Opinnäytetyö (AMK)

Tietotekniikka

Sulautetut ohjelmistot

2013

Niko Nieminen

VERKKOKAUPAN OHJELMOINTIRAJAPINTA JA WWW-SISÄLLÖNHALLINNAN TYÖKALU

TURUN AMMATTIKORKEAKOULU
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tietotekniikka | Sulautetut ohjelmistot

2013 | 53

Jari-Pekka Paalassalo

Niko Nieminen

VERKKOKAUPAN OHJELMOINTIRAJAPINTA JA WWW-SISÄLLÖNHALLINNAN TYÖKALU

Tämän opinnäytteen tarkoituksena oli valmistaa verkkokauppojen ohjelmointia helpottava rajapinta, jonka kautta voisi hakea mm. tuotteita ja tuoteryhmiä tietokannasta. Kaikki verkkokaupan sisältö olisi saatavilla monikielisenä. Sisällönhallintaa, kuten tuotteiden lisäämistä ja poistamista varten, suunniteltiin tehtäväksi erityinen työkalu.

Kaikki kokonaisuuden osat ohjelmoitiin PHP:llä ja rajapinnan tietokantana päätettiin käyttää MySQL:ää MyISAM -tietokantamoottorilla. Rajapinnan pääasiallinen luokka nimettiin Webshopiksi. Rajapintaa käytettäessä tästä luokasta luodaan olio, jonka kautta kaikkia sen toimintoja kutsutaan. Sisällönhallinnan työkalu on itsenäinen verkkosivu, jonka tietokantaa muokkaava ohjelmalogiikka ohjelmoitiin Webadmin-luokkaan. Työkaluun ohjelmoitiin sisäänkirjautumis-, loki- ja käytönrajoitustoimintoja.

Lopuksi rajapintaa ja työkalua käytettiin projektissa, joka todistaa tuotetun kokonaisuuden hyödyllisyyden ja jonka perusteella työn tuloksia arvioitiin. Rajapinta ja sisällönhallinnan työkalu sisältävät halutut ominaisuudet hakea ja muokata kaupan eri tietoja. Lisäksi tuotetun ohjelmakoodin rakenne tekee siitä helpon ylläpitää siinä sovelletun MVC-periaatteen ansiosta.

Kun rajapintaa hyödynnetään verkkokaupan ohjelmoinnissa, ohjelmoijan tarvitsee kiinnittää huomiota vain verkkosivujen ulkoasuun ja joihinkin niillä navigointiin liittyviin tekniikoihin.

Työtä voidaan laajentaa ohjelmoimalla erilaisia maksutapoja rajapintaan, tuen useammalle kuin yhdelle kuvalle tuotetta kohti ja tehtyjen tilausten seurantamoduulin sisällönhallinan työkaluun.

ASIASANAT:

PHP, MySQL, MyISAM, verkkosivut, verkkokauppa, www-sisällönhallinta, web-sisällönhallinta, WCMS

BACHELOR´S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information Technology | Embedded Software

2013 | 53

Jari-Pekka Paalassalo

Niko Nieminen

AN APPLICATION PROGRAMMING INTERFACE FOR WEB SHOPS AND A TOOL FOR WEB CONTENT MANAGEMENT

The objective of this thesis was to develop an application programming interface that would help the programming of Internet web shop applications. It was intended that through this interface it would be possible to retrieve product and product category data from the database where all the information was to be made available in multiple languages.

In addition, for web content management, a special tool was developed to allow graphical editing of shop data. Functions such as login, usage restriction, and action logging are built into this tool.

All the components in the project were programmed in PHP. The interface uses MySQL as its database and MyISAM as its database engine. The interface can be used by including the central class named Webshop and then creating an instance of it. Through this Webshop object, one can execute all of the functions of the interface that are required to run a web shop.

The content management tool is a standalone web application. Its program logic that handles modifying the database was separated inside the class known as Webadmin.

Finally, as proof of concept, both the interface and the content management tool were used in a real life project. The example web shop was also utilized in the evaluation of the success of the whole project. Both the interface and content management tool have an easy-to-read-and-maintain program structure that mimics the MVC design principle.

When programming web shops that make use of the interface, one does not need to focus on other things than visuals and some techniques regarding page navigation.

The project can be further developed by programming additional payment options to the Webshop class, support for multiple images per product and a module for order handling to the content management tool.

KEYWORDS:

PHP, MySQL, MyISAM, web page, web shop, www content management, web content management, WCMS

SISÄLTÖ

KUVAT

SANASTO

BLOB	Binary Large Object. Yksi mahdollinen tyyppi binääri-datalle MySQL-tietokannassa. Sopii esim. tiedostojen tallettamiseen.
MVC	Model View Controller -ohjelmointimalli. Erottaa näkymän (käyttöliittymän) mallista (esim. tietokannan tiedosta) ohjelmarakenteessa.
Tietokantamoottori	Tietokantamoottori on se osa tietokannan hallintajärjestelmästä, joka määrittelee, miten tietoa käsitellään. Se vaikuttaa mm. siihen, miten tietoa lisätään, luetaan, muokataan tai poistetaan.
WCMS	Web Content Management System. Web-sisällönhallinta järjestelmä, jolla voidaan hallita jonkin sivuston sisältöä.
XSS	Cross Site Scripting -haavoittuvuus. Mahdollistaa haitallisen tiedon esittämisen tai ohjelmakoodin suorittamisen haavoittuvan verkkosivun kautta.

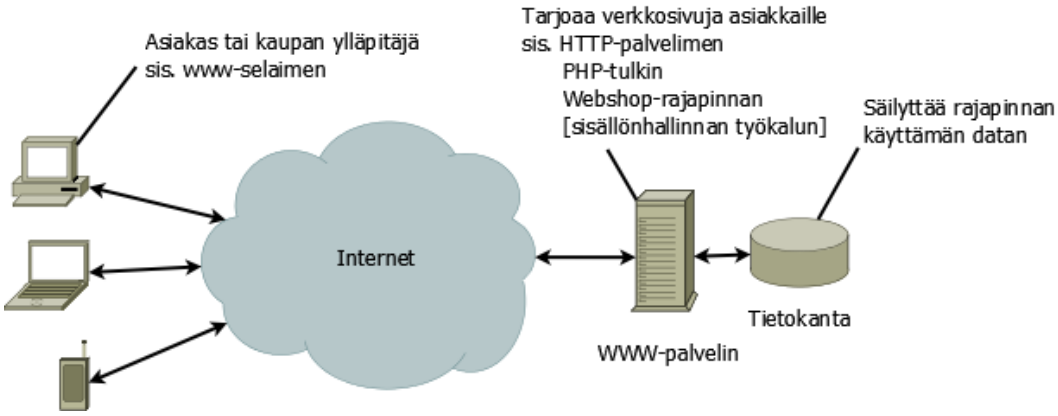
1 JOHDANTO

Tämän työn tarkoituksena on valmistaa ohjelmointikirjasto, joka helpottaisi verkkokauppojen ohjelmoimista PHP:llä. Lopputuloksen halutaan olevan eräänlainen ohjelmointirajapinta verkkokaupan rakentamiseen. Kirjaston käyttämälle tietokannalle kehitetään www-sisällönhallintatyökalu, jolla voidaan lisätä ja muokata tietoja. Lisäksi kyseisellä kirjastolla toteutetaan verkkokauppa Iitut Oy:lle, joka toimii esimerkkitapauksena kirjaston ja työkalun käytöstä.

Ohjelmointikirjastossa, www-sisällönhallintatyökalussa ja verkkokaupassa ei tulla käyttämään mitään kolmannen osapuolen alustoja tai kirjastoja JQueryä lukuunottamatta, jota saatetaan käyttää joissakin esimerkkisivun graafisissa elementeissä.

2 TEKNIIKKA

Työn rajapinta koostuu monesta PHP-luokasta. Sen asennus tapahtuu kopioimalla sen tiedostot verkkokaupan tarjoavalle WWW-palvelimelle ja sisällyttämällä yksi, rajapinnan käyttöönottava tiedosto verkkokaupan pääasiallisessa lähdekoodissa. Myös sisällönhallinnan työkalu, jolla muokataan verkkokaupan tarjoamaa sisältöä, sijaitsee WWW-palvelimella. Kumpaakin käytetään selaimen välityksellä. Sisällönhallinnan työkalu, verkkokaupan lähdekoodi rajapintoineen sekä tietokanta voivat kaikki sijaita eri palvelimilla, mutta tässä työssä ne sijaitsevat kaikki yhdellä palvelimella. (Kuva 1)



Kuva 1 Yleiskuva rajapinnasta ja sisällönhallinnan työkalusta. Sekä verkkokauppaa että sisällönhallinnan työkalua käytetään selaimen kautta. Tuotettu ohjelmistokokonaisuus sijaitsee kokonaan WWW-palvelimella.

PHP on yleisesti käytetty ohjelmointikieli [1] www-ohjelmoinnissa. PHP-koodi suoritetaan palvelimella ennen asiakkaalle lähettämistä ja sitä voidaan kirjoittaa HTML-koodin kanssa lomittain [2]. PHP:n käyttöön liittyy kuitenkin huomionarvoisia vaaroja. Jotkut ovat sitä mieltä, että PHP:n kehityksessä ei olla oltu johdonmukaisia, mikä näkyy mm. siinä, että jotkin toisiaan muistuttavat funktiot ottavat parametrinsa eri järjestyksessä tai niiden nimeäminen on epäjohdonmukaista. [3] Ongelmaa ei auta, että PHP sisältää valtavan määrän näitä sisäänrakennettuja, joidenkin mielestä toisiaan toistavia, funktioita. Tämänkaltaiset yksinkertaiset poikkeamat lisäävät riskiä epähuomiossa

tehtyihin virheisiin palveluita kehitettäessä [4] ja joskus johtavat jopa tietoturva-aukkojen syntymiseen. Tästä huolimatta PHP on monipuolinen ohjelmointikieli, ja kun sen kanssa työskenneltäessä kiinnitetään huomiota hyviin ohjelmointitapoihin, se on myös erittäin käyttökelpoinen.

MySQL-tietokanta on suosittu nimenomaan www-palveluissa [5] ja sitä käytetään rajapinnan tietokantana. Tietokantamoottorina voitaisiin käyttää joko MyISAMia tai InnoDB:tä. InnoDB on moderneissa MySQL-versioissa vakioarvoisesti käytettävä moottori. MyISAM on kuitenkin yhä tuettu ja ollut kauemmin käytössä. Vanhoissa MySQL:n versioissa InnoDB saattaa joidenkin lähteiden mukaan toimia hitaammin verrattuna MyISAMiin. Jotkin lähteet suosittelevat ehdottomasti säätämään MySQL-palvelimen vakioarvoisia InnoDB-asetuksia suorituskyvyn parantamiseksi. Joillekin verkkosivujen ylläpitäjille tämä ei ole mahdollista (esim. verkkoisännöintipalveluissa). MyISAM valittiin varmemman yhteensopivuuden ja yksinkertaisemman MySQL:n pystyttämisen takia. Näin rajapinnan tietokanta voidaan sijoittaa laajempaan määrään olemassa olevia palvelimia ja sen käyttömahdollisuudet ovat paremmat. InnoDB:n ja MyISAMin käytön välillä on joitain huomionarvoisia eroja, joten niiden välillä vaihtaminen ei ole täysin triviaalia. [6][7]

3 WEBSHOP RAJAPINTA

Alkuperäinen visio rajapinnan toiminnasta on se, että kaikki verkkokaupan toiminnot voidaan hoitaa yhden PHP-luokan kautta. Luokasta luodaan olio, ja kaikkia verkkokaupan toimintoja kutsutaan seuraavasti:

```
$ws = new Webshop();

$something = $ws->getSomething();

$ws->doSomething();
```

Toimintoihin kuuluisi esim. ostoskorin ylläpitäminen, verkkokaupan tuotteiden hakeminen tietokannasta ja tilauksen suorittaminen. Tietokantaan ei tarvitsisi kiinnittää huomiota, koska uusien tuotteiden lisäämistä varten olisi olemassa oma graafinen työkalunsa. Käyttäjän tarvitsisi tietää vain PHP:n ja olio-ohjelmoinnin perusteet pystyäkseen rakentamaan verkkokaupan. Se voisi ulkonäöllisesti olla juuri sellainen, kuin sen halutaan olevan, koska rajapinta ei pakottaisi mihinkään valmiiseen muottiin.

Rajapinta-luokka nimettiin Webshopiksi. Toteutusta ja sen suunnittelua voidaan tarkastella seuraavassa järjestyksessä: tarvittavien toimintojen kartoitus, tietokannan suunnittelu vastaamaan haluttuja toimintoja, tietorakenteiden suunnittelu tietokannan pohjalta, rajapinnan rakenteen suunnittelu ja eri toimintojen lisääminen siihen.

Webshop, ohjelmointikirjasto ja rajapinta ovat tässä opinnäytetyössä synonyymejä toisilleen.

3.1 Tarvittavat ominaisuudet

Erittelemällä tuotteiden ostamiseen tarvittavia välivaiheita, voidaan selvittää mitä ominaisuuksia rajapinnan on vähintään tarjottava. Kaupassa käyminen voidaan jakaa kahteen vaiheeseen: ostosten tekemiseen ja oston suorittamiseen. Ostokset tehdään seuraavasti:

- Asiakas tarkastelee saatavilla olevia tuoteryhmiä.
- Asiakas tarkastelee määrättyä tuoteryhmää.
- Asiakas tarkastelee määrättyä tuotetta edellisestä tuoteryhmästä.
- Asiakas kerää tuotteen ostoskoriin.

Rajapinnan avulla on siis voitava esittää kaikki tuoteryhmät, sitten määrätyn tuoteryhmän kaikki tuotteet ja lopuksi tuote. Tuotteita on pystyttävä lisäämään käyttäjän ostoskoriin ja ostosten täytyy säilyä siellä sivulatausten välillä.

Oston suorittaminen voitaisiin määritellä näin:

- Asiakas tarkastelee ostoskoriaan ja tekee lopullisen ostopäätöksen kunkin tuotteen suhteen.
- Asiakas maksaa ostoksensa.

Rajapinnan on tarjottava mahdollisuus tarkastella ostoskoriin lisättyjä tuotteita ja pystyttävä poistamaan niitä sieltä. Mutta maksun suorittaminen on monitahoista. Asiakkaan on täytettävä yhteystietolomake siitä, mihin ostokset lähetetään ja kuka ne maksaa. Tämän takia oston suorittaminen on paremmin mallinnettuna seuraavanlainen:

- Asiakas tarkastelee ostoskoriaan ja tekee lopullisen ostopäätöksen kunkin tuotteen suhteen.
- Asiakas täyttää yhteystietolomakkeen, jonka kaupan omistaja on laatinut.
- Kaupan omistaja käsittelee tilauksen annettujen tietojen pohjalta.

Rajapinnan on siis pystyttävä tallentamaan yhteystietoja ja muita tietoja, joita verkkokaupan ylläpitäjä haluaa kerättävän ostojen yhteydessä. Näitä tietoja on

pystyttävä hyödyntämään laskutuksessa ja tuotteiden toimituksessa, eli kaupan ylläpitäjän on pystyttävä tarkastelemaan niitä. (Kuva 2)

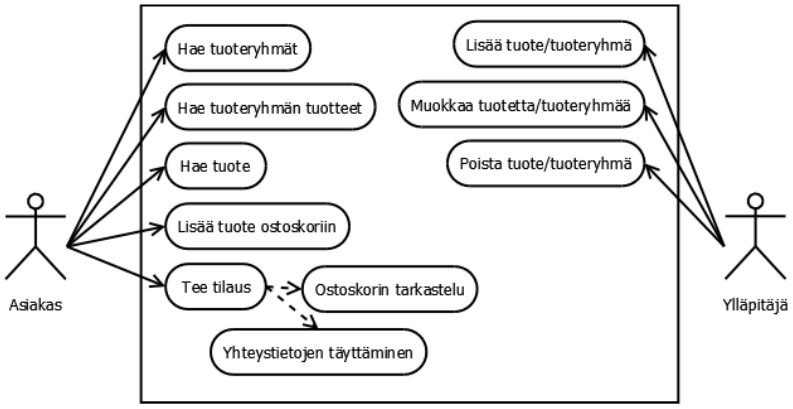
Resurssit, joita toteutuksessa aivan ilmeisesti täytyy hyödyntää ovat tietokanta ja istuntovaihtujat. Tietokantaa käytetään luonnollisesti tuotetietojen tallettamiseen ja istuntovaihtujia ostoskorin ja muun tilatiedon tallentamiseen. Rajapinnan on tarkoitus ns. abstraktoida [8] näiden resurssien käyttö, eli rajapintaa käyttävän ohjelmoijan ei ole tarkoitus päästä käsittelemään niitä suoraan. Jos tietokannan rakennetta syystä tai toisesta muutetaan, joudutaan muuttamaan vain rajapintaa, eikä ohjelmakoodia, joka on siitä riippuvainen. Lisäksi istuntovaihtujien suora käyttö voi suurissa projekteissa tehdä ohjelmalogiikan sekavaksi, samalla tavalla kuin globaalit vaihtujat muissa ohjelmointikielissä [9].

Rajapinnalle asetetaan seuraavat rajoitukset:

- Ainoa tässä työssä toteutettu maksutapa on postiennakko, mutta maksutapoja on modulaarisesti voitava lisätä jälkeenpäin.
- Tuoteryhmät ovat toistaiseksi hierarkiattomia (ei sisäkkäisiä tuoteryhmiä).

ja seuraavat lisäominaisuudet:

- Verkkokaupan kaikki sisältö on oltava saatavissa niin monella kielellä, kuin kaupan ylläpitäjä haluaa.
- Tuotteilla on oltava kuvat.



Kuva 2 Verkkokaupan toiminnot. Verkkokauppaa käyttävä asiakas tarkastelee ja ylläpitäjä muokkaa verkkokaupan tietoja

3.2 Tietokannan rakenne

Tietokannassa on oltava taulukot ainakin tuoteryhmille ja tuotteille. (Kuva 3)

Tuotekuvat voitaisiin tallettaa palvelimen levylle, jolloin vain tiedostopolut olisivat tietokannassa, mutta järjestelmän yhtenäisyyden kannalta ne nähtiin hyväksi sijoittaa tietokantaan. Rajapinnan data on näin helppo varmuuskopioida ja palauttaa tarvittaessa. Binääridatan (BLOB) hakeminen tietokannasta on joissain tapauksissa hitaampaa kuin levyjärjestelmästä, mutta on vastaavasti olemassa tutkimuksia, jotka osoittavat, että pienten tiedostojen (<1 Mt) hakeminen tietokannasta on nopeampaa [10]. Punnittaessa vaihtoehtoja tultiin siihen tulokseen, että ylläpidollisten toimien helpottuminen on tässä tapauksessa tärkeämpää, kuin vähäiset vaikutukset hakunopeuteen, koska kuvien voidaan olettaa olevan verkkokaupoissa usein suhteellisen pieniä, eivätkä kymmenien megatavujen suuruisia. Kuten myöhemmin tullaan osoittamaan, rajapinnan ei tarvitse palauttaa montaa kuvaa yhdellä kertaa, ja yksittäisten kyselyjen nopeus ei nouse pullonkaulaksi. Jos useiden kuvien data haettaisiin yhdellä haulla, asiat voisivat olla toisin. Kuvat voitaisiin tallettaa myös yhdessä tuotteiden kanssa tuote-taulukkoon, mutta tällöin jouduttaisiin päättämään ennalta, kuinka monta kuvaa tuotteella voi enintään olla. Kun kuvilla on oma taulukkonsa, niitä voi olla kuinka monta tahansa yhtä tuotetta kohti.

Käännöksillä eli erikielisillä mutta samaa asiaa tarkoittavilla lauseilla tai sanoilla on oltava oma taulukkonsa. Käännökset päätettiin jakaa selvyyden vuoksi kolmeen eri taulukkoon: tuote-, tuotekategoria- ja yleiskäännös taulukoihin. Yleiskäännös-taulukossa on kaikki se tekstisisältö, joka ei kuuluu tuotteisiin tai tuoteryhmiin. Näihin kuuluvat esim. ohjetekstit, käyttöliittymässä esiintyvien painikkeiden tekstit, kuten 'Takaisin'-painike, 'Päivitä'-painike tai mitkä tahansa muut näiden kaltaiset käännöksen tarvitsevat elementit. Käännöksissä käytettäville kielille luotiin oma taulukkonsa.

Kunkin taulukon perusavaimeksi asetettiin juokseva kokonaisluku, ellei seuraavaksi toisin määritellä.

Tuotteet talletetaan taulukkoon nimeltä 'products'. Taulukko sisältää vierasavaimen tuotekategoriaan ja tuotteen hinnan, joka on jaettu euroihin ja sentteihin liukuluku-laskuvirheiden välttämiseksi [11].

Tuoteryhmä-taulukolla 'categories' on vain yksi sarake, joka on sen perusavain. Tämä saattaa vaikuttaa erikoiselta, mutta näin käännöksiä voi olla ennalta määrittelemättömällä määrällä kieliä, kuten tuotetaulukolla.

Kielille luotiin oma taulukkonsa. On totta, että pelkästään kielen nimen normalisoiminen on turhaa, mutta jos kieliin päätetäisiin myöhemmin liittää lisätietoja, on hyvä että ne ovat valmiiksi omassa taulukossaan. Tämä myös helpottaa uusien kielien lisäämistä ja niiden hallinta. Kielten taulukon nimi on 'languages'.

Tuote- ja tuoteryhmä-taulukoilla on kummallakin oma käännöstaulukkonsa, joiden nimet ovat 'product_trans' ja 'category_trans', tässä järjestyksessä. Nämä sisältävät erikielisiä käännöksiä tuotteiden ja tuoteryhmien nimille ja kuvauksille. Nimi on lyhyt ja kuvaava otsikko ja kuvaus on valinnainen, lisätietoja antava pitempi teksti.

Yleiskäännösten taulukko 'general_trans' eroaa muista käännöstaulukoista siten, että sen perusavain on merkkijonon ja kielen vierasavaimen yhdistelmä, eikä juokseva kokonaisluku. Toisin kuin esimerkiksi tuotekäännösten perusavainten, näiden perusavainten on oltava kuvaavia, jotta rajapinnan kautta voidaan helposti hakea eri käännöksiä.

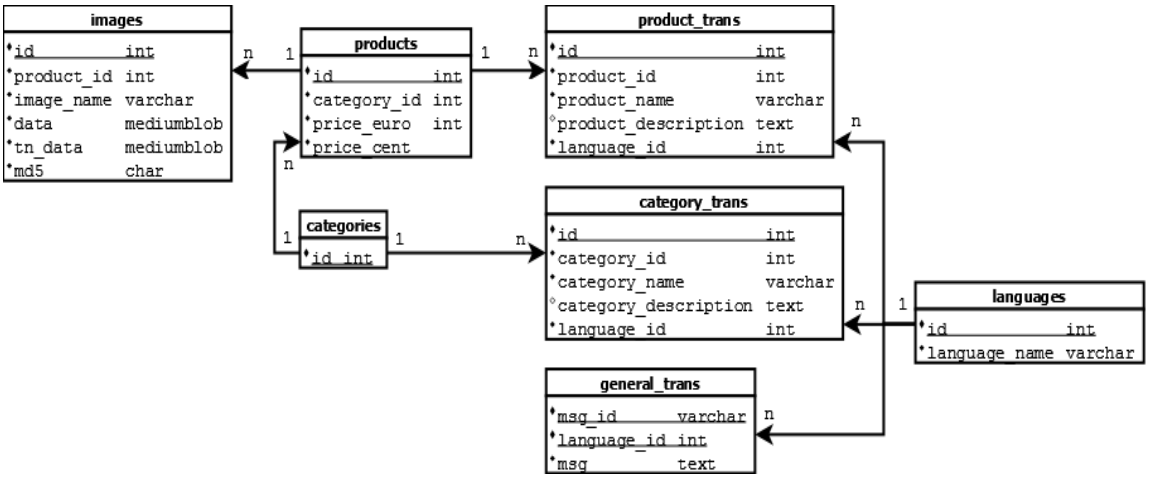
Koodiesimerkki 1

```
$ws->translate('PRICE');
```

on kuvaavampi ja helpompi ymmärtää kuin

```
/* Käännös sanalle 'hinta' */
$ws->translate(195);
```

Kuvien taulukossa 'images', on sekä kuvan että sen mahdollisen pienoiskuvan data BLOB-muodossa. Taulukossa on myös kuvan kuvan nimi, sen tuotteen vierasavain, johon kuva kuuluu ja kuvan MD5-tarkiste, jota tullaan käyttämään ns. HTTP-ETagissä, joka liittyy kuvien säilyttämiseen selaimen välimuistissa.



Kuva 3 Tietokannan rakenne

3.3 Rajapinnan rakenne

Rajapinnan tarjoaman tiedon on oltava riippumaton tietokannan rakenteesta. Muuten muutokset tietokannassa muodostavat ongelmia yhteensopivuuden kanssa rajapintaa hyödyntävässä koodissa. Paras ratkaisu tähän on tietenkin hyödyntää olio-ohjelmointiperiaatetta. Jokaista tietokannasta löytyvää monimutkaista rakennetta kohden luodaan oma luokkansa, jolla on omat tiedon hakumetodinsa. Tämä on paljon parempi vaihtoehto, kuin palauttaa rajapinnalta epämääräisiä vektoreita, joiden solujen järjestys tai avaimet olisi tarkoin dokumentoitava.

Koodiesimerkki 2

```
$p = $ws->getProduct(1);
echo $p->getName().' '.$p->getDesc();
```

on paljon ylläpidettävämpi ja modulaarisempi, kuin

```
$p = $ws->getProduct(1);
echo $p['name'].' '.$p['desc'];
```

Tietorakenteiden luokkiin voidaan myös ohjelmoida tietoturvan kannalta oleellisia toimintoja, kuten erillisiä HTML-turvallisia hakumetodeja, jotka poistavat tai korvaavat merkkejä, jotka saattaisivat mahdollistaa XSS-hyökkäyksiä. Varsinkin PHP:n tapauksessa, jonka tyypittömyys voi joskus aiheuttaa arvaamattomia tilanteita, tämä vähentää riskiä vahingossa avautuviin tietoturva-aukkoihin. Voidaan sanoa, että tietoturva-aukkoja syntyy vääjäämättä ja eräällä tavalla kyse onkin todennäköisyyksistä: mitä vähemmän ohjelmakoodia kirjoitetaan, sitä vähemmän huolimattomuusvirheitä voi syntyä [12][13][14][15]. Kun kaikki ongelmakohdat ovat samassa paikassa (tiedostossa tai luokassa), niiden suojaaminen on yksinkertaisempaa ja todennäköisyys virheisiin on pienempi, koska koodi on luettavampaa. Lisäksi kun ongelmakohtiin voidaan näin kiinnittää huomiota rajapinnan sisällä, sen avulla suunnitellut verkkokaupat ovat tietenkin varmemmin turvallisia.

Koodiesimerkki 3

```
echo htmlspecialchars($p['name']);
```

sijaan voidaan käyttää

```
echo $p->getWebSafeName();
```

Tietokannan pohjalta suunniteltiin kolme luokkaa: Product (tuote), Image (kuva) ja Category (tuoteryhmä). Product-luokkaa suunniteltaessa huomattiin tarpeelliseksi luoda myös Price-luokka (hinta), koska tietokannassa eurot ja sentit ovat toisistaan erillään ja siksi hintojen laskutoimituksiin tarvitaan yksinkertaisia vektorilaskuja.

Tietokantayhteyttä varten luotiin DBSocket-luokka, josta periytettiin WebshopDBSocket-luokka. WebshopDBSocket-luokkaan ohjelmoidaan verkkokaupan kyselymetodeja tarpeen mukaan. DBSocket tarjoaa sen perivälle luokalle vain yksinkertaisimman mahdollisen alustan kyselyjen suorittamiseen tiettyyn tietokantamoottoriin. Tämä lisää modulaarisuutta siten, että DBSocket voidaan ohjelmoida käyttämään eri tietokantaa kuin MySQL muuttamatta sitä perivää luokkaa sillä oletuksella, että kyselykielen syntaksi pysyy samana. Käytännössä, yleensä myös kyselyjä joudutaan kuitenkin hieman muuttamaan. Kun kaikki kyselyt ovat samassa paikassa, niiden ylläpitäminen on huomattavasti helpompaa. Kuten tietorakenteidenkin tapauksessa, tässäkin mahdollisuus tietoturva-aukkojen syntymiseen on pienempi, kun kaikki kyselyt tehdään yhden luokan kautta. WebshopDBSocket abstraktoi tietokannan rakenteen muodostamalla kyselyjen tulosten pohjalta tietorakenneluokkien tyyppisiä olioita.

Rajapinnan on tarkoitus tarjota ostoskorisovelluksen ominaisuus. Tämän toteutus tapahtuu luonnollisesti istuntovaihtujilla. PHP:ssä istunnot toimivat siten, että palvelin generoi uniikin istuntotunnisteen kullekin palvelimen asiakkaalle. Tunniste lähetetään asiakkaalle ja jokaisen sivulatauksen yhteydessä asiakas lähettää tunnisteen takaisin palvelimelle evästeenä [16]. Ohjelmakoodissa istuntovaihtujia pääsee käsittelemään $_SESSION-globaalin kautta. $_SESSION on vektori kaikista istuntovaihtujista. istuntoglobaalin käyttöön liittyy samoja ongelmia, joita yleensäkin liittyy globaalien vaihtujien käyttämiseen muissa ohjelmointikielissä. Se voitaisiin luokitella jaetuksi resurssiksi siinä missä tietokantakin, ja siksi nähtiin sopivaksi luoda sen ympärille eräänlainen ajuriluokka. Rajapinnan minkään muun osan ei ole tarkoitus olla kosketuksissa siihen suoraan. Luokan nimeksi valittiin SessionData. SessionData luodaan kerran ja sen jälkeen se serialisoidaan $_SESSION-globaaliin. Jokaisen sivulatauksen yhteydessä SessionData ladataan $_SESSION-globaalista.

SessionData- ja WebshopDBSocket-luokkia ei ole tarkoitus käyttää rajapinnan ulkopuolella. Se luokka, jonka kanssa verkkokauppaa ohjelmoiva on

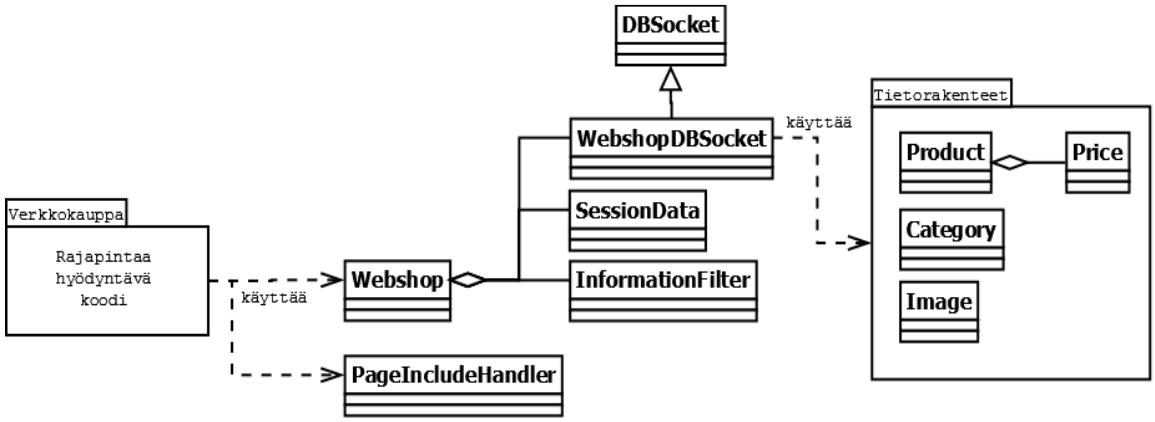
tekemisissä on Webshop (Kuva 4). Rajapintaa on tarkoitus pystyä käyttämään esimerkiksi seuraavasti:

Koodiesimerkki 4

```
$ws = new Webshop();
/* Hae tuote nro 1 (tietokanta) */
$p = $ws->getProduct(1);
/* Aseta tuote ostoskoriin ($_SESSION) */
$ws->addToCart($p->getProductId());
```

Rajapintaa käyttäessä ei tarvitse käyttää tietokantaa tai istuntovaihtujia. (Koodiesimerkki 4)

Tietokannasta saadaan mm. kokonaislukuja ja merkkijonoja, mutta rajapintaa hyödyntävän verkkokaupan koodi saa käsitellä ylemmän tason tietorakenteita kuten 'tuotteita' ja 'tuoteryhmiä'.



Kuva 4 Rajapinnan rakenne. Kaikkia rajapinnan toimintoja käytetään Webshop-luokan kautta, joka käyttää WebshopDBSockettia ja SessionDataa tietokannan ja istuntovaihtujien hallitsemiseen. Verkkokaupan koodi käsittelee tietokannan tietoa tietorakenneluokkiin enkapsuloituna.

3.4 Rajapinnan toiminnot

Verkkokaupan on ensiksi pystyttävä tarjoamaan asiakkaalle eri tuoteryhmiä läpikäytäväksi. Tuoteryhmien hakemiseen rajapintaan valmistettiin getCategories- ja getCategory-metodit. Ensimmäinen metodeista palauttaa vektorin Category-luokan olioita, toinen palauttaa vain yhden tuoteryhmän, joka vastaa siihen parametriksi annetun tuoteryhmän tunnistetta eli tietokannassa olevaa perusavainta. Metodeja varten WebshopDBSocket-luokkaan luotiin queryCategories- ja queryCategory-metodit, jotka hakevat tuoteryhmät tietokannasta ja luovat niistä Category-luokan olioita.

Koodiesimerkki 5

```
$ws = new Webshop();
$categories = $ws->getCategories();
foreach ($categories as $c)
        echo $c->getWSCategoryName();
```

Koodiesimerkki tulostaa jokaisen järjestelmässä olevan tuoteryhmän nimen. (Koodiesimerkki 5)

Tuotteiden hakemista varten tehtiin getProduct-, getProducts-, getProductPage- ja getNumberOfProductPages-metodit. Ensimmäinen metodeista hakee vain yhden tuotteen, joka vastaa sille parametriksi annettua tuotetunnistetta. Toinen hakee kaikki määrätyn tuoteryhmän tuotteet. Kolmas hakee yhden sivun tuotteita määrätystä tuoteryhmästä: tuotteita harvoin halutaan kaikkia yhdellä kertaa, joten niiden sivutus rakennettiin osaksi rajapintaa. Sitä, kuinka monta tuotetta sivua kohden haetaan, voidaan ohjata globaalin asetustiedoston avulla. Neljäs mainituista metodeista palauttaa kokonaisluvun, joka kertoo kuinka monta sivua tuotteita määrätyssä tuoteryhmässä on yhteensä. Tätä voidaan hyödyntää tuotteita esittelevän sivun sivunumerolinkeissä. Metodien tarvitsemat kyselymetodit ohjelmoitiin WebshopDBSocket-luokkaan.

Koodiesimerkki 6

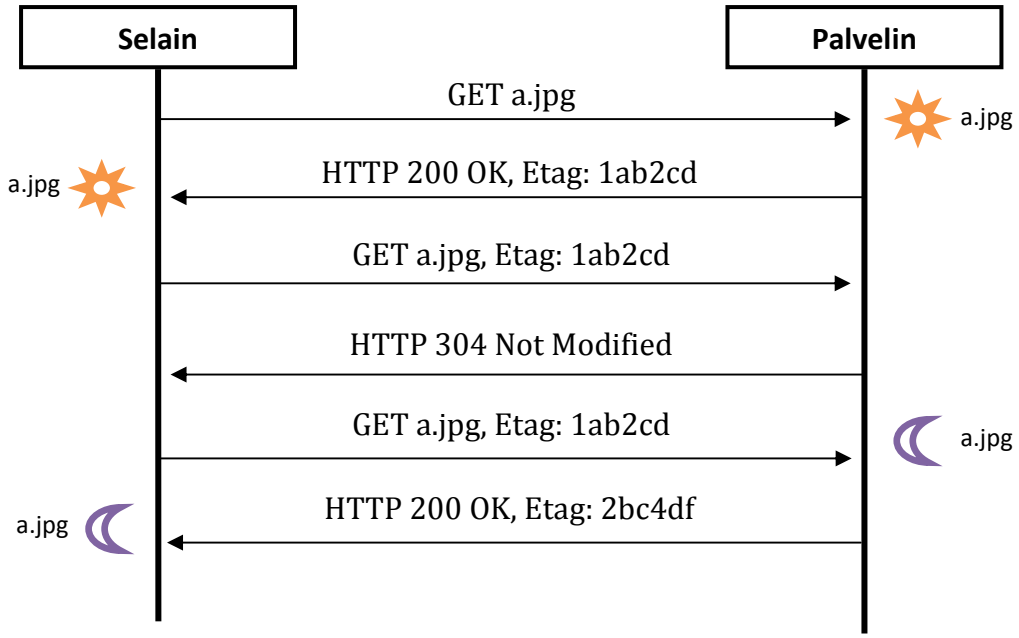
```
$ws = new Webshop();
$products = $ws->getProductPage($_GET['category'], $_GET['page']);
foreach ($products as $p)
            echo $p->getWSProductName();
```

Koodiesimerkki tulostaa pyydetyn tuoteryhmän ja sivun tuotteiden nimet. (Koodiesimerkki 6)

Tuotekuvia varten ohjelmoitiin getProductImageId-metodi, jonka avulla voi hakea määrätyn tuotteen kuvan tunnisteen (perusavain tietokannasta). Lisäksi ohjelmoitiin getImage- ja getImageThumbnail-metodit, joilla voidaan hakea itse kuva datoineen. Tietokantaan tallennetaan myös kuvien MD5-tarkisteet ja näitä varten tehtiin getImageMD5-metodi. Kuvan esittämistä varten luotiin dumpImage-metodi, jota käytetään kuvan datan lähettämisessä selaimeen HTTP-protokollan mukaisesti. dumpImage käyttää kaikkia edellä mainittuja kuviin liittyviä metodeja.

Selaimet ovat jo hyvän aikaa tukeneet kuvien ja muiden resurssien tallentamista välimuistiin. Paikallisen välimuistin lisäksi, välimuisteja voi myös olla välityspalvelimilla. Jos resurssi, jota sivuilla käytetään on jo ladattu välimuistiin, sitä ei ladata verkkopalvelimelta, jolla on resurssin alkuperäiskopio. Jos resurssi muuttuu alkuperäispalvelimella, se ladataan uudelleen. Se tekniikka, joka HTTP-protokollassa tämän mahdollistaa on ns. Etag. Etag on palvelimen resurssille antama tunniste, joka lähetetään kuvan mukana selaimelle. Kun selain pyytää resurssia palvelimelta toistamiseen, se lähettää myös sen aikaisemman Etagin. Jos Etag on sama kuin palvelimen resurssin Etag, palvelin lähettää HTTP-304 "ei muuttunut" -vastauksen. Muutoin resurssi lähetetään uudelleen uudella ETagilla varustettuna. (Kuva 5) On selvää, että Etagin siirtäminen verkon yli on paljon kevyempää kuin kuvan siirtäminen, ja tämän tekniikan hyödyntäminen vähentää verkon kuormitusta joissain tapauksissa huomattavasti. dumpImage hyödyntää tietokantaan tallennettua MD5-tarkistetta ETagin valmistamisessa ja mahdollistaa sen, että selain voi sijoittaa tuotekuvat välimuistiin. Kun on mahdollista, se lähettää HTTP-304 "ei muuttunut" -

vastauksen selaimelle. dumpImage-metodia on tarkoitus käyttää erillisessä php-tiedostossa, joka asetetaan kuvan lähteeksi HTML-koodissa. [17]



Kuva 5 Etag. Etagin ansiosta kuvaa ei tarvitse aina ladata uudelleen palvelimelta, vaan ainoastaan, kun kuva muuttuu.

Koodiesimerkki 7

kuva1.php:

```
$ws = new Webshop();
$ws->dumpImage(1); // kuva nro 1
```

index.html:

```
<img src="kuva1.php" />
```

Kun selain tulkitsee index.html-tiedostoa, se lataa kuvan siinä ilmoitetusta lähteestä. Kun kuva1.php-tiedostoa ollaan lähettämässä selaimelle, sen PHP-koodi tulkitaan ja suoritetaan palvelimella. Kuva haetaan tietokannasta ja lähetetään selaimelle. dumpImage lähettää myös kuvan Etagin, jonka takia selain voi asettaa kuvan välimuistiinsa. Jos kuvia on useita samalla sivulla, ne voidaan ladata palvelimelta rinnakkain. (Koodiesimerkki 7)

dumpImage-metodi lähettää HTTP-otsakkeita selaimelle. Otsakkeita muodostaessa on oltava varovainen. Jos esimerkiksi lähetetään otsake, jossa kerrotaan kuvan nimi, ja nimi saadaan tietokannasta, on varmistuttava, että nimessä ei ole rivinvaihtoja. Muuten kuvan nimen kautta mahdollistuu ns. otsakeinjektio. Jos 'hyökkääjän' oletetaan pääsevän käsiksi tietokantaan, hän voisi pahimmassa tapauksessa ohjata liikennettä omille sivuilleen huijaustarkoituksessa tai mahdollisesti ladatakseen haittaohjelmia asiakkaiden laitteille. Uusimmissa PHP:n versioissa tämä heikkous on suojattu jo PHP:n oman sisäisen koodikirjaston puolella [18], mutta asia on silti mainitsemisen arvoinen: otsakkeissa käytettävä data on puhdistettava ylimääräisistä rivinvaihdoista.

Ostoskorin muokkaamista varten ohjelmoitiin metodeja, joista oleellisimmat ovat: addToCart, removeFromCart ja getProductsFromCart. Niiden avulla voidaan lisätä ja poistaa tuotteita ostoskorista tai hakea kaikki siinä olevat tuotteet. Istuntovaihtujiin tallennetaan vain tuotteiden tunnisteet, ja kun getProductsFromCart metodia kutsutaan, niitä vastaavat tuotteet haetaan uudelleen tietokannasta. Tämä on yksi niistä mekanismeista, jotka varmistavat, että ostoskoriin ei voi vahingossa lisätä olemattomia tuotteita, mutta suurin saavutettu hyöty on nopeudessa: istuntovaihtujat pysyvät mahdollisimman pieninä. Ostoskorin hinnan hakemista varten ohjelmoitiin getPriceOfCart-metodi, joka laskee ostoskorin tuotteiden hinnat yhteen.

Ennen tuotteiden tilaamista on käyttäjältä kerättävä yhteystietoja. Riippuu paljon verkkokauppaa ylläpitävästä yrityksestä ja käytettävästä maksutavasta, mitä yhteystietoja asiakkaalta tarvitaan. Tämän ominaisuuden on oltava siis mahdollisimman modulaarinen. Tietoja ei voi kuitenkaan vain hyväksyä, kun ne saadaan verkkokauppaa käyttävältä asiakkaalta, vaan ne täytyy todeta oikein muotoilluiksi. Esim. suomalainen postinumero koostuu viidestä numerosta.

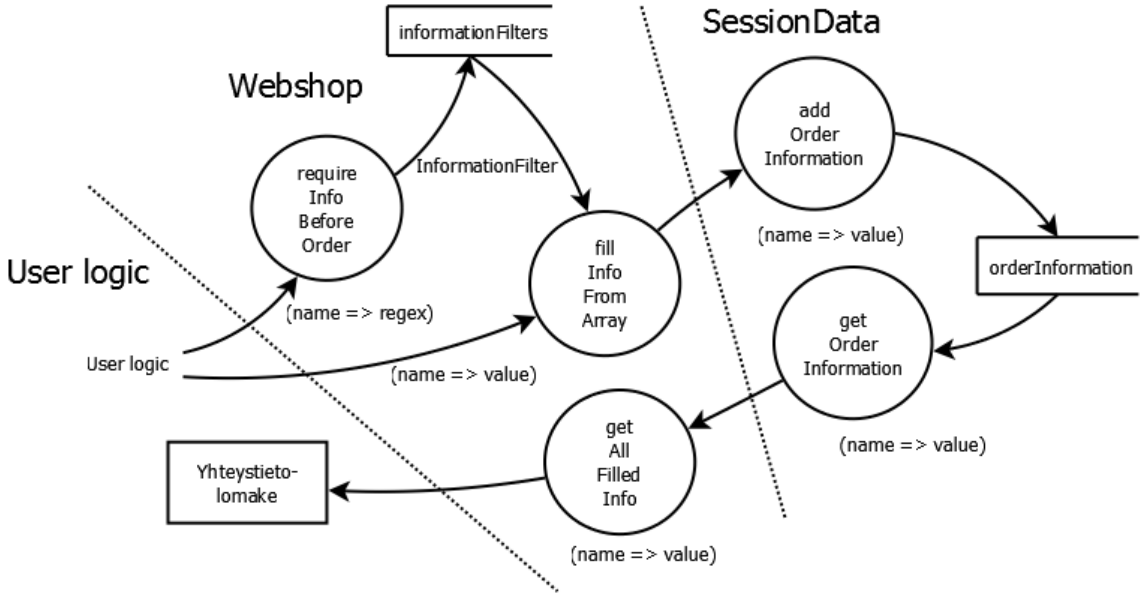
Muotoilun tarkistamisen avuksi luottiin uusi luokka: InformationFilter eli tiedon suodatin. Tiedon suodatin voidaan luoda nimestä ja säännöllisestä lausekkeesta (engl. "regular expression" tai "regex"). Tämän jälkeen sille

voidaan yrittää antaa jokin arvo. Jos arvo täyttää säännöllisen lausekkeen ehdot, se asetetaan suodattimen arvoksi.

Tietojen tallentamista varten luotiin metodeja SessionData-luokkaan, mutta tiedon oikeellisuuden tarkistaminen tehdään Webshop-luokassa. Webshop-luokkaan lisättiin metodi requireInfoBeforeOrder, joka ottaa parametrikseen nimen ja sännöllisen lausekkeen. Näistä se luo uuden tiedon suodattimen, jonka se tallentaa sisäiseen vektoriinsa. fillInfoFromArray -metodilla voidaan yrittää syöttää tietoja suodattimiin. Metodi ottaa parametrikseen vektorin, joka voi olla esim. PHP:n globaali $_POST-vektori, jolla yleensä vastaanotetaan lomaketietoja. Vektorin avaimen nimisiin suodattimiin yritetään syöttää avaimia vastaavia arvoja. Jos suodatin pysyy tyhjänä, se tarkoittaa, että arvo oli väärin muotoiltu. Metodi palauttaa tyhjänä pysyneiden suodattimien nimet. getAllFilledInfo-metodilla saadaan haettua kaikki tiedot, jotka täyttivät annetut vaatimukset (säännölliset lausekkeet). Rajapinta ei kuitenkaan pakota täyttämään kaikkia suodattimia: tällä tavoin se antaa mahdollisuuden valinnaisten tietokenttien tekemiseen. (Kuva 6)

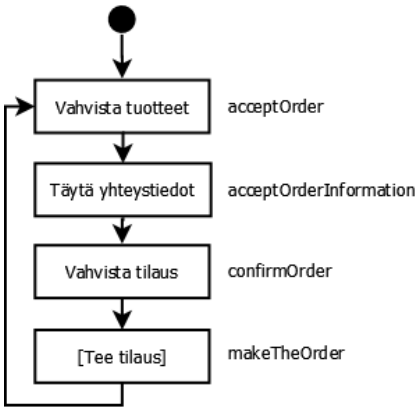
Koodiesimerkki 8

```
$ws = new Webshop();
$ws->requireInfoBeforeOrder('etunimi', '^[a-z]*$');
$errors = $ws->fillInfoFromArray($_POST);
$info = $ws->getAllFilledInfo();
/* Näytä seuraavaksi virheet
 * tai käsittele täytetyt tiedot */
```



Kuva 6 InformationFilterin käyttö. Rajapinnalle kerrotaan mitä tietoja halutaan käyttäjältä. Tämän jälkeen sille voidaan antaa vektori, joka sisältää käyttäjän täyttämän lomakkeen tietoja (esim. $_POST-globaali). Rajapinta vertaa tietoja saamiinsa ehtoihin. Jos arvot täyttävät annetut ehdot, se tallentaa ne.

Ostosten tilaaminen on itse asiassa tilakone. (Kuva 7) Ensin asiakkaalle yleensä näytetään ostoskorin tuotteet, sitten yhteystietolomake, sitten varmistetaan halutaanko tuotteet todella tilata ja varmistetaan ovatko annetut tiedot todella oikeat. Rajapintaan luotiin tällainen tilakone, jota noudattamalla voidaan varmistua, että kaikki tarpeellinen on tehty ennen tilauksen tekemistä. Tilakone toimii varmistimena, niin että tilausta ei tehdä vahingossa liian aikaisin. Sitä voidaan ja tulisi myös hyödyntää ostoskorin eri tilojen toteuttamisessa. Metodit ja niiden sallittu kutsumisjärjestys ovat seuraavat: acceptOrder, acceptOrderInformation, confirmOrder. Tiloissa päästään taaksepäin goBackInOrder-metodin avulla. Kun confirmOrder-metodia on kutsuttu, tilakone sallii tilauksen tekemisen rajapinnan avulla. Nykyinen tila saadaan getOrderState-metodilla, joka palauttaa kokonaisluvun, joka vastaa jotakin Webshop-luokassa määritellyistä tilavakioista.



Kuva 7 Oston suorittaminen. Ennen oston lopullista suorittamista rajapinta vaatii tilakoneen läpikäymisen, joka varmistaa, että kaikki tarpeellinen on tehty

Tuotteiden tilauksen suunniteltiin tapahtuvan siten, että sekä kaupan ylläpitäjälle että tuotteiden tilaajalle lähetetään tilausvahvistus sähköpostilla. Kaupan ylläpitäjän vastuulle jää, miten hän tilaukset hoitaa. Kun vahvistus on lähtenyt kummallekin osapuolelle, rajapinta ei enää ota osaa tapahtumien kulkuun. Tilausmetodi makeTheOrder palauttaa virhearvon, jos tilausvahvistuksia ei voitu lähettää. Vahvistukset lähetetään niin, että ensimmäisenä yritetään lähettää vahvistusta kaupan ylläpitäjälle ja vasta sen jälkeen ostosten tekijälle. Rajapintaa käyttävän verkkokaupan tulee ilmoittaa onnistuneesta tai epäonnistuneesta tilauksesta kaupan asiakkaalle. Tilausmetodi ottaa parametriksi asiakkaan sähköpostiosoitteen, joka yleensä saadaan yhteystietojen keräyksen yhteydessä, ja tilausvahvistuksen HTML-koodin. HTML-koodista korvataan erityisiä avainsanoja vastaanotetuilla tilaustiedoilla. Jos makeTheOrder huomaa joidenkin avainsanojen jääneen tyhjäksi se palauttaa virhekoodin, eikä lähetä tilausta.

Koodiesimerkki 9

```
$ws = new Webshop();
$ws->requireInfoBeforeOrder('sposti', '^[a-z_-\.\@]*$');
$ws->requireInfoBeforeOrder('nimi', '^[a-z]*$');
$ws->fillInfoFromArray($_POST);
$info = $ws->getAllFilledInfo();
$html = "Hei! !nimi!. Kiitos tilauksesta";
$ws->makeTheOrder($info['sposti'], $html);
```

Koodiesimerkissä lähetetyssä tilausvahvistuksessa avainsana !nimi! korvataan vastaanotetun nimen kanssa. Avainsana on korostettu esimerkissä hahmottamisen helpottamiseksi. (Koodiesimerkki 9)

Pakolliset avainsanat ovat !ORDER_FEE! ja !PRICE_SUM!, joilla määritellään mihin toimituskulut ja tilauksen yhteishinta tulee laittaa. Edellinen esimerkki ei siis lähettäisi tilausta, koska nämä avainsanat puuttuvat HTML-asemoinnista. Kuvaillulla tekniikalla annetaan vapaus verkkokaupan rakentajalle muotoilla tilausvahvistus.

Uusia maksutapoja voi kehittää luomalla makeTheOrderin kaltaisia metodeja rajapintaan. On oletettavaa että useimmilla maksutavoilla on tapauskohtaisia yhteystietojen keräämiseen liittyviä riippuvaisuuksia, jotka täytyy ottaa niissä huomioon. Postiennakossa tarvitaan vain tilaajan yhteystiedot ja sähköpostiosoite, jonne tilausvarmistus lähetetään.

Kaikilla kolmella käsitellyllä pääluokalla on setLanguage-metodi. WebshopDBSocketissa se asettaa millä kielellä tulokset kaikista sen kautta tehtävistä kyselyistä annetaan. Se vaikuttaa siis suoraan sivuston sisällön kieleen. SessionDatassa se tallentaa halutun kielen istuontovaihtujiin. Webshopin setLanguage-metodi käyttää kumpaakin edelläkuivailluista avukseen. Kun Webshop luodaan sivulatausten välillä uudelleen, se jatkaa SessionDataan tallennetun kielen käyttämistä WebshopDBSocketissa.

Koodiesimerkki 10

Sivu 1 ladataan

```
$ws = new Webshop();
$ws->setLanguage(2);
```

Sivu 2 ladataan

```
$ws = new Webshop();
echo $ws->getLanguage(); // tulostaa 2
```

Koodiesimerkki tulostaa 2. sivun latauksen yhteydessä "2", koska kieli pysyy sivulatausten välillä SessionData-luokan ansiosta. Sivulla 2 haetut tuoteryhmät tai muut tiedot olisivat kielellä, jonka tunniste on 2. (Koodiesimerkki 10)

Käännösten hakemista varten luotiin translate ja getTranslation metodit. Ensimmäinen tulostaa ja toinen palauttaa haetun käännöksen. Kumpikin ottaa parametriksi käännöksen tunnisteen (merkkijono), jonka vastinetta haetaan tietokannasta.

Koodiesimerkki 11

```
$ws = new Webshop();
$ws->translate('LANG_PRICE');
```

Tulostetun merkkijonon kieli riippuu koodiesimerkissä rajapinnan sen hetkisestä kielestä. (Koodiesimerkki 11)

Useimpiin tietorakenteisiin ohjelmoitiin tapauskohtaisia metodeja, jotka jotenkin auttavat niiden tietojen esittämisessä. Esimerkiksi Image-luokkaan tehtiin getMime-metodi, joka päättelee kuvan MIME-tyypin ja Price-luokkaan erilaisia laskutoimintoja. Ne hakumetodit, jotka palauttavat merkkijonon, ja joita on tarkoitus nimenomaisesti käyttää tietojen esittämiseen HTML-ohjelmakoodin seassa, päätettiin nimetä getWS-alkuisina, joka on lyhenne getWebSafesta. Näissä metodeissa on huomioitu mahdolliset XSS-haavoittuvuudet ja pyritty suojautumaan niiltä. Yleensäkin hakumetodeissa on kiinnitetty huomiota

vaihtujien tyyppeihin. Hakumetodit tyypittävät oletetut kokonaisluvut kokonaisluvuiksi jne.

Webshop-rajapinnan käyttö aloitetaan luomalla Webshop-olio. Rajapinnalla on oma asetustiedostonsa "settings". Tätä kautta voidaan vaikuttaa moniin verkkokaupan eri piirteisiin, kuten kuinka monta tuotetta kullekin tuotesivulle halutaan, minkä kielen pitäisi olla vakioarvoisesti valittuna ja mitä käyttäjätunnuksia tietokantayhteydessä tulisi käyttää.

4 WWW-SISÄLLÖNHALLINTATYÖKALU

Webshop-rajapinnan on tarkoitus abstraktoida tiedonhaku tietokannasta, kun taas suunnitellun www-sisällönhallintatyökalun on tarkoitus abstraktoida tiedon syöttäminen tietokantaan. Työkalu toteutetaan PHP-verkkosivuna siten, että sen ohjelmalogiikan rakenne muistuttaa Webshop-rajapinnan rakennetta. Rajapintaa käyttävän ei ole tarkoitus ohjelmoida työkalua uudelleen omaan käyttöönsä, vaan hyödyntää sitä verkkokaupan pystyttämisessä ja ylläpitämisessä. Tästä huolimatta ohjelmalogiikka rakennetaan erilleen sen graafisesta ulkoasusta, jotta sen ylläpitäminen olisi mahdollisimman helppoa.

Tässä asiakirjassa työkalulla tarkoitetaan aina www-sisällönhallintatyökalua, joka koostuu Webadmin-luokasta ja sitä hyödyntävästä käyttöliittymästä.

Rajapintaa käyttävän ei ole tarkoitus käsitellä tietokantaa suoraan, vaan käyttää työkalua. Sillä on pystyttävä lisäämään, muokkaamaan ja poistamaan Webshop-rajapinnan tarjoamia tietoja, jotka ovat: tuoteryhmät ja niiden käännökset, tuotteet ja niiden käännökset, sekä kielet ja yleisluontoiset käännökset. Lisäksi huomattiin tarve joillekin lisäominaisuuksille:

- Työkalua ei saa päästä käyttämään ketään ulkopuolinen, joten siihen on ohjelmoitava mekanismi sisäänkirjaantumista ja käyttäjän todentamista varten.
- Pääsyä työkalun joihinkin toimintoihin on pystyttävä rajoittamaan ja tätä varten tarvitaan käyttöoikeuksien hallintaa.
- Työkalua on pystyttävä käyttämään usealla työasemalla samanaikaisesti.
- Sitä, mitä työkalulla tehdään on pystyttävä seuraamaan.

4.1 Työkalun rakenne

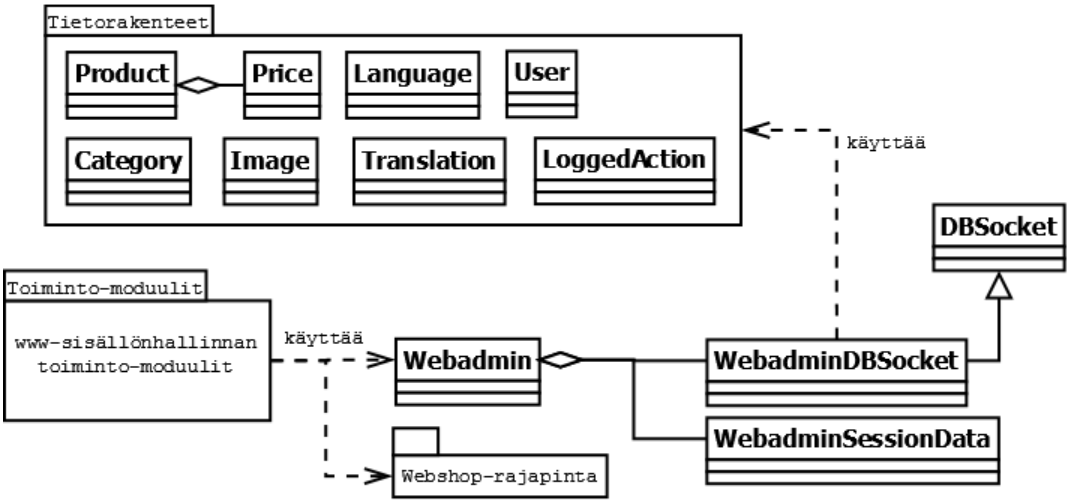
Työkalua varten ohjelmoitiin Webadmin-luokka, joka käyttää hyödykseen WebadminDBSocket- ja WebadminSessionData-luokkia. Työkalun on tarkoitus hyödyntää Webshop-rajapinnan tiedonhaku-ominaisuuksia tietojen

hakemisessa, silloin kun ei ole perusteltua ohjelmoida erillisiä hakumetodeja Webadminiin.

WebadminDBSocket periytettiin aiemmin luodusta DBSocket-luokasta. Luokan on tarkoitus ottaa huomioon se, miten tietokannan tiedon eheys säilytetään. Esim. tuoteryhmää poistettaessa, ottaa huomioon mitä tapahtuu sen sisältämille tuotteille. Metodien on tarkoitus abstraktoida Webadmin-luokalta tiedon eheyteen liittyvät asiat tarjoamalla toimintoja kuten 'lisää tuote' tai 'poista tuoteryhmä'.

WebadminSessionDatan on tarkoitus tallentaa käyttäjätietoja istuntovaihtujiin, jotta Webadmin-luokalla voidaan toteuttaa sisäänkirjautumismekanismi.

Webadmin-luokka käyttää hyödykseen ylläkuvailtuja luokkia mahdollistaessaan käyttäjän todentamisen, suoritettujen toimintojen seurannan ja, jos käyttäjän oikeudet eivät riitä määrätyn toiminnon suorittamiseen, toimintojen suorittamisen rajoittamisen. (Kuva 8)



Kuva 8 WWW-sisällönhallintatyökalun rakenne

4.2 Tietokannan rakenne

Jotta sisäänkirjautuminen olisi mahdollista, täytyy tietokantaan luoda taulukko käyttäjätietoja varten. Salasanojen tallentamisessa on luonnollisesti oltava huolellinen, eikä niitä tulisi koskaan pitää selkokielisenä tietokannassa [19].

Tietokantaan talletetaan vain salasanoista luotuja tarkistearvoja. Tarkistearvolla tarkoitetaan tässä jostakin arvosta (A) jollakin algoritmilla saatua arvoa (B), jota ei voida purkaa alkuperäiseksi arvoksi (A). Esim. SHA2-algoritmit ovat tarkoitukseen hyviä vaihtoehtoja, joista yhtä työkalussa päätettiinkin käyttää, mutta se voidaan helposti vaihtaa mihin tahansa tarkisteen luomiseen tarkoitettuun algoritmiin. Koska SHA2-algoritmit on suunniteltu suorittumaan nopeasti, tarkisteen luomista täytyy hidastaa esim. kutsumalla sitä useita kertoja peräkkäisesti. Tämä hidastaa tarkisteiden arvausyrityksiä siltä varalta, että tietokanta joutuisi hyökkääjien käsiin.

Tietoturvan kannalta on hyvä, että salasanoihin liitetään ns. suola ennen tarkistearvon luomista [19]. Suola on salasanaan liitettävä satunnainen arvo, joka estää esilaskettujen tarkistekirjastojen käyttämisen salasanojen tarkisteiden purkamisessa. Se tallennetaan suolatun salasanan tarkistearvon kanssa tietokantaan. Esimerkiksi hyökkääjä on saattanut laskea valmiiksi tai hankkia lukemattomia määriä tarkisteita yleisistä salasanoista nopeuttaakseen niiden tarkisteiden purkamista. Kun salasanat suolataan, niiden tarkisteet ovat täysin erilaisia kuin valmiiksi lasketuissa tarkisteissa. Jokaista suolaa varten olisi oletetussa hyökkäyksessä laskettava oma tarkistekirjasto.

Tietokantaan luotiin näiden tietojen pohjalta taulukko nimeltä users. (Kuva 9) Taulukon perusavain on juokseva kokonaisluku ja siinä on sarakkeet käyttäjän nimeä, salasanan tarkistetta ja sen suolaa varten. Lisäksi luotiin kokonaislukusarake käyttöoikeuksien tallentamista varten. Käyttöoikeuksille ei luotu omaa taulukkoa, vaan Webadmin määrää mitä kullakin oikeudella voi tehdä.

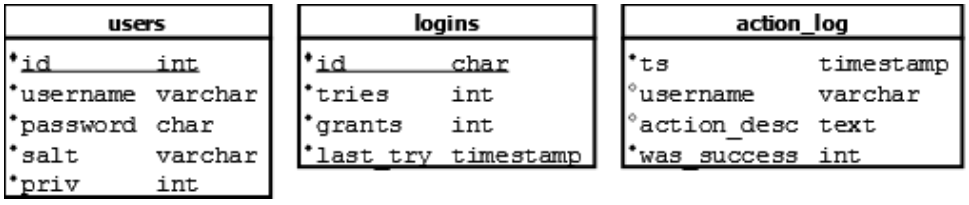
Tarkistearvon luominen ja salasanan suolaaminen suojelee sellaisissa tapauksissa, joissa hyökkääjä pääsee käsiksi tietokantaan. Se ei suojaa yrityksiltä arvata salasanoja kirjautumisvalikon kautta. Tällaisia yrityksiä voidaan parhaiten vaikeuttaa aikalukoilla tai viiveillä. Kun määrätystä osoitteesta on pyritty kirjautumaan sisään epäonnistuneesti tarpeeksi monta kertaa, sen yritykset estetään vähäksi aikaa. Jos yrittäminen jatkuu tarpeeksi pitkään, yritykset estetään kokonaan.

Tätä mekanismia varten luotiin 'logins'-taulukko. (Kuva 9) Sen perusavaimeksi, joka on merkkijono, luodaan tarkiste jostain tiedosta, jonka avulla sisäänkirjautumista yrittävä taho voidaan tunnistaa (esim. IP-osoite). Tietoa ei tallenneta tietokantaan selkokielisenä samasta syystä kuin salasanoja säilytettäessä. Taulukossa on lisäksi sarake

- sisäänkirjautumisyritysten määrälle ('tries')
- järjestelmän aiheuttamien aikalukkojen määrälle ('grants')
- aikaleimalle viime yrityksestä ('last_try').

Kun järjestelmä päättää, että yrityksiä on tapahtunut liian paljon, se aiheuttaa aikaviiveen tai -lukon siten, että sisäänkirjautumista ei yritetä laisinkaan. Kun aikaviive tai -lukko on rauennut se kasvattaa grants-sarakkeen arvoa ja nollaa yritysten määrän tries-sarakkeesta. Jos järjestelmä päättää, että se on viivästyttänyt kirjautumista tarpeeksi monta kertaa, se estää kirjautumisen kokonaan.

Suoritettujen toimintojen seuraamista varten suunniteltiin action_log-taulukko. (Kuva 9) Taulukossa ei ole perusavainta, koska sen arvoja ei ole tarkoitus muokata eikä poistaa osittaisesti. Jos se tyhjennetään, se tyhjennetään kokonaan. Sarakkeita siinä on aikaleimaa, toiminnon suorittaneen käyttäjän nimeä, toiminnon vapaamuotoista kuvausta ja palautuskoodia (totuusarvo: 1 tai 0) varten.



Kuva 9 Tietokantaan www-sisällönhallintatyökalua varten lisätyt taulukot

Joissain tapauksissa taulukot (MyISAM tietokantamoottorissa) täytyy lukita, ennen kuin niitä voi muokata turvallisesti [20]. Työkalun oletetaan olevan käytössä samanaikaisesti usealla eri työasemalla, joten tieto, joka toiminnon alussa oli olemassa ei välttämättä enää ole toiminnon lopussa. Taulukoiden

lukitseminen otetaan huomioon WebadminDBSocket-luokassa. Esimerkiksi, kun uusi käyttäjä luodaan tietokantaan, on ensin tarkistettava, onko haluttu käyttäjänimi vapaa. Jos käyttäjänimi on vapaa, tunnukset voidaan luoda. Käyttäjä-taulukko on lukittava ettei tämä prosessi tapahdu useassa paikassa samanaikaisesti, jolloin tietokantaan voisi päätyä samannimisiä käyttäjiä. Vastaavia tilanteita voisi syntyä suuressa osaa tietokantaan tehtäviä muutoksia, ellei taulukoita lukittaisi.

4.3 Tietorakenteet

Tietokantaan luodut uudet taulukot osoittivat tarpeen uusille tietorakenne-luokille. Näiden pohjalta luotiin mm. User- ja LoggedAction-luokat. Lisäksi Webadminin toimintoja kehitettäessä nähtiin hyväksi luoda Translation- ja Language-luokat. Näiden luokkien tarkoitus on helpottaa järjestelmän dokumentointia ja ylläpitoa enkapsuloimalla tieto johdonmukaisiksi ja intuitiivisiksi olioiksi, sen sijaan että metodeille annettaisiin epämääräisiä ja monimutkaisia vektoreita.

Tietokantaa muokkaavat toiminnot ottavat yleensä vastaan tietorakenneluokiksi ryhmiteltyä tietoa. Esimerkiksi tuotteiden muokkaus ottaa vastaan tuote-olion.

4.4 Toiminnot

Webadminiin suunniteltiin ns. nonce-järjestelmä. Jokaista sen kautta suoritettavaa toimintoa varten on pyydettävä vuoronumero eli nonce (engl. "number used only once") [21]. Vuoronumero upotetaan toimintoja aloittavaan HTML-lomakkeeseen ja annetaan seuraavan sivun latauksen yhteydessä toimintoa suoritettaessa. Tämä estää saman toiminnon suorittamisen vahingossa useaan kertaan käytettäessä selaimen paluunäppäintä tai sivua päivitettäessä. Lisäksi kun HTTP-palvelin konfiguroidaan käyttämään SSL-salausta, tämä vaikeuttaa ns. replay-hyökkäyksen tekemistä, jossa salatut viestit kaapataan ja lähetetään uudelleen palvelimelle. Vuoronumeroa

hyödynnetään kaikissa metodeissa, jotka muokkaavat tietokantaa. Nonce-järjestelmään kuuluu getNewNonce-, getNonce- ja isCorrectNonce-metodit. Ensimmäinen luo uuden vuoronumeron järjestelmään ja palauttaa sen, toinen palauttaa vanhan vuoronumeron ja kolmannella voidaan tarkistaa, onko jokin vastaanotettu vuoronumero se, mikä viimeksi annettiin.

Koodiesimerkki 12

Test.php:

```
<?php
$wa = new Webadmin();
$nonce = $_POST['nonce'];
if (isset($_POST['poista'])){
            $id = $_POST['id'];
            $nonce = $_POST['nonce'];
            $wa->deleteProduct($id, $nonce);
}
$nonce = $wa->getNewNonce();
?>
<form action="test.php" method="post">
  <input type="hidden" name="nonce" value="<?php echo $nonce ?>">
  <input type="text" name="id" value="" />
  <input type="submit" name="poista" />
</form>
```

Koodiesimerkissä jokaisen sivulatauksen jälkeen Webadmin luo uuden vuoronumeron, ja aikaisemmin haettu vuoronumero ei enää toimi. Uusi vuoronumero täytyy hakea getNewNonce-metodilla ja välittää lomakkeen kautta seuraavaan sivulataukseen, jossa se on käytettävä toiminnon suorittamiseen, kuten esimerkin deleteProduct-metodissa. (Koodiesimerkki 12)

Toimintojen suorittamisen seurantaa varten ohjelmoitiin logAction-metodi Webadminiin, joka käyttää insertActionToLog-metodia WebadminDBSocketista. Tätä metodia kutsutaan jokaisesta Webadminin toiminnosta, jonka suorittumisesta halutaan jäävän merkintöjä. Metodilla voidaan tallentaa milloin, kenen toimesta ja mitä on tapahtunut järjestelmän kautta. logActionia ei ole tarkoitus käyttää Webadminin ulkopuolella, vaan merkinnät tehdään automaattisesti toimintoja suoritettaessa.

Sisäänkirjautumista varten Webadminiin luotiin login, logout ja isLoggedIn-metodit. Metodien nimet ovat varsin kuvaavia ja toimivat oletetulla tavalla kirjaten sisään, kirjaten ulos ja tarkistaen, onko käyttäjä kirjautunut sisään (palauttaa totuusarvon). Login-metodi vaatii parametrikseen käyttäjänimen ja salasanan lisäksi vuoronumeron. Jos sisäänkirjautuminen epäonnistuu, se kirjataan järjestelmään. Aikalukon aktivoitumista, sen keston pituutta ja lopullisen sisäänkirjautumiseston ehtoa voidaan säätää globaalin asetustiedoston avulla. Jos aikalukko aktivoituu, metodi palauttaa Webadmin::DENY_TEMPORARY- tai Webadmin::DENY_PERMANENT-virhevakion. Aikalukkojärjestelmä ei vaadi ohjelmoijalta muuta ulkopuolista kontaktia. Jos annetuilla tunnuksilla ei ole oikeutta järjestelmään, metodi palauttaa Webadmin::DENY_NO_RIGHTS-vakion. Kyseiset metodit hyödyntävät WebadminDBSockettiin ja WebadminSessionDataan luotuja metodeja. Istuntovaihtujiin talletetaan se, onko käyttäjä kirjautunut sisään, ja tietokannasta haettu käyttäjän nimi sekä oikeudet. Käyttöliittymän ohjelmoinnin helpottamiseksi Webadmin tarjoaa getPrivileges-metodin, joka palauttaa kirjautuneen käyttäjän oikeudet. Oikeuksia voidaan verrata Webadmin-luokan oikeus-vakioihin: täydet oikeudet (Webadmin::PRIV_ALL_RIGHTS) ja sisällön muokkausoikeudet (Webadmin::PRIV_CONTENT_RIGHTS). Sisällön muokkausoikeuksille on tarkoitus sallia vain tuotteiden ja tuoteryhmien hallinta.

Yllämainittujen metodien jälkeen Webadminiin luotiin erilaisia toimintoja, joita käytetään keskenään hyvin samantapaisesti. Kukin 'create'- ja 'edit'-metodi ottaa parametriksi olion, jonka perusteella se luo tai muokkaa tietoa tietokantannassa. Jos kohde on monikielinen, parametriksi annetaan yhden olion sijaan vektori olioita, joissa sisältö on erikielisenä. Monikielistä tietoa sisältävissä luokissa on setLanguageMeta-metodi, jolla voidaan merkitä, millä kielellä sisältö on. 'Create'-metodit eivät kuitenkaan käytä olioiden tunnistetta perusavaimena tietokannassa, vaan luovat uusia tietueita, joilla on uudet tunnisteet. 'Edit'-metodit eivät luo uutta tietoa, vaan päivittävät vanhaa, joka vastaa oliosta saatavaa tunnistetta. 'Delete'-metodit ottavat parametriksi vektorin tunnisteita, joita vastaavat tiedot poistetaan. Tämän kaavan mukaisesti toimivia metodeja ovat esim. createProduct, editProduct, deleteProducts,

createCategory, editCategory, deleteCategories, editTranslation, deleteTranslations, createUser, editUser, deleteUsers, createLanguage ja editLanguage. Kaavasta poikkeavat metodit ovat createTranslation, joka antaa käyttäjän määritellä perusavaimen, jota käytetään uudessa kännöksessä, ja deleteLanguage, joka vektorin sijaan ottaa vastaan vain yhden tunnisteen kerrallaan, eikä siten salli useiden kielien poistamista samanaikaisesti.

Jos tuoteryhmä poistetaan, kaikki sen tuotteet siirretään automaattisesti 0-ryhmään. Tämä ryhmä on ns. piiloryhmä, jonka tuotteiden ei ole tarkoitus näkyä verkkokaupan kautta. Ne voidaan myöhemmin siirtää työkalun avulla toiseen ryhmään. Jos työkalulla halutaan väliaikaisesti poistaa tuote kaupasta, se voidaan siirtää 0-ryhmään.

Kun kieli poistetaan, kaikki sille kielelle tehty sisältö poistuu sen mukana. Tästä syystä tämä voi olla vaarallinen toimenpide, ja kieliä voi poistaa vain yhden kerrallaan.

Muista toiminnoista poikkeavat metodit ovat getActionLog, getActionLogSize ja clearActionLog, joiden avulla voidaan hallita ja tarkastella toimintojen seurannan lokikirjaa.

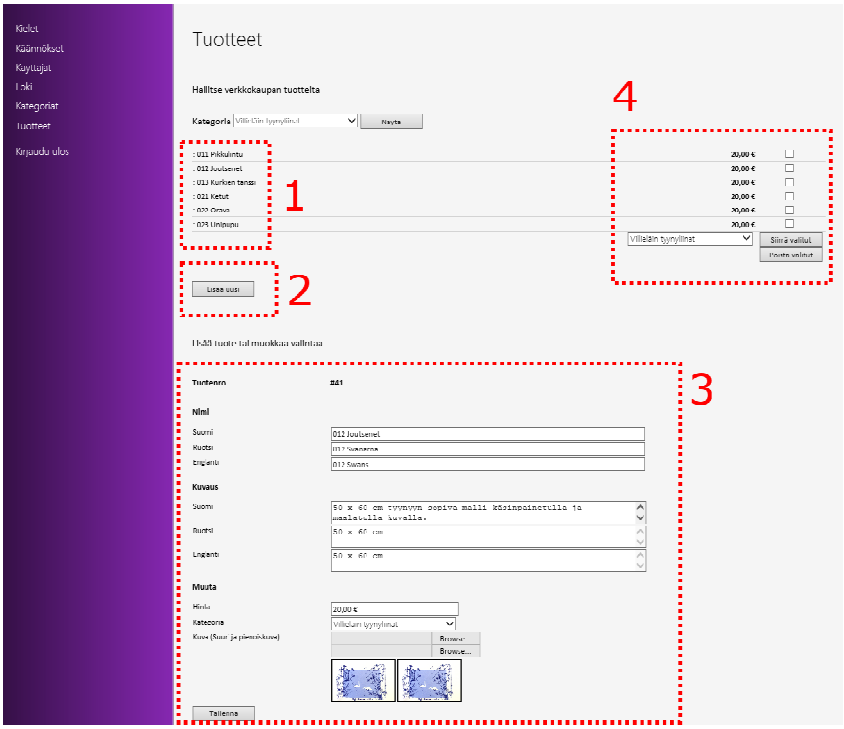
Kielten, käännösten, toiminnon seurannan ja käyttäjien hallintaan päätettiin vaatia täydet oikeudet. Muiden toimintojen suorittaminen onnistuu myös sisällön muokkaus oikeuksilla.

4.5 Ulkonäkö

Työkalun käyttöliittymä koostuu kahdesta alueesta: linkeistä toimintoihin ja sisältöalueesta. Linkit ovat selaimen vasemmalla puolella ja sisältö aukeaa sivun oikealle puolelle.

Sisältö on ohjelmoitaessa jaettu ns. toiminto-moduuleihin, eli jokaisella toiminnolla on oma tiedostonsa, joka sisällytetään pohjana toimivaan lähdekoodiin (PHP-kielen 'include'-funktiolla). Toiminto-moduulit muistuttavat ulkonäöllisesti toisiaan varsin paljon. Moduuli alkaa otsikolla, jota seuraa lyhyt

kuvaus siitä, mitä moduulilla voi tehdä. Seuraavaksi esitetään taulukko olemassa olevista, moduulilla hallittavista kohteista, kuten tuoteryhmistä. Näiden vieressä on valintaruutuja. Taulukon jälkeen on 'poista'-painke, jolla valitut kohteet voidaan poistaa. Jos taulukon kohteita painetaan, ne avautuvat moduulin alalaidassa olevalle lomakkeelle muokattaviksi. Lomakkeen yläpuolella on painike uuden kohteen luomiseksi. Jokaisen onnistuneen toiminnon jälkeen näytetään sivun ylälaidassa ilmoitus tapahtuneesta. Myös oleellisista virheistä annetaan kuvaava ilmoitus. (Kuva 10)



Kuva 10 Toiminto-moduulien ulkonäkö. Toiminto-moduulit muistuttavat ulkonäöllisesti toisiaan. Kun kohteiden linkkejä (1) painaa ne aukeavat muokkaus lomakkeelle (3). Uusia kohteita voi lisätä painamalla 'lisää'-painiketta (2). Kohteita voi poistaa monta kerrallaan (4).

Tuote-moduulissa on se lisäominaisuus, että kohteita voi myös nopeasti siirtää tuoteryhmästä toiseen suurissa erissä. Tässä moduulissa 'poista'-painikkeen yläpuolella on 'siirrä'-painke, jolla valitut tuotteet voidaan siirtää määrättyyn tuoteryhmään. (Kuva 10)

Vain yhden tuoteryhmän tuotteet näytetään kerrallaan ja tuoteryhmää voi vaihtaa pudotusvalikon avulla. 0-ryhmä eli tuotteet, jotka ovat piilossa, on pudotusvalikossa ensimmäisenä.

Tuotteelle voi antaa eri pienoiskuvan kuin täysikokoinen kuva on. Jos sille annetaan vain täysikokoinen kuva, sitä käytetään myös pienoiskuvana. Jos tuotetta muokattaessa ei anneta uutta kuvaa, vanha kuva säilytetään.

Työkalussa on monikielisiä kohteita muokattaessa tai lisättäessä kenttä jokaista järjestelmässä olevaa kieltä kohden. Kieliä voi poistaa avaamalla ne ensin muokattavaksi ja painamalla muokkauslomakkeen 'poista'-painiketta. Kuten Webadminin ohjelmalogiikkaa käsiteltäessä selitettiin, kieliä ei voi poistaa montaa yhdelläkertaa turvallisuussyistä.

5 ESIMERKKI KÄYTÖSTÄ

Rajapinta asennetaan purkamalla se pakatusta hakemistostaan verkkosivujen hakemiston juureen. Juuresta löytyy tämän jälkeen 'Includes'-hakemisto, jossa on 'settings.php'-tiedosto, jolla voidaan vaikuttaa rajapinnan toimintaan. Ohjelmakoodissa, jossa rajapintaa halutaan käyttää, suoritetaan komennot include('includes/settings.php') ja include(INC_webshopClass). Tämän jälkeen kaikki tietorakenneluokat ovat käytettävissä ja Webshop-olio voidaan luoda. Rajapinnan tietokanta luodaan suorittamalla tätä varten tehty komentosarja.

5.1 Asemointi ja ulkoasu

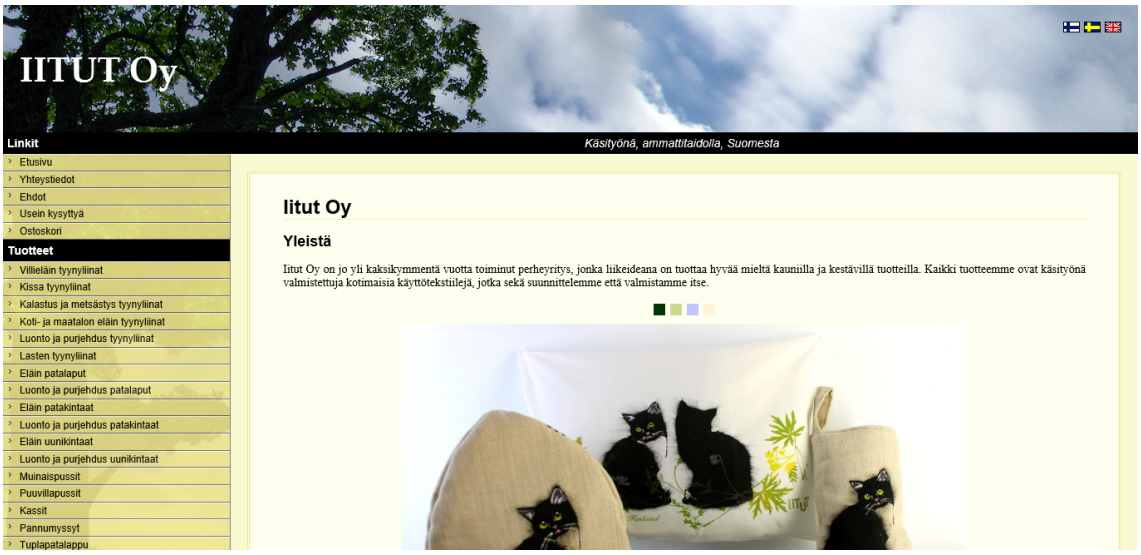
Helppokäyttöisyys on varmasti ulkoasua sunniteltaessa yksi tärkeimmistä huomioon otettavista seikoista. Verkkokaupan sisällön asemoinnissa kannattaa siksi karttaa erikoisia ratkaisuja, koska tarkoitus on saada tuotteita myytyä ei häkellyttää asiakkaita. Toisaalta graafisen ulkoasun on viestitettävä ammattimaisuutta. Ulkoasu ei saa myöskään olla täysin irrallinen yrityksen olemassaolevasta imagosta. Edellämainitut arvot ovat pitkälti käsitteellisiä ja suhteellisia, mutta ne on pyritty ottamaan työn aikana huomioon seuraavissa kappaleissa kuvailluilla tavoilla. Vähemmän käsitteellinen ja kuitenkin tärkeä seikka on sivuston yhteensopivuus mahdollisimman monen eri selaimen kanssa.

Sivuston sisältö on asemoitu perinteisellä tavalla. Linkit on asemoitu allekkain sivun vasempaan reunaan alkaen linkeistä etusivulle, yhteystietoihin, ostoskoriin jne. Staattisten linkkien jälkeen on lista tuotekategorioista, jotka haetaan rajapinnan avulla. Sisältö avautuu linkkien oikealle puolelle. Tilan hyötykäyttö maksimoidaan siten, että sisällölle varatun tilan annetaan kasvaa vaakasuunnassa selaimen tarjoaman tilan mukaan. Linkkien ja sisällön yläpuolella on tila yrityksen logolle ja motolle. Sivun oikeassa yläkulmassa on kuvake jokaista kieltä kohden, jolla sisältöä tarjotaan; niitä painamalla sivun

kieltä voidaan muuttaa. Sivun alareunassa on alue, alatunniste, jota käytetään yleensä kopiointisuojaan liittyvien asioiden ilmoittamiseen. (Kuva 11)

Kuva 11 Sivuston asemointi. Sivusto on asemoitu perinteisellä tavalla, ja sisällölle varattu tila skaalautuu selaimen koon mukaan täyttämään koko sivun

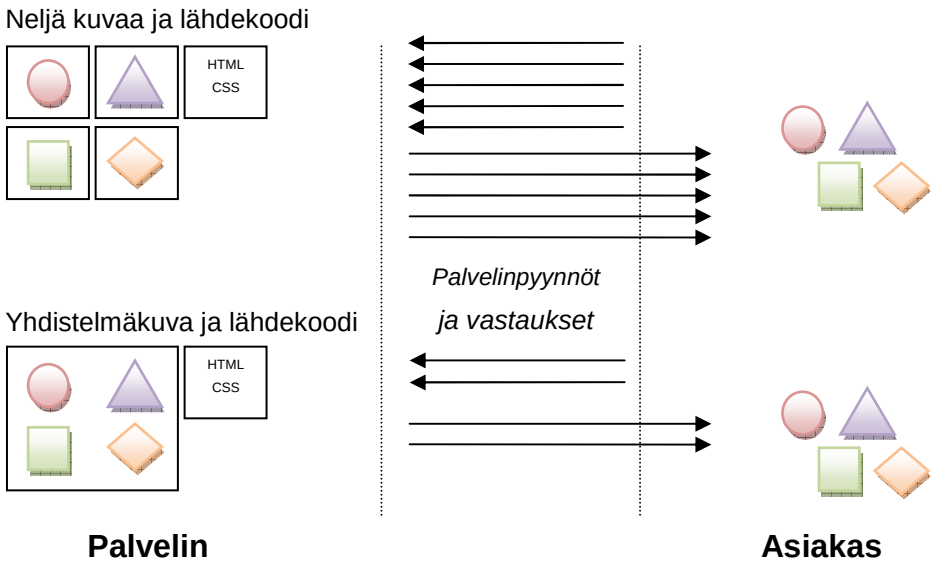
Graafinen ulkoasu sunniteltiin vastaamaan yrityksen edistämää imagoa. Yrityksestä mainittakoon, että se myy kotimaisia käsityötuotteita. Tämä on otettu huomioon sunniteltaessa sivuston värimaailmaa ja ideaa. Resurssien käyttöä pyrittiin optimoimaan mm. sillä, että taustakuva on toistuva kuvio, eikä vie paljon tilaa palvelimella, eikä siten aiheuta paljoa ruuhkaa ladattaessa. (Kuva 12)



Kuva 12 Sivuston graafinen ulkoasu. Sivuston graafinen ulkoasu on suunniteltu vastaamaan yrityksen imagoa.

Sivuston lähdekoodi on yleisesti hyväksyttyjen standardien mukaista. Ohjelmoitu XHTML ja CSS läpäisee W3C:n standardien validaation. Lisäksi ulkoasua on testattu yleisesti käytössä olevilla selaimilla ja varmistettu, että sisältö näkyy hyväksyttävällä ja yhdenmukaisella tavalla. Näihin kuuluvat mm. IE7-10 ja eri Google Chromen, Mozilla Firefoxin, Operan ja useiden mobiililaitteiden selainten versioita.

Verkkosivu on suunniteltu siten, että sen staattinen sisältö on mahdollisimman nopeasti ladattavissa: suurin osa grafiikasta on luotu CSS:n avulla ja kuvien käyttöä on minimoitu. Palvelimelle lähetettyjen HTTP-pyyntöjen määrää voidaan pienentää siten, että kuvat yhdistetään yhdeksi suureksi kuvaksi [22]. CSS:n avulla voidaan määritellä mikä osa tällaisesta kokoelmakuvasta käytetään. Kokoelmakuvan lataaminen palvelimelta muodostaa vain yhden palvelinpyynnön, siinä missä erillisten kuvien lataaminen muodostaisi monta. Saatu hyöty kasvaa, kun sivu on vilkkaasti liikennöity, kuvia on paljon tai kuvat ovat pieniä. Kuvien koko vaikuttaa hyötyyn, koska palvelinpyynnön koko on suhteessa suurempi tiedostokoon pienetessä. Jos kuva on palvelinpyynnön kokoinen, muodostuu liikennettä kaksinkertainen määrä tiedoston kokoon verrattuna. (Kuva 13)



Kuva 13 Yhdistelmäkuvat. Yhdistelmäkuvan lataaminen erillisten kuvien sijaan vähentää tarvittavien palvelinpyyntöjen ja vastausten määrää.

5.2 Ohjelmalogiikka

Ohjelmalogiikka jaettiin moduuleihin samantapaisesti kuin www-sisällönhallintatyökalussa. Kaikki moduulit, paitsi tuoteryhmä-moduuli, aukeavat verkkosivun sisältöalueelle keskelle sivua. Tuoteryhmä-moduuli esittää linkit eri tuoteryhmiin, joten se aukeaa aina sivun linkki-alueelle. Sisältöalueelle aukeavat moduulit ovat tuotteet, tuote ja ostoskori.

Tuotteet-moduuli on ikäänkuin tuotekatalogi, joka esittää kaikki tuotteet yhdestä tuoteryhmästä. Jos tuotteita on tarpeeksi paljon, ne jakautuvat eri sivuille. Sivun alalaidassa on linkkejä sivujen välillä navigoimista varten. Kun tuotteen kuvaa tai otsikkoa painaa, avataan tuote-moduuli. Tuote-moduuli näyttää yhden tuotteen ja tarjoaa mahdollisuuden lisätä se ostoskoriin.

Moduuleissa pyrittiin erottamaan HTML-asemointi PHP:n avulla luodusta ohjelmalogiikasta mahdollisimman hyvin. Tätä varten niihin ohjelmoitiin funktioita, jotka korvaavat parametriksi annetusta HTML:stä jotkin avainsanat parametreina annetuilla vaihtujilla. Tällä yritettiin simuloida MVC-ajattelua. Vaikka tämä ei tietenkään teknisesti ole sitä täydellisesti, se onnistuu siinä riittävän hyvin, eikä tee asioista tarpeettoman monimutkaista.

Koodiesimerkki 13

```
/* Ohjelmalogiikka (control) */
$product = $ws->getProduct(1);
/* Tiedon esittäminen (view) */
echoProduct('
        <table>
        <tr>
           <td>!NAME!</td>
           <td>!DESC!</td>
        </tr>
        </table>
        ', $product);
```

HTML on koodiesimerkissä eristetty ohjelmalogiikasta. Avainsanat '!NAME!' ja '!DESC!' korvataan 'product'-oliosta saatavalla tiedolla. Kun ohjelmalogiikkaa on paljon, ja kun HTML:n avulla tehty asemointi on monimutkaista, tästä tekniikasta koettiin olevan hyötyä koodin luettavuuden säilyttämisessä. (Koodiesimerkki 13)

Ostoskori-moduuli koostuu viidestä eri tiedostosta: cart ja cartState1:stä 4:ään. Cart-tiedosto on ostoskori-tilakoneen runko, joka käyttää hyödykseen Webshop-rajapinnan tilakonetta. CartState-tiedostot ovat tilakoneen eri tiloissa näytettäviä lomakkeita. Cart-tiedosto sisällyttää tilakoneen tilaa vastaavan cartState-tiedoston. Eri tilat ovat tuotteiden vahvistus, yhteystietojen keräys, tilaustietojen vahvistus ja tilauksen suorittaminen. CartState2 hyödyntää Webshop-rajapinnan tilaustietojen keräämiseen tehtyä mekanismia. Tilakonessa on hyödynnetty ns. PRG-navigointimallia (Post/Redirect/Get). PRG estää lomaketietojen lähettämisen useaan kertaan vahingossa palvelimelle [23]. Kun lomaketiedot vastaanotetaan (kerätään POST-globaalista) ne käsitellään normaalisti, mutta käsittelyn jälkeen selaimelle lähetetään uudelleenohjauskäsky seuraavalle sivulle (redirect). Uudelleenohjaus aiheuttaa uuden sivulatauksen, joka estää selainta enää käyttämästä edellisen sivulatauksen yhteydessä lähetettyjä lomaketietoja. Jos uudelleenohjausta ei tehdä, sivun päivitys aiheuttaa edellisten lomaketietojen uudelleenlähettämisen.

Tuotteiden lisäksi verkkokaupalla täytyy olla sivuja, kuten toimitusehdot ja yhteystiedot. Kuten sivun muunkin sisällön, näiden täytyy olla saatavilla monella eri kielellä. Tätä varten luotiin hakemistoja eri kieliä varten, jotka nimettiin kielten tunnisteiden mukaan. Hakemistot sisältävät samat HTML-tiedostot, mutta eri kielisinä. Riippuen siitä mikä kieli on valittu, tiedostoja sisällytetään sivustolle eri kansioista. Sama toiminnallisuus oltaisiin saatu aikaan Webshop-rajapinnan käännös-ominaisuuden avulla, mutta tällä tavalla kyseiset, paljon tekstiä sisältävät tiedostot, pysyvät puhtaasti HTML-kielisinä, joka tekee niiden ylläpitämisen helpommaksi. Muualla sivustossa monikielisyys toteutettiin rajapinnan käännös-ominaisuudella.

6 TIETOTURVA

Datan tai tiedon puhtaudella tarkoitetaan tässä opinnäytetyössä sitä, että se on halutussa ja jonkin oletuksen mukaisessa muodossa. Esimerkiksi syntymävuoden katsotaan tavanomaisesti olevan kokonaisluku (tyyppi). Lisäksi, tämän kokonaisluvun oletetaan olevan jonkin ajanlaskun mukainen, kuten länsimaissa ns. yleistä ajanlaskua. Jos lisäksi määritellään, että tämä syntymävuosi on elossa olevan henkilön syntymävuosi, on sen oltava (asiakirjan kirjoitushetkellä) n. 1900 – 2013 (rajaus). Kaikki nämä ovat oletuksia, mutta ohjelmoijan vastuuna on varmistaa, että hänen käsittelemänsä tieto vastaa hänen oletuksiaan, eli puhdistaa tieto.

HTML ei ota huomioon lomakkeiden kautta lähetettävän tiedon datatyyppiä, eikä sitä voida käyttää käyttäjän syöttämän tiedon rajoittamiseen. Myöskään sellaiset kielet kuten JavaScript eivät koskaan voi toimia tiedon puhdistamisessa, koska kaikki yritykset puhdistaa dataa selaimessa, tai missä tahansa muussa ohjelmassa, joka suoritetaan käyttäjän päätelaitteella, voidaan kytkeä pois päältä tai kiertää muokkaamalla niitä. Puhdistus on tehtävä laitteessa, jonka voidaan hyvällä tahdolla katsoa kuuluvan vain ohjelmoijan hallintapiiriin, eli missä ohjelman käyttäjällä tai asiakkaalla ei ole vaikutusvaltaa. Verkkosivujen tapauksessa tietoturvaratkaisut on tehtävä aina palvelinpuolella. Kaikkeen mitä verkkosivujen kautta oletettavasti vastaanotetaan on suhtauduttava varauksella. Lisättäessä mikä tahansa mekanismi, jolla käyttäjä voi vaikuttaa ohjelmalogiikan etenemiseen on kiinnitettävä erityistä huomiota tietoturvaan.

On esimerkiksi virheellistä olettaa, että HTML:llä luotu pudotusvalikko estää käyttäjää syöttämästä väärin muotoiltua tietoa. Se mitä palvelimelle saapuu ei välttämättä tule ohjelmojian verkkosivujen kautta, tai edes selaimelta. Ei voida myöskään olettaa, vaikka sen pitäisikin olla itsestään selvää, että käyttäjän selain esittää pudotusvalikon oikein. Saattaa kenties olla, että se esitetäänkin

tekstikenttänä, jolloin käyttäjä voi, vaikkakin mahdollisesti tarkoittaen hyvää, syöttää täysin odottamattomia tietoja lomakkeen avulla.

Puhdistamaton data johtaa usein tietoturva-aukkoihin. Yksi tunnetuimmista heikkouksista on ns. SQL-injektio. Tällainen heikkous syntyy kun puhdistamaton data pääsee vaikuttamaan tietokantaan tehtävään kyselyyn. PHP on erityisen herkkä tässä suhteessa, koska sen vaihtujat ovat tyypittömiä, jolloin oletetut kokonaisluvut voivatkin olla merkkijonoja jne.

6.1 SQL-injektiot

Oletetaan, että tietokanta sisältää 'tuotteet'-taulukon. Verkkosivu antaa käyttäjän valita tuoteryhmän pudotusvalikon avulla. Valikossa on tuoteryhmien tunnuksia kuten 1, 2, 3 jne. Määritellään että kyselyn suorittaa seuraava ohjelmakoodi:

```
$ryhmaTunnus = haeTunnusPudotusvalikosta();
$kysely = "SELECT nimi, kuvaus
           FROM tuotteet
           WHERE ryhma = $ryhmaTunnus";
$kyselynTulos = suoritaSqlKysely($kysely);
piirraTaulukko($kyselynTulos);
```

Ohjelmakoodissa on tässä esimerkissä tehty se oletus, että ryhmaTunnus on kokonaisluku HTML-sivun pudotusvalikosta. Jos tälle PHP-ohjelmakoodille lähetetään tietoa kiertämällä HTML-sivu, voidaan tietokannasta kysellä seuraava "ryhmaTunnus":

```
1 UNION ALL SELECT 'SQL-injektio', 'onnistui'
```

Tällöin verkkosivu suorittaisi tietokantaan kyselyn:

```
SELECT nimi, kuvaus
FROM tuotteet
WHERE ryhma = 1
```

```
UNION ALL
SELECT 'SQL-injektio', 'onnistui'
```

Tämän haavoittuvuuden avulla voitaisiin suorittaa sellaisia kyselyjä tietokantaan, joita ohjelmoijan ei ollut tarkoitus sallia. Vastaavia heikkouksia näennäisen vähäpätöisissä toiminnoissa on hyödynnetty suurissakin tietomurroissa lähiaikoina. Vaara SQL-injektioihin on yleisesti tiedossa, mutta tästä tietoudesta huolimatta niitä havaitaan aika ajoin jopa tietoturva-alan yritysten verkkosivuilla. [24][25][26][27]

SQL-injektiota vastaan voidaan suojautua PHP:ssä ja monissa muissakin ohjelmointikielissä:

1. Puhdistamalla merkkijonot erikoismerkeistä [28]

Merkkijonotyyppiset parametrit voidaan tehdä turvallisiksi kyselyä varten mysqli_real_escape_string-funktion tai mysqli::real_escape_string-metodin avulla (PHP:n ja MySQL:n tapauksessa). Kyseiset toiminnot valmistelevat merkkijonon niin, että seuraavaksi suoritettavassa kyselyssä se ymmärretään oikein, eivätkä siinä mahdollisesti olevat erikoismerkit pääse vaikuttamaan kyselyn rakenteeseen.

On erittäin huomionarvoista, että esim. kokonaisluvut on tehtävä turvallisiksi tyypittämällä, eikä tässä mainittuilla toiminnoilla ole mitään vaikutusta niiden turvallisuuteen [28]. Tämä pätee merkkijonoja lukuunottamatta kaikkiin tyyppeihin. Esim. seuraava kysely on haavoittuva:

```
"SELECT data FROM table WHERE id = " . mysql_real_escape_string($id)
```

Käytetty funktio estää sitaattien vahingollisen käytön id-vaihtujassa, mutta koska sitä käsitellään kyselyssä kokonaislukuna, se ei estä syöttämästä kyselyyn esim. seuraavaa:

```
1 UNION ALL SELECT valuableData FROM sensitiveTable
```

Vaihtuja tulisi siksi tehdä turvalliseksi yksinkertaisesti tyypittämällä:

```
"SELECT data FROM table WHERE id = " . (int) $id
```

2. Rajoittamalla käytetyn SQL-yhteyden oikeuksia [28]

On aivan ilmeistä, että Webshop-rajapinta ei tarvitse lukuoikeuksien lisäksi muita oikeuksia tietokantaan. Sillä ei ole tarkoitus muokata verkkosivujen sisältöä millään tavalla. Se ei myöskään tarvitse mitään oikeuksia logins-, users- tai action_log-taulukoihin. WWW-sisällönhallintatyökalu toisaalta tarvitsee kirjoitus oikeuden tietokannan kaikkiin taulukoihin, mutta sekään ei tarvitse esim. ALTER, DROP tai GRANT lausekkeita, joilla mahdollisesti saisi aikaan paljon vahinkoa väärin käytettynä. Sille ei siksi ole perinteisten SELECT--, INSERT-, UPDATE- ja DELETE-oikeuksien lisäksi annettu muuta kuin LOCK TABLES -oikeus, jota tarvitaan taulukoiden lukitsemisessa MyISAM-tietokantamoottorissa.

3. Harjoittamalla hyviä ohjelmointikäytäntöjä

Tyypittömyys ei ole PHP:ssä vapaus, vaan riski. Jos ohjelmoijan tarkoitus on tallettaa kokonaisluku vaihtujaan, on se hyvä nimenomaisesti tyypittää sellaiseksi. Dataa puhdistaessa ei tule miettiä mikä määrä puhdistamista poistaa uhan, vaan kuinka paljon sitä voidaan puhdistaa niin että siitä saa vielä halutun hyödyn. Esim. erikoismerkkien poistaminen on huonompi vaihtoehto kuin kaiken paitsi kirjainten ja numeroiden poistaminen. Jos kaikki tietokantaan tehtävät kyselyt löytyvät yhdeltä, selvästi rajatulta alueelta, todennäköisyys huolimattomuusvirheisiin luonnollisesti laskee. Webshop-rajapinnassa ja Webadmin-luokassa kyselyt on eristetty DBSocket-luokkiin, joten niiden tekemät kyselyt ovat kaikki samassa paikassa. Kyselyissä käytettävät vaihtujat on pyritty pitämään mahdollisimman lähellä sitä riviä jolla kysely lopulta suoritetaan. Tämä helpottaa sen hahmottamisessa, mitä tietoa kyselyssä käytetään ja onko se puhdistettua.

4. Käyttämällä hyödyksi parameterisoituja kyselyjä (engl. prepared statements)

MySQL ja monet muut tietokantajärjestelmät sallivat kyselyjen parametrisoinnin, tai esivalmistellut kyselyt. Tietokannalle kerrotaan etukäteen miltä kysely näyttää, jonka jälkeen sille voidaan antaa parametreja. Näin tietokanta tietää

miten parametrit tulisi sijoittaa kyselyyn ja sopivatko niiden tyypit haettujen sarakkeiden tyyppeihin. [28]

6.2 XSS-hyökkäykset

Toinen yleinen tietoturvariski, joka johtuu puhdistamattomasta datasta on XSS eli 'Cross Site Scripting'. Haavoittuvuus syntyy aina, kun käyttäjältä saatua dataa päästetään sivustolle suodattamatta sitä HTML-kohtaisista erikoismerkeistä. Datan ei saa antaa vaikuttaa sivuston HTML-koodiin.

Koodiesimerkki 14

Lomake:

```
<form method="get" action="sivu.php">
        <input type="text" name="user" />
        <input type="submit" />
</form>
```

sivu.php:

```
/* XSS haavoittuvuus! */
echo 'Terve! ' . $_GET['user'];
/* Ei haavoittuvuutta */
echo 'Terve! ' . htmlspecialchars($_GET['user']);
```

Joskus virheellisesti ajatellaan, että koska data esitetään vain datan lähettäneen selaimessa, tällä ei ole väliä. Historia kuitenkin osoittaa, että tätä on onnistuttu hyödyntämään laajoissakin huijauksissa. [29][30]

Jopa seuraava aiheuttaa XSS-haavoittuvuuden:

```
echo $_SERVER['HTTP_USER_AGENT'];
```

Ylläoleva rivi tulostaa sivulle käyttäjän selaimen nimen ja muita tietoja, mutta avaa XSS-haavoittuvuuden, koska tietoa ei puhdisteta. Kaikki tieto mitä palvelin vastaanottaa voi mahdollisesti olla haitallista. Mikään ei varmista, että HTTP-

pyynnön User Agent -otsake sisältää olemassa olevan, todellisen selaimen tietoja, eikä esim. mahdolliseen hyökkäykseen suunniteltua merkkijonoa. Jokin haittaohjelma saattaisi esimerkiksi muokata vaarantuneiden käyttäjien selaimen toimintaa niin, että kuvaillulla tavalla haavoittuvaista sivustoa voitaisiin käyttää osana huijausta, jossa hyödynnettäisiin sivuston ylläpitäjän mainetta.

6.3 Salasanat ja niiden säilyttäminen tietokannassa

Vastoin yleistä käsitystä, salasanan minimipituuden asettaminen mahdollisimman suureksi ei aina paranna järjestelmän tietoturvaa. Jos pituudeksi määritellään esim. 8 merkkiä, on todennäköistä, että suurin osa käyttäjistä valitsee tasan 8 merkkiä pitkän salasanan. 8 merkin kombinaatioita on vähemmän kuin esim. 4 – 8 merkin kombinaatioita. Suurempi minimipituus siis saattaa jopa laskea järjestelmän yleistä tietoturvatasoa, ja siksi käyttäjien valistaminen onkin parempi vaihtoehto kuin pakottaminen. Sama pätee luonnollisesti sääntöön, joka pakottaa käyttämään kirjaimia, numeroita ja erikoismerkkejä salasanoissa. Toisaalta, mikäli pituutta ei rajoiteta, suuri osa käyttäjistä todennäköisesti valitsee riittämättömän salasanan. Jos siis salasanan muodostamiselle asetetaan ehtoja, on niiden oltava riittävän vaativia. [31]

Webadmin-luokkaan rakennettiin mekanismi, jolla minimipituus ja monimutkaisuus voidaan pakottaa.

Samaa periaatetta voidaan soveltaa salasanojen suolaamiseen tietokannassa. Jos suolan pituuden sallitaan vaihdella, eri suola kombinaatioita on paljon enemmän. Suolan ei tarvitse aina olla absoluuttisen uniikki, mutta tarpeeksi satunnainen, jotta se piilottaa samat salasanat eri tarkisteiksi ja tekee esilaskelmoitujen tarkisteiden käytön mahdottomaksi.

Salasanojen tarkisteiden luomiseen käytetyn algoritmin on tarkoitus olla mahdollisimman hidas, eli vaativa laskea. Mitä vaativampi algoritmi on, sitä kauemmin yhden tarkisteen arvaaminen kestää ja sitä mahdottomampaa kaikkien tietokannan tarkisteiden arvaaminen on. Yksi vaihtoehto on käyttää

tarkistealgoritmia useita kertoja peräkkäin; toinen on valita kryptografiaan tarkoitettu algoritmi, joka on valmiiksi raskas (esim. bcrypt, jonka laskelmointivaikeutta voidaan jopa muuttaa vastaamaan laskentatehoja).

Joskus ajatellaan, että jonkin sovelluksen ei tarvitse kiinnittää suurta huomiota salasanojen turvalliseen säilyttämiseen, koska sitä ei käytetä vakaviin tarkoituksiin. Käyttäjät usein kuitenkin kierrättävät samoja salasanoja eri palveluissa, joten tietoturvaan on aina kiinnitettävä huomiota [31].

6.4 Replay-hyökkäykset

Replay-hyökkäyksessä salattuja viestejä kaapataan ja lähetetään uudelleen palvelimelle, joka saa palvelimen toistamaan viestien pyytämät toimenpiteet. Nonce-järjestelmällä tämä voidaan kuitenkin estää, koska vuoronumerot eivät ole enää samat, kun salattuja viestejä lähetetään palvelimelle uudelleen.

7 TULOSTEN ARVIOINTI

Rajapinnan avulla toteutettu esimerkkisivusto todistaa, että alkuperäinen visio saavutettiin ja siihen saatiin lisättyä tarvitut ominaisuudet verkkokaupan pystyttämiseksi.

WWW-sisällönhallintatyökalulla voi muokata kaikkea sitä tietoa mitä Webshop-rajapinnan kautta voi hakea tietokannasta. Tietokantaa ei tämän ansiosta tarvitse käsitellä suoraan sen luomisen jälkeen. Työkalun rakenne on ylläpidon kannalta helppo, sillä kaikki sen todellinen ohjelmalogiikka on erillään käyttöliittymästä Webadmin-luokassa. Lisäksi jaettujen resurssien käyttö on eristetty WebadminDBSocket ja WebadminSessionData-luokkiin. Käyttöliittymän toiminto-moduulit sisältävät vain HTML-koodia ja lomaketietoja käsittelevää ohjelmalogiikkaa, joka käyttää Webadmin-luokan toimintoja. Vaikka ei voida puhua puhtaasta MVC-rakenteesta, on siinä kuitenkin selvästi eroteltu käyttöliittymä, tietokanta ja ohjelmalogiikka toisistaan.

Työtä on mahdollista jatkaa kehittämällä erilaisia maksutapoja rajapintaan. Myös sisäkkäiset tuoteryhmät olisivat hyvä lisä. Järjestelmän tietokanta tukee usean kuvan liittämistä yhteen tuotteeseen, mutta rajapinta hakee vain niistä ensimmäisen. WWW-sisällönhallintotyökalussa on sama rajoitus. Järjestelmästä jätettiin lisäksi pois toiminto, joka mahdollistaisi ns. ominaisuuksien liittämisen tuotteisiin. Tuotteilla voisi olla esim. eri väri- tai materiaalivariaatioita. Eri tuotevariaatiosta voidaan tietenkin tehdä kokonaan erillisiä tuotteita, mutta toiminto olisi hyödyllinen. Järjestelmään voitaisiin myös lisätä tilausten seuranta, sen sijaan, että tilaukset lähetetään kaupan ylläpitäjän sähköpostiin. Tämä parantaisi tilausten käsittelyn luotettavuutta, joka nyt riippuu sähköpostin välittymisestä.

LÄHTEET

[1] TIOBE Software. TIOBE Programming Community Index for February 2013. [www-dokumentti]. Saatavilla: http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html. Luettu Helmikuu 2013.

[2] PHP. PHP manual, What is PHP. [www-dokumentti]. Saatavilla: http://www.php.net/manual/en/intro-whatis.php. Luettu Helmikuu 2013.

[3] Scriven, Selene. Problems with PHP. [www-dokumentti]. Saatavilla: http://toykeeper.net/soapbox/php_problems/. Luettu Helmikuu 2013.

[4] McIver, Linda. "The effect of programming language on error rates of novice programmers." 12th Annual Workshop of the Psychology of Programming Interest Group. 2000. Saatavilla: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.33.9413&rep=rep1&type=pdf.

[5] MySQL. Market Share. [www-dokumentti]. Saatavilla: http://www.mysql.com/why-mysql/marketshare/. Luettu: Helmikuu 2013.

[6] MySQL. "MySQL 5.5 manual, 14.3.1. InnoDB as the Default MySQL Storage Engine". [www-dokumentti]. Saatavilla: http://dev.mysql.com/doc/refman/5.5/en/innodb-default-se.html. Luettu: Helmikuu 2013.

[7] Zaitsev, Peter. Should you move from MyISAM to InnoDB. 2009. [www-dokumentti]. Saatavilla: http://www.mysqlperformanceblog.com/2009/01/12/should-you-move-from-myisam-to-innodb/. Luettu: Helmikuu 2013.

[8] Ward, Martin. A Definition of Abstraction.1996.

[9] Wrap Those Session Variables! . 2007. [www-dokumentti]. Saatavilla: http://www.codeproject.com/Articles/17995/Wrap-Those-Session-Variables. Luettu: Helmikuu 2013.

[10] Sears, Russell; Van Ingen, Catharine & Gray, Jim. To BLOB or Not To BLOB: Large Object Storage in a Database or a Filesystem?. 2006.

[11] Imelda C. Go. Rounding in SAS®: Preventing Numeric Representation Problems.

[12] 10 Security Trends To Watch In 2012. [www-dokumentti]. Saatavilla: http://www.informationweek.com/security/vulnerabilities/10-security-trends-to-watch-in-2012/232400392. Luettu: Helmikuu 2013.

[13] Crispin, Cowan & Calton, Pu. Death, Taxes, and Imperfect Software: Surviving the Inevitable.

[14] The Register. Googlephone security team seeks bug hunters. [www-dokumentti]. Saatavilla: http://www.theregister.co.uk/2008/08/20/googlephone_bug_hunt/. Luettu: Helmikuu 2013.

[15] Android Security Team. Google, Introducing the Android Security Team. [www-dokumentti]. Saatavilla: http://seclists.org/fulldisclosure/2008/Aug/378. Luettu: Helmikuu 2013.

[16] PHP. Sessions, Introduction. [www-dokumentti]. Saatavilla: http://www.php.net/manual/en/intro.session.php. Luettu: Helmikuu 2013.

[17] Google. Make The Web Faster, Optimized caching. [www-dokumentti]. Saatavilla: https://developers.google.com/speed/docs/best-practices/caching. Luettu: Helmikuu 2013.

[18] PHP. header. [www-dokumentti]. Saatavilla: http://php.net/manual/en/function.header.php#refsect1-function.header-changelog. Luettu: Helmikuu 2013.

[19] Wille, Christoph. Storing Passwords - done right!. [www-dokumentti]. Saatavilla: http://www.aspheute.com/english/20040105.asp. Luettu: Helmikuu 2013.

[20] MySQL. Internal Locking Methods, MySQL documentation. [www-dokumentti]. Saatavilla: http://dev.mysql.com/doc/refman/5.0/en/internal-locking.html. Luettu: Helmikuu 2013.

[21] Wikipedia. Cryptographic nonce. [www-dokumentti]. Saatavilla: http://en.wikipedia.org/wiki/Cryptographic_nonce. Luettu: Helmikuu 2013.

[22] Google. Make The web faster, Combine Images using CSS Sprites. [www-dokumentti]. Saatavilla: https://developers.google.com/speed/docs/best-practices/rtt#SpriteImages. Luettu: Helmikuu 2013.

[23] Jouravlev, Michael. Redirect After Post. [www-dokumentti]. Saatavilla: http://www.theserverside.com/news/1365146/Redirect-After-Post. Luettu: Helmikuu 2013.

[24] Kirk, Jeremy. Microsoft.co.uk Succumbs to SQL Injection Attack. [www-dokumentti]. Saatavilla: http://www.pcworld.com/article/133583/article.html. Luettu: Maaliskuu 2013.

[25] Kirk, Jeremy. MySQL Website Falls Victim to SQL Injection Attack. [www-dokumentti]. Saatavilla: http://www.pcworld.com/article/223457/article.html. Luettu: Maaliskuu 2013.

[26] BBC news. Royal Navy website attacked by Romanian hacker. [www-dokumentti]. Saatavilla: http://www.bbc.co.uk/news/technology-11711478. Luettu: Maaliskuu 2013.

[27] BBC news. US man 'stole 130m card numbers'. [www-dokumentti]. Saatavilla: http://news.bbc.co.uk/2/hi/americas/8206305.stm. Luettu: Maaliskuu 2013.

[28] MySQL. Guide to PHP Security, 3 SQL-injection.

[29] CERT. CERT® Advisory CA-2000-02 Malicious HTML Tags Embedded in Client Web Requests. [www-dokumentti]. Saatavilla: http://www.cert.org/advisories/CA-2000-02.html. Luettu: Helmikuu 2013.

[30] Apache. Cross Site Scripting Info. [www-dokumentti]. Saatavilla: http://httpd.apache.org/info/css-security/. Luettu: Maaliskuu 2013.

[31] The Imperva Application Defense Center (ADC). Consumer Password Worst Practices. Saatavilla: http://www.imperva.com/docs/wp_consumer_password_worst_practices.pdf.