OULUN YLIOPISTO
SÄHKÖ- JA TIETOTEKNIIKAN OSASTO
TIETOTEKNIIKAN KOULUTUSOHJELMA

VERKKOKAUPPASOVELLUSALUSTAN TOTEUTUS

Työn tekijä _____
Lassi Heikkinen

Työn valvoja _____
Mika Ylianttila

Hyväksytty _____ /_____ 2010

Arvosana _____

Heikkinen L. (2010) Implementation of E-commerce Platform in English. Department of Electrical and Information Engineering, University of Oulu, Oulu, Finland. Master's thesis, 73 p.

ABSTRACT

Along with the widespreading of the Internet, web stores have attained a remarkable status in trading and the growth is expected to continue. The most significant benefits of web stores compared to physical stores are the independence of time and place along with more effortless comparison of the products.

The subject of this master's thesis is to implement a general e-commerce platform, with which actual web store applications can be customized according to the customers' interests. The primary requirements of the platform are the basic features and functionalities which were decided to include management of products, categories, design and orders along with a simple reporting system in the administration side of the application. An additional requirement for the public side of the application is a demo store, in which products can be browsed by categories, added to shopping cart and orders can be made. More detailed requirements for the application are modern multilingual support, content management system and integration to the most common tracking softwares, such as Google Analytics.

The theory chapter of the work briefly describes World Wide Web, web stores in general and as web applications. The topics are being followed by discussion on Model-View-Control architectural pattern and object-oriented programming paradigm. The state of the art of web store platforms is done by reviewing three applications which currently are on the market: osCommerce, Magento and Interspire Shopping Cart. After the theory chapter the goals of the project are presented in a more detailed way with plans on how to meet the given goals. In this part the characteristics of the project application, such as the pricing logic of products and delivery costs, information security, design management and localization are described.

As the application is, on the time of writing (December 18th 2009), in production use in five web stores, it can be presumed that the main requirements for the work have been achieved. Some of the more detailed features were left unimplemented, such as support for multiple database management systems and an inclusive auditing of administration actions. The work has proven though, that a modern e-commerce platform can be produced in six months.

Keywords: web, MVC, web store, e-commerce platform.

Heikkinen L. (2010) Verkkokauppasovellusalustan toteutus. Oulun yliopisto, sähkö- ja tietotekniikan osasto. Diplomityö, 73 s.

TIIVISTELMÄ

Internetin yleistymisen myötä verkossa toimivat kaupat ovat reilussa vuosikymmennessä saavuttaneet merkittävän aseman kaupankäynnissä ja kasvun odotetaan jatkuvan. Verkkokauppojen huomattavimmat edut perinteisiin fyysisiin kauppoihin verrattuna ovat aika- ja paikkariippumattomuus sekä tuotevertailun vaivattomuus.

Tämän diplomityön aiheena on toteuttaa yleinen verkkokauppasovellusalusta, jonka avulla voidaan räätälöidä varsinaisia verkkokauppasovelluksia. Alustan päävaatimuksina ovat verkkokaupan perusominaisuudet- ja toiminallisuudet, joilla päädyttiin tarkoittamaan tuotteiden ja kategorioiden, ulkoasun ja tilauksien hallintaa sekä yksinkertaisia raportteja sovelluksen ylläpitopuolella. Lisäksi sovelluksen julkisen puolen vaatimuksena on demokauppa, jossa tuotteita voidaan selata kategorioittain, lisätä tuotteita ostoskoriin ja tehdä tilauksia. Yksityiskohtaisimpina vaatimuksina sovellukselle ovat nykyaikainen monikielisyystuki, sisällönhallintajärjestelmä ja integrointi yleisimpiin seurantaohjelmiin, kuten Google Analyticsiin.

Työn teoriaosiossa käsitellään lyhyesti World Wide Webia, verkkokauppaa yleisesti ja web-sovelluksena. Lisäksi tarkastellaan MVC-ohjelmointiarkkitehtuuria (eng. Model-View-Controller) ja olio-ohjelmointiparadigmaa sekä tehdään katsaus verkkokauppasovellusalustojen nykytilaan tarkastelemalla lähemmin kolmea markkinoilla olevaa sovellusta: osCommercea, Magentoa, Interspire Shopping Cartia. Työn käytännön osiossa esitellään tarkemmin projektin tavoitteet ja kuinka ne pyritään saavuttamaan. Osiossa kuvataan sovelluksen tiettyjä piirteitä, kuten tuotteiden ja toimituskulujen hinnoittelulogiikkaa, tietoturvallisuutta, ulkoasun hallintaa ja lokalisointia.

Koska sovellus on kirjoitushetkellä (18. joulukuuta 2009) tuontantokäytössä viidessä eri kaupassa, voitaneen vahvasti olettaa, että työn päävaatimuksessa on onnistuttu. Osa yksityiskohtaisemmista ominaisuuksista jäi toteuttamatta, kuten tuki useille tietokantahallintajärjestelmille ja ylläpitotoimien kattava auditointi. Työ osoittaa kuitenkin, että puolessa vuodessa voidaan kehittää nykyaikainen verkkokauppasovellusalusta.

Avainsanat: web, MVC, verkkokauppa, verkkokauppasovellusalusta.

SISÄLLYSLUETTELO

ABSTRACT

TIIVISTELMÄ

SISÄLLYSLUETTELO

ALKUSANA

LYHENTEIDEN JA MERKKIEN SELITYKSET

1. JOHDANTO **8**

2. TEORIA **10**
2.1. World Wide Web 10
2.2. Web-sovellus 10
2.3. Verkkokauppa 12
2.4. Verkkokauppasovellus 13
2.5. Ohjelmistokehykset 14
2.6. Olio-ohjelmointi 15
2.7. MVC-ohjelmointiarkkitehtuuri 16
2.8. Katsaus verkkokauppasovellusalustojen nykytilaan 18
2.8.1. osCommerce 18
2.8.2. Magento 19
2.8.3. Interspire Shopping Cart 21

3. SUUNNITTELU **22**
3.1. Tavoitteet 22
3.2. Prosessimalli 23
3.3. Tietokantahallintajärjestelmän valinta 23
3.4. Ohjelmointikielen valinta 24
3.5. Ohjelmistokehyksen valinta 24
3.6. Käyttöliittymäkielten valinta 26
3.7. Alustava tietokantaskeema 26

4. TOTEUTUS **27**
4.1. Arkkitehtuuri 27
4.2. Esimerkki palvelupyynnöstä 27
4.3. Tietoturvallisuuden huomioiminen 29
4.4. Lokalisaatio 32
4.5. Ulkoasun hallinta 33
4.6. Hinnoittelulogiikka 34
4.7. Toimituskulujen laskentalogiikka 35
4.8. Tuotteiden näkyvyyskriteerit 37
4.9. Tietokanta-abstraktio 38

5. MITTAUKSET **39**
5.1. Ominaisuusvertailu 39
5.2. Rivimääräanalyysi 40
5.3. Muutokset ohjelmakoodikannassa 42

6. POHDINTA **45**

7. YHTEENVETO **47**

8. LÄHTEET **49**

9. LIITTEET **52**

ALKUSANA

Tämä diplomityö toteutettiin kesän ja syksyn 2009 aikana Koodiviidakko Oy:n toimeksiannosta. Olen kiitollinen Koodiviidakolle mahdollisuudesta toteuttaa diplomityöni.

Haluan kiittää ohjaajaani, Mika Ylianttilaa, opastuksesta ja arvokkaista vinkeistä työn aikana.

Lisäksi haluan osoittaa erityiset kiitokset työni teknisille ohjaajille, Juha-Mikko Ahoselle ja Samuli Tursaalle sekä varsinkin sovelluksen toteuttamisen käytännön ongelmissa auttaneelle Antti Pikkaraiselle. Kiitos koko Koodiviidakon porukalle, että olen saanut työskennellä heidän kanssaan.

Lopuksi haluaisin vielä kiittää perhettäni ja kaikkia kavereitani, jotka ovat tukeneet ja rohkaisseet minua koko opiskelujeni aikana. Viimeisimpänä, mutta tärkeimpänä, haluan kiittää rakasta avopuolisoani, Johannaa, koko sydämestäni hänen tuestaan ja ymmärryksestä opiskelukiireitäni kohtaan. Hän on myös auttanut paljon työn oikeinkirjoituksessa.

Oulu, Suomi 18. helmikuuta 2010

Lassi Heikkinen

LYHENTEIDEN JA MERKKIEN SELITYKSET

Ajax	Asynchronous JavaScript and XML, asynkroninen JavaScript ja XML
ALV	arvonlisävero
CAPTCHA	Completely Automated Public Turing test to tell Computers and Humans Apart, täysin automatisoitu Turingin testi, joka erottaa ihmisen ja tietokoneen
CMS	Content management system, sisällönhallintajärjestelmä
CSS	Cascading Style Sheets, kaskadiset tyyliohjeet
CVV2	Card Verification Value, luottokortin turvakoodi
HTML	HyperText Markup Language, hypertekstin kuvauskieli
HTTP	HyperText Transfer Protocol, hypertekstin siirtoprotokolla
IP	Internet Protocol, TCP/IP-mallin Internet-kerroksen protokolla
ISC	Interspire Shopping Cart -verkkokauppasovellusalusta
MD5	Message-Digest algorithm 5 -tiivistealgoritmi
MVC	Model-View-Controller, malli-näkymä-ohjain
ORM	Object-relational Mapping, oliorelaatiokuvaus
OSC	osCommerce-verkkokauppasovellusalusta
OSL	Open Software License -ohjelmistolisenssi
PDF	Portable Document Format -tiedostoformaatti
PDO	PHP Data Objects -tietokantaohjelmointimalli
PHP	PHP: Hypertext Preprocessor -ohjelmointikieli
RSS	Really Simple Syndication -verkkosyötemuoto
SHA1	Secure Hash Algorithm 1 -tiivistealgoritmi
SQL	Structured Query Language -relaatiotietokantakyselykieli
URI	Uniform Resource Identifier -verkko-osoite
URL	Uniform Resource Locator -verkko-osoite
VoIP	Voice over Internet Protocol -verkkotekniikka
VS	ViidakkoStore-verkkokauppasovellusalusta
WYSIWYG	What You See Is What You Get, mitä näet, sitä saat
WWW	World Wide Web -hypertekstijärjestelmä
XHTML	Extensible Hypertext Markup Language, laajennetty hypertekstin kuvauskieli
XML	Extensible Markup Language, laajennettu kuvauskieli
XSS	Cross-site scripting -tietoturva-aukko

1. JOHDANTO

Internetin yleistymisen myötä verkossa toimivat kaupat ovat reilussa vuosikymmennessä saavuttaneet merkittävän aseman kaupankäynnissä ja kasvun odotetaan jatkuvan. Nopeasti kehittyvä ala on tarjonnut huokuttelevia markkinoita uusille palveluille ja samalla yrittäjille mahdollisuuksia koetella onneaan ja osaamistaan. Edellä kuvatulle kehitykselle ei näy loppua. Verkkokaupankäynnin alalla liikkuu suuria summia rahaa, ja näihin rahavirtoihin on paljon halukkaita pyrkijöitä.

Verkkokaupalla on useita etuja verrattuna fyysiseen kauppaan. Näistä ehkä merkittävin on aika- ja paikkariippumattomuus: asiakas voi tehdä ostoksia verkkokaupassa kotonaan vaikka keskellä yötä. Lisäksi verkkokauppojen tuotteista on helppo etsiä lisätietoa Internetistä ja vertailla eri kauppojen tarjontaa ja hintoja keskenään. Tavallisessa fyysisessä kaupassa nämä tiedot pitäisi yleensä kysyä myyjältä, joka on jäävi opastaja, ja toisekseen tämä tapa on paljon tehottomampi. Verkkokaupoissa kulut on myös usein saatu fyysisiä kauppoja pienemmiksi, jolloin tuotteiden loppuhinnat ovat asiakkaille mahdollisesti edullisempia. Vaikka verkkokaupoilla on myös huonoja puolia verrattuna fyysisiin kauppoihin, kuten tuotteiden testaamiseen ja toimittamiseen liittyvät rajoitteet, niin merkittävien hyvien puolien myötä voidaan melko varmasti ennustaa verkkokauppojen suosion kasvua entisestään, kun Internetin käyttö leviää vanhempiin ikäryhmiin ja kehittyviin maihin.

Diplomityötä suunniteltaessa huomattiin, etteivät ainakaan tällä hetkellä Suomen markkinoilla olevat verkkokauppasovellukset tarjoa kattavasti eri myyjien tarvitsemia ominaisuuksia, kuten nykyaikaista ulkoasun hallintaa, integrointia yleisimpiin seurantaohjelmiin ja suomalaisten maksu- ja toimitustapojen tukea. Tämän myötä nähtiin mahdollisuus kannattavalle liiketoiminnalle toteuttamalla nykyaikainen verkkokauppasovellusalusta. Lisäksi aihealueen käsittely koettiin tarpeelliseksi, koska havaittiin, ettei akateeminen tutkimus ole pysynyt kunnolla verkkokaupankäynnin kehityksen mukana, vaan alaa on luotu pitkälti teollisuudessa ja ihmisten vapaa-ajalla, kuten kehitettäessä monia ilmaisia avoimen lähdekoodin sovelluksia.

Tarkemmin tämän diplomityön aiheena on toteuttaa yleinen verkkokauppasovellusalusta, jonka avulla voidaan räätälöidä varsinaisia verkkokauppasovelluksia. Kyseessä on siis ohjelmistoprojekti. Alustan vaatimuksena on verkkokaupan perusominaisuudet ja -toiminallisuudet, mikä tarkoittaa tuotteiden ja kategorioiden, ulkoasun ja tilauksien hallintaa sekä yksinkertaisia raportteja kaupan ylläpitopuolelle. Vastaavasti julkipuolelle asetetaan tavoitteeksi rakentaa demokauppa, jossa tuotteita voidaan selata kategorioittan, lisätä tuotteita ostoskoriin ja tehdä tilauksia. Tavoitteena on siis, että sovelluksen avulla voidaan perustaa verkkokauppoja, joissa voidaan myydä esimerkiksi hammasharjoja, tietokoneosia tai digitaalisia kuvia. Diplomityön jälkeen sovelluksen kehitystä ollaan suunniteltu jatkettavan.

Sovellus toteutetaan PHP 5 -ohjelmointikielellä käyttäen Liaani Framework -ohjelmistokehystä. Ohjelmointiarkkitehtuurina käytetään MVC:tä (Model-View-Controller) ja ohjelmointiparadigmana olio-ohjelmointia. Sovelluksessa kiinnitetään erityistä huomiota ulkoasun hallintaan, jotta sovellus ei rajottaisi miltään osin kauppojen ulkoasun rakentamista. Lisäksi lokalisointi, tietoturvallisuuden huomioiminen ja esimerkiksi toimituskulujen laskentalogiikka pyritään toteuttamaan nykyajan standardien mukaiseksi.

Diplomityön teoriaosassa käsitellään lyhyesti World Wide Webia ja verkkokauppaa yleisesti sekä web-sovelluksena. Lisäksi tarkastellaan MVC-ohjelmointiarkkitehtuuriaa ja olio-ohjelmointiparadigmaa sekä tehdään katsaus verkkokauppasovellusalustojen nykytilaan tarkastelemalla lähemmin kolmea markkinoilla olevaa sovellusta.

Suunnittelu-luvussa esitellään tarkemmin projektin tavoitteita ja sitä kuinka ne pyritään saavuttamaan. Luvussa perustellaan myös ohjelmistokehyksen sekä muiden valittujen ratkaisujen valintoja. Työn ongelman mahdollisesti ratkaisevan sovelluksen piirteitä tarkastellaan Toteutus-luvussa. Lisäksi Mittaukset-luvussa suoritetaan erillisiä mittauksia ohjelmistolle ja Pohdinta-luvussa analysoidaan, saavutettiinko asetetut tavoitteet ja jos ei, niin miltä osin ja miksi ei. Lisäksi arvioidaan sovelluksen tulevaisuutta.

Sovelluksen kehitykseen on osallistunut allekirjoittaneen lisäksi kaksi kokeneempaa ohjelmoijaa, jotka ovat paitsi ohjanneet ja valvoneet työn etenemistä, myös osallistuneet sen toteuttamiseen. On vaikea eritellä, kuka kehittäjistä on tehnyt mitäkin, koska työosuudet ovat olleet hyvin samanlaisia ja usein myös päällekkäisiä. Suuret suunnittelulinjaukset ovat johtoportaan ja kokeneempien kehittäjien tekemiä, mutta suuri osa pienemmistä ominaisuuksista ja ohjelman toimintalogiikasta allekirjoittaneen. Työn edetessä allekirjoittaneen osuus ja vastuu ovat kasvaneet.

2. TEORIA

Teoria-luvussa käsitellään lyhyesti World Wide Web, määritellään nykyaikainen web-sovellus ja tarkastellaan erityisesti verkkokauppasovellusta. Lisäksi pureudutaan tarkemmin MVC-ohjelmointiarkkitehtuuriin ja olio-ohjelmointiparadigmaan sekä tehdään katsaus verkkokauppasovellusalustojen nykytilaan arvioimalla kolmea markkinoilla olevaa sovellusta.

2.1. World Wide Web

World Wide Web (lyhyemmin WWW tai web) on hajautettu hypertekstitietojärjestelmä, joka sisältää web-sivuja toisiinsa linkittyneinä. Web-sivut voivat sisältää tekstiä, kuvia, ääntä, videota ja muuta multimediaa. Webin keksijänä pidetään englantilaista Tim Berners-Leetä, joka käynnisti yhdessä belgialaisen Robert Cailliaun kanssa webin joulukuussa 1990. [1]

Webiä käytetään usein virheellisesti Internetin synonyymina, mutta tosiasiassa web on vain yksi Internetin palvelumuodoista. Muita ovat esimerkiksi sähköposti, VoIP (Voice over Internet Protocol) ja tiedostonjako.

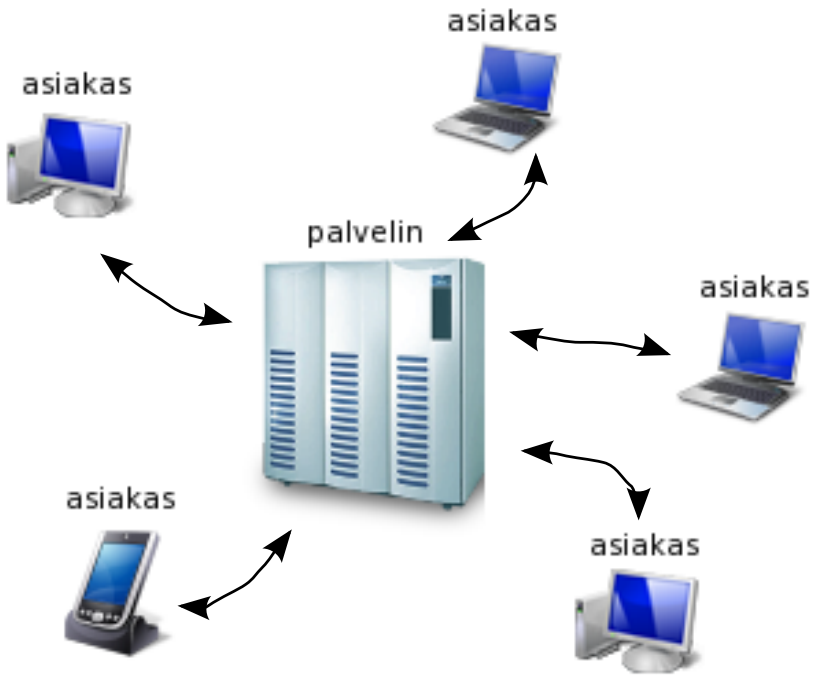
WWW pohjautuu palvelin-asiakas-arkkitehtuurimalliin (kuva 1). Malli määritellään usein ohjelmaprosessiksi, joka on jaettu useammalle prosessointialustalle, jolla prosessoijat pyrkivät yhdessä suorittamaan annetun tehtävän. Webin yhteydessä kyseessä on tyypillisesti erillinen palvelin, joka säilyttää, prosessoi ja jakaa tietoa palvelupyyntöjä tekeville asiakkaille. Tavallisesti yksi asiakas on yhteydessä useisiin palvelimiin ja vastaavasti yksi palvelin palvelee useita asiakkaita. Palvelintietokoneet voivat myös olla samanaikaisesti asiakkaita: esimerkiksi hakukoneet ovat palvelimia web-selaimille, mutta toimivat myös asiakkaina indeksoidessaan web-sivuja. [2, 3]

Asiakkaiden lähettämät palvelupyynnöt ja palvelinten suorittamat vastaukset siirretään tietoverkon yli käyttäen geneeristä HTTP-prokollaa (Hypertext Transfer Protocol). HTTP-protokolla on tilaton ja sen näkökulmasta palvelimet ja asiakkaat vain lähettävät vuorotellen pyyntöjä (esimerkiksi GET, POST, HEAD) ja niihin vastauksia. [4]

Webin alkuhistorian staattisista toisiinsa linkittyneistä fyysikoiden tutkimusraporteista on vajaassa kahdessakymmenessä vuodessa kehittynyt hyvin monimuotoinen verkkomaailma, jossa ihmiset muodostavat kehittyneitä yhteisöjä, jakavat multimediaa lähes rajoituksetta, käyttävät työpöytäohjelmistojen kaltaisia web-sovelluksia, pelaavat pelejä ja tekevät verkkokaupoissa jokapäiväisiä ostoksia. Webistä on tullut lyhyessä ajassa hyvin tärkeä isolle osalle ihmisistä. [5, 6]

2.2. Web-sovellus

Jotta ymmärrettäisiin ero perinteisen staattisen tai osittain dynaamisen sisältökeskeisen web-sivun ja web-sovelluksen välillä, pitää ensin määritellä *sovellus* käsitteenä. Sovellus on ohjelma tai ohjelmisto, jota käyttämällä pyritään ratkaisemaan tietty ongelma. Tavanomainen esimerkki sovelluksesta on tekstinkäsittely- tai taulukkolaskentaohjelma. On huomattava, etteivät kaikki tietokoneohjelmat ole sovelluksia: esimer-



Kuva 1. Palvelin-asiakas-arkkitehtuurimalli

kiksi käyttöjärjestelmällä ei ole tiettyä sovellusaluetta, vaan sen sijaan se mahdollistaa kaikkien muiden ohjelmien toiminnan. [7]

Vastaavasti web-sovellus on sovellus, joka hyödyntää web-teknologiaa. Perinteisesti tämä on tarkoittanut webin palvelin-asiakas-mallin perusasetelmaa eli web-palvelinta ja -selainta. Web-pohjaisuus mahdollistaa sovelluksen käyttöjärjestelmä-, aika- ja paikkariippumattomuuden. Web-sovelluksen ohjelmakoodi sijaitsee tyypillisesti web-palvelimella, josta se voidaan suorittaa käyttäjän omalla web-selaimella mistä ja milloin tahansa.

Baxley (2003) määrittelee web-sovelluksen tehtäväkeskeiseksi web-sivustoksi, joka tunnistaa käyttäjän ja pystyy tallentamaan sekä prosessoimaan käyttäjän syöttämää tietoa. Määritelmä on löyhä, mutta ainakin web-sivustot, jotka sisältävät toiminnallisuutta, logiikkaa ja erilaisia tiloja, täyttävät Baxleyn kriteerit. Lisäksi web-sivustot tyypillisesti sisältävät tällöin kehittynyttä interaktiivisuutta käyttäjien ja sovelluksen välillä. Perinteiset staattiset tai vähän dynaamisuutta sisältävät web-sivut eivät yleensä toteuta näitä piirteitä. [8]

Nykyaikaiset web-sovellukset tarvitsevat tilatietoa toimiakseen. Palvelinten ja asiakkaiden väliseen kommunikointiin käytetty HTTP-protokolla on tilaton, mutta palvelimilla olevat kehitysympäristöt, kuten esimerkiksi web-sovelluksissa yleisesti käytetty PHP (PHP: Hypertext Preprocessor), tarjoavat ratkaisuja, joilla voidaan avata istuntoja käyttäjille. Tällöin palvelimella ajettavalla ohjelmalla on mahdollisuus tallentaa istuntoon liittyviä tietoja helposti ja siten, että ne ovat useista eri HTTP-pyynnöistä huolimatta aina uuden pyynnön saapuessa sovelluksen käytettävissä. PHP:n istunnot perustuvat joko evästeiden tai HTTP-pyynnön mukana kulkevan istuntotunnisteen käyttöön. [9, 10]

2.3. Verkkokauppa

Sähköisessä kaupankäynnissä asiakkaat ostavat tuotteita ja palveluita webissä. Verkkokaupat toimivat sähköisessä kaupankäynnissä virtuaalisena analogiana perinteisille fyysisille kaupoille. Verkkokauppaostosten teossa minimivaatimuksena on Internetyhteydellä varustettu päätelaite: esimerkiksi työpöytätietokone, kannettava tietokone, kämmentietokone tai älypuhelin. Myös verkkopankkitunnukset ja luottokortti ovat hyödyllisiä, koska ilman niitä kauppias joutuu käytännössä toimittamaan tuotteet laskulla tai asiakas noutamaan tuotteet fyysisestä kaupasta. Verkkokauppoja käytetään myös tuotetietojen tarkasteluun ja vertailuun.

Verkkokaupan tavallisessa käyttötapauksessa selataan ja vertaillaan tuotteita yhden tai useamman kaupan kesken. Tarkastelun perusteella valitaan virtuaaliseen ostoskoriin kiinnostavat tuotteet, jonka jälkeen voidaan vielä miettiä lopullista kokonaishintaa, toimitustapaa, -aikaa ja -kustannuksia. Kun ostopäätös on tehty, tuotteet tilataan ja maksetaan, minkä jälkeen jäädään odottamaan tilauksen käsittelyä ja tuotteiden toimitusta.

Asiakkaan näkökulmasta verkkokaupan tärkeimpiä hyviä puolia verrattuna perinteiseen fyysiseen kauppaan ovat aika- ja paikkariippumattomuus: asiakas voi tehdä tilauksia käytännössä missä ja milloin vain. Huomioitavaa on kuitenkin, että usein tuotteen toimitus tapahtuu virka-aikana, jolloin asiakkaan pitää olla itse vastaanottamassa tai noutamassa tuotetta. Sekä yksittäisissä verkkokaupoissa että varsinkin useamman kaupan kesken voi yleensä tavallista kauppaa kattavammin ja tehokkaammin etsiä, vertailla ja tarkastella tuotteita. Joidenkin tuotteiden kohdalla verkkokauppojen hinnat ovat fyysisiä kauppoja edullisempia ja verkkokaupoista pystyy usein ostamaan erikoisempia tavaroita, joita ei välttämättä ole tarjolla oman asuinpaikan lähellä olevissa kaupoissa. [11]

Myyjän näkökulmasta yksi verkkokaupan hyvistä puolista on uuden kaupan helppo perustaminen: lähes kuka tahansa pystyy pienilläkin resursseilla käynnistämään kauppatoiminnan. Varastotyyliset verkkokaupat eivät altistu samalla tavalla myymälävarkauksille kuin perinteiset kaupat. Myöskään myyjän henkilökohtainen ominaisuus, kuten puhevamma tai liikuntarajoite, ei rajoita verkkokaupan pitämistä, kun taas fyysisen kaupan pitoa se voisi haitata.

Verkkokauppaostosten huonoimpia puolia ovat tuotteiden toimituskulut ja -viiveet. Tuotteiden toimitusviive on lyhimmillääkin päivän, pisimmillään ulkomailta tilatessa kuukausia. Tyypillisesti verkkokaupat ilmoittavat tuotteilleen saatavuusarvion, mutta todellisuudessa asiakkaalla ei ole mitään konkreettista varmuutta siitä, että hänen tilaamaansa tuotetta on tai edes tulee olemaan tarjolla. Näin on mahdollista, että myyjä asettaa esille halvalla hinnalla tuotteita, joita hän ei todellisuudessa edes aio myydä. Tällöin tuotteet toimivat niin sanottuina sisäänheittotuotteina, joiden avulla myyjät houkuttelevat asiakkaita kauppaansa.

Virtuaalisessa kaupassa tuotetta ei yleensä pysty kokeilemaan itse, vaan asiakas joutuu turvautumaan myyjän mahdollisesti tarjoamiin kuviin ja muihin multimediaesityksiin. Tämä puute korostuu erityisesti tuotteissa, joissa maku, haju, tunto, koko ja näkö ovat tärkeitä ominaisuuksia. Toisaalta Suomessa on voimassa kuluttajansuojalaki, joka takaa asiakkaan verkkokaupasta ostetulle tuotteelle neljäntoista vuorokauden palautusoikeuden mistä tahansa syystä. Tuotteen palautus on verkkokaupoissa usein tavallisia kauppoja vaivalloisempaa, koska tuote pitää yleensä toimittaa myyjälle pos-

titse. Tuotteissa, joiden arvo ei ole suuri tai joiden arvo suhteessa painoon ei ole suuri, toimituskulujen osuus kauppahinnasta voi olla kohtuuttoman suuri. Tälläiset tuotteet soveltuvat huonosti verkkokaupankäyntiin. [12]

Verkkokauppaostamisessa asiakkaan ja myyjän välinen luottamus on vaikeampi taata: asiakas ei, varsinkaan uuden kaupan kohdalla, voi olla varma myyjän luotettavuudesta. Toisaalta taas asiakkaat voivat tehdä häirikkötilauksia, jotka kuormittavat myyjää turhaan, ja vaikka sekä myyjä että asiakas olisivat luotettavia, niin sähköisessä asioinnissa pahantahtoinen kolmas osapuoli voi onnistua tekemään luottokorttitietotai identiteettivarkauden. Näistä syistä osa käyttäjistä ei halua luovuttaa mitään henkilötietoja myyjille, jolloin yleensä verkko-ostaminen on mahdotonta. Perinteisestä kaupasta pystyy tavallisesti ostamaan ilman minkäänlaisia henkilötietoja, jos kyseessä ei ole erikseen säädelty tuote, kuten alkoholi tai ase.

Jiang (2008) osoittaa asiakkaiden luottamuksen verkkokauppoja kohtaan korreloivan tiedon määrään sähköisestä kaupankäynnistä. Osa ihmisistä asioi mielummin toisen ihmisen kanssa, kun taas osa haluaa tehdä ostoksensa yksin tietokoneen äärellä. Viimeistään tästä syystä sekä perinteisille fyysisille kaupoille että verkkokaupoille tulee jatkossakin olemaan asiakkaita. [13]

2.4. Verkkokauppasovellus

Verkkokauppasovellukset voidaan jakaa toteutustavan mukaan viiteen eri ryhmään. Ensimmäiseen ryhmään kuuluvat yksinkertaiset ohjelmat, joiden ominaisuudet ovat hyvin rajalliset. Tälläisissä ohjelmissa voidaan tyypillisesti lisätä tuotteita ostoskoriin ja lopulta tehdä tilaus, josta lähtee ilmoitus myyjälle. Ratkaisu soveltuu pienen volyymin myymiselle, jolloin erillistä hallintapuolta ei tarvita. Esimerkkinä voisi olla musiikkiyhtyeen virallisten sivujen alasivulla toimiva pienimuotoinen levynmyynti. [14]

Ensimmäisen ja toisen ryhmän verkkokauppasovellusten yhteinen piirre on toteuttamistapa, jossa sovellus kehitetään itse alusta alkaen täysin valmiiksi lopputuotteeksi ilman minkään toisen verkkokauppaohjelman hyödyntämistä. Ensimmäisen ryhmän toteutukset kannattaa usein tehdä tällä tavoin siitä syystä, että yksinkertaiset ratkaisut on usein helpompi toteuttaa tyhjästä kuin jotain olemassaolevaa alustaa käyttäen. Toisen ryhmän verkkokaupat taas tarvitsevat usein paljon yksilöllisiä ominaisuuksia, kuten integrointeja kaupan varaston- ja kassanhallintaohjelmiin, joten valmiiden verkkokauppa-alustojen räätälöinti tulisi työläämmäksi kuin kaiken tekeminen alusta asti itse. [14]

Kolmanteen ryhmään kuuluvat verkkokaupat, jotka pyörivät jossain portaalipalvelussa, joka tarjoaa palvelintilan ja ohjelmiston. Tällöin verkkokaupan pitäjän ei tarvitse huolehtia muusta kuin sisällöstä, mutta toisaalta verkkokauppaan ei myöskään saada pienellä vaivalla uusia ominaisuuksia ja toiminnallisuuksia. Ratkaisu soveltuu erityisesti sellaisille, joilla ei ole tarvetta tai resursseja toteuttaa verkkokauppaohjelmistoa itse. Esimerkki tälläisestä on suomalainen Omaverkkokauppat-palvelu. [14, 15]

Verkkokauppa voidaan toteuttaa myös käyttämällä verkkokauppasovellusalustaa. Alustat sisältävät joukon toiminnallisuuksia ja ominaisuuksia sekä kolmatta ryhmää kattavammat mahdollisuudet kustomoida ohjelma käyttäjän tarpeisiin sopivaksi. Toisaalta tämä tarkoittaa, että kauppaa perustettaessa tarvitaan enemmän resursseja.

Yleensä neljännen ryhmän ratkaisuissa on mahdollista räätälöidä ulkoasut, käytetyt tietokannat, monikielisyystuet ja vastaavat matalan tason ominaisuudet. Verkkokauppasovellusalustat sisältävät tavallisesti jonkinlaisen oletusasennuksen, jolla onnistuu perusverkkokaupan pitäminen. Tunnettuja verkkokauppasovellusalustoja ovat esimerkiksi osCommerce, Magento, Interspire Shopping Cart, Clover Shop, Zen Cart ja VirtueMart. [14]

Ryhmään viisi kuuluvat ratkaisut, jossa hyödynnetään olemassaolevaa verkkokauppasovellusalustaa, mutta sen ohjelmakoodiin tehdään merkittäviä muutoksia. Tälläisessä tapauksessa on laskettu, että toteutustapa on halvempi kuin kokonaan oman ratkaisun toteuttaminen, toisin kuin ryhmän kaksi verkkokaupoissa. Toisaalta kauppa tarvitsee ominaisuuksia, joita ei missään yksittäisessä alustassa ole. Ratkaisun tekee useassa tapauksessa houkuttelevaksi markkinoilla olevat avoimen lähdekoodin ohjelmistot, joiden muokkaus kaupalliseen tarkoitukseen on sallittua. [14]

Baxleyn (2003) määritelmän mukaan web-teknologiaa käyttävä verkkokauppaohjelma ei ole aina web-sovellus. Nykyaikaiset verkkokauppasovellusalustat, joita tässä työssä käsitellään, sisältävät käyttäjäkohtaista personointia ja mahdollistavat tiedon manipuloinnin, joten niitä ja niiden päälle rakennettuja verkkokauppasovelluksia voidaan pitää web-sovelluksina. Sen sijaan ensimmäisen ryhmän ohjelmat eivät Baxleyn määritelmän mukaan ole web-sovelluksia. [8]

2.5. Ohjelmistokehykset

Sovelluksia toteuttaessa käytetään yleisesti valmiita ohjelmistokirjastoja, -kehyksiä ja -alustoja. Nämä ohjelmakoodikokoelmat sisältävät tiettyjä ohjelmallisia työkaluja, joiden tarkoitus on helpottaa varsinaisen sovelluksen kehittämistä. Esimerkiksi voisi olla olemassa oma ohjelmistokirjasto kuvatiedostojen käsittelyyn, jolloin ohjelmoijan ei tarvitse toteuttaa esimerkiksi toiminnallisuutta, joka generoi matalamman resoluution versioita originaalikuvista. Siinä missä ohjelmistokirjastot yleensä tarjoavat apua yhden alueen ongelmaan, niin vastaavasti ohjelmistokehysten tarkoituksena on tarjota kokonaisvaltanen rajapinta koko sovelluksen ohjelmointiin. Ohjelmistokehyksessä voi olla esimerkiksi toteutettuna ominaisuus, joka pakottaa validoimaan kaiken käyttäjän syöttämän datan. [16]

Yksi ohjelmistokehysten keskeisimmistä eduista on ajan säästäminen, kun kaikkia perustoiminallisuuksia, kuten tietokanta-, testi- ja virheenkäsittelyluokkia, ei tarvitse toteuttaa alusta asti itse. Tällöin aikaa säästyy varsinaisen sovelluksen logiikan toteuttamiseen ja testaamiseen. Toisekseen yleisesti käytettyjen ohjelmien ohjelmakoodien laatu on korkeampaa, koska jo useat ihmiset ovat käyttäneet, katselmoineet ja kehittäneet sitä. Ohjelmistokehyksissä on usein huomioitu tietoturvanäkökohtia, joilla pyritään estämään ohjelmoijan kirjoittamia, joko tiedonpuutteesta tai huolimattomuudesta johtuvia, tietoturva-aukkoja sisältäviä ohjelmakoodeja. Lisäksi ohjelmistokehykset tarjoavat hyviä periaatteita ja käytäntöjä esimerkiksi kommentointiin ja muuttujien, luokkien, metodien ja tiedostojen nimeämiseen. Edellä mainittujen asioiden lisäksi ohjelmiston ylläpidettävyyden kannalta suuri merkitys on myös sillä, kuinka helppo uudella ohjelmoijalla on tulla mukaan projektiin. Pääosin näistä syistä ohjelmistokehykset ovat suosittuja kehitettäessä uusia nykyaikaisia sovelluksia. [16]

Ohjelmistokehyksiä kohtaan esitetyssä kritiikissä tuodaan yleensä esille niiden tuottama ylimääräinen laskentatehokuorma, mikä on väistämätöntä, kun alkuperäisen ohjelmointikielen päälle rakennetaan erillinen ohjelmointirajapinta. Tästä syystä esimerkiksi tiukkojen reaaliaikavaatimusten projekteissa voi olla perusteltua olla käyttämättä ohjelmistokehystä. Web-ohjelmoinnissa tiukat aikavaatimukset ovat harvinaisia, joten ohjelmistokehyksiä voidaan käyttää ja käytetäänkin yleisesti web-sovellusten kehityksessä. [16]

Useat nykyaikaiset web-ohjelmistokehykset soveltavat olio-ohjelmoinnin periaatteita sekä pohjautuvat erityiseen MVC-malliin. Näistä tarkemmin luvuissa 2.6 ja 2.7. [17, 18]

MVC-malliin perustuvien sovellusten suunnittelumalli vaihtelee ohjelmistokehyksittäin, mutta joitain yhteisiä piirteitä kuitenkin on olemassa. Toisin kuin muissa PHP-ohjelmissa, MVC-malliin perustuvilla kehyksillä on yksi keskitetty skripti, joka käsittelee kaikki web-sivun palvelupyynnöt. Eli kun tavallisesti selain pyytää web-palvelimella olevaa fyysistä tiedostoa (esimerkiksi `/hakemisto/tiedosto.php`) niin MVC:ssä kysytään ohjelmistokehyksen sisäisen rakenteen mukaan (esimerkiksi `/ohjain/metodi/parametri1/parametri2`). Huomioitavaa on, ettei kyseistä hakemistopolkua ole todellisuudessa olemassa, vaan URI:n (Uniform Resource Identifier) uudelleenkirjoittaja (esimerkiksi `mod_rewrite`) ohjaa kaikki HTTP-kyselyt keskitetylle skriptille, jota kutsutaan esilatausohjelmaksi (engl. bootstrap file). Tämä skripti alustaa ohjelmistokehyksen, lataa tarvittavat tiedostot, huomioi konfiguraatiot, jäsentelee URI:n osiin ja alustaa ohjaimet. URI siis sisältää tiedon ohjaimesta, metodista ja muista parametreistä, joista päätellään, mitä nimenomaiselle palvelupyynnölle tehdään. [19]

2.6. Olio-ohjelmointi

Perinteinen proseduraalinen ohjelmointi on ohjelmointilähestymistapa, joka sisältää listan ohjeita tietokoneelle. Olio-ohjelmoinnissa ohjelmointiongelma sen sijaan ratkaistaan kokoelmalla olioita, jotka tekevät yhteistyötä keskenään. Niitä luodaan luokista, jotka sisältävät eräänlaisen olion valmistusmallin. Luokille määritellään joukko ominaisuuksia ja toimintamalleja (metodeja), jotka oliot saavat syntyessään. Jokainen olio pystyy vastaanottamaan viestejä, käsittelemään tietoja ja lähettämään tietoa muille olioille. [17]

Luokat luodaan hierarkiseen rakenteeseen, jossa alempana hierarkiassa olevat luokat (lapsiluokat) perivät yläpuolellaan olevan luokan (äitiluokan) ominaisuudet ja toimintamallit. Siten ylätasoilla olevat luokat ovat tavallisesti abstraktimpia perusluokkia, kun taas hierarkiassa alemmilla tasoilla olevat luokat ovat erikoistuneempia. Luokkien periytyminen on yksi olio-ohjelmoinnin tärkeimmistä ominaisuuksista, koska se mahdollistaa abstraktit rajapinnat, joiden kautta voidaan käsitellä tuntemattomien tahojen rakentamia olioita ennaltamääritellyllä tavalla. [17, 20]

Olio-ohjelmointi mahdollistaa myös tiedon ja toiminnallisuuksien kapseloinnin eli kokoamisen yhteen yksikköön, olioon. Sen lisäksi olioille pystytään yleensä määrittämään sisäisiä muuttujia, joihin ei päästä käsiksi olion ulkopuolelta, jolloin ohjelmointivirheiden havaitseminen helpottuu. Olioita käytettäessä ohjelmoijan ei tarvitse enää tietää, miten olio toimii sisäisesti, vaan riittää tieto siitä, miten oliota käytetään ja miten

olio käyttäytyy. Kapseloinnilla pyritään helpottamaan ja selkeyttämään yhä monimutkaisemmaksi tullutta ohjelmistojen kehitystä ja ennenkaikkea ylläpitoa. Koska yhden olion sisäinen ohjelmakoodi on tavallisesti lyhyt, on se yleensä helpommin ymmärrettävissä ja ylläpidettävissä. [17, 20]

Web-ohjelmoinnin kannalta 13. heinäkuuta 2004 oli merkittävä, koska PHP 5 -web-ohjelmointikielen olio-ohjelmointituki parani huomattavasti. Tätä aiemmat PHP-versiot on suunniteltu proseduraalisiksi eli niissä onnistuu funktioiden luominen, mutta ei varsinaisesti olioiden käyttö. Tosin PHP 3:ssa on jo alkeellinen tuki olioille ja PHP 4:ssa tukea on parannettu, mutta laajojen ja kompleksisten ohjelmien kehittäminen olioihin pohjautuen on vaikeaa, ellei jopa mahdotonta. PHP 5 toi mukanaan muun muassa jäsenmuuttujat, rajapinnat, abstraktit luokat sekä abstraktit funktiot ja lopulliset (eng. final) luokat ja lopulliset funktiot. Itse asiassa PHP 5:ssä koko olio-ohjelmointituki uudistettiin vaihtamalla ohjelmointikielen ydin käyttämään Zend Engine II:ta. PHP on laajasti käytetty ilmainen, alustariippumaton, avoimen lähdekoodin web-ohjelmointikieli, jota käyttävät niin suuret (esimerkiksi Facebook, Wikipedia, WordPress) kuin pienetkin sivustot. [17, 21, 22, 23, 24]

2.7. MVC-ohjelmointiarkkitehtuuri

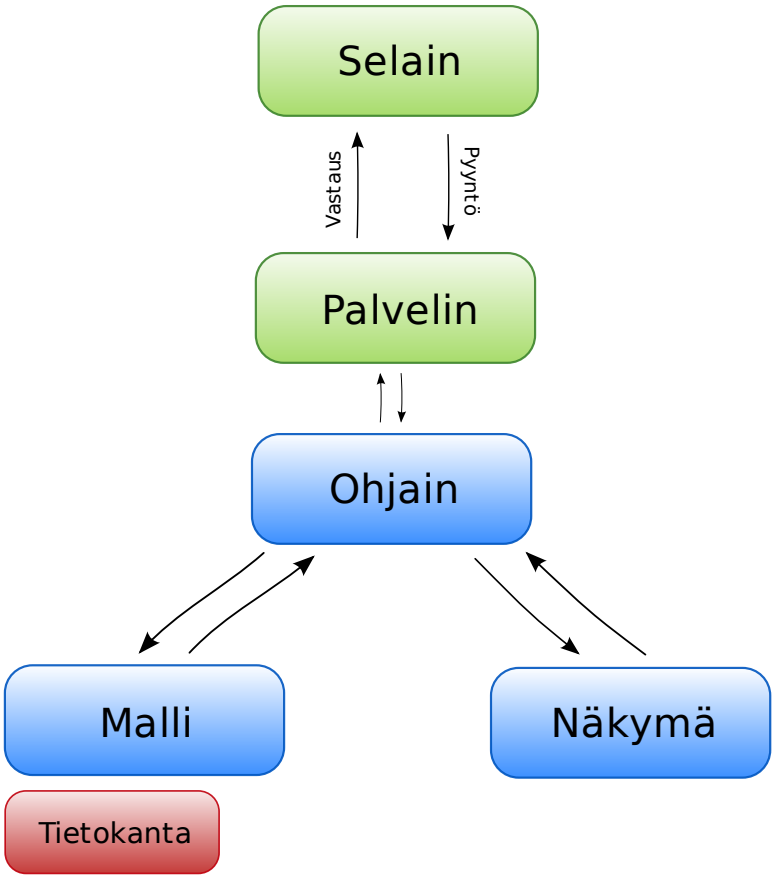
Malli-näkymä-ohjain (engl. Model-View-Controller) on ohjelmistoarkkitehtuurityyli, joka pyrkii selkeyttämään ja yksinkertaistamaan ohjelman kehitystä ja ylläpitoa jakamalla ohjelman kolmeen loogiseen komponenttiin (kuva 2). Tässä luvussa tarkastellaan erityisesti MVC-arkkitehtuurin roolia web-sovelluksissa. [19]

MVC-arkkitehtuurin ensimmäinen komponentti, mallitaso, vastaa järjestelmän sovellusaluekohtaisen tiedon tallentamisesta, ylläpidosta ja käsittelystä. Tyypillisesti mallitaso tarjoaa välineitä esimerkiksi tietokanta-abstraktioon, sähköpostimanipulointiin, tiedon validointiin, autentikointiin ja lokalisaatioon. [19]

Näkymätaso huolehtii vastaavasti siitä, missä muodossa ja miten informaatio välittyy käyttäjälle. Graafinen ilme ja ohjelman käyttöliittymä siis toteutetaan näkymätasolla. Usein käytössä on jokin sivupohjamoottori, jonka avulla käyttäjille pystytään muodostamaan dynaamisesti erilaisia näkymiä. [19]

Kolmas komponentti eli ohjaintaso on sovelluksen varsinainen toimintalogiikka eli se kuinka ohjelman suoritus etenee. Ohjain yhdistää näkymätason tyylittelyn mallitason toiminnallisuuteen ja vastaa syötteiden keräämisestä sekä päättää, miten missäkin tilanteessa toimitaan. Ohjain käyttää hyväksi mallitason tarjoamia välineitä ja tulkitsee mallitason tarjoamaa tietoa näkymälle. Ohjain huolehtii myös ohjelman poikkeustiloista. [19]

MVC-arkkitehtuurin etuja ovat, että malli ei riipu näkymästä eikä ohjaimesta, ja myös näkymän ja ohjaimen riippuvuus toisistaan on vähäistä. Malli voidaan suunnitella, toteuttaa ja testata riippumatta järjestelmän muista osista ja vastaavasti ohjaimen ja näkymän toimintoihin voidaan tehdä muutoksia muuttamatta mallia. Täten mallin tarjoamiin tietovarantoihin voidaan tehdä useita erilaisia käyttöliittymiä. Myös näkymän tai ohjaimen pystyy vaihtamaan verraittain pienillä muutoksilla. Tälläinen arkkitehtuurin modulaarisuus tekee ohjelmakoodista siistimpää ja sen ylläpidettävyys paranee. [19]



Kuva 2. MVC-ohjelmistoarkkitehtuuri

MVC-arkkitehtuurin ongelma on kunkin sovelluskäsitteen jakautuminen tyypillisesti järjestelmän kolmelle eri tasolle. Näin järjestelmästä voi tulla hajanaisempi ja vaikeammin ymmärrettävä. Tämä on ongelma erityisesti silloin, kun projektiin tulee mukaan uusi jäsen, sillä yksinkertaisenkin asian tekeminen voi viedä suhteettoman kauan aikaa ihmiseltä, joka ei tunne järjestelmää. [19]

Ohjelman rakenteen myötä myös sovelluskehitysvastuut voidaan jakaa sujuvasti kolmelle eri roolille. Tästä syystä monet web-alan yritykset suosivat MVC-arkkitehtuuria, koska se mahdollistaa sovellusten tehokkaan kehittämisen. Rooleista ensimmäinen on kehittäjä eli kokeneempi ohjelmoija, joka työskentelee mallitasolla. Hänen tehtävänään on tavallisesti hallita tietokantoja, algoritmejä ja tiedon validointia. Vastaavasti suunnittelijan tehtävänä on huolehtia ohjelman graafisesta ilmeestä ja käyttöliittymästä. Lopulta toinen ohjelmoija yhdistää ohjaintasolla näkymä- ja mallitason. Hänen tehtävänään on myös tyypillisesti purkaa staattinen sivupohja dynaamisiksi osiksi, välittää käyttäjän antamia syötteitä mallitasolle, tulkita mallitason tarjoamia vastauksia ja ohjata ne näkymille. [19]

2.8. Katsaus verkkokauppasovellusalustojen nykytilaan

Verkkokauppasovellusalustojen nykytilaa arvioidaan tekemällä katsaus kolmeen markkinoilla olevaan sovellukseen. Ensimmäiset kaksi, osCommerce ja Magento, ovat avoimen lähdekoodin ohjelmia, joiden käyttäminen on ilmaista. Interspire Shopping Cart taas on suljetun lähdekoodin sovellus, jonka lisenssistä pitää maksaa. Suomalainen Clover Shop Oy julkaisi maaliskuussa 2009 tutkimuksen, jonka perusteella Suomen viisi suosituinta verkkokauppasovellusalustaa ovat osCommerce (22%), Clover Shop (20%), ePages (10%), Workspace (7%) ja VirtueMart (5%). Lukuihin tulee suhtautua varovaisesti, koska Clover Shop Oy:n intresseissä on luonnollisesti saada oma tuotteensa, Clover Shop, näyttämään mahdollisimman suositulta. Voitaneen kuitenkin todeta, että osCommerce on Suomessa yksi suosituimmista verkkokauppasovellusalustoista. Lisäksi vaikuttaa siltä, että Clover Shop on suosituin suomalainen verkkokauppasovellusalusta Suomen markkinoilla. ePagesin suosio perustunee ohjelmaa käyttävään Omaverkkokauppa-palvelun suosioon. [25, 26, 27, 28]

Arvioitavaksi valitut kolme verkkokauppasovellusalustaa perustellaan sillä, että osCommerce on ollut jo pitkään suosituin verkkokauppasovellusalusta niin Suomessa kuin maailmallakin. Magento taas on viime vuosina kasvattanut suosiotaan merkittävästi ja sen avoimen lähdekoodin ohjelmakoodikanta vaikuttaa edistykselliseltä. Kolmanneksi arvioitavaksi ohjelmaksi valittiin aluksi Clover Shop, suosittu suomalainen sovellus, mutta myöhemmin tämä vaihdettiin Interspiren Shopping Cart -ohjelmaan, jonka ratkaisut vaikuttavat Clover Shopia tavoiteltavammilta. Markkinoilla on myös monia muita suosittuja verkkokauppasovellusalustoja, kuten Zen Cart, VirtueMart, CubeCart, GoodBarry, Shopify, X-Cart, Workspace ja ePages, mutta niitä ei tämän työn puitteissa arvioida. [27, 28]

Verkkokauppasovellusalustojen esittelyissä kerrotaan ensin yleisesti ohjelmasta, sen historiasta ja suosiosta tällä hetkellä. Sen jälkeen tarkastellaan sovellusten teknisiä piirteitä sekä nostetaan esiin ohjelmien hyviä ja huonoja puolia sekä käyttäjän että kehittäjän näkökulmasta. Arvioitavien verkkokauppasovellusalustojen versiot ovat osCommerce 2.2 RC 2a, Magento 1.3.2.4 ja Interspire Shopping Cart 5.0.3. Lisäksi katsaukseen on asetettu rajoitus, ettei kiinnitetä huomiota ohjelmien tarjoamiin maksu- ja toimitustapoihin, jotka sinänsä ovat kyllä olennaisia ominaisuuksia verkkokauppasovellusalustalle. Rajoitus on tehty siitä syystä, että arvioitavat kolme sovellusalustaa on tarkoitettu globaaleille markkinoille, kun taas tässä työssä kehitettävä sovellus on tarkoitettu ensisijaisesti kotimaan markkinoille. Sovellusten tarjoamat toimitus- ja maksutapatarjonnat ja -tarpeet eroavat siis suuresti.

Sovelluksia arvioidaan ja verrataan keskenään lähinnä Internetistä löytyvien tietojen ja käyttäjäkokemuskertomusten perusteella. Tämän diplomityön puitteissa ei ole juurikaan tutustuttu omakohtaisesti sovelluksien käyttöön.

2.8.1. osCommerce

osCommerce-projekti (OSC) aloitettiin maaliskuussa 2000 ja se on lisensoitu GNU yleiselle lisensille, mikä tarkoittaa käytännössä, että käyttäjä voi muokata sovelluksen ohjelmakoodia, jakaa ohjelmaa edelleen muille ja jopa myydä ohjelmaa. Tärkein rajoitus on, että ohjelma ja sen muunnellut versiot pitää myös olla GNU yleisen lisens-

sin alaisia. osCommercelle onkin tehty lukuisia muunnelmia, joista ehkä kuuluisin on Zen Cart. osCommercen omien sivujen mukaan (luettu 16. syyskuuta 2009) ohjelmaa käyttää 12162 verkkokauppaa. Luku on tosin varovainen arvio, koska verkkokauppojen käyttäjät pystyvät itse päättämään, haluavatko he laittaa oman verkkokauppansa osCommercen asiakaslistalle. [29, 25]

osCommerce:n ohjelmakoodi on kirjoitettu PHP 3:n ja PHP 4:n aikaan ja ohjelma edelleen tukee näitä versioita. Myös PHP 5 on tuettu lukuunottamatta osaa kolmannen osapuolen moduuleista. Kuten versiotuestakin voidaan päätellä, osCommercen ohjelmointiparadigma on proseduraalinen, toisin kuin kahden muun arvioitavan verkkokauppasovellusalustan. osCommerce käyttää tietokantahallintajärjestelmänään MySQL:aa, joka on tuettuna versiosta 3 ylöspäin. Sekä HTML-merkintäkieli (Hypertext Markup Language) että suorat tietokantakyselyt on kirjoitettu ohjelmalogiikan sekaan, eli näkymiä ei ole erotettu toimintalogiikasta, kuten esimerkiksi MVC-ohjelmistoarkkitehtuurissa. [25]

Sekä hyvät että huonot puolet osCommercen osalta liittyvät sen verrattain korkeaan ikään. Koska osCommerce on ollut markkinoilla jo lähes kymmenen vuotta, sen ympärille on kerääntynyt suuri joukko käyttäjiä, jotka ovat kaupoillaan testanneet ohjelmaa, kehittäneet siihen uusia ominaisuuksia ja kysyneet apua käyttöön liittyvissä kysymyksissä Internetin keskustelupalstoilla. Näin osCommercen peruskoodikanta on kehittynyt kypsäksi, eli se sisältää verrattain vähän ohjelmavirheitä perustoiminnoissaan. Lisäksi lukemattomat käyttäjien kehittämät lisämoduulit tarjoavat hyvin kattavasti erilaisia lisäominaisuuksia osCommercen peruspakettiin. Uusien kauppojen perustajat myös löytävät varmasti useimpiin ongelmiinsa ratkaisut Internetin hakukoneilla, koska samoja ongelmia ovat todennäköisesti ratkoneet aiemmin myös monet muut käyttäjät. [25]

osCommercen suurimmat ongelmat liittyvät ohjelmiston vanhentuneeseen arkkitehtuuriin ja puutteelliseen ulkoasun hallintaan, jota ei oikeastaan ole. Ulkoasua voidaan toki muuttaa vaihtamalla kuvia ja CSS-tyylejä (Cascading Style Sheets), mutta jos halutaan muuttaa myös kaupan HTML-merkintää, niin se pitää tehdä suoraan osCommercen ohjelmakooditiedostoihin. Lisäksi osCommerce ei ole modulaarinen, joten siihen ei pystytä kehittämään uusia ominaisuuksia koskematta ytimen koodikantaan. Tämä tarkoittaa, että sekä muutettaessa HTML-merkintää että asennettaessa käyttäjien tekemiä lisäominaisuuksia hankaloitetaan olennaisesti ohjelmapäivityksiä, koska päivitysten tuomat muutokset pitää tarkastaa yksitellen, jotta saataisiin selville, miten ne vaikuttavat oman kauppa-asennuksen ohjelmakoodeihin. Tämän tekemiseen vaaditaan ohjelmointiosaamista, ja sen lisäksi työ on usein aikaavievää. Tosin modulaariselle ohjelmistoarkkitehtuurille on yleistä sen vaikeampi ymmärrettävyys kokenemattomille ohjelmoijille, joten jos kauppaan tarvitaan vain pieniä muutoksia, voi osCommercen arkkitehtuuri olla joillekin kehittäjille modernimpia sovelluksia helpompi vaihtoehto.

