SAVONIA-AMMATTIKORKEAKOULU
LIIKETALOUS, KUOPIO

B2B-TILAUSJÄRJESTELMÄN RAJAPINTASOVELLUKSEN OHJELMOINTI BEST FRIEND GROUP OY: LLE

Tommi Poikolainen

Tradenomin opinnäytetyö

Tietojenkäsittelyn koulutusohjelma

Marraskuu 2010

SAVONIA-AMMATTIKORKEAKOULU
LIIKETALOUS, KUOPIO
Koulutusohjelma, suuntautumisvaihtoehto (jos on)

Tietojenkäsittelyn koulutusohjelma

Tekijä(t)

Tommi Poikolainen

Työn nimi

B2B-Tilausjärjestelmän rajapintasovelluksen ohjelmointi Best Friend Group Oy:lle

Työn laji	Päiväys	Sivumäärä
Opinnäytetyö	10.8.2010	53 + 24

Työn ohjaaja(t)	Toimeksiantaja
Jyrki Linja	Best Friend Group Oy

Tiivistelmä

Opinnäytetyön tarkoituksena oli toteuttaa Best Friend Group Oy:lle rajapintasovellus, jonka avulla mahdollistettiin integraatio ja automaatio yrityksen käytössä olevan resurssienhallintajärjestelmän ja Magento- verkkokauppasovelluksen välillä.

Työssä kuvataan valmiin järjestelmän merkitystä yritykselle, rajapintasovelluksen toteutusta ja rakennetta sekä toteutukseen käytettyjä välineitä. Lisäksi esitellään järjestelmän arkkitehtuuria kokonaisuudessaan sekä rajapintasovelluksen sisäistä arkkitehtuuria.

Rajapintasovelluksen toteutus tapahtui pääasiallisesti käyttämällä PHP-ohjelmointikieltä ja ohjelmoinnissa pyrittiin soveltamaan mahdollisimman kattavasti olio-ohjelmoinnin periaatteita. Työssä keskitytään esittelemään pääosin PHP-ohjelmointia ja siihen sisältyviä tekniikoita, mutta työssä esitellään myös vähemmän käytettyjä, mutta vaadittuja menetelmiä ja niiden käyttöä. Näitä menetelmiä ovat XML- ja HTML-kielet sekä tietokantakäsittelyyn liittyvät sovellukset ja järjestelmät, kuten Oracle- tietokanta ja MySql-tietokannanhallintaohjelma.

Työn tuloksena on korkeatasoista automatiikkaa sisältävä tilausjärjestelmä, joka integroi kahden järjestelmän tiedot ylläpitäen niitä automaattisesti, mutta ei kuitenkaan sisällä tilausjärjestelmän ulkoasuun liittyvää toteutusta.

Asiasanat

Rajapintasovellus, olio-ohjelmointi, PHP, Magento

Huomioitavaa

SAVONIA UNIVERSITY OF APPLIED SCIENCES
UNIT OF BUSINESS AND ADMINISTRATION, KUOPIO
Degree Programme, option

Degree Programme in Computer Science

Author(s)

Tommi Poikolainen

Title of study

Programming a B2B-order system's interface application for Best Friend Group Ltd

Type of project	Date	Pages
Thesis	10.8.2010	53 + 24

Supervisor(s) of study	Executive organisation
Jyrki Linja	Best Friend Group Ltd.

Abstract

The purpose of the final project was to create an interface application for Best Friend Group Ltd. to make integration and automatization possible between the company's resource management system and Magento e-commerce platform.

The thesis contains a description of the value of the complete system for the company, the implementation and structure of the interface application as well as the instruments used in implementation. In addition, the system architecture and the inner architecture of the interface solution are presented.

The execution of the interface application was primarily completed by using PHP-programming language with the intention to use the principles of object-oriented programming as extensively as possible. In this work, focus is on presenting PHP-programming and the techniques it includes, without disregarding other less used but yet required methods. These methods include XML- and HTML-languages as well as applications and systems used for database management, such as Oracle-database and MySql-database management system.

The project resulted in an order system, which contains high-quality automatization and integrates information between two systems with automatic update feature. Less attention was paid to the appearance intentionally.

Keywords

Interface solution, object-oriented programming, PHP, Magento

Note

LYHENTEET JA ERIKOISMERKIT

- PHP: Hypertext Preprocessor – ohjelmointikieli
- Axapta: Microsoft Axapta – toiminnanohjausjärjestelmä
- Magento: avoimeen lähdekoodiin perustuva verkkokauppasovellus
- Oracle: Oracle Corporationin kehittämä relaatiotietokanta
- ... : Tarkoittaa lyhennettyä koodia tilanteessa, jossa on esitetty kokonainen metodi, mutta haluttu lyhentää koodia piilottamalla tilanteen kannalta merkityksetöntä koodia.
- Ylikirjoitus: Ylikirjoittamisella tarkoitetaan tämän opinnäytetyön yhteydessä metodia tai moduulia, joka kumoaa toisen metodin tai moduulin.
- Räätälöinti: Räätälöinnillä tarkoitetaan tämän opinnäytetyön yhteydessä itse ohjelmoimalla luotua luokkaa, metodia tai moduulia. Räätälöinti tulee englanninkielisestä sanasta ”custom”, jota käytetään paljon ohjelmoinnin yhteydessä.

SISÄLLYS

1 JOHDANTO 7

2 BEST FRIEND GROUP OY 8

2.1 Tarve tilausjärjestelmälle 8

3 SÄHKÖINEN KAUPANKÄYNTI 9

3.1 Verkkokauppa 9

3.2 Verkkokaupan edut kaupankäynnille 9

4 RAJAPINTASOVELLUKSEN LÄHTÖKOHTA 11

5 KÄYTETYT TEKNIIKAT JA JÄRJESTELMÄT 13

5.1 PHP 13

5.2 Magento 13

5.3 HTML 14

5.4 XML 14

5.5 Oracle ja MySql 14

6 INTEGRAATION TOTEUTUS RAJAPINTASOVELLUKSEN KAUTTA 16

6.1 Integraation rakenne 16

6.2 Yhteydet 17

6.3 Tietojen haku Axaptasta 18

6.4 Tietojen haku Magentosta 20

6.5 Tietojen vertaaminen 21

6.6 Tietojen välittäminen Magentolle 23

7 RAJAPINTASOVELLUKSEN TOTEUTUS 26

7.1 Sovelluksen rakenne 26

7.2 Standardisointi 27

7.3 Tietokantakyselyt 28

7.4 Tiedonvälitys luokille 30

7.5 Luokkien ominaisuudet ja niiden käsittely 32

7.5.1 Ominaisuudet 33

7.5.2 Ominaisuuksien arvojen käsitteleminen 35

7.6 Virheiden ja poikkeustilanteiden hallinta 37

7.6.1 Poikkeustilanteiden hallinta 38

7.6.2 Virheiden hallinta 40

7.7 Aliluokat 41

8 MAGENTON TOIMINNALLISUUDEN LAAJENTAMINEN MODUULEILLA 42
8.1 Moduulin asettaminen Magenton käytettäväksi 43
8.2 Moduulin konfigurointi 43
8.3 Magenton ohjelmointirajapinnan laajentaminen 44
8.4 Magenton sisällönhallintamoduulien laajentaminen 47
9 POHDINTA 50
LIITE 1 Rajapintasovelluksen tuotteet päivittävä luokka 54
LIITE 2 Tuotteita mallintava luokka 64
LIITE 3 Tuoteryhmänä myytäviä tuotteita mallintava luokka 69
LIITE 4 Abstrakti luokka 73
LIITE 5 Rajapinnat 75
LIITE 6 Asiakasryhmiä käsittelevälle moduulille määritetty luokka 76

1 JOHDANTO

Tässä opinnäytetyössä on toteutettu rajapintasovellus, jonka avulla saadaan integroitua ja automatisoitua Best Friend Group Oy:n toiminnanohjausjärjestelmä ja tilausjärjestelmä. Rajapintasovelluksen ohjelmoinnissa on käytetty PHP-ohjelmointikieltä ja painopiste ohjelmoinnissa on pyritty pitämään proseduraalisen ohjelmoinnin sijaan olio-ohjelmoinnissa.

Koska opinnäytetyön ohjelmoinnillinen toteutus on pyritty pääasiallisesti toteuttamaan hyödyntämällä olio-ohjelmoinnin periaatteita, ajattelutapoja ja menetelmiä kuitenkaan näitä esittelemättä, on tämän opinnäytetyön lukijalta oletettavaa tuntea olio-ohjelmointin liittyvät käsitteet entuudestaan.

Toteutettu rajapintasovellus oli hyvin laaja ja sisälsi 29 eri tilausjärjestelmän kohdetta mallintavaa luokkaa sekä 13 tilausjärjestelmän ohjelmointirajapintaa- ja toiminnallisuutta laajentavaa moduulia. Johtuen rajapintasovelluksen laajuudesta, on tämä opinnäytetyö rajattu esittelemään tilausjärjestelmän keskeisimpiä kohteita mallintavia luokkia, jotka ovat asiakas- ja tuotetietoja käsittelevät luokat. Myös moduulien esittely on samasta syystä rajattu sisältämään vain yksittäisen ohjelmointirajapintaa laajentavan moduulin sekä yksittäisen tilausjärjestelmän toiminnallisuutta laajentavan moduulin.

Työssä puhutaan integraatiosta Microsoft Axapta-toiminnanohjausjärjestelmän kanssa. Toiminnanohjausjärjestelmä sisältää paljon monenlaisia toiminnallisuuksia ja komponentteja, joista kuitenkin vain kyseisen järjestelmän tietokanta on tämän opinnäytetyön osalta relevantti. Tästä johtuen muu Microsoft Axapta-järjestelmän esitteleminen on rajattu pois ja Axaptasta puhutaan vain kokonaisuutena, joka sisältää tietokannan.

Tilausjärjestelmän toteutus vaati myös sisällönhallintaan liittyviä toimenpiteitä, joilla muokattiin ulkoasun rakennetta sekä järjestelmän visuaalista toiminnallisuutta. Tämä osuus työstä on kuitenkin laajuutensa takia jätetty käsittelemättä tässä opinnäytetyössä.

2 BEST FRIEND GROUP OY

Best Friend Group Oy on Kuopiossa toimiva yritys, jonka liiketoiminta keskittyy lemmikkieläintarvikkeiden kehittämiseen, markkinointiin ja jakeluun. Yrityksen kaksi keskeisintä jakelukanavaa ovat suuret päivittäistavaraketjut sekä erikoistuneet lemmikkieläinkaupat ja – ketjut.

Suomessa, Ruotsissa, Norjassa sekä Tanskassa toimiva Best Friend Group Oy on lemmikkitarvikkeiden selkeä markkinajohtaja Pohjoismaissa.

2.1 Tarve tilausjärjestelmälle

Tilausjärjestelmän perimmäisenä tavoitteena toimeksiantajan kannalta on tarjota asiakkaille tilausjärjestelmä, joka tarjoaa heille enemmän kuin kilpailijoiden verkkokaupat.

Toteutuessaan tilausjärjestelmä tarjoaa Best Friend Group Oy:lle kilpailuetua erikoisliikekanavassa, jossa kattavat tilausjärjestelmäratkaisut ovat suhteellisen harvinaisia.

Palvellakseen Best Friend Group Oy:n toimintaa, tulee tilausjärjestelmän sisältää korkeatasoista automaatiota ja integraatiota Best Friend Group Oy:n toiminnanohjausjärjestelmän sekä tilausjärjestelmän välillä, sillä yrityksen tuotekatalogi on liian suuri pidettäväksi manuaalisesti yllä kahdessa eri järjestelmässä.

3 SÄHKÖINEN KAUPANKÄYNTI

Sähköisellä kaupankäynnillä tarkoitetaan tietokoneverkkojen välityksellä suoritettavaa tavaroiden ja palvelujen tilaustoimintaa. Sähköinen kaupankäynti jaetaan kahteen ryhmään, joista toinen on verkkokauppaa, jossa ostajana on ihminen ja toinen organisaatioiden välistä tiedonsiirtoa hyödyntävää kauppaa, eli EDI-menetelmään perustuvaa kauppaa (Hallavo ym. 2009).

3.1 Verkkokauppa

Verkkokaupalla kuvataan erityisesti kaupankäyntiä verkossa, missä ostaja on ihminen. Verkkokauppa voi olla sekä yritysten välistä (B2B), että kuluttajille suunnattua (B2C) kauppaa. Jos verkkokauppaa ajatellaan prosessimielessä, siirretään siinä aikaisemmin kauppiaan tekemästä työstä osa asiakkaan tehtäväksi, jolloin saadaan kauppiaalle koituvia kustannuksia vähennettyä. (Hallavo ym 2009.)

Verkkokaupan osuus Suomessa sähköisestä kaupankäynnistä oli vuonna 2008 19 mrd. €, mikä on 5,8 % kokonaisliikevaihdosta. Vastaavasti organisaatioiden väliseen tiedonsiirtoon perustuvan EDI-myynnin arvo oli samana vuonna 35 mrd. € (Tilastokeskus, 2009).

3.2 Verkkokaupan edut kaupankäynnille

Verkkokaupan ansioista yrityksen maantieteellisen sijainnin merkitys vähentyy ja asiakkailla on mahdollisuus helposti tilata yrityksen tuotteita mistä vain. Samalla verkkokauppaa harjoittavalle yritykselle avautuu edullisesti uusia markkinoita, mikä on suuri etu varsinkin pienille yrityksille, joilla ei ole aikaisemmin ollut varaa rakentaa kansainvälistä markkinointia.

Verkkokaupan avulla voidaan myös tarjota asiakkaille ympärivuorokautinen saatavuus, johon päästään käsiksi välittömästi verkkoyhteyksien sen salliessa. Ympärivuorokautinen saatavuus on tärkeä tekijä kansainvälisessä kaupassa, sillä se poistaa aikaeroista johtuvia vaikeuksia.

Asiakkaiden profilointi on verkkokauppojen avulla helppoa. Asiakkaan toiminnasta verkkokaupassa on helppo tallentaa tietoa, jota voidaan analysoida myöhemmin.

Tällaista tietoa voi olla esimerkiksi asiakkaan tuotehaussa käyttämät hakusanat, asiakkaan käyttämät tuotekategoriat, ostetut tuotteet, vertaillut tuotteet jne., joiden avulla voidaan päätellä asiakkaan mielenkiinnon kohteita ja haluttuja tuotteita.

Tiedonjakoa voidaan suorittaa välittömästi kaikille asiakkaille verkkokaupan välityksellä riippumatta asiakkaan sijainnista. Verkkokaupan avulla voidaan helposti tiedottaa esimerkiksi uutuustuotteista, ajankohtaisista tarjouksista sekä kampanjoista.

Suurimmalla osalla yrityksistä on melko vähän kosketusta asiakkaiden mielipiteisiin. Tämä koskee erityisesti yhtiöitä, jotka käyvät B2B-kauppaa, joissa yrityksellä ei ole suoraa yhteyttä kuluttajaan. Verkkokaupan avulla saadaan luotua asiakkaalle helppo ja nopea lähestymistapa yritystä kohtaan.

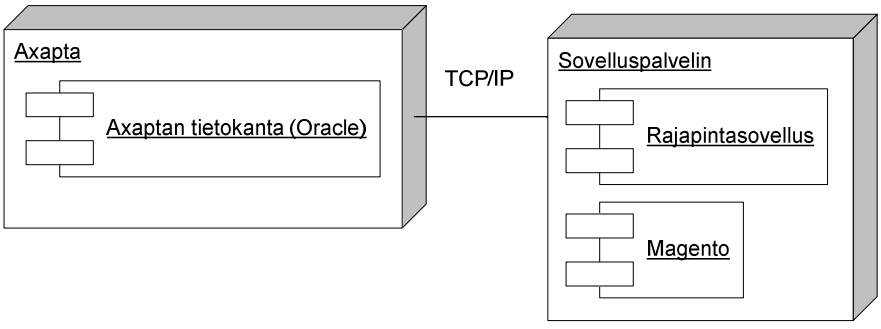
Verkkokaupan avulla saadaan myös tietoa digitalisoitua. Koska verkkokaupat käyttävät digitaalista tietoa, on sitä helppo varastoida ja käsitellä.(Timmers 2000; Rosen 1999.)

4 RAJAPINTASOVELLUKSEN LÄHTÖKOHTA

Rajapintasovellukselle lähtökohtana oli toteuttaa Best Friend Group Oy:n käytössä olevia järjestelmiä ja laitteistoja hyväksikäyttäen rajapintasovellus. Rajapintasovellus tuli toteuttaa integroimaan yhtiön Axapta-toiminnanohjausjärjestelmä (Microsoft Axapta) ja Magento verkkokauppasovellus. Toiminnanohjausjärjestelmä pysyisi koskemattomana integraatiossa lukuun ottamatta järjestelmän tietokantaa, joka toimisi käsiteltävien tietojen perimmäisenä lähteenä.

Axaptan tietokanta oli sijoitettu erilliselle tietokantapalvelimelle, johon yhteys oli saavutettavissa TCP/IP – verkkoprotokollien avulla (ks.kuvio 1). Rajapintasovelluksen toteuttamisen aikana sovellus sijoittui samalle palvelimelle Magento- verkkokauppasovelluksen kanssa. Rajapintasovellus tuli kuitenkin olla mahdollista siirtää eri palvelimille ilman sovelluksen muokkaamista.

Tilausjärjestelmä oli yhtiön kannalta määritetty toteutettavaksi Magento-verkkokauppasovelluksen avulla. Magento soveltui yrityksen käyttöön hyvin, sillä järjestelmä tarjoaa kattavan ohjelmointirajapinnan muihin järjestelmiin integrointia varten. Lisäksi Magento tarjosi valmiin käyttöliittymän mm. asiakkaille, asiakasryhmille, tuotteille, tuotekategorioille, hinnastoille, verotukselle, kielille ja eri valuutoille.



Kuvio 1. *Järjestelmän osien sijoittuminen järjestelmäkokonaisuuteen sekä fyysisten osien väliset yhteydet.*

Tilausjärjestelmän toiminnan tuli olla täysin automaattista Axaptan kanssa, lukuun ottamatta asiakastilien luomista, mikä kuitenkin tuli toteuttaa rajapintasovelluksen välityksellä. Lähtökohtana Magenton kannalta oli siis se, että Magenton alkuperäisen sisällönhallintajärjestelmän kautta ei pidetä mitään tietoja yllä, vaan kaikki

päivitykset, lisäykset ja poistot tapahtuisivat automaattisesti rajapintasovelluksen toimesta, Axaptan tietojen mukaisesti.

5 KÄYTETYT TEKNIIKAT JA JÄRJESTELMÄT

Seuraavaksi esitellään rajapintasovelluksessa käytettyjä järjestelmiä ja tekniikoita sekä niille ominaisia piirteitä.

5.1 PHP

PHP (Hypertext Preprocessor) on ohjelmointikieli, joka toimii palvelinympäristössä ja voidaan sisällyttää HTML-koodiin tai käyttää sellaisenaan. PHP- on tulkattava kieli, eli PHP-koodi suoritetaan joka kerta, kun koodia sisältävä sivu lähetetään palvelimelle. (Converse ym. 2004.)

PHP sisältää kaikki eri ohjelmointikielille tyypilliset rakenteet ja se on myös oliopohjainen ohjelmointikieli, jolloin sillä voidaan toteuttaa luokkiin perustuvia sovelluksia perinteisiin funktioihin perustuvien, eli proseduraalisen ohjelmoinnin periaatteiden mukaisesti toteutettujen sovellusten sijaan (Heinisuo, Rauta 2007).

5.2 Magento

Magento on open-source verkkokauppasovellus, joka tarjoaa käyttäjälleen joustavan järjestelmän ylläpitää sähköisen kaupan ulkoasua, sisältöä ja toiminnallisuutta (Varien 2009). Magenton avulla käyttäjä voi monipuolisesti ylläpitää erilaisia verkkokaupankäynnille olennaisia elementtejä.

Magento sisältää valmiin käyttöliittymän asiakastileille, joille voidaan määrittää mm. erilaiset hinnastot, verot, kuljetuskustannukset ja tuotevalikoimat. Magento sisältää myös tuen monikielisille verkkokaupoille sekä eri valuutoille, mikä tekee siitä helposti muokkautuvan kansainväliseksi verkkokaupaksi.

Moduuleista rakentuvana kokonaisuutena Magento tarjoaa ohjelmoijille laajat mahdollisuudet muokata ja kehittää turvallisesti järjestelmää. Magenton ylläpitoa ja kehitystä tukemaan on Magentoon luotu ohjelmointirajapinta, jonka kautta erinäiset ulkoiset tai sisäiset järjestelmät voivat keskustella Magenton kanssa (Huskisson 2010).

Moduuleiden avulla pystytään helposti jaottelemaan sovellus pienemmiksi kokonaisuuksiksi, joita on helpompi hallita ja ylläpitää, kuin yhtä suurta

kokonaisuutta. Hyvin suunniteltu moduulikokonaisuus tarjoaa myös helpon ja loogisen tavan kehittää sovellusta ilman, että toiminnallisuuksia sekoitetaan keskenään mahdollisesti rikkoen alkuperäisiä toiminnallisuuksia.

Magenton ohjelmointirajapinta mahdollistaa Magenton tietojen käsittelyn SOAP-protokollan avulla. SOAP- protokolla on XML-kieleen pohjautuva protokolla, jonka avulla voidaan vaihtaa rakenteellista tietoa verkon yli (Gosnell 2005).

5.3 HTML

HTML (Hypertext Markup Languge) on kuvauskieli, jolla kuvataan hypertekstiä. Hyperteksti on elektronista tekstiä, joka sisältää sivujen välisiä linkkejä. HTML-kielellä siis kuvataan tekstiä, kuvia ja muuta tietoa sisältävien järjestystä, muotoilua ja linkitystä toisiinsa nähden.(Oliver 2001.)

HTML on siis täysin hypertekstin kuvaamista varten tehty kieli, eikä sillä voida toteuttaa ohjelmointia kuten PHP:llä. HTML onkin tunnetusti kuvauskieli, jolla luodaan verkkosivulle visuaalinen rakenne.

5.4 XML

XML (Exstensible Markup Language) on World Wide Web Consortiunim virallisesti suosittelema merkintäkieli tekstimuotoisille dokumenteille. XML tarjoaa käyttäjälleen tyyppimäärityskielen, jonka avulla voidaan mallintaa dokumenttien loogista rakennetta. (Nykänen 2004.)

XML muistuttaa jokseenkin HTML-kuvauskieltä mm. elementtien osalta, mutta XML-kielillä voidaan kuvata kaikkea tietoa, kun taas HTML on tarkoitettu hypertekstin kuvaamiseen.

5.5 Oracle ja MySql

Sekä Oracle, että MySql ovat tietokannanhallintajärjestelmiä, joilla ohjataan tietovarastoina toimivia tietokantoja. Tietokannanhallintajärjestelmät vastaavat kaiken tietokantajärjestelmän tiedon tallennuksesta, turvallisuudesta, eheydestä, ajantasaisuudesta, toipumiskyvystä ja siihen käsiksi pääsystä (Page & Hughes 1999).

Tietokannanhallintajärjestelmät käyttävät asiakas-palvelin arkkitehtuuria. Tämä tarkoittaa käytännössä sitä, että järjestelmät voivat toimia joko omana erillisenä sovelluksenaan, tai palvelimena toisille ohjelmille.

Tietokannanhallintajärjestelmien kanssa keskustelu muista ohjelmista tapahtuu yleisimmin SQL-kyselykielen (Structured Query Language) avulla, jolla käyttäjä pääsee käsiksi tietokantaan tutkiakseen tai muuttaakseen tietokannan dataa.(Heinisuo & Rauta 2007.)

PHP kykenee keskustelemaan sekä Oracle-, että MySql-tietokannanhallintajärjestelmän kanssa, joskin kummallekin järjestelmälle annetaan komentoja eri PHP:n funktioilla ja hieman toisestaan poikkeavilla tavoilla.

Rajapintasovelluksen järjestelmistä Magento käyttää MySql-tietokannanhallintajärjestelmää ja Axapta Oracle- tietokannanhallintajärjestelmää.

6 INTEGRAATION TOTEUTUS RAJAPINTASOVELLUKSEN KAUTTA

Tässä luvussa käsitellään Axaptan ja Magenton integraatiota yhteyksien luomisesta ja käsittelemisestä järjestelmien tietojen kohtaamisiin ja niistä seuraaviin toimenpiteisiin

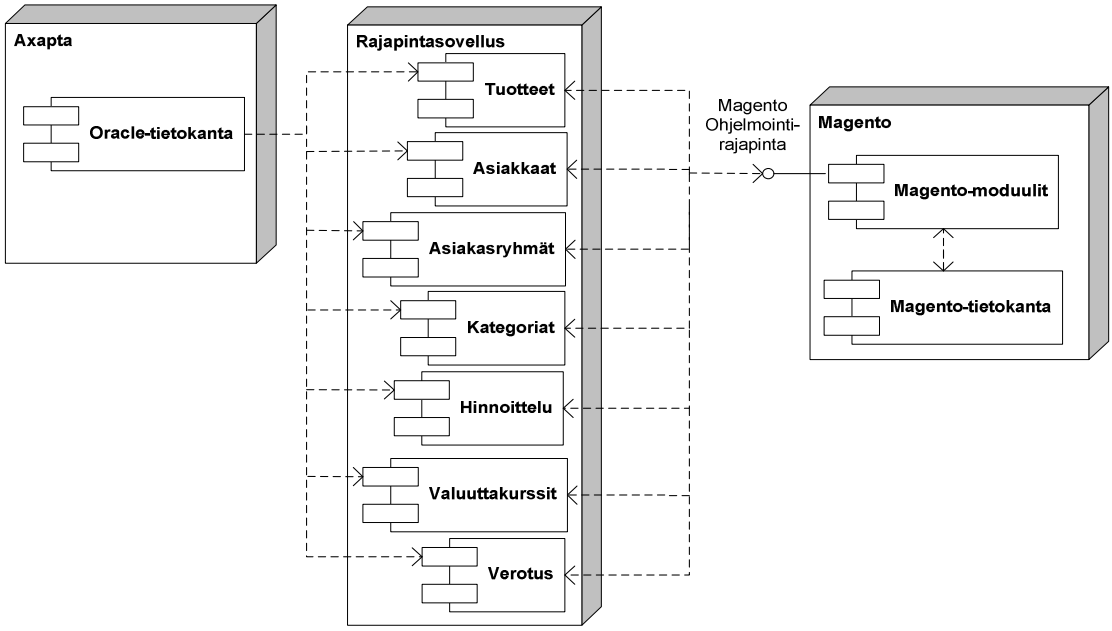
6.1 Integraation rakenne

Rajapintasovelluksessa integraatioarkkitehtuurisena vaatimuksena oli yhdensuuntainen tietovirta Axaptan käyttämästä Oracle-tietokannasta. Tämä tarkoitti käytännössä sitä, että Axaptan tietokannan tietoja ei ikinä lisätä, poisteta tai muokata rajapintasovelluksen kautta.

Rajapintasovellus on integroitu Magentoon Magenton ohjelmointirajapinnan kautta ja vastaavasti Axaptan tietokantaan rajapintasovelluksessa toteutettujen Oracle-tietokantakyselyiden välityksellä.

Rajapintasovellus hakee tietoja Axaptan tietokannasta, käsittelee ne ja siirtää ne Magenton ohjelmointirajapinnan kautta Magenton moduuleille. Moduulit välittävät tiedot edelleen aina Magenton tietokantaan, josta ne ovat Magenton sisällönhallintamoduuleiden käytössä.

Kuten kuviossa 2 esitetään, on Axaptan tietovirta siis yhdensuuntainen rajapintasovellukselle ja riippumaton muista järjestelmistä. Rajapintasovelluksen ja Magenton välillä on molemmansuuntainen tietovirta ja näin ollen ne ovat toisistaan täysin riippuvia tietojärjestelmiä, kuten myös Magenton tietokanta ja sitä hyödyntävät Magenton moduulit.



Kuvio 2. *Järjestelmän tietovirrat ja komponenttien väliset riippuvuudet*

Rajapintasovelluksen avulla haettuja tietoja verrataan toisiinsa rajapintasovelluksessa, minkä jälkeen Axaptasta saapunut tieto joko tallennetaan Magentoon uutena tai päivitettävänä tietueena tai se poistetaan Magentosta.

6.2 Yhteydet

Rajapintasovelluksen yhteydet eri järjestelmiin on toteutettu luomalla yhteyksille oma olio, jonka avulla tietokannan ja ohjelmointirajapinnan yhteydet jaetaan muun rajapintasovelluksen käyttöön. Connections-objektille on määritetty jokaiselle yhteydelle oma public-käyttöalueen kenttä, jolloin luokan kenttiin asetetut yhteydet ovat muiden luokkien saatavilla Connections-objektin ulkopuolelta (ks.kuvio 3).

```
Class Connections{

        public $mag_proxy;
        public $mag_session;
        public $ax_connection;
        public $mag_sql_conn;

        public function OpenAllConnections(){

                $this->OpenMagentoSqlConnection();
                $this->OpenAxaptaConnection();
                $this->OpenMagentoConnection();
                $this->StartMagentoSession();

        }

        public function CloseAllConnections(){

                $this->CloseAxaptaConnection();
                $this->EndMagentoSession();
                $this->CloseMagentoSqlConnection();

        }
```

Kuvio 3. *Connections-luokan yhteyksien hallintaan käytetyt metodit ja luokasta instantioidun objektin ulkopuolelta käytettävissä olevat kentät.*

Jotta luokan yhteysasetusten avaaminen ja sulkeminen olisi yksinkertaisempaa, on luokalle määritetty yksi metodi kaikkien yhteyksien avaamiseen ja toinen kaikkien yhteyksien sulkemiseen.

6.3 Tietojen haku Axaptasta

Tietojen noutaminen Axaptasta toteutettiin kolmella eri kriteerillä tai niiden yhdistelmillä, riippuen Axaptan tietokannan taulujen tarjoamista mahdollisuuksista. Tiedot haetaan joko tiettynä ajanjaksona tapahtuneiden muokkausten perusteella, tietojen aktiiviseen aikaan perustuen tai hakemalla kaikki Axaptasta löytyvät tietueet.

Mikäli Axaptan tietokannan taulu tarjoaa solun, josta käy ilmi viimeisin tietueen muokkauspäivämäärä, on kyselyn tuloksien suodattamiseen käytetty rajapintasovellukselle update_range-vakion arvon avulla määritettyä aikaväliä. Tällöin kyselyn tuloksiin otetaan mukaan vain aikavälin sisällä muokatut tiedot (ks.kuvio 4).

Aikavälin alku määrittyy vähentämällä update_range-vakion arvo vuorokausina sysdate-funktiolla saatavasta sovelluksen suoritusajankohdasta ja aikavälin loppuna toimii sovelluksen suoritusajankohta.

```
SELECT

    trim(accountnum) as accountnumber,
    replace( name, chr( 2 ), " ) as company,
    replace( currency, chr( 2 ), " ) as currency,
    replace( c.email, chr( 2 ), " ) as email,
    trim(replace(pricegroup, chr( 2 ), '')) as pricegroup,
    trim(replace(taxgroup, chr( 2 ), '')) as taxgroup,
    trim(replace(replace(replace(languageid, chr( 2 ), ''),'sv','se'),'en-us','en')) as language,
    trim(replace(round(dimension3_,-1), chr( 2 ), '')) as purpose,
    trim(replace(invoiceaccount, chr( 2 ), " )) as invoiceaccount

FROM

    bmssa.custtable c

WHERE

  modifieddate> = trunc(sysdate)-:update_range
  and dataareaid='vsf'
```

Kuvio 4. *Tietokantakysely Axaptan Oracle-tietokantaan, jossa haetaan sovelluksen suoritusajankohdan ja update_range-vakion arvon välisenä aikana muokattuja tietoja.*

Osalle Axaptan tietokannan tiedoista on määritelty aikaväli, jolloin tietueen arvot ovat voimassa. Tämänkaltaiset tietueet on haettu Axaptan tietokannasta suodattamalla pois kaikki tietueet, joiden voimassaoloajan alkamispäivämäärä on myöhempi ja loppumispäivämäärä aikaisempi kuin kyselyn suoritusajankohta, jonka arvo asetettiin Oracle-tietokantakyselyihin current_date-funktiolla (ks.kuvio 5).

```
SELECT

    …

FROM

    bmssa.pricedisctable p

INNER JOIN

    bmssa.inventtablemodule m

    on m.itemid = p.itemrelation
    and m.dataareaid=p.dataareaid

INNER JOIN

    bmssa.inventtable i

     on i.itemid = p.itemrelation
     and i.dataareaid=p.dataareaid

WHERE

    p.dataareaid='vsf'
    and p.accountcode=1 -
    and p.accountrelation=:id
    and p.fromdate<=current_date
    and (p.todate = to_date( '01-01-1900','dd-mm-yyyy') or current_date<=p.todate)
    and m.moduletype = 2
    and p.relation = 4
    and p.module=1

  order by itemrelation
```

Kuvio 5. *Tietokantakysely Oracle-tietokantaan, jossa haetaan rajapintasovelluksen suoritusajankohtana voimassa olevia tietoja current_date-funktion avulla.*

6.4 Tietojen haku Magentosta

Tietojen haku Magentosta toteutettiin käyttämällä Magenton tarjoamaa ohjelmointirajapintaa, jolla voidaan hakea ja käsitellä Magenton tietoja ulkoisista järjestelmistä. Tietoja voidaan hakea Magentosta joko yksittäisinä tietueina, jolloin ohjelmointirajapinta hakee yksittäisen objektin tiedot, tai listana jolloin ohjelmointirajapinta hakee listan kaikkien objektien tiedoista (ks. kuvio 6).

```
$attribute_sets=$this->connections->mag_proxy->call($this->connections->mag_session,
'product_attribute_set.list');
```

Kuvio 6. *Listan hakeminen Magentosta ohjelmointirajapinnan avulla, joka sisältää kaikki product_attribute_set-objektit.*

Tiedonhaun avuksi Magento tarjoaa myös suodatinmetodin, jonka avulla Magenton ohjelmointirajapinnan kautta voidaan hakea lista Magenton tiedoista käyttämällä objektien ominaisuuksien arvoja hakukriteereinä.

Listaan haettavat objektit suodatetaan antamalla suodattimena toimiva monitasoinen taulukko Magenton ohjelmointirajapinnan kutsulle. Suodatuksessa tarkasteltava ominaisuus määritetään antamalla ominaisuuden nimi avain-arvona suodattimena toimivalle taulukolle. Hyväksyttävät avain-arvona määritetyn ominaisuuden arvot asetetaan avain-arvoa vastaavan elementin arvoksi. Näin listaan valitaan vain ne objektit, joiden avain-arvona suodattimelle määritetyn ominaisuuden arvo on joku hyväksyttävistä arvoista.

Kyseistä tapaa hyödynnettiin rajapintasovelluksessa mm. tuotteiden haussa, jossa Axaptan päivitettyjen tuotteiden tuotenumerot listataan ja ohjelmointirajapinnan avulla haetaan vain Axaptan tuotenumeroita vastaavat objektit vertailua varten (ks.kuvio 7).

Näin Magentosta haetaan vain ajankohtaisia tuotteita, eikä ohjelmointirajapinnan tarvitse hakea kaikkia Magenton tuotteita, jolloin ohjelmointirajapinnan avulla suoritettu kutsu rasittaisi turhaan järjestelmää hakemalla massiivisen taulukon n.2500 tuotteen tiedoista.

```
foreach ($ax_products as $product){

    $product_ids[]=$product->info->itemid;

}

 $filterData = array('sku'=> $product_ids);

$magento_products=$this->connections->mag_proxy->call($this->connections->mag_session,
'product.list',array($filterData));
```

Kuvio 7. *Axaptan tuotteita vastaavien tuotteiden haku Magentosta käyttäen Axaptan tuotenumeroita suodattimen kriteerinä.*

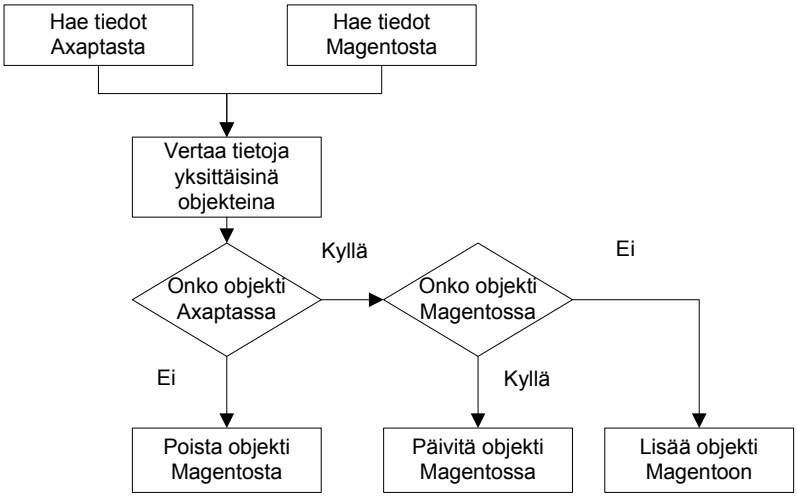
6.5 Tietojen vertaaminen

Sekä Magenton tiedoista, että Axaptan tiedoista muodostetut objektit listataan assosiatiivisiin taulukoihin, joiden indeksiviittaukseksi määritetään tietueen

tunnuksena toimiva arvo, esimerkkinä tuotenumero tuotteiden tiedoista muodostetussa listassa.

Rajapintasovelluksen kautta liikkuvan tietovirran käsittely perustuu järjestelmien tiedoista luotujen objektien vertaamiseen toisiinsa, minkä avulla selvitetään järjestelmät, joissa objektin mukaiset tiedot sijaitsee. Pääperiaatteena rajapintasovelluksen kautta vertailtavat aktiiviset tiedot löytyvät molemmista järjestelmistä, eikä koskaan vain toisesta.

Kuten kuviossa 8 pelkistetysti esitetään, objektien vertailussa käydään läpi molempien järjestelmien objektit yksitellen. Objekteja verrataan toisiinsa, jolloin saadaan selville, missä järjestelmissä objektin mukaiset tiedot sijaitsee ja sen perusteella päätetään objektia koskeva toimenpide.



Kuvio 8. *Pelkistetty tietovirtakaavio järjestelmien tietojen vertailusta ja niiden mukaisista toimenpiteistä.*

Magenton ja Axaptan tietojen mukaisia objekteja verrataan toisiinsa etsimällä toisen järjestelmän objektia vastaavia tietoja toisen järjestelmän objektitaulukosta. Haku objektitaulukosta tapahtuu etsimällä objektien tunnuksena toimivan ominaisuuden arvoa vastaavaa indeksiviittausta. Mikäli vertailtavasta taulukosta löytyy tunnusta vastaava indeksiviittaus, tunnuksen mukainen objekti löytyy molemmista järjestelmistä.

Kuviossa 9 esitetään rajapintasovelluksen asiakasryhmien vertailusta huolehtiva metodi, joka vertailee molempien järjestelmien asiakasryhmäobjekteja käyttäen tunnisteena objektien customer_group-ominaisuuden arvoa.

```
foreach ($ax_groups as $ax_group){
    $id=$ax_group->info->customer_group;
    if (!isset($mag_groups[$id])){
        $ax_group->action="create";
    }
     else if ($mag_groups[$id]){
          $ax_group->action="update";
          $ax_group->magento_id=$mag_groups[$id]->info->customer_group_id;
     }
     $compared_list[$id]=$ax_group;
}
foreach ($mag_groups as $mag_group){
     $id=$mag_group->info->customer_group;
    if (!isset($ax_groups[$id]) ){
        $mag_group->magento_id=$mag_group->info->customer_group_id;
        $mag_group->action="delete";
        $compared_list[$id]=$mag_group;
    }
}
```

Kuvio 9. *Järjestelmien tiedoista luotuja objekteja vertaileva metodi, joka etsii customer_group-ominaisuuden arvon mukaista objektia toisen järjestelmän tiedoista luodusta objektilistasta.*

6.6 Tietojen välittäminen Magentolle

Kuten luvussa 6.1 esiteltiin, ei tietovirta missään sovelluksen vaiheessa ole rajapintasovelluksesta Axaptan tietokantaan päin, vaan Axapta toimii pelkästään tiedonhaun kohteena. Magenton tietoja sen sijaan päivitetään jatkuvasti Axaptan tietojen mukaisesti.

Magenton tietokantaan tietovirta yhdistettiin Magenton ohjelmointirajapinnan kautta. Magenton ohjelmointirajapintaan tietovirta yhdistetään rajapintasovelluksesta SOAP-protokollan avulla, jolle määritetään välityspalvelin ja aktiivinen istunto (ks. kuvio 10).

```
$mag_proxy = new SoapClient( 'http://serveri.fi/magento/api/soap/?wsdl' );

$mag_session = $this->mag_proxy->login( User', 'Password' );
```

Kuvio 10. *Välityspalvelimen määritys SOAP-protokollalle ja istunnon aloittaminen Magenton ohjelmointirajapintaan.*

Magenton ohjelmointirajapintaan muodostetun yhteyden kautta voidaan kutsua Magenton ohjelmointirajapinnalle määritettyjä luokkia. Luokkien metodien kutsuminen toteutetaan antamalla välityspalvelimelle Magentossa xml-kielellä luokalle määritetty alias ja metodin nimi. Kun välityspalvelimelle on annettu luokkaa kuvaava alias ja metodi, annetaan metodille vaaditut parametrit (ks. kuvio 11).

```
private function SetValues($attributes){

  try{

      switch ($this->action){

         case "create":

                  $new_id=$this->connections->mag_proxy->call($this->connections->mag_session,
                  'customer.create', array($attributes));

                  return $new_id;

          break;

           case "delete":

                  $this->connections->mag_proxy->call($this->connections->mag_session,
                  'customer.delete', $this->magento_id);

          break;

          case "update":

                  $this->connections->mag_proxy->call($this->connections->mag_session,
                  'customer.update', array($this->magento_id,$attributes));

          break;

     }

  }

   catch (Exception $e){

        $error=new CustomException($e);

   }

}
```

Kuvio 11. *Asiakasryhmien toimenpiteet suorittava metodi, joka action-arvon mukaisesti suorittaa toimenpiteen Magenton ohjelmointirajapinnan avulla.*

7 RAJAPINTASOVELLUKSEN TOTEUTUS

Rajapintasovelluksen rakenne toteutettiin PHP-ohjelmointikielellä. PHP valittiin ohjelmointikieliksi johtuen PHP:n monipuolisuudesta ja käytännöllisyydestä. Magento-verkkokauppasovellus on pääasiallisesti myös rakennettu PHP-ohjelmointikielellä ja saman kielin käyttäminen oli loogista, jotta järjestelmästä tuli mahdollisimman yhdenmukainen, eivätkä eri ohjelmointikielet aiheuttaneet tarvetta laajentaa rajapintasovellusta tai tarvetta luoda omaa rajapintasovellusta ohjelmointikielin välille.

PHP soveltuu myös olio-ohjelmointiin, mikä on erittäin käytännöllistä sovelluksen jatkokehityksen ja uudelleenkäytettävyyden näkökulmasta. Tästä johtuen rajapintasovellus pyrittiin toteuttamaan noudattaen mahdollisimman tarkasti olio-ohjelmoinnille olennaisia menetelmiä ja ajattelutapoja.

Seuraavissa luvuissa esitellään rajapintasovelluksen yleiseen rakenteeseen liittyvät ratkaisut ja toteutusmallit.

7.1 Sovelluksen rakenne

Rajapintasovellus jaettiin tilausjärjestelmän kohteita mallintaviin luokkiin olio-ohjelmoinnin periaatteiden mukaisesti. Luokat jaettiin kahteen isompaan ryhmään niiden rakenteen mukaisesti. Päivittävät luokat (ks.liite 1) hoitavat päivitettyjen ja uusien tietojen haun, vertailun sekä tietojen välityksen Magenton kohteita mallintaville luokille. Päivittävien luokkien rakenne on toteutettu vastaamaan logiikaltaan ja metodeiltaan läheisesti toisiaan, mikä tekee sovelluksesta yhdenmukaisemman.

Toisena pääluokkaryhmänä voidaan pitää luokkia, jotka mallintavat verkkokaupan yksittäisiä kohteita, kuten tuotteita (ks.liite 2), asiakkaita, tuotekategorioita jne.. Kuten myös päivittävien luokkien rakenne, on myös yksittäisiä kohteita mallintavien luokkien rakenne melkein identtinen.

7.2 Standardisointi

Rajapintasovelluksen osat pyrittiin rakentamaan mahdollisimman yhdenmukaisiksi. Yhdenmukaistaminen helpottaa jatkokehittämistä auttamalla ohjelmoijaa ymmärtämään sovelluksen logiikkaa. Kun ohjelmoija ymmärtää yhden luokan toiminnallisuuden ja logiikan, ymmärtää hän myös muiden luokkien rakenteen.

Standardisointia toteutettaessa hyödynnettiin PHP:n tähän tarkoitukseen tarjoamia elementtejä, joita ovat abstraktiluokat ja rajapinnat.

Monet rajapintasovelluksen luokat omaavat täysin identtisiä metodeja ja kenttiä. Tästä johtuen sovelluksen ohjelmoinnissa pyrittiin välttämään identtisten metodien ja kenttien määrittämistä jokaiselle luokalle erikseen luomalla abstrakti luokka, jonka jäseniksi luokille identtiset metodit ja kentät määritettiin. Perimällä abstraktin luokan lapsiluokat saavat nämä luokan jäsenet käyttöönsä, eikä niitä tarvitse määrittää kuin kerran abstraktiin luokkaan.

Liitteessä 4 esitellään rajapintasovelluksen abstraktiluokka, johon on määritetty kaikille rajapintasovelluksen luokille identtiset metodit: muodostinfunktio sekä aksessori- ja muuttujametodi, joiden käyttöalueeksi on määritetty public, koska metodeja käytetään abstraktin luokan perivien luokkien ulkopuolelta. Abstraktiin luokkaan on määritetty myös kaikille luokille yhteisiä kenttiä, joiden käyttöalueeksi on määritetty protected, jotta ne olisivat kaikkien abstraktiluokan perivien luokkien käytössä.

Rajapintasovelluksessa Magenton tietueita mallintavat luokat sisältävät samaan käyttötarkoitukseen toteutettuja metodeja, joiden toteutustavat voivat kuitenkin poiketa toisistaan huomattavasti. Saman käyttötarkoituksen omaavia metodeja ei siis aina ollut mahdollista määrittää abstraktiin luokkaan johtuen poikkeavuuksista toteutustavoissa. Käyttötarkoitukseltaan samanlaisia luokkia pyrittiin kuitenkin standardisoimaan rajapintojen (ks. liite 5) avulla, jotta kaikki samaan käyttötarkoitukseen tarkoitetut metodit nimettäisiin samoin.

Kuviossa 12 on esitetty rajapinta, johon on määritetty kaikki rajapintasovellukselle tarpeelliset toimenpiteet Magenton tietojen ylläpitämiseksi. Rajapinnan avulla

määritettiin kaikille Magenton tietueita mallintaville luokille toimenpiteet, jotka luokkien tulee pitää sisällään. Rajapinnoilla ei kuitenkaan määritetä metodien toteutustapaa, vaan se jää toiminnallisuuden toteuttavan luokan päätettäväksi, jolloin samoihin käyttötarkoituksiin käytettävät metodit voivat olla toteutuksiltaan tarvittaessa täysin erilaisia. Kuten Gilmore (2005) asian esittää, on rajapinta-metodien käytössä kysymys siitä, että luodaan sääntöjoukko, jota voidaan pitää toteutettuna vasta kun se on toteutettu rajapinnan määritelmien mukaisesti.

```
interface iObject
{
        function update();
        function create();
        function delete();
}
```

Kuvio 12. *Rajapinnan perivältä luokalta vaaditut metodit määriteltynä rajapinnassa.*

Rajapinnassa määritetyt metodit kuitenkin vaativat perivän luokan metodeille public-käyttöalueen. Vastaavasti olio-ohjelmoinnissa on yleisesti pyritty määrittämään metodien käyttöalueet niin, että metodin käyttöalueen tulee olla private, mikäli sitä ei käytetä kuin metodin luokan sisällä. Kuten Newmankin (2005) väittää, hyvässä luokassa on kaikki sisäinen työskentely piilotettu ulkopuolisilta tekijöiltä ja luokassa on julkisia metodeja vain ne, jotka vaaditaan luokan yksinkertaisen rajapinnan toteutukseen luokan toiminnallisuuteen.

Tämän ristiriidan seuraamuksena määriteltiin monista samaan käyttötarkoitukseen käytettävistä metodeista vain muutama rajapintoihin, koska muiden saman käyttötarkoituksen omaavien metodien käyttö tapahtuu rajapinnassa määritettyjen metodien toimesta.

7.3 Tietokantakyselyt

Rajapintasovelluksen tietokantakyselyt on pyritty parametrisoimaan mahdollisten tietokannan väärinkäytösten ja vahinkojen ehkäisemiseksi.

Erityisen alttiita vahingoille ovat tietokantakyselyt, jolloin käyttäjän lomakkeelta saatu arvo sijoitetaan tietokantakyselyyn. Tällaisissa tilanteissa käyttäjä voisi sijoittaa

lomakkeelta tietokantakyselyyn tulevaksi arvoksi SQL-koodia, joka suoritettaisiin varsinaisen tietokantakyselyn sijaan. Näin käyttäjä voisi vahingoittaa tai muokata tietokantaa tai saada haltuunsa arkaluontoista tietoa.

Tällaisten tapausten estämiseksi lähes kaikki rajapintasovelluksen tietokantakyselyt parametrisoitiin varsinkin niissä tapauksissa, joissa käyttäjän syöttämä arvo sijoitetaan kyselyyn. Kuviossa 13 on esitelty asiakastietojen noutamisesta huolehtivan tietokantakyselyn parametrisointi PHP:n Oracle-tietokannoille tarkoitetun oci_bind_by_name-funktion avulla, jolla haetaan käyttäjän lomakkeelta saapuvan accountnumber-arvon mukaiset asiakastiedot Axaptan tietokannasta.

```
$stmt = oci_parse($this->connections->ax_connection, "

    SELECT

        trim(accountnum) as accountnumber,
        replace( name, chr( 2 ), " ) as company,
        replace( currency, chr( 2 ), " ) as currency,
        replace( c.email, chr( 2 ), " ) as email,
        trim(replace(pricegroup, chr( 2 ), '')) as pricegroup,
        trim(replace(taxgroup, chr( 2 ), '')) as taxgroup,
        trim(replace(replace(replace(languageid, chr( 2 ), ''),'sv','se'),'en-us','en')) as language,
        trim(replace(round(dimension3_,-1), chr( 2 ), '')) as purpose,
        trim(replace(invoiceaccount, chr( 2 ), " )) as invoiceaccount

    FROM

         bmssa.custtable c

    WHERE

        trim(c.accountnum)=:accountnumber
        and dataareaid='vsf'

");

oci_bind_by_name($stmt,':accountnumber',$this->accountnumber);

oci_execute( $stmt );
```

Kuvio 13. *Tietokantakyselyn parametrisointi käyttämällä Oracle-tietokannalle tarkoitettua oci_bind_by_name-funktiota.*

Kuviosta 13 käy myös ilmi, kuinka tietokantakyselyitä toteutettaessa muutettiin tietokantataulun kenttien nimet tietokantakyselyn select-osassa vastaamaan Magenton kohteita mallintavien luokkien ominaisuuksia yhdenmukaisuuden saavuttamiseksi.

Rajapintasovelluksen tietokantakyselyiden tulokset muokattiin vastaamaan rajapintasovelluksen rakennetta paremmin myös muuntamalla taulukkona saapuvat kyselyn tulokset objekteiksi. Samalla rajapintasovellus määritettiin muuntamaan taulukon avainarvot pienikirjaimisiksi, jotta myös nämä olisivat yhtenäisiä rajapintasovelluksen muun rakenteen kanssa.

Kuviossa 14 on esitetty metodin osa, joka suorittaa stmt-muuttujan mukaisen kyselyn Oracle-tietokantoja varten käytettävällä oci_execute-funktiolla ja muuttaa tietokantakyselyn taulukkona saapuvat tiedot PHP:n stdClass-oletusluokiksi, minkä jälkeen metodi asettaa objektit assosiatiiviseen taulukkoon.

```
oci_execute( $stmt );

$ax_product_list=array();

while ( $ax_product = oci_fetch_assoc( $stmt ) ) {

    $product=new stdClass();

    $product->info=(object)array_change_key_case($ax_product,CASE_LOWER);

     $ax_product_list[$product->info->itemid]=$product;

}

oci_free_statement( $stmt );
```

Kuvio 14. *Kyselystä taulukkona saatavien tietojen muuttaminen objekteiksi ja ominaisuuksien nimien muuntaminen pienikirjaimisiksi, minkä jälkeen objektit asetetaan assosiatiiviseen taulukkoon.*

7.4 Tiedonvälitys luokille

Koska kyseessä on rajapintasovellus, on yhteysasetukset eri järjestelmiin elinehto sovellukseen sisältyville luokille. Tästä johtuen jokaiselle objektille jaetaan yhteysasetukset välittömästi instantioinnin yhteydessä muodostinfunktion avulla. Tämän jälkeen muodostinfunktio asettaa ne luokan kentän arvoksi kuvion 15 mukaisesti, josta yhteysasetukset on kaikkien luokan ja sen perivien luokkien metodien käytettävissä.

```
Abstract class Object_Abstract{

        protected $action;
        protected $connections;
```

```
public function __construct($Connections) {

    $this->connections=$Connections;

}
```

Kuvio 15. *Muodostinfunktio ja sen toiminnallisuus, joka asettaa yhteys-objektin luokan kentän arvoksi.*

Koska kyseessä on kaikille luokille identtinen ja ehdoton toiminnallisuus, määritettiin muodostinfunktio ja yhteysasetuksien kenttä abstraktiin luokkaan, joka jakaa toiminnallisuuden kaikkien rajapintasovellusten luokkien kesken.

Rajapintasovelluksen luokille jaettiin arvoja myös rajapinnan kautta määrittämällä vakioita ja antamalla niille arvot. Kuviossa 16 esitetään rajapinta, jonka kaikki rajapintasovelluksen luokat toteuttavat ja näin sen avulla voidaan helposti jakaa sovelluksen laajuisesti vaikuttavia arvoja.

```
interface iUpdater
{
        const update_range=1;
        const first_run=false;
}
```

Kuvio 16. *Rajapinnalle määritetyt vakiot, jotka jaetaan rajapinnan toteuttaville luokille.*

Rajapintasovelluksen iUpdater-rajapinta sisältää update-range-vakion, jonka avulla saadaan välitettyä kaikille rajapintasovelluksen päivitettäviä tietoja hakeville metodeille aikaväli, jonka sisällä muokatut tiedot tietokantakyselyssä noudetaan.

Kun rajapintasovelluksessa suoritettava tietokantakysely on aikamääreeseen perustuva, annetaan metodin muuttujalle update-range-vakion mukainen arvo, joka parametrisoidaan ja asetetaan kyselyyn aikamääreeksi kuvion 17 esittämällä tavalla.

```
private function GetUpdatedAxCustomers(){

    $range=self::update_range;

        $stmt = oci_parse($this->connections->ax_connection, "

            SELECT

                trim(accountnum) as accountnumber,
                replace( name, chr( 2 ), " ) as company,
                replace( currency, chr( 2 ), " ) as currency,
                replace( c.email, chr( 2 ), " ) as email,
                trim(replace(pricegroup, chr( 2 ), ")) as pricegroup,
                trim(replace(taxgroup, chr( 2 ), ")) as taxgroup,
                trim(replace(replace(replace(languageid, chr( 2 ), "),'sv','se'),'en-us','en')) as language,
                trim(replace(round(dimension3_,-1), chr( 2 ), ")) as purpose,
                trim(replace(invoiceaccount, chr( 2 ), " )) as invoiceaccount

            FROM

                bmssa.custtable c

            WHERE

                modifieddate = trunc(sysdate)-:update_range
                and dataareaid='vsf'");

                oci_bind_by_name($stmt,':update_range',$range);
```

Kuvio 17. *Metodi, joka suorittaa kyselyn Axaptaan hyödyntämällä parametrisoitua update_range-vakiota.*

Kuten kuviosta 16 käy ilmi, rajapintasovellus sisältää myös toisen vakion, jonka avulla rajapintasovellukselle ilmoitetaan ensimmäisestä rajapintasovelluksen suorituskerrasta, jolloin sovelluksen metodit eivät turhaan suorita vertailua Magenton tietoihin, vaan lisäävät kaikki Axaptasta löytyvät aktiiviset tiedot ilman sovellusta kuormittavaa vertailua.

7.5 Luokkien ominaisuudet ja niiden käsittely

Rajapintasovelluksen luokille määritettiin olio-ohjelmoinnin periaatteiden mukaisesti luokkamuuttujat eli ominaisuudet. Ominaisuuksia käytettiin kuvaamaan objektien attribuutteja, sekä avustamaan metodien toimintaa jakamalla tietoa käyttöalueiden

mukaisesti. Ominaisuuksien määrittelyn ja niiden käsittelyyn tarkoitettujen metodien räätälöinnillä vastaamaan rajapintasovelluksen tarpeita saatiin rajapintasovelluksen luokkien muuttujien käsittely hallitummaksi, estämällä luokkamuuttujien vapaa käsittely.

7.5.1 Ominaisuudet

Koska luokat mallintavat Magenton kohteita, määritettiin luokille Magenton kohteiden attribuutteja vastaavat ominaisuudet (ks.kuvio 18). Näillä ominaisuuksilla kuvaillaan luokasta instantioitua objektia ja niille määritettiin public-käyttöalueet, jotta niille voidaan antaa arvoja objektin ulkopuolelta. Näihin ominaisuuksiin päästään käsiksi PHP:n muuttuja- ja aksessorimetodeilla.

```
class Product  extends Object_Abstract implements iUpdater,iObject{

    public $itemname;
    public $itemid;
    public $websites;
    public $attribute_group;
    public $categories;
    public $description="";
    public $price;
    public $brand_id;
    public $barcode;
    public $unitprice;
    public $itemgroupid;
    public $itembuyergroupid;
    public $brand;
    public $sales_unit;
    public $unitid;
    public $netweight;
    public $length;
    public $height;
    public $width;
    public $in_stock;
    public $tax_class_id;
```

Kuvio 18. *Magenton tuotteita mallintavalle luokalle määritetyt ominaisuudet.*

Osa rajapintasovelluksen luokista tarvitsee metodeilleen muista luokista avustavia tietoja. Koska näillä tiedoilla ei kuvailla objektia ja ne liittyvät aina jonkin tietyn yksittäisen metodin toiminnallisuuden suorittamiseen, määritettiin näille tiedoille private-käyttöalueen ominaisuus. Näin pääsy näihin ominaisuuksiin estetään muuttuja- ja aksessorimetodien kautta ja käsittely voi tapahtua vain erikseen näiden ominaisuuksien käsittelyyn määritettyjen metodien kautta (ks. kuvio 19). Nämä ominaisuudet ovat myös tarkoitettu rajapintasovelluksessa yleensä käytettäväksi vain

luokan sisällä, jolloin olio-ohjelmoinnin periaatteet tukevat myös private-käyttöalueen määrittämistä ominaisuudelle.

```
Class PricingUpdate extends Object_Abstract implements iUpdater{

    private $customer_groups;

     public function update($customer_group_list){

            $this->customer_groups=$customer_group_list;
                …
    }
```

Kuvio 19. Luokan metodi, joka ottaa attribuuttina tietoa vastaan ja asettaa sen kentän arvoksi, josta se on kaikkien luokan metodien käytössä.

Abstraktiluokkaan määritettiin rajapintasovelluksen luokille yhteisiä luokkamuuttujia, joiden avulla saadaan jaettua abstraktiluokan periville luokille avustavaa tietoa. Kuten kuviossa 20 esitetään, abstraktin luokan ominaisuudeksi määritettiin luokkamuuttuja yhteysasetukset sisältävälle Connections-objektille, sekä ominaisuus, jolle asetetaan Magenton kohteita mallintavien luokkien toimenpiteistä kertova arvo. Näin vältyttiin luokille identtisten luokkamuuttujien määrittelemisestä jokaiselle luokalle erikseen.

Connections- ja action-luokkamuuttujien käyttöalueeksi rajapintasovelluksen kantaluokkana toimivassa abstraktiluokassa määritettiin protected, jolloin abstraktiluokan perivät luokat voivat käyttää luokkamuuttujia, mutta niihin ei voida vaikuttaa luokkien ulkopuolelta. Näin luokille tärkeät muuttujat ovat suojassa ulkopuolisilta vaikutuksilta.

```
Abstract Class Object_Abstract{

    protected $action;
    protected $connections;
```

Kuvio 20. *Abstraktiluokkaan määritetyt kentät, jotka ovat käyttöalueeltaan protected ja näin ollen kaikkien abstraktinluokan perivien luokkien käytössä.*

7.5.2 Ominaisuuksien arvojen käsitteleminen

Luokkamuuttujien arvojen asettaminen määritettiin rajapintasovelluksessa tapahtumaan abstraktiluokassa sijaitsevan räätälöidyn muuttujametodin kautta, ylikirjoittamalla PHP:n oma muuttujametodi. Räätälöidyn muuttujametodin avulla huolehditaan siitä, että arvoja ei voida asettaa kuin objektille määritetyille ominaisuuksille, jolloin objektin tiedot pysyvät hallittuna kokonaisuutena ja objektit sisältävät vain niille olennaisia tietoja.

Muuttujametodi laukaisee toiminnallisuutensa, kun objektille määrittelemättömälle tai väärän käyttöalueeltaan ominaisuudelle pyritään antamaan arvo. Tätä toiminnallisuutta hyväksikäytettiin rajapintasovelluksessa määrittämällä räätälöity muuttujametodi luomaan poikkeustilanne pyrkimyksestä asettaa arvoa objektille määrittelemättömälle ominaisuudelle. Poikkeustilanteessa ilmoitetaan käyttäjälle sekä virheellisen ominaisuuden nimi että arvo, jota ominaisuudelle pyrittiin antamaan (ks.kuvio 21). Näin ohjelmoinnin ja suorituksen aikana saadaan tietoa virheelliseen ominaisuuteen viittaamisesta ominaisuuden arvon asettamisen aikana, mikä voi johtua virheistä tai väärinkäytöksistä.

```
public function __set($prop,$value){
		try{
			throw new Exception('Invalid property '.$prop.' with value '.$value);
		}
		catch(Exception $e){
			$error=new CustomException($e);
		}
}
```

Kuvio 21. *PHP:n oman muuttujametodin ylikirjoittava räätälöity muuttujametodi ja toiminnallisuus, joka laukeaa määrittelemätöntä ominaisuutta kutsuessa.*

Objektin ominaisuuksien arvojen noutaminen määritettiin tapahtumaan muuttujametodin tavoin abstraktiluokassa sijaitsevan räätälöidyn metodin kautta ylikirjoittamalla PHP:n oma aksessorimetodi ja luomalla sille rajapintasovelluksen tarpeita paremmin vastaava toiminnallisuus.

