Magento-Moodle-integraatio

Koulutuksen myynti verkossa

HAMK
HÄMEEN AMMATTIKORKEAKOULU

Ammattikorkeakoulun opinnäytetyö

Tietojenkäsittelyn koulutusohjelma

Visamäki, syksy 2014

Kimmo Kava

HAMK
HÄMEEN AMMATTIKORKEAKOULU

TIIVISTELMÄ

Visamäki
Tietojenkäsittelyn koulutusohjelma
Systeemityö

Tekijä Kimmo Kava **Vuosi** 2014

Työn nimi Magento-Moodle -integraatio

TIIVISTELMÄ

Opinnäytetyön tavoitteena oli selvittää, kuinka voitaisiin toteuttaa Magento-verkkokauppajärjestelmän ja Moodle-oppimisalustan välinen integraatio ja tämän selvitysprosessin kautta saadun tietotaidon avulla toteuttaa näiden kahden järjestelmän välinen integraatiomoduuli Magentoon. Integraatiomoduulin avulla Magentosta kurssituotteen ostava asiakas saisi itselleen tunnuksen ja osallistumisoikeuden vastaavalle Moodle-oppimiasalustalle luodulle kurssille.

Avoimeen lähdekoodiin pohjautuvien ohjelmisto- ja verkkokauppajärjestelmien suosio ovat viimeisen vuosikymmenen aikana kasvaneet merkittävästi. Näistä järjestelmistä erityisesti Magenton ja Moodlen jatkuva yleistyminen ovat luoneet kasvavan kysynnän edellä mainittujen järjestelmien välisen integraation toteuttamiseksi.

Työn toimeksiantaja Tawasta OS Technologies on ohjelmistoalan yritys, jolla on pitkäaikainen kokemus Magentoon pohjautuvien verkkokauppa-ratkaisuiden ja -sivustojen toteutuksesta. Yritys toimii osana Mediamaisteri Groupia, jonka yhtenä olennaisimpana osana puolestaan on laaja-alainen ja vankka kokemus Moodleen liittyvistä toteutuksista.

Koska aikaisempaa tietoa tai toteutusmalleja vastaavan integraation toteutuksesta järjestelmien välillä ei ollut projektin aloittamisen aikaan saatavilla, perehdyttiin työssä aluksi yleisesti Magenton moduulinkehitykseen, hakemisto- ja tiedostorakenteeseen sekä sen ohjelmistoarkkitehtuuriin. Lisäksi projektin aikana tutustuttiin tarkemmin Moodle-oppimisalustan tarjoamiin integraatiorajapintoihin ja funktioihin. Näitä tietoja ja tekniikoita soveltamalla saavutettiin tarvittava tietotaito integraation ja integraatiomoduulin toteuttamiseen. Työn lopputuloksena toteutettiin projektin alussa sille määritellyt toiminnallisuudet sisältävä integraatiomoduuli, joka projektin jälkeen on jo otettu käyttöön myös tuotantokäytössä.

Avainsanat magento, moodle, avoin lähdekoodi, integraatio, moduulikehitys

Sivut 28 s. + liitteet 1 s.

HAMK
UNIVERSITY OF APPLIED SCIENCES

ABSTRACT

Visamäki
Degree Programme in Business Information Technology
System Engineering

Author Kimmo Kava **Year** 2014

Subject of Bachelor's thesis Magento-Moodle integration

ABSTRACT

During the past decade, the popularity of open source based solutions and buying and selling of goods and services online has increased significantly. Of these open source based solutions, the increased use of Magento e-commerce and Moodle learning management systems (LMS) have created a growing demand for the integration of these two popular platforms.

The commissioner of this thesis, Tawasta OS Technologies, is a software company, which has a long term experience in creating Magento based e-commerce solutions. The company operates as a subsidiary of Mediamaisteri Group's business cluster, which in turn has a wide-ranging and extensive experience in implementing Moodle platform based web service solutions.

The aim of this thesis was to find out, how it would be possible to integrate Magento e-commerce and Moodle LMS platforms, and based on this study, to create an integration module for Magento. The idea of the integration module is to automatically create a user ID to Moodle when a customer buys a course product from Magento, and also give this user ID the rights to attend the corresponding course in Moodle.

Since there were no prior knowledge or implementation models available for this kind of integration at the start of this project, the first phase was to study and learn the basics and general principles of Magento module development, Magento coding standards and Magento's software architecture in general. In addition to this, the second important phase was to study the Moodle LMS integration interfaces and protocols, and the related web service functions. Using and applying the knowledge and techniques gathered from these studies formed the basis for creating the integration module. The final result of the thesis was a completed integration module, with all the functionalities that were set for it at the beginning of the project. The module has also already been implemented to production use.

Keywords magento, moodle, open source, e-commerce, integration

Pages 28 p. + appendices 1 p.

Keskeisiä käsitteitä

Magento	Suosittu avoimeen lähdekoodiin pohjautuva verkkokauppajärjestelmä.
Moodle	Avoimeen lähdekoodiin pohjautuva oppimisalusta ja koulutuksenhallintajärjestelmä.
E-commerce	Verkossa tapahtuva tuotteiden ja palveluiden myyminen.
Trac	Avointa lähdekoodia hyödyntävä projektinhallintajärjestelmä.
SSH	Secure Shell. Salattu verkkoprotokolla, jota käytetään yleisesti turvalliseen tiedonsiirtoon ja palvelinkomentojen suorittamiseen.
REST	Representational State Transfer. Arkkitehtuurimalli, jota hyödynnetään esimerkiksi Web service -ohjelmistorajapintojen toteutuksessa.
Web service	Ohjelmistojärjestelmä/-rajapinta, jonka avulla voidaan välittää dataa ja ohjelmallisia kutsuja eri verkkosovellusten välillä.
W3C	World Wide Web Consortium. Kansainvälisesti toimiva yhteisöjen ja yritysten muodostama yhteenliittymä, joka kehittää ja ylläpitää www-tekniikkaan liittyviä standardeja.

SISÄLLYS

1 JOHDANTO 1

2 AIHEEN ESITTELY JA PROJEKTIN TAUSTAT 2

2.1 Toimeksiantaja 2
2.2 Työn taustat ja historiaa 2
2.3 Sovellukset ja kehitysympäristö 2
2.4 Ohjelmointi- ja merkkauskielet ja rajapinnat 3
2.4.1 PHP 3
2.4.2 XML 3
2.4.3 JSON 3
2.4.4 REST 3

3 WEB SERVICE 4

3.1 Yleistä Web servicestä 4
3.2 Web service -sovelluspalvelun osat ja yleinen toimintaperiaate 4
3.3 Web serviceen liittyviä standardeja 5
3.3.1 SOAP 5
3.3.2 WSDL 5

4 MOODLE 6

4.1 Yleistä Moodlesta 6
4.2 Moodle ja REST-protokolla 7
4.3 Moodlen Web service -funktiot 7

5 MAGENTO 9

5.1 Yleistä Magentosta 9
5.2 Zend-ohjelmistokehys 9
5.3 Magenton MVC-malli 10
5.4 Magenton tiedostorakenne 11
5.4.1 App-kansio 11
5.4.2 Ydin- ja paikalliset moduulit – app/code/core ja local -kansiot 11
5.4.3 Yhteisömoduulit – app/code/community-kansio 12
5.4.4 Moduulirekisteri – app/etc/modules-kansio 12
5.4.5 Käännöstiedostot – app/locale-kansio 12
5.4.6 Teematiedostot – app/design-kansio 13
5.5 Moduulin rakenne 13
5.5.1 Moduulin hakemistorakenne 13
5.5.2 Block-hakemisto 14
5.5.3 controllers-hakemisto 15
5.5.4 etc-hakemisto 15
5.5.5 Helper-hakemisto 16
5.5.6 Model-hakemisto 16
5.5.7 sql-hakemisto 16

6 INTEGRAATIO-MODUULIN SUUNNITTELU JA TOTEUTUS 17

6.1 Moduulin toimintaprosessi, -periaate ja tarkoitus 17
6.2 Moduulin suunnittelu ja toteutusvaiheet 18
6.3 Moduulin hakemistorakenne ja tiedostot 19

6.3.1 controllers-hakemisto 20
6.3.2 etc-hakemisto 20
6.3.3 Helper-hakemisto 21
6.3.4 Model-hakemisto 21
6.3.5 sql-hakemisto 21
6.3.6 etc/modules-hakemisto 22
6.4 Moduulin globaalien asetusten määrittäminen 22
6.5 Moduulin toiminnallisuudet 23
6.5.1 Moodle-kurssi tuoteattribuuttien luominen 23
6.5.2 Moodlen Web service -funktioiden kutsuminen 24
6.5.3 Käyttäjätunnuksen luominen ja osallistujaksi lisääminen kurssille 25

7 YHTEENVETO 26

7.1 Tulokset ja saavutetut edut 26
7.2 Haasteet 26
7.3 Jatkokehitys 26

LÄHTEET 28

Liite 1 ADDNEWUSERACTION -FUNKTIO

1 JOHDANTO

Verkkokauppojen ja sähköisen kaupankäynnin suosio on räjähdysmäisesti kasvanut viimeisen vuosikymmenen aikana. Joidenkin viimeisimpien tutkimusten mukaan jopa yli 75 % suomalaisista on tehnyt ostoksia verkossa (Verkkokauppatilasto 2013. Perustietoa verkkokauppaseurannasta sekä ensimmäinen vuosipuolisko 2013). Samaan aikaan tarve sähköisten oppimisalustojen ja koulutuksenhallintajärjestelmien ja verkkokaupan yhdistämisestä ja tätä kautta koulutuksien myymisestä verkossa on tullut koko ajan ajankohtaisemmaksi. Tästä nimenomaisesta tarpeesta sai myös alkunsa tämä opinnäytetyö, jonka toimeksiantajana toimi työantajani Tawasta OS Technologies.

Opinnäytetyön tavoitteena oli selvittää, kuinka olisi mahdollista toteuttaa toimiva integraatio Magento-verkkokauppajärjestelmän ja Moodle-oppimisalustan välille, ja toteuttaa tätä integraatiota varten moduuli Magentoon. Moduulin avulla Magentosta kurssituotteen ostavalle asiakkaalle luotaisiin automaattisesti käyttäjätili Moodleen, ja samalla käyttäjätili lisättäisiin myös automaattisesta osallistujaksi ostamalleen Moodle-kurssille. Tarkemmin sanottuna integraation tarkoituksena oli siis luoda yhteys Magentosta ostettavan kurssituotteen ja Moodleen luodun kurssin välille, sekä toteuttaa samaan yhteyteen toiminnallisuus Moodle-käyttäjätilin luontiin ja osallistujaksi lisäämiseen.

Opinnäytetyössä käydään Magento-Moodle-integraation lisäksi yleisellä tasolla läpi niin Magento-verkkokauppajärjestelmän kuin Moodle-oppimisalustankin perustoiminnallisuuksia sekä hieman tarkemmin myös Moodlen REST-protokollaa, Web service -funktioita, Magenton moduulinkehitystä sekä sen moduulien hakemisto- ja tiedostorakenteita.

Integraatio toteutettiin pitkälti hyödyntämällä Magenton moduulinkehityksen yleisiä toimintamalleja, PHP-ohjelmointikieltä, REST-protokollaa sekä Moodlen Web service -funktiota. Moodle-oppimisalustaan liittyvissä teknisissä kysymyksissä saatiin projektin aikana tukea emoyhtiö Mediamaisteri Groupin Moodle-asiantuntijoilta.

Opinnäytetyö pyrkii vastaamaan seuraaviin tutkimuskysymyksiin: Miten ja millä tekniikoilla on mahdollista toteuttaa Magento- ja Moodle-järjestelmien välinen integraatio? Minkälaisista ominaisuuksista ja toiminnallisuuksista integraatiomoduuli koostuu? Minkälainen on integraatioon kehitettävän moduulin tiedostorakenne / arkkitehtuuri? Millä tavoin integraatiomoduulin ominaisuuksia / toiminnallisuuksia voitaisiin edelleen jatkokehittää vastaamaan paremmin loppukäyttäjän toiveita/tarpeita?

2 AIHEEN ESITTELY JA PROJEKTIN TAUSTAT

Tässä luvussa esitellään tarkemmin itse integraatioprojektin, toimeksiantajan ja työn taustat sekä kuinka ja miksi integraatiomoduulia päädyttiin rakentamaan. Lisäksi luvussa käydään läpi projektissa käytetyt sovellukset ja kehitysympäristö sekä työssä käytetyt ohjelmointi- ja merkkauskielet ja rajapinnat.

2.1 Toimeksiantaja

Tawasta OS Technologies on vuonna 2004 perustettu Hämeenlinnassa toimiva ohjelmistoalan yritys, joka tarjoaa monipuolisia avoimeen lähdekoodiin perustuvia asiakkaan toivomusten mukaan räätälöitäviä ohjelmistopalveluja. Yrityksen pääasiallisina tuotteina toimivat Joomla-, Drupal-, vTiger-, OpenERP- ja Magento-järjestelmien pohjalta toteutetut ohjelmistoratkaisut sekä niihin liittyvät kattavat ylläpito- ja tukipalvelut. Tawasta on osa Mediamaisteri Group Oy:n yritysketjua, johon kuuluvat mm. Digital Lessons Oy ja OpenTrainers Oy. Intregaatiomoduulin kehittäjä toimii yrityksessä tehtävänimikkeellä Software Specialist. Hänen päätoimenkuvaansa yrityksessä kuuluvat muun muassa edellä mainittujen ohjelmistoratkaisujen suunnittelu, toteutus, asentaminen, konfigurointi ja ohjelmointitehtävät sekä asiakastuki- ja ylläpitotehtävät.

2.2 Työn taustat ja historiaa

Loppuvuodesta 2013 sain toimeksiantajaltani selvitettäväksi, miten olisi mahdollista toteuttaa Magento-verkkokauppajärjestelmän ja Moodle-oppimisalustan välinen integraatio. Tämän tutkimusprosessin aikana sain selville, että näiden järjestelmien välistä suoraa integraatiota varten ei selvityksenteon aikaan ollut saatavilla valmista lisäosaa/moduulia, vaan kyseinen integraatio jouduttaisiin toteuttamaan ns. talon sisäisenä projektina emoyhtiö Mediamaisteri Groupin Moodle-asiantuntijoiden kanssa.

Koin projektin haasteellisena, koska en ollut aikaisemmin kehittänyt tai toteuttanut Magentoon moduuleja/lisäosia. Näin ollen ennen varsinaisen työn aloittamista tutustuin ja opiskelin yleisesti Magenton tiedosto- ja hakemistoarkkitehtuuria, ydin ja yhteisö-moduulien rakennetta ja toimintaperiaatteita sekä lisäosan kehityksen perusteita. Näiden tietojen pohjalta lähdin suunnittelemaan ja toteuttamaan integraatiolisäosaa.

2.3 Sovellukset ja kehitysympäristö

Moduulin kehitystyön aikana tärkeimpiin työvälineisiin lukeutuivat WinSCP-sovellus salatun SSH-palvelinyhteyden ylläpitämiseen kehityspalvelimen ja työkoneen välillä sekä PUTTY ssh-etäkäyttösovellus palvelin- ja tietokantakomentojen syöttämiseen. Ohjelmointitiedostojen luomisessa ja muokkauksessa hyödynnettiin Notepad++-tekstieditoria. Työhön liittyvän dokumentaation tallentamiseen hyödynnettiin toimeksiantajan tarjoamaa avoimeen lähdekoodiin pohjautuvaa Trac-projektinhallintasivustoa.

Työn tekemisessä kehitysympäristönä toimi toimeksiantajan tarjoama Debian 6 -käyttöjärjestelmäpohjainen virtuaalinen kehityspalvelin. Kehityspalvelimella olivat asennettuina työn tekohetkellä viimeisimmät versiot PHP-ohjelmointirajapinnasta ja MySQL-tietokantaohjelmistoista. Kehityspalvelimelle asennettiin työn toteutushetkellä oletusasetuksin Magenton viimeisin saatavilla ollut yhteisöversio (Community Edition) v. 1.8.1.0 sekä Moodle oppimisalustan viimeisin saatavilla ollut versio 2.6.

2.4 Ohjelmointi- ja merkkauskielet ja rajapinnat

Tässä luvussa esitellään lyhyesti opinnäytetyön kannalta oleellisimmat web-ohjelmointi- ja merkkauskielet. Näiden lisäksi esitellään työn kannalta oleellisimmat rajapinnat.

2.4.1 PHP

PHP (Hypertext Preprocessor) on erityisesti palvelinympäristöissä käytettävä ohjelmointikieli dynaamisten web-sivustojen luomiseen. PHP-koodi kirjoitetaan yleensä suoraan HTML-koodin sisään tai omana php-päätteisenä tiedostonaan. Kun PHP-koodia sisältävä web-sivu ladataan, palvelin käsittelee sen ennen varsinaisen HTML-tiedoston lähettämistä. Tästä johtuen sivuston loppukäyttäjä näkee vain tämän prosessin lopputuloksena syntyvän HTML-tiedoston. (PHP (Hypertext Preprocessor) 2006.)

2.4.2 XML

XML (Extensible Markup Language) on laitteisto ja sovellusriippumaton merkintäkieli, jonka avulla tekstimuotoista tietoa voidaan helposti tallentaa ja välittää erilaisten järjestelmien välillä (XML (Extensible Markup Language) 2007).

2.4.3 JSON

JSON (JavaScript Object Notation) on kevyt tiedonsiirtoformaatti, joka nimestään huolimatta on JavaScriptistä riippumaton, eli sitä voidaan käyttää myös muilla ohjelmointikielillä kuten esim. C, C++, C# ja Java. JSON perustuu kokoelmaan nimi/arvo pareja (tekstimuotoisia olioita) sekä listamaiseen (array) arvojen tallennustapaan.

2.4.4 REST

REST (Representational State Transfer) on HTTP-protokollaan perustuva arkkitehtuurimalli, jota hyödynnetään muun muassa Web service ohjelmistorajapintojen toteutuksessa. Sen avulla voidaan myös lukea, luoda, muokata ja poistaa tietoa yksinkertaisten palvelimella tehtävien HTTP-kutsujen avulla. Näin ollen sitä itsessään voidaan myös tarpeen vaatiessa hyödyntää rajapintana Web service -toteutuksissa.

3 WEB SERVICE

Tässä luvussa kuvataan lyhyesti, mikä Web service eli verkkopohjainen sovelluspalvelu on ja mihin sitä voidaan käyttää. Lisäksi käydään lyhyesti läpi Web service -sovelluspalvelun olennaisimmat osat ja esitellään yleisimmät Web serviceen liittyvät W3C:n määrittelemät standardit.

3.1 Yleistä Web servicestä

Web service on termi, jolla IT-maailmassa tarkoitetaan yleensä verkkopohjaisia sovelluspalveluita ja ohjelmistorajapintoja. Käytännössä nämä ohjelmistorajapinnat mahdollistavat joko eri tai samalla palvelimella sijaitsevien tietokoneiden, ohjelmistojen tai tietojärjestelmien välisen yhteistoiminnan ja tietojenvälityksen HTTP tai muun internet-pohjaisen protokollan kautta (Web Services Architecture 2004).

Yleensä syy Web service -ohjelmistosovelluksen/-rajapinnan käyttöönottoon on tarve kahden erillisen verkossa toimivan järjestelmän toiminnallisuuksien ja rajapintojen yhdistämiseksi. Esimerkiksi Magento-verkkokauppajärjestelmän ja Moodle-oppimisalustan välillä voidaan Web service -ohjelmistosovellustekniikoita ja -funktiota hyödyntämällä rakentaa mainittujen järjestelmien välinen integraatio, jossa toisen järjestelmän kautta kutsuttujen funktioiden avulla voidaan käyttää toisen järjestelmään liittyviä toimintoja. Käytännön esimerkki tällaisesta toiminnallisuudesta voisi olla Magentoon tilauksen yhteydessä syötettyjen käyttäjätietojen hyödyntäminen uuden käyttäjätunnuksen luomisessa Moodleen.

3.2 Web service -sovelluspalvelun osat ja yleinen toimintaperiaate

Web service -sovelluspalvelu tarvitsee toimiakseen palvelun tarjoajan (Service provider) ja käyttäjän (Serivce requester) sekä näiden kahden välisestä kommunikaatiosta huolehtivan agentin (agent). Tämä agentti on käytännössä sovellus tai laitteisto, jonka tarkoituksena on välittää ja vastaanottaa viestejä palvelun tarjoajan ja käyttäjän välillä (Web Services Architecture 2004). Näiden osien välillä tapahtuva kommunikaatio puolestaan tapahtuu käytännössä erilaisten XML tai HTTP-pohjaisten protokollien välityksellä.

On olemassa useita erilaisia tapoja, joilla edellä mainitut osat (palvelun tarjoaja, käyttäjä ja agentti) voidaan saada toimimaan keskenään Web service -toteutuksessa. Yleensä kuitenkin näiden osien välinen toimintaprosessi noudattaa alla kuvattua kaavaa.

Ensimmäisessä vaiheessa palvelun tarjoaja ja käyttäjä tulevat jollain tapaa toisilleen tunnetuksi (tai vähintään toinen näistä ns. tuntee toisen). Toisessa vaiheessa palvelun tarjoaja ja käyttäjä sopivat jollain tapaa sovelluspalvelun toiminnoista, joilla hallinnoidaan näiden kahden osan välistä toimintaa. Kolmannessa vaiheessa palvelun tarjoajan ja käyttäjän agentit toteuttavat käytetyn sovelluspalvelun mukaiset toiminnot. Lopuksi palvelun tarjoajan ja käyttäjän puoleiset agentit välittävät jonkin viestin järjestelmien

välillä, suorittaen jonkin toiminnon toisessa tai molemmissa viestin vastaanottavista järjestelmistä (Web Services Architecture 2004).

3.3 Web serviceen liittyviä standardeja

Tässä luvussa esitellään lyhyesti W3C:n Web servicelle määrittelemiä ja siinä hyödynnettäviä kriittisimpiä standardeja. XML-merkkauskieli sivuutetaan tässä luvussa, koska kyseinen tekniikka on jo esitelty aiemmin luvussa 2.4.2.

3.3.1 SOAP

SOAP (Simple Object Access Protocol) on hyvin yleisesti käytetty tiedoliikenneprotokolla, jonka avulla Web service -palvelun käyttäjä (Service requester) voi lähettää komennon palvelun tarjoajalle (Service provider). Samaa tekniikkaa hyödyntämällä palvelun tarjoaja pystyy puolestaan välittämään vastauksen palvelun käyttäjälle XML-muotoisessa paketissa.

SOAP-protokolla pohjautuu XML-merkkauskieleen, ja sille löytyy myös omat toteutuksensa useimmista yleisimmin käytetyistä ohjelmointikielistä, kuten C# ja Java. Lisäksi se voidaan saada toimimaan useiden erilaisten protokollien yli. Yleisimmin Web service -toteutuksissa käytetään kuitenkin HTTP-protokollaa.

3.3.2 WSDL

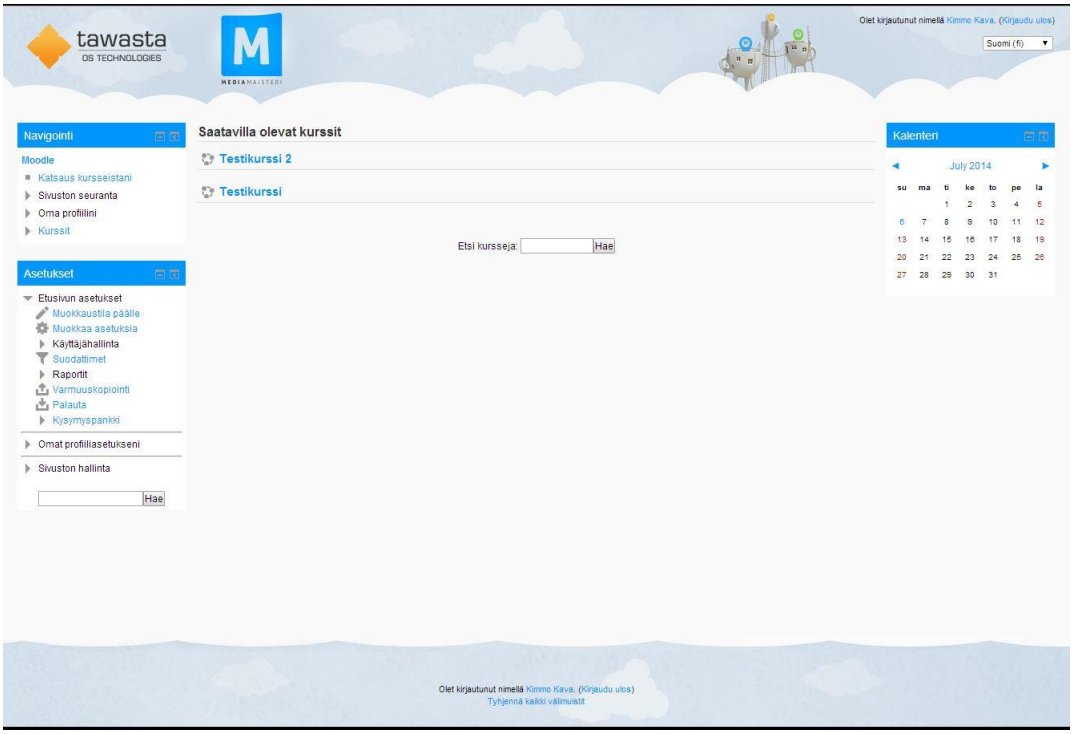
WSDL (Web Service Description Language) on SOAP-protokollan tavoin XML-pohjainen kieli, jonka avulla SOAP:in kautta tarjotun Web service -sovelluspalvelun pyyntö- ja vastausmuodot sekä formaatit on mahdollista määritellä. WSDL:n kautta määritetty kuvaus puolestaan koostuu abstraktista ja konkreettisesta osasta. Abstraktissa osassa kuvataan palvelun rajapinta sitomatta sitä vielä mihinkään protokollaan tai osoitteeseen. Konkreettisessa osassa taas kuvataan palvelussa käytettävä protokolla ja viesteissä käytetty muoto/formaatti sekä määritetään palvelussa käytettävä verkko-osoite.

4 MOODLE

Tässä luvussa esitellään lyhyesti, mikä on Moodle-järjestelmä ja mikä sen käyttötarkoitus on. Lisäksi selvitetään opinnäytetyön kannalta oleellisten Moodlen ja REST-protokollan sekä integraatiomoduulissa käytettyjen Moodlen Web service -funktioiden toimintaa.

4.1 Yleistä Moodlesta

Moodle on alkujaan Australiassa vuonna 1999 kehitetty täysin ilmainen avoimeen lähdekoodiin ja pääosin PHP-ohjelmointikieleen pohjautuva virtuaalinen oppimisalusta ja koulutuksenhallintajärjestelmä (Learning Management System). Ensimmäinen järjestelmää käyttänyt sivusto ja vapaasti ladattavissa ollut versio julkaistiin jo vuonna 2001. (About Moodle – History 2014.) Opinnäytetyöraportin kirjoittamisen aikaan järjestelmän viimeisin saatavilla oleva versio on 2.7.



Kuva 1. Opinnäytetyön tekemisessä käytetyn Moodle testiympäristön hallintanäkymä.

Moodle on suunniteltu erityisesti käytettäväksi oppilaitoksissa ja kouluissa, joissa opettajat ja opiskelijat toimivat jatkuvassa vuorovaikutuksessa keskenään. Sillä on mahdollista luoda monipuolisia kurssi- ja wikisivustoja.

Moodle on tällä hetkellä yksi maailman käytetyimmistä ja nopeimmin kehittyvistä koulutuksenhallintajärjestelmistä. Se on käytössä yli 200 maassa ja maailmanlaajuisesti sillä on jo yli 70 miljoonaa käyttäjää ja 65 000 rekisteröityä sivustoa. (Moodlen tilastot, 2014.) Suomessa Moodle-järjestelmä on käytössä opinnäytetyöraportin kirjoitushetkellä yhteensä 288 sivustolla (Registered moodle sites 2014).

4.2 Moodle ja REST-protokolla

Moodlessa käytettävä REST-protokolla hyväksyy internet-selaimen kautta GET/POST metodien avulla Moodlen Web service -sovellukselle (client) välitettäviä kutsuja/parametreja, jotka palauttavat joko XML tai JSON muotoisen arvon/vastauksen. REST-protokolla ja siihen liittyvät toiminnallisuudet, kuten muutkin Web service -protokollat, otetaan Moodlessa käyttöön järjestelmään asennettavina liitännäisinä (plugin). Näiden liitännäisten toimintaperiaatteena on välittää ohjelmallisia kutsuja ulkoisten sovellusten ja Moodle-järjestelmään koodattujen Web service -funktioiden välillä.

Moodlen REST-protokollan toiminnallisuuden perustana toimivat Web service -sovellus (client), jonka kautta Web service -funktioita käytännössä kutsutaan, Web service -käyttäjätunnus sekä Web service -käyttäjätunnukselle luotava salausavain (security token). Kun käyttäjätunnukselle on määritetty salausavain, voidaan sille määritellä oikeudet Web service -funktioihin, joita sen on sallittu käyttää. Alla on esitetty lyhyt esimerkkikuvaus siitä, miten REST-protokollan käyttö Moodlessa käytännössä toimii.

Käytännön esimerkki REST-protokollan toiminnasta Moodlessa voisi esimerkiksi mennä seuraavasti: Web service -käyttäjätunnukselle luodaan oma yksilöllinen salausavain. Tämän jälkeen Web service -sovellus kutsuu haluttua Web service -funktiota REST-protokollapalvelimelta ja välittää sille käyttäjän salausavaimen. Seuraavaksi protokollapalvelin tarkistaa vastaanotetun salausavaimen avulla, onko kyseisellä käyttäjällä oikeuksia kutsua haluttua funktiota. Tämän jälkeen protokollapalvelin kutsuu kyseistä Web service -funktiota, jonka jälkeen kutsuttu Web service -funktio tarkistaa, onko käyttäjällä oikeudet suorittaa ko. funktiota ja tekee sitten kutsun Moodlen vastaavaan sisäiseen (core) tai paikalliseen (local) funktioon. Seuraavaksi Moodlen sisäinen funktio palauttaa vastauksen protokollapalvelimelle. Lopuksi protokollapalvelin palauttaa funktion vastauksen Web service -sovellukselle.

4.3 Moodlen Web service -funktiot

Moodlen Web service -funktiot ovat käytännössä metodeja, joilla voidaan suorittaa Moodle-järstelmän toimintoja, kuten opiskelijan tai uuden kurssin luominen, jonkin ulkoisen järjestelmän kautta.

Integraatiomoduulissa hyödynnetään seuraavia Moodlesta oletuksena löytyviä Web service -funktioita sekä yhtä integraatiomoduulia varten luotua funktiota:

Taulukko 1. Integraatiomoduulin käyttämät Moodlen Web service - funktiot.

Funktion nimi	Tyyppi	Kuvaus	Vaaditut oikeudet
core_user_create_users	Core	Käyttäjien luominen.	moodle/user:create
core_user_get_users	Core	Palauttaa annettuja parametreja vastaavien käyttäjien tiedot.	moodle/user:viewdetails, moodle/user:viewhiddendetails, moodle/course:useremail, moodle/user:update
core_course_get_courses	Core	Palauttaa haetun / haettujen kurssin / kurssien tiedot.	moodle/course:view,moodle/course:update,moodle/course:viewhiddencourses
enrol_manual_enrol_users	Core	Lisää käyttäjän kurssille.	enrol/manual:enrol
local_tawasta_get_course_enrolment_duration	Local	Palauttaa kurssin ilmoittautumisen oletusvoimassaoloajan keston.	enrol/manual:config

Core_user_create_users -funktio huolehtii nimensä mukaan uuden käyttäjätunnuksen luomisesta Moodle-järjestelmään. Funktiolle tulee antaa vähintään seuraavat tekstijono-muotoiset parametrit: käyttäjänimi, salasana, etunimi, sukunimi, kokonimi ja sähköpostiosoite. Funktio hyväksyy vaihtoehtoisina parametreina myös muita käyttäjäjäätunnukseen liittyviä tietoja, kuten esimerkiksi Moodle-järjestelmässä käytettävän oletuskielen/-teeman sekä käyttäjän kuvauksen tai kotikaupungin/-maan. Core_user_get_users-funktio puolestaan palauttaa funktiolle annettujen käyttäjätunnukseen liittyviä parametreja vastaavien käyttäjätunnuksien tiedot. Kyseiset parametrit tulee antaa funktiolle avain-arvopari muotoisina:

Esimerkki 1. core_user_get_users-funktiolle annettava arvo-arvopari parametri.

```
 'key'=>'id', 'value'=> 2
```

Core_course_get_courses-funktio toimii hieman vastaavalla periaatteella, mutta sillä erotuksella, että funktiolle annetaan parametrina taulukkomuotoinen lista yksilöllisistä kurssitunnuksista (course id), joiden tiedot halutaan palauttaa. Enrol_manual_enrol_users-funktiolla jo olemassa oleva käyttäjätunnus voidaan lisätä osallistujaksi halutulle kurssille.

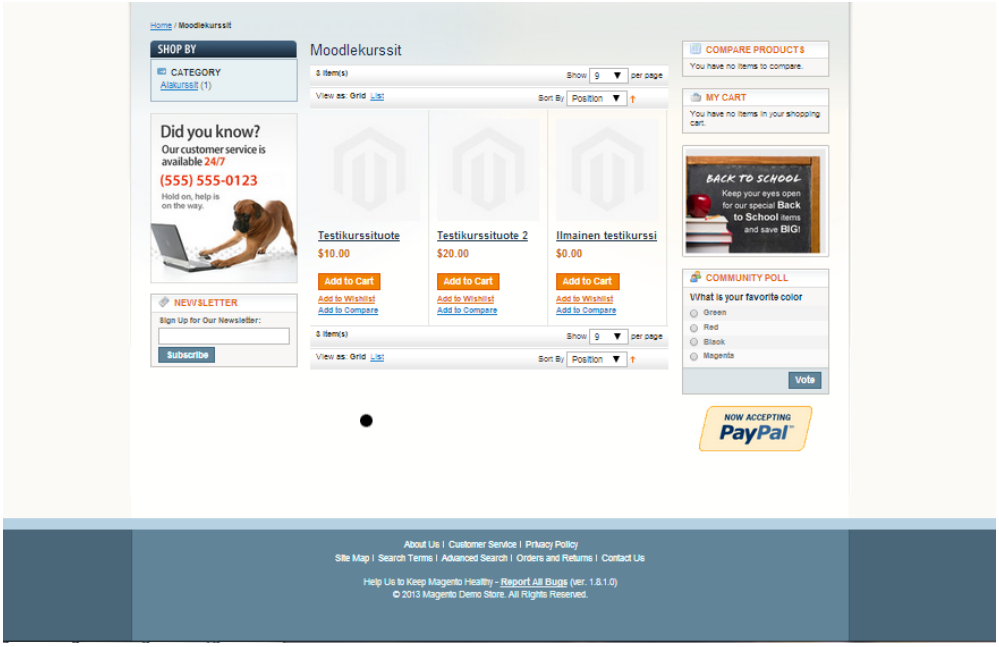
Local_tawasta_get_course_enrolment_duration-funktio on puolestaan integraatiolisäosaa varten Moodleen luotu uusi Web service -funktio, jolla voidaan hakea tietyn kurssin ilmoittautumisen oletusvoimassaoloajan kesto. Tämä tieto on oleellinen, jotta integraation kautta kurssille lisättävälle käyttäjätunnukselle voidaan määrittää aika, johon asti kyseisellä tunnuksella on pääsyoikeus kurssille.

5 MAGENTO

Tässä luvussa esitellään lyhyesti, mikä Magento-verkkokauppajärjestelmä on. Lisäksi luvussa käydään läpi Magenton arkkitehtuuria ja rakennetta sekä Magenton moduulinkehitykseen yleisesti liittyviä osia.

5.1 Yleistä Magentosta

Magento on yksi tämän hetken suosituimmista avoimeen lähdekoodiin pohjautuvista verkkokauppa-alustoista. Se pohjautuu teknisesti MVC-malliin ja PHP-ohjelmointikieleen ja sen yhteisöversio (Community Edition) on täysin ilmainen. Magento tarjoaa verkkokaupan pitäjälle erittäin monipuoliset työkalut harjoittaa ja ylläpitää verkkokauppaa.



Kuva 2. Opinnäytetyössä hyödynnetyn Magento-testiympäristön julkinen näkymä.

Magenton suurimmat edut verrattuna sen kilpailijoihin ovat järjestelmän jo valmiiksi mukana tulevat kattavat ominaisuudet, siihen saatavilla olevat lukuisat lisäosat, järjestelmän helppo muokattavuus sekä vahva yhteisön tuki. Magento on alkujaan Varien nimisen web-ohjelmistoyrityksen kehittämä. Nykyään järjestelmän omistaa eBay Inc.

5.2 Zend-ohjelmistokehys

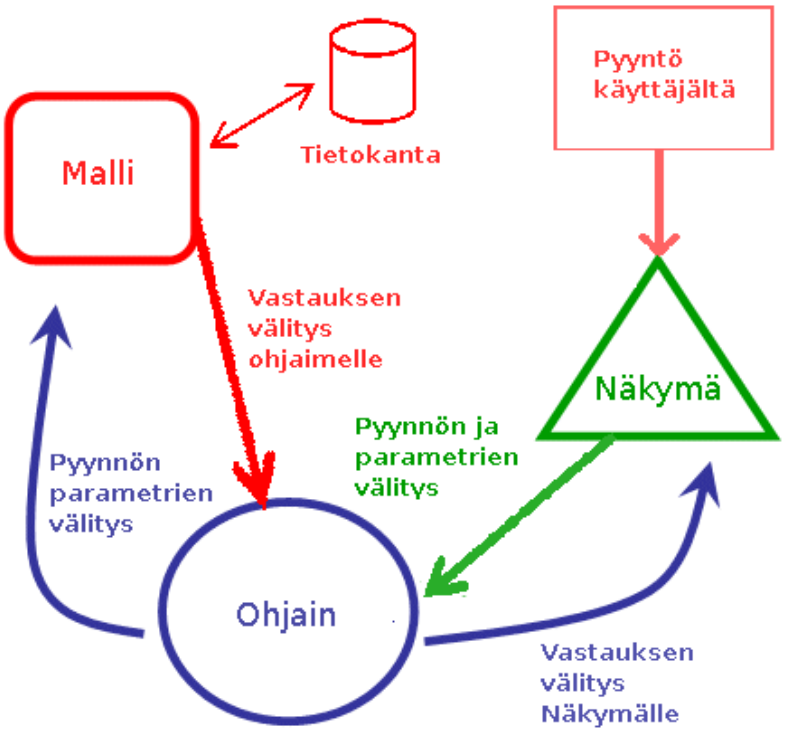
Magenton toiminta perustuu pitkälti Zend-ohjelmistokehykseen, jonka pohjalle Magento-järjestelmä on rakennettu. Zend-ohjelmistokehys, tai lyhyemmin pelkkä Zend, on avointa lähdekoodia hyödyntävä täysin oliopohjainen (Object Oriented) PHP-ohjelmointikieleen pohjautuva ohjelmistokehys (framework).

Zend noudattaa toiminnassaan ja koodissaan MVC-suunnittelumallia. Sen eduksi muihin vastaavankaltaisiin PHP-ohjelmistokehyksiin nähden voidaan laskea olio-ohjelmointimallin kautta saavutettu järjestelmän osien modulaarisuus, sekä kehyksen mukana tulevat laajat ja täysin käyttäjän laajennettavissa olevat koodikirjastot.

5.3 Magenton MVC-malli

MVC-malli on ohjelmoinnissa hyödynnettävä suunnittelumalli, jonka avulla sovellusohjelman koodi on jaettu kolmeen loogiseen osaan, Malliin (Model), Näkymään (View) ja Ohjaimeen (Controller). MVC-mallissa sovelluksen koodi pyritään siis ideaalisesti kirjoittamaan sen tarkoituksen pohjalta omiin luokkiinsa.

MVC-mallin Malli koostuu luokista, jotka säilyttävät tietoa ja toteuttavat varsinaiset toiminnallisuudet. Näkymän tehtävänä on huolehtia tietojen esittämisestä käyttäjälle. Ohjain puolestaan on käytännössä vastuussa Näkymän ja Mallin välisestä tiedonvälityksestä. Alla olevassa kuvassa on esitetty MVC-mallin toimintaperiaate Magenton näkökulmasta.



Kuva 3. MVC-mallin toimintaperiaate Magentossa

Magenton moduulien rakenne ja suunnittelu pohjautuu pitkälti konventionaaliseen MVC-malliin joitakin erityisiä poikkeuksia lukuun ottamatta. Toisin kuin konventionaalisessa MVC-arkkitehtuurimallissa, jossa järjestelmä hakee automaattisesti uudet luokat hyödyntämällä esiasetettuja polkuja, Magentossa käytössä olevassa MVC-mallissa vastaava toiminnallisuus hoidetaan keskitetysti kokoonpanotiedostojen avulla. Nämä kokoonpanotiedostot ovat käytännössä moduulikohtaisia xml-tiedostoja, joiden kautta Magentolle kerrotaan jokaisesta uudesta luokasta, mallista ja käsittelijästä. Kokoonpanotiedostoista ja niitten sisällöstä puhutaan tarkemmin myöhemmissä luvuissa.

5.4 Magenton tiedostorakenne

Tässä luvussa käydään tarkemmin läpi Magento-järjestelmän tiedostorakennetta ja niiden toiminnallisuuksia. Luvun tarkoituksena ei ole käydä syväluotaavaa analyysiä koko Magenton tiedostorakenteesta, vaan ainoastaan niistä osista, jotka ovat keskeisimpiä Magenton sisältämien moduulien ja moduulinkehityksen kannalta.

5.4.1 App-kansio

Magenton moduulien kansiorakenteen pohjalla toimii jokaisen Magento-asennuksen juuressa sijaitseva app-kansio. Tämä kansio sisältää käytännössä kaikki moduuleiden toiminnallisuuksiin liittyvät kooditiedostot, asennuksen julkisen sekä hallintapuolen teemoihin liittyvät tiedostot (pois lukien teemoihin liittyvät tyyli- ja kuvatiedostot ja mahdolliset JavaScript-kirjastot) sekä Magenton kieli-/käännöstiedostot. Lisäksi se sisältää moduulirekisterin, jossa listataan kaikki Magentoon asennetut moduulit. Seuraavissa luvuissa käsitellään tarkemmin app-kansion alikansioita ja niiden sisältämiä toiminnallisuuksia.

5.4.2 Ydin- ja paikalliset moduulit – app/code/core ja local -kansiot

Magenton ydin- eli core-moduulit sijaitsevat Magenton app/code/core-kansiossa. Ydinmoduulit käytännössä vastaavat kaikesta Magenton ohjelmallisista perustoiminnallisuuksista. Magentoon koodattaviin uusiin toiminnallisuuksiin tai sen jo olemassa olevien toiminnallisuuksien parannuksiin liittyvät moduulit, joita ei ole tarkoitettu jaettavaksi Magento-yhteisöön, sijoitetaan yleensä app/code/local-kansioon.

Yleinen nyrkkisääntö on, ettei Magento-kehittäjän koskaan tulisi muokata suoraan ydinmoduulien (app/code/core) tiedostoja. Tämä siksi, että mikäli Magenton ydinmoduulin tiedostoja suoraan muokattaisiin, seuraavat Magenton versiopäivitykset eivät olisi enää yhteensopivia kyseisessä asennuksessa. Tällöin ne aiheuttaisivat päivityksen käyttöönoton jälkeen todennäköisesti vakavia yhteensopivuusongelmia.

Mikäli Magenton perustoiminnallisuuksiin halutaan tehdä muutoksia turvallisesti kajoamatta ydinmoduuleiden tiedostoihin, tai luoda kokonaan uusi moduuli, tulee nämä muutokset/lisäykset ja uudet moduulit tehdä jo-

ko paikallisten moduulien (app/code/local) tai yhteisömoduulien (app/code/community) kansioon. Kaikki näihin kansioihin kirjoitettu koodi ylittää ydinmoduulissa olevat toiminnallisuudet. Kun Magenton toimintoihin liittyvät muutokset tehdään yllä kuvatulla tavalla, säilyy asennus päivityskelpoisena ja mahdolliset ongelmatilanteet on helpompi paikallistaa.

5.4.3 Yhteisömoduulit – app/code/community-kansio

Magento-yhteisön kautta saatavilla olevat moduulit sijaitsevat järjestelmän kansiossa app/code/community. Nämä moduulit ovat useimmiten suurten tai keskisuurten kaupallisten Magento-kehitysfirmojen maksullisia lisäosia, jotka lisäävät Magentoon siitä oletuksena puuttuvia toiminnallisuuksia.

5.4.4 Moduulirekisteri – app/etc/modules-kansio

Magenton moduulirekisterin kautta voidaan selvittää, mitä moduuleja Magentoon on asennettu, missä ne sijaitsevat ja mitkä niistä ovat käytössä. Jokainen Magentoon asennettu moduuli löytyy moduulirekisteristä omana xml-tiedostonaan. Mikäli moduulilla ei ole tässä kansiossa omaa tiedostoa, Magento ei voi löytää sitä, eikä moduuli näin ollen voi myöskään toimia.

Esimerkki 2. Tawasta_Helloworld.xml

```
<config>
    <modules>
        <Tawasta_Helloworld>
            <active>true</active>
            <codePool>local</codePool>
        </Tawasta_Helloworld>
    </modules>
</config>
```

Moduulin rekisteritiedoston nimi muodostetaan yleensä nimiavaruudesta ja moduuli nimestä (Nimiavaruus_ModuulinNimi.xml). Tiedosto sisältää moduulista vain kaksi tietoa: moduulin tilan (aktiivinen/ei aktiivinen) ja moduulin tiedostojen sijainnin (local/community).

5.4.5 Käännöstiedostot – app/locale-kansio

Magenton kieli-/käännöstiedostot sijaitsevat app/locale-kansiossa. Kansio sisältää jokaisen Magentoon asennetun kielen käännöstiedostot omassa kielikoodin mukaisesti (esim. suomenkieliset käännökset löytyvät kansiosta fi_FI) nimetyssä alikansiossaan.

Jokainen käännöstiedosto on nimetty sen moduulin mukaan, jonka toiminnallisuuksiin käännökset liittyvät. Esimerkiksi tuotteisiin ja tuotenäkymiin liittyvät käännökset löytyvät tiedostostosta Mage_Catalog.csv.

5.4.6 Teematiedostot – app/design-kansio

Magenton moduulien teematiedostot sijaitsevat app/design-kansiossa. Tämä kansio sisältää kaikki teemoihin liittyvät tiedostot lukuun ottamatta tyyli- ja kuvatiedostoja ja mahdollisia JavaScript-kirjastoja. Kulloinkin käytössä oleva teema määritetään Magenton hallinnan kautta.

Magenton teemat periytyvät app/design/frontend/base-kansion sisältämistä tiedostoista. Mikäli Magentoon luodaan uusi teema, johon halutaan tehdä perusteemasta poikkeavia muutoksia, voidaan näiden teemamuutosten pohjana hyödyntää base-kansion sisältämiä tiedostoja.

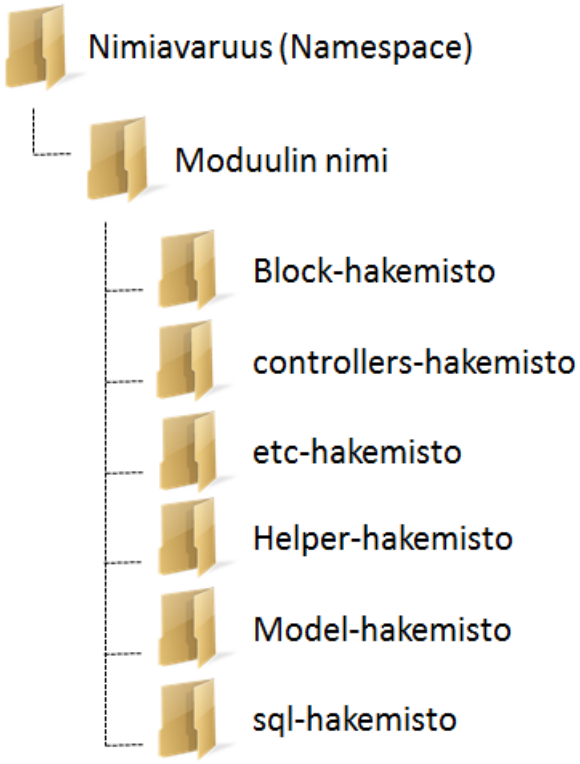
Kuten Magenton ydintiedostojen (app/code/core) kohdalla luvussa 5.4.2 kerrottiin, ei teemaan liittyviä muutoksiakaan kirjoiteta suoraan pohjana toimivaan base-teeman tiedostoihin, vaan ne tulee kopioida ensin uuteen teemaan, jolloin niihin tehdyt muutokset ohittavat niitten pohjalla toimivat tiedostot. Muutoin Magento-päivityksen yhteydessä menetettäisiin teemoihin tehdyt muutokset.

5.5 Moduulin rakenne

Tässä luvussa käydään tarkemmin läpi Magenton moduulien hakemistorakennetta ja niiden sisältämiä toiminnallisuuksia.

5.5.1 Moduulin hakemistorakenne

Kuten luvussa 4.3 mainittiin, Magenton moduulien rakenne pohjautuu tiettyjä eroavaisuuksia lukuun ottamatta konventionaalisen MVC-mallin mukaiseen rakenteeseen. Alla olevassa kuvassa esitellään tarkemmin Magenton moduulin hakemistorakenne.



Kuva 4. Magento-moduulin sisältämät hakemistot

Jokaisella Magenton moduulilla, olipa se ytimeen kuuluva, yhteisön tai paikallisesti luotu, tulee olla määritetty yksilöllinen ja uniikki nimiavaruus (Namespace). Se ja siihen liittyvä hakemisto nimetään yleensä joko esim. moduulin luoneen yrityksen nimen tai vastaavan tiedon mukaan. Nimiavaruus-hakemiston alle luodaan kaikki moduulin liittyvät alihakemistot, joista tarkemmat kuvaukset selvitetään tämän osion tulevissa luvuissa.

Kaikkia edellä mainittuja kansiota ei välttämättä vaadita moduulin toiminnan kannalta. Jos esimerkiksi moduulille ei syystä tai toisesta ole tarvetta lisätä julkiselle tai admin-hallintanäkymään graafista näkymää, ei siinä tapauksessa ole välttämätöntä luoda moduulille Block-hakemistoa.

5.5.2 Block-hakemisto

Magento-moduulin Block-hakemiston sisältämät toiminnallisuudet vastaavat MVC-mallin Näkymää. Block-hakemiston tiedostojen sisältämän koodin tarkoituksena on siis yhdistää Mallin (Model) kautta saadut tiedot Magenton teeman .phtml-loppuisiin tiedostoihin, joiden kautta saatu tieto esitetään sivuston loppukäyttäjälle Magenton sivuston niin kutsutussa julkisessa näkymässä.

5.5.3 controllers-hakemisto

Magento-moduulin controllers-hakemiston sisältämät toiminnallisuudet vastaavat MVC-mallin Ohjainta. Controllers-hakemisto koostuu yleensä funktiosta, joiden toiminnallisuuksia moduuli voi kutsua selaimen kautta. Jotta funktiota on mahdollista käyttää, on niille ensin luotava reititys moduulin config.xml kokoonpanotiedoston kautta.

Esimerkki 3. config.xml

```
<frontend>
  <routers>
   <tawasta_moodlecourses>
     <use>standard</use>
     <args>
      <module>Tawasta_Moodlecourses</module>
      <frontName>moodlecourses</frontName>
     </args>
   </tawasta_moodlecourses>
  </routers>
</frontend>
```

Tiedoston frontend-tagi kertoo, että kyseinen toiminto on tarkoitus suorittaa sivuston julkisessa näkymässä, esimerkiksi kaupan tuotesivulla. Muita mahdollisia vaihtoehtoja ovat <admin>, eli vain Magenton hallintanäkymässä suoritettavat, sekä <install>, eli moduulin asennuksen yhteydessä suoritettavat toiminnallisuudet. Routers-tagin sisällä tärkeimmät määritykset ovat moodulin nimi (tawasta_moodlecourses ja module) ja frontName-tagi. Tämä nimetään yleensä moduulin nimen mukaan.

Esimerkki 4. controllers-hakemistossa sijaitsevan metodin kutsuminen selaimen kautta

```
http://www.esimerkki.fi/frontName/actionControllerName/actionMethod/
```

Verkko-osoitteen jälkeen annettava ensimmäinen määritys frontName on käytännössä alias, jonka avulla Magentolle välitetään tieto siitä mitä funktiota kutsua, kun käyttäjä avaa tietyn Magento-sivun. ActionControllerName on tiedoston nimi, jossa kutsuttava funktio sijaitsee. Osoitteen viimeinen arvo actionMethod on luonnollisesti suoritettavan funktion nimi.

Funktioiden kutsuminen tapahtuu siis käytännössä menemällä selaimen kautta verkko-osoitteeseen, joka muodostuu moduulin nimestä ja ajettavan funktion nimestä ilman funktion nimen perässä olevaa Action loppuosaa. Itse funktiot kirjoitetaan yleensä tiedostoon IndexController.php.

5.5.4 etc-hakemisto

Magento-moduulin etc-hakemisto sisältää moduulin järjestelmä- (system.xml) ja kokoonpanotiedostot (config.xml). Näistä ainut pakollinen tiedosto on moduulin kokoonpanotiedosto config.xml. Järjestelmätiedosto system.xml sisältää käytännössä kaikki moduulin hallintaan liittyvät määritykset, jotka ovat hallinnoitavissa Magenton hallintanäkymän puolella. Tiedostoa ei ole pakko luoda lainkaan, mikäli moduuliin liittyen ei ole tar-

vetta tehdä sellaisia määrityksiä, jotka nähtäisiin välttämättömiksi hallinnoida Magenon graafisen hallintasivuston puolella.

Konfiguraatiotiedosto config.xml sisältää moduulin nimen ja versiotiedot, moduulin globaalit määritykset sekä moduulin asennukseen liittyvät määritykset. Lisäksi tiedostoon voidaan mm. määrittää esimerkiksi toimintoja, jotka suoritetaan tietyn Magentoon ennalta määritetyn prosessin aikana.

5.5.5 Helper-hakemisto

Magento-moduulin Helper-hakemisto sisältää moduulia varten kirjoitettuja toiminnallisuuksia ja apumetodeja, joiden avulla moduulin kautta voidaan muun muassa hyödyntää Magenton sisäisiä luokkia ja niiden kautta yleisimpiä toimintoja. Helper-hakemisto sisältää yleensä vain yhden tiedoston, Data.php. Tämän tiedoston yksi yleisimmistä käyttötarkoituksista on luoda moduulin käyttöön apufunktioita, joita kutsutaan usein moduulin varsinaisten funktioiden sisällä.

5.5.6 Model-hakemisto

Magento-moduulin Model-hakemiston sisältämät toiminnallisuudet vastaavat MVC-mallin Mallia. Toisin kuin yleisesti MVC-mallissa, Magenton Malli ei suoraan käsittele dataa tai ole yhteydessä tietokantaan tai suorita SQL-kyseilyitä. Tämän sijaan Magenton Mallissa hyödynnetään ORM-ohjelmointitekniikkaa (Object-Relational Mapping).

ORM-tekniikan avulla Magenton Mallissa tietokannan rivejä käsitellään oliona, jonka johdosta tietokantakyselyjä ei tarvitse suorittaa perinteisillä SQL-kyselyillä. Tämä helpottaa ja nopeuttaa huomattavasti Mallin ohjelmoijan työtä, koska se poistaa tarpeen kirjoittaa pitkiä ja monimutkaisia SQL-kyselyitä.

5.5.7 sql-hakemisto

Magento-moduulin sql-hakemisto sisältää moduulin asennuksen ja mahdollisten päivitysten yhteydessä Magenton tietokantaan asennettavien ominaisuuksien ja attribuutteihin liittyvät tiedostot. Sql-hakemiston tiedostot sijoitetaan hakemiston alikansioon, joka nimeämismuoto tulee aina olla moduulinnimi_setup. Itse asennustiedoston nimetään esimerkiksi muotoon install-0.1.0.php, jossa install-sanan perään annetaan asennettavan moduulin versionumero. Päivitystiedosto nimetään puolestaan esim. upgrade-0.1.1-0.1.2.php, jossa nykyisen versionumeron perään annetaan moduulin päivitetyn moduulin versionumero.

6 INTEGRAATIO-MODUULIN SUUNNITTELU JA TOTEUTUS

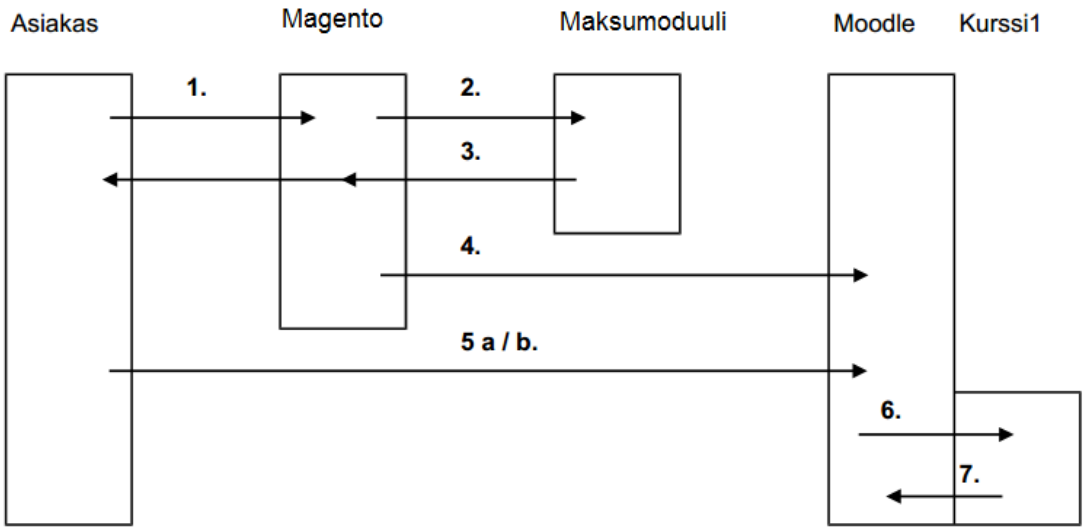
Tässä luvussa käydään läpi integraatiomoduulin suunnittelu- ja toteutusprosessin vaiheet, moduulin toimintaperiaate- ja tarkoitus sekä kuvataan tarkemmin, minkälaisista osista integraatiomoduuli koostuu. Luvussa käydään myös tarkemmin läpi integraatiomoduulin hakemisto- ja tiedostorakennetta ja niihin liittyviä toiminnallisuuksia.

6.1 Moduulin toimintaprosessi, -periaate ja tarkoitus

Magento-Moodle-integraatiomoduulin ideana on luoda yhteys Magento- ja Moodle -järjestelmien sekä Magentoon luodun kurssituotteen ja Moodlessa olevan kurssin välille. Kun verkkokauppa-asiakas ostaa kurssituotteen Magentosta, välittyy tästä integraatiomoduulin kautta automaattisesti tieto Moodleen, jossa REST-rajapintaa hyödyntävät integraatiomoduulin tapahtumankäsittelijän kautta kutsumat Moodlen Web service -funktiot huolehtivat integraation toiminnallisuuksista.

Integraatiomoduulin kautta Magentoon luodaan moduulin asennuksen yhteydessä taulukkolistatyyppinen tuoteominaisuus/-attribuutti, Moodle-courses. Tämän listan sisältönä ovat kaikki integroitavan Moodle-järjestelmän kurssit, jotka on moduulin kautta haetettu kurssitunnuksen perusteella core_course_get_courses web service -funktion avulla. Magenton puolella integroitavaa kurssituotetta luodessa moduuli luo listan sisältämistä kursseista pudotusvalikon, jolta haluttu kurssi valitaan tuotteeseen.

Ostettavaan kurssiin liittyvät muut tiedot eivät kuitenkaan ole järjestelmien kesken yksi yhteen samat. Tämä siksi, ettei verkkokaupan puolella ole luonnollisesti tarvetta olla tietoa esim. kurssin opettajista, tenteistä tai vastaavista tiedoista. Moodleen luodulle kurssille ei vastaavasti taas ole mitään tarvetta määrittää tai siirtää hintatietoja, alennusprosentteja, verosääntöjä tai muita selkeästi vain verkkokauppaan luotuun kurssituotteeseen liittyviä tietoja. Moodlen puolella kaikki verkkokauppaan tulevat kurssituotteet lisätään saman kategorian alle. Alla kuvataan tarkemmin integraatiomoduulin toimintaprosessi Magentosta kurssituotteen ostavan verkkokauppa-asiakkaan näkökulmasta.



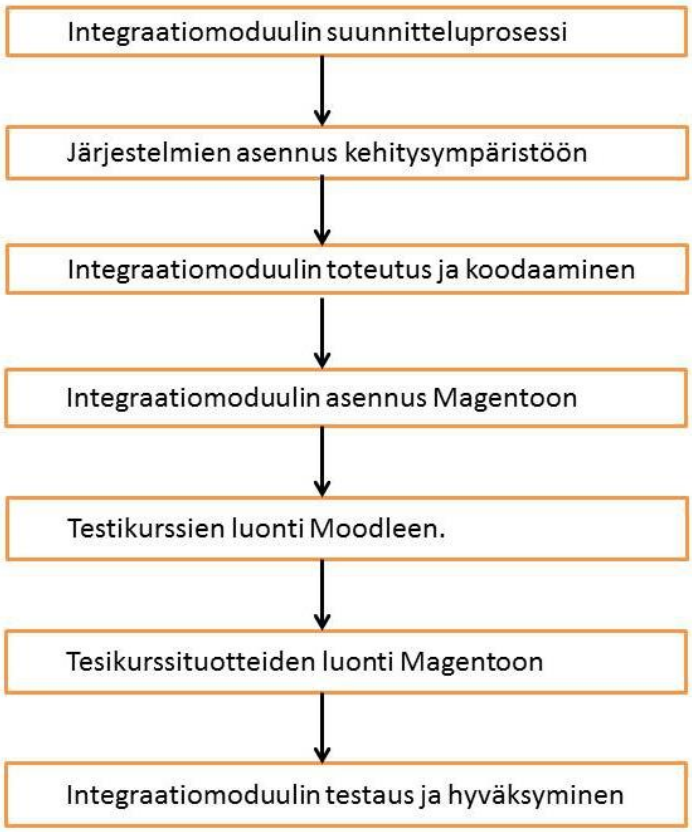
Kuva 5. Magento-Moodle -integraatiomoduulin toimintaprosessi verkkokauppa-asiakkaan näkökulmasta.

Aluksi asiakas rekisteröityy tai kirjautuu olemassa olevilla tunnuksilla verkkokauppaan ja ostaa tuotteen Magentosta (vaihe 1). Tämän jälkeen asiakas siirtyy maksamaan tilauksen verkkopankkiin Magentoon asennetun maksumoduulin kautta (vaihe 2). Onnistuneen maksutapahtuman jälkeen asiakkaan sähköpostiosoitteeseen lähetään tilausvahvistus ja erillinen sähköpostiviesti, joka sisältää hänelle integraatiomoduulin kautta luodun tunnuksen ja salasanan Moodle-järjestelmään (vaihe 3). Tämän jälkeen Magento välitää integraatiomoduulin kautta Moodleen tiedon ostetusta kurssista (vaihe 4).

Mikäli asiakkaalla on jo tunnus Moodlessa, asiakas lisätään tällöin osallistujaksi ostamalleen kurssille (vaihe 5a). Jos taas asiakkaalla ei vielä ole tunnusta Moodlessa, luodaan hänelle käyttäjätunnus Moodleen ja tämä tunnus lisätään osallistujaksi ostamalleen kurssille (vaihe 5b). Asiakas saa tunnuksellaan oikeuden osallistua ostamalleen kurssille ajan X. Kurssille osallistuvien käyttäjien oletusvoimassaoloaika määritetään Moodlen kurssinhallinnan kautta (vaihe 6). Kun käyttöoikeus kurssille päättyy, asiakas pääsee edelleen kirjautumaan Moodleen integraatiomoduulin kautta luodulla tunnuksella, muttei pääse enää näkemään tai osallistumaan aikaisemmin ostamalleen kurssille, koska osallistumisen voimassaoloaika on umpeutunut (vaihe 7).

6.2 Moduulin suunnittelu ja toteutusvaiheet

Koska ennen projektin alkua ei ollut aikaisempaa kokemusta Magenton moduulinkehityksestä, integraatiomoduulin suunnitteluprosessi aloitettiin tutkimalla ensin yleisesti Magenton moduulinkehitystä ja siihen liittyviä vaiheita ja toimenpiteitä. Alla olevassa kuvassa on eritelty tarkemmin projektiin kuuluneet vaiheet.



Kuva 6. Integraatiomoduulin suunnittelu ja toteutusprosessin vaiheet

Suunnitteluprosessin aikana käytiin läpi useita Magento-yhteisön oppaita, digitaalisia e-kirjoja ja moduulinkehitystä käsitteleviä sivustoja. Kun Magenton moduulinkehityksestä oli saatu riittävät tiedot ja valmiudet integraatiomoduulin kehittämiseen, lähdettiin seuraavaksi asentamaan testiympäristöä integraatiomoduulin toteutusta ja testausta varten projektia varten käytössä olleelle kehityspalvelimelle. Integraatiomoduulin toteutusta varten asennettiin toimeksiantajan kehityspalvelimelle molemmista järjestelmistä viimeisimmät työn tekohetkellä saatavilla olleet versiot.

Integraatiomoduulin toteutus, koodaaminen ja testaus veivät luonnollisesti isoimman osan projektiin käytetystä ajasta. Toteutusprosessin aikana integraatiomoduuli jouduttiin muutamaan otteeseen asentamaan järjestelmään uudelleen. Onnistuneen asennuksen jälkeen luotiin ensin Moodle-järjestelmään integrointia varten testikurssit. Näiden kurssien välille luotiin tämän jälkeen vastaavasti Magentoon luotuihin kurssituotteisiin integraatiomoduulin Moodle courses -attribuutin (josta tarkemmin luvussa 6.5.1) avulla.

6.3 Moduulin hakemistorakenne ja tiedostot

Tässä luvussa käydään tarkemmin läpi integraatiomoduulin hakemisto- ja tiedostorakennetta ja tiedostojen sisällä olevia funktioita ja niiden toiminnallisuuksia.

6.3.1 controllers-hakemisto

Integraatiomoduulin controllers-hakemisto sisältää vain yhden tiedoston, IndexController.php. Tämä tiedosto sisältää integraatiomoduulin kehittäjän kirjoittamia funktiota, joiden tarkoiksensa on testata integraation sekä luvussa 4.3 käsiteltyjen Moodlen Web service -funktioiden toimivuutta Magento-testiasennuksessa. Funktioiden testaus tapahtui käytännössä menemällä selaimen kautta verkko-osoitteeseen, joka muodostuu moduulin nimestä ja ajettavan funktion nimestä ilman Action loppuosaa, esim. http://www.esimerkki.fi/moodlecourses/addnewuser.

Mainitussa esimerkkifunktiossa (Liitteessä 1) testataan uuden käyttäjän luomista Moodle-järjestelmmään Magento-testiasennuksen kautta hyödyntämällä Moodlen core_user_create_users Web service -funktiota. Funktion alussa määritellään, mitä Moodlen Web service -funktiota call_moodle-apumetodin kautta tullaan kutsumaan. Muuttujan $customer kautta Magenton sisäisen metodin getSingleton() kautta haetaan funktion testausta varten Magentoon luodun asiakaskäyttäjätunnuksen tiedot tunnuksen ollessa kirjautuneena sisään Magento-testiasennukseen. Muuttujassa $helper puolestaan alustetaan funktion käyttöön helper-olio, jonka avulla voidaan kutsua integraatiomoduulin Helper-kansiossa (kuvattu tarkemmin luvussa 6.3.3) olevan Data.php tiedostossa sijaitsevia apufunktiota.

Funktion ensimmäinen if-lause tarkistaa, onko käyttäjä kirjautunut Magento-verkkokauppaan. Mikäli näin on, funktio hakee kirjautuneen asiakaskäyttäjätunnuksen tiedot (kokonimi, etunimi, sukunimi, sähköpostiosoite). Saadut arvot siirretään tämän jälkeen taulukkomuodossa muuttujaan $new_user. Tämän muuttujan sisältämä tieto siirretään muuttujaan $params, joka toimii call_moodle-funktioon annettavana toisena parametrina.

Lopuksi funktio tarkistaa, löytyykö Moodlesta jo funktion kautta lisättävää käyttäjätunnusta. Mikäli vastaavalla tunnuksella/sähköpostiosoitteella löytyy jo käyttäjätunnus, tulostetaan ruudulle viesti, joka ilmoittaa, että käyttäjä on jo olemassa. Jos taas vastaavaa tunnusta ei vielä löydy järjestelmästä, luodaan uusi käyttäjätunnus kutsumalla call_moodle-apufunktion avulla core_user_creater_users-funktiota ja sille parametrina funktion kautta Magentoon kirjautuneen käyttäjätunnuksen tiedot.

6.3.2 etc-hakemisto

Integraatiomoduulin etc-hakemisto sisältää yhden tiedoston, config.xml. Tämä on moduulin kokoonpanotiedosto, josta selviää moduulin nimi ja versionumero. Tämän lisäksi config.xml-kokoonpanotiedosto kertoo moduulille kaikki sen tarvitsemat riippuvuudet, mitä Magenton malleja siinä hyödynnetään sekä minkä tapahtumien yhteydessä integraatioon liittyvän tapahtumakuuntelijan funktioita kutsutaan.

6.3.3 Helper-hakemisto

Integraatiomoduulin Helper-hakemisto sisältää kaksi tiedostoa: Data.php ja curl.php. Data.php tiedosto sisältää kaksi apufunktiota, call_moodle ja generatePassword. Call_moodle -funktio toimii REST-protokollaa hyödyntäen siltana integroitujen Magento ja Moodle -järjestelmien välillä. Integraatiomoduulissa tätä funktiota käytetään aina, kun Magenton kautta kutsutaan mitä tahansa Moodlen web service -funktioista. Ilman sitä integraatio ei käytännössä toimisi lainkaan. GeneratePassword -funktion tehtävänä on generoida integraatiomoduulin kautta Moodleen-luotavalle käyttäjätunnnukselle Moodlen salasanan vähimmäisvaatimukset (8 merkkiä pitkä, sisältäen vähintään yhden numeron, ison kirjaimen ja erikoismerkin) täyttävä salasana.

Curl.php puolestaan on call_moodle-funktion kautta lähtevien kutsujen lähetystä varten oleva tiedosto. Tiedoston sisältämät metodit hyödyntävät ilmaiseksi saatavilla olevaa libcurl-kirjastoa, jonka avulla on mahdollista siirtää helposti tietoa HTTP-protokollan kautta. Ilman tämän tiedoston sisältämiä metodeja Data.php-tiedoston kautta tehdyt komennot eivät välity Moodle-järjestelmään.

6.3.4 Model-hakemisto

Integraatiomoduulin Model-hakemisto sisältää tiedoston Observer.php sekä sen sisällä olevan hakemistopolun Product\Attribute\Source alla sijaitsevan Unit.php-tiedoston. Nämä tiedostot sisältävät toiminnallisuudet, joiden avulla integraatiomoduuli saa tarvitsemansa tiedot, kuten kurssituotteen ostavan asiakaan käyttäjätunnuksen ja nimen siirrettäväksi Magentosta Moodle -järjestelmään, sekä lisätäkseen Moodlen kurssitiedot omaksi tuoteattribuutikseen (Moodle courses) Magentoon.

6.3.5 sql-hakemisto

Integraatiomoduulin sql-hakemisto sisältää alihakemistossa moodlecourses_setup olevan asennustiedoston install-0.1.0.php sekä asennuksen päivitystiedoston upgrade-0.1.0-0.1.1.php. Asennustiedoston avulla integraatiomoduuli luo Magenton tietokantaan integroitavaa kurssituotettava varten uuden tuoteattribuutin, moodle_courses.

Päivitystiedosto puolestaan asentaa Magenton tietokantaan tuoteattribuuttia varten uuden kentän tilaus-taulukoon, jonne uusi tuoteattribuutti (kurssitunnus) tallennetaan, kun asiakas tekee tilauksen Magentossa. Tämän uuden kentän kautta integraatiomoduuli myös hakee tiedon siitä, mille kurssille uusi tai jo olemassa oleva käyttäjä lisätään.

6.3.6 etc/modules-hakemisto

Etc/modules-hakemistosta löytyy moduulin Magenton moduulirekisteriin listaava Tawasta_Moodlecourses.xml tiedosto. Moduulissa määritetään moduulin tila (active = true) ja missä polussa/koodipoolissa moduuli sijaitsee (codePool = local).

Esimerkki 5. Tawasta_Moodlecourses.xml

```
<config>
    <modules>
        <Tawasta_Moodlecourses>
            <active>true</active>
            <codePool>local</codePool>
        </Tawasta_Moodlecourses>
    </modules>
</config>
```

6.4 Moduulin globaalien asetusten määrittäminen

Integraatiomoduulin globaalien asetusten määrittäminen tapahtuu moduulin etc/modules-kansion config.xml tiedoston kautta. Moduulin globaaleihin määrityksiin lukeutuvat muun muassa moduulin malli-, apu- ja tapahtumankäsittelijä-luokkien nimeäminen sekä moduulin asennukseen liittyvät määritykset ja resurssit. Moduulin globaaleissa määrityksissä kerrotaan myös, minkä verkkokaupan tapahtumien yhteydessä integraatiomoduulin toimintoihin liittyviä funktioita kutsutaan. Alla on esimerkki tapahtuman kutsumisesta:

Esimerkki 6. etc/modules/config.xml

```
<sales_order_place_after>
  <observers>
    <tawasta_moodlecourses>
      <class>tawasta_moodlecourses/observer</class>
      <method>addToCourse</method>
      <type>singleton</type>
    </tawasta_moodlecourses>
  </observers>
</sales_order_place_after>
```

Esimerkissä ensimmäiseksi moduulissa kerrotaan Magentolle tapahtuman nimi (sales_order_place_after), joka yllä olevan esimerkin tapauksessa suoritetaan ostotapahtuman jälkeen. Seuraavaksi moduulille määritetään tapahtumankäsittelijän luokka (class), suoritettavan metodin nimi (addToCourse), joka sijaitsee Model-hakemiston Observer.php -tiedostossa sekä suoritettavan tapahtuman tyyppi.

Eräs integraation toiminnan kannalta oleellinen moduulin globaaleihin määrityksiin kuuluva osa on tilaustietokantaan kirjoitettavien viitekenttien nimeäminen. Tämä on tärkeää siksi, koska ilman tätä tilaukseen tallennettavaa tietoa integraatiomoduuli ei pysty välittämään ostetun kurssin tunnistetta Moodleen.

6.5 Moduulin toiminnallisuudet

Tässä luvussa kuvataan tekniseltä toteutukseltaan tarkemmin integraatiomoduulin keskeisimmät toiminnallisuudet, eli Moodle-kurssi tuoteattribuutin luominen, Moodlen Web service -funktioiden kutsuminen sekä käyttäjätunnuksen luominen ja käyttäjän lisääminen Moodle-kurssille. Integraatiomoduulin yleinen toimintaperiaate sekä tässä luvussa tarkemmin kuvattuihin toiminnallisuusiin liittyvä yleinen toimintaprosessi on kuvattu jo aiemmin luvussa 6.1.

6.5.1 Moodle-kurssi tuoteattribuuttien luominen

Kun integraatiomoduuli asennetaan ja otetaan ensimmäistä kertaa käyttöön Magentossa, integraatiomoduulin kautta luodaan uusi tuoteattribuutti, moodle_courses. Tämä tuoteattribuutti hyödyntää aikaisemmin luvussa 6.3.4 mainitun Unit.php-tiedoston kautta kutsuttua Moodlen Web service -funktiota core_course_get_courses, jonka kautta se luo tuoteattribuutille taulukkolistan Moodlessa olevista kursseista. Tämän jälkeen haluttu kurssi on mahdollista valita kurssituotteelle Magentosta tuotteen hallintasivun kautta löytyvästä pudotusvalikosta.

Esimerkki 7. Moodle courses -tuoteattribuutin luominen (sql/moodlecouses_setup/install-0.1.0.php)

```
$installer = $this;
$installer->startSetup();

$installer->addAttribute('catalog_product', 'moo-
dle_courses', array(
'group'             => 'General',
'type'              => 'int',
'backend'           => '',
'frontend'          => '',
'label'             => 'Moodle Courses',
'input'             => 'select',
'class'             => '',
'source'            =>
'tawasta_moodlecourses/product_attribute_source_unit',
'global'            =>
Mage_Catalog_Model_Resource_Eav_Attribute::SCOPE_STORE,
    'visible'           => true,
    'required'          => false,
    'user_defined'      => false,
    'searchable'        => false,
    'filterable'        => false,
    'comparable'        => false,
    'visible_on_front'  => false,
    'unique'            => false,
    'is_configurable'   => false
));

$installer->endSetup();
```

Uutta attribuuttia luodessa asennustiedostolle kerrotaan addAttribute -funktion kautta ensimmäisenä parametrina minkä tyyppinen luotava attribuutti on (catalog/product). Toiseen parametriin määritetään uuden attribuutin yksilöivä tunniste/nimi (moodle_courses). Funktion viimeinen parametri on taulukkomuotoisena tuoteattribuutille annettavat muut määritykset, kuten ryhmä, johon attribuutti lisätään, tallennettavan tiedon tyyppi ja syöttötapa, attribuutin otsikko sekä miten laajalti kyseinen attribuutti on käytössä Magento-asennuksen sisällä.

Edellä mainittujen valintojen lisäksi voidaan määrittää boolean-arvoilla muun muassa onko attribuutin arvon oltava uniikki tai pakollinen, määrittää tuotteelle, onko se verrattavissa tai käytettävissä Magenton tuotehaun kautta tai näytetäänkö sitä tuotteen yhteydessä sivuston julkisessa näkymässä.

Integraatiomoduuli kannalta addAttribute -funktion kolmannen parametrin tärkein määritys on kuitenkin lähde (source), jonka kautta kaikki tuoteattribuutin kautta luodun alasvetovalikon sisältämät kurssit haetaan. Source-määrityksellä funktiolle käytännössä kerrotaan tiedoston sijainti, jonka kautta Moodlessa olevat kurssit haetaan Magento-järjestelmään.

6.5.2 Moodlen Web service -funktioiden kutsuminen

Kuten luvussa 6.3.3. mainittiin, tapahtuu Moodlen Web service -funktioiden kutsuminen hyödyntämällä integraatiomoduulin call_moolde -funktiota. Tämä Helper-kansion Data.php-tiedostossa sijaitsevassa funktiossa määritellään integroitavan Moodle-asennuksen domainin sijaintiosoite ja Web service -käyttäjän tunnistusta varten Moodlen puolella aikaisemmin luotu salausavain. Funktioon annettavat parametrit ovat Web service -funktion nimi ja kyseiseen funktioon mahdollisesti liittyvät parametrit.

Esimerkki 8.Data.php

```
public function call_moodle($function_name, $params) {

 $token = 'nb54t551t2Y1j650lD18y172u0r1C508';
 $domain = 'http://example.moodle.dev';

 header('Content-Type: text/plain');
 $serverurl = $domain . '/webservice/rest/server.php'.
'?wstoken=' . $token . '&wsfunction='.$function_name;

 require_once('curl.php');
 $curl = new curl;

$restformat = 'json';
$restformat = ($restformat ==
'json')?'&moodlewsrestformat=' . $restformat:'';

 $resp = $curl->post($serverurl . $restformat, $params);
 $json_output = json_decode($resp, true);
 return $json_output;
 }
}
```

6.5.3 Käyttäjätunnuksen luominen ja osallistujaksi lisääminen kurssille

Kurssituotteen ostavan verkkokauppa-asiakaan Moodle-käyttäjätunnuksen luominen ja osallistujaksi lisääminen ostetulle kurssille tapahtuu integraatiomoduulin Model-hakemistossa sijaitsevan Observer.php-tiedoston addToCourse-funktion kautta. Jotta käyttäjätunnuksen luominen onnistuisi, tulee käyttäjän ensin luoda käyttäjätili Magentoon. Tämä on välttämätöntä, koska integraatiomoduuli hakee käyttäjään liittyvät tiedot, kuten käyttäjätunnuksena toimivan sähköpostiosoitteen Magenton käyttäjätietokannasta. Ostoprosessi on kuvattu tarkemmin luvussa 6.1.

AddToCourse -funktiota kutsutaan onnistuneen maksutapahtuman jälkeen. Funktio hakee kutsumisen jälkeen ensimmäiseksi tehdyn tilauksen tiedot (getOrder). Tilauksen tiedoista oleellisimmat ovat käyttäjän sähköpostiosoite sekä etu- ja sukunimi.

7 YHTEENVETO

Tässä luvussa käydään läpi opinnäytetyön tulokset ja saavutetut edut. Lisäksi käydään läpi opinnäytetyön aikana koettuja haasteita ja tulevaisuudessa integraatiomoduuliin mahdollisesti toteutettavia jatkokehitysideoita.

7.1 Tulokset ja saavutetut edut

Ottaen huomioon että projektin alussa sen toteuttajalla ei ollut aikaisempaa kokemusta Magenton moduulinkehityksestä tai Moodlen Web service -rajapinnoista, projektin tuloksena syntynyt integraatiomoduuli saatiin valmiiksi jopa ennakkoon odotettua nopeammin. Työn tuloksena syntyneeseen valmiiseen integraatiomoduuliin saatiin myös implementoitua kaikki siihen ennen projektin alkua suunnitellut toiminnallisuudet.

Integraatiomoduuli on jo myös otettu tuotantokäyttöön. ja sitä tullaan mitä todennäköisimmin hyödyntämään myös muissa vastaavankaltaisissa toteutuksissa.

7.2 Haasteet

Kuten työssä jo aikaisemmin on todettu, projekti koettiin alkuun hyvin haasteelliseksi, koska sen toteuttajalla ei ennen projektin alkua ollut aikaisempaa kokemusta Magenton moduulinkehityksestä. Tämän lisäksi ei myöskään ollut saatavilla aikaisempaa tietoa tai toteutusmalleja siitä, kuinka Magenton ja Moodlen välistä suoraa integraatiota olisi aikaisemmin käytännössä toteutettu.

Magenton moduulinkehitykseen sekä Moodlen REST-protokollan kautta hyödynnettävissä olevien Web service -funktioihin tutustumisen aikana työn tekijälle kuitenkin selkeni, että järjestelmien välinen integraatio ja siihen liittyvät perustoiminnallisuudet ja projektin kautta määritetyt ominaisuudet olisi loppujen lopuksi mahdollista toteuttaa melko vähällä vaivalla näitä tekniikoita ja Magenton moduulinkehitykseen perehtymisen yhteydessä opittuja periaatteita hyödyntämällä.

7.3 Jatkokehitys

Yhtenä tulevaisuuden haasteena projektin tuloksena syntyneen integraatiomoduulin jatkokehityksen kannalta on tilanne, jossa esimerkiksi opettaja ostaisi useammalle oppilaalle tunnukset tietylle Moodle-kurssille. Tämä ei tällä hetkellä kuitenkaan ole vielä mahdollista Moodlen Web service -funktioiden asettamien rajoitteiden takia. Tämän lisäksi yksi haasteellinen toiminto, jota mahdollisesti integraatiomoduuliin voitaisiin tulevaisuudessa tarvita, olisi palvelujen välisen kertakirjautumismenetelmän (Single sign-on; SSO) toteuttaminen.

Edellä mainittujen lisäksi integraatiomoduulin kehityksessä tullaan tulevaisuudessa luonnollisesti ottamaan huomioon myös Magenton versiopäivitysten yhteydessä järjestelmän ydinprosesseihin, rakenteeseen tai toi-

minnallisuuksiin tapahtuvien muutosten vaikutukset integraatiomoduulin toimintaan.

LÄHTEET

About Moodle – History 2014. Viitattu 29.6.2014.
http://docs.moodle.org/27/en/History

Moodlen tilastot 2014. Viitattu 29.6.2014.
https://moodle.org/stats/

Registered moodle sites 2014. Viitattu 29.6.2014.
https://moodle.org/sites/

Web Services Architecture 2004. W3C Working Group -ryhmittymä. Viitattu 1.11.2014.
http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/

MacGregor, Allan. Magento PHP Developer's Guide. Packt Publishing. 2013.

Verkkokauppatilasto 2013. Perustietoa verkkokauppaseurannasta sekä ensimmäinen vuosipuolisko. 2013. TNS Gallup. Viitattu 3.11.2014
http://www.tns-gallup.fi/doc/uutiset/Verkkokauppatilasto_2013_H1.pdf

XML (Extensible Markup Language) 2007. Margaret Rouse. Viitattu 5.11.2014.
http://searchsoa.techtarget.com/definition/XML

PHP (Hypertext Preprocessor) 2006. Margaret Rouse. Viitattu 5.11.2014.
http://searchenterpriselinux.techtarget.com/definition/PHP

Installing custom attributes with your module 2011. Viitattu 18.7.2014.
http://www.magentocommerce.com/wiki/5_-_modules_and_development/0_-_module_development_in_magento/installing_custom_attributes_with_your_module

Liite 1

ADDNEWUSERACTION -FUNKTIO

```
public function addnewuserAction() {
$add_moodle_user = 'core_user_create_users';
$get_moodle_users = 'core_user_get_users';
$customer = Mage::getSingleton('customer/session')-
>getCustomer();
$helper = Mage::helper('tawasta_moodlecourses');

if (Mage::getSingleton('customer/session')->isLoggedIn()) {
  //Get the customer data
  $customer = Mage::getSingleton('customer/session')-
>getCustomer();
    $fullname = $customer->getName();
    $firstname = $customer->getFirstname();
    $lastname = $customer->getLastname();
    $customerEmail = $customer->getEmail();
}

$preferences   =   array('type'=>'auth_forcepasswordchange',
'value'=>1);

$new_user = array(
  'username'          => $customer->getEmail(),
  'password'          => '',
  'firstname'         => $customer->getFirstname(),
  'lastname'          => $customer->getLastname(),
  'email'             => $customer->getEmail(),
  'preferences'       => array($preferences)
);

$params = array('users' => array($new_user));

$search = array('key'=>'email', 'value'=> $customer-
>getEmail());
$params = array('criteria'=>array($search));
$response = $helper->call_moodle($get_moodle_users,
$params);
$email_moodle = $response['users'][0]['email'];

if ($customerEmail != $email_moodle) {
  Mage::helper('tawasta_moodlecourses')-
>call_moodle($add_moodle_user, $params);
    echo 'New user added to Moodle';
  } else {
    echo 'Customer already exists';
  }
}
```