OPINNÄYTETYÖ

AKI KARVONEN 2012

TIETOKANTAPOHJAINEN VERKKOPORTAALI

Rovaniemen
ammattikorkeakoulu
University of Applied Sciences
LUC

TIETOJENKÄSITTELYN KOULUTUSOHJELMA

ROVANIEMEN AMMATTIKORKEAKOULU

LUONNONTIETEIDEN ALA

Tietojenkäsittelyn koulutusohjelma

Opinnäytetyö

TIETOKANTAPOHJAINEN VERKKOPORTAALI

Aki Karvonen

2012

Ohjaaja Aarre Jortikka

Hyväksytty _____ 2012 _____

Rovaniemen ammattikorkeakoulu
University of Applied Sciences
LUC

Luonnontieteiden toimiala
Tietojenkäsittelyn koulutusohjelma

Opinnäytetyön tiivistelmä

Tekijä	Aki Karvonen	Vuosi	2012
Toimeksiantaja			
Työn nimi	Tietokantapohjainen verkkoportaali		
Sivu- ja liitemäärä	30		

Tämän opinnäytetyön aiheena on tietokantapohjaisen verkkosivuston rakentaminen. Sivustolla käyttäjätunnuksin ja salasanoin sisäänkirjautuneet käyttäjät voivat keskenään jakaa ja kommentoida kuvatiedostoja. Työn tavoite on saavuttaa ja osoittaa taidot toimivan verkkoyhteisön luonnista. Tietoturvan kehittämiseen kiinnitettiin erityistä huomiota.

Sivuston toteutukseen käytettiin kotikoneelle asennettua kehitys- ja testauspalvelinta sekä HTML-, PHP-, CSS- JavaScript- ja MySQL-tekniikoita. Työn lähteinä on käytetty Internetiä ja oppikirjoja.

Sivuston sisällön tuottavat sen käyttäjät. Tässä opinnäytetyössä rakennettiin verkkoyhteisön tarkoituksiin toimiva alusta. Sivustoa on tarkoitus jatkossa kehittää huomattavasti monipuolisemmaksi Web 2.0:n hengessä.

Sivustolle asetetut tavoitteet täyttyivät. Opin tarvittavat taidot verkkoyhteisön rakentamiseen ja ylläpitämiseen sekä sain paljon uusia ideoita jatkokehitystä varten. Minulla on selkeä kuva asioista, joita tulee jatkossa opiskella seuraavien tavoitteitteni saavuttamiseksi.

Avainsana(t) PHP, MySQL, Web 2.0

Muita tietoja Sivustoa voi testata osoitteessa http://www.kuvapost.net/ tunnuksella ”oparitestaaja”. Salasana on sama.

Rovaniemen ammattikorkeakoulu
University of Applied Sciences
LUC

School of Technology
Degree Programme in
Information Technology

Abstract of Thesis

Author Aki Karvonen

Year 2012

Commissioned by
Subject of thesis A Database Driven Web Portal
Number of pages 30

The subject of this thesis is how to build a database driven web portal. Users log into the site by their user names and passwords to share and comment on pictures found on the Internet or perhaps taken by the users themselves. A personal goal when beginning the thesis was to teach oneself – and demonstrate – the ability to build a base for a functional web community. Special attention was paid to the issue of information security.

A developmental web server was installed on a personal computer for easy testing of the site. HTML, CSS, PHP, JavaScript and MySQL were used to execute the portal with sources both from the Internet and from study books.

All of the contents – with the exception of menus and headings, of course – are generated by the users. It is intended that after this thesis is reviewed the site will be opened for general use and it will be developed further in the guidelines of Web 2.0.

The goals for this thesis were reached by learning the necessary skills of developing and managing a communal web site. Also good ideas for the future development of the site came up. A good understanding of what to study next to expand and develop the site further has been gained.

Key words PHP, MySQL, Web 2.0

Special remarks The URL for the site is http://www.kuvapost.net/ and to review this thesis you can use “oparitestaaja” as the username and password.

SISÄLTÖ

KUVIOLUETTELO

1 JOHDANTO

Tämän opinnäytetyön aiheena on rakentaa Internetissä toimiva verkkoportaali, johon käyttäjät kirjautuvat sisään käyttäjätunnuksen ja salasanan yhdistelmällä. Käyttäjät tallentavat sivustolle tiedostoja omalta koneeltaan. Ainakin aluksi sallitut tiedostot ovat konseptin mukaisesti kuvatiedostoja. Käyttäjät näkevät uusimmat kuvat listattuna etusivulla, jonka lisäksi käyttäjätunnuksia voi tarkastella tarkemmin ja selata eri käyttäjien tallennushistoriaa.

Verkkoportaalin ohjelmointi on toteutettu käyttämällä HTML/CSS, PHP-, ja JavaScript-tekniikoita. Sivuston tietokantana toimii MySQL.

Internetissä on useita suosittuja sivustoja, joiden idea on jotakuinkin sama kuin tässä opinnäytetyössä, mutta tarkoituksena on jatkossa laajentaa sivustoa kattamaan muitakin Web 2.0:aan yhdistettyjä tekniikoita ja toimintoja, jolloin sivustosta tulee massasta erottuvampi ja se kerää oman käyttäjäkuntansa.

Työn tarkoituksena oli opettaa itselle PHP/MySQL-yhdistelmän hallinta ja verkkoyhteisön perustan rakentaminen. Tavoitteena on myös, että tämän raportin lukija saa hyvän yleiskuvan tietokantapohjaisen verkkosivuston rakentamisesta.

Tällä hetkellä sivustolla on ylimääräinen salasanasuojaus ennen varsinaista kirjautumissivua. Suojaus poistuu sivuston varsinaisen käyttöönoton jälkeen eli kun tämä opinnäytetyö on arvioitu.

Sivusto on osoitteessa http://www.kuvapost.net/. Tunnus, jolla tämän opinnäytetyön tarkastajat voivat sivustolle kirjautua, on ”oparitestaaja”. Salasana on myöskin ”oparitestaaja”.

2 TERMIT JA TYÖKALUT

Internet-sivut sijaitsevat aina jollain palvelimella. Apache-palvelin on Apache Software –säätiön julkaisema Internet-palvelin, joka perustuu avoimeen lähdekoodiin. Apache on maailman käytetyin palvelin ja sen voi ladata säätiön WWW-sivuilta ilmaiseksi. (Apache Software Foundation 2011).

CSS (Cascading Style Sheets) on yksinkertainen, mutta tehokas tapa muokata verkkosivuston ulkoasua, kuvien asettelua, fontteja ja värejä. Sen on kehittänyt World Wide Web Consortium eli W3C. (World Wide Web Consortium 2011a.)

HTML (Hypertext Markup Language) on myös W3C:n kehittämä WWW-tekniikka. Se ei ole ohjelmointikieli, vaan kokoelma eräänlaisia komentoja eli tageja, joilla sivuston sisältö kuvaillaan Internet-selaimelle. (World Wide Web Consortium 2011b.)

JavaScript ei nimestään huolimatta ole pelkkä ”skriptauskieli”, vaan sitä pidetään ohjelmointikielenä. JavaScriptillä sivustolle lisätään interaktiivisuutta ja sen koodin voi joko upottaa HTML- ja/tai PHP -koodin sekaan tai sen voi sijoittaa erilliseen tiedostoonsa, jota kutsutaan käyttöön tarvittaessa. JavaScript-koodi sijoitetaan HTML-koodin sekaan omalla <script>-tagillaan. (Negrino-Smith 2007, 2.)

MySQL on relaatiotietokantojen hallintajärjestelmä, jonka on kehittänyt ruotsalainen MySQL AB. Sen käyttäminen on ilmaista ei-kaupallisessa toiminnassa, mutta se on täysin pätevä kilpailija maksullisten tietokantajärjestelmien kanssa. Kaupallisessa käytössäkin se on monia kertoja halvempi, kuin kilpailijansa. Käyttötarkoituksesta riippuen MySQL:ää käyttämällä voi säästää tuhansia euroja ohjelmistolisenssimaksuissa. (Heinisuo-Rauta 2007, 37-38.)

PHP (Hypertext Preprocessor) on ohjelmointikieli, joka sopii erinomaisesti WWW-ohjelmointiin. Toisin kuin JavaScript, joka suoritetaan lähettämällä ohjelmakoodi käyttäjän selaimeen, PHP-koodi suoritetaan WWW-palvelimella. Näin ollen sivuston käyttäjä ei voi tarkastella sivuston lähdekoodia sellaisena, kuin se on palvelimelle tallennettu. (The PHP Group 2011.)

Apache-palvelimelle on asennettava PHP-tulkki niin sanotuksi moduuliksi, jonka jälkeen PHP toimii osana Apachea. (Heinisuo-Rauta 2007, 26.)

SQL eli Structured Query Language on 1970-luvulla IBM:n laboratorioissa kehitetty relaatiotietokantojen määrittely- ja käsittelykieli. SQL-kielellä tehdään muun muassa kaikki tietokantataulujen haku-, päivitys-, poisto ja määrittelytoiminnot. SQL-komennot voidaan suorittaa asiaan soveltuvalla hallintatyökalulla tai suoraan ohjelmistokoodiin upotettuina toimintoina. (Hovi-Huotari-Lahdenmäki 2005, 345.)

Web 2.0 on vaikeasti määriteltävissä. Hakusanoilla ”web+2.0” saa Googlesta tällä hetkellä 119,000,000 hakutulosta, mutta sen määritelmästä ei ole yksimielisyyttä. Lyhyesti määriteltynä Web 2.0 tarkoittaa Internetin vuorovaikutteisuutta, dynaamisuutta aikaisemman staattisuuden sijaan.

1990-luvulla, jolloin Internetin käyttö yleistyi, WWW-sivut olivat staattisia sivustoja, joihin käyttäjät eivät juuri voineet vaikuttaa. 2000-luvun alun IT-kuplan puhjettua lupaavat, uudet ideat alkoivat menestyä. Internet-kehittäjät alkoivat ymmärtää, että käyttäjät haluavat Internetiltä vuorovaikutteisuutta, mahdollisuutta tuottaa itse sisältöä ja olla osana yhteisöjä. Web 2.0 on siis nykyinen Internet-kokemus, jossa enää harva asia on täysin staattista, yksisuuntaista informaatiota. Siinä missä ennen pidettiin henkilökohtaisia verkkosivuja, nyt pidetään blogeja, joiden sisällöistä voidaan käydä mielenkiintoisia keskusteluja niiden kommenttiosioissa. Myös erittäin suosittu Wikipedia (ja oikeastaan kaikki wikit) on iso osa Web 2.0:aa. (O’Reilly, T. 2011.)

3 VERKKOPORTAALIN KÄYTTÖTARKOITUS

Sivuston käyttötarkoitus on puhtaasti viihteellinen. Sillä ei ole yhtä tiettyä kohderyhmää, mutta Internetissä tämän työn kaltaiset sivustot (reddit.com, thismight.be/offensive, naurunappula.com, jne.) houkuttelevat yleensä nuoria miehiä, jotka ovat tottuneet ”tuhlaamaan” aikaansa selailemalla hauskoja ja/tai mielenkiintoisia kuvia, animaatiopätkiä tai videoita verkossa ja jakamaan niitä kaveripiirinsä tai jonkin verkkoyhteisön kesken.

Ilman käyttäjiä sivusto olisi tyhjä. Verkkoportaalin luonteesta johtuen portaalin kehittäjä ja/tai ylläpitäjä ei itse tuota sivulle varsinaista yhteisöllistä sisältöä muuten kuin olemalla yksi käyttäjistä. Kehittäjä luo puitteet, joissa käyttäjät voivat itse tuottaa ja jakaa sisällön. Sisällölle ei ole olemassa muuta rajausta, kuin se, että ladattavien tiedostojen on oltava kuvatiedostoja. Tähän tulee muutos myöhemmin, jolloin sivustolle toteutetaan ainakin linkkitietokanta, joka mahdollistaa suoran linkittämisen ulkoiselle sivustolle. Tällöin palvelimelle ei tarvitse tallentaa tilaa vieviä tiedostoja. Toki sellainen rajoitus on sanomattakin selvää, että kuvien sisältö ei saa olla laitonta, mutta muuten ”kaikki käy” sananvapauden nimissä. Toimivassa verkkoyhteisössä yhteisö itse on paras sisällön tason säätelijä. Vastenmieliset julkaisut saavat nopeasti negatiivista palautetta ja viimeistään ylläpitäjät poistavat yhteisöistä käyttäjät, jotka tahallaan tai tyhmyyttään julkaisevat laitonta tai moraalitonta materiaalia.

4 TYÖN TOTEUTUS

4.1 Tekstieditori

Varsinaisen opinnäytetyöni toteuttamisen aloitin valitsemalla työvälineet. Pyrin käyttämään mahdollisimman paljon avoimen lähdekoodin ilmaisia sovelluksia, paitsi siinä tapauksessa, että jo ennakkoon omistin lisenssit maksullisista ohjelmista, kuten Windows Vista ja Windows 7 ja Microsoft Office.

WWW-sivujen kehittäjät käyttävät yleisesti tarkoitukseen kehitettyjä kehitysohjelmistoja. Yksi suosittu on Adobe Dreamweaver, jolla WWW-sivujen teko tapahtuu pitkälti ohjelman avustamana. Suurta osaa koodista ei kirjoiteta itse, vaan projekti kasataan Dreamweaverin tarjoamin apuvälinein. Tässä työssä Dreamweaveria ei kuitenkaan käytetty muun muassa sen maksullisuuden vuoksi, mutta pääasiassa siksi, että pelkkä tekstieditori soveltuu täysin hyvin tämän opinnäytetyön koodaamisen vaatimuksiin. Tekstieditoria käyttämällä ohjelmoija joutuu kirjoittamaan kaiken koodinsa itse, jolloin aiheeseen tulee syvennyttyä paremmin ja ohjelmointiprosessi on opettavaisempi.

Tekstieditoriksi valikoitui Notepad++. Jo nimestä voi päätellä sen viittaavan Windowsista tuttuun Muistio- eli Notepad-ohjelmaan, joka sekin on kelpo ohjelma pienimuotoiseen ohjelmointiin. Notepad++ on kuitenkin kehittyneempi, ohjelmointiin tarkoitettu työkalu. Sillä on erittäin kätevä koodata PHP/HTML/CSS-yhdistelmää. Yksi ohjelman hyvistä ominaisuuksista on havainnollistava värikoodaus. Notepad++ värittää oikein kirjoitetut komennot tietyillä väreillä, jolloin on helppo huomata vaikkapa kirjoitusvirheet, joiden takia koodi ei toimi. Tämä säästää huomattavasti aikaa, kun pitää etsiä virheitä koodista.

4.2 WWW-palvelin

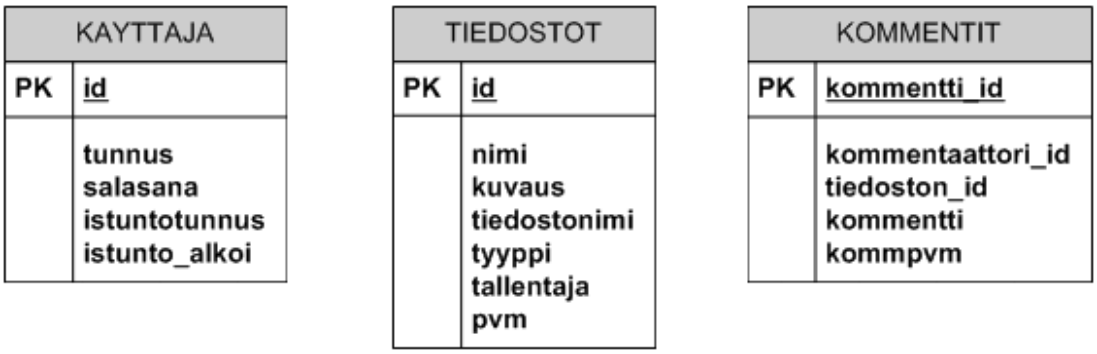
Jotta verkkosivuja voi testata koodailun ohessa, tulee ne sijoittaa jollekin palvelimelle. Työn alussa vaihtoehtoina on hakea jostain Internetin webhotelleista kotisivutilaa tai asentaa palvelin henkilökohtaiselle koneelle. Tässä työssä on päädytty jälkimmäiseen. Sivujen testailu koodailun ohessa on vaivattomampaa, kun jokaisen pienen muutoksen jälkeen ei tarvitse ladata muokattua tiedostoa ulkoiselle palvelimelle.

Kehityspalvelimena toimii Apache Software Foundationin kehittämä Apache 2.2 -palvelimen, johon sen jälkeen asennettiin PHP5 ja MySQL. MySQL-tietokannan hallintaan käytettiin Windowsin Command Prompt –konsolia. Tietokannan hallintaan olisi toki tarkoitukseen sopivia apuohjelmia, mutta tässäkin tapauksessa työskentely tapahtui mahdollisimman perustasolla ja tekstipohjaisesti. Myöhemmin, kun sivut siirrettiin ulkoiselle palvelimelle, tietokannan hallintaan otettiin käyttöön web-hotellin asiakkailleen tarjoama selainpohjainen tietokannanhallintaohjelma phpMyAdmin.

4.3 Tietokannan suunnittelu ja perustaminen

Itse työn tekeminen alkaa niin sanotulla käsiteanalyysilla. Tässä vaiheessa suunnitellaan tietokannan looginen toiminta yleisellä tasolla, ilman tarkempaa teknistä pohdintaa. (Hovi ym. 2005, 24.)

Käsiteanalyysini alkoi tarvittavien tietokantataulujen ja niiden välisten yhteyksien luonnostelulla kynällä paperille. Tämän jälkeen taulut mallinnettiin Microsoft Visio 2010 –ohjelmalla, joka on niin sanottu CASE-ohjelma (Computer Aided Software Engineering).



Kuvio 1. Tietokannan taulut.

Sivuston tietