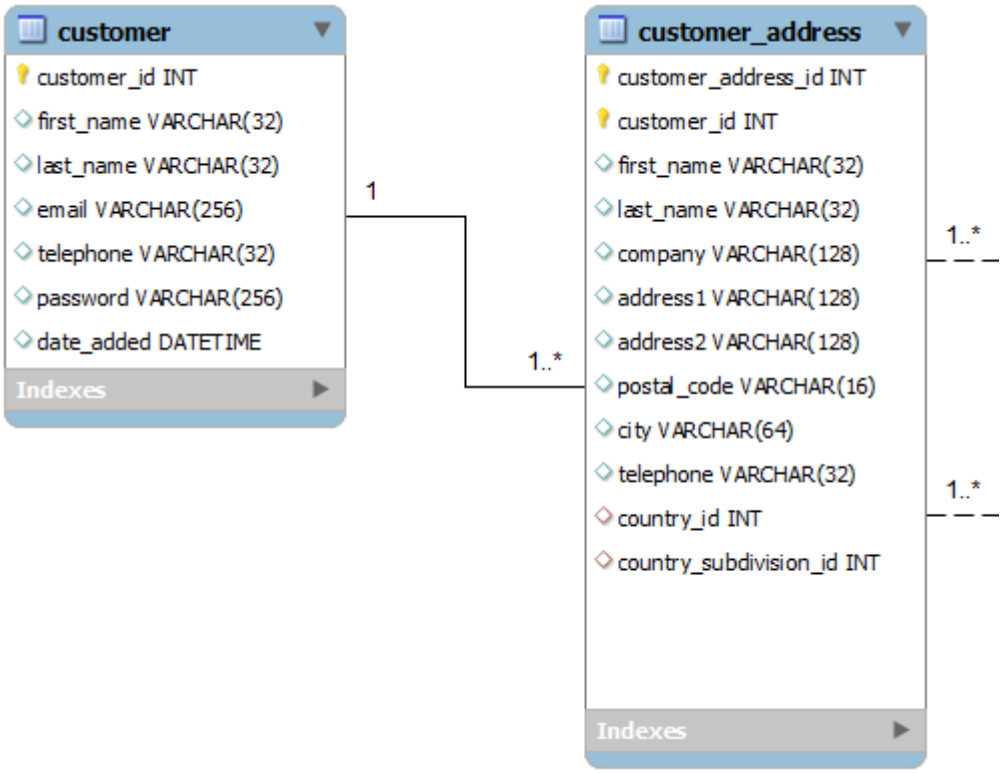
6.8. Relaatiotietokanta

Verkkokauppaohjelmiston relaatiotietokannan suunnitteluun käytettiin MySQL AB:n kehittämää MySQL Workbench ohjelmistoa. Ohjelma tarjoaa huomattavia etuja verrattuna siihen, että tietokanta ”suunniteltaisiin” kirjoittamalla se käsin. Tietokannan suunnittelussa käytettiin InnoDB tallennusmoottorin tarjoamaa foreign key -ominaisuutta, jolla voidaan helpottaa tietokannan ylläpitämistä. Esimerkiksi tilanteessa, jossa uusi taulukko lisätään tietokantaan, sen tiedon eheydestä ei tarvitse murehtia ohjelmistossa, sillä tietokanta pystyy automaattisesti päivittämään ja poistamaan siihen liittyvät tiedot.

Alla olevissa alaotsikoissa käydään läpi tärkeimmät osat tietokannasta ja miksi kyseisiin ratkaisuihin päädyttiin.

6.8.1. Asiakkaat

Asiakastaulukko on ”jaettu kahtia”, kuten kuvassa 15 on esitetty. Customer-taulukko sisältää asiakkaan tyypilliset tiedot, mutta hänellä voi olla yksi tai useampi tallennettu osoite, jota customer_address-taulukko kuvastaa. Foreign key on asetettu customer_address-taulukkoon, joten jos asiakas poistetaan tietokannasta, kaikki hänen tallennetut osoitteet poistuvat myös automaattisesti.



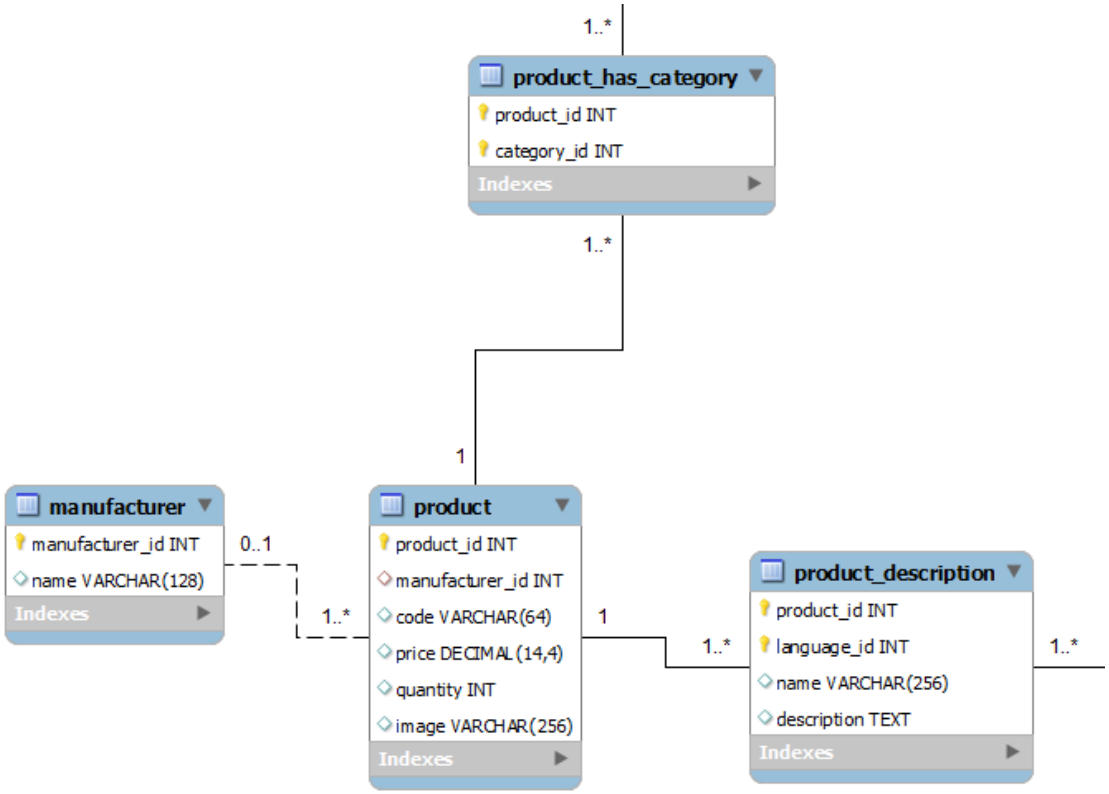
Kuva 15. Asiakastaulukot

6.8.2. Tuotteet

Kuten kuvasta 16 huomataan, tuotetaulukkoa edustaa product-taulukko, joka sisältää tuotteen tyypillisemmät tiedot. Tuotteella ja tuoteryhmällä on many-to-many -yhteys, jolloin niille luotiin yhteinen taulukko, jonka avulla yhdellä tuotteella voi olla useampi tuoteryhmä ja yhdellä tuoteryhmällä useampi tuote.

Tuotteella voi olla myös yksi valmistaja ja yksi valmistaja voi edustaa useampaa tuotetta. Tuotteen valmistajan ja tuotteen välille luotiin heikko foreign key -yhteys. Jos tuotteella on valmistaja ja se poistetaan tietokannasta, niin foreign key:llä asetetaan valmistajan ID-numero null-tilaan, joka kuvastaa, että tuotteella ei ole tällä hetkellä valmistajaa.

Tuoteseloste löytyy product_description-taulukosta, jossa jokainen tuoteseloste on liitetty yhteen kieleen. Jos kieli poistetaan ohjelmasta, niin tietokanta poistaa myös automaattisesti foreign key:n avulla kaikki kielen liittyvät tuoteselostukset.

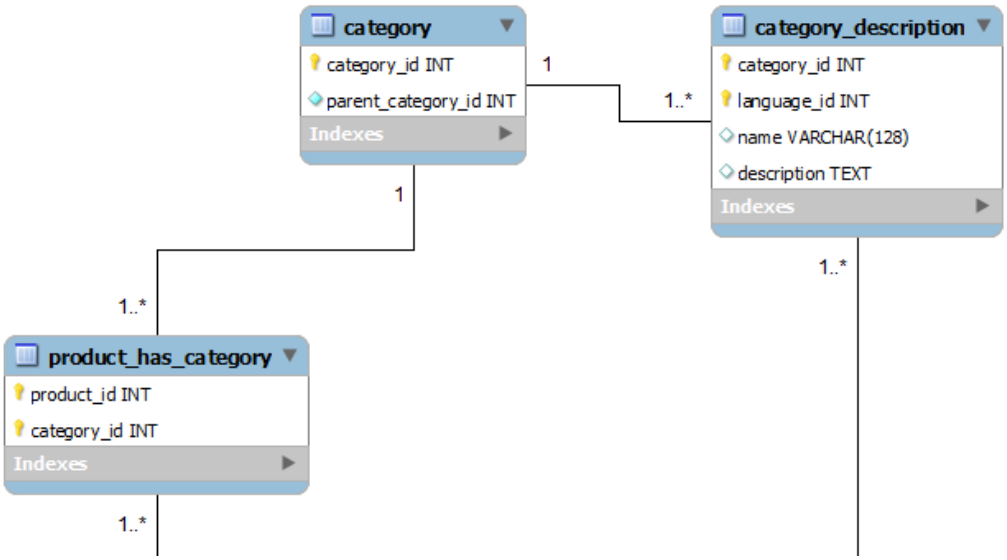


Kuva 16. Tuotetaulukko ja sen riippuvuudet

6.8.3. Tuoteryhmät

Tuoteryhmät toimivat melkein samalla tavalla kuin tuotteet. Yhdellä tuoteryhmällä voi olla useampi tuote ja tuotteella useampi tuoteryhmä. Tuoteryhmän kuvaus löytyy category_description-taulukosta, jossa jokainen kuvaus on liitetty yhteen kieleen. Jos kieli poistetaan ohjelmasta, niin tietokanta poistaa myös automaattisesti foreign key:n avulla kaikki kielen liittyvät tuoteryhmäkuvaukset.

Jokaisella tuoteryhmällä on oma ID-numero, sekä sen emotuoteryhmän ID-numero, kuten alla olevassa kuvassa on esitetty. Kun emotuoteryhmän ID-numero on nolla, kyseinen tuoteryhmä luokitellaan olevan päätuoteryhmä. Jos emotuoteryhmän ID-numero on suurempi kuin nolla, se kuvastaa jotain tuoteryhmää, jonka alla nykyinen tuoteryhmä on. Tällä tavoin voidaan helposti tietää kuinka ”syvällä” kukin tuoteryhmä on, minkä avulla taas voidaan siirtyä takaisin päätuoteryhmään.

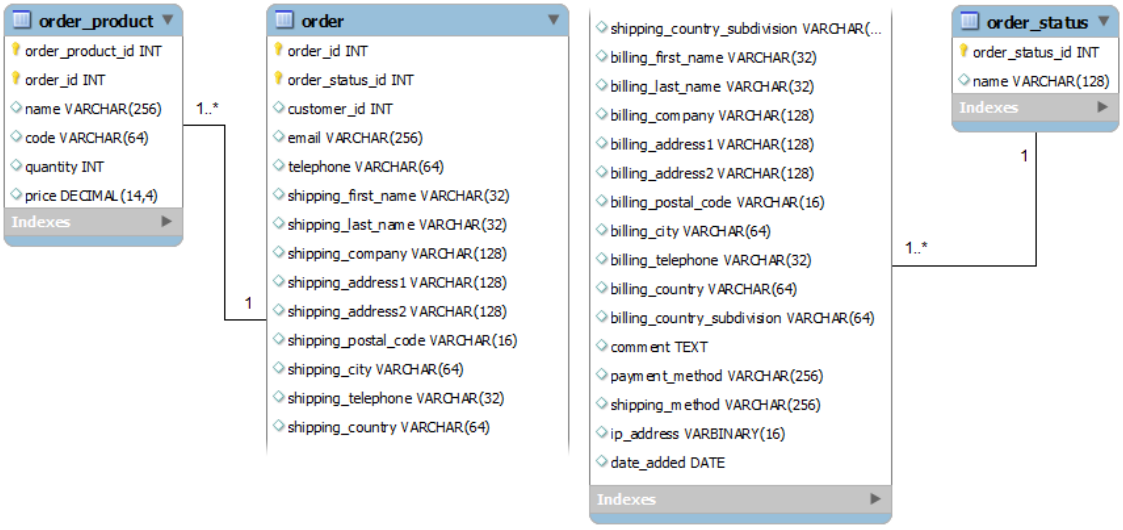


Kuva 17. Tuoteryhmät ja sen riippuvuudet

6.8.4. Tilaukset

Tilaustaulukko on jaettu kolmeen osaa, kuten kuvassa 18 on näkyvillä. Jokaisen tilauksen tiedot tallennetaan suoraan order-taulukkoon, eikä yhdistetä suoraan asiakkaaseen. Syy tähän on se, että jos asiakas poistetaan tietokannasta, myös tilauksen tiedot katoaisivat, mutta tätä ei haluta tapahtuvan. Ikävä kyllä kyseisellä menetelmällä samaa tietoa toistuu tietokannassa turhaan, mutta se on ainoa keino varmistaa, että tilauksien tiedot ovat aina oikein.

Sama pätee myös tilattuihin tuotteisiin, eli kaikki tuotteet tallennetaan order_product-taulukkoon. Jos tuotteet olisi yhdistetty alkuperäisiin tuotteisiin, tuotteiden muokkauksen yhteydessä myös vanhojen tilauksien tiedot muuttuisivat. Tilauksen nykyinen tilanne on yhdistetty order_status-taulukkoon, jossa on nimetty eri tiloja. Tiloina voisi olla esimerkiksi ”vastaanotettu”, ”postitettu”, ”maksettu”, jne.



Kuva 18. Tilaukset

6.9. Tietoturva

Tässä osiossa käsitellään verkkokauppaohjelmiston tietoturvaratkaisuja.

6.9.1. Tiedostojen tietoturva

Kuten yleisessä tietoturvaosiossa aikaisemmin mainittiin, jos web-sovellus antaa käyttäjän vaikuttaa ladattuihin tiedostoihin ilman tarkistuksia, hyökkääjä voi käyttää heikkoutta hyväksi ja manipuloida järjestelmää lataamaan tiedostoja, joihin hän ei normaalisti pääsisi käsiksi. Verkkokauppaohjelmiston MVC-arkkitehtuurissa käyttäjä pääsee vain etuohjaimen kautta vaikuttamaan minkä ohjaimen järjestelmä käsittelee. Kun käyttäjä käskee järjestelmää lataamaan tietyn ohjaimen, kyseessä on pohjimmiltaan PHP-tiedosto, jonka sisällä on luokka ja sen metodeja.

Verkkokauppaohjelmiston reititin pitää huolen, kun se tutkii pyydettyä osoitetta, että mahdollinen hyökkääjä ei pääse huijaamaan järjestelmää lataamaan vääriä tiedostoja. Kuten aikaisemmin reititin kappaleessa mainittiin, osoiterivi jaetaan kauttaviivalla taulukkoon ja jokainen osa käsitellään yksi kerrallaan, niin kauan, että ohjain löytyy ja on ladattu onnistuneesti. Reitittimessä tarkistetaan myös, että ohjaimen nimi sisältää vain a-z ja 0-9 -merkkejä – jos muita merkkejä havaitaan, niin käsittely lopetetaan ja

”sivua ei löydy” -ohjain ladataan. Mainitulla ohjain tarkistuksella vältetään directory traversal ja null-merkki hyökkäykset, jotka ovat erittäin suosittuja script kiddien keskuudessa.

Verkkokauppaohjelmiston tiedostorakenne on myös järjestetty niin, että tärkeisiin tiedostoihin ei pääse ulkopuolinen käsiksi. Kaikki kansiot, jotka eivät sisällä julkisia tiedostoja, on suojattu .htaccess tiedostolla, joka estää Apache palvelimissa ulkopuolisien käyttäjien pääsyn kansioihin. Vaikka hyökkääjä pääsisi kutsumaan nettiselaimella ohjain- tai mallitiedostoja, suojauksesta huolimatta, niin ne ovat silti suojassa, sillä kyseessä on PHP-tiedostoja, jotka sisältävät vain luokan. PHP:ssä, kuten muissakin ohjelmointikielissä, luokat eivät tee mitään ilman, että niistä luodaan olio ja olion metodeja kutsutaan, joten hyökkääjä ei hyödy tiedostoista mitään.

6.9.2. SQL-injektioiden esto

Tyypillisesti PHP web-sovelluksissa käytetään mysql_query-funktiota SQL-lauseiden suorittamiseen. Luonnollisesti, ilman tarkistuksia, hyökkääjä voisi joissakin tilanteissa muokata SQL-lausetta, joten mysql_real_escape_string -funktiota käytetään hyökkäyksien torjumiseen. Ongelmaksi kuitenkin muodostuu se, että jos ohjelmiston kehittäjä unohtaa käyttää kyseistä funktiota jossain kohtaa, ohjelmistoon voi syntyä vaarallinen tietoturva-aukko.

Verkkokauppaohjelmistossa otettiin käyttöön PDO (PHP Data Objects) – luokka, joka sisältää tuen valmistetuille SQL-lausekkeille (prepared statements), joilla voidaan estää automaattisesti SQL-injektiot. Ideana on, että käyttäjän syötteitä ei aseteta suoraan SQL-lauseeseen, vaan liitetään mukaan erillisellä käskyllä, jolloin syötteiden tarkistusta ei tarvita. (PDO Introduction, hakupäivä 24.3.2012; Prepared statements and stored procedures, hakupäivä 24.3.2012)

```
$dbh = new PDO(/* asetukset */);

$sth = $dbh->prepare("SELECT * FROM tilaukset WHERE tuote = ?");
$sth->bindValue(1, $KÄYTTÄJÄN_SYÖTE);
$sth->execute();
```

Ylhäällä olevassa esimerkissä luodaan aluksi PDO-luokasta olio, jota voidaan käyttää SQL-lauseen valmisteluun. Sen sijaan, että käyttäjän syöte asetetaan suoraan SQL-lauseeseen, jätetään tilalle kysymysmerkki ja käsketään erikseen PDO-luokka asettamaan käyttäjän syöte kysymysmerkin kohdalle. Vaikka hyökkääjä yrittäisi aiheuttaa SQL-injektiota, niin valmistettuun SQL-lausekkeeseen ei tule muutoksia ja sen lähettäminen tietokantaa on turvallista.

6.9.3. Syötteiden tarkistus

Verkkokauppaohjelmistossa otetaan luonnollisesti XSS-hyökkäykset huomioon, sillä ne voivat aiheuttaa suuria ongelmia. Tyypillisesti PHP web-sovelluksissa varmistetaan käyttäjien syötteet htmlspecialchars–funktiolla, joka muuttaa esimerkiksi < ja > -merkit HTML-entiteeteiksi. Muunnoksen ansiosta JavaScriptillä, sekä muilla HTML-elementeillä, ei voi aiheuttaa minkäänlaista XSS-hyökkäystä. Esimerkiksi ”<img src=”jotain”>” merkkijono muuttuu htmlspecialchars–funktiolla ”<img src="jotain">” merkkijonoksi, joten kuva ei lataannu käyttäjän nettiselaimessa. (PHP: htmlspecialchars - Manual, hakupäivä 24.3.2012)

Ongelmana on edelleen se, että jos ohjelmiston kehittäjä unohtaa käyttää esimerkiksi htmlspecialchars–funktiota jossain vaiheessa, niin hyökkääjä voisi aiheuttaa sen kautta XSS-hyökkäyksen. Verkkokauppaohjelmistossa asia hoidettiin järkevämmin, eli kun sivusto ladataan, kaikki GET- ja POST-pyynnöt käydään läpi automaattisesti foreach-silmukassa, jossa jokainen merkkijono käsitellään htmlspecialchars–funktiolla, jolloin asiasta ei tarvitse sen enempää murehtia.

Toinen alue, jossa käyttäjän syöte on syytä tarkistaa, on sähköpostien lähetyksessä. Oli kyseessä PHP:n mail-funktio tai vastaava tapa lähettää viestejä, niin syötteet tulee tarkistaa tai muuten hyökkääjä voi lähettää web-sovelluksella roskapostia. (Mail headers injections with PHP, hakupäivä 24.3.2012) Sähköpostien lähetys on erittäin yksinkertainen prosessi, ohessa on tyypillisen sähköpostin raaka versio:

```
To: vastaanottaja@vastaanottajanosoite.fi
Subject: Viestin otsikko
From: lahetta@lahettajanosoite.fi
```

```
Viestin sisältö.
```

Jos esimerkiksi sähköpostin vastaanottajan syötettä ei tarkisteta, niin hyökkääjä voi lisätä loputtomasti vastaanottajia. Hyökkäys on myös erittäin yksinkertainen, sillä jokaisen sähköpostin loppuun lisätään vain \r\n –merkkijono, joka tarkoittaa uutta riviä. (Mail headers injections with PHP, hakupäivä 24.3.2012)

Jos vastaanottajaksi asetettaisiin vaikka ”vastaanottaja@vastaanottajanosoite.fi\r\nTo: vastaanottaja1@roskaposti.fi”, sähköpostin sisältö muuttuisi seuraavaksi:

```
To: vastaanottaja@vastaanottajanosoite.fi
To: vastaanottaja1@roskaposti.fi
Subject: Viestin otsikko
From: lahetta@lahettajanosoite.fi

Viestin sisältö.
```

Nyt viestillä on kaksi vastaanottajaa yhden sijasta. Hyökkäykset on helppo estää poistamalla kaikki \r ja \n -merkit käyttäjien syötteistä, jotka lisätään sähköpostin ylätunnisteisiin - verkkokauppaohjelmistossa poistetaan luonnollisesti edellä mainitut merkit.

6.9.4. Cross site request forgery esto

CSRF-hyökkäyksiin on osittain vaikea puuttua, mutta onneksi uusimmista nettiselaimesta löytyy tuki X-Frame-Options HTTP-ylätunnisteille, joilla kyseisiä hyökkäyksiä voidaan osittain torjua. Jos selaimelle lähetetään ”X-Frame-Options: SAMEORIGIN” HTTP-ylätunniste, niin nettiselain ei lataa sivustoa, jos se on <frame> tai <iframe> -kehyksien sisällä toisessa sivustossa. Tyypillinen CSRF-hyökkäys toteutetaan juurikin kehyksien avulla, joten X-Frame-Optionsilla voidaan estää iso määrä CSRF-hyökkäyksiä, joihin ei voida normaalisti vaikuttaa muulla tavalla. Verkkokauppaohjelmisto lähettää luonnollisesti X-Frame-Options HTTP-ylätunnisteen joka pyynnön yhteydessä. (The X-Frame-Options response header, hakupäivä 25.3.2012)

Toinen tapa estää ylläpitoon liittyvät CSRF-hyökkäykset on asettaa jokaiselle kirjautuneelle ylläpitäjälle niin sanottu salainen tunniste (token). Ideana tunnisteessa on luoda satunnainen merkkijono ylläpitäjälle, mikä liitetään kaikkiin linkkeihin ja lomakkeisiin, kun sivua rakennetaan. Ainoastaan ylläpitäjän nettiselain ja ylläpitäjä näkee tunnisteen, mutta hyökkääjä, joka yrittää huiputtaa nettiselainta tekemään pyyntöjä, ei näe sitä. Koska hyökkääjä ei näe tunnistetta, web-sovelluksen tarvitsee vain tarkistaa kaikkien pyyntöjen yhteydessä, että onko tunniste lähetetty ja onko se oikein. (Palmer 2008, hakupäivä 25.3.2012)

Verkkokauppaohjelmistossa tarkistus on viety astetta kovemmaksi. Jos ylläpito-osioon tulee pyyntö ilman tunnistetta tai se on väärä, koko ylläpitäjän istunto poistetaan järjestelmästä. Sen sijaan, että annetaan hyökkäysyrityksien jatkua, on parempi, että istunto poistetaan, sillä jokin hyökkäysyrityksistä voi onnistua ennemmin tai myöhemmin.

6.9.5. Salasanojen suojaus

Asiakkaiden ja ylläpitäjien salasanojen suojaus on erittäin tärkeä asia. Esimerkiksi jos jokin ylläpitäjistä päättää varastaa kopion verkkokauppaohjelmiston tietokannasta, salasanojen on pakko olla suojattu. Suurin osa ihmisistä käyttää samaa salasanaa useisiin eri paikkoihin, joten jos salasanojen vuotaminen voi tulla kalliiksi (Password Security 2011, hakupäivä 3.4.2012).

Sen sijaan, että salasanat tallennettaisiin tietokantaan pelkkänä tekstinä, salasanoista muodostetaan kryptografinen tiiviste. Tiivistealgoritmit kehiteltiin alun perin digitaalisiksi sormenjäljiksi, jolloin vastaanotettu viesti tai tiedosto voitaisiin todeta muokkaamattomaksi. Koska tiivistettä ei voida muuntaa takaisin alkuperäiseen muotoon, se sopii hyvin salasanojen suojaukseen. Salasanojen tarkistus on silti helppoa; kun asiakas lähettää haluamansa salasanan, luodaan siitä tiiviste ja tallennetaan se tietokantaan. Kun asiakas kirjautuu sisään, käytetään samaa tiivistealgoritmia syötettyyn salasanaan ja jos tiiviste täsmää tallennetun tiivisteen kanssa, salasana oli oikein. (Rivest 1992, hakupäivä 25.3.2012)

Esimerkiksi MD5-tiivistealgoritmi tuottaa 128-bittisen tiivisteen mistä tahansa syötteestä, esimerkiksi lauseesta ”tämä on opinnäytetyö” syntyy 46236881efe5e96b51b3b01f07d35996 heksadesimaali merkkijono. Jos edellisestä lauseesta yksikin kirjain muuttuu, niin koko tiiviste muuttuu. (The MD5 Message-Digest Algorithm, hakupäivä 25.3.2012)

Ongelmana kuitenkin MD5:ssa ja muissakin tiivisteissä on se, että niitä on helppo ”murtaa” nykypäivän tietokoneilla, etenkin näytönohjaimilla, jotka pystyvät niiden arkkitehtuurin ansiosta suorittamaan tuhansia säikeitä yhtä aikaa (Parallel Programming and Computing Platform | CUDA | NVIDIA, hakupäivä 25.3.2012). Murtamisella tässä tapauksessa tarkoitetaan kaikkien mahdollisien kirjainyhdistelmien kokeilua tiivistealgoritmin kanssa, niin kauan, että tiiviste täsmää tunnetun tiivisteen kanssa.

Esimerkiksi oclHashcat -ohjelma pystyy murtamaan 10,9 miljardia MD5-tiivistettä sekunnissa AMD Radeon HD 6990 näytönohjaimella (oclHashcat-lite - advanced password recovery, hakupäivä 25.3.2012). Pelkän tiivisteen käytöstä muodostuu muitakin ongelmia. Jos kahdella henkilöllä on sama salasana, niin ainoastaan yksi tiiviste täytyy murtaa, sillä hyökkääjä huomaa, että kummallakin henkilöllä on sama salasana. Sateenkaaritaulukot (Rainbow Tables) ovat myös suuri ongelma, sillä ne sisältävät valmiiksi laskettuja tiivisteitä kaikista kirjainyhdistelmistä (Free Rainbow Tables, hakupäivä 25.3.2012).

Ratkaisu kahteen edelliseen ongelmaan on antaa jokaiselle käyttäjälle suola (salt), eli satunnainen merkkijono kirjaimia, jotka syötetään aina mukaan tiivisteeseen, jolloin kaksi samaa salasanaa tuottaa eri tiivisteet. Samalla sateenkaaritaulukot muuttuvat hyödyttömiksi, koska jokaiselle suolalle pitäisi laskea oma taulukko, joka on hidasta. (Hall, Kelsey, Schneier & Wagner 1997, hakupäivä 25.3.2012)

Vaikka nykyään, heikkouksista huolimatta, MD5 on yksi suosituimmista tiivisteistä, verkkokauppaohjelmisto tulee käyttämään uudempaa SHA-512-tiivistealgoritmiä. Etuna SHA-512:ssa on sen hitaus, joten sitä on vaikeampi murtaa, sekä 512-bittinen tiiviste, joka ei ole yhtä heikko kuin MD5:sen 128-bittinen tiiviste. (Jansen 2009, hakupäivä 25.3.2012)

Avaimenvenytys (key stretching) tulee olemaan myös osana salasanojen salausalgoritmia. Ideana avaimenvenytyksessä on toistaa tiivistealgoritmia uudestaan ja uudestaan käyttämällä edellistä tiivistettä mukana. Tapauksissa, joissa tiiviste halutaan murtaa, hyökkääjällä kuluu avaimenvenytyksen ansiosta huomattavasti enemmän aikaa, jolloin hän voi mahdollisesti luovuttaa. (Hall ym. 1997, hakupäivä 25.3.2012) Alla olevassa esimerkissä pyöritetään salasana ja suola yhdistelmää 8192 kertaa SHA-512 algoritmilla:

```
$tiiviste = hash('sha512', $salasana . $suola);

for ($i = 0; $i < 8192; $i++)
{
    $tiiviste = hash('sha512', $tiiviste . $salasana . $suola);
}
```

Koska tiivistealgoritmi toistetaan 8192 kertaa, hyökkääjä joutuu toistamaan saman prosessin, jotta hän saa yhden tiivisteen tarkistettua. Jos hyökkääjän tietokone pystyy esimerkiksi murtamaan 600 miljoonaa SHA-512 tiivistettä sekunnissa, edellisellä avaimenvenytys algoritmilla se pienenee n. 70 tuhanteen yritykseen sekunnissa. Täten suurin osa salasanoista jää murtamatta, sillä niiden käsittelyyn voi kulua miljoonia vuosia.

6.9.6. Istuntojen suojaus

Verkkokauppaohjelmistossa istuntojen suojaus on otettava huomioon, sillä ohjelmisto toimii välikätenä rahan vaihdossa ja verkkokaupan omistaja on vastuussa riskeistä. Kuten Cross site request forgery esto -osiossa mainittiin, sivusto käyttää X-Frame-Options HTTP-ylätunnistetta suojaamaan käyttäjien istuntoja mahdollisilta CSRF-hyökkäyksiltä, mutta se ei ikävä kyllä riitä, sillä istunnot ovat silti vaarassa.

Koska istunnot ovat varastoitu käyttäjien tietokoneiden evästeisiin, niiden suojeleminen ulkopuolisilta hyökkäyksiltä on melkein mahdotonta. Yksi tapa estää evästeiden varastelu JavaScript-hyökkäyksiltä on antaa istunnon evästeille httponly-asetus, joka kertoo nettiselaimelle, että JavaScriptillä ei voi pyytää evästeitä kyseiseltä nettisivulta (Mitigating Cross-site Scripting With HTTP-only Cookies, hakupäivä 20.3.2012). Silti

suurin osa nettiselaimista, httponly-asetuksesta huolimatta, eivät pysty suojaamaan evästeitä JavaScript-hyökkäyksiltä tarpeeksi hyvin (Grossman 2003, hakupäivä 20.3.2012).

Yksinkertaisimmista tavoista suojata istunto on tallentaa käyttäjän IP-osoite kirjautumishetkellä jokaiselle istunnolle palvelimeen. Jos istunnolle tulee pyyntö eri IP-osoitteesta kuin kirjautumishetkellä, niin se yleensä tarkoittaa sitä, että kyseessä voi olla mahdollisesti istunnon kaappaus, eli eväste on kaapattu. Ikävä kyllä kyseinen varmistus on kuitenkin liiankin tehokas, sillä monet Internetin käyttäjät ovat välityspalvelimen takana, jolloin käyttäjien IP-osoite voi vaihtua minä hetkenä hyvänsä. Jos hyökkääjä on myös samassa verkossa kuin uhri, niin hänellä voi olla myös sama IP-osoite, joten IP-osoitteen tarkistus on kyseisessä tapauksessa myös hyödytön.

Koska verkkokauppaohjelmistossa halutaan, että käyttäjillä ei ole mitään ongelmia istuntojen kanssa, IP-osoite tarkistus joudutaan jättämään pois. Käyttäjät eivät ymmärrä miksi heidän istuntonsa poistuvat, joten he syyttävät verkkokauppaa ja omistaja verkkokauppaohjelmistoa, koska hän menettänyt kauppoja. Ainoaksi ratkaisuksi jäi aika, eli jos käyttäjän istunto on yli tunnin vanha, se poistetaan järjestelmästä.

6.9.7. Satunnaisuus

Satunnaisuus tietoturvan kannalta on erittäin tärkeä asia, eli jos käyttäjälle lähetetään uusi salasana tai ylläpitäjälle annetaan tunniste, hyökkääjä ei saa tietää mitä ne mahdollisesti sisältävät. Kaikissa ohjelmointikielissä löytyy tyypillisesti rand-funktio, joka palauttaa satunnaisia positiivisia kokonaislukuja, riippuen minkä siemenen (seed) srand-funktiolle on annettu. Ongelmana kyseisessä funktiossa on, että se ei ole oikeasti satunnainen, vaan arvot, jotka tulostetaan, ovat riippuvaisia siemenestä. Jos srand-funktiolle annetaan siemeneksi numero 10, rand-funktio tulostaa aina samat luvut. (Introduction to Randomness and Random Numbers, hakupäivä 21.3.2012)

Esimerkiksi jos 64-bittisessä Windows 7 käyttöjärjestelmässä PHP 5.3.8:lle annetaan srand-funktiolle luku 10 ja rand-funktiolla tulostetaan luku kolme kertaa, arvoiksi

saadaan aina 16507, 7832 ja 1777. Arvot voivat olla eriä, riippuen PHP:n versiosta ja alustasta, mutta samat luvut toistuvat aina kyseisessä testissä.

Käyttämällä kyseistä tietoa hyväksi, hyökkääjä pystyisi esimerkiksi arvaamaan käyttäjälle annetun uuden salasanan, jos rand-funktiota käytetään sen luomiseen ohjelmistossa. Esimerkiksi suositussa Joomla! PHP web-sovelluksessa on versiosta 1.5.7 alaspäin haavoittuvuus liittyen salasanan uudelleen luomiseen. Hyökkääjä pystyy nollaamaan ylläpitäjän salasanan ja sen jälkeen laskemaan kaikki potentiaaliset uudet salasanat, joista yksi on asetettu ylläpitäjälle. Koska asetettu siemen srand-funktiolle on liian heikko salasanan luontiin, hyökkääjä pystyy murtautumaan järjestelmään kokeilemalla kaikkia eri yhdistelmiä - alle kolmessa tunnissa. (Esser 2008, hakupäivä 20.3.2012)

Verkkokauppaohjelmistossa on otettu kyseinen ongelma huomioon ja luotu satunnaisien lukujen luomiseen erillinen luokka, jota voi käyttää pitkin ohjelmaa. Luokan metodi palauttaa oletuksena 128 tavua, mikä luo 256^{128} eri vaihtoehtoa, mutta käyttäjä voi muokata pituutta tarpeiden mukaan. Metodi yrittää aluksi päästä lukemaan Unixin /dev/urandom tiedostoa, joka sisältää oikeasti satunnaista tietoa, jota hyökkääjä ei pysty arvaamaan (Random(4) - Linux manual page, hakupäivä 21.3.2012).

Luonnollisesti jos /dev/urandom tiedostoa ei pystytä lukemaan, kuten Windows käyttöjärjestelmissä, luokan metodi yrittää lukea mcrypt_create_iv ja openssl_random_pseudo_bytes –funktioiden tarjoamaa satunnaista tietoa. Jos kahta edellistä lisäosaa ei löydy, niin lopuksi ”satunnaista tietoa” luodaan mt_rand-funktiolla, joka ei ole ikävä kyllä tarpeeksi luotettava, mutta sen on kelvattava. Jos ohjelma joutuu käyttämään mt_rand-funktiota, niin etuna kuitenkin on se, että hyökkääjä ei tiedä mikä oli lopulta ohjelman satunnaisuuden lähde.

7. HAKUKONEOPTIMOINTI

Kuten reitin osiossa mainittiin, verkkokauppaohjelmiston osoitteet ovat automaattisesti hakukoneystävällisiä, mutta niitä on myös optimoitu vielä paremmiksi eri ohjaimissa, esimerkiksi tuotteissa. Tyypillisesti verkkokaupoissa osoitteet ovat tyyliin:

www.esimerkki.com/tuote.php?id=50

Kyseinen osoitemuoto ei ole hakukoneystävällinen, sillä se ei kerro hakukonerobotille, saati asiakkaalle mitään. Verkkokauppaohjelmistossa tuotteiden osoitteet ovat muodostettu seuraavan näköisiksi:

www.esimerkki.com/index.php?/p/50/nokia-3310-matkapuhelin/

Tässä tapauksessa osoite sisältää vain olennaisen. Ohjaimen nimeksi valittiin pelkkä p (product), jolloin se on lyhyt, eikä ole ”tiellä”. Tämän jälkeen tulee tuotteen ID-numero, jolla ohjelmisto tunnistaa tuotteen. Viimeisenä on tuotteen nimi, joka on muunnettu osoiteriviä varten automaattisesti hakukoneystävälliseksi. Ideana kyseisessä järjestelyssä on se, että tuotteen nimi on osoitteen kohokohta, sekä hakukoneroboteille kuin asiakkaille.

Koska verkkokauppaohjelmisto käyttää UTF-8 merkistökoodausta, se myös tukee myös muitakin kuin ASCII-merkkejä. Tämä antaa verkkokauppaohjelmiston käyttäjille huomattavan edun ulkomaankaupassa, sillä monet muut kilpailijat eivät tue Unicode-merkkejä heidän tuotteiden osoiterivissä - saati yleensä tuotteiden nimiä ilman kikkailuja. Verkkokauppaohjelmistossa tuotteen nimi voi siis olla vaikka kiinaksi:

www.esimerkki.com/index.php?/p/60/汉语-漢語/

Ikävä kyllä, jos jokin asiakkaista kopioi linkin sähköpostiin tai vastaavaan ohjelmaan, osoitteen Unicode-merkistön merkistökoodaus saattaa muuttua. Sellaisissa tapauksissa linkin loppuosa saattaa ”rikkoutua”, mutta ohjelmisto on koodattu tarkistamaan kyseiset virheet, jolloin käyttäjän nettiselain ohjataan oikeaan tuotteeseen, käyttämällä tuotteen

ID-numeroa. Tämä auttaa myös hakukoneoptimoinnissa, jossa tuotteen osoitteeseen on tullut kirjoitusvirhe, jolloin hakukonerobotti ohjataan aina oikeaan osoitteeseen.

Verkkokauppaohjelmistossa otettiin myös käyttöön Schema.org mikrotiedot (microdata), joka on Googlen, Microsoftin ja Yahoo!:n tuottama palvelu. Mikrotiedot ovat osa HTML5:n ominaisuuksia, joilla voidaan antaa hakukoneille enemmän tietoa sivustosta, eli kuinka jokaista elementtejä voidaan tulkita. Tämän avulla Google ja muut hakukoneet ymmärtävät sivustoa paremmin, mikä voi taata paremman sijoituksen hakutuloksissa. (Schema.org FAQ, hakupäivä 2.4.2012)

Esimerkiksi breadcrumbs mikrotiedolla voidaan muuttaa sivuston osoitteet, jotka näkyvät hakutuloksissa, käyttäjäystävällisemmäksi. Normaalisti tuotteen osoite näkyisi Googlen hakutuloksessa seuraavasti:

www.esimerkki.com/index.php?/p/50/nokia-3310-matkapuhelin/

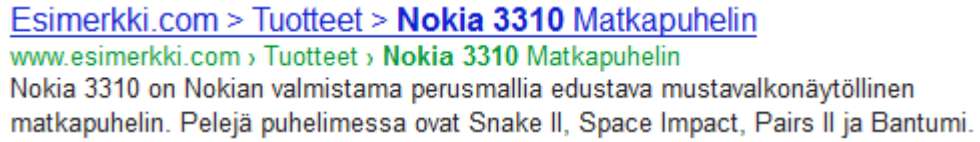
Kyseinen osoite ei ole huono, mutta lisäämällä oikean breadcrumb mikrotiedon, osoite voitaisiin saada näkymään seuraavasti:

www.esimerkki.com > Tuotteet > Nokia 3310 Matkapuhelin

Tulos saadaan lisäämällä alla oleva mikrotieto sivun HTML-dokumenttiin:

```
<div itemprop="breadcrumb">
    <a href="HTTP_OSOITE_TÄHÄN">Tuotteet</a> >
    <a href="HTTP_OSOITE_TÄHÄN">Nokia 3310 Matkapuhelin</a>
</div>
```

Tämä muuttaisi Googlen hakutuloksen kyseiselle sivulle seuraavaksi:



Kuva 19. Googlen hakutulos breadcrumb mikrotiedolla

8. JATKOKEHITYS

Tässä osiossa käsitellään verkkokauppaohjelmiston jatkokehitystä. Suurin osa tulevaisuuden jatkokehityksestä tulee todennäköisesti perustumaan verkkokauppaohjelmiston käyttäjien palautteisiin, mutta suurin osa tämän osion ideoista tullaan kuitenkin lisäämään ohjelman ensimmäiseen julkiseen versioon.

8.1. Koukut

Koukut (hooks) ovat tyypillinen tapa laajentaa web-sovellusten ydintoimintaa ilman, että ohjelmiston tiedostoja tarvitsee muokata. Normaalisti, jos ohjelmistoon lisätään toiminnallisuutta tai olemassa olevia toimintoja halutaan muokata, ohjelman ”ydintiedostoja” joudutaan muokkaamaan. Tämä aiheuttaa sen ongelman, että kun ohjelmasta julkaistaan uusi päivitys, vanhat tiedostot joudutaan korvaamaan uusilla, mitä kautta kaikki muutokset joudutaan lisäämään tiedostoihin uudestaan.

Koukuilla on tarkoitus välttää kyseiset tilanteet. Perus idea on, että lisäosissa luodaan uusi funktio, joka rekisteröidään koukku-järjestelmään. Kun ohjelmisto ajaa koodia, se pysähtyy väliajoin koukku-funktion kohdalle, mikä ajaa kaikki funktiot, jotka on rekisteröity kyseiseen kohtaan. Koukut voivat myös lähettää tietoa takaisin rekisteröidylle funktiolle, jolloin ne pystyvät muokkaamaan ohjelmiston syötteitä ja muita tapahtumia.

8.2. Muistipohjaiset välimuistit

Verkkokauppaohjelmistosta löytyy välimuisti-luokka, jota voi käyttää tallentamaan tietoa kiintolevylle. Esimerkiksi jos tietokannasta pyydetään raskaalla SQL-lausekkeella tietoa, niin kyseistä toimenpidettä ei haluta ajaa uudestaan, joten SQL-lauseen tulos tallennetaan kiintolevylle uudelleenkäyttöä varten. Vaikka kyseinen menetelmä toimii ja nopeuttaa ohjelman suoritusta, sen ylläpitäminen on silti raskasta. Esimerkiksi tiedostot joudutaan parsimaan väliajoin ja poistamaan.

PHP:lle löytyy muistipohjaisia lisäosia, joilla voi tallentaa jaettuun muistiin tietoa, jotka myös säilyivät pyynnöstä toiseen. Ohjelmassa olisi tarkoitus tukea Memcache, APC ja XCache muistipohjaisia lisäosia tulevaisuudessa. Kyseiset lisäosat nopeuttaisivat ohjelmistoa parhaissa tapauksissa 5-10 kertaa nopeammaksi, koska tallennetut tiedot löytyvät muistista, eikä kiintolevyltä (Zaitsev 2006, hakupäivä 2.4.2012). Kyseisissä lisäosissa voi myös asettaa jokaiselle tallennetulle tiedolle ”elinajan”, jolloin se poistetaan automaattisesti muistista. Kyseisiä lisäosia ei kuitenkaan löydy jokaisesta webhotellista, joten se on tarkoitettu keskisuurille yrityksille, jotka omistavat esimerkiksi oman palvelimen, jota kautta he voivat asentaa kyseiset PHP lisäosat.

8.3. Maksu-, posti- ja verojärjestelmä kohdemaan mukaan

Ohjelmistoon on tarkoitus lisätä ominaisuus, joka antaa mahdollisuuden asettaa eri kohdemaille ja osavaltiolle / lääneille eri maksu- ja postitustavat, sekä myös veroluokat. Esimerkiksi Amerikassa eri osavaltiolla on eri veroluokat, jolloin verkkokaupan omistajan pitää pystyä asettamaan oikea vero jokaisen osavaltion mukaan. Suomessa kyseiselle ominaisuudelle ei todennäköisesti löydy paljon käyttöä, mutta ohjelma tullaan julkaisemaan myös englanniksi, joten se on tarpeellinen.

Verkkokauppaohjelmistossa postitustapoja tulee myös pystyä asettamaan eri maihin, sekä myös painon, että ostoskorin yhteishinnan mukaan. Esimerkiksi kansainvälinen verkkokauppa pystyisi asettamaan jokaiseen maahan postitustavaksi kirjaamattoman kirjeen, kunhan paino on alle 2 kg, muissa tapauksissa vain EMS postitus on mahdollista. Ideana olisi myös se, että säännöille voi antaa tärkeysjärjestyksen, jolloin edellisessä esimerkissä kaikki Kiinaan menevät tuotteet voidaan vain lähettää kirjattuna kirjeenä. Myös ostoskorin yhteishinnan voi asettaa kriteeriksi, jolloin esimerkiksi kaikki yli 100 € lähetykset toimitetaan vain EMS lähetyksenä.

8.4. Muita

Alla on esitetty muita pienempiä jatkokehitys ideoita, jotka eivät välttämättä ole kovin tärkeässä roolissa, mutta tulisi sisällyttää ohjelmistoon jossain vaiheessa.

PCI DSS yhteensopivuus

PCI DDS (Payment Card Industry Data Security Standard) on esimerkiksi American Express, Discover, JCB, MasterCard ja Visa luottokortti yrityksien vaatima standardi, joka tarkoittaa, että verkkokauppaohjelmiston ja palvelimen täytyy noudattaa tiettyjä ehtoja, jotta ne saisivat käsitellä, tallentaa tai välittää luottokorttitietoja. Luonnollisesti verkkokauppaohjelmisto ei pysty vaikuttamaan palvelimen vaatimuksiin, mutta PCI DSS vaatii myös ohjelmistolta joitain velvoitteita. Esimerkiksi käyttäjien salasanat pitää olla minimissään 7 merkkiä pitkät ja jos henkilö kirjoittaa tilinsä salasanan väärin 6 kertaa, tili pitää lukita ainakin 30 minuuttia. (Payment Card Industry (PCI) Data Security Standard, hakupäivä 2.4.2012)

Koska Amerikassa luottokorttien käyttö on erittäin suosittua, ohjelmiston täytyy jossain vaiheessa tukea kaikkia PCI DDS:n vaatimuksia, jolloin verkkokaupan omistajat voivat käsitellä luottokortteja.

Kumppanuusohjelma

Pienemmillä yrityksillä on nykyään tarjolla kumppanuusohjelmia, eli kun käyttäjä houkuttelee kauppaan uusia asiakkaita, hän saa siivun houkuttelemansa uuden asiakkaan ostoksista. Kyseisellä tavalla on helppo saada lisää kauppaa pienellä vaivalla, koska verkkokaupan nykyiset asiakkaat tuovat itse lisää uusia maksavia asiakkaita.

Jokainen kumppanuusohjelman asiakas saa itselleen erikoistunnuksen, jonka hän liittää verkkokaupan osoitteiden perään. Kun asiakas tulee kauppaan kyseisen linkin kautta, hänelle asetetaan eväste, joka kertoo kuka houkutteli hänet verkkokauppaan.

Alennuskupongit

Tulevaisuudessa ohjelmistossa pystyy jakamaan asiakkaille alennuskuponkeja, joilla he saavat esimerkiksi 5 % pois tuotteen hinnasta. Kupongeille pystyisi myös asettamaan viimeisen käyttöpäivän, sekä kuinka monta kertaa sitä voi käyttää – tällä vältetään mahdolliset hyväksikäytöt.

Alennuskampanjat

Jokaiselle tuotteelle tai tuoteryhmälle olisi mahdollisuus myös asettaa jossain vaiheessa alennuksia, jotka toimivat tietyn ajan. Ideana olisi myös, että niitä voisi asettaa useampi ennakkoon, jolloin kaupan omistajan ei tarvitse huolehtia jokaisesta kampanjasta erikseen.

PDF-kuitti

Tällä hetkellä verkkokaupasta voi vain tulostaa kuitin sivulta, mutta tulevaisuudessa asiakkailla olisi mahdollisuus ladata erillinen PDF-kuitti omaan käyttöön. Jos verkkokaupan omistaja haluaa, ohjelma voisi myös esimerkiksi lähettää PDF-kuitin automaattisesti jokaisen tilauksen mukana.

Istunnot MySQL tietokantaan

Verkkokauppaohjelmisto käyttää tällä hetkellä PHP:n sisäänrakennettua istunnonhallintaa. Sen ainoa haittapuoli on, että istunnot tallennetaan kiintolevylle, jolloin suosituimmissa sivustoissa se muodostuu pullonkaulaksi. Tulevaisuudessa verkkokauppaohjelmistossa voi valita MySQL-pohjaisen istunnonhallinnan, jossa istunnot tallennetaan taulukkoon, joka käyttää MEMORY tallennusmoottori. Kyseinen tallennusmoottori on kokonaan muistipohjainen, jolloin se on erittäin nopea.

Useampia ylläpitäjiä

Verkkokauppaohjelmisto tukee tällä hetkellä vain yhtä ylläpitäjätiliä, mutta tarkoitus on, että ohjelman julkaisuvaiheessa verkkokaupalla voisi olla useampi ylläpitäjä. Ideana olisi myös se, että jokaiselle ylläpitäjälle voi valita käyttäjäoikeudet eri ylläpito ohjaimiin. Esimerkiksi myyjän tehtävä olisi vain käsitellä tilauksia, joten hänellä ei mitään asiaa verkkokaupan asetuksiin tai lokeihin.

mod_rewrite tuki

Vaikka verkkokauppaohjelmiston osoitteet ovat jo nyt hakukoneystävällisiä, niitä olisi tarkoitus vielä parannella mod_rewrite moduulin tukemisella. Kyseinen moduuli on osa Apache palvelinohjelmistoa, jolla voi kirjoittaa eri sääntöjä osoitteille, jolloin niiden hakukoneystävällisyyttä voidaan parantaa. Esimerkiksi nykyinen osoite

www.esimerkki.com/index.php?/p/50/nokia-3310-matkapuhelin/

Voitaisiin muuttaa seuraavaksi:

www.esimerkki.com/p/50/nokia-3310-matkapuhelin/

Eli osoitteesta kirjoitettiin pois ”index.php?/” osuus. Luonnollisesti ohjelmiston pitää tukea kyseistä ominaisuutta, jotta sitä voidaan käyttää.

Yhden sivun kassa

Suurin osa verkkokaupoista käyttää useamman sivun kassaa, jossa tilauksen tiedot pyydetään asiakkaalta sivu kerrallaan. Esimerkiksi prosessi voisi kulkea seuraavasti: yhteystiedot, maksutapa, postitustapa ja viimeiseksi yhteenveto ennen vahvistusta. Kyseisessä menetelmässä ei ole mitään vikaa, mutta tuoreimpien tutkimuksien mukaan niin sanottu yhden sivun kassa tuo yli 20 % enemmän kauppaa. Kyseisessä kassassa

kaikki tiedot syötetään yhdessä sivussa ja vahvistetaan sen jälkeen, jolloin asiakas ei ehdi katua ostosta. (A/B Test Case Study: Single Page vs. Multi-Step Checkout 2010, hakupäivä 3.4.2012)

9. YHTEENVETO

Sovelluskehyksen suunnittelu ja toteutus onnistui hyvin, minkä ansiosta ohjelmaa on helppo laajentaa perinteisesti MVC-arkkitehtuurin ohjaimilla, sekä ohjelmistoon liitettävillä erillisillä maksu-, postitus- ja widget-lisäosilla. Turvautumalla MVC-arkkitehtuuriin ja Dependency Injection suunnittelumalliin ohjelmiston sovelluskehys toimii luotettavasti ja takaa, että suurin osa kehittäjistä pystyy laajentamaan ohjelmistoa ongelmitta - käyttämällä tunnettuja suunnittelumalleja, ohjelmiston toiminnan ymmärtäminen on myös erittäin helppoa.

Yksi haasteellisimmista osuuksista verkkokauppaohjelmiston sovelluskehyksen suunnittelussa oli sen laajennettavuus. Vaikka sovelluskehys suunniteltiin hyvin, niin silti ohjelmiston joitain osia jouduttiin uudelleen kirjoittamaan muutamaan otteeseen, koska huomattiin, että se ei ollut tarpeeksi joustava. Pienistä vastoinkäymisistä huolimatta, ohjelman sovelluskehys on opinnäytetyön loppuvaiheessa hyvin laajennettavissa. Jatkokehityksen kannalta ohjelman laajennettavuutta tullaan parantamaan koukuilla, jolloin ohjelman ydintoimintoja voidaan muokata ilman, että ohjelman tiedostoja tarvitsee muokata käsin.

Ohjelmiston tietoturvaan keskityttiin olennaisesti. Kaikki käyttäjien syötteet tarkistetaan ja pidetään huoli, että käyttäjä ei voi tehdä mitään sellaista, mitä hän ei normaalisti pystyisi tekemään. SQL-injektioiden varalta PDO-luokan käyttäminen perinteisien SQL-funktioiden sijaan on tietoturvan kannalta erittäin hyvä asia, sillä se takaa, että ohjelmistossa ei vahingossakaan lähetetä tarkastamattomia syötteitä tietokantaan. Kyseisen luokan käyttäminen pitkin ohjelmaa tulee myös pakottamaan muut kehittäjät käyttämään sitä, jolloin korkea tietoturvataso taataan myös lisäosissa. XSS-hyökkäykset on myös puhdistettu automaattisesti ohjelman jokaisessa pyynnössä, jolloin ohjelmistoon ei pääse haitallista HTML- tai JavaScript-koodia.

Ohjelmisto tulee olemaan todennäköisesti kilpailukykyinen Suomen markkinoilla, koska se on täysin ilmainen, julkaistaan suomeksi ja sisältää kaikki tyypillisimmät suomalaiset maksu- ja postitustavat. Käyttäjäystävällinen ylläpitopuoli tulee myös houkuttelemaan ihmisiä siirtymään verkkokauppaohjelmistoon.

Verkkokauppaohjelmistolle annettiin nimeksi **mvC**art, jonka kolme ensimmäistä kirjainta kuvastaa ohjelman **MVC**-arkkitehtuuria – sana cart on perinteisesti melkein jokaisen verkkokauppaohjelmiston nimessä. Ohjelma julkaistaan kesällä 2012 ja sen voi ladata julkaisun jälkeen suomeksi osoitteesta www.mvcart.fi ja englanniksi osoitteesta www.mvcart.com.

10. LÄHDELUETTELO

7.5.3. The MySQL Query Cache. Hakupäivä 19.2.2012.
<http://dev.mysql.com/doc/refman/4.1/en/query-cache.html>

13.6.6.1. The InnoDB Recovery Process. Hakupäivä 20.2.2012.
<http://dev.mysql.com/doc/refman/5.1/en/innodb-recovery.html>

717.6.4. C API Prepared Statements. Hakupäivä 19.2.2012.
<http://dev.mysql.com/doc/refman/4.1/en/c-api-prepared-statements.html>

A Look at MySQL 5.0 Performance Benchmarks 2006. Hakupäivä 19.2.2012.
<http://imysql.cn/system/files/mysql_wp_benchmarks_50.pdf>

A/B Test Case Study: Single Page vs. Multi-Step Checkout 2010. Hakupäivä 3.4.2012.
<http://www.getelastic.com/single-vs-two-page-checkout/>

ASP.NET Platform Requirements. Hakupäivä 12.2.2012.
<http://msdn.microsoft.com/en-us/library/t6dbcb8d(v=vs.71).aspx>

Backward Incompatible Changes. Hakupäivä 25.2.2012.
<http://www.php.net/manual/en/migration53.incompatible.php>

Baker, Marcus & Reierøl, Dagfinn & Shiftlett, Chris 2007. PHP In Action. 1. painos. Manning.

Balling, Derek & Lentz, Arjen & Schwartz, Baron & Tkachenko, Vadim & Zaitsev, Peter & Zawodny, Jeremy. High Performance MySQL. 2. painos. O'Reilly.

Benchmarking PostgreSQL vs. MySQL performance using Drupal 5.x 2007.
Hakupäivä 19.2.2012.
<http://2bits.com/articles/benchmarking-postgresql-vs-mysql-performance-using-drupal-5x.html>

Bergmann, Sebastian 2008. Benchmark of PHP Branches 3.0 through 5.3-CVS.
Hakupäivä 26.2.2012.
<http://sebastian-bergmann.de/archives/745-Benchmark-of-PHP-Branches-3.0-through-5.3-CVS.html>

Bibeault, Bear & Katz, Yehuda 2008. jQuery in Action. 1. painos. Manning.

Chapter 13. Storage Engines. Hakupäivä 18.2.2012.
<http://dev.mysql.com/doc/refman/5.1/en/storage-engines.html>

CPU minutes. Hakupäivä 31.3.2012.
<http://wiki.dreamhost.com/CPU_minutes>

CPU Resource Usage. Hakupäivä 31.3.2012.
<http://support.hostgator.com/articles/pre-sales-policies/rules-terms-of-service/cpu-resource-usage>

Deprecated features in PHP 5.3.x. Hakupäivä 25.2.2012.
<http://www.php.net/manual/en/migration53.deprecated.php>

Dev Shed Forums. Hakupäivä 12.2.2012. <http://forums.devshed.com/>

Dreamhost.com Site Info. Hakupäivä 19.2.2012.
<http://www.alexa.com/siteinfo/dreamhost.com>

Escaping from HTML. Hakupäivä 11.2.2012.
<http://www.php.net/manual/en/language.basic-syntax.phpmode.php>

Esser, Stefan 2008. Joomla Weak Random Password Reset Token Vulnerability. Hakupäivä 20.3.2012.
<http://www.sektioneins.de/en/advisories/advisory-042008-joomla-weak-random-password-reset-token-vulnerability/index.html>

Fawcett, Joe & McPeak, Jeremy & Zakas, Nicholas 2007. Professional Ajax. 2. painos. Wiley Publishing, Inc.

Flanagan, David 2010. jQuery Pocket Reference. 1. painos. O'Reilly.

Flanagan, David 2011. JavaScript: The Definitive Guide. 6. painos. O'Reilly.

Free Rainbow Tables. Hakupäivä 25.3.2012.
<http://www.freerainbowtables.com/en/tables2/>

Garrett, Jesse 2005. Ajax: A New Approach to Web Applications. Hakupäivä 28.2.2012.
<http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications>

Gilmore, Jason 2010. Beginning PHP and MySQL. 4. painos. Apress.

Godaddy.com Site Info. Hakupäivä 25.2.2012.
<http://www.alexa.com/siteinfo/godaddy.com>

Grossman, Jeremiah 2003. Cross-Site Tracing (XST). Hakupäivä 20.3.2012.
<http://www.cgisecurity.com/whitehat-mirror/WhitePaper_screen.pdf>

Hall, Chris & Kelsey, John & Schneier, Bruce & Wagner, David 1997. Secure Applications of Low-Entropy Keys. Hakupäivä 25.3.2012.
<http://www.schneier.com/paper-low-entropy.pdf>

History of PHP. Hakupäivä 11.2.2012.
<http://www.php.net/manual/en/history.php.php>

Hostgator.com Site Info. Hakupäivä 19.2.2012.
<http://www.alexa.com/siteinfo/hostgator.com>

IBM Informix Dynamic Server v10 Information Center. Hakupäivä 20.2.2012.
<http://publib.boulder.ibm.com/infocenter/idshelp/v10/index.jsp?topic=/com.ibm.sqlt.doc/sqltmst159.htm>

Introduction to Randomness and Random Numbers. Hakupäivä 21.3.2012.
<http://www.random.org/randomness/>

Jansen, Jelte 2009. Use of SHA-2 Algorithms with RSA in DNSKEY and RRSIG Resource Records for DNSSEC. Hakupäivä 25.3.2012.
<http://tools.ietf.org/html/rfc5702>

jQuery: The Write Less, Do More, JavaScript Library. Hakupäivä 29.2.2012.
<http://jquery.com/>

jQuery Usage Statistics. Hakupäivä 4.3.2012.
<http://trends.builtwith.com/javascript/JQuery>

Kohavi, R. & Longbotham, R.. Online Experiments: Lessons Learned. Computer 40, no. 9 (2007), 103-105.

Lecky-Thompson, Ed & Myer, Thomas & Nowicki, Steven 2009. Professional PHP6. 1. painos. Wiley Publishing, Inc.

Martin, Adam 2011. LulzSec's Sony Hack Really Was as Simple as It Claimed. Hakupäivä 3.4.2012.
<http://www.theatlanticwire.com/technology/2011/09/lulzsecs-sony-hack-really-was-simple-it-claimed/42851/>

Mackay, Colin 2005. SQL Injection Attacks and Some Tips on How to Prevent Them. Hakupäivä 4.3.2012.
<http://www.codeproject.com/Articles/9378/SQL-Injection-Attacks-and-Some-Tips-on-How-to-Prev>

Magento - System Requirements. Hakupäivä 31.3.2012.
<http://www.magentocommerce.com/system-requirements>

Mail headers injections with PHP. Hakupäivä 24.3.2012.
<http://www.phpsecure.info/v2/article/MailHeadersInject.en.php>

McArthur, Kevin 2008. Pro PHP. 1. painos. Apress.

Mitigating Cross-site Scripting With HTTP-only Cookies. Hakupäivä 20.3.2012.
<http://msdn.microsoft.com/en-us/library/ms533046%28VS.85%29.aspx>

Myer, Thomas & Snyder, Chris & Southwell, Michael 2010. Pro PHP Security. 2. painos. Apress.

MySQL 5.1 Reference Manual :: 10.1.13.1 Unicode Character Sets. Hakupäivä 22.3.2012. <http://dev.mysql.com/doc/refman/5.1/en/charset-unicode-sets.html>

MySQL 5.5: Storage Engine Performance Benchmark for MyISAM and InnoDB 2011. Hakupäivä 20.2.2012. <http://www.oracle.com/partners/en/knowledge-zone/mysql-5-5-innodb-myisam-522945.pdf>

MySQL Market Share. Hakupäivä 18.2.2012. <http://www.mysql.com/why-mysql/marketshare/>

MySQL Performance: MySQL-5.4.0 and other InnoDB engines @dbSTRESS Benchmark 2009. Hakupäivä 19.2.2012. <http://dimitrik.free.fr/db_STRESS_MySQL_540_and_others_Apr2009.html>

Namespaces overview. Hakupäivä 25.2.2012. <http://www.php.net/manual/en/language.namespaces.rationale.php>

News Archive – 2010. Hakupäivä 25.2.2012. <http://www.php.net/archive/2010.php#id2010-12-09-1>

OWASP Top 10 Most Critical Web Application Security Risks 2010. Hakupäivä 3.4.2012. <http://owasptop10.googlecode.com/files/OWASP%20Top%2010%20-%202010.pdf>

oclHashcat-lite - advanced password recovery. Hakupäivä 25.3.2012. <http://hashcat.net/oclhashcat-lite/>

Operating Systems Supported by MySQL Community Server. Hakupäivä 18.2.2012. <http://dev.mysql.com/doc/refman/5.1/en/supported-os.html>

Palmer, Chris 2008. Secure Session Management With Cookies for Web Applications. Hakupäivä 25.3.2012. <http://www.isecpartners.com/files/web-session-management.pdf>

Parallel Programming and Computing Platform | CUDA | NVIDIA. Hakupäivä 25.3.2012. <http://www.nvidia.com/object/cuda_home_new.html>

Password Security 2011. Hakupäivä 3.4.2012.
<https://www.paypal-media.com/assets/pdf/fact_sheet/cis_paypal_whitepaper_final.pdf>

Payment Card Industry (PCI) Data Security Standard. Hakupäivä 2.4.2012.
<https://www.pcisecuritystandards.org/documents/pci_dss_v2.pdf>

PDO Introduction. Hakupäivä 24.3.2012.
<http://www.php.net/manual/en/intro.pdo.php>

Performance Considerations. Hakupäivä 25.2.2012.
<http://php.net/manual/en/features.gc.performance-considerations.php>

PHP 5 ChangeLog. Hakupäivä 25.2.2012.
<http://www.php.net/ChangeLog-5.php#5.3.0>

PHP 5.3 or bust…. Hakupäivä 25.2.2012.
<http://support.godaddy.com/groups/web-hosting/forum/topic/php-5-3-or-bust#post-251897>

PHP and Zend. Hakupäivä 11.2.2012.
<http://www.zend.com/en/community/php/>

PHP: Autoloading Classes – Manual. Hakupäivä 31.3.2012.
<http://www.php.net/manual/en/language.oop5.autoload.php>

PHP Framework benchmark: Zend, CodeIgniter & CakePHP 2009. Hakupäivä 1.4.2012.
<http://leftblank.nl/php-framework-benchmark-zend-codeigniter-cakephp-481.html>

PHP framework comparison benchmarks 2009. Hakupäivä 1.4.2012.
<http://www.phpframeworks.com/news/p/474/php-framework-comparison-benchmarks-2>

PHP: htmlspecialchars – Manual. Hakupäivä 24.3.2012.
<http://php.net/manual/en/function.htmlspecialchars.php>

PHP: Introduction – Manual. Hakupäivä 22.3.2012.
<http://www.php.net/manual/en/intro.mbstring.php>

PHP MVC Framework Performance – Part 1 2010. Hakupäivä 1.4.2012.
<http://www.sheldmandu.com/php/php-mvc-frameworks/php-mvc-framework-performance-part-1>

PHP: Overloading – Manual. Hakupäivä 29.3.2012.
<http://www.php.net/manual/en/language.oop5.overloading.php>

PHP: Strings – Manual. Hakupäivä 22.3.2012.
<http://www.php.net/manual/en/language.types.string.php>

PHP Usage Stats 2007. Hakupäivä 12.2.2012. <http://php.net/usage.php>

PostgreSQL – DreamHost. Hakupäivä 19.2.2012.
<http://wiki.dreamhost.com/PostgreSQL>

PostgreSQL « HostGator.com Support Portal. Hakupäivä 19.2.2012.
<http://support.hostgator.com/articles/specialized-help/postgresql>

Prasanna, Dhanji 2009. Dependency Injection. 1. painos. Manning.

Prepared statements and stored procedures. Hakupäivä 24.3.2012.
<http://www.php.net/manual/en/pdo.prepared-statements.php>

Rain TPL - Speed Test. Hakupäivä 29.3.2012.
<http://www.raintpl.com/PHP-Template-Engines-Speed-Test/>

Random(4) - Linux manual page. Hakupäivä 21.3.2012.
<http://www.kernel.org/doc/man-pages/online/pages/man4/random.4.html>

Rasmus Lerdorf | PHP on Hormones 2007. Hakupäivä 11.2.2012.
<http://itc.conversationsnetwork.org/shows/detail3298.html>

Rivest, Ronald 1992. The MD5 Message-Digest Algorithm. Hakupäivä 25.3.2012.
<http://tools.ietf.org/html/rfc1321>

Schema.org FAQ. Hakupäivä 2.4.2012.
<http://support.google.com/webmasters/bin/answer.py?hl=en&answer=1211158>

Scripts. Hakupäivä 12.2.2012. <http://www.hotscripts.com/category/scripts/>

Shared hosting – WHTwiki. Hakupäivä 31.3.2012.
<http://www.webhostingtalk.com/wiki/Shared_hosting>

Shiflett, Chris 2006. Essential PHP Security. 1. painos. O'Reilly.

Spolsky, Joel 2003. The Absolute Minimum Every Software Developer Absolutely, Positively Must Know About Unicode and Character Sets. Hakupäivä 22.3.2012.
<http://www.joelonsoftware.com/articles/Unicode.html>

Stats – Wordpress.com. Hakupäivä 12.2.2012. <http://en.wordpress.com/stats/>

Stogov, Dmitry 2008. PHP.internals: Re: re2c scanner issue. Hakupäivä 25.2.2012.
<http://news.php.net/php.internals/36484>

Sun to Acquire MySQL 2008. Hakupäivä 19.2.2012.
<http://mysql.com/news-and-events/sun-to-acquire-mysql.html>

Search « Exploits Database by Offensive Security. Hakupäivä 4.3.2012. <http://www.exploit-db.com/search/?action=search&filter_page=1&filter_description=&filter_author=&filter_platform=31&filter_type=0&filter_lang_id=0&filter_exploit_text=&filter_port=0&filter_osvdb=&filter_cve>

The Go Daddy Group. Hakupäivä 25.2.2012. <http://www.inc.com/inc5000/profile/the-go-daddy-group>

The Main Features of MySQL. Hakupäivä 18.2.2012. <http://dev.mysql.com/doc/refman/5.1/en/features.html>

The Model-View-Controller Design Pattern: 'Model 2' JSP Development. Hakupäivä 31.3.2012. <http://ptgmedia.pearsoncmg.com/imprint_downloads/informit/chap2_0672324725.pdf>

The MyISAM Storage Engine. Hakupäivä 19.2.2012. <http://dev.mysql.com/doc/refman/5.5/en/myisam-storage-engine.html>

The PHP License, version 3.01. Hakupäivä 12.2.2012. <http://www.php.net/license/3_01.txt>

The X-Frame-Options response header. Hakupäivä 25.3.2012. <https://developer.mozilla.org/en/The_X-FRAME-OPTIONS_response_header>

Trachtenberg, Adam 2004. Why PHP 5 Rocks!. Hakupäivä 12.2.2012. <http://onlamp.com/pub/a/php/2004/07/15/UpgradePHP5.html>

Usage of JavaScript libraries for websites. Hakupäivä 4.3.2012. <http://w3techs.com/technologies/overview/javascript_library/all>

Usage of server-side programming languages for websites. Hakupäivä 4.3.2012. <http://w3techs.com/technologies/overview/programming_language/all>

What is PHP?. Hakupäivä 11.2.2012. <http://www.php.net/manual/en/intro-whatis.php>

What can PHP do?. Hakupäivä 11.2.2012. <http://www.php.net/manual/en/intro-whatcando.php>

Widenus, Michael 2008. Oops, we did it again (MySQL 5.1 released as GA with crashing bugs). Hakupäivä 19.2.2012. <http://planet.mysql.com/entry/?id=16232>

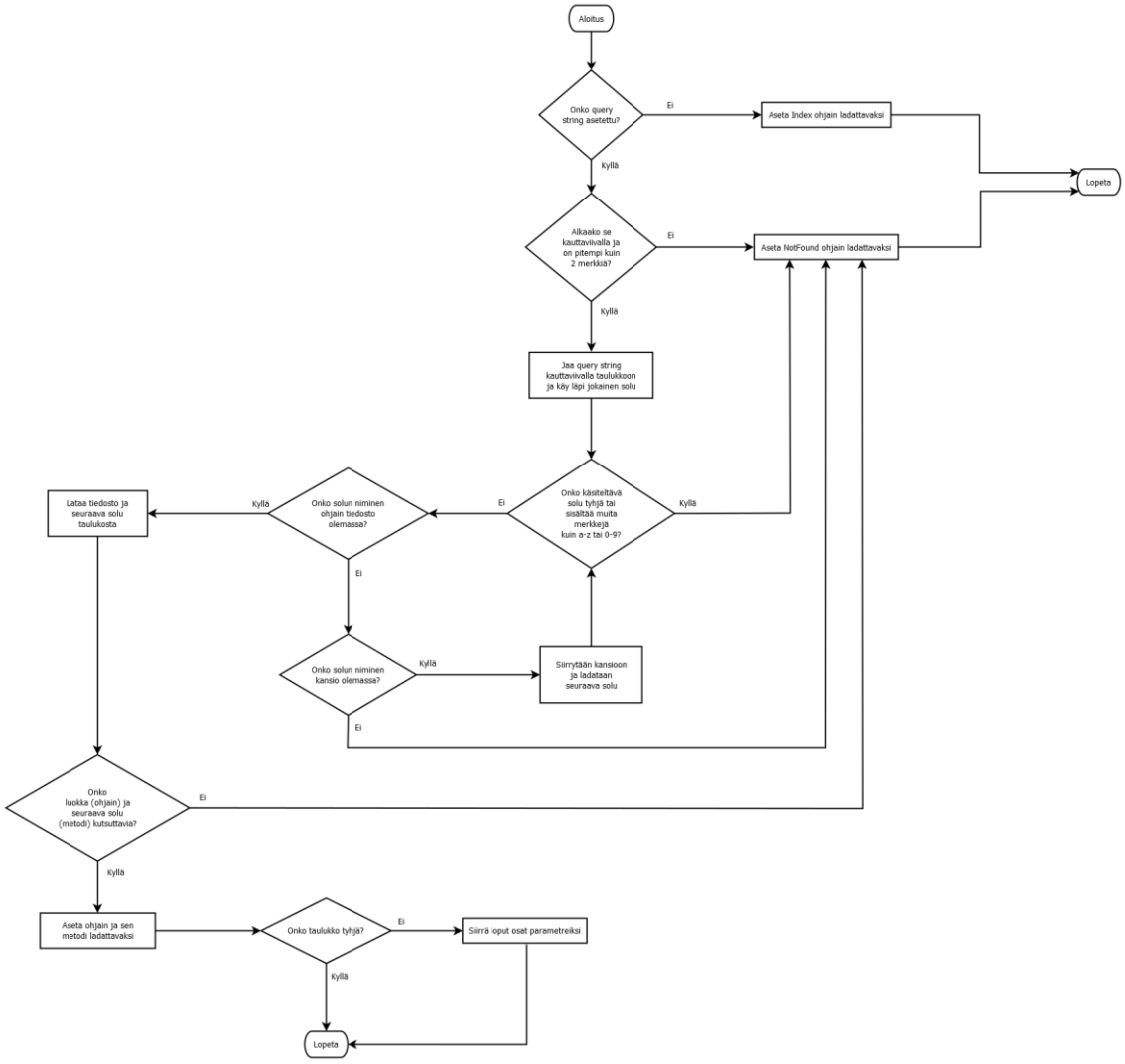
WordPress Download Counter. Hakupäivä 12.2.2012. <http://wordpress.org/download/counter/>

Zaitsev, Peter 2006. Cache Performance Comparison. Hakupäivä 2.4.2012. <http://www.mysqlperformanceblog.com/2006/08/09/cache-performance-comparison/>

Zakas, Nicholas 2010. How many users have JavaScript disabled?. Hakupäivä 28.2.2012. <http://developer.yahoo.com/blogs/ydn/posts/2010/10/how-many-users-have-javascript-disabled/>

Zandstra, Matt 2010. PHP Objects, Patterns, and Practice. 3. painos. Apress.

11. LIITELUETTELO


Aloitus
Onko query string asetettu?
Ei
Aseta Index ohjain ladattavaksi
Kyllä
Lopeta
Alkaako se kauttaviivalla ja on pitempi kuin 2 merkkiä?
Ei
Aseta NotFound ohjain ladattavaksi
Kyllä
Jaa query string kauttaviivalla taulukkoon ja käy läpi jokainen solu
Onko käsiteltävä solu tyhjä tai sisältää muita merkkejä kuin a-z tai 0-9?
Ei
Kyllä
Onko solun niminen ohjain tiedosto olemassa?
Kyllä
Lataa tiedosto ja seuraava solu taulukosta
Ei
Onko solun niminen kansio olemassa?
Kyllä
Siirrytään kansioon ja ladataan seuraava solu
Ei
Onko luokka (ohjain) ja seuraava solu (metodi) kutsuttavia?
Ei
kyllä
Aseta ohjain ja sen metodi ladattavaksi
Onko taulukko tyhjä?
Ei
Siirrä loput oset parametreiksi
Kyllä
Lopeta