2.8.2. Magento

Magenton kehittäminen aloitettiin tammikuussa 2007 ja siitä on olemassa sekä ilmainen että kaupallinen versio. Tässä työssä keskitytään ainoastaan ilmaiseen, OSL v. 3.0 (Open Software License) lisenssin versioon. OSL:n isoin ero GNU yleiseen lisenssiin on patentoinnin estäminen käytännössä, sillä hyväksymällä OSL v 3.0 lisenssin käyt-

täjä sitoutuu ohjelman käytön lopettamiseen, jos ohjelmaa kohtaan suorittaa minkäänlaisia patentoimistoimia. Magenton kotisivut eivät kerro ohjelmaa käyttävien verkkokauppojen lukumäärää, mutta Google Trendsin mukaan Magenton suosio on ohittanut osCommercen vuoden 2009 alkupuoliskolla. [30, 31]

osCommercea modernimpi Magento on kirjoitettu PHP 5:n kehittynyttä olio-ohjelmointitukea hyväksi käyttäen. Magento hyödyntää Zend-ohjelmistokehystä, joka taas soveltaa MVC-ohjelmistoarkkitehtuuria. Magento hyväksyy tietokantahallintajärjestelmäksi MySQL:n version 4.1.20 tai uudemman. Mageton erillisellä sivupohjien hallintajärjestelmällä ulkoasun muokkaus on helpompaa, koska muutoksia ei tarvitse tehdä ohjelmalogiikan sekaan, toisin kuin esimerkiksi osCommercessa. Kuten MVC-ohjelmistoarkkitehtuurin tavoite on, Magenton ohjelmakoodien keskeinen ominaispiirre on modulaarisuus eli tiedon ja toiminnallisuuksien kapselointi. Kapseloinnilla pyritään helpottamaan ja selkeyttämään ohjelmistojen kehitystä ja ennen kaikkea niiden ylläpitoa. [26, 32]

Magenton hyvät ja huonot puolet liittyvät sen ohjelmistoarkkitehtuuriratkaisuun, joka on hyvin modulaarinen. Toisaalta se mahdollistaa ohjelmiston jatkokehittämisen ja räätälöinnin hyvin pitkälle, mutta vastaavasti ohjelmakoodin moderni arkkitehtuuri voi olla kokemattomille kehittäjille monimutkaisuudessaan ylivoimainen. Magento sisältää esimerkiksi noin kymmenen kertaa enemmän tiedostoja verrattuna osCommerceen ja edelleen kolme kertaa enemmän kuin Interspire Shopping Cart (taulukko 2). Magenton puolustukseksi on kuitenkin todettava, että se sisältää kahta muuta arvioitavana olevaa verkkokauppasovellusalustaa enemmän ominaisuuksia (liite 1). Useat käyttäjät ovat raportoineet, että Magenton suorituskyky on muita verkkokauppasovellusalustoja heikompi, mikä johtunee juuri arkkitehtuurin modulaarisuudesta, joka usein vaatii palvelimelta enemmän resursseja. Ohjelmiston kypsyessä sen ongelmia korjataan ja tämän myötä myös suorituskyky luultavasti paranee. Viimeistään palvelinten suorituskyvyn kehittyminen vuosien myötä helpottaa raskaiden web-sovellusten käyttöä.

Toisin kuin osCommerceen, lisäominaisuuksien toteuttaminen Magentoon tapahtuu käyttäen erityisiä liitännäisosia (eng. plugin), joilla sovellusta voidaan laajentaa koskematta peruskoodikantaan. Tällöin ohjelmiston päivitettävyyttä ei hankaloiteta, koska ohjelmiston lisäominaisuudet ovat eroteltuna perustoiminnallisuuksista kapseloinnin ja periytymisen avulla. Magentossa, kuten useissa muissakin moderneissa web-sovelluksissa, ulkoasuhallinta erotetaan ohjelman varsinaisesta toimintalogiikasta, jolloin uusia sivustoja (kuten verkkokauppoja) toteuttaessa ei sivuston rakentajan (eng. site builder) tarvitse olla tekemisissä ohjelmakoodien kanssa, vaan hän voi keskittyä HTML-merkinnän, CSS-tyylien, grafiikoiden ja muiden tämän tason tekniikoiden toteuttamiseen.

Magenton dokumentaatio on puutteellinen, mitä alati kasvava käyttäjäkunta pitääkin yhtenä Magenton pahimpana heikkoutena. Sekä osCommercella että Interspire Shopping Cartilla on tarjolla kattavat dokumentaatiot, jotka helpottavat niin kauppojen ylläpitäjien kuin sovelluskehittäjienkin työtä. Sen lisäksi, että Magenton ohjelmakoodi on monia muita verkkokauppasovellusalustoja vaikeampi ymmärtää, sille ei myöskään tarjota kattavia oppaita oppimisen ja kehittämisen tueksi. [25, 26, 33]

2.8.3. Interspire Shopping Cart

Interspire Shopping Cart (ISC) julkaistiin vuoden 2007 lopulla ja yrityksen kotisivuilla mainitaan (luettu 29. lokakuuta 2009), että ohjelmaa käyttää 5200 kauppaa. ISC on maksullinen ohjelma, joka tarjoaa käyttäjilleen asiakastukea. Toisaalta sen suosio Google Trendsissä on vaatimaton verrattuna Magentoon ja osCommercen, joiden käyttäjät neuvovat toisiaan kaikkialla Internetissä, ja kasvattavat näin käyttämiensä ohjelmien näkyvyyttä Googlessa. [33, 31]

Myös Interspire Shopping Cart on ohjelmoitu PHP 5:lle. Tietokantahallintajärjestelmänään se käyttää MySQL:n versiota 4.1 tai uudempaa. Sovellus ei käytä mitään kolmannen osapuolen ohjelmistokehystä, ja se hyödyntää osittain olio-ohjelmointiparadigmaa. Tietyt ISC:n ominaispiirteet kielivät hieman vanhentuneesta ohjelmistoarkkitehtuurifilosofiasta, sillä esimerkiksi globaaleja muuttujia käytetään tiedon kuljettamiseen, ohjelmavuota keskeytetään `die()` ja `exit()` -funktioilla ja tietoa ja toiminnallisuutta ei ole kapseloitu niin paljon kuin olio-ohjelmointi PHP:ssa mahdollistaisi. [33]

Interspire Shopping Cartin maksullisuus karsii varmasti ison osan käyttäjistä. Sovellus on myös uusin arvioitavista kolmesta verkkokauppasovellusalustasta. Näistä syistä sovelluksesta löytyy vähiten tietoa Internetistä. Toisaalta ISC:n omat dokumentaatiot ovat hyvin kattavia. Kun Magenton idealistinen ohjelmistoarkkitehtuuri on osoittautunut käytännössä osittain puutteelliseksi lähestymistavaksi, niin ISC:n ohjelmakoodissa yritetään tehdä kompromissejä modulaarisuuden ja selkeyden välillä. Tässä onnistumisen arviointi on vaikeaa, koska Magento toimii eri markkinasekvenssissä ilmaisella lisenssillään eli suhteellisen harva käyttäjä pohtii valintaa juuri näiden kahden sovelluksen välillä ja tämän myötä Internetissä on vähän käyttäjäkokemuksia, joissa vertailtaisiin juuri näitä sovelluksia. [25, 26, 33]

3. SUUNNITTELU

Suunnittelu-luvussa esitellään työssä toteutettavalle verkkokauppasovellusalustalle asetetut tavoitteet ja perustellaan valittu prosessimalli, ohjelmointikieli ja tietokannan hallintajärjestelmä sekä pureudutaan lähemmin ohjelmistokehyksen valintaan. Lisäksi esitellään alustava tietokantaskeema.

Tässä työssä käytettävälle ohjelmistonkehittämisprosessimallille on tyypillistä, että suunnitteluun ei käytetä alussa paljoa resursseja, vaan kehittäminen tapahtuu asiakaslähtöisesti: ohjelmistoa kehitetään sitä mukaa kuin asiakkaat tarvitsevat uusia ominaisuuksia. Tästä syystä projektin alussa ei tehty esimerkiksi tarkempia luokkakaavioita tai arkkitehtuuri- ja tietokantasuunnitelmia, vaan lähdettiin suoraan toteuttamaan verkkokauppasovellusalustan perusominaisuuksia ja -toiminnallisuuksia.

3.1. Tavoitteet

Tämän diplomityön tavoitteena on kehittää verkkokauppasovellusalusta perusominaisuuksilla ja -toiminnallisuuksilla. Kuten useissa sovelluksissa, niin verkkokaupoissakin on tavallisesti ylläpito- ja julkipuoli. Ylläpitopuolen perusominaisuuksiksi päädyttiin valitsemaan tuotteiden ja kategorioiden, ulkoasun ja tilauksien hallinnan sekä yksinkertaisten raportointien toteutus. Tilausten hallinta sisältää myös toimitus- ja maksutapojen hallinnan. Lisäksi voidaan harkita käyttäjähallinnan ja käyttäjäryhmien toteuttamista. Julkiselle puolelle rakennetaan demokauppa, jossa tuotteita voidaan selata kategorioittain, lisätä tuotteita ostoskoriin ja tehdä tilauksia. Julkipuoli tarkoittaa sovelluksen osaa, jota kaupan tavalliset asiakkaat käyttävät ja näkevät. Sen sijaan ylläpitopuoli on tarkoitettu kaupan omistajille, myyjille ja tilausten käsittelijöille.

Lisäksi asetetaan tavoitteeksi joukko erikoisempia ominaisuuksia, joita ei voida ajatella kuuluvaksi verkkokauppasovellusalustan peruspakettiin. Kehitettävässä verkkokauppasovellusalustassa tulee olla monikielisyystuki, joka ei saa perustua käännöskuviin, ja oletustekstejä täytyy myös voida vaihtaa. Verkkokauppa-alustassa tulee olla yksinkertainen sisällönhallinta, jonka avulla käyttäjät voivat luoda vapaata tekstiä sisältäviä sivuja. Sovellukseen dokumentoidaan rajapinta, jonka avulla voidaan integroida markkinoilla olevia seurantaohjelmia. Tavoite on toteuttaa integrointi valmiiksi Google Analyticsiin ja Snoobiin. Verkkokauppaan voidaan tuoda sisältöä toisista verkkokauppasovelluksista, kuten Clover Shopista ja osCommercesta. Lisäksi dokumentoidaan rajapinta muita tuonteja varten. Verkkokauppa-alusta auditoi vierailijoiden toimet lokitiedostoon, jota voidaan analysoida myöhemmin.

Sovelluksesta halutaan myös aidosti eriytetyt paketit, jotta eri asiakkaille voidaan myydä samaa ohjelmaa eri ominaisuuksilla. Tarjolla olisi esimerkiksi kevyt, normaali ja kattava versio. Paketoinnissa ja ohjelmakoodin jakelussa on otettava huomioon, ettei suppeamman version ostaja voi ottaa käyttöön ominaisuuksia, joita hän ei ole ostanut. Tämän takia ominaisuudet täytyy olla irrotettavissa ohjelmakoodista helposti. Toiminnallisuuksien erottelua voidaan harkita toteutettavaksi joko periytymisellä tai liitännäisosilla (eng. plugin) erikoistaen ja laajentaen. Sovellusalusta tukee aluksi MySQL 5, PostgreSQL 8 ja myöhemmin mahdollisesti MS SQL Server -tietokantahallintajärjestelmiä.

Pitkän tähtäimen tavoite on täyttää tällä hetkellä markkinoilla olevien verkkokauppasovellusalustojen puutteet ja palvella parhaalla mahdollisella tavalla mainostoimistoja ja niiden tarpeita. Verkkokauppasovellusalustan pitää olla yksinkertainen asentaa ja ottaa käyttöön tavallisissa web-hotelleissa. Lisäksi tavoitteena on, että ohjelmakoodi olisi myös projektin ulkopuolisten kehittäjien nopeasti ymmärrettävissä.

3.2. Prosessimalli

Projektin ohjelmistonkehitysprosessimallina käytetään ketteriä menetelmiä ja niistä erityisesti Scrumia. Ketterässä ohjelmistokehityksessä painotetaan yhteistyötä asiakkaiden kanssa, muutokseen reagoimista, toimivan sovelluksen kehittämistä ja vuorovaikutusta kehitystyöryhmän sisällä. Vastaavasti tällöin kokonaisvaltaisten suunnitelmien ja dokumentaatioiden tekeminen jää vähemmälle. Tämä ei tarkoita, etteikö suunnittelua tehtäisi, vaan päinvastoin sitä tehdään jatkuvasti koko projektin ajan. Suunnittelmia vain muutetaan helpommin kuin perinteisissä prosessimalleissa. [34]

Erityisesti Scrumissa kehitystyö tapahtuu noin kuukauden mittaisissa pyrähdyksissä, joiden lopuksi työstetään valmis ohjelmisto asiakkaalle tai sovellusta kehittävän tahon sisäiseen käyttöön. Jokainen pyrähdys sisältää kaikki uusien ominaisuuksien julkaisemiseen tarvittavat tehtävät: projektisuunnittelu, vaatimusmäärittely, ohjelmistosuunnittelu, toteutus, testaus ja dokumentointi. Tyypillisesti pyrähdyksen alussa työryhmä sopii yhdessä, mitä ominaisuuksia ja muutoksia työlistalta (eng. backlog) seuraavaksi toteutetaan, kuka toteuttaa ja millä aikataululla. Tarpeet uusille ominaisuuksille pyritään saamaan asiakkailta, joilla osa kehitystyön kuluista voidaan tällöin maksattaa. [34]

Ketterien menetelmien erinomaisuus on nopea reagoiminen muutoksiin. Jokaisen pyrähdysten lopussa tuote saadaan asiakkaalle testattavaksi, jolloin kehitystyöryhmän ja asiakkaan välinen vuorovaikutus paranee. Asiakas näkee jo aikaisemmassa vaiheessa, mitä haluaa tehtävän toisin, minkä myötä kehitystyöryhmä voi muuttaa suunnitelmia. Näin kehitystyön edetessä saadaan asiakkailta jatkuvasti palautetta, johon voidaan prosessissa nopeasti reagoida. Perinteisemmissä malleissa tuote saadaan esille vasta projektin lopussa, ja siinä vaiheessa suunnitteluvirheet tulevat kalliimmiksi. [34]

3.3. Tietokantahallintajärjestelmän valinta

Tavoitteissa asetetaan vaatimukseksi tuki MySQL 5 ja PostgreSQL 8 -tietokantahallintajärjestelmille. Näistä valitaan primaariksi MySQL 5, joka on ensinnäkin tutumpi sovelluksen pääkehittäjälle, mutta tällä hetkellä myös Google Trendsin mukaan tunnetumpi - joskin laskevalla trendillä. Myös kehitystyöryhmän oman käsityksen mukaan se on web-hotelleissa yleisimmin tuettu tietokantahallintajärjestelmä. PostgreSQL otetaan mukaan rinnalle myöhemmässä vaiheessa. [35]

3.4. Ohjelmointikielen valinta

Ohjelmointikieleksi valitaan PHP 5, koska kehitystyöryhmällä on sen käytöstä aikaisempaa kokemusta ja myös muut, samaan aikaan meneillään olevat projektit toteutetaan pääsääntöisesti PHP 5:llä. PHP-ohjelmointikieli on ilmainen ja sitä pidetään helposti opittavana, minkä myötä sen osaajia on paljon. Täten asiakkaat, jotka ostavat tässä työssä toteutettavan verkkokauppasovellusalustan, voivat löytää kohtalaisen helposti henkilöitä, jotka kykenevät räätälöimään verkkokauppaa.

3.5. Ohjelmistokehyksen valinta

Niin kuin nykyaikaiset web-sovellukset usein, tämänkin diplomityön sovellus kehitetään ohjelmistokehystä käyttäen. Huomioitavaa on, että Teoria-luvussa arvioitavina olleista kolmesta verkkokauppasovellusalustasta kaksi (Magento ja Interspire Shopping Cart) käyttää ohjelmistokehystä.

Web-ohjelmistokehyksiä on useita erilaisia ja niitä on vaikea laittaa paremmuusjärjestykseen. Pikemminkin kyse on siitä, mikä ohjelmistokehys sopii parhaiten juuri tiettyyn projektiin. Ohjelmistokehyksen valinta tulisi arvioida aina uudestaan uuden projektin alkaessa. Seuraavassa käsitellään kriteerejä, joiden tärkeyttä kannattaa arvioida valintaa tehdessä. [19, 16]

Ohjelmistokehyksen arkkitehtuurissa on olennaista, pohjautuuko se proserudaaliseen vai olio-ohjelmointiparadigmaan. Kannattaa valita ohjelmistokehys, jonka arkkitehtuurin kehittäjät ymmärtävät. Tyypillisesti sovellusta on kehittämässä useita ihmisiä, joiden osaaminen vaihtelee paljon. Olisi tavoiteltavaa, että kaikki kehittäjät tuntisivat kehyksen, mutta koska tämä on usein mahdotonta, niin realistisempi tavoite on, että edes yksi kehittäjistä tuntee valittavan kehyksen. Eri kehittäjillä voi myös olla erilainen näkemys ohjelmointikäytänteistä, sivupohjakielestä ja suunnittelumalleista. [19, 32]

Ohjelmistokehysten dokumentaatiot vaihtelevat suuresti lyhyistä asennusohjeista kokonaisiin oppimisluentosarjoihin. Valintaa tehdessä kannattaa tarkistaa, onko ohjelmistokehyksen dokumentaatio ylipäänsä ajantasalla, ja sen lisäksi selkeä ja ymmärrettävä. Kattaako dokumentaatio kaiken vai onko se mahdollisesti keskeneräinen? Iso osa tarjolla olevista ohjelmistokehysten dokumentaatioista on tietokoneiden automaattisesti generoimia, joiden hyödyllisyys on usein kyseenalainen. [19]

Myös ohjelmistokehyksen ympärillä olevalla yhteisöllä on merkitystä. Mitä suositumpi kehys on, sen todennäköisemmin internetin keskustelupalstoilta löytyy auttajia ja osaajia. Myös mahdollisesti mukaan tulevat uudet kehittäjät voivat tuntea kehyksen. Miten yhteisön keskustelupalstoilla suhtaudutaan kysymyksiin? Kuinka nopeasti yhteisö reagoi ohjelmakoodivirheilmoituksiin? Kuinka usein kehyksestä julkaistaan uusi versio? Nämä kysymykset auttavat rakentamaan parempaa mielikuvaa yhteisöstä. [19, 16]

Kuinka arvioitavana olevan ohjelmistokehyksen tuki on hoidettu? On yleistä, että kehysten tuet ovat maksullisia ja ulkoistettu kolmannelle osapuolelle. Internetin keskustelupalstojen ja sähköpostiryhmien tarjoamaan ilmaistukeen ei välttämättä kannata luottaa kaupallisissa projekteissa, sillä vastausten vasteajoista ja luotettavuudesta ei ole takeita sekä joihinkin erikoisempiin kysymyksiin ei apua välttämättä saa ollenkaan.

Omista tuensaantimahdollisuuksista kannattaa olla realistinen kuva, sillä ongelmien ilmetessä voi tulla hyvin kalliiksi havaita, ettei apua ole välittömästi tarjolla. [19]

Useilla ohjelmistokehyksillä on yllättäviä rajotteita: esimerkiksi sivupohjakieli voi olla määrätty joksikin tietyksi, tai tuki tietokannoille vajavainen. Ohjelmistokehyksissä on myös eroja siinä, miten olemassaoleva sovellus siirretään käyttämään kehystä. Kehitettävän ohjelmiston elinkaarella voi olla myös merkitystä kehystä valittaessa. [19, 16]

Tässä diplomityössä kehitettävän sovelluksen alla päätetään käyttää Liaani Framework -ohjelmistokehystä (lyhyemmin Liaani). Kyseessä on ohjelmistokehys, jonka on kehittänyt sama organisaatio (Koodiviidakko Oy), jonka toimeksiannosta tämän diplomityön sovellus kehitetään. Allekirjoittaneella oli mahdollisuus vaikuttaa valittavaan ohjelmistokehykseen, mutta koska aikaisempaa kokemusta mistään kehyksestä ei ollut, päädyttiin valitsemaan Liaani Framework, jonka käytöstä organisaatiolla itsellään on paljon kokemusta. Ohjelmistokehykseen liittyvät tukitoimet onnistuvat helposti, kun sen kehittäjä työskentelee samassa organisaatiossa verkkokauppasovellusalustaa kehittävien työntekijöiden kanssa. Lisäksi ohjelmistokehyksen kehityssuuntaa voidaan ohjata tarpeen mukaan. Organisaation oman ohjelmistokehyksen käyttämisen huono puoli on se, ettei kukaan uusi työntekijä tunne Liaania entuudestaan.

Liaanin arkkitehtuuri on yksinkertainen, mikä mahdollistaa monipuolisen ohjelmistokehityksen. Yksinkertaisuudesta johtuen alustassa ei ole runsaasti valmiita työkaluja. Liaani käyttää MVC-ohjelmistoarkkitehtuuria ja hyödyntää PHP 5:n parannettuja ominaisuuksia, kuten metodien näkyvyyksiä. Liaanin tietokanta-abstraktiona käytetään oliorelaatiokuvausta (eng. Object-relational Mapping, ORM). Liaanissa on tuki MySQL:n ja PostgreSQL:n lisäksi monikielisyydelle, auditoinnille ja automatisoiduille testeille sekä immunisointi XSS-hyökkäyksiä (Cross-site Scripting) vastaan. [36]

Taulukossa 1 on esitelty Liaanin hakemistorakenne. Taulukossa *kehyksellä* tarkoitetaan Liaanin sisäiseen ja *sovelluksella* kehitettävänä olevan sovelluksen käyttöön tarkotettuja tiedostoja. [36]

Taulukko 1: Liaanin hakemistorakenne

Hakemisto	**Sisältö**	**Käyttö**
lib/	Liaanin ohjelmakoodit	kehys
lib/Renderer/	valmiina tarjotut renderöijät	kehys
Controller/	sovelluksen ohjaimet	sovellus
Model/	sovelluksen mallit	sovellus
Plugin/	sovelluksen liitännäisosat	sovellus
Renderer/	sovelluksen renderöijät	sovellus
Template/	sovelluksen näkymät	sovellus
htdocs/static/	sovelluksen staattinen materiaali	sovellus
base/t/	automatisoidut testit	yhteinen

Kehyksen sisäiseen käyttöön tarkoitetut hakemistot sisältävät Liaanin omat ohjelmakoodit. Vastaavasti kehitettävälle sovellukselle tarkoitettuihin hakemistoihin sijoitetaan sovelluksen ohjelmakoodit. Staattinen materiaali tarkoittaa esimerkiksi kuvia, CSS-tyylimäärittely- ja JavaScript-tiedostoja. Testihakemistoon kuuluvat sekä Liaanin omat että kehitettävän sovelluksen automatisoidut testit. [36]

Kaikki selaimen lähettämät palvelupyynnöt ohjataan esilatausohjelmalle lukuunottamatta `htdocs/static`-hakemistoon kohdistuneita pyyntöjä. Seuraavassa on esitetty, kuinka Liaani käsittelee palvelupyynnön. [36]

1. **Valmistelu:** Alustetaan Liaani ja valmistellaan perustoiminnallisuudet, kuten tiedostojen automaattinen lataaminen (eng. autoload) ja virheiden käsittely sekä ladataan sovelluksen konfiguraatio.

2. **Ohjaimen valinta:** Tutkitaan pyydettyä URI:a ja valitaan sen perusteella ohjain. Jos ohjainta ei ole olemassa, päädytään virhetilanteeseen.

3. **Ohjaimen valmistelu ja metodin valinta:** Luodaan olio valitun ohjaimen luokasta. Valitaan palvelupyynnön mukainen ohjaimen metodi.

4. **Sovelluksen toimintalogiikka:** Suoritetaan palvelupyynnön vaatima ohjelmalogiikka.

5. **Näkymän tuottaminen:** Ohjain valitsee palvelupyynnön mukaisen sivupohjan ja renderöijän, joiden avulla näkymä renderöidään.

3.6. Käyttöliittymäkielten valinta

Tässä työssä ei keskitytä käyttöliittymäkielten valinnan pohtimiseen, vaan todetaan vain, että näkymien toteuttamiseen käytetään kuvauskielenä XHTML (Extensible Hypertext Markup Language) 1.0 Transitionalia, tyylijärjestelmänä CSS 2.0:aa ja komentosarjakielenä JavaScriptiä, erityisesti jQuery-kirjastoa.

3.7. Alustava tietokantaskeema

Projektin alussa kokeneempi ohjelmistokehittäjä teki alustavan tietokantasuunnitelman luomalla neljä tietokantataulua tuotteiden ja kategorioiden hallintaan. Tästä sovelluksen varsinainen toteuttaminen alkoi. Tietokantaskeema on esitelty liitteessä 2.

4. TOTEUTUS

Toteutus-luvussa tarkastellaan pääosin tässä työssä kehitettävän sovelluksen toteutusta, mutta myös hieman sovelluksen käyttämää ohjelmistokehystä. Tarkasteltavana olevan sovelluksen revisionumero on 778.

Aluksi kuvataan yleisellä tasolla, kuinka sovelluksen arkkitehtuuri on rakennettu, eli millasia hakemistoja, tiedostoja ja tietokantatauluja on käytössä. Lisäksi esitellään esimerkkipalvelupyyntö ja sen käsittely sekä tarkastellaan, miten tietoturvallisuus on huomioitu. Yksittäisistä ominaisuuksista tarkempaan tarkasteluun nostetaan sovelluksen lokalisointi, ulkoasun hallinta, tuotteiden hinnoittelulogiikka, tilauksen toimituskulujen laskentalogiikka ja tuotteiden näkyvyyskriteerit. Lopuksi tutkitaan, kuinka tietokannan abstraktointi on toteutettu ja kuinka sitä käytetään sovelluksessa.

4.1. Arkkitehtuuri

Arkkitehtuuriin pureudutaan kuvaamalla ensin sovelluksen hakemistorakennetta ja tiedostoja, minkä jälkeen esitetään, kuinka tieto varastoidaan sovelluksessa.

Verkkokauppasovellusalustan tiedostojen sijainnit noudattavat Liaani-ohjelmistokehyksen suosittamaa jakoa, eli pääpiirteissään juuressa on vain konfiguraatiotiedosto, testit sijoitetaan `base`-, ohjaimet `Controller`-, käännöstiedostot `Data`-, esilautausohjelma ja staattinen sisältö `htdocs`-, Liaanin ohjelmakoodit `lib`-, mallit `Model`-, liitännäisosat `Plugin`-, renderöijät `Renderer`- ja näkymät `Template`-hakemistoon.

Käytännössä kaikki ohjaimet perivät erityisen juuriohjaimen, joka taas perii Liaanin abstraktin ohjainluokan. Vastaavasti melkein kaikki mallit perivät Liaanin malliluokan ja renderöijät renderöijäluokan.

Liitteessä 3 on listattuna aakkosjärjestyksessä osa sovelluksen tiedostoista ja kerrottu lyhyesti niiden tarkoitus.

Verkkokauppasovellusalustan data tallennetaan pääosin SQL-tietokantatauluihin (Structured Query Language), jotka ovat listattuna liitteessä 4. Lisäksi jonkin verran dataa, kuten tuotekuvat, ylläpidon kielikäännökset, sivupohjarungot ja debug-tulosteet, tallennetaan sovelluksen hakemistorakenteeseen.

4.2. Esimerkki palvelupyynnöstä

Seuraavassa esimerkki siitä, mitä tapahtuu ohjelmakooditasolla, kun lähetetään POST- tai GET-palvelupyyntö osoitteeseen: `http://domain/admin/category/edit/32432`.

1. Web-selain lähettää palvelupyynnön.
2. Ohjelmistokehyksen esilatausohjelma alustaa sovelluksen ja valitsee `AdminController`-ohjaimen ja siitä `category()`-metodin. Valinta tapahtuu URI:sta, missä *admin* viittaa `AdminController`:iin ja *category* julkiseen `category()`-metodiin.

3. Annetaan `category()`-metodille parametreinä *edit* ja *32432*. Metodin sisällä nämä tarkoittavat toimintoa (muita voisi olla esimerkiksi *add*, *move* ja *delete*) ja kategorian tunnistetta (eng. id).

4. Metodin sisällä ensimmäisenä tarkistetaan, että palvelupyynnön tehneellä käyttäjällä on oikeus kategorian tietojen tarkasteluun. Jos tätä oikeutta ei ole, käyttäjä ohjataan julkiselle puolelle.

5. Seuraavaksi alustetaan kategoriaiteraattori näkymää varten. Iteraattori on apuväline saman tyyppisten mallien läpikäymiseen. Tässä tapauksessa kategoriahallinan näkymässä halutaan tulostaa kaikki kaupan kategoriat yhdessä listassa, joista ylläpitäjä voi valita haluamansa kategorian muun muassa muokattavaksi.

6. Koska metodi saa syötteenä *edit*-parametrin, siirrytään sovelluksen logiikan mukaisesti `editCategory()`-metodiin ja annetaan sille syötteenä kategorian tunniste. `editCategory()` on suojattu metodi, joten siihen ei pääse käsiksi muuten kuin julkisen `category()`-metodin kautta.

7. Metodin alussa asetetaan sivupohjaksi `Template/admin/category/edit.tpl.php`, joka tullaan myöhemmässä vaiheessa täyttämään datalla.

8. Luodaan `Request`-olio, jonka avulla tullaan myöhemmin validoimaan POST-dataa.

9. Luodaan `Category`-olio annetulla kategoriatunnisteella.

10. Alustetaan näkymää varten taulukkomuuttujat (eng. array) lokalisoiduille nimille ja kuvauksille.

11. Jos palvelupyyntö on GET-tyyppinen, siirrytään takaisin `category()`-metodiin, mistä siirrytään vielä esilatausohjelmaan, joka renderöi ohjaimen keräämän datan `Template/admin/category/edit.tpl.php`-sivupohjalle. GET-tyyppisen palvelupyynnön käsittely päättyy tähän. Käytännössä GET-tyyppinen palvelupyyntö tarkoittaa kategorian muokkaussivulle menemistä selaimella.

12. Sen sijaan, jos palvelupyyntö on POST-tyyppinen, siirrytään sanitoimaan POST-palvelupyynnössä mukana tullut data.

13. Hyödynnetään ohjelmistokehyksen tarjoamia työkaluja datan sanitoimiseen ja tallennetaan käsitelty data `Category`-olion ominaisuuksiin.

14. Tallennetaan uusi data tietokantaan `Category`-olion `save()`-metodin avulla.

15. Siirrytään takaisin `category()`-metodiin, mistä siirrytään vielä esilatausohjelmaan, joka renderöi ohjaimen keräämän datan `Template/admin/category/edit.tpl.php`-sivupohjalle. POST-tyyppisen palvelupyynnön käsittely päättyy tähän. Käytännössä POST-tyyppinen palvelupyyntö tarkoittaa, että kategoriaa on muokattu kategoriahallinnasta.

4.3. Tietoturvallisuuden huomioiminen

Vaikka verkkokauppasovelluksissa rahan käsittely yleensä ulkoistetaan kolmannelle osapuolelle, on tietoturvallisuudella silti iso merkitys kehitettäessä verkkokauppasovellusalustaa, joten on tärkeää arvioida, millaisia uhkia mahdollisesti kohdataan ja miten niihin varaudutaan. Uhka voidaan joko välttää, siirtää, lieventää tai hyväksyä. Välttäminen tarkoittaa, että suojaudutaan uhalta niin, ettei se toteudu. Lisäksi uhka voidaan siirtää sopimuksen kolmannelle osapuolelle, kuten vakuutusyhtiölle. Uhka voidaan myös hyväksyä, kuten esimerkiksi sodan syttyminen, jolloin asialle ei ole käytännössä mitään tehtävissä. Uhan lievennys taas tarkoittaa toimia, missä uhan toteutumisen todennäköisyyttä vähennetään tai uhan realisoitumisen tuottaman vahingon suuruutta pienennetään. Esimerkki tälläisestä on datan varmuuskopiointi. [37]

Lähestykäämme erilaisia uhkia Whitmanin (2004) esittämän yleisen uhkaluokituksen perusteella. Useat tekstikappaleet sopivat useiden otsakkeiden alle, mutta tekstejä ei ole toistettu, eli otsakkeiden alla on kerrottu vain niistä ominaisuuksista, joita ei ole vielä mainittu aiemmin tässä luvussa.

Inhimillinen virhe tai laiminlyönti: Sovellus on kehitetty niin, ettei mitään kriittistä dataa pystytä todellisuudessa poistamaan suoraan ylläpidon hallinnasta, vaan esimerkiksi poistetut tuotteet merkitään ainoastaan poistetuiksi. Tuote häviää kyllä kaikista näkymistä, mutta tiedot ovat edelleen tietokannassa. Sama tehdään myös kategorioille, kuville, tuotevariaatioille, tilauksille ja käyttäjille. Lisäksi roolipohjainen pääsynvalvontamalli mahdollistaa erilaisten oikeuksien antamisen eri rooleille eli käyttäjäryhmille. Näin esimerkiksi tilausten käsittelijää voidaan estää pääsemästä kaupan ulkoasun hallintaan tai vaikka poistamaan tuotteita.

Myöhemmin sovellukseen voidaan harkita toteutettavan ominaisuus, joka varoittaa käyttäjän lisäämisestä käyttäjäryhmään, jolla on oikeus ylläpitoon, jottei esimerkiksi yritysasiakasta vahingossa lisättäisi ylläpitäjäryhmään. Myös käyttäjien salasanoihin tullaan todennäköisesti lisämään vaatimuksia, etteivät ne olisi liian lyhyitä tai yksinkertaisia.

Tilaushallinnassa - ja ylläpidossa muutenkin - on tärkeää erottaa selvästi, mitkä tilaukset ovat myyjän hoidettavana, jotta tilausten toimittaminen ei hidastuisi ainakaan verkkokauppasovelluksen takia. Sovelluksessa on tilausten eri tiloille toteutettu omat värikoodit ja luokittelu tilojen mukaan. Näitä ominaisuuksia tullaan tarkkailemaan huolella ja kehittämään, jos niissä havaitaan puutteita. Se, kuinka nopeasti verkkokauppa toimittaa tuotteet asiakkaalleen, on yksi tärkeimmistä asioista verkkokaupan liiketoiminnan kannalta.

Immateriaalioikeuksien rikkominen: Kirjoitushetkellä sovellus ei tue digitaalisia tuotteita, eli että kaupassa voitaisiin myydä esimerkiksi kuvatiedostoja, jotka asiakas voisi välittömästi maksamisen jälkeen ladata omalle tietokoneelleen. Viimeistään tämän ominaisuuden myötä tulee tarve miettiä, kuinka immateriaalioikeuksien rikkomista estetään. Yksi kuvatiedostojen suojaamiskeino on vesileimaus, jota voidaan harkita käytettäväksi myös tuote- ja kategoriakuvissa.

Vakoilu tai luvaton tunkeutuminen: Sovelluksen pääsynvalvontaa hoidetaan käyttäjä- ja roolinhallinnalla. Eri käyttäjäryhmillä ja täten myös eri käyttä-

jillä on vaihtelevia oikeuksia. Esimerkiksi kuluttaja- tai yritysasiakasryhmiin kuuluvia käyttäjiä ei tyypillisesti päästetä ylläpitoon. Ohjelmakooditasolla tämä on tehty hyvin yksinkertaisesti niin, että `AdminController`-ohjaimen konstruktori tarkistaa jokaisella ylläpidon palvelupyynnöllä, että käyttäjällä on oikeus olla ylläpidossa. Lisäksi eri entiteettien (esimerkiksi tuotteet, kategoriat, asetukset, ulkoasut) hallintaan liittyvät oikeustarkistukset tapahtuvat yksinkertaisesti entiteettien päämetodin tasolla. Tämän etu on, että lisättäessä uusia ominaisuuksia sovellukseen oikeuksien tarkisteluun ei tarvitse tehdä isoja muutoksia.

Käyttäjien salasanat tallennetaan tietokantaan SHA1-tiivistettynä (eng. hashed, Secure Hash Algorithm), ja lisäksi tietokanta voidaan itsessään suojata tunnus-salasana-parilla. Myös yhteydet tietokantaan voidaan sallia vain tietyistä IP-osoitteista esimerkiksi niin, että yhteydet sallitaan vain samasta lähiverkosta.

Sovelluksessa käyttäjän ja ostoskorin tilatietoa tallennetaan evästeisiin. Evästeiden tunnisteet on myös tiivistetty MD5- (Message-Digest algorithm 5) tai SHA1-algoritmeillä. Tiiviste sisältää kauppakohtaisen suolan sekä kauppakohtaisen tunnisteen käyttäjälle tai ostoskorille. Nämä toimenpiteet on tehty siksi, ettei evästettä pystyttäisi generoimaan kolmannen osapuolen toimesta ja näin onnistuttaisi kirjautumaan järjestelmään ja ostamaan tuotteita toisen käyttäjän tiedoilla.

Myöhemmin sovellukseen voidaan lisätä auditointiominaisuuksia, jolloin saataisiin lokitietoa siitä, kuka teki mitäkin, mistä IP-osoitteesta ja milloin. Tällöin lokeista voitaisiin seurata, ettei mitään odottamatonta tapahtuisi tai jos tapahtuu, niin tapahtumia voitaisiin tutkia lokeista jälkikäteen.

Informaatioon liittyvä kiskonta ja kiristys: Käyttäjien luottokorttien turvakoodia (niin sanottu CVV [Card Verification Value]) ei tallenneta tietokantaan, jolloin tietokantavarkauden yhteydessä asiakkaiden luottokorttinumeroista ei ole hyötyä, koska nykyään useimmat etämyyjät vaativat turvakoodin luottokortilla ostaessa.

Sabotaasi ja vandalismi: Pahantahtoiset käyttäjät voivat tehdä valetilauksia. Kauppias voi eri toimitustapoja tarjoamalla rajata, haluaako hän esimerkiksi sallia postiennakkoa. Suomen laki mahdollistaa myös tuotteiden palautuksen mistä tahansa syystä neljäntoista vuorokauden sisällä tilauksen vastaanottamisesta. Nämä ongelmat eivät ole varsinaisesti sovelluksen ongelmia, mutta järjestelmä voi tarjota työkaluja ongelmien lieventämiseksi. Verkkokaupassa voitaisiin esimerkiksi estää kaupassa vierailu tai ostaminen tietyistä IP-osotteista tai maista. Samalla tavalla voitaisiin rajoittaa mahdollisesti myöhemmin toteutettavia käyttäjäkommentointi- ja tuotearviointiominaisuuksia. Järjestelmä voisi myös muistaa, ketkä ovat tehneet tuotepalautuksia ja mistä syistä.

Tiedossa on myös toinen pienimuotoinen kiusantekotapa, jota vastaan ei ole varauduttu. Verkkokaupan tuotesivun osoite voi olla esimerkiksi `www.kauppa.fi/product/show/1/Autot/1/Mersu`, missä itse asiassa sanat “Autot” ja “Mersu” on ainoastaan tarkoitettu hakukoneoptimointia varten eli sovellus itsessään ei tarkista niiden sisältöä ollenkaan. Tästä syystä käyttäjät voivat muuttaa osoitteen esimerkiksi muotoon

`www.kauppa.fi/product/show/1/Autot/1/Mersu-on-huono` ja levittää tätä osoitetta muille käyttäjille. Tämä ongelma on lähinnä kosmeettinen ja tullaan korjaamaan vain, jos siitä koetaan aiheutuvan haittaa.

Varkaus: Digitaalisia tuotteita voitaisiin ajatella olla mahdollista varastaa, mutta digitaalisia tuotteita ei ole sovellukseen vielä toteutettu. Myös tietokantoja voidaan varastaa, mutta tämä on jo käsitelty aiemmin. Sen lisäksi itse palvelinlaitteisto voidaan varastaa, mutta näiden ongelmien tarkastelu ei kuulu tämän työn piiriin.

Ohjelmistohyökkäykset: Liaani Framework -ohjelmistokehys pakottaa sanitoimaan käyttäjän syöttämän datan ennen sen käsittelyä. Lisäksi tietokantakyselyjen syötteet tulee antaa erityisten paikanvaraajien (eng. placeholder) kautta. Näiden avulla järjestelmä estää hyvin kattavasti erilaisia XSS-injektointeja ohjelmakoodiin ja tietokantoihin. XSS-injektoinnin idea on yksinkertaisuudessaan se, että käyttäjä antaa esimerkiksi käyttäjätietoja rekisteröidessään nimekseen suoran SQL-kyselyn, joka voi esimerkiksi tuhota koko tietokantataulun.

Luonnonvoimat: Luonnonvoimiin varaudutaan siirtämällä uhkaa vakuutusten avulla kolmannelle osapuolelle, sopimalla asiakkaan kanssa palvelusopimuksessa reunaehdoista ja viime kädessä hyväksymällä, että aivan kaikkeen ei voida varautua.

Poikkeamat palvelunlaadussa: Poikkeamiin palvelulaadussa varaudutaan asiakkaan kanssa tehdyssä sopimuksessa. Tässä työssä ei käsitellä tarkemmin sopimusteknisiä asioita.

Teknilliset laitteistopuutteet ja -virheet: Palvelintietokoneiden kriittisin osa on kiintolevyjärjestelmä. Kiintolevyn hajoaminen on ensinnäkin tavallista ja toisekseen kiintolevyillä on dataa, jonka häviämistä ei yleensä voida hyväksyä. Lähes poikkeuksetta erilaisissa palvelinympäristöissä kiintolevyt on kahdennettu niin, että jos yksi (tai muutama, riippuen kahdennusjärjestelmästä) kiintolevy hajoaa, järjestelmä ei kadota dataa ja parhaimmillaan jatkaa toimintaa ilman keskeytyksiä asiakkaan näkökulmasta. Levyrikosta lähtee ilmoitus palvelimen ylläpitäjälle, joka käy vaihtamassa järjestelmään uuden kiintolevyn, jolloin tilanne on taas sama kuin ennen rikkoutumista.

Vakavia ongelmia tulee yleensä vasta silloin, kun levyjä rikkoutuu useita kerrallaan. Tälläisiä tilanteita varten käytetään tietojen varmuuskopiointia. Tavallisesti palvelinten kriittisiä datoja varmuuskopioidaan periodisesti toisille tietokoneille ja medioille, jotka mielellään myös sijaitsevat fyysisesti toisaalla, jotta tietoa ei menetettäisi edes tulipalon tai vastaavan onnettomuuden tapahtuessa.

Muiden osien hajoaminen palvelintietokoneissa ei ole niin kriittistä. Kehittyneissä palvelinympäristöissä muisti- ja laskentatehot on klusteroitu niin, että eri tietokoneiden suorituskyvyt jaetaan niiden kesken. Tällöin yhden muistikamman tai prosessorin hajoaminen ei juurikaan vaikuta koko palvelinryppään suorituskapasiteettiin. Toisaalta jos palvelin työskentelee yksin, muistikamman hajoaminen voi pysäyttää palvelimen toiminnan kokonaan, jolloin ylläpitäjän tulee käynnistää palvelin uudelleen muistikamman vaihtamisen jälkeen. Tällöin ei kuitenkaan dataa menetetä lukuunottamatta tietoa, joka on ollut ainoastaan tietokoneen välimuistissa.

Teknilliset ohjelmistohäiriöt ja -virheet: Ohjelmistohäiriöihin ja -virheisiin varaudutaan ennen kaikkea testeillä. Testausta suoritetaan tietenkin koko sovelluksen kehityksen ja käytön ajan, mutta sen lisäksi tehdään automatisoituja yksikkötestejä perustoiminnallisuuksien testaamiseen. Yksikkötestien erinomainen puoli on niiden suorittamisen vaivattomuus. Niillä voidaan helposti ja usein testata sovellusta. Yksikkötestien ongelma on, että niiden kirjoittaminen ja päivittäminen vie verrattain paljon aikaa ja tästä syystä testeillä yleensä katetaan vain osa toiminnallisuuksista. Yksikkötesteillä ei myöskään voida testata käyttöliittymää, eli esimerkiksi poistaako "poista tuote" -nappi todellakin tuotteen. Yksikkötestit testaavat lähinnä, toimiiko ohjelmakoodin logiikka tarkoitetulla tavalla.

Teknologian vanhentuneisuus: Teknologian vanhentuneisuuteen ei tarvitse erityisemmin varautua niin kauan kuin sovelluksen kehitys on aktiivista. Kun kehittäminen loppuu, tehdään erityinen suunnitelma sovelluksen ylläpidon jatkamisesta. Esimerkiksi osCommercen kehitys on loppunut jo vuosia sitten, mutta aktiivisen ylläpidon ansiosta sovellus on pysynyt suosittuna.

4.4. Lokalisaatio

Kuten useiden nykyaikaisten web-sovellusten, niin myös yksi tässä työssä kehitettävän sovelluksen päätavoitteista on kokonaisvaltainen lokalisointi, jotta kauppoja olisi mahdollista yhtä aikaa pitää useilla eri kielillä. Tavallisesti kauppa halutaan saada paikallisella, esimerkiksi suomen, ja englannin kielellä, mutta kasvavissa määrin myös ei-latinalaisten aakkosten kielillä kuten kiinaksi tai venäjäksi.

Tavallisesti lokalisoinnin piiriin kuuluvat tekstit, kuvat, numerot, kellonajat, päivämäärät ja valuutat. Erilaisten listojen järjestäminen aakkosjärjestykseen on myös eräs lokalisaation ongelmista. Lokalisointi on erittäin haastava ohjelmistokehityksen osa, ei vähiten siksi, että lokalisaatiotuki on PHP 5:ssä vielä varsin puutteellinen, mutta ylipäänsä eri web-teknologiastandardeissa asiaa ei ole alunperinkään huomioitu riittävällä tarkkuudella, koska ei ole ymmärretty, että web tulee leviämään kaikkien ihmisten käyttöön ympäri maapallon. [38]

Lokalisaatio on huomioitu tämän työn sovelluksen kehityksessä alusta asti, mutta monelta osin toteutus on kirjoitushetkellä tekemättä. Seuraavassa on eriteltynä, missä vaiheessa lokalisaatio on sovelluksen eri osissa.

Ylläpidon kaikki tekstit tulevat XML-käännöstiedostoista (Extensible Markup Language), joten siltä osilta lokalisointi on toteutettu. Uusia kieliä pystytään lisäämään helposti kääntämällä vain käännöstiedosto uudelle kielelle ja lisäämällä tuettu kieli kaupan asetuksiin.

Vastaavasti julkipuolen sivupohjissa käytetään erityisiä käännösavainsanoja (`<% LANG:FI %>Tervetuloa<% ENDLANG:FI %> <% LANG:EN %>Welcome<% ENDLANG:EN %>`), joiden avulla voidaan sivupohjiin kirjoittaa tekstejä eri kielille. Tämän lähestymistavan hyvä puoli on selkeys, koska tekstejä ei kirjoiteta erilliseen käännöstiedostoon. Tekstien hallinnointi on oletettavasti helpompaa ainakin silloin, kun eri kielivaihtoehtoja on vain muutamia. Toisaalta julkisen puolen teksteihin tullaan myöhemmin mahdollisesti tekemään ominaisuus, että tekstit voivat olla myös käännöstiedostoissa, jolloin sivupohjissa vain viitattaisiin

tiettyyn käännöstiedoston kohtaan (esimerkiksi `<% TEXT:MAIN:WELCOME %>`). Tämän huono puoli olisi, että uutta sisältöä sivuille tehtäessä tekstejä pitäisi kirjoittaa eri paikkaan kuin tekstin muotoilut tehdään, mutta toisaalta kokonaan uuden kielen lisääminen olisi hyvin helppoa: käännöstiedosto tarvitsee vain kääntää uudelle kielelle.

Myös tuote- ja kategoriatiedot voidaan kääntää eri kielille. Lisäksi tietokannassa on varauduttu jo useiden muidenkin tietueiden lokalisointiin, mutta niiden käyttöä ei ole vielä toteutettu. Tälläisiä ovat esimerkiksi tuotekuvat, tuotevariaatiot sekä toimitus- ja maksutavat.

Sovelluksen asetuksissa voidaan asettaa lyhyt ja pitkä päivämääräformaatti, mutta näitä käytetään vain ylläpidon näkymissä. Kirjoitushetkellä ei ole tukea julkipuolen lokalisoiduille päivämääräformaateille. Myöskään kellonaikoja ei ole lokalisoitu eikä käyttäjä pysty asettamaan asetuksistaan aikavyöhykettään.

Lukujen erityispiirteitä, eli millainen desimaalierotin on tai miten tuhannet erotetaan toisistaan, jos mitenkään, ei ole sovelluksessa huomioitu ollenkaan. Lisäksi valuuttoja ei ole toteutettu. Valuuttojen lokalisoinnissa erityishuomioitavaa on valuuttasymbolin sijoittaminen, sillä esimerkiksi yhdysvaltojen dollarin valuuttatunnus merkitään ennen lukua ja euron valuuttatunnus luvun jälkeen. Valuuttatuki on verkkokauppasovellusalustassa olennainen ominaisuus ja se tullaan varmasti toteuttamaan lähitulevaisuudessa.

4.5. Ulkoasun hallinta

Sovelluksen ulkoasun hallinta on jaettu neljään eri osioon: sivupohjien, lisäkkeiden, tiedostojen ja sisältösivujen hallintaan. Seuraavassa osiot on esitelty yksinkertaisuusjärjestyksessä yksinkertaisimmasta alkaen.

Tiedostojenhallinta on tarkoitettu staattisten tiedostojen, kuten kuvien, tyyliohje- (CSS) ja asiakirjatiedostojen hallintaan. Tiedostoja voidaan käyttää esimerkiksi ulkoasun rakentamisessa ja vaikkapa PDF-muotoisten (Portable Document Format) sopimusehtojen liittämisessä sivulle. Tiedostojenhallinnassa voi lisätä, poistaa ja katsella tiedostoja sekä muokata tekstimuotoisia tiedostoja.

Sisältösivut (tai CMS-sivut [Content management system]) ovat sivuja, joita kauppias itse voi muokata helposti haluamansa näköisiksi. Esimerkiksi on tavallista, että halutaan luoda oma staattinen sivu yhteystiedoille, rekisteriselosteelle ja “Tietoa yrityksestä” -sivulle. Sisältöhallinnassa on käytössä WYSIWYG-editori (What You See Is What You Get), eli käyttäjä pystyy rakentamaan sivun, joka sisältää esimerkiksi kuvia, taulukoita ja listoja, osaamatta ollenkaan HTML- tai muuta erityisestä kuvauskieltä. Sisältösivulle tulee aina määrätä käytettävä sivupohja.

Sivupohjat ovat runkoja kaupan eri sivuille. Oma sivupohja on käytössä esimerkiksi etusivulla, käyttäjän rekisteröintisivulla, tuotelistaus- ja tuotesivulla ja ostoskorin eri vaiheissa. Sivupohjat sisältävät HTML-kuvauskieltä sekä viittauksia lisäkkeisiin (`<% INCLUDE:lisakkeen_nimi %>`). Sivupohjissa voidaan myös käyttää avainsanaa `CONTENT`, joka osoittaa kohtaa, johon mahdollinen sisältösivun sisältö liitetään. Sivupohjat ovat tallennettuna sovelluksen hakemistorakenteeseen.

Lisäkkeet ovat sivupohjien liitännäisosia, joihin voidaan sisältää ehdollista logiikkaa (if ... else -lausekkeita), toisia lisäkkeitä, tietoa satojen eri avainsanojen kautta

(esimerkiksi `PRODUCT_PRICE`, `IS_CURRENT` tai `HAS_EMAIL_ERROR`) ja myös varsinaista HTML-kuvauskieltä. Lisäkkeiden avulla rakennetaan siis dynaaminen ulkoasu. Lisäke voi olla joko lista- tai objektityyppinen. Listatyyppinen lisäke liitetään niin monta kertaa kuin listalla on rivejä. Objektityyppinen lisäke sen sijaan liitetään vain kerran. Lisäkkeet tallennetaan järjestelmän tietokantaan.

Tallennettaessa lisäkettä järjestelmä kääntää lisäkkeen PHP-kieliseksi ohjelmakoodiksi eli avainsanoista tehdään muuttujia ja lisäkkeiden ehtorakenteet muutetaan PHP:n ehtorakennesyntaksin mukaiseksi. Lisäkkeiden kääntämisen jälkeen käännetään myös sivupohjat, joihin lisäkkeet on liitetty. Lopputuotteena saadaan kokonaisia PHP-kielisiä sivupohjia, jotka sisältävät myös lisäkkeiden tekstit, muuttujat ja ehtorakenteet.

4.6. Hinnoittelulogiikka

Kaupassa vierailevalle kävijälle näytettävä tuotteen hinta riippuu useista tekijöistä. Kaupan asetuksissa määrätään, ovatko tietokannan hinnat verollisia vai verottomia ja näytetäänkö hinnat sivulla oletuksena verollisina vai verottomina. Tuotteen perushinta voidaan asettaa joko tuotekohtaisesti tai kaupan, kategorian tai tuotteen kateprosentin ja inventaarihinnan avulla. Lopullista hintaa voivat muuttaa myös mahdolliset tuotevariaatiot. Lisäksi kaupassa voi olla kauppa-, kategoria- ja tuotekohtaisia alennuksia joko kaikille kävijöille tai vain tietyille käyttäjäryhmille.

Tuotemallin (`Model/Product.php`) `getPrice()`-metodi ottaa syöteparametreinä käyttäjäryhmän, tiedon huomioidaanko alennukset, tiedon halutaanko hinta verollisena vai verottomana sekä mahdolliset tuotevariaatioiden tunnisteet. Lisäksi kaupan asetuksista haetaan tieto, onko tietokannan hinnat verollisia vai verottomia sekä sallitaanko hinnan laskeminen kaupan ja kategorian kateprosenttien avulla.

Hinnan laskentalogiikka etenee seuraavasti:

1. Alustetaan metodi: huomioidaan syöteparametrit, haetaan kaupan asetukset, tarkistetaan, että tuotteella on kategoria ja että tuote on ylipäänsä olemassa.

2. Lasketaan tuotteelle niin sanottu perushinta `getBasePrice()`-metodin avulla. Perushinta määräytyy joko tuotteelle kiinteästi asetetusta hinnasta tai lasketuna tuotteen, kategorian tai kaupan kateprosentin ja tuotteen inventaariohinnan avulla. Kateprosentin avulla tapahtuva laskenta on hierarkinen, eli jos tuotteelta löytyy kateprosentti, käytetään sitä ja jätetään huomioimatta kaupan ja kategorian kateprosentit. Metodi palauttaa hinnan aina verollisena, joten viimeisessä vaiheessa tarkistetaan ovatko tuotehinnat tietokannassa verollisia vai verottomia, ja jos hinnat ovat verollisia, niin lasketaan veron osuus mukaan palautettavaan arvoon.

3. Jos tuotteen hinnanalennus on sallittu ja `getPrice()`-kyselyssä pyydetään huomioimaan alennukset, niin seuraavaksi lasketaan alennushinta. Alennushinta voidaan asettaa tuotteelle kiinteästi tai alennusprosentin kautta, jolloin alennushinta lasketaan tuotteen perushinnasta. Alennusprosentti voidaan asettaa myös kategorialle ja kaupalle. Alennushinnat lasketaan sekä oletuskäyttäjäryhmälle, että omalle käyttäjäryhmälle, jos sellainen annettiin `getPrice()`-metodille

syöteparametrina. Lopulliseksi alennushinnaksi määräytyy alin hinta. Huomioitavaa on, että jos käyttäjän oman käyttäjäryhmän alennushinta on kalliimpi kuin oletuskäyttäjäryhmän, niin silloin valitaan oletuskäyttäjäryhmän alennushinta. Jos näin ei tehtäisi, niin erityisen käyttäjäryhmän kävijälle tarjottaisiin eri hintaa riippuen siitä, onko kävijä kirjautunut kauppaan sisään vai ei.

4. Neljännessä vaiheessa summataan, mahdollisesti alennuksen sisältävään, tuotehintaan mahdollisten tuotevariaatioiden tuomat lisähinnat.

5. Viimeisessä vaiheessa palautetaan hinta joko verollisena tai verottomana riippuen `getPrice()`-metodin kolmannesta syöteparametrista.

Liitteissä 5, 6, 7, 8 ja 9 on esitetty tuotehinnan laskentalogiikan vuokaaviot.

Sovelluksen hinnoittelulogiikan rajoituksena on, että tuotteella pitää olla kategoria. Syynä on sovelluksen tietokanta-arkkitehtuuri, jossa kauppa sisältää kategorioita ja kategoriat tuotteita. Itse tuotteella ei ole tietoa, minkä kaupan tuote se on. Lisäksi, jos tuotteella ei ole kategoriaa, niin tämän hetken ylläpidon näkymissä tuotetta ei voida hallinnoida, joten tästäkin syystä kategorioimattoman tuotteen hinnoittelu on tarkoitukseton tilanne. Luonnollisesti hintaa ei voida myöskään laskea, jos tiedot ovat puutteelliset. Tälläinen tilanne esimerkiksi on, jos tuotteella ei ole kiinteää hintaa eikä kateprosenttia ja kaupan asetuksista on estetty hinnan laskeminen kategorian tai kaupan kateprosentin avulla.

Tuotelistausten järjestäminen tuotteiden hinnan perusteella on ongelmallista, koska yhden tuotteen hinnan laskeminen vaatii useita tietokantakyselyitä, ja jos halutaan järjestää esimerkiksi kahden tuhannen tuotteen kategorian tuotteet hintajärjestykseen, niin kyselyitä pitää tehdä kymmeniä tuhansia. Tarkalleen ottaen kyselyitä tehdään yhdellä `getPrice()`-pyynnöllä seuraaviin tauluihin: `shop_setting`, `product`, `product_localization`, `account`, `shop_account`, `shop_role` (kolmesti), `shop_role_permission` (kahdesti), `category_products`, `category` (kahdesti), `category_locazalition`, `shop_role_product_discount` (kahdesti), `shop_role_category_discount` (kahdesti). Tietokannoista on kerrottu tarkemmin luvussa 4.1. Ratkaisun ongelmaan tuonee aikanaan välimuistin käyttäminen hinnoittelulogiikassa.

4.7. Toimituskulujen laskentalogiikka

Toimituskulujen laskentalogiikka on ominaisuutena monitahoinen, koska kauppojen pitäjät haluavat hinnoitella toimituskuluja hyvin vaihtelevasti eri tavoilla. Joku kauppias haluaa kiinteät toimituskulut, toinen määrätä tuotteiden painon mukaan, kolmas tilauksen hinnan mukaan ja niin edelleen.

Tässä työssä kehitetyssä verkkokauppasovellusalustassa on mahdollista asettaa eri toimitustavoille erilaisia hinnoitteluperiaatteita. Jokaisella tuotteella on niin sanottu toimituskuluyksikkömäärä, jota hyväksi käyttämällä toimituskulujen hinta voidaan laskea. Toimituskuluyksikön voi mieltää esimerkiksi painona, eli mitä painavampi tuote on, sen enemmän se vaikuttaa toimituskulujen hintaan. Televisio voisi olla esimerkiksi 20 toimituskuluyksikköä ja kirja kaksi toimituskuluyksikköä. Se, miksi sovelluksessa ei käytetä suoraan painoa toimituskulujen määräytymisperusteena on tilan-

ne, jossa tuote on kevyt, mutta hankalan muotoinen tai muuten vaatii poikkeuksellisia toimia, kuten helposti särkyvä taulu. Tälläisessä tilanteessa taululle voidaan määrätä suurempi toimituskuluyksikkömäärä kuin muille samanpainoisille tuotteille.

Se, miten lopullisten toimituskulujen hinnan suuruus lasketaan, voidaan asettaa porrastetulla taulukolla. Voidaan määrätä, että toimituskulujen suuruus on esimerkiksi 10 euroa aina 100 toimituskuluyksikköön asti, 20 euroa 500 toimituskuluyksikköön asti ja 30 euroa 100000 toimituskuluyksikköön asti. Tämä tarkoittaisi käytännössä, että 20 toimituskuluyksikön televisioita myytäessä toimituskulujen suuruus olisi 10 euroa viiteen televisioon asti. Vastaavasti 20 euron toimituskuluilla voisi ostaa 25 televisiota ja sen jälkeen toimituskulut olisivat käytännössä aina 30 euroa. Jos joku ostaa 5000 televisiota, niin tilanne on jo muutenkin niin epänormaali, että toimituskuluilla ei luultavasti olisi enää merkitystä. Porrastuksen voi luonnollisesti rakentaa mieleisekseen.

Sovelluksessa voidaan toimituskulujen määräytymisperusteena käyttää myös tuotteiden hintaa. Tällöin toimituskuluyksiköillä ei ole merkitystä, vaan tuotteiden toimituskuluyksikkösumma määräytyy suoraan tilauksen loppuhinnasta. Tämä ominaisuus mahdollistaa “toimituskulut 10 % tilauksen hinnasta” -tyyppisen hinnoittelun.

Sovellukseen voidaan asettaa myös kiinteä pohjahinta eli niin sanotut käsittelykulut. Tätä ominaisuutta hyväksi käyttäen voidaan määrätä kaupalle kiinteät toimituskulut: asetetaan vain haluttu kiinteä hinta käsittelykuluiksi ja jätetään huomioimatta tuotteiden toimituskuluyksiköt. Käsittelykulut voidaan myös asettaa nollaksi, jolloin kaupassa on ilmaiset toimituskulut. Lisäksi on mahdollista määrätä tilauksen arvolle raja, jonka ylittyessä toimituskulut ovat ilmaiset, mikä mahdollistaa “yli 100 euron tilauksissa ilmaiset toimituskulut” -tyyppiset toimitusehdot.

Seuraavassa vielä listattuna mahdolliset erilaiset toimituskulujen hinnoittelulogiikat:

- Ilmaiset toimituskulut.
- Ilmaiset toimituskulut tietyn suuruisille tilauksille, joita pienemmille hinta määräytyy halutuin perustein.
- Kiinteät toimituskulut.
- Porrastettu toimituskulujen määräytyminen toimituskuluyksikköjen perusteella. Käsittelykulujen avulla voidaan määrätä toimituskulujen minimisuuruus. Myös maksimihinta voidaan määrätä porrastuksen avulla.
  - Toimituskuluyksiköt voivat määräytyä esimerkiksi tuotteiden painon, tilavuuden tai muodon perusteella. Tällöin voitaisiin esimerkiksi suoraan punnita tuotteet ja muuntaa painot toimituskuluyksiköiksi ja tehdä sen jälkeen korjauksia yksittäisten tuotteiden kohdalla (esimerkiksi taulu), jossa tuotteen paino ei vielä korreloi tarpeeksi hyvin toimituskulujen suuruuden kanssa.
  - Toimituskuluyksiköt voivat määräytyä myös tuotteiden lukumäärän perusteella. Tällöin jokaiselle tuotteelle asetettaisiin toimituskuluyksiköksi yksi (1). On tavallista, että kauppias hinnoittelee toimituskulut juuri ostettujen tuotteiden määrän perusteella.
  - Myös tuotteiden hinta voidaan asettaa toimituskuluyksiköksi, jolloin toimituskulujen suuruus määräytyy suoraan tilauksen hinnasta.

Toimitustapojen hallinnassa ei voida asettaa tilauksen minimitoimituskuluyksikköjen määrää, mikä tarkoittaa, että toimitustapa olisi saatavilla vasta kun tilauksen suuruus olisi tarvittavan suuruinen. Saman toiminnallisuuden pystyy kuitenkin toteuttamaan näkymissä, joissa voidaan estää tilaustapahtumassa eteneminen, jos tilauksen suuruus on liian pieni. Tällaisessa ratkaisussa ei tosin pystytä huomioimaan toimitustapaa, eli rajoitus olisi voimassa kaikille toimitustavoille.

Sovelluksessa ei ole myöskään mahdollista tehdä toimituskulujen laskentaa - eikä täten näyttää asiakkaalle lopullista toimituskulujen suuruuttaa - ennenkuin tuote on lisätty ostoskoriin ja tilaustapahtumassa on siirrytty yhteenvetoon, joka on tilaustapahtuman kolmas vaihe. Kolmanteen vaiheeseen päästäkseen myös tilaajan tiedot pitää olla annettuna. Tämän rajoituksen perusongelma on, että käyttäjän haluama toimitus- ja maksutapa pitää olla tiedossa. Jos kaupassa ei ole vaihtoehtoja toimitus- ja maksutavoille eivätkä toimituskulut määräydy esimerkiksi vastaanottajan asuinpaikan perusteella, niin toimituskulujen arviointi olisi periaatteessa mahdollinen jo aikaisemmassa vaiheessa. Tälläinen ominaisuus on vaikea toteuttaa rikkomatta nykyisen toimintalogiikan selkeyttä ja yksinkertaisuutta liikaa, joten sitä ei ole ainakaan vielä toteutettu. Käytäntö on osoittanut, että riittävän hyvä ratkaisu on esittää näkymissä asiakkaille yksinkertaisesti toimituskulujen määräytymisperuste, esimerkiksi "toimituskulut alkaen 5 euroa".

Teoria-luvussa arvioitavana olleessa Magentossa on mahdollista määrätä myös toimituskulujen suuruus tilauksen vastaanottajan etäisyyteen perustuen. Tällöin voidaan esimerkiksi määrätä ulkomaille toimitettavien tilausten hinnan olevan suurempi kuin kotimaan toimitusten. Lisäksi Magentossa voidaan rajata toimitustapa sallituksi vain tiettyihin maihin. [26]

Etäisyyksiin perustuvan toimituskulujen laskennan lisäksi kehitettävään verkkokauppasovellusalustaan voitaisiin harkita toteutettavaksi ominaisuus, jonka avulla voitaisiin rajata toimitustavan maksimitoimituskuluyksikköjen määrä. Tällöin voitaisiin määrätä, ettei tietyllä toimitustavalla olisi mahdollista toimittaa rajattoman suuria tilauksia.

Porrastetun hinnoittelun lisäksi voisi olla tarvetta myös asettaa erityinen kerroin, jonka avulla toimituskulujen suuruus lasketaan toimituskuluyksiköistä. Eli esimerkiksi jos kerroin olisi 0.05 ja tilauksen toimituskuluyksikkösumma 20, niin toimituskuluiksi tulisi yksi euro, jos käsittelykuluja ei ole asetettu. Tämän hyvä puoli olisi toimituskulujen suuruuden portaaton määräytyminen, eli pienikin muutos tilauksen hinnassa vaikuttaisi toimituskulujen suuruuteen. Toisaalta huono puoli olisi lineaarisuus, eli kauppias ei pystyisi tarjoamaan arvokkaammille tilauksille suhteellisesti halvempia toimituskuluja. Toki olisi mahdollista myös toteuttaa eräänlainen hybridi porrastetuilla kertoimilla, mutta tämä olisi monimutkaisuudessaan mahdollisesti kannattamaton ominaisuus.

4.8. Tuotteiden näkyvyyskriteerit

Tuotteiden näkyvyys julkisella puolella riippuu useista kriteereistä, joita ovat poistopäivämäärä, saatavuusaika, varastosaldo, ovatko ennakkomyynti ja loppuunmyytynä myyminen sallittuja ja onko tuote, tuotteen kategoria tai kategorian yläkategoria piilotettu. Kompleksista ehtolauseketta on ehkä helpointa lähestyä kysymyssarjalla: jos

jokaiseen kysymykseen vastataan *ei*, niin tuote näkyy julkisella puolella. Vastaavasti ylläpidossa tuote näkyy, jos vähintään ensimmäiseen kohtaan vastataan *ei*.

1. Onko tuote merkitty poistetuksi?
2. Onko tuote piilotettu?
3. Onko tuotteen kategoria tai jokin tuotteen kategorian yläkategorioista piilotettu?
4. Jos varastosaldo on nolla, niin onko loppuunmyytynä myyminen estetty?
5. Jos saatavuusajan alkamispäivämäärä on asetettu myöhemmäksi kuin nykyhetki, niin onko ennakkomyynti estetty?
6. Onko saatavuusajan loppumispäivämäärä sama tai pienempi kuin nykyhetki?

4.9. Tietokanta-abstraktio

Liaani Framework -ohjelmistokehyksessä tietokanta-abstraktio on toteutettu käyttäen PHP 5:n tarjoamaa PHP Data Objects (PDO) -laajennusta. Lisäksi Liaanin omaan PDO-luokkaan on toteutettu muutamia parannuksia, jotka mahdollistavat paremmin saman ohjelmakoodin käytön eri tietokantahallintajärjestelmillä. Pääosin tietokannan kanssa tosin operoidaan Liaanin `Model`- ja `ModelIterator`-luokkien avulla, jolloin Liaani hoitaa varsinaisten SQL-kyselyt. Tätä tekniikkaa kutsutaan oliorelaatiokuvaukseksi (eng. Object-relational Mapping, ORM).

Käytännössä Liaanissa ORM:n idea on, että luodaan relaatioille oma malli, jossa kerrotaan relaation nimi, listataan attribuutit ja kuvataan relaation assosiaatioita muihin relaatioihin. Kun tämä on tehty, voidaan mallista tehdä olio ja pyytää oliolta tietoa, jolloin palvelun tarjoaja - tässä tapauksessa Liaani - hakee tiedon tietokannasta. Vastaavasti olioon voidaan tallentaa tietoa ja pyytää järjestelmää tallentamaan sen tietokantaan. Mallissa kuvataan yleensä myös relaation iteraattori, jonka avulla voidaan käydä läpi useita samanlaisia relaatioita. Jos malli kuvaisi vaikka tuotetta, niin mallin iteraattori listaisi tuotteita. Iteraattorilta voidaan pyytää kaikki tuotteet tai rajoittaa hakua erilaisilla ehdoilla ja suodatuksilla. ORM nopeuttaa huomattavasti kehittäjän työtä, kun tavanomaisia SQL-kyselyjä ei tarvitse jatkuvasti kirjoittaa.

Tietokannan kanssa halutaan kuitenkin ajottain operoida ORM:n ohi, kun tarvitaan monimutkaisempia SQL-kyselyitä tai ORM:n tuottamat kyselyt eivät ole tarpeeksi optimaalisia. Tällöin käytetään PHP:n PDO-laajennusta.

5. MITTAUKSET

Työn mittausosiossa tehdään uudestaan katsaus Teoria-luvussa arvioitavina oleviin kolmeen verkkokauppasovelluslustaan ja tutkitaan tarkemmin niiden ominaisuustarjontaa, jota verrataan tässä työssä kehitettävän sovelluksen ominaisuuksiin. Näin saataneen karkea kuva, missä vaiheessa kehitys on ominaisuuksiensa puolesta muihin markkinoilla oleviin sovelluksiin nähden. Lisäksi lasketaan ja analysoidaan kehitettävän sovelluksen ohjelmakoodirivimääriä sekä tutkitaan, kuinka sovellusta on muutettu ajan kuluessa.

5.1. Ominaisuusvertailu

Liitteessä 1 listataan arvioitavien verkkokauppasovellusalustojen tarjoamia ominaisuuksia. Lista ei ole kattava pelkästään jo siitä syystä, että eri ominaisuudet on vaikea määrittää ja rajata tarkasti, minkä lisäksi eri sovelluksissa tiettyjä toiminnallisuuksia on toteutettu hyvin eri tavoin. Lista osoittaa kuitenkin karkeasti eri sovellusten erot. Ominaisuuksien järjestys on määrätty niin, että ensimmäisenä ovat ominaisuudet, jotka ovat kaikilla kolmella arvioitavilla verkkokauppasovellusalustoilla, seuraavana sellaiset ominaisuudet, jotka löytyvät kahdesta sovelluksesta ja lopulta ominaisuudet, jotka löytyvät vain yhdestä sovellusalustasta. Listalla neljäntenä on tässä työssä kehitettävä verkkokauppasovellusalusta, ja sitä merkitään kirjainlyhenteellä VS (ViidakkoStore). ViidakkoStoren ominaisuudet eivät ole vaikuttaneet listan järjestykseen.

Listalta on rajattu mielivaltaisesti pois itsestäänselvät ja epäkiinnostavat ominaisuudet, jotka sisältyvät käytännössä kaikkiin markkinoilla oleviin verkkokauppasovellusalustoihin. Listalla *automaattinen* tarkoittaa, että sovellus muodostaa informaation todellista datasta, johon käyttäjä ei voi vaikuttaa. Esimerkiksi *suosittujen tuotteiden* kohdalla tämä tarkoittaa, että suositut tuotteet ovat todellakin niitä, joita ihmiset ovat eniten ostaneet, eikätkä sellaisia, joita myyjä on merkinnyt suosituiksi.

Vertailussa saadaan yleiskäsitys sovellusten ominaisuustarjonnasta. Nähdään, että Magento sisältää selkeästi kattavimmin ominaisuuksia, seuraavaksi eniten ISC ja viimeisinä ovat osCommerce ja ViidakkoStore. Tätä tarkempaa johtopäätöstä on turha tehdä, koska listan ominaisuudet on valikoitu niin mielivaltaisesti. Lista antaa osviittaa siitä, mitä ominaisuuksia nykyaikaiselta verkkokauppasovellusalustalta vaaditaan.

ViidakkoStoresta vielä puuttuvia ominaisuuksia tullaan tulevissa kehityskeskusteluissa arvioimaan ja sen myötä mahdollisesti myös toteuttamaan. Erityisen kiinnostavia vertailussa ovat ominaisuudet, jotka löytyvät kaikista kolmesta arvioitavana olevasta sovellusalustasta, mutta puuttuvat ViidakkoStoresta. Seuraavassa tarkastellaan näitä tarkemmin (tilanne 11. joulukuuta 2009).

Uutiskirje: Toteutusta ei ole suunniteltu, joten todennäkösesti ominaisuutta ei tulla tekemään lähitulevaisuudessa.

Ostetuimmat tuotteet (automaattinen): Ominaisuus olisi helppo toteuttaa, mutta sille ei ole ollut vielä ainakaan kysyntää. Dataa, jota toiminnallisuus käyttäisi, kerätään jo.

Tuotearvostelut: Ominaisuutta, jossa käyttäjät voisivat kommentoida tuotteita, ei näillä näkymin tulla toteuttamaan lähiaikoina. Ominaisuus vaatii paljon huolellista suunnittelua, koska kommenteille tarvitaan hallintatyökalut (eng. moderating tools).

Tuotteiden pisteytys: Tuotteiden pisteytystä ei ole myöskään suunniteltu toteutettavan lähitulevaisuudessa.

Kehittynyt haku: Julkipuolen kehittynyt haku, jota voisi rajata esimerkiksi hinnan, saatavuuden ja avainsanojen mukaan, on tärkeä ominaisuus, jonka toteutus on aikataulutettu seuraavaan kehityspyrähdykseen.

Digitaaliset tuotteet: Kehityksen alusta asti on ollut selvää, että digitaalisille tuotteille tullaan tekemään tuki, eli että ostaja voi maksamisen jälkeen välittömästi ladata ostamansa tuotteen. Ominaisuus tullaan toteuttamaan todennäköisesti sitten, kun ensimmäinen asiakas tarvitsee sitä.

Monivaluuttatuki: Lokalisoitavan verkkokauppasovelluksen tulee tukea useita valuuttoja yhtä aikaa. Sen takia tämä ominaisuus tullaan varmasti toteuttamaan lähiaikoina.

Asiakas voi seurata tilauksensa tilaa: Ominaisuus on harkinnassa, mutta sitä ei ole vielä aikataulutettu.

Asiakas näkee tilaushistoriansa: Ominaisuus on aikataulutettu seuraavaan kehityspyrähdykseen, eli se tullaan toteuttamaan lähiaikoina.

Raportointi parhaiten myyvistä tuotteista: Sovelluksen raportointi nojaa tällä hetkellä pelkästään Google Analyticsin varaan, mutta sovelluksen omat monipuoliset raportointityökalut graafeineen tullaan toteuttamaan lähitulevaisuudessa.

Raportointi useimmiten katsotuista tuotteista: Käsitelty edellisessä kohdassa.

Raportointi tilausten tuotoista: Käsitelty edellä.

Havaitaan, että puuttuvista ominaisuuksista, jotka löytyvät kaikista muista arvioitavana olleista sovelluksista, useimmat tullaan toteuttamaan lähiaikoina ViidakkoStoreen. Voidaankin olettaa, että kehittyessään ViidakkoStoresta tullee ainakin ominaisuuksiensa puolesta varteenotettava kilpailija markkinoiden muille verkkokauppasovellusalustoille.

5.2. Rivimääräanalyysi

Sekä tässä diplomityössä toteutettavalle että Teoria-luvussa arvioitavana oleville kolmelle verkkokauppasovellusalustalle suoritettiin rivimääräanalyysi erityisellä cloc-ohjelmalla [1], joka laskee annetun hakemiston sisältämien tiedostojen määrän sekä tutkii hakemiston tiedostojen sisältöä. Ohjelman tuottamasta loppuraportista (liite 10) käy

ilmi, montako tiedostoa on milläkin ohjelmointikielellä kirjoitettu sekä kuinka monta tyhjää riviä, kommenttiriviä ja varsinaista ohjelmakoodiriviä tiedostoissa on.

Seuraavassa taulukossa esitetään yhteenvetona arvioitavien verkkokauppasovellusalustojen rivimääräanalyysin tulokset. Taulukossa on tiedostojen lukumäärä, tyhjien rivien lukumäärä, kommenttirivien lukumäärä, varsinaisten ohjelmarivien lukumäärä, kommenttirivien osuus suhteessa ohjelmarivien lukumäärään sekä montako riviä ohjelmakoodia on keskimäärin yhdessä tiedostossa. Nimet on lyhennetty niin, että OSC tarkoittaa osCommercea, ISC Interspire Shopping Cartia ja VS ViidakkoStorea eli tässä diplomityössä kehitettävää sovellusalustaa.

Taulukko 2. Rivimääräanalyysi

	OSC	**Magento**	**ISC**	**VS**
Tiedostoja	571	5659	1350	484
Tyhjiä rivejä	10186	80160	42205	11310
Kommenttirivejä	6233	318912	40608	7874
Ohjelmakoodirivejä	65773	654220	204719	59399
Kommenttien osuus	9.5%	48.7%	19.8%	13.3%
Riviä tiedostossa keskimäärin	115	115	152	162

Nähdään, että Magentossa on noin kolminkertainen määrä ohjelmakoodirivejä suhteessa Interspire Shopping Cartiin, vaikka ominaisuuksien määrässä ero Magenton eduksi ei ole kovin suuri, kuten luvussa 5.1 todettiin. Myös tiedostojen määrä on Magentossa huomattavan suuri, mikä tukee Teoria-luvussa esitettyä väitettä sen ohjelmistoarkkitehtuurin vaikeaselkoisuudesta.

Havaitaan myös, että varsinkin Magentossa ja jossain määrin myös ISC:ssä kommenttirivien osuus on muita sovelluksia huomattavasti suurempi. Kommentoinnilla helpotetaan projektin ulkopuolisten ihmisten sisäänpääsyä sovellukseen, mutta myös muistutetaan kehittäjiä itseään, mikä missäkin on tarkoituksena. Erityisesti vapaiden ohjelmalisenssien sovelluksissa kommentoinnilla on iso merkitys, koska sovellusta päätyvät käyttämään useat monentasoiset käyttäjät. Tältä pohjalta tarkasteltuna voidaan todeta, että osCommercessa kommentointi on liian vähäistä, mutta myös ViidakkoStoressa tulisi kasvattaa kommentoinnin määrää lähemmäs ISC:n tasoa, joka on otettu kehityksessä esikuvaksi muissakin yhteyksissä.

Luvussa 5.1 huomattiin, että osCommercen ominaisuusmäärä on samaa suuruusluokaltaan ViidakkoStoren kanssa. Tästä syystä kyseisten sovellusten rivimääräanalyysejä voidaan verrata ja todeta, että ViidakkoStoressa on toteutettu pienemmällä ohjelmakoodirivimäärällä sama määrä ominaisuuksia. Arvio on hyvin karkea, mutta voitaneen silti olettaa, että tältä osin kehitys on ollut tähän asti hyvää ja näinollen voitaneen jatkaa samaan tapaan.

[1] Ohjelma saatavilla osoitteesta http://cloc.sourceforge.net/

5.3. Muutokset ohjelmakoodikannassa

Työssä toteutettiin erityinen skripti (eng. script), joka käy läpi kaikki sovelluksen versiohallinnan revisionit ja tutkii niistä, kuinka monta riviä ohjelmakoodia on versiohallintajärjestelmään lisätty ja poistettu missäkin vaiheessa. Sen jälkeen perustelluista syistä raakadataa muokataan hieman ja esitetään data havainnollisemmassa muodossa. Lopuksi analysoidaan olennaisia kohtia datassa ja tehdään niistä johtopääksiä.

Skripti toimii niin, että se ajaa ensin syötteenä annetulle hakemistolle `svn log -r 1:HEAD` -komennon, joka tutkii, mitä revisionumeroita versiohallinnassa on. Sen jälkeen skripti käy läpi kaikki revisiot ja tutkii, montako riviä lisättiin ja poistettiin missäkin revisiossa. Scriptin vaste tulostetaan XML-tiedostoon, josta se on muunnettu listaksi ja esitetty liitteessä 11. Liitteen taulukossa *Rev* tarkoittaa revisiota, *Ins* lisättyjen rivimäärien lukumäärää ja *Del* poistettujen rivimäärien lukumäärää.

Nähdään, että revisioissa 462, 536, 539, 540, 546, 548 ja 551 on lisätty tai poistettu suuria rivimääriä. Kyseessä on ulkopuolinen laajennus, TinyMCE, jonka lisääminen versiohallintaan tuotti ongelmia ja on sen takia versiohallinnan historiassa useita kertoja poistettu ja lisätty. TinyMCE sisältää huomattavan paljon ohjelmakoodirivejä. Näistä syistä raakadataa käytetään niin, että kaikki yli 35000 rivin lisäykset ja poistot jätetään huomioimatta.

Tämän jälkeen muokatusta raakadatasta lasketaan lisättyjen rivien summa ja poistettujen rivien summa aina kymmenen revisionin välein ryhmiteltynä, eli revisionit 1-10 kuuluvat ensimmäiseen ryhmään, 11-20 toiseen ja niin edelleen. Summaus vähentää yksittäisten piikkien vaikutusta kokonaiskuvaan. Saadusta datasta piirretään kuvaaja, joka on esitetty liitteessä 12.

Otetaan mielivaltaisesti tarkastelun kohteeksi raja-arvon tuhat poistettua koodiriviä ylittävät revisioryhmät. Kuvaajasta nähdään, että raja-arvo ylitetään kohdissa 90, 130, 230, 270, 290, 300, 310, 330, 660 ja 720. Aina kun versiohallinnasta on poistettu paljon ohjelmakoodia, voidaan esittää epäily, että poiston syynä on huono ohjelmistosuunnittelu. Seuraavassa tutkitaan tarkemmin kyseiset kohdat versiohallintahistoriasta ja selvitetään poistojen syyt. Pohditaan myös, paljastavatko ne huonoa suunnittelua, joka olisi ollut vältettävissä. Tutkimisessa käytetään hyväksi komentoja `svn log` ja `svn diff`. Esimerkiksi ensimmäiselle kohdalle tutkimista tehdään komennoilla `svn log -v -r 81:90`, `svn diff -r 81:90` ja `svn diff -r 81:90 | diffstat`.

81-90: Kohta sisältää paljon pieniä korjailuja, joista osa olisi jäänyt tekemättä paremmalla ohjelmointikokemuksella. Lisäksi revisioiden aikana ylläpitopuolelle on asennettu uusi ulkoasunäkymä, minkä johdosta paljon ohjelmakoodia liittyen vanhaan ulkoasuun on poistettu. Ylläpidon näkymä on kehityksessä suunniteltu toteutettavaksi niin, että aluksi ulkoasusta ei välitetä ollenkaan, sen jälkeen toteutetaan tyydyttävä ratkaisu ja vasta joskus myöhemmin tehdään lopullinen ulkoasu. Tässä kohdassa on kyse toisesta vaiheesta.

121-130: Kohta sisältää jälleen paljon pieniä korjauksia, jotka olisi olleet isolta osin vältettävissä paremmalla ohjelmointikokemuksella ja ohjelmistokehyksen kattavamalla dokumentaatiolla. Toisaalta tämä projekti on ollut toteuttajalle myös ohjelmointikoulutusta, joten tietty määrä virheitä on hyväksytty osana proses-

sia. Tässä vaiheessa ongelmakohtia ovat korjanneet myös kaksi kokeneempaa kehittäjää.

221-230: Tässä kohdassa versiohallintaan lisättiin virheellisesti julkisen puolen sivupohjat, joita generoidaan automaattisesti, kun sivupohjia tai lisäkkeitä muokataan ulkoasuhallinnassa. Sivupohjat ovat kauppakohtaisia, eikä niitä haluta versiohallintaan. Virheen syynä oli väärinymmärrys, jonka mukaan versiohallintaan halutaan lisätä kaupan oletusulkoasu. Sivupohjien lisäyksen takia myös poistoja tehtiin paljon.

261-270: Edelleen poistoja aiheuttamassa edellisessä piikissä (221-230) lisätyt sivupohjat.

281-290: Edelleen julkisen puolen sivupohjat ovat virheellisesti versiohallinnassa aiheuttamassa runsaasti muutoksia. Lisäksi tässä vaiheessa siirrettiin tiedostoja versiohallinnan hakemistosta toiseen, mikä aiheutti paljon muutoksia versiohallintaan. Tämän olisi voinut välttää tekemällä projektin alussa tarkemman suunnitelman tiedostojen sijoituksesta hakemistoihin. Ongelmia ovat aiheuttaneet lähinnä staattisten tiedostojen, kuten tyyliohjeiden ja ulkoasuun liittyvien kuvatiedostojen sijoittelut.

291-300: Edelleen automaattisesti generoidut sivupohjat ovat aiheuttamassa paljon muutoksia versiohallinnassa.

301-310: Tässä vaiheessa julkisen puolen sivupohjat poistettiin versiohallinnasta.

321-330: Julkisen puolen sivupohjien staattisia tiedostoja poistettiin vasta tässä vaiheessa.

651-660: Vaiheessa 81-90 käsiteltiin ylläpidon ulkoasua. Tässä kohtaa versiohallintaan lisättiin kolmas ulkoasu eli niin sanottu lopullinen ulkoasu. Päivityksen myötä vanha ulkoasu poistettiin. Tämä vaihe oli suunniteltu.

711-720: Osa vanhasta ulkoasusta oli jäänyt vielä versiohallintaan, ja tässä kohtaa loputkin poistettiin. Poisto oli suunniteltu, tosin se olisi voinut tapahtua jo aiemmin.

Tarkastelu toi ilmi lähinnä ongelmia, jotka liittyivät tiedostojen sijoittamiseen sovelluksen hakemistopuuhun. Ongelmia olisi voitu välttää paremmalla suunnittelulla ja kommunikoinnilla, jolloin hakemistorakenne olisi kerralla saatu oikeanlaiseksi. Toisaalta kyseessä ei ole ajallisesti iso ongelma, koska muutokset ovat tulleet pääosin kokonaisten tiedostojen siirtelystä.

Ohjelmakoodia on muutamissa kohdissa myös parannettu sekä kehittäjän itsensä, että kokeneempien kehittäjien toimesta. Näitä olisi voitu välttää paremmalla ohjelmointikokemuksella, mutta koska yksi projektin tarkoituksista oli kouluttaa sovelluksen pääkehittäjää, niin tietty määrä virheitä on katsottu kuuluvaksi prosessiin.

Jos tutkittaisiin tarkemmin, mitkä yksittäiset toimintalogiikat tai ohjelmakoodirivit ovat vaatineet uudelleenkirjoittamista, esimerkiksi jättämällä huomioimatta kokonaisten tiedostojen lisäämiset ja poistamiset versiohallintaan, voitaisiin mahdollisesti paremmin selvittää suunnitteluvirheitä ja pohtia miltä osin näitä virheitä olisi

voitu välttää. Lisäksi tarkempi analyysi voisi selventää sovelluksen kehittäjien suurimpia ongelmakohtia. Tätä tietoa voitaisiin hyödyntää esimerkiksi seuraavan, vastaavassa tilanteessa olevan kehittäjän tehtäväkuvassa. Varsinkin Liaani Framework -ohjelmistokehyksen kanssa aloitteleville kokemattomille kehittäjille tiedosta voisi olla hyötyä.

6. POHDINTA

Diplomityön tarkoitukseksi ja selkeäksi päätavoitteeksi asetettiin verkkokauppasovellusalustan toteuttaminen perusominaisuuksilla ja -toiminnallisuuksilla. Tarkemmin tämän ajalteltiin tarkoittavan tuotteiden ja kategorioiden, ulkoasun ja tilauksien hallintaa sekä yksinkertaisia raportteja ylläpitopuolelle. Lisäksi harkittiin käyttäjien ja käyttäjäryhmien hallinnan toteuttamista. Vastaavasti julkipuolelle asetettiin tavoitteeksi rakentaa demokauppa, jossa tuotteita voidaan selata kategorioittan, lisätä tuotteita ostoskoriin ja tehdä tilauksia. Kaikki edellämainitut ominaisuudet toteutettiin lukuunottamatta raportointia, jonka kehitys on siirretty seuraavaan vaiheeseen muuttuneiden suunnitelmien mukaisesti. Koska sovellus on tuotantokäytössä viidessä eri kaupassa (tilanne 11. joulukuuta 2009), voidaan vahvasti olettaa, että päätavoitteessa on onnistuttu. Lisäksi tällä hetkellä kehityksessä ja testauksessa on kuusi kauppaa, jotka tullaan siirtämään tuotantoon lähitulevaisuudessa.

Itse sovellukselle asetettiin tavoitteeksi myös monia muita erikoisempia ominaisuuksia. Sovelluksessa tulee olla monikielisyystuki, joka ei saa perustua käännöskuviin, ja oletustekstejä täytyy voida myös vaihtaa. Tässä onnistuttiin: sekä sivupohjien että tuotteiden ja kategorioiden tekstejä voidaan kääntää useille kielille ja tehdä kaupasta näin monikielinen. Kaupan kokonaisvaltaisessa lokalisoinnissa on vielä puutteita muun muassa valuuttojen, päivämäärien, tekstin suunnan ja listojen järjestämisen osalta, mutta näitä ei tavotteissa listattu. Sovellukseen toteutettiin myös yksinkertainen sisällönhallintajärjestelmä, joka oli yksi tavotteista. Lisäksi sovellukseen on toteutettu tuotetuonti Clover Shop -verkkokauppasovellusalustasta sekä integroinnit GNT:n, BookMaster:n, ja Sivuviidakon kanssa. Sovelluksen rajapintoja ei ole dokumentoitu. Myös sovelluksen auditointi on vielä puutteellinen: kaupan kävijöiden ja ylläpitäjien toimia ei juurikaan tallenneta. Tietoa kerätään tällä hetkellä (11. joulukuuta 2009) vain kirjautumisista, tilausten tilojen muuttamisesta sekä ostoskoriin kerätyistä tuotteista riippumatta siitä tilataanko tuotteita vai ei.

Sovellukseen haluttiin myös dokumentoitu rajapinta, jonka avulla voidaan integroida markkinoilla olevia seurantaohjelmia. Tätä dokumentaatiota ei ole toteutettu, mutta integrointi Google Analyticsin kanssa on juuri testattavana (tilanne 11. joulukuuta 2009). Sovelluksen arkkitehtuurin osalta asetettiin vaatimus, että sovelluksesta pitää saada aidosti eriytetyt paketit, jotta eri asiakkaille voidaan myydä samaa ohjelmaa eri ominaisuuksilla. Tarjolla olisi esimerkiksi kevyt, normaali ja kattava versio. Tälläistä eriyttämistä ei ole toteutettu ollenkaan, joten siltä osin tavoite on saavuttamatta. Sovelluksen haluttiin aluksi tukevan MySQL 5 ja PostgreSQL 8 sekä myöhemmin MS SQL Server -tietokantahallintajärjestelmiä. Tällä hetkellä (11. joulukuuta 2009) sovellusta on käytetty ainoastaan MySQL 5 -tietokantahallintajärjestelmällä.

Sovelluksen pitkän tähtäimen tavotteita, eli nykyisten markkinoilla olevien verkkokauppasovellusalustojen puutteiden täyttämistä, mainostoimistojen tarpeiden parasta mahdollista palvelemista ja sovelluksen asentamisen ja kehittämisen yksinkertaisuutta ei voida tässä ohjelmakehityksen vaiheessa arvioida. Myöhemmin nähdään, miten näissä onnistuttiin.

Työn onnistumista tulee arvioida juuri tavoitteiden saavuttamisen näkökulmasta, mutta sen lisäksi työssä toteutettiin erillisiä mittauksia, joilla arvioidaan tavoitteissa onnistumisen sijaan sovelluksen tiettyjä piirteitä. Mittaukset-luvussa tutkittiin tässä työssä kehitettävän sovelluksen lisäksi kolmen markkinoilla olevan verkkokauppaso-

velluksen ominaisuuksia ja tehtiin lähinnä määrällistä vertailua. Lisäksi tehtiin rivimääräanalyysi sovellusten ohjelmakoodikannalle sekä tutkittiin kehitettävän sovelluksen ohjelmakoodirivien poistamisen syitä versiohallinnasta.

Ominaisuusvertailussa havaittiin, että kehitettävän sovelluksen ominaisuudet ovat määrällisesti vielä vähäiset verrattuna kahteen suureen verkkokauppasovellusalustaan, mutta jatkuvan kehitystyön tuloksena ominaisuuksien määrä lisääntyy jatkuvasti. Rivimääräanalyysissa havaittiin, että kommenttien osuutta olisi todennäköseisesti hyvä lisätä - viimeistään siinä vaiheessa, kun ohjelmistokehitykseen osallistuu yrityksen ulkopuolisia tahoja. Lisäksi havaittiin, että ominaisuusmäärä suhteessa ohjelmakoodien rivimäärään on vähintäänkin samansuuruinen kuin muilla sovelluksilla - mahdollisesti jopa optimaalisempi.

Mittaukset-luvun viimeisessä osassa analysoitiin kehitettävän sovelluksen versiohallinnan historiaa ja tutkittiin, voitaisiinko paljastaa huonoa suunnittelua. Havaittiin, että ongelmia on tuottanut pääasiassa tiedostojen sijoittelu ohjelmistokehyksen sisäisessä rakenteessa, koska tiedostoja on siirrelty edes takaisin kehityksen edetessä. Analyysin tarkkuus oli liian karkea, jotta se olisi paljastanut yksittäisiä suunnitteluvirheitä. Tarkemmalla analyysilla olisi mahdollisesti voitu tutkia, mitkä yksittäiset toimintalogiikat ja ohjelmakoodirivit ovat vaatineet uudelleenkirjoittamista ja näin selvittää tarkemmin suunnitteluvirheitä ja niiden syitä. Lisäksi analyysissä voitaisiin mahdollisesti jättää kokonaan huomioimatta tiedostojen lisäämiset ja poistamiset versiohallintaan.

Tekniikan aloista erityisesti web-alalla kehitys on nopeaa. Uusia tekniikoita ja käytenteitä muodostuu koko ajan, joten yksittäisten tekniikoiden akateeminen tutkiminen jää usein vähäiseksi. Tässä työssä tarkasteltiin erästä web-sovellustyyppiä, nykyaikaista verkkokauppasovellusalustaa, ja lisäksi MVC-ohjelmistoarkkitehtuuria ja web-ohjelmistokehyksiä. Nähtäväksi jää, tuleeko MVC ja web-ohjelmistokehykset vakiintumaan web-sovelluksiin pidemmäksikin aikaa, ja toisaalta miten verkkokauppasovellusalustojen kehitys jatkuu.

Toteutus-luvussa esitellyt ratkaisut tarjonnevat hyödyllistä tietoa muille verkkokauppasovellusten kehittäjille, oli kyseessä aivan uuden verkkokaupan tekeminen, tai kehittäjät, jotka tulevaisuudessa työskentelevät tässä työssä kehitetyn sovelluksen parissa. Suunnittelu-luvun ohjelmistokehyksen valintaan ja esittelyyn liittyvät osiot ovat sovellettavissa myös muiden web-sovellusten kehittämiseen.

Työn merkitys sen tilaajalle, Koodiviidakko Oy:lle, on pääasiassa uuden ohjelmiston saaminen tuotevalikoimaan. Lisäksi työn kirjallisessa osiossa on tehty katsausta markkinoiden nykytilaan ja osittain sen myötä pohdittu, miten sovelluksen kehitys voisi jatkua. Lisäksi osaa työssä tuotetusta materiaalista voidaan käyttää sovelluksen tukisivustolla.

Sovelluksen kehitys tulee jatkumaan myös tämän diplomityön jälkeen. Lähitulevaisuuden suunnitelmissa on tarkoitus toteuttaa lisää ominaisuuksia, kuten kattavia raportointi- ja lokalisointityökaluja ja monia muita pienemmän mittakaavan ominaisuuksia. Lisäksi, asiakasmäärien koko ajan kasvaessa, tullaan kehittämään uusien verkkokauppa-asennusten luomista sekä vanhojen asennusten päivittämistä helpommaksi. Lopullisista suunnitelmista päättää luonnollisesti yrityksen johto.

Jatkotutkittavaa sovelluksessa olisi mahdollisesti testauksen kehittämisen parissa ja siinä miten sovellus toimii suurissa kaupoissa suurilla tuote- ja kävijämäärillä. Sovelluksen tehokkuutta voitaisiin parantaa esimerkiksi optimoimalla ohjelmakoodia sekä lisäämällä palvelinkapasiteettia ja välimuistin käyttöä.

7. YHTEENVETO

Tämän diplomityön päätavoitteena on toteuttaa yleinen verkkokauppasovellusalusta, jonka avulla voidaan räätälöidä varsinaisia verkkokauppasovelluksia. Alustan vaatimuksena on verkkokaupan perusominaisuudet ja -toiminallisuudet, joilla päädyttiin tarkoittamaan tuotteiden ja kategorioiden, ulkoasun ja tilauksien hallintaa sekä yksinkertaisia raportteja ylläpitopuolelle. Vastaavasti julkipuolelle asetetaan tavoitteeksi rakentaa demokauppa, jossa tuotteita voitaisiin selata kategorioittan, lisätä tuotteita ostoskoriin ja tehdä tilauksia.

Kaikki edellämainitut ominaisuudet toteutettiin lukuunottamatta raportointia, jonka kehitys on siirretty seuraavaan vaiheeseen muuttuneiden suunnitelmien mukaisesti. Verkkokauppasovellusalusta on kirjoitushetkellä (18. joulukuuta 2009) tuotantokäytössä viidessä eri kaupassa, joten voidaan vahvasti olettaa, että päätavoitteessa on onnistuttu. Lisäksi tällä hetkellä kehityksessä ja testauksessa on kuusi kauppaa, jotka tullaan siirtämään tuotantoon lähitulevaisuudessa.

Yksityiskohtaisempina vaatimuksina työlle asetetaan muun muassa nykyaikainen monikielisyystuki ja ulkoasun hallinta, sisällönhallintajärjestelmä, integrointi yleisiin seurantaohjelmiin ja aidosti eriytetyt ohjelmakoodipaketit, jotta eri asiakkaille voitaisiin myydä ohjelmaa eri ominaisuuksilla. Tarjolla olisi esimerkiksi kevyt, normaali ja kattava versio. Lisäksi sovellukselle asetettiin tavoitteeksi tukea MySQL 5, PostgreSQL 8 ja myöhemmin MS SQL Server -tietokantahallintajärjestelmiä.

Sovelluksen monikielisyystuki on vielä puutteellinen: esimerkiksi käytössä olevia valuuttoja, päivämääräformaatteja, tekstin suuntaa ja listojen järjestämisiä ei ole lokalisoitu. Sen sijaan sivupohjat sekä tuote- ja kategoriatiedot voidaan kääntää useille kielille. Sovelluskehityksessä monikielisyystukea tullaan lähitulevaisuudessa laajentamaan vaatimusten mukaiseksi. Sen sijaan ulkoasun hallinta on toteutettu täysimääräisesti suunnitelmien mukaan: verkkokaupan ulkoasun pystyy rakentamaan lähes mielivaltaisesti. Ulkoasun hallintaa tullaan tarvittaessa vielä kehittämään asiakkaiden toiveiden mukaan.

Sovellukselle asetetaan tavoitteeksi myös integrointi yleisimpiin seurantaohjelmiin, kuten Google Analyticsiin ja Snoobiin sekä rajapinnan dokumentointi muiden seurantaohjelmien lisäämiselle. Tämä tavoite toteutui vain osittain: sovellus tukee suoraan Google Analyticsia, mutta ei muita seurantaohjelmia eikä rajapintaa ole dokumentoitu. Sovelluskehityksessä tullaan lähitulevaisuudessa kiinnittämään erityistä huomioita erilaisiin raportointi- ja seurantatyökaluihin.

Sovelluksen arkkitehtuurin aidosti eriyttäminen erilaisiin paketteihin on kokonaan toteuttamatta ja tältä osin alkuperäinen tavoite on saavuttamatta. Lisäksi sovellus on kehitetty yksinomaan MySQL 5 -tietokantahallintajärjestelmälle, joten tuki tavotteissa asetetetuille kahdelle muulle hallintajärjestelmälle on puutteellinen.

Työn teoriaosassa käsitellään lyhyesti World Wide Webia, verkkokauppaa yleisesti ja web-sovelluksena. Lisäksi tarkastellaan MVC-ohjelmointiarkkitehtuuria ja olio-ohjelmointiparadigmaa sekä tehdään katsaus verkkokauppasovellusalustojen nykytilaan tarkastelemalla lähemmin kolmea markkinoilla olevaa sovellusta: osCommercea, Magentoa, Interspire Shopping Cartia. Suunnittelu-luvussa esitellään tarkemmin projektin tavoitteet ja kuinka ne pyritään saavuttamaan. Luvussa perustellaan myös ohjelmistokehyksen sekä muiden ratkaisujen valinnat. Ongelman ratkaisevan sovelluksen

muutamia piirteitä, kuten tuotteiden ja toimituskulujen hinnoittelulogiikkaa, tietoturvallisuutta, ulkoasun hallintaa ja lokalisointia, tarkastellaan Toteutus-luvussa.

Mittaukset-luvussa suoritetaan ohjelmistolle erillisiä mittauksia, esitetään tulokset ja tehdään niistä johtopäätöksiä. Tutkittavia asioita ovat kolmen markkinoilla olevan verkkokauppasovelluksen ominaisuuksien määrällinen vertailu ja ohjelmakoodikantojen rivimääräanalyysi verrattuna tässä työssä toteutettavaan sovelluksen vastaaviin arvoihin. Lisäksi analysoidaan ohjelmakoodirivien poistamisen syitä versiohallinnasta. Tutkimuksissa havaitaan, että kehitettävän verkkokauppasovelluksen ominaisuuksien määrä ei yllä vielä markkinoiden monipuolisimpien sovellusten tasolle, mutta tämä oletettavasti korjaantuu kehitystyön jatkuessa. Ohjelmakoodirivien poistamisen syiksi paljastuu lähinnä ongelmia hakemistorakenteen ja tiedostojen sijoittelun suunnittelussa. Luvussa esitetään parannusehdotuksia tutkimusmetodiin, jotta versiohallinnasta poistettujen ohjelmakoodirivien tiedosta voitaisiin löytää esimerkiksi viitteitä huonosti ohjelmistosuunnittelusta, jota voitaisiin jatkossa välttää.

Sovelluksen kehitys tulee jatkumaan myös tämän diplomityön jälkeen. Lähitulevaisuudessa on tarkoitus toteuttaa lisää ominaisuuksia, kuten esimerkiksi kattavia raportointi- ja lokalisointityökaluja sekä monia muita pienemmän mittakaavan ominaisuuksia, kuten toimituskululogiikan mukauttamista monimuotoisempiin vaatimuksiin, useiden kategorioiden tukea tuotteille sekä digitaalisia tuotteita. Asiakasmäärien koko ajan kasvaessa, tullaan lisäksi kehittämään uusien verkkokauppa-asennusten luomista sekä vanhojen asennusten päivittämistä helpommiksi.

Jatkotutkittavaa sovelluksessa olisi mahdollisesti testauksen kehittämisen parissa ja siinä, miten sovellus toimii suurissa kaupoissa isoilla tuote- ja varsinkin kävijämäärillä. Sovelluksen tehokkuutta voitaisiin parantaa esimerkiksi optimoimalla ohjelmakoodia sekä lisäämällä palvelinkapasiteettia ja välimuistin käyttöä.

8. LÄHTEET

[1] (Luettu 15.6.2009), Pre-W3C Web and Internet Background. URL: http://www.w3.org/2004/Talks/w3c10-HowItAllStarted/?n=15.

[2] Dixon M.W., McGill T.J. & Karlsson J.M. (1997) Using a network simulation package to teach the client-server model. In: Proceedings of the 1996 Winter Simulation Conference, pp. 1210–1217.

[3] Ala-Kurikka J. (2007) Context-Aware P2P Middleware for Mobile Wellness Applications. Master's thesis, Oulun yliopisto. URL: http://www.mediateam.oulu.fi/publications/pdf/1036.pdf.

[4] Fielding, et al. (1999), Hypertext Transfer Protocol – HTTP/1.1. URL: http://www.ietf.org/rfc/rfc2616.txt.

[5] Berners-Lee T. & Cailliau R. (luettu 15.6.2009), WorldWideWeb: Proposal for a HyperText Project. URL: http://www.w3.org/Proposal.html.

[6] O'Reilly T. (luettu 15.6.2009), What Is Web 2.0. URL: http://oreilly.com/pub/a/web2/archive/what-is-web-20.html.

[7] Kärkkäinen T.T. (2005) Projektinhallintajärjestelmä web-sovelluksena. Master's thesis, Teknillinen korkeakoulu.

[8] Baxley B. (2003), What is a Web Application? URL: http://web.archive.org/web/20030207193154/http://www.boxesandarrows.com/archives/what_is_a_web_application.php.

[9] Kangas V. (2008), Ajax: WWW-sovellusten uudet mahdollisuudet.

[10] (Luettu 24.6.2009), PHP: Passing the Session ID - Manual. URL: http://fi2.php.net/manual/en/session.idpassing.php.

[11] (Luettu 17.9.2009), Sähköisen kaupankäynnin aapinen. URL: http://www.tieke.fi/mp/db/file_library/x/IMG/12422/file/Sahkoisenkaupankaynninaapinen.pdf.

[12] Kuluttajansuojalaki 6. luku 15 §.

[13] Jiang, Chen & Wang (2008), Knowledge and Trust in E-consumers' Online Shopping Behavior.

[14] (Luettu 24.6.2009), Online shopping - Wikipedia, the free encyclopedia. URL: http://en.wikipedia.org/w/index.php?title=Online_shopping&oldid=298051150.

[15] (Luettu 30.10.2009), Omaverkkokaupat - vilkas group oy. URL: http://www.vilkas.fi/Omaverkkokaupat.

[16] Jansch I. (2009) The Rise of Frameworks. phplarchitect , pp. 9–10.

[17] Sklar D. & Trachtenberg A. (2006) PHP Cookbook. O'Reilly Media.

[18] (Luettu 21.7.2009), Comparison of web application frameworks - Wikipedia, the free encyclopedia. URL: http://en.wikipedia.org/w/index.php?title=Comparison_of_web_application_frameworks&oldid=303231843.

[19] McArthur K. (2008) Pro PHP: Patterns, Frameworks, Testing and More. Apress.

[20] Tseng Y.J., Chang C.C. & Tseng (2007) Modeling and Implementation of Object-Oriented E-Commerce Platform. In: The 9th IEEE International Conference, pp. 357 – 366.

[21] (Luettu 21.7.2009), PHP 5 ChangeLog. URL: http://www.php.net/ChangeLog-5.php.

[22] Shire B. (Luettu 21.7.2009), PHP and Facebook. URL: http://blog.facebook.com/blog.php?post=2356432130.

[23] (Luettu 21.7.2009), MediaWiki. URL: http://www.mediawiki.org/w/index.php?title=MediaWiki&oldid=65192.

[24] (Luettu 21.7.2009), WordPress > About. URL: http://wordpress.org/about/.

[25] (Luettu 17.9.2009), osCommerce, Open Source Online Shop E-Commerce Solutions. URL: http://www.oscommerce.com/.

[26] (Luettu 17.9.2009), Magento - Home - eCommerce Software for Growth. URL: http://www.magentocommerce.com/.

[27] (Luettu 17.9.2009), Clover Shop - Verkkokauppaohjelmisto ja sähköisen kaupan ratkaisu. URL: http://www.clovershop.com/.

[28] Penttilä E. (2009), Verkkokauppatutkimus. URL: http://www.apilaratas.fi/verkkokauppatutkimus2009_p1.pdf.

[29] (Luettu 17.9.2009), A quick guide to gplv3. URL: http://www.gnu.org/licenses/quick-guide-gplv3.html.

[30] (Luettu 17.9.2009), Open source initiative osi - the open software license 3.0:licensing. URL: http://www.opensource.org/licenses/osl-3.0.php.

[31] (Luettu 17.9.2009), Google trends: magento, oscommerce, interspire shopping cart. URL: http://www.google.com/trends?q=magento%2C+oscommerce%2C+%22interspire+shopping+cart%22.

[32] (Luettu 29.10.2009), Magento ecommerce - bloated or brilliant? URL: http://www.pickledshark.com/magento-ecommerce-complicated-bloated-brilliant/.

[33] (Luettu 29.10.2009), Shopping cart | e-commerce software | shopping cart software. URL: http://www.interspire.com/shoppingcart/.

[34] Larman C. (2003) Agile and Iterative Development: A Manager's Guide. Addison-Wesley Professional.

[35] (Luettu 30.10.2009), Google trends: mysql, postgresql. URL: http://www.google.com/trends?q=mysql%2C+postgresql%2C.

[36] Ahonen J.M. (Luettu 14.11.2009), Liaani framework. URL: http://liaani.sivuviidakko.fi/.

[37] Whitman M.E. & Mattord H.J. (2004) Principles of Information Security. Course Technolog.

[38] Malyshev S. (2008) Internationalization in PHP 5. php|architect , pp. 18–27.

9. LIITTEET

Liite 1. Arvioitavien verkkokauppasovellusalustojen ominaisuusvertailu

Liite 2. Alustava tietokantaskeema

Liite 3. Sovelluksen tiedostojen esittelyä

Liite 4. Sovelluksen tietokantataulujen esittelyä

Liite 5. `getPrice()`-metodin vuokaavio

Liite 6. `getBasePrice()`-metodin vuokaavio

Liite 7. `getDiscountPrice()`-metodin vuokaavio

Liite 8. `getVariationsTotalPrice()`-metodin vuokaavio

Liite 9. `getVAT()`-metodin vuokaavio

Liite 10. cloc-rivimääräanalyysiohjelman tuottamat raportit

Liite 11. Versiohallinnan historiassa lisättyjen ja poistettujen rivimäärien todelliset lukumäärät

Liite 12. Versiohallinnan historiassa lisättyjen ja poistettujen rivimäärien lukumäärät havainnollisessa muodossa

Liite 1. Arvioitavien verkkokauppasovellusalustojen ominaisuusvertailu

	OSC	**Magento**	**ISC**	**VS**
monikantainen verotus	kyllä	kyllä	kyllä	kyllä
tuotekohtaiset alennukset	kyllä	kyllä	kyllä	kyllä
rekisteröityneiden asiakkaiden osoitteiden tallennus	kyllä	kyllä	kyllä	kyllä
uutiskirje	kyllä	kyllä	kyllä	ei
uutuustuotteet (automaattinen)	kyllä	kyllä	kyllä	kyllä
ostetuimmat tuotteet (automaattinen)	kyllä	kyllä	kyllä	ei
tuotearvostelut	kyllä	kyllä	kyllä	ei
tuotteiden pisteytykset	kyllä	kyllä	kyllä	ei
perushaku	kyllä	kyllä	kyllä	kyllä
kehittynyt haku	kyllä	kyllä	kyllä	ei
varastosaldohallinta	kyllä	kyllä	kyllä	kyllä
pääsynhallinta	kyllä	kyllä	kyllä	kyllä
digitaaliset tuotteet	kyllä	kyllä	kyllä	ei
monivaluuttatuki	kyllä	kyllä	kyllä	ei
asiakas voi seurata tilauksensa tilaa	kyllä	kyllä	kyllä	ei
asiakas näkee tilaushistoriansa	kyllä	kyllä	kyllä	ei
sähköpostitilausvahvistus	kyllä	kyllä	kyllä	kyllä
tuotevariaatiot	kyllä	kyllä	kyllä	kyllä
raportointi parhaiten myyvistä tuotteista	kyllä	kyllä	kyllä	ei
raportointi useimmiten katsotuista tuotteista	kyllä	kyllä	kyllä	ei
raportointi tilausten tuotoista	kyllä	kyllä	kyllä	ei
monisyvyyksinen kategoriointi	kyllä	kyllä	kyllä	kyllä
varastohälytykset	kyllä	kyllä	kyllä	kyllä
kiinteähintaiset toimituskulut	kyllä	kyllä	kyllä	kyllä
tilauksen vähimmäishinta	kyllä	kyllä	ei	ei
monikielisyystuki	kyllä	kyllä	ei	kyllä
kerro kaverille -toiminto	kyllä	kyllä	ei	kyllä
raportti hylätyistä ostoskoreista	kyllä	kyllä	ei	ei
lista tuotteista, mitä muut saman tuotteen ostaneet asiakkaat ovat ostaneet (automaattinen)	kyllä	ei	kyllä	ei
ajaxin hyödyntämistä	ei	kyllä	kyllä	kyllä
ulkoasuhallinta	ei	kyllä	kyllä	kyllä
tuotepalautukset	ei	kyllä	kyllä	kyllä
sivukartta (XML)	ei	kyllä	kyllä	ei
toivelista	ei	kyllä	kyllä	ei
alennuskupongit	ei	kyllä	kyllä	ei
lahjakortit	ei	kyllä	kyllä	ei
käyttäjäryhmät	ei	kyllä	kyllä	kyllä
käyttäjäryhmärajoitetut kategoriat	ei	kyllä	kyllä	ei
käyttäjäryhmärajoitetut tuotteet	ei	kyllä	kyllä	ei
RSS-syöte (Really Simple Syndication)	ei	kyllä	kyllä	ei

ohjelmointirajapinta	ei	kyllä	kyllä	ei
usean kaupan hallinta yhdestä ylläpitopaneelista	ei	kyllä	kyllä	ei
Google Analytics -integraatio	ei	kyllä	kyllä	kyllä
tilauksen teko ylläpidosta	ei	kyllä	kyllä	ei
sisällönhallintajärjestelmä	ei	kyllä	kyllä	kyllä
kauppatason alennukset	ei	kyllä	kyllä	kyllä
kategoriatason alennukset	ei	kyllä	kyllä	kyllä
käyttäjäryhmäkohtaiset alennukset	ei	kyllä	kyllä	kyllä
yksisivuinen tilaustapahtuma	ei	kyllä	kyllä	ei
tuotevertailu	ei	kyllä	kyllä	ei
alennukset perustuen tuotemääriin	ei	kyllä	kyllä	ei
ilmaiset toimituskulut, kun tilauksen arvo ylittää tietyn rajan	ei	kyllä	kyllä	kyllä
räätälöidyt lomakekentät sisällönhallintasivuilla	ei	kyllä	kyllä	ei
tuotelajittelu hinnan mukaan	ei	kyllä	kyllä	kyllä
tilaaminen ilman rekisteröitymistä	ei	kyllä	kyllä	kyllä
useita kuvia tuotteella	ei	kyllä	kyllä	kyllä
toimitusehtojen hyväksyminen ennen tilaamista	ei	kyllä	kyllä	kyllä
raportti varastotilanteesta	ei	kyllä	kyllä	ei
asiakasraportit	kyllä	kyllä	kyllä	ei
toimitustapojen rajoittaminen perustuen vastaanottajan sijaintiin	ei	kyllä	kyllä	ei
yhden tilauksen toimittaminen useaan eri osoitteeseen	ei	kyllä	kyllä	ei
tuotekohtaiset toimituskulut	ei	kyllä	kyllä	kyllä
toimituskulujen hinta perustuen vastaanottajan sijaintiin	ei	kyllä	kyllä	ei
kuittien tulostaminen tilaushallinnasta	ei	kyllä	kyllä	ei
toimitusasiakirjojen tulostaminen tilaushallinnasta	ei	kyllä	kyllä	ei
räätälöitävät tilaussähköpostit	ei	kyllä	kyllä	kyllä
tuotteiden avainsanat (eng. tag)	ei	kyllä	kyllä	kyllä
tuotekohtaiset sivupohjat	ei	kyllä	kyllä	ei
kategoriakohtaiset sivupohjat	ei	kyllä	kyllä	ei
hakukoneoptimoidut URI:t	ei	kyllä	kyllä	kyllä
tilaustapahtuman lomakekenttien räätälöinti	ei	kyllä	kyllä	kyllä
ilmaiset toimituskulut perustuen tuotemääriin	ei	kyllä	ei	kyllä
asiakkaiden manuaalinen hyväksyminen	ei	kyllä	ei	ei
äänestykset	ei	kyllä	ei	ei
hakupilvi	ei	kyllä	ei	ei
sivukartta	ei	kyllä	ei	ei
sosiaaliset merkkaukset (eng. social bookmarking)	ei	kyllä	ei	ei
käyttäjien rekisteröinnin sallimminen perustuen heidän sijaintiin	ei	kyllä	ei	ei

URI:n uudelleenkirjoitushallinta (eng. URI rewrite)	ei	kyllä	ei	ei
uutuuksien manuaalinen kontrollointi	ei	kyllä	ei	kyllä
kuvien vesileimaus	ei	kyllä	ei	ei
kategorioiden kerrostettu navigointi (eng. layered navigation)	ei	kyllä	ei	ei
raportti toivelistoista	ei	kyllä	ei	ei
raportti hakutermien käytöstä	ei	kyllä	kyllä	ei
tulostinystävälliset sivut	ei	ei	kyllä	kyllä
WYSIWYG-editorit	ei	ei	kyllä	kyllä
CAPTCHA-kuvavarmennus rekisteröinnin yhteydessä	ei	ei	kyllä	ei
ylläpidon toimien auditointi	ei	ei	kyllä	ei
asiakkaalle viestittäminen suoraan tilaushallinnasta	ei	ei	kyllä	ei
tilaajien maantieteelliset sijainnit kartalla	ei	ei	kyllä	ei

Liite 2. Alustava tietokantaskeema

```
CREATE TABLE shop (
        id INTEGER NOT NULL AUTO_INCREMENT,
        name VARCHAR(255) NOT NULL,
        PRIMARY KEY (id)
) ENGINE=innodb DEFAULT CHARACTER SET = 'UTF8';

CREATE TABLE category (
        id INTEGER NOT NULL AUTO_INCREMENT,
        shop INTEGER NOT NULL,
        position INTEGER NOT NULL,
        level INTEGER NOT NULL DEFAULT 1,
        image VARCHAR(255) DEFAULT NULL,
        PRIMARY KEY (id),
        FOREIGN KEY (shop) REFERENCES shop (id)
) ENGINE=innodb DEFAULT CHARACTER SET = 'UTF8';

CREATE TABLE category_localization (
        category INTEGER NOT NULL,
        language CHAR(2) NOT NULL,
        name VARCHAR(255) NOT NULL,
        description MEDIUMTEXT DEFAULT NULL,
        image VARCHAR(255) DEFAULT NULL,
        PRIMARY KEY (category, language),
        FOREIGN KEY (category) REFERENCES category (id) ON DELETE CASCADE
) ENGINE=innodb DEFAULT CHARACTER SET = 'UTF8';

CREATE TABLE product (
        id INTEGER NOT NULL AUTO_INCREMENT,
        created TIMESTAMP DEFAULT NOW(),
        modified TIMESTAMP,
        base_price NUMERIC(16, 2) DEFAULT 0.0,
        inventory_price NUMERIC(16, 2) DEFAULT 0.0,
        delivery_price_unit NUMERIC(16, 2) DEFAULT 1.0,
        stock INTEGER DEFAULT 0,
        sold INTEGER DEFAULT 0,
        product_code VARCHAR(40),
        ean_code VARCHAR(255),
        vat NUMERIC(16, 2) DEFAULT 0.0,
        type INTEGER DEFAULT 1,
        availability_begins DATE DEFAULT '0000-00-00',
        availability_ends DATE DEFAULT '0000-00-00',
        miniature VARCHAR(255) DEFAULT NULL,
        image VARCHAR(255) DEFAULT NULL,
        discountable BOOLEAN DEFAULT true,
        PRIMARY KEY (id)
) ENGINE=innodb DEFAULT CHARACTER SET = 'UTF8';

CREATE TABLE product_localization (
        product INTEGER NOT NULL,
        language CHAR(2) NOT NULL,
        miniature VARCHAR(255) DEFAULT NULL,
        image VARCHAR(255) DEFAULT NULL,
        name VARCHAR(255) NOT NULL,
        description MEDIUMTEXT DEFAULT NULL,
        teaser TEXT DEFAULT NULL,
        PRIMARY KEY (product, language),
        FOREIGN KEY (product) REFERENCES product (id) ON DELETE CASCADE
) ENGINE=innodb DEFAULT CHARACTER SET = 'UTF8';
```

Liite 3. Sovelluksen tiedostojen esittelyä

base/sql-change/577.sql: Tiedosto sisältää SQL-lausekkeita (eng. query), jotka päivittävät tietokannan vastaamaan revision 577 tarpeita. Esimerkiksi sovellukseen toteutettava uusi ominaisuus voi vaatia uuden sarakkeen johonkin tietokantatauluun.

base/t/20_category.t.php: Automatisoituja yksikkötestejä liittyen kategorioihin.

base/verkkokauppa.mysql.sql: Sovelluksen tietokantaskeema.

config: Konfiguraatiodata, joka sisältää muun muassa tunnukset tietokannan käyttämiseen.

config-test: Oma konfiguraatiodata yksikkötesteille, koska testejä varten halutaan käyttää eri asetuksia.

Controller/AdminAjax.php: Ohjain sisältää ylläpitopuolen Ajax-metodeja (asynchronous JavaScript and XML).

Controller/Admin.php: Ylläpitopuolen pääohjain, joka sisältää runsaasti toimintoja, kuten tuotehallinnan, kategoriahallinnan, tilaushallinnan, ulkoasun hallinnan, käyttäjähallinnan, käyttäjäryhmien hallinnan, alennusten hallinnan, kaupan asetusten hallinnan.

Controller/Cart.php: Ostoskoriohjain, joka hoitaa tuotteiden lisäämisen ostoskoriin ja tilaustapahtuman (tuotteiden valinta, tilaajatietojen kerääminen, maksaminen, kuitti) läpiviemisen ostajalle.

Controller/Import.php: Työkaluja tuotetietojen tuontiin muista sovelluksista.

Controller/Index.php: Ohjain etusivun tarpeille.

Controller/Page.php: Ohjain, joka näyttää CMS-sisältösivuja.

Controller/Product.php: Tuotelistausten ja tuotesivujen näyttämiseen erikoistunut ohjain.

Controller/Profile.php: Ohjain, joka mahdollistaa käyttäjien rekisteröinnin ja henkilötietojen päivittämisen.

Controller/Search.php: Julkisen puolen hakutoiminnallisuudet.

Controller/Site.php: Juuriohjain, jonka kaikki muut julkisen puolen ohjaimet perivät. Ohjain sisältää toimintoja, joiden on hyvä olla saatavilla kaikkialla, kuten kaupan asetusten hakeminen, ostoskorin tarkistaminen ja hakutoiminnallisuudet.

Data/debug.log: Jos sovellus on debug-tilassa, tähän tiedostoon kerätään lisäinformaatiota sovelluksen toiminnasta.

Data/text/fi.xml: Suomenkielinen käännöstiedosto ylläpidon näkymiä varten.

Data/text/en.xml: Englanninkielinen käännöstiedosto ylläpidon näkymiä varten.

htdocs/.htaccess: Sovelluksen juuren hakemistokohtainen asetustiedosto, joka ohjaa kaikki palvelupyynnöt samassa juuressa olevalle `index.php`-tiedostolle lukuunottamatta `htdocs/static`-hakemistoon kohdistuvia pyyntöjä.

htdocs/index.php: Esilatausohjelma, joka alustaa sovelluksen ja sen jälkeen valitsee palvelupyynnön mukaisen ohjaimen ja metodin.

htdocs/robots.txt: Hakukoneita varten tehty lista hakemistoista, joita ei saa listata hakupalveluiden tietokantoihin.

htdocs/static/css/: Hakemisto sisältää sovelluksen tyyliohjemäärittelyjä, joita käytetään ylläpitopuolella.

htdocs/static/logos/viidakkostore.jpg: Esimerkki sovelluksen kuvatiedostosta, joita `htdocs/static`-hakemisto sisältää runsaasti. Kuvat ovat käytössä ylläpitopuolella.

htdocs/static/js/window.js: Esimerkki JavaScript-tiedostosta, joka on tarkoitettu ylläpitopuolen käyttöön.

htdocs/static/media/: Hakemisto sisältää käyttäjän syöttämää dataa, kuten tuotekuvia, sivuston ulkoasun kuvia, julkisen puolen tyyliohjemäärittelyjä ja JavaScript-komentosarjoja.

lib/: Hakemisto sisältää ohjelmistokehyksen ohjelmakoodit.

Model/Account.php: Luokka, joka sisältää käyttäjäentiteetin ominaisuuksia ja toimintamalleja. Ominaisuuksia ovat esimerkiksi sähköpostiosoite, nimi, rooli, käyttäjälle sallitut toimitustavat ja salasana. Toimintamalleja ovat muun muassa käytännössä kaikille entiteeteille välttämättömät tietojen tallennus- ja poistometodit sekä esimerkiksi käyttäjäoikeuksien tarkistusmetodi.

Model/AdminHelper.php: Malli sisältää työkaluja ylläpitopuolen näkymien luomiseen, kuten sivutuksen ja kategorialistan generointityökalut.

Model/Cart.php: `Cart`-luokka sisältää metodit esimerkiksi ostoskorin tuotteiden lisäämiselle, poistolle, saatavuuksien tarkistamiselle ja hintojen laskemiselle. Ominaisuuksia ovat esimerkiksi luontiaika, maksajan nimi, ostoskorin kokonaishinta ja maksutapa.

Model/Category.php: Luokka sisältää kategorian ominaisuudet ja toimintamallit. Ominaisuuksia ovat esimerkiksi kategorian lokalisoitu nimi, lokalisoitu kuva ja muokkausaika.

Model/CategoryPublic.php: Luokka perii `Category`-luokan ja lisää rajoitteeksi näkyvyyskriteerit, jotka varmistavat, ettei julkisella puolella näytetä kategorioita, jotka eivät täytä näkyvyyskriteerejä. Ylläpitopuolella näistä kriteereistä ei tarvitse välittää. Toisin sanoen, jos luodaan olio `Category`-luokasta, niin ei tarkisteta, onko kyseinen kategoria esimerkiksi piilotettu, mutta jos saman kategorian olio luodaan `CategoryPublic`-luokasta, niin tarkistetaan, ettei kategoria ole piilotettu.

Model/Context.php: Luokka sisältää staattisia asetus- ja saantimetodeja (eng. setter ja getter), joiden avulla siirretään tietoa ohjaimilta ulkoasumoottorille.

Model/Delivery.php: Luokka toimitustapoja varten sisältää muun muassa metodin toimituskulujen laskennalle.

Model/FileManagement.php: Tiedostonhallinnan luokka.

Model/Hook.php: `Hook`-luokan avulla voidaan sovellukseen sisällyttää asennuskohtaisia ohjelmakoodeja.

Model/ImageManipulation.php: Kuvien muokkaamiseen tarkoitettuja työkaluja, kuten kuvien koon muuttaminen.

Model/Image.php: Kuvaentiteetin ominaisuudet ja toimintamallit. Ominaisuuksia ovat esimerkiksi tunniste, kieli ja kuvaus.

Model/Includes.php: Sivupohjien lisäkkeiden ominaisuudet ja metodit. Ominaisuuksia ovat esimerkiksi tyyppi, lähdekoodi sekä nimi ja vastaavasti metodeja ovat lisäkkeen ja sen muuttujien parsiminen PHP-ohjelmakoodiksi.

Model/Mail.php: Sähköpostien käsittelyyn tarkoitettuja toimintamalleja.

Model/Message.php: Tämän luokan staattisten metodien kautta kuljetetaan ohjaimilta virheilmoituksia näkymille.

Model/Order.php: Tilausentiteetin ominaisuudet ja toimintamallit, kuten esimerkiksi tilausvahvistuksen lähettäminen asiakkaalle ja tilauksen peruuttaminen. Ominaisuuksia ovat muun muassa tilausnumero, toimituskulujen hinta ja tilaajan mahdollinen Y-tunnus.

Model/Page.php: CMS-sisältösivujen ominaisuukset ja toimintamallit.

Model/Product.php: Tuote-entiteetin ominaisuudet ja toimintamallit. Ominaisuuksia ovat esimerkiksi lokalisoitu nimi, saatavuusajan alkamispäivä ja varastosaldo.

Model/ProductPublic.php: Kuten `CategoryPublic`-luokka, niin tämäkin perii `Product`-luokan ja lisää julkisen puolen näkyvyyskriteerit, jotta asiakkaille ei näytettäisi tuotteita, jotka on piilotettu tai joiden varastosaldo on nolla, jos niin on määrätty.

Model/Role.php: Käyttäjäryhmien toimintamallit ja ominaisuudet.

Model/Setting.php: Kaupan asetusten toimintamallit ja ominaisuudet. Sisältää muun muassa tiedon kaupan nimestä, osoitteesta, arvonlisäveroista, tuotekuvien maksimimitoista ja käytetyistä kielistä.

Model/Template.php: `Template`-luokka sisältää ulkoasun sivupohjien ominaisuudet ja toimintamallit, kuten sivupohjien kääntämisen ja tallentamisen.

Model/Util.php: Luokka sisältää sekalaisia työkaluja sovelluksen tarpeille.

Plugin/epayment/: Ulkopuolinen liitännäisosa, joka sisältää maksumodulin, jonka avulla verkkomaksuja voidaan hoitaa.

Renderer/VK.php: Sovelluskohtainen renderöijä, joka käyttää käyttäjän tekemiä sivupohjia julkisen puolen näkymien muodostamiseen.

Template/admin/category/edit.tpl.php: Ylläpitopuolen kategoriahallinnan muokkaussivun näkymien XHTML-rakenne.

Template/adminajax/category_controls.tpl.php: Ylläpitopuolen kategorioiden toiminnot-ikkunan XHTML-rakenne.

Template/User/category/index.php: Sivuston rakentajan (eng. site builder) kirjoittamasta sivupohjasta generoitu PHP-ohjelmakoodi julkisen puolen tuotelistausten näyttämiseen.

Template/User/category/index.src: Sivuston rakentajan kirjoittama sivupohja julkisen puolen tuotelistausten näyttämiseen.

Liite 4. Sovelluksen tietokantataulujen esittelyä

shop: Yksi sovelluksen tavoitteista on, että yhdestä ylläpidosta voidaan hallinnoida useampaa kauppaa. Tämä taulu sisältää eri kauppojen tunnisteet.

shop_setting: Taulu sisältää kaupan asetuksia kuten nimen, verotasot, peukalonkynsikuvien maksimikoot, oletuskateprosentin ja tuotteiden oletusasetukset. Rakenne on suunniteltu niin, että uusien asetusten lisääminen on vaivatonta.

category: Taulu sisältää tuotekategorioiden tiedot, kuten mille kaupalle kategoria kuuluu, sen oletuskuvan ja paikan kategoriahierarkiassa.

category_localization: Kategorioiden lokalisaatiodata eli esimerkiksi nimi, kuvaus ja kuva.

product: Taulu sisältää tuotteiden tiedot eli esimerkiksi inventaariohinnan, toimitusajan, varastosaldon ja saatavuusajan.

product_localization: Tuotteiden lokalisaatiodata, kuten nimi, lyhenne, kuvaus ja kuva.

category_products: Taulu sisältää kategorioiden ja tuotteiden assosiaatiotiedon, eli mihin kategoriaan mikäkin tuote kuuluu.

product_compatible: Taulu sisältää tietoa yhteensopivista tuotteista, minkä avulla voidaan esimerkiksi näyttää tulostinta ostavalle kävijälle linkki sopiviin mustekasetteihin.

product_descriptions: Tuotteille voidaan määrittää ylläpidossa rajattomasti kuvauksia. Tämä taulu sisältää kaikki lisäkuvaukset.

image: Oletus- ja oletushoukutinkuvan lisäksi tuotteilla voi olla rajattomasti lisäkuvia. Tämä taulu sisältää lisäkuvien tiedot, kuten tunnisteen ja kuvan nimen.

image_localization: Tuotteiden lisäkuvien lokalisaatiodata eli esimerkiksi kuvan otsikko ja kuvaus.

variation: Taulu sisältää tuotevariaatioiden tiedot, kuten tuotevariaation tuoma lisähinta, varastosaldo ja saatavuus.

variation_localization: Tuotevariaatioiden lokalisaatiodata, kuten nimi ja kuvaus.

cart: Tauluun tallennetaan kävijöiden luomat ostoskorit. Järjestelmässä on erotettu luodut ostoskorit ja tehdyt tilaukset, vaikka ne sisältävätkin paljon samaa dataa. Erottelu on tehty selkeyttämään tiedon tallennusta. Kaikki ostoskorit tallennetaan tietokantaan, jotta ylläpidossa voidaan näyttää, mitä tuotteita on harkittu ostettavan.

cart_products: Taulu sisältää tiedon, mitä tuotteita ja kuinka paljon on missäkin ostoskorissa.

order_status: Tilauksen tila voi olla esimerkiksi tilattu, maksettu tai käsittelyssä. Tämä taulu sisältää tiedot eri tiloista, joita myyjä voi myös laajentaa omilla tiloillaan.

vk_order: Tauluun tallentuu kaikki tehdyt tilaukset. Taulussa on paljon samaa dataa kuin cart-taulussakin. Tauluun tallennetaan myös, mikä on ollut tilauksen loppusumma, verojen osuus ja toimituskulujen suuruus. Koska SQL:ssä sana *order* on varattu, taulun nimi on hieman erikoinen.

order_products: Tilaukseen kuuluvat tuotteet listataan `order_products`-taulussa. Taulu sisältää - erona `cart_products`-tauluun - myös tuotteiden lopulliset hinnat ja verot. Tämä siitä syystä, että kun tilaus on tehty, niin hinnat eivät saa enää muuttua asiakkaalle.

cart_product_variations: Taulu sisältää ostoskorin tuotteiden tuotevariaatiot ja lisäksi esimerkiksi niiden nimet ja lisähinnat.

order_product_variations: Taulu sisältää tilauksen tuotteiden tuotevariaatiot tietoineen, kuten nimet ja lisähinnat. Taulu on periaatteessa identtinen `cart_product_variations`-taulun kanssa.

cart_payment: Ostoskorin tilaajan eli maksajan tiedot tallennetaan tähän tauluun. Tietoina ovat esimerkiksi nimi, katu- ja sähköpostiosoite sekä maksutapa.

order_payment: Tilauksen tehneen tilaajan tiedot tallennetaan tähän tauluun. Tietoina ovat esimerkiksi nimi, sähköpostiosoite ja verkkomaksupalvelun tuottama vaste. Taulu on lähes identtinen `cart_payment`-taulun kanssa.

cart_delivery: Taulu sisältää ostoskorin vastaanottajatiedot eli muun muassa vastaanottajan nimen, puhelinnumeron ja toimitustavan.

order_delivery: Taulu sisältää tilauksen vastaanottajatiedot eli muun muassa vastaanottajan nimen, postinumeron ja toimitustavan. Taulu on lähes identtinen `order_delivery`-taulun kanssa.

delivery: Kaupalla käytössä olevien toimitustapojen tiedot tallennetaan tähän tauluun.

delivery_localization: Kaupalla käytössä olevien toimitustapojen lokalisaatiodata tallennetaan tähän tauluun.

delivery_cost: Toimitustapojen kuluporrastukset tallennetaan kyseessä olevaan tauluun.

payment: Taulu sisältää käytössä olevien maksutapojen tiedot.

payment_localization: Taulu sisältää kaupalla käytössä olevien maksutapojen lokalisoidut tiedot.

includes: Ulkoasuhallinnan sivupohjien lisäkkeiden lähdekoodit ja muut tiedot tallennetaan tähän tauluun.

page: Tämä taulu sisältää ulkoasuhallinnan sisältösivujen tiedot.

account: Kaupan käyttäjätiedot tallennetaan tähän tauluun. Tietoja ovat esimerkiksi salasana, käyttäjän nimi, yhteystiedot ja yhteyshenkilö yritysasiakkaille.

shop_role: Kaupassa on käytössä erilaisia käyttäjäryhmiä ja näiden ryhmien tiedot tallennetaan tähän tauluun. Ryhmien tiedoissa on muun muassa sallitut toimitustavat, ryhmän nimi ja koko kaupan alennusprosentti.

shop_role_permission: Taulu sisältää käyttäjäryhmien oikeudet kaupan eri osiin. Ylläpidon eri alueet voidaan rajata eri käyttäjäryhmille niin, että esimerkiksi vain *ylläpitäjät*-ryhmään kuuluvat pääsevät muokkaamaan kaupan ulkoasua ja *myyjät*-ryhmään kuuluvat pääsevät käyttämään vain tilaus- ja tuotehallintaa.

shop_account: Taulu sisältää assosiaatiotietoa: mihin käyttäjäryhmään kukakin käyttäjä kuuluu.

shop_role_category_discount: Käyttäjäryhmille voidaan antaa kategoriakohtaisia alennusprosentteja, jotka tallennetaan tähän tauluun.

shop_role_product_discount: Käyttäjäryhmille voidaan antaa kiinteitä ja prosenttialennuksia, jotka tallennetaan tähän tauluun.

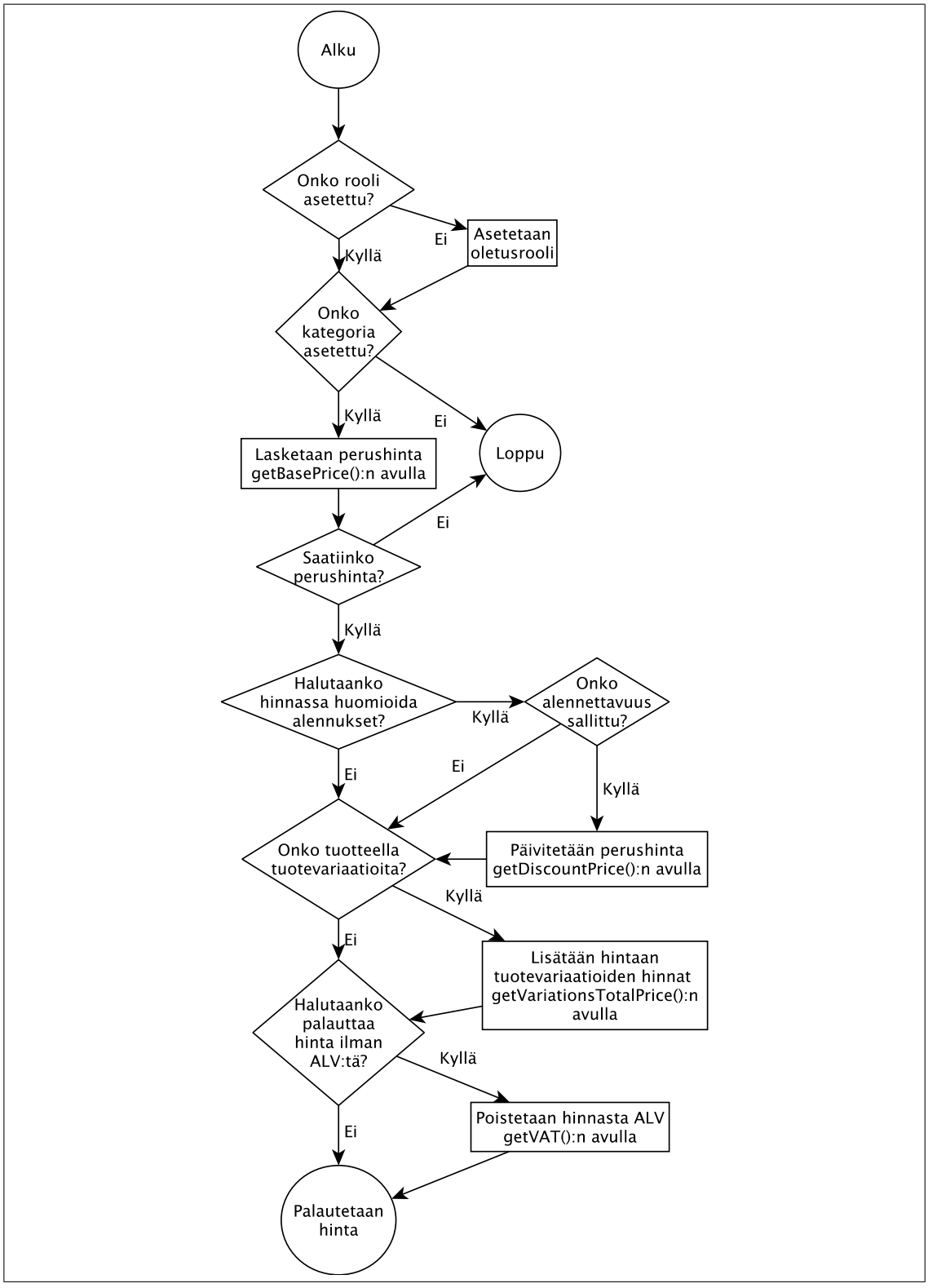
login: Taulu sisältää tietoa käyttäjien sisäänkirjautumisista. Tallennettavia tietoja ovat esimerkiksi käyttäjän tunniste, ajankohta ja IP-osoite.

tag: Tuotteisiin voidaan kiinnittää erityisiä avainsanoja (eng. tag), jotka voivat olla esimerkiksi valmistajia. Tämä taulu sisältää tietoa näistä avainsanoista.

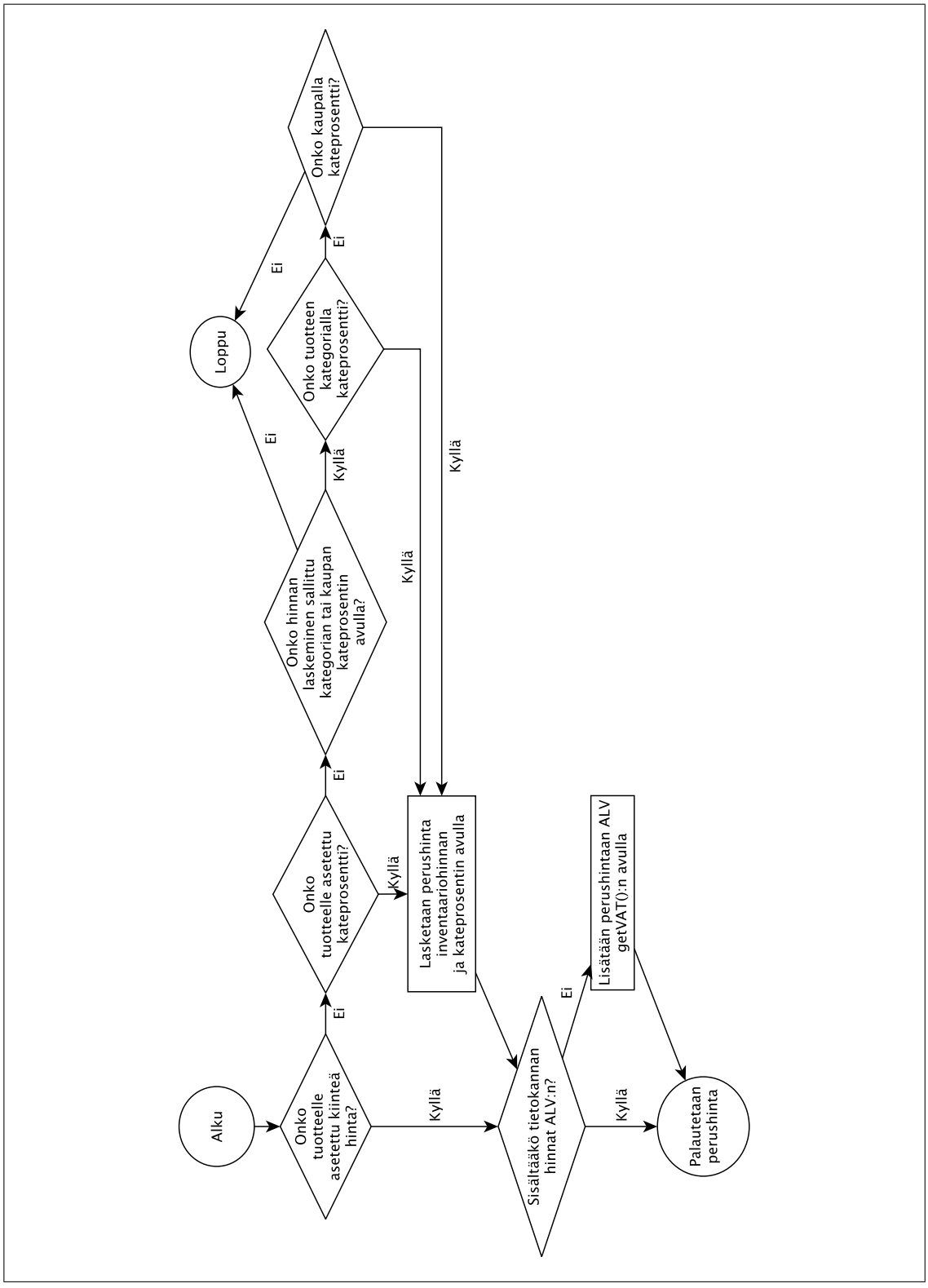
tag_localization: Avainsanojen lokalisaatiodata tallennetaan tähän tauluun.

product_tags: Taulu sisältää assosiaatiotiedon siitä, mitkä avainsanat liittyvät mihinkin tuotteeseen.

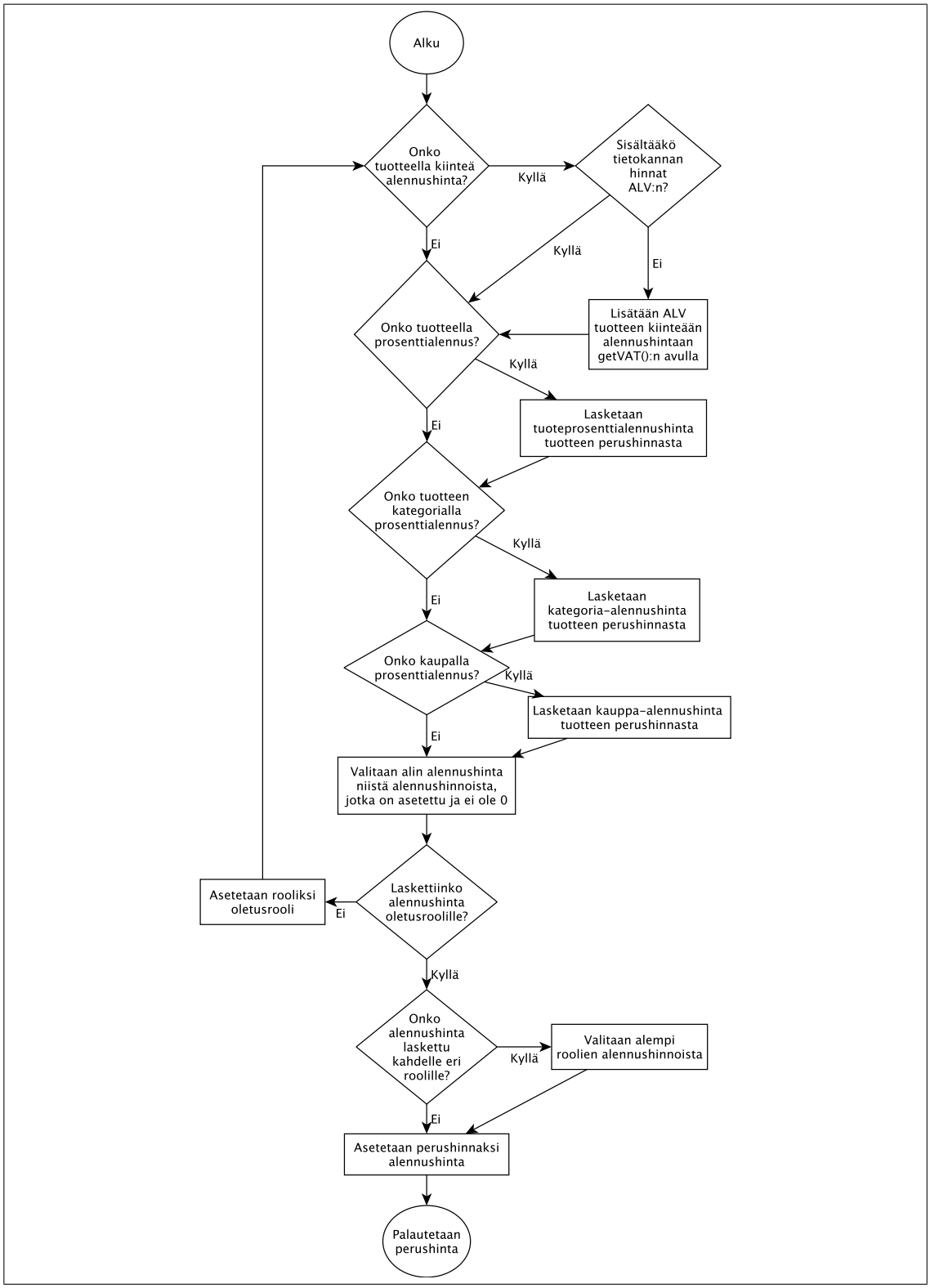
Liite 5. getPrice()-metodin vuokaavio



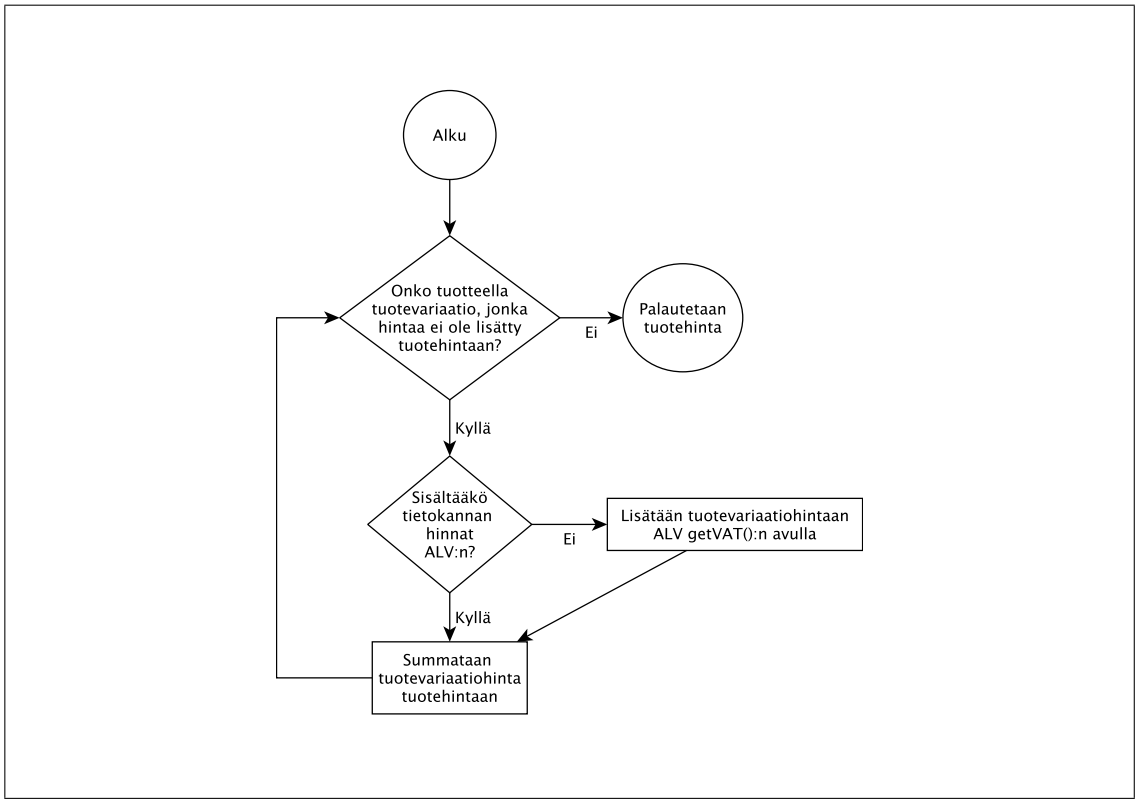
Liite 6. getBasePrice()-metodin vuokaavio



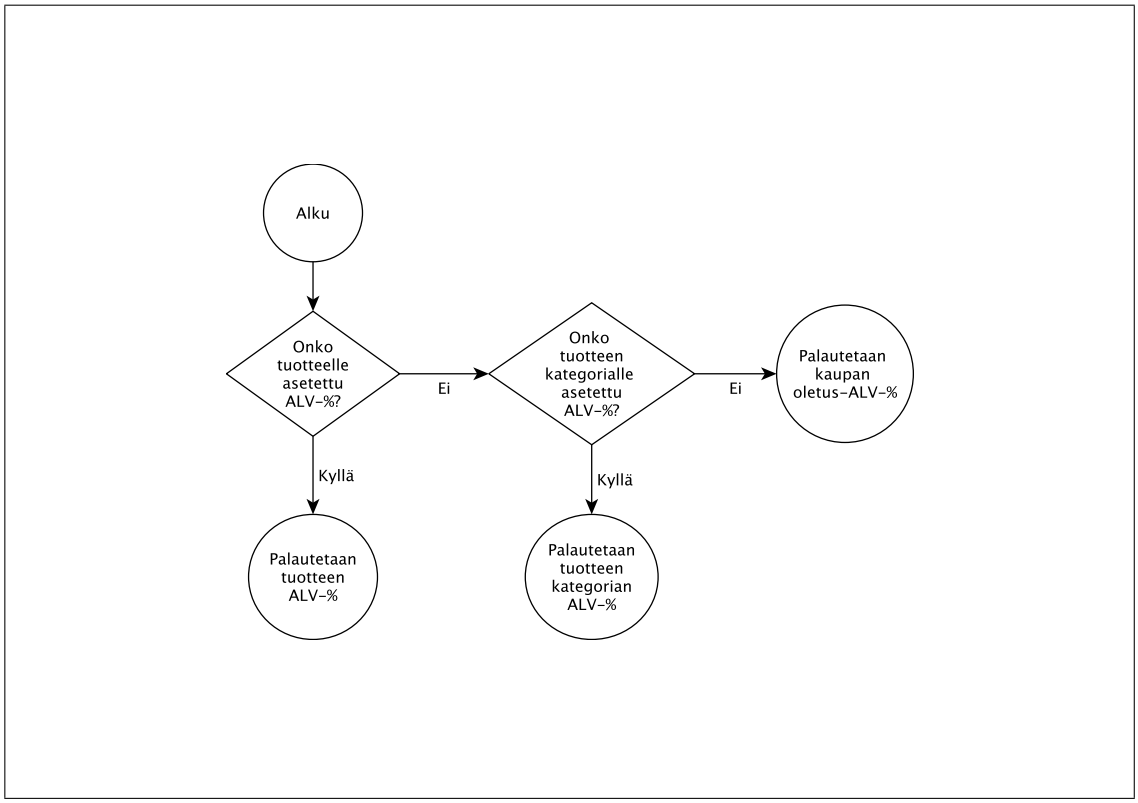
Liite 7. getDiscountPrice()-metodin vuokaavio



Liite 8. getVariationsTotalPrice()-metodin vuokaavio



Liite 9. getVAT()-metodin vuokaavio



Liite 10. cloc-rivimääräanalyysiohjelman tuottamat raportit

```
$ perl cloc oscommerce-2.2rc2a --progress-rate=1 --no3
     604 text files.
     585 unique files.
      17 files ignored.

http://cloc.sourceforge.net v 1.08  T=3.0 s (190.3 files/s, 27397.3 lines/s)
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
PHP                            556           9085           6015          42283
HTML                             2            535              0          19462
Javascript                       6            337            178           1857
SQL                              1             99              0           1492
CSS                              6            130             40            679
-------------------------------------------------------------------------------
SUM:                           571          10186           6233          65773
-------------------------------------------------------------------------------


$ perl cloc magento-1.3.2.4 --progress-rate=1 --no3
    6412 text files.
    6368 unique files.
     734 files ignored.

http://cloc.sourceforge.net v 1.08  T=43.0 s (131.6 files/s, 24495.2 lines/s)
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
PHP                           4792          72402         308971         370266
XML                            650           1049           4749         233368
CSS                            107           2882           1937          24477
Javascript                      63           3564           2932          24142
HTML                            36            123             79           1252
ActionScript                     3             92            225            482
Bourne Shell                     5             29             17            162
XSLT                             1             15              2             53
DTD                              1              3              0             16
Bourne Again Shell               1              1              0              2
-------------------------------------------------------------------------------
SUM:                          5659          80160         318912         654220
-------------------------------------------------------------------------------


$ perl cloc ISC-5.0.3 --progress-rate=1 --no3
    2320 text files.
    1855 unique files.
     511 files ignored.

http://cloc.sourceforge.net v 1.08  T=11.0 s (122.7 files/s, 26139.3 lines/s)
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
PHP                            533          28383          36644         145044
Javascript                     199           9146           3425          34540
HTML                           542           3040            188          17708
CSS                             69           1520            275           6250
XML                              7            116             76           1177
-------------------------------------------------------------------------------
SUM:                          1350          42205          40608         204719
-------------------------------------------------------------------------------


$ svn checkout https://svn.viidakko.fi/verkkokauppa/trunk vk
$ perl cloc.pl vk/base/ vk/Controller/ vk/Filter/ vk/Header/ vk/Data/text/
vk/lib/ vk/Model/ vk/Template/ vk/htdocs/static/css/ vk/htdocs/static/js/
vk/htdocs/static/flags/ vk/htdocs/static/icons/ vk/htdocs/static/logos/
vk/htdocs/static/skin/ vk/Renderer/ vk/Plugin/ --exclude-dir=vk/Template/User/
--progress-rate=1
     499 text files.
     490 unique files.
    3096 files ignored.
```

```
http://cloc.sourceforge.net v 1.08  T=2.0 s (242.0 files/s, 39291.5 lines/s)
-------------------------------------------------------------------------------
Language           files     blank   comment      code    scale   3rd gen. equiv
-------------------------------------------------------------------------------
PHP                  231      4077      4648     26748 x   3.50 =       93618.00
Javascript           136      5880      2820     21965 x   1.48 =       32508.20
HTML                  35       522        29      4939 x   1.90 =        9384.10
CSS                   37       641       195      3582 x   1.00 =        3582.00
SQL                   41       127       176      1119 x   2.29 =        2562.51
XML                    2        46         0       929 x   1.90 =        1765.10
Bourne Shell           1        12         5       108 x   3.81 =         411.48
make                   1         5         1         9 x   2.50 =          22.50
-------------------------------------------------------------------------------
SUM:                 484     11310      7874     59399 x   2.42 =      143853.89
-------------------------------------------------------------------------------
```

Liite 11. Versiohallinnan historiassa lisättyjen ja poistettujen rivimäärien todelliset lukumäärät

Rev	Ins	Del
3	0	0
4	161	0
7	627	363
8	0	0
9	10	2
10	4	0
11	6	0
12	33	3
13	0	3
14	12	4
15	26	5
16	6	17
17	496	0
20	19	4
21	17	13
22	27	10
23	0	0
24	47	20
25	2	0
26	46	31
27	23	15
28	54	30
29	3	0
30	289	131
31	41	0
32	7	12
33	4	0
34	46	12
35	0	0
36	4	4
37	0	5
38	0	0
39	0	4
40	44	8
41	3	14
42	0	0
43	5	4
44	828	14
45	35	0
46	108	4
47	2043	0
48	254	148
49	33	7
50	46	2
51	0	63
52	0	65
53	112	26
54	110	65
55	40	19
56	629	66
57	25	8
58	100	43
59	253	74
60	29	23
61	145	11
62	105	7
63	210	182
64	598	0
65	174	37
66	27	20
67	207	252
68	0	3
69	258	90
70	163	28
71	76	10
72	84	23
73	202	22
74	389	43
75	26	0
76	278	53
77	202	20
78	287	55
79	267	109
80	670	334
81	0	143
82	127	290
83	7	13
84	45	16
85	17	8
86	204	41
87	170	108
88	15	0
89	395	234
90	571	376
91	3	20
92	439	466
93	1627	0
94	0	33
95	0	64
96	0	9
97	11	4
98	48	44
99	14	0
100	518	288
101	397	172
102	38	25
103	37	0
104	2	2
105	3	2
106	38	17
107	48	36
108	15	5
109	13	5
110	14	10
111	532	158
112	47	29
113	150	33
114	908	16
115	20	23
116	68	43
117	106	106
118	8	7
119	425	0
120	36	39
121	32	46
122	148	135
123	374	288
124	9	17
125	95	224
126	342	0
127	405	319
128	105	54
129	411	150
130	133	27
131	22	137
132	231	74
133	0	127
134	0	0
135	222	0
136	425	0
137	0	131
138	0	0
139	0	0
140	0	0
141	0	0
142	5	0
143	113	27
144	91	0
145	131	0
146	0	91
147	63	20
148	40	16
149	57	24
150	2	8
151	0	0
152	37	14
153	253	231
154	53	92
155	5	4
156	2	0
157	55	50
158	20	24
159	763	278
160	183	88
161	7	2
162	37	0
163	19	12
164	91	29
165	20	38
166	35	11
167	32	16
168	88	26
169	26	0
170	0	0
171	26	26
172	107	71
173	24	0
174	22	17
175	76	65
176	26	30
177	366	272
178	140	140
179	39	15
180	19	19
181	38	28
182	28	15
183	0	0
184	0	2
185	4	4
186	14	14
187	17	13
188	59	8
189	114	10
190	26	26
191	6	3
192	3	3
193	59	96
194	49	14
195	14	5
196	299	8
197	6	13
198	262	197
199	28	30
200	51	60
201	0	0
202	5	4
203	41	36
204	19	18
205	80	40
206	0	0
207	29	31
208	156	17
209	7	3
210	14	10
211	3	2
212	10	10
213	7	4
214	31	21
215	125	12
216	0	0
217	11	6
218	2403	437
219	294	294
220	0	0
221	0	0
222	161	161
223	161	161
224	1356	898
225	646	646
226	113	608
227	117	92
228	9	0
229	46	9
230	270	238
231	293	227
232	120	53
233	222	62
234	139	10
235	140	80
236	188	61
237	62	28
238	45	56
239	255	166
240	22	5
241	34	14
242	266	187
243	270	67
244	188	84
245	36	4
246	130	68
247	118	6
248	0	0
249	24	24
250	89	137
251	32	0
252	57	23
253	347	64
254	50	44
255	153	42
256	178	70
257	2	0
258	11	6
259	3	0
260	121	44
261	570	463
262	0	0
263	396	110
264	24	21
265	22	19
266	231	54
267	51	22
268	207	204
269	925	386
270	87	31
271	1060	175
272	150	146
273	72	61
274	8	2
275	83	29
276	42	35
277	406	409
278	154	73
279	17	9
280	198	23
281	1697	1707
282	683	107
283	97	55
284	71	27
285	361	444
286	8	8
287	23	23
288	31	30
289	0	0
290	83	19
291	0	0
292	17	9
293	3	3
294	1412	1131
295	18	20
296	8	15
297	71	42
298	76	42
299	596	180
300	5	2
301	0	77
302	77	0
303	0	0
304	160	176
305	423	423
306	61	46
307	21	15
308	45	5082
309	2	0
310	12	8
311	27	0
312	5	0
313	2	2
314	43	28
315	4	0
316	46	38
317	4	5
318	9	9
319	8	8
320	5	5
321	6	8
322	0	1475
323	0	193
324	0	15
325	0	95
326	33	4
327	28	0
328	13	14
329	30	12
330	21	10
331	21	6
332	0	0
333	4	0
334	4	6
335	7	8
336	5	2
337	7	7
338	9	9
339	6	7
340	0	0
341	17	3
342	0	2
343	7	5
344	3	2
345	4	8
346	7	5
347	6	0
348	0	0
349	5	4
350	2	2
351	14	9
352	8	7
353	5	6
354	41	35
355	29	12
356	5	0
357	13	9
358	113	9
359	3	2
360	10	8
361	133	55
362	4	3
363	0	0
364	13	7
365	7	14
366	81	46
367	22	13
368	20	10
369	0	0
370	19	0
371	0	0
372	19	0
373	3	22
374	53	51
375	5	3
376	58	91
377	26	46
378	3	2
379	3	3
380	5	2
381	47	35
382	8	0
383	2	3
384	36	34
385	0	0
386	21	5
387	10	0
388	5	4
389	17	6
390	29	66
391	0	20
392	25	15
393	0	0
394	32	12
395	67	17
396	20	3
397	0	470
398	36	11
399	49	9
400	546	183
401	115	18
402	42	30
403	15	18
404	3	3
405	56	23
406	12	2
407	2	2
408	2	2
409	143	66
410	8	7
411	2	0
412	13	5
413	0	0
414	22	27
415	4	5
416	5	3
417	49	19
418	52	0
419	4	4
420	4	6
421	0	3
422	22	0
423	3	3
424	2	2
425	12	4
426	29	9
427	110	71
428	10	9
429	3	3
430	372	74
431	0	0
432	19	10
433	43	22
434	12	4
435	129	42
436	13	3
437	0	2
438	9	4
439	7	4
440	45	7
441	5	4
442	73	16
443	7	6
444	4	2
445	7	6
446	21	25
447	11	5
448	2	2
449	118	8
450	178	175
451	80	55
452	65	104
453	54	29
454	58	32
455	6	0
456	0	0
457	7	0
458	229	110
459	64	16
460	75	75
461	60	58
462	35052	0
463	89	131
464	0	0
465	92	57
466	8	8
467	50	0
468	11	14
469	16	10
470	30	10
471	12	3
472	6	4
473	0	2
474	0	0
475	43	13
476	2	0
477	52	184
478	0	0
479	153	19
480	0	0
481	0	0
482	33	22
483	3	2
484	6	7
485	89	42
486	77	160
487	72	20
488	0	9
489	30	5
490	45	38
491	497	249
492	74	18
493	113	41
494	0	0
495	2	0
496	31	12
497	64	63
498	0	0
499	0	19
502	463	100
503	18	0
504	21	33
505	2	0
506	6	2
507	0	0
508	25	13
509	0	0
510	90	16
511	7	3
512	6	3
513	77	63
514	0	0
515	49	24
516	4	3
517	14	14
518	29	16
519	0	0
520	45	26
521	45	58
521	45	58
522	61	4
523	461	265
524	3	0
525	34	6
526	6	0
527	5	0
528	13	9
529	2	2
530	7	6
531	9	4
532	44	119
533	17	5
536	0	35034
539	35034	0

Rev	Ins	Del
540	0	35034
541	0	0
545	0	0
546	35034	0
547	165	41
548	0	35034
551	35034	0
554	370	14
555	13	3
556	340	146
557	57	44
558	33	34
559	47	10
560	744	22
561	16	2
562	169	24
563	8	3
564	41	39
565	33	6
566	2	2
567	8	25
568	17	17
569	14	80
570	12	3
571	123	2
572	55	231
573	22	5
574	13	21
575	10	10
576	5	0
577	326	130
578	13	13
579	2	0
580	5	19
581	75	68
582	30	16
583	6	0
584	0	2
585	17	16
586	0	0
587	4	3
588	103	42
589	140	184
590	39	33
591	94	5
592	11	5
593	4	0
594	42	32
595	6	4
596	260	15
597	86	6
598	39	13
599	4	15
600	14	0
601	4	0
602	9	0
603	15	0
604	197	22
605	272	0
606	0	0
607	198	32
608	15	7
609	172	112
610	10	8
611	175	75
612	333	206
613	13	10
614	2	0
615	64	14
616	7	3
617	13	8
618	57	2
619	2	2
620	5	2
621	4	3
622	7	0
623	5	0
624	2	2
625	61	32
626	2	2
627	2	2
628	2	0
629	0	0
630	4	0
631	0	0
632	2	2
633	624	46
634	0	0

Rev	Ins	Del
635	7	0
636	408	20
637	12	14
638	30	12
639	11	0
640	217	29
641	0	10
642	2	2
643	5	0
644	6	6
645	4	4
646	4	0
647	57	21
648	0	0
649	456	60
650	2	0
651	0	2
652	0	3
653	350	79
654	21	9
655	2	0
656	19	12
657	7697	1432
658	1692	0
659	406	0
660	5	5
661	66	21
662	406	397
663	129	43
664	303	316
665	26	3
666	5	5
667	98	42
668	44	0
669	88	16
670	6	7
671	19	7
672	0	0
673	2	4
674	26	22
675	27	10
676	100	0
677	47	47
678	172	43
679	11	0
680	8	5
681	0	2
682	182	21
683	0	0
684	59	57
685	2	2
686	0	0
687	53	17
688	4	0
689	144	61
690	6	4
691	11	0
692	285	73
693	61	12
694	233	132
695	77	9
696	64	25
697	5	2
698	2	0
699	0	0
700	5	2
701	4	3
702	53	0
703	0	0
704	0	0
705	117	102
706	150	2
707	8	4
708	21	5
709	0	2
710	25	10
711	33	21
712	2	2
713	134	12
714	2	18
715	54	17
716	141	2579
717	3	0
718	9	8
719	21	4
720	24	0
721	0	0
722	0	0

Rev	Ins	Del
723	41	16
724	3	0
725	2	2
726	7	3
727	35	12
728	14	15
729	35	15
730	37	18
731	45	3
732	7	3
733	2	2
734	88	68
735	29	26
736	26	22
737	0	434
738	3	3
739	298	61
740	5	2
741	5	2
742	12	5
743	7	3
744	216	50
745	11	0
746	11	7
747	2	2
748	9	0
749	55	26
750	11	8
751	9	8
752	0	0
753	64	18
754	21	2
755	2	2
756	45	37
757	0	0
758	103	76
759	19	12
760	37	21
761	3	3
762	6	3
763	0	0
764	4	2
765	8	8
766	5	0
767	3	0
768	4	0
769	9	6
770	7	6
771	11	3
772	11	9
773	16	10
774	2	0
775	0	0
776	2	2
777	3	0
778	49	2

Liite 12. Versiohallinnan historiassa lisättyjen ja poistettujen rivimäärien lukumäärät havainnollisessa muodossa