Räätälöidyn aksessorimetodin toiminnallisuus on määritetty hakemaan käyttöalueeltaan sallitut objektin ominaisuudet get_object_vars-funktiolla ja etsimään niistä muodostetusta taulukosta metodille parametrina annettua ominaisuuden nimeä vastaavaa ominaisuutta. Nimeä vastaavan ominaisuuden löytyessä metodi palauttaa ominaisuuden arvon. Muuten se luo poikkeustilanteen, joka ilmoittaa virheelliseen ominaisuuteen viittaamisesta kuvion 22 mukaisesti. Näin saadaan tietoa virheelliseen ominaisuuteen viittaamisesta myös ominaisuuksien arvojen noutamisen aikana.

```
public function __get($prop){
                try{
                    if (array_key_exists($prop, get_object_vars($this)))
                    {
                        return $this->prop;
                    }
                     else{
                        throw new Exception('Trying to get non-existing property :'.$prop);
                     }
                }
                catch(Exception $e){
                                $error=new CustomException($e);
                }
}
```

Kuvio 22. *PHP:n oman aksessorimetodin ylikirjoittava räätälöity aksessorimetodi ja toiminnallisuus, jolla tarkistetaan haettavan ominaisuuden oikeellisuus.*

Kuten luvussa 7.3 esiteltiin, tietokantakyselyiden tietokantataulujen kenttien nimet muutettiin vastaamaan Magenton kohteita mallintavien luokkien ominaisuuksia. Lisäksi taulukkona saadut tulokset muunnetaan rajapintasovelluksessa objekteiksi, jolloin tuloksista saadaan ominaisuuksiltaan Magenton kohteita mallintavia luokkia vastaavia objekteja.

Johtuen tietokantakyselyiden tuloksista luotujen objektien yhdenmukaisuudesta rajapintasovelluksen Magenton kohteita mallintavien luokkien kanssa, on niistä helppo asettaa Magenton kohteita mallintavista luokista instantioiduille objekteille ominaisuuksien arvot. Tämä määritettiin toteutettavaksi käymällä tietokantakyselyn

mukaisen objektin ominaisuus-arvoparit läpi yksitellen, asettaen ne samalla Magenton kohteita mallintavasta luokasta luotuun objektiin (ks.kuvio 23).

Jotta ylimääräisten ominaisuuksien arvojen asettamiselta Magenton kohdetta mallintavasta luokasta instantioituun objektiin vältytään, tarkistetaan tietokantakyselyn mukaisen objektin ominaisuuden olemassaolo property_exists-funktiolla, ennen sen mukaisen arvon asettamista objektille. Näin rajapintasovelluksessa vältytään objektille määrittelemättömän ominaisuuden arvon asettamisesta johtuvista poikkeustilanteista.

```
$Customer=new Customer($this->connections);
$Customer->customer_groups=$customer_groups;
    foreach ($customer->info as $prop=>$value){
        if(property_exists($Customer,$prop)){
            $Customer->$prop=$value;
        }
    }
```

Kuvio 23. *Objektin luominen Magenton kohteita mallintavasta luokasta ja sen ominaisuuksien asettaminen tietokantakyselyn tuloksista luodun objektin arvojen mukaisesti.*

7.6 Virheiden ja poikkeustilanteiden hallinta

Virheiden ja poikkeustilanteiden hallinta on ohjelmoinnissa tärkeä ja erittäin hyödyllinen osa-alue. Virheiden ja poikkeustilanteiden oikeanlainen käsittely antaa ohjelmoijalle kattavasti tietoa sovelluksen suorituksen ja ohjelmoinnin aikana tapahtuvista poikkeuksista, virheistä ja mahdollisista väärinkäyttötapauksista.

Rajapintasovelluksen ohjelmoinnin ensimmäisinä toimenpiteinä toteutettiin virhelokin luominen, johon sovellus kirjaa kaikki suorituksen aikana tapahtuneet poikkeustilanteiden ja virheiden tiedot (ks. kuva 1).

1	8	416	/var/www/Ax_interface/product/product_update.php	Undefined index: 6600	2010-09-29 14:33:13
2	8	416	/var/www/Ax_interface/product/product_update.php	Trying to get property of non-object	2010-09-29 14:33:13
3	8	416	/var/www/Ax_interface/product/product_update.php	Trying to get property of non-object	2010-09-29 14:33:13
4	8	416	/var/www/Ax_interface/product/product_update.php	Undefined index: 6200	2010-09-29 14:33:13
5	8	416	/var/www/Ax_interface/product/product_update.php	Trying to get property of non-object	2010-09-29 14:33:13
6	8	416	/var/www/Ax_interface/product/product_update.php	Trying to get property of non-object	2010-09-29 14:33:13
7	8	416	/var/www/Ax_interface/product/product_update.php	Undefined index: 6200	2010-09-29 14:33:13

Kuva 1. *Asiakkaiden luominen suoritetaan Magenton sisällönhallintamoduuliin luodun lomakkeen avulla, joka välittää tiedot rajapintasovellukselle.*

7.6.1 Poikkeustilanteiden hallinta

Poikkeustilanteiden hallinta toteutettiin rajapintasovelluksessa käyttämällä try-catch-lausetta ja laajentamalla PHP:n kantapoikkeusluokkaa. Poikkeusten etsimisessä sovelluksen suorituksen kannalta kriittisissä kohteissa, rajapintasovellus määritettiin käyttämään try-lohkoa, joka yrittää suorittaa lohkon sisällä olevan toiminnallisuuden. Mikäli try-lohkon sisällä olevaa toiminnallisuutta ei jostain syystä voida suorittaa, seuraa tilanteesta automaattisesti poikkeustilanne, joka catch-lohko määritettiin ottamaan talteen.

Catch-lohkolle määritettiin toiminnallisuus, joka instantioi PHP:n kantapoikkeusluokkaa laajentavan CustomException-luokan ja samalla välittää tälle try-lohkon sisällä tapahtuneen poikkeustilanteen tiedot (ks.kuvio 24).

```
private function CreatePassword() {

  try{

      if($this->password==null){

          $chars = "234567890abcdefghijkmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ";
          $length = 8;
          $i = 0;
          $password = "";

          while ($i <= $length) {

               $password=$password.$chars{mt_rand(0,strlen($chars))};

               $i++;
           }

           $this->password=$password;

       }

  }

   catch (Exception $e){
```

```
            $error=new CustomException($e);

        }

}
```

Kuvio 24. *Poikkeusten etsiminen try-lohkon sisältä salasanan luomisessa käytettävässä metodissa ja niiden välittäminen catch-lohkon avulla PHP:n kantapoikkeusluokkaa laajentavalle CustomException-luokalle.*

PHP:n kantapoikkeusluokkaa laajennettiin vastaamaan paremmin rajapintasovelluksen vaatimuksia luomalla PHP:n poikkeusluokkaa laajentava oma poikkeusluokka, jolle määritettiin virhelokia käsittelevä toiminnallisuus.

Poikkeusluokan virhelokia koskeva toiminnallisuus on määritetty laukeamaan heti luokan instantioituessa muodostinfunktion avulla. Näin poikkeusluokan instantioinnin jälkeen ei jouduta kutsumaan erillistä virhelokia käsittelevää metodia catch-lohkossa tai luokan sisällä.

Poikkeustilanteiden kirjaaminen lokiin määritettiin ohjelmoimalla CustomException-luokan muodostinfunktion toiminnallisuus ottamaan talteen sille parametrina välitetty PHP:n generoima poikkeusobjekti. PHP:n poikkeusobjekti sisältää ominaisuuksia, jotka kertovat poikkeustilanteen tietojen mukaisesti muodostetun poikkeusobjektin virhekoodin, poikkeustilanteen aiheuttaneen tapahtuman rivinumeron ja tiedoston sekä virheilmoituksen. Nämä tiedot otetaan talteen ja tallennetaan Magenton MySql-tietokantaan (ks. kuvio 25), josta käyttäjän on helppo seurata sovelluksen suorituksen aikana tapahtuneita poikkeustilanteita.

```
class CustomException extends Exception{

    public function __construct($e) {

                $error_code=$e->code;
                $error_line=$e->line;
                $error_file=$e->file;
                $error_msg=$e->message;

                mysql_query("

                        INSERT INTO bf_error_log

                              (error_level,
                               error_line,
                               error_file,
                               error_msg)

                        VALUES

                               ($error_code,
                                $error_line,
                                '$error_file',
                                '$error_msg')");

                }

}
```

Kuvio 25. *PHP:n kantapoikkeusluokan laajentaminen ja poikkeuksen tallentaminen tietokantaan CustomException- luokan muodostinfunktiossa.*

7.6.2 Virheiden hallinta

Poikkeuksia kriittisemmät tapaukset, virheet, määritettiin myös kirjattavaksi MySql-tietokannassa sijaitsevaan virhelokiin. Tämä edellytti poikkeusluokan mukaisesti PHP:n kantapoikkeusluokan laajentamista ja räätälöityä poikkeusluokkaa vastaavan toiminnallisuuden määrittelemistä. Eroavaisuutena poikkeusluokkaan oli kuitenkin se, että PHP vaatii virheidenkäsittelystä vastaavan metodin määrityksen erityisellä set_error_handler-funktiolla (ks. kuvio 26). Virheiden kirjaaminen ei myöskään vaadi luokan instantioimista kuin kerran sovelluksen alussa, jonka jälkeen PHP suorittaa virhekäsittelymetodin automaattisesti virheen tapahtuessa.

```
class Error extends Exception{
    public function __construct(){
        set_error_handler(array(&$this, 'BfErrorHandler'));
        $this->CreateErrorLogDb();
    }
```

Kuvio 26. *Virhekäsittelymetodin määritys set_error_handler-funktiolla.*

7.7 Aliluokat

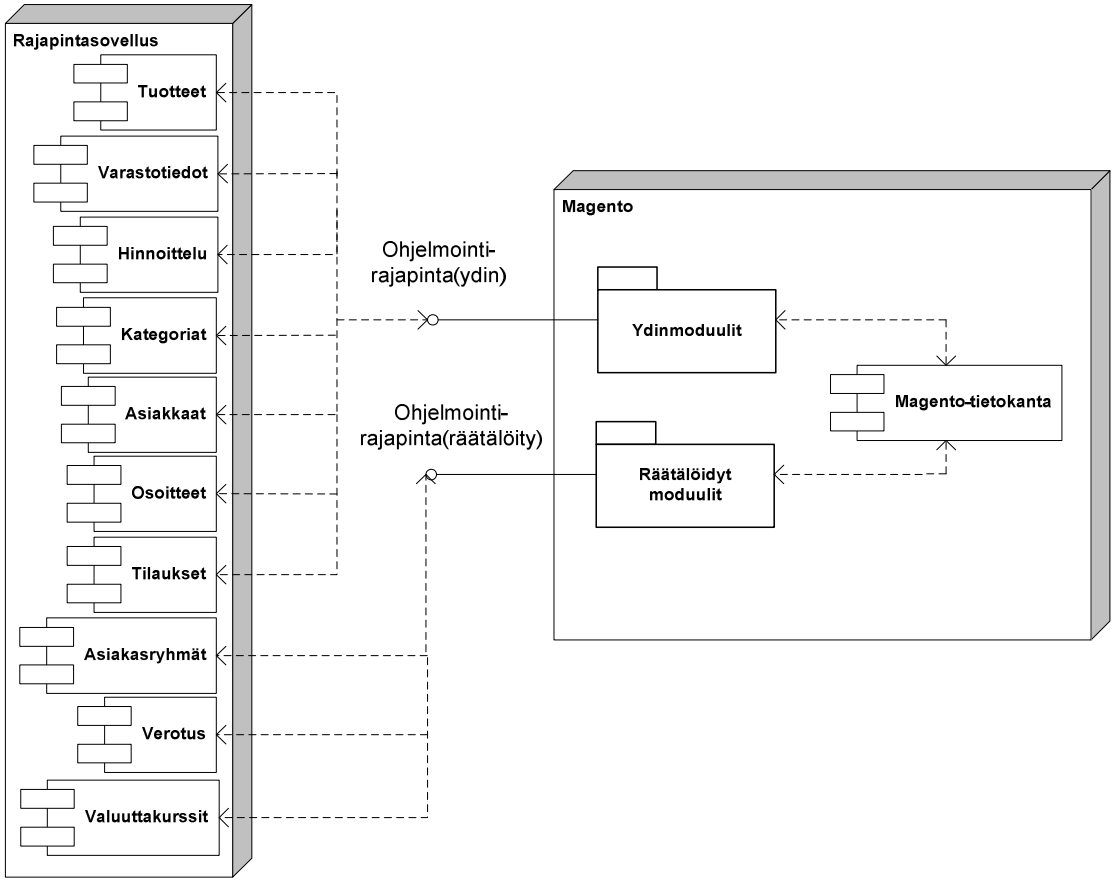
Rajapintasovellus sisältää tuoteryhminä myytäviä tuotteita mallintavan luokan (ks. liite 3). Tuoteryhminä myytävät tuotteet ovat identtisiä perustuotteiden kanssa, muutamaa poikkeusta lukuun ottamatta. Tämä tekee tuoteryhminä myytäviä tuotteita mallintavasta luokasta ihanteellisen käytettäväksi perustuotteita mallintavan luokan (ks. liite 2) aliluokkana.

Perustuotteita mallintavan luokan perivänä luokkana tuoteryhminä myytäviä tuotteita mallintava luokka saa käyttöönsä kaikki perityn luokan metodit ja ominaisuudet, eikä sille näin ollen tarvitse määrittää tuotteisiin liittyviä perustoimintoja tai ominaisuuksia erikseen. Lisäksi perivälle luokalle voitiin näin määrittää sen luokkakohtaiset metodit säilyttämällä perustuotteita mallintava luokka muuttumattomana.

Tuoteryhmänä myytäviä tuotteita mallintavalle luokalle määriteltiin omat metodit luomista ja päivitystä varten, johtuen poikkeavaisuuksista isäntäluokan vastaaviin metodeihin. Koska olio-ohjelmoinnissa suoritetaan aina ensisijaisesti perivän luokan metodi, ei näille metodeille kuitenkaan tarvinut määrittää muista luokista poikkeavia metodien nimiä, vaan perustuotteita mallintavan luokan metodit ylikirjoitetaan tuoteryhmänä myytäviä tuotteita mallintavassa luokassa.

8 MAGENTON TOIMINNALLISUUDEN LAAJENTAMINEN MODUULEILLA

Toimiva ja looginen rajapintasovellus vaati monia toiminnallisuuksia, joiden toteuttamiseksi osoittautui tarpeelliseksi laajentaa Magenton ydinjärjestelmää räätälöidyillä moduuleilla. Järjestelmän turvalliseen laajentamiseen Magento vaatii Magenton ydinjärjestelmän ulkopuolisia moduuleita, joilla laajennetaan Magenton ydinarkkitehtuuria. Magenton ydinjärjestelmää jouduttiin laajentamaan niin ohjelmointirajapinnan kuin sisällönhallintamoduulienkin osalta (ks.kuvio 27).



Kuvio 27. *Magenton ydinjärjestelmää laajentavien moduulien ja ohjelmointirajapinnan sijoittuminen järjestelmien integraatiossa.*

8.1 Moduulin asettaminen Magenton käytettäväksi

Magenton moduulit vaativat xml-kielellä määritetyn asetuksen, joka sisältää moduulin Magenton sisäisesti käytettävän nimen, moduulin aktiivisuudesta kertovan elementin sekä moduulin määrityksen. Määritys kertoo kuuluuko moduuli räätälöityihin moduuleihin vai Magenton ydinmoduuleihin (ks.kuvio 28). Räätälöidyt moduulit sijoitetaan local-ryhmään, josta Magento tarkistaa räätälöidyt moduulit ennen ydinmoduuleiden käyttöä. Näin Magento saa tietoa mahdollisista Magenton ydinmoduuleita ylikirjoittavista- ja laajentavista moduuleista.

```
<?xml version="1.0"?>
<config>
  <modules>
    <Bf_Tax>
      <active>true</active>
      <codePool>local</codePool>
    </Bf_Tax>
  </modules>
</config>
```

Kuvio 28. *Magenton moodulin asetukset, joilla asetetaan moduuli aktiiviseksi ja määritetään moduulin järjestelmän sisäinen nimi.*

8.2 Moduulin konfigurointi

Magenton moduulit konfiguroidaan käyttöä varten luomalla jokaiselle moduulille erillinen konfiguraatiot xml-kielellä. Konfiguraatiotiedostossa määritellään Magentolle moduulin luokat ja ne metodit, joita moduuli hyödyntää. Konfiguraatiotiedostoissa määritellään myös Magenton ydinjärjestelmän moduulit, jotka ylikirjoitetaan räätälöidyillä moduuleilla (ks.kuvio 29).

```
  <global>
    <models>

      <customer>
        <rewrite>
          <customer>Bf_Customer_Model_Customer</customer>
        </rewrite>
      </customer>

      <customer_entity>
        <rewrite>
          <customer>Bf_Customer_Model_Entity_Customer</customer>
```

```
                </rewrite>
            </customer_entity>

        </models>
    </global>
</config>
```

Kuvio 29. *Moduulin konfiguraatio, jossa määritellään moduulin käyttämät luokat ja metodit sekä moduulilla ylikirjoitettavat Magenton ydinmoduulit.*

8.3 Magenton ohjelmointirajapinnan laajentaminen

Vaikka Magenton ydinjärjestelmä tarjosi tarpeeksi kattavasti luokkia ja metodeja Axaptan ja ohjelmointirajapinnan tietojen käsittelyyn, ei Magenton ohjelmointirajapinta kuitenkaan ollut tarpeeksi laaja kaikkien täydelliseen integraatioon tarvittavien Magenton ydinjärjestelmän ja metodien käyttämiseen. Tästä johtuen myös Magenton ohjelmointirajapintaa jouduttiin laajentamaan räätälöidyllä ohjelmointirajapinnalla.

Magenton ohjelmointirajapintaa laajennettiin luomalla räätälöityjä moduuleita, joiden toiminnallisuuteen päästään käsiksi ohjelmointirajapinnan kautta. Jokaiselle moduulille luotiin oma luokka, jonka ohjelmointirajapinnan välityksellä käytettävät metodit instantioivat Magenton sisäisiä luokkia, saaden näin toimivan integraation mahdollistavat metodit käyttöönsä.

Kuviossa 30 esitetään räätälöidyn moduulin luokka (ks. liite 6), joka instantioi muodostinfunktion avulla Magenton asiakasryhmiä mallintavan luokan. Instantionnin jälkeen objekti jaetaan luokan muiden metodien käytettäväksi luokalle määritetyn ominaisuuden avulla, minkä jälkeen räätälöidyn luokan kaikki metodit voivat käyttää toimivan integraation mahdollistavia objektin metodeja. Magenton sisäisestä luokasta instantioidun objektin metodeilla voidaan suorittaa Magenton tietokantaa käsitteleviä toimintoja. Tällainen toiminto on mm. kuviossa 30 esitetty create-metodilla suoritettava asiakasryhmän luominen.

```
class Bf_CustomerGroupApi_Model_ObjectModel_Api
{

  private $model;

  public function __construct() {

       $this->model=Mage::getModel('customer/group');

   }

  public function create($group_name,$tax_class_id){

      $this->model->setCode($group_name)->Save();

      $this->model->setData('tax_class_id',$tax_class_id)->Save();

      return $this->GetIdByCode($group_name);

  }
```

Kuvio 30. *Magenton ydinluokan instantiointi ja sen metodien hyväksikäyttö asiakasryhmien luomisessa.*

Jotta räätälöityjen moduuleiden luokkiin ja niiden metodeihin päästäisiin käsiksi ohjelmointirajapinnan kautta, täytyi luokille ja niiden metodeille määrittää SOAP-tietoliikenneprotokollan kautta käytettävät ohjelmointirajapintakutsut. Ohjelmointirajapintakutsujen määritys tuli toteuttaa jokaiselle moduulille luomalla moduulin käyttöä kuvaava xml-kielinen konfiguraatiotiedosto. Tiedostoihin määritettiin moduulille luotua luokkaa vastaava alias, johon ohjelmointirajapinnan kutsussa viittaamalla saadaan aliasta vastaava luokka instantioitua. Lisäksi konfiguraatiotiedostossa tuli määrittää aliasta vastaavan luokan metodit, jotka ovat käytössä ohjelmointirajapinnan kautta.

Kuviossa 31 on esitetty asiakasryhmien käsittelyyn luodulle moduulille määritetyn luokan xml-kielinen määritys ohjelmointirajapinnalle. Luokan ohjelmistorajapinnan

kutsussa käytettävä alias määritettiin luomalla bf_customer_group-elementti resources-elementin sisään. Luokan käytettävissä olevat metodit määritettiin methods-elementin sisään luomalla jokaista käytettävissä olevaa metodia vastaava elementti.

```
<config>
  <api>
    <resources>
      <bf_customer_group translate="title" module="Bf_CustomerGroupApi">

      …

      <methods>

        <create translate="title" module="bf_customergroupapi">

        …

        </create>

        <delete translate="title" module="bf_ customergroupapi">

        …

        </delete>

        <update translate="title" module="bf_ customergroupapi">

        …

        </update>
      </methods>

      </bf_customer_group>
    </resources>
  </api>
</config>
```

Kuvio 31. *Xml-kielinen konfiguraatio, jolla määritetään ohjelmointirajapinnan kautta käytettävä luokka ja sen metodit.*

Määritysten jälkeen räätälöityihin ohjelmointirajapinnan metodeihin voidaan välittää tietoa vastaavasti välityspalvelimen kautta, kuin Magenton ydin-ohjelmointirajapinnallekin (ks. kuvio 32).

```
$new_group_id=$this->connections->mag_proxy->call($this->connections->mag_session,
'bf_customer_group.create',array($this->customer_group,$this->tax_class_id));
```

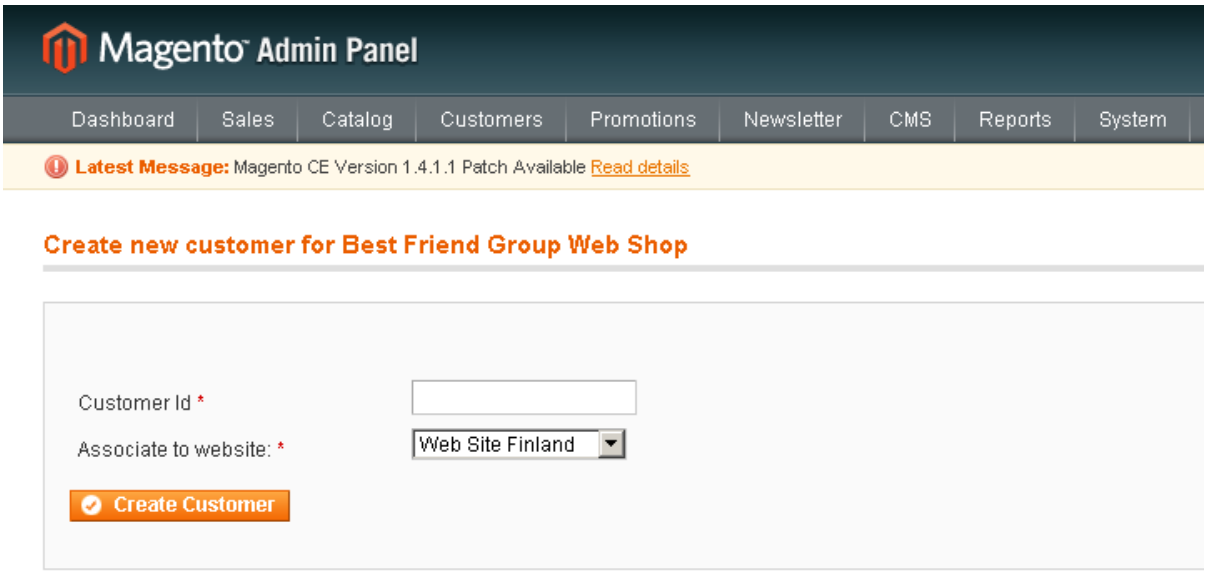
Kuvio 32.*Räätälöidyn moduulin käyttäminen rajapintasovelluksessa Magenton ohjelmointirajapinnan kautta.*

8.4 Magenton sisällönhallintamoduulien laajentaminen

Kuten luvussa 4 mainittiin, oli rajapintasovelluksen asiakastilien luominen tapahduttava Magenton välityksellä. Tätä varten oli tarpeellista luoda Magentoon räätälöity moduuli asiakastilien luomista varten.

Koska rajapintasovellus sisälsi valmiit luokat ja metodit asiakastietojen käsittelyyn, hyödynnettiin niitä myös asiakastilien luomisessa toteuttamalla asiakastilien luominen kokonaan rajapintasovelluksen luokkia ja metodeja hyväksikäyttäen.

Magenton sisällönhallintajärjestelmään luotiin lomakkeen sisältävä moduuli. Sen avulla saadaan luotua yksittäinen asiakas, antamalla lomakkeen tiedoiksi asiakasnumero ja verkkokauppasivusto, jolle asiakastili kohdistetaan (ks.kuva 2). Lomakkeen tietojen avulla sovellus hakee ja tallentaa asiakkaan tiedot rajapintasovelluksen välityksellä Magentoon.



Kuva 2. *Asiakkaiden luominen suoritetaan Magenton sisällönhallintamoduuliin luodun lomakkeen avulla, joka välittää tiedot rajapintasovellukselle ja suorittaa asiakastilin luomisen ohjelmointirajapinnan kautta.*

Sisällönhallintajärjestelmään luotavalle lomakkeelle tuli ensiksi määrittää ulkoasu HTML-kielillä. HTML-kuvauskielen avuksi käytettiin PHP:tä sen alkuperäisessä tarkoituksessaan eli dynaamisten verkkosivujen luomisessa. Näin saatiin lisättyä palvelimelta haettua tietoa kuvauskielen sekaan.

PHP:tä käytettiin lomakkeen pudotusvalikon luomisessa, jossa käyttäjälle tuli näkyä asiakastilille kohdistettavan verkkosivun nimi ja sisällönhallintajärjestelmälle tuli välittää verkkosivun nimeä vastaava tunnusnumero. Koska verkkosivujen tiedot saattavat ajan myötä muuttua, toteutettiin listaus pudotusvalikkoon noutamalla verkkosivun nimet ja niitä vastaavat tunnukset Magenton tietokannasta Magenton ydinjärjestelmän luokkien ja metodien avulla, kiinteiden tunnus-nimi parien sijaan (ks. kuvio 33). Näin pudotusvalikon tiedot pysyvät aina ajantasalla, vaikka verkkosivujen tietoja muokattaisiinkin muualla järjestelmässä.

```
<tr>
    <td class="label">
          <?=$this->__('Customer Id')?> <span class="required">*</span>
    </td>

    <td class="input-ele">
          <input class="input-text required-entry" name="customerform[customerfield]" />
      </td>

      <td class="a-right"></td>
      <br /><br />
</tr>

<tr>

   <td class="label">
          <? $this->__('Associate to website:')?> <span class="required">*</span>
    </td>

    <td class="input-ele">

        <select name="customerform[websitefield]" id="dropdown" title="" value="" class="input-
                    text" type="dropdown" />

       <?php

            $websites=Mage::app()->getWebsites();

            foreach($websites as $website){

                $website_data=$website->getData(); ?>

               <option value="<?php echo $website_data["website_id"]; ?>"
                    <?php if($website_data["is_default"]==1):?> selected<?php endif;?>>
                    <?php echo $website_data["name"]; ?>
               </option>
```

```
<?php } ?>

 </select>
```

Kuvio 33. *HTML-kuvauskielellä toteutettu ulkoasu asiakastilien luomiseen tarkoitetulle lomakkeelle. Apuna on käytettiin PHP-koodia, jonka avulla lomakkeelle saadaan haettua tietoa palvelimella toimivasta Magenton tietokannasta.*

Asiakastilin luomiseen tarkoitettu lomake välittää arvot moduulille määritetylle luokalle. Luokka vastaanottaa arvot, instantioi objektit rajapintasovelluksen luokista ja pääsee näin käsiksi rajapintasovelluksen luokkien metodeihin ja ominaisuuksiin, joiden avulla luokka suorittaa asiakastilin luomiseen tarvittavat toimenpiteet (ks.kuvio 34).

```
public function postAction()
{

   try{

        $post = $this->getRequest()->getPost();

        $customer_id=$post["customerform"]["customerfield"];
        $website_id=$post["customerform"]["websitefield"];

        $connections=new Connections();
        $connections->OpenAllConnections();

        $Customer=new Customer($connections);
        $Customer->website=$website_id;
        $new_pass=$Customer->CreateByAccountNumber($customer_id);

         if(!$new_pass){

              throw new Exception("Account with accountnumber: ".$customer_id." doesn't exist!");

         }

          else{

                    $message = $this->__('Customer Id '.$customer_id.' is created succesfully.<br /><br />
                    Password is : <b><u>'.$new_pass.'</u></b><br /><br />Please write this down!');

                    Mage::getSingleton('adminhtml/session')->addSuccess($message);

          }

           } catch (Exception $e) {

                Mage::getSingleton('adminhtml/session')->addError($e->getMessage());

           }
```

Kuvio 34. *Asiakastilien luomiseen tarkoitetun moduulin luokka, joka vastaanottaa lomakkeen tiedot ja suorittaa niiden mukaiset toimenpiteet rajapintasovelluksen välityksellä.*

9 POHDINTA

Opinnäyteyön tuloksena saatiin Best Friend Group Oy:lle toteutettua vaatimuksia vastaava, lähes täydellisesti tilausjärjestelmän automatisoiva rajapintasovellus, joka sisältää myös moduulin asiakastilien luomiseen. Kuitenkin ennen sovelluksen siirtämistä tuotantoon, on tilausjärjestelmälle kehitettävä vielä ulkoasu sekä suoritettava tilausjärjestelmän kattava testaus.

Tilausjärjestelmän toteutus perustui moduulipohjaiseen Open Source – järjestelmään. Tällaiset järjestelmät ovat entistä laajemmin yleistymässä ja ne tarjoavat lukemattomia mahdollisuuksia ohjelmistokehittäjille luoda kattavia järjestelmiä omiin tarpeisiinsa.

Nykyisissä Open Source – järjestelmissä on selvästi kiinnitetty enemmän huomiota integraatioon muiden järjestelmien kanssa. Tämä näkyy mm. ohjelmointirajapintojen yleistymisessä pienemmissäkin sovelluksissa, sekä eri sovelluksia integroivien moduulien saatavuudesta järjestelmille.

Rajapintasovellus toteutettiin mahdollisimman tarkasti olio-ohjelmoinnin periaatteita noudattaen. Olio-ohjelmoinnin periaatteilla luotu rajapintasovellus tarjoaa hyvän perustan jatkokehitykselle loogisen ja selkeän rakenteen ansiosta.

Rajapintasovelluksen laajuudesta johtuen toteutuksessa tärkeäksi tekijäksi osoittautui standardisointi, jota varten ohjelmointikielit sisältävät nykyään melko kattavasti erilaisia työkaluja. Näiden työkalujen avulla saadaan laajat sovellukset pidettyä yhtenäisenä kokonaisuutena ja luotua helposti ymmärettävä rakenne jatkokehitystä ajatellen.

Rajapintasovelluksen toteutuksen myötä karttuneesta kokemuksesta voi myös sanoa, että proseduraalisen ohjelmoinnin soveltaminen rajapintasovelluksen kaltaiseen laajaan kokonaisuuteen tekisi sen rakenteen ymmärtämisestä jatkokehityksessä hyvin haastavaa. Proseduraalisessa ohjelmoinnissa on huomattavasti vaikeampaa, ellei jopa tällaisessa mittakaavassa mahdotonta, pitää järjestelmän osia hallittuina ja selkeinä kokonaisuuksina.

Opinnäytetyöhön liittyvänä negatiivisena kokemuksena voi nostaa esiin Magenton kohdalla dokumentaation puutteen. Magenton kokoisissa olio-ohjelmointiin

perustuvissa sovelluksissa kattava dokumentaatio helpottaisi kehitystyötä huomattavasti, säästäen aikaa ja resursseja.

LÄHTEET

Converse, T., Joyce, P. & Morgan, C. 2004
PHP5 and MySql Bible. Wiley Publishing Inc. Indianapolis, USA

Gilmore, W.J. 2005
PHP & MySQL – Tehokas hallinta. Suom.Arto Kuvaja. Readme.fi. Gummeruksen Kirjapaino Oy. Helsinki.

Gosnell, D. 2005
Professional development with Web Api's. Wiley Publishing Inc. Indianapolis, USA

Hallavo, J., Valvanne, J., Lindqvist, M. & Luoto, E. 2009
Verkkokaupan rautaisannos. Smilehouse Oy. Helsinki

Heinisuo, R. & Rauta, I. 2007
PHP ja MySQL: tietokantapohjaiset verkkopalvelut. Gummerus Kirjapaino Oy. Helsinki

Huskisson, J. 2010
Magento 1.3: PHP Developer's Guide. Packt Publishing Ltd. Birmingham, UK

Nykänen, O.2004
XML. Toolkit. Docendo Finland Oy. Jyväskylä

Newman, C.2005
Sams Teach Yourself PHP in 10 Minutes. Sams Publishing. Indianapolis, USA

Oliver, D. 2001
HTML & XHTML – Trainer Kit. Suom. Jussi Arola. IT Press. Edita Publishing Oy. Helsinki.

Page, W. & Hughes, N. 1999

Using Oracle 8. Que Publishing. New York, USA

Rosen, A.1999

The e-commerce question and answer book: a survival guide for business managers. Amacom. New York, USA

Tilastokeskus 2009

Verkkodokumentti. Luettu 17.8.2010.

Etusivu > Tilastot > Tiede, teknologia ja yhteiskunta > Tietotekniikan käyttö yrityksissä > 2009

http://www.tilastokeskus.fi

Timmers, P.2000

Electronic commerce: strategies and models for business-to-business trading. John Wiley & Sons. Chichester, UK

Varien 2009

Magento User Guide. Irubin Consulting Inc. Los Angeles, USA

LIITE 1 Rajapintasovelluksen tuotteet päivittävä luokka

```
<?php

require_once("bom_products.php");
require_once("stock.php");

class ProductUpdate extends Object_Abstract implements iUpdater{

   /*
   Metodi:Suorittaa  tuotteiden päivityksen.
   Vastaanottaa:Lista Magenton tuotekategorioista(array), lista Magenton veroryhmistä(array).
   Palauttaa:Listan uusista tuotteista(array).
    */
   public function update($mag_categories,$tax_classes){

      $new_products=array();

       if (self::first_run==true){

          $compared_products=$this->CompareProducts(null,$this->GetAxProducts());

       }

       else{

          $ax_products=$this->GetUpdatedAxProducts();

          $compared_products=$this->CompareProducts( $this->
          GetMagProducts($ax_products),$ax_products);

       }

       $new_products=$this->SetValues($this->SetTaxClasses($this->
       SetCategories($compared_products,$mag_categories),$tax_classes));

       return $new_products;

    }

    /*
   Metodi:Hakee Axaptan päivitettyjä tuotteita vastaavat Magenton tuotteet.
   Vastaanottaa:Listan Axaptan tuotteista(array).
   Palauttaa:Listan Magenton tuotteista(array).
   */
   public function GetMagProducts($ax_products){

      foreach ($ax_products as $product){

         $product_ids[]=$product->info->itemid;

      }

      $filterData = array('sku'=> $product_ids);

      $magento_products=$this->connections->mag_proxy->call($this->connections->
      mag_session, 'product.list',array($filterData));

      $mag_product_list=array();
```

```
    foreach ($magento_products as $mag_product){

      $product=new stdClass();

      $product->info=(object)array_change_key_case($mag_product,CASE_LOWER);

      $mag_product_list[$product->info->sku]=$product;

    }

     return $mag_product_list;

 }

 /*
Toiminto:Hakee kaikki Axaptan aktiiviset tuotteet.
Palauttaa:Listan Axaptan tuotteista(array).
*/
private function GetAxProducts(){

    $stmt = oci_parse($this->connections->ax_connection, "

        SELECT

            trim(i.itemid) as itemid,
            i.itemname,
            i.itemtype,
            substr( i.itemgroupid, 1, 2 ) || '00' as itemgroupid,
            replace( i.barcode, chr( 2 ), '' ) as barcode,
            replace( i.vpg_brandid, chr( 2 ), '' ) as brand_id,
            replace( i.itembuyergroupid, chr( 2 ), '' ) as itembuyergroupid,
            m.taxitemgroupid,
            m.price * m.quantity as price,
            m.price as unitprice,
            m.quantity as sales_unit,
            m.unitid,
            decode( m.blocked, '0', '1','0' ) as in_stock,
            replace( b.name, chr( 2 ), '' ) as brand,
            p.width,
            p.height,
            p.length,
            p.netweight

        FROM

            bmssa.inventtable i

        LEFT JOIN

            bmssa.inventtablemodule m

            on  m.itemid = i.itemid

            and m.dataareaid = i.dataareaid

            and m.moduletype = 2 -- 0 = inventory, 1 = purchase, 2 = sales

        LEFT JOIN

            bmssa.btable b

            on b.brandid = i.vpg_brandid
```

```
            and b.dataareaid=i.dataareaid
        LEFT JOIN
            bmssa.info p
            on  p.itemid = i.itemid
            and p.dataareaid = i.dataareaid
            and p.packagetype = 0
        LEFT JOIN
            bmssa.bomversion v
            on v.itemid = i.itemid
            and v.dataareaid = i.dataareaid
        LEFT JOIN
           bmssa.bom bom
        ON
           v.bomid=bom.bomid
           and bom.dataareaid = i.dataareaid
        WHERE

           i.dataareaid='vsf'
          and i.itembuyergroupid <> 'del'
          and i.itemtype in ( 0, 1 )
       GROUP BY
          i.itemid,
          i.itemname,
          i.itemtype,
          i.itemgroupid,
          i.barcode,
          i.vpg_brandid,
          i.itembuyergroupid,
          m.taxitemgroupid,
          m.price,
          m.quantity,
          m.unitid,
          m.blocked,
          b.name,
          p.width,
          p.height,
          p.length,
          p.netweight

" );
```

```
    oci_execute( $stmt );

    $ax_product_list=array();

    while ( $ax_product = oci_fetch_assoc( $stmt ) ) {

        $product=new stdClass();

        $product->info=(object)array_change_key_case($ax_product,CASE_LOWER);

        $ax_product_list[$product->info->itemid]=$product;

    }

    oci_free_statement( $stmt );

    return $ax_product_list;

}

/*
Toiminto:Hakee päivitetyt tuotteet Axaptasta.
Palauttaa:Listan Axaptan tuotteista(array).
*/
private function GetUpdatedAxProducts(){

    $range=self::update_range;

    $stmt = oci_parse( $this->connections->ax_connection, "

        SELECT

            trim(i.itemid) as itemid,
            i.itemname,
            i.itemtype,
            substr( i.itemgroupid, 1, 2 ) || '00' as itemgroupid,
            replace( i.barcode, chr( 2 ), '' ) as barcode,
            replace( i.vpg_brandid, chr( 2 ), '' ) as brand_id,
            replace( i.itembuyergroupid, chr( 2 ), '' ) as itembuyergroupid,
            m.taxitemgroupid,
            m.price * m.quantity as price,
            m.price as unitprice,
            m.quantity as sales_unit,
            m.unitid,
            decode( m.blocked, '0', '1','0' ) as in_stock,
            replace( b.name, chr( 2 ), '' ) as brand,
            p.width,
            p.height,
            p.length,
            p.netweight

        FROM

            bmssa.inventtable i

        LEFT JOIN

            bmssa.inventtablemodule m

            on  m.itemid = i.itemid

            and m.dataareaid = i.dataareaid
```

```
      and m.moduletype = 2 -- 0 = inventory, 1 = purchase, 2 = sales
 LEFT JOIN
      bmssa.btable b
      on b.brandid = i.vpg_brandid
      and b.dataareaid=i.dataareaid
 LEFT JOIN
      bmssa.info p
      on  p.itemid = i.itemid
      and p.dataareaid = i.dataareaid
      and p.packagetype = 0
LEFT JOIN
      bmssa.bomversion v
      on v.itemid = i.itemid
      and v.dataareaid = i.dataareaid
LEFT JOIN
     bmssa.bom bom
ON
     v.bomid=bom.bomid
     and bom.dataareaid = i.dataareaid
WHERE
     GREATEST(
     NVL(i.modifieddate, to_date('1900-01-01','YYYY-MM-DD')),
     NVL(m.modifieddate, to_date('1900-01-01','YYYY-MM-DD')),
     NVL(p.modifieddate, to_date('1900-01-01','YYYY-MM-DD')),
     NVL(v.modifieddate, to_date('1900-01-01','YYYY-MM-DD')),
     NVL(v.fromdate,to_date('1900-01-01','YYYY-MM-DD')),
     NVL(v.todate, to_date('1900-01-01','YYYY-MM-DD')),
     NVL(bom.fromdate,to_date('1900-01-01','YYYY-MM-DD')),
     NVL(bom.todate, to_date('1900-01-01','YYYY-MM-DD')))>=trunc(sysdate)-:update_range


     and GREATEST(
     NVL(i.modifieddate, to_date('1900-01-01','YYYY-MM-DD')),
     NVL(m.modifieddate, to_date('1900-01-01','YYYY-MM-DD')),
     NVL(p.modifieddate, to_date('1900-01-01','YYYY-MM-DD')),
     NVL(v.modifieddate, to_date('1900-01-01','YYYY-MM-DD')),
     NVL(v.fromdate,to_date('1900-01-01','YYYY-MM-DD')),
     NVL(v.todate, to_date('1900-01-01','YYYY-MM-DD')),
     NVL(bom.fromdate,to_date('1900-01-01','YYYY-MM-DD')),
     NVL(bom.todate, to_date('1900-01-01','YYYY-MM-DD')))<=trunc(sysdate)
```

```
            and i.dataareaid='vsf'

            and i.itemtype in ( 0, 1 )

        GROUP BY

            i.itemid,
            i.itemname,
            i.itemtype,
            i.itemgroupid,
            i.barcode,
            i.vpg_brandid,
            i.itembuyergroupid,
            m.taxitemgroupid,
            m.price,
            m.quantity,
            m.unitid,
            m.blocked,
            b.name,
            p.width,
            p.height,
            p.length,
            p.netweight
             " );

    oci_bind_by_name($stmt,':update_range',$range);

    oci_execute( $stmt );

    $ax_product_list=array();

    while ( $ax_product = oci_fetch_assoc( $stmt ) ) {

        $product=new stdClass();

        $product->info=(object)array_change_key_case($ax_product,CASE_LOWER);

        $ax_product_list[$product->info->itemid]=$product;

     }

     oci_free_statement( $stmt );

     return $ax_product_list;

}
/*
Metodi:Vertaa Magenton ja Axaptan tuotteita toisiinsa ja asettaa niille arvon, joka kertoo
kuinka tuote tulee käsitellä.
Vastaanottaa:Lista Axaptan tuotteista(array), lista Magenton tuotteista(array).
Palauttaa:Lista toisiinsa verratuista tuotteista(array).
*/
private function CompareProducts($magento_product_list,$ax_product_list){

    $compared_product_list=array();

     if ($magento_product_list!=null){

        foreach ($ax_product_list as $ax_product){

            $id=$ax_product->info->itemid;
```

```
            if (isset($magento_product_list[$id]) && $ax_product->info->itembuyergroupid=="del"){

                $ax_product->action="disable";

            }

           else if(isset($magento_product_list[$id]) && $ax_product->info->itembuyergroupid!="del"){

                $ax_product->action="update";

            }

            else{

                $ax_product->action="create";

            }

            $compared_product_list[$id]=$ax_product;

          }

       }

       else{

           foreach ($ax_product_list as $ax_product){

              $ax_product->action="create";
              $compared_product_list[$ax_product->info->itemid]=$ax_product;
           }

      }

      return $compared_product_list;

}

   /*
   Metodi:Asettaa tuotteille tuoteryhmien mukaiset Magento-tunnukset tuotekategorioita varten.
   Vastaanottaa:Lista toisiinsa verratuista tuotteista(array),lista Magenton tuotekategorioista(array).
   Palauttaa:Tuotelistan, johon on lisätty Magento Api:n kanssa yhteensopivat tuotekategoriatunnukset(array).
   */
   private function SetCategories($products,$mag_categories){

      foreach ($products as $product){

         $categories=array();

         $categories[]=$mag_categories[$product->info->itemgroupid]->info->category_id;

         $categories[]=$mag_categories[$product->info->itemgroupid]->info->parent_id;

         if (isset($mag_categories[$product->info->brand_id])){

             $categories[]=$mag_categories[$product->info->brand_id]->info->category_id;

             $categories[]=$mag_categories[$product->info->brand_id]->info->parent_id;

          }
```

```
            else {

              $categories[]=$mag_categories[200000]->info->category_id; //Mikäli tuote ei kuulu strategisiin brändeihin, laitetaan se "Others"-kategoriaan(20000)

               $categories[]=$mag_categories[200000]->info->parent_id;

             }

            $product->categories=$categories;

          }

       return $products;

    }

    /*
    Metodi:Asettaa tuotteille niiden veroryhmien mukaiset Magento-tunnukset.
    Vastaanottaa:Lista toisiinsa verratuista tuotteista(array), lista Magenton veroryhmistä(array).
    Palauttaa:Tuotelistan, johon on lisätty Magento Api:n kanssa yhteensopivat veroryhmätunnukset(array).
    */
    private function SetTaxClasses($products,$tax_classes){

       foreach ($products as $product){

          $product->tax_class_id=$tax_classes[$product->info->taxitemgroupid]->info->mag_id;

       }

       return $products;

    }

    /*
    Metodi:Hakee kaikkien Magenton-sivustojen Magento-tunnukset.
    Palauttaa:Listan Magento-sivustojen tunnuksista(array).
    */
    private function GetAllWebsiteIds(){

      $websites=$this->connections->mag_proxy->call($this->connections->mag_session,
      'bf_website.website_list');

       $ids=array();

       foreach($websites as $website){

          $ids[]=$website["website_id"];

       }

       return $ids;

    }

    /*
    Metodi:Hakee kaikki Magenton attribuuttiryhmät.
    Palauttaa:Listan Magenton attribuuttiryhmistä(array).
    */
    private function GetAttributeGroups(){
```

```
    $attribute_sets=$this->connections->mag_proxy->call($this->connections->
    mag_session, 'product_attribute_set.list');

    $sets=array();

    foreach($attribute_sets as $set){

        $sets[$set["name"]]=$set;

     }

     return $sets;

}

/*
Metodi:Luo jokaisesta tuotteesta Product-objektin,antaa sille tarvittavat arvot ja
suorittaa päivitystoimenpiteet.
Vastaanottaa:Lista toisiinsa verratuista tuotteista(array).
Palauttaa: Listan Magentoon lisätyistä tuotteista(array).
*/
private function SetValues($product_list){

    $attributes_groups=$this->GetAttributeGroups();

    $all_website_ids=$this->GetAllWebsiteIds();

    $new_products=array();

    foreach ($product_list as $product){

        $action=$product->action;

        if($product->info->itemtype==1){

            $Product=new BomProduct($this->connections);
            $Product->attribute_group=$attributes_groups["bf_attribute_set_bom"]["set_id"];

        }

        else if($product->info->itemtype==0){

            $Product=new Product($this->connections);
            $Product->attribute_group=$attributes_groups["bf_attribute_set"]["set_id"];

        }

        foreach($product->info as $property=>$value){

            if(property_exists($Product,$property)){

                $Product->$property=$value;

            }

        }

        $Product->categories=$product->categories;
        $Product->tax_class_id=$product->tax_class_id;
        $Product->websites=$all_website_ids;
```

```
            $Product->$action();
            if($action=="create"){
                $new_products[$product->info->itemid]=$product;
            }
        }
        return $new_products;
    }
}
?>
```

LIITE 2 Tuotteita mallintava luokka

```
<?php

class Product  extends Object_Abstract implements iUpdater,iObject{

    public $itemname;
    public $itemid;
    public $websites;
    public $attribute_group;
    public $categories;
    public $description="";
    public $price;
    public $brand_id;
    public $barcode;
    public $unitprice;
    public $itemgroupid;
    public $itembuyergroupid;
    public $brand;
    public $sales_unit;
    public $unitid;
    public $netweight;
    public $length;
    public $height;
    public $width;
    public $in_stock;
    public $tax_class_id;

    /*
    Metodi:Hakee ja asettaa tuotteen nimelle käännökset,sekä asettaa ominaisuudet Magento Api:n
            kanssa yhteensopiviksi attribuuteiksi ja päivittää tuotteen tiedot Magento Api:n välityksellä.
    */
    public function update(){

        $this->action="update";

        $this->SetValues($this->SetAttributesForMagento(),$this->SetTranslationAttributes(
        $this->GetNameTranslations($this->itemid,$this->itemname)));
}

    /*
    Metodi:Hakee ja asettatta tuotteen nimen käännöset,sekä asettaa ominaisuudet Magento Api:n
    kanssa yhteensopiviksi attribuuteiksi ja luo uuden tuotteen Magento Api:n välityksellä.
    */
    public function create(){

       $this->action="create";

       $this->SetValues($this->SetAttributesForMagento(),
       $this->SetTranslationAttributes($this->GetNameTranslations($this->itemid,$this->itemname)));

    }

    /*
    Metodi:Poistaa tuotteen Magenton tuotekategorioista.
    */
    public function disable(){

       $this->action="disable";

       $this->SetValues(null,null);
```

```
}

/*
Metodi:Poistaa tuotteen Magentosta.
*/
public function delete(){

   $this->action="delete";

   $this->SetValues(null,null);

}

/*
Metodi:Muuntaa tarvittavat luokan ominaisuudet yhteensopiviksi Magento Api:n kanssa.
Palauttaa:Magento Api:n kanssa yhteensopivan attribuuttilistan(array).
*/
protected function SetAttributesForMagento(){

   $attributes= array(
                'name' => $this->itemname,
                'websites' => $this->websites,
                'categories' => $this->categories,
                'description' => '',
                'price' => $this->price,
                'bf_brand_id' =>$this->brand_id,
                'bf_item_barcode' => $this->barcode,
                'bf_unit_price'=> $this->unitprice,
                'bf_item_group' => $this->itemgroupid,
                'bf_item_brand' => $this->brand,
                'bf_sales_unit'=> $this->sales_unit,
                'bf_unit'=> $this->unitid,
                'weight'=> $this->netweight,
                'bf_length'=> $this->length,
                'bf_height'=> $this->height,
                'bf_width'=> $this->width,
                'bf_blocked'=> $this->in_stock,
                'status'=> $this->CheckIfEnabled(),
                'tax_class_id' => $this->tax_class_id
  );

   return $attributes;

}

/*
Metodi:Luo listan Magento Api:n kanssa yhteensopivista kielikohtaisista attribuuteista.
Palauttaa:Magento Api:n kanssa yhteensopivan kielikohtaisen attribuuttilistan(array).
*/
protected function SetTranslationAttributes($language_list){

   $translation_attributes=array();

   foreach ($this->store_language_list as $wsview=>$code){

      $translation_attributes[$code]=array('name'=>$language_list[$code]." x ".$this->sales_unit);

   }

   return $translation_attributes;
```

```
}

/*
Metodi:Määrittää onko tuote aktiivinen.
Palauttaa:Tiedon tuotteen aktiivisuudesta(bool).
*/
protected function CheckIfEnabled(){

   if($this->price<=0 || $this->itembuyergroupid=="del"){

      return false;

   }

   else {

      return true;

   }

}

/*
Metodi:Hakee tuotteen nimelle käännökset.
Palauttaa:Listan tuotteen nimelle löytyneistä käännöksistä(array).
*/
protected function GetNameTranslations($itemid,$itemname){

   $stmt = oci_parse( $this->connections->ax_connection, "

      SELECT

         replace(txt, chr( 2 ), " ) as txt,
         replace(replace(languageid,'sv','se'),'en-us','en') as LANGUAGEID

      FROM

         bmssa.inventtxt

      WHERE

         itemid= '$itemid'
         and dataareaid= 'vsf'
    " );

   oci_execute( $stmt );

   $ax_language_list=array();

   while ( $ax_language = oci_fetch_assoc( $stmt ) ) {

      $language=new stdClass();

      $language=(object)array_change_key_case($ax_language,CASE_LOWER);

      $ax_language_list[$language->languageid]=$language->txt;

   }

   oci_free_statement($stmt);

   foreach ($this->store_language_list as $code){
```

```
        if (!isset($ax_language_list[$code])){

            $ax_language_list[$code]=$itemname;

        }

    }

    return $ax_language_list;

}

/*
Metodi:Suorittaa luokan action-ominaisuuden mukaiset toimenpiteet Magento Api:n kautta,
sekä asettaa tuotteen nimelle käännökset erikielisille kaupoille.
Vastaanottaa:Tuotteen attribuuttilista(array),tuotteen nimen mukaisten
käännösten attribuuttilistan(array).
*/
protected function SetValues($attributes,$translation_attributes){

    try{

        switch ($this->action){

            case "disable":

                $this->connections->mag_proxy->call($this->connections->mag_session,
                'product.update', array($this->itemid, array('status'=>0)));

            break;

            case "delete":

                $this->connections->mag_proxy->call($this->connections->mag_session,
                'product.delete', array($this->itemid));

            break;

            case "update":

                $this->connections->mag_proxy->call($this->connections->mag_session,
                'product.update', array($this->itemid, $attributes));

            break;

            case "create":

                $new_id=$this->connections->mag_proxy->call($this->connections->mag_session,
                'product.create', array('simple',$this->attribute_group, $this->itemid, $attributes));

            break;

        }

        if($translation_attributes!=null){

            foreach ($this->store_language_list as $wsview=>$code){

                $this->connections->mag_proxy->call($this->connections->mag_session,
                'product.update', array($this->itemid,$translation_attributes[$code],$wsview));
```

```
            }
        }
    }
    catch (Exception $e){
        $error=new CustomException($e);
    }
  }
}
```

LIITE 3 Tuoteryhmänä myytäviä tuotteita mallintava luokka

```
<?php

require_once ('product.php');

Class BomProduct extends Product{

   /*
   Metodi:Päivittää tuoteryhmän tiedot Magento Api:n välityksellä.
   */
   public function update(){

      $this->action="update";

      $this->SetBomProperties();

   }

   /*
   Metodi:Luo uuden tuoteryhmän Magento Api:n välityksellä.
   */
   public function create(){

      $this->action="create";

      $this->SetBomProperties();

   }

   /*
   Metodi:Asettaa ominaisuudet Magento Api:n kanssa yhteensopiviksi attribuuteiksi ja luo tai päivittää
   tuoteryhmän Magento Api:n välityksellä.
   */
   private function SetBomProperties(){

      $bom_items=$this->GetAxBomItems();

      $combined_bom_items=array();

      foreach($this->store_language_list as $language){

          $combined_bom_items[$language]=null;

      }

      foreach ($bom_items as $product){

          $combined_bom_items=$this->SetBomItemLanguages($product,$combined_bom_items);

      }

      $language_attributes=$this->SetTranslationAttributes($this->GetNameTranslations(
      $this->itemid,$this->itemname));

      $this->SetValues($this->SetAttributesForMagento(),
      $this->CombineLanguageAttributes($language_attributes,$combined_bom_items));

   }

   /*
```

```
Metodi:Yhdistää tuoteryhmän ja siihen kuuluvien tuotteiden erikieliset nimet Magento Api:n kanssa
yhteensopiviksi attribuuteiksi.
Vastaanottaa:Listan tuoteryhmän erikielisistä nimistä(array),listan tuoteryhmään kuuluvien
tuotteiden erikielisistä nimistä(array).
Palauttaa:Magento Api:n kanssa yhteensopivan attribuuttilistan(array).
*/
private function CombineLanguageAttributes($language_attributes,$bom_language_attributes){

    foreach ($this->store_language_list as $wsview=>$code){

        $language_attributes[$code]["bf_bom_items"]=$bom_language_attributes[$code];

    }

    return $language_attributes;

 }

/*
Metodi:Hakee tuoteryhmään kuuluvat tuotteet Axaptasta.
Palauttaa:Listan tuoteryhmän tuotteista(array).
*/
private function GetAxBomItems(){

   $stmt = oci_parse( $this->connections->ax_connection, "

      SELECT

         (b.bomqty*m.quantity) as quantity,
         i.itemid,
         i.itemname as itemname

       FROM

          bmssa.bomversion v

       INNER JOIN

          bmssa.bom b

          on b.bomid = v.bomid
          and b.dataareaid = v.dataareaid

       INNER JOIN

          bmssa.inventtable i

          on i.itemid = b.itemid
          and i.dataareaid = b.dataareaid

       INNER JOIN

          bmssa.inventtablemodule m

          on m.itemid = v.itemid
          and m.dataareaid = v.dataareaid
          and m.moduletype = 2

      WHERE

          v.itemid =:itemnumber
          and v.active = 1
```

```
            and v.approved = 1
            and i.itemtype in ( 0, 1 )
            and v.dataareaid= 'vsf'
            and v.todate = to_date( '01-01-1900','dd-mm-yyyy') OR CURRENT_DATE<=v.todate
            and v.fromdate<=CURRENT_DATE

     " );

    oci_bind_by_name($stmt,':itemnumber',$this->itemid);

    oci_execute( $stmt );

    $ax_bom_list=array();

    while ( $ax_bom = oci_fetch_assoc( $stmt ) ) {

        $bom=new stdClass();

        $bom->info=(object)array_change_key_case($ax_bom,CASE_LOWER);

        $ax_bom_list[$bom->info->itemid]=$bom;

    }

    oci_free_statement($stmt);

    return $ax_bom_list;

}

/*
Metodi:Hakee tuoteryhmän tuotteen nimen erikieliset käännökset ja yhdistää ne tuoteryhmän muiden
tuotteiden kanssa käännöksen kielen mukaisesti.
Vastaanottaa:Tuoteryhmän tuotteen(object),kielikohtaisen listan tuoteryhmän muiden tuotteiden
käännöksistä(array)
Palauttaa:Kielikohtaisen listan tuoteryhmän tuotteiden käännöksistä(array).
*/
private function SetBomItemLanguages($product,$combined_bom_items){

    $languages=$this->GetNameTranslations($product->info->itemid,$product->info->itemname);

    $language_list=array();

    foreach ($this->store_language_list as $code){

        if(!in_array($code,$language_list)){

            $language_list[]=$code;

        }

    }

    foreach ($language_list as $code){

        $combined_bom_items[$code]=$combined_bom_items[$code].$product->info->quantity.
        ' x '.$languages[$code].'<br />';

    }

    return $combined_bom_items;
```

```
    }
}
?>
```

LIITE 4 Abstrakti luokka

```
Abstract Class Object_Abstract{

    protected $action;
    protected $connections;

     /*
     Metodi:Suorittaa itsensä kun objekti instantioidaan.
     Vastaanottaa:Yhteys-objekti(object).
     Asettaa:Yhteys-objektin luokan kentäksi, josta se on kaikkien luokan metodien käytettävissä.
     */
      public function __construct($Connections) {

            $this->connections=$Connections;

       }

       /*
       Metodi:Asettaa sille annetun arvon luokan ominaisuudelle.
       Vastaanottaa:Luokan ominaisuuden nimeä vastaavan arvon(string),arvo ominaisuudelle.
       Asettaa:Annetun arvon annettua nimeä vastaavalle ominaisuudelle.
       */
      public function __set($prop,$value){

          try{

                throw new Exception('Invalid property '.$prop.' with value '.$value);

           }

           catch(Exception $e){

                $error=new CustomException($e);

           }

       }

     /*
     Metodi:Hakee annettua nimeä vastaavan ominaisuuden arvon.
     Vastaanottaa:Luokan ominaisuutta vastaava nimi (string).
     Palauttaa:Luokan ominaisuutta vastaavan nimen arvon.
     */
     public function __get($prop){

       try{

          if (array_key_exists($prop, get_object_vars($this))) {

             return $this->prop;

       }

       else{

            throw new Exception('Trying to get non-existing property :'.$prop);

      }

   }
```

```
        catch(Exception $e){
            $error=new CustomException($e);
        }
    }
}

?>
```

LIITE 5 Rajapinnat

```
<?php

   //Rajapintamääritys luokille, joita käytetään Magenton päivitykseen.
   interface iUpdater
   {

      const update_range=1; //Määrittää, monen päivän sisällä päivitetyt tiedot Axapta-kysely ottaa
      huomioon
      const first_run=true; //Ensimmäisellä suorituskerralla oltava true.

   }

   //Rajapintamääritys luokille, jotka luodaan Axaptan päivitettyjen tietojen perusteella.
   interface iObject{

         function update();
         function create();
         function delete();

    }


?>
```

LIITE 6 Asiakasryhmiä käsittelevälle moduulille määritetty luokka

```
<?php

class Bf_CustomerGroupApi_Model_ObjectModel_Api
{

   private $model;

   /*
   Metodi:Suorittaa itsensä luokan instantioituessa
   Asettaa:Magenton mallin/luokan luokan metodien käytettäväksi
   */
   public function __construct() {

      $this->model=Mage::getModel('customer/group');

   }

   /*
   Metodi:Luo asiakasryhmän
   Vastaanottaa:Ryhmän nimen (string), veroryhmän tunnuksen (int)
   Palauttaa:Ryhmätunnuksen(int)
   */
   public function create($group_name,$tax_class_id){

      $this->model->setCode($group_name)->Save();

      $this->model->setData('tax_class_id',$tax_class_id)->Save();

      return $this->GetIdByCode($group_name);

   }

   /*
   Metodi:Päivittää asiakasryhmän tiedot
   Vastaanottaa:Ryhmätunnuksen (int), ryhmän nimen (string), veroryhmän tunnuksen (int)
   */
   public function update($group_id,$group_name,$tax_class_id)
   {

   $this->model->setData(array(
                                 'customer_group_id' => $group_id,
                                 'customer_group_code' => $group_name,
                                 'tax_class_id' => $tax_class_id)
                                 )->save();

   }

   /*
   Metodi:Poistaa asiakasryhmän tiedot
   Vastaanottaa:Ryhmätunnuksen (int)
   */
   public function delete($customer_group_id){

      $this->model->load($customer_group_id)->delete();

   }

   /*
   Metodi:Hakee ryhmäkoodia vastaavan tunnuksen
```

```
    Vastaanottaa:Ryhmäkoodin (string)
    Palauttaa:Ryhmätunnuksen (int)
    */
    private function GetIdByCode($code){

        $collection=$this->model->getCollection();

        $results = array();

        foreach ($collection as $group) {

            $result = $group->toArray(array('customer_group_id', 'customer_group_code'));

            $results[$result["customer_group_code"]]=$result;

        }

        return $results[$code]["customer_group_id"];

    }

}

?>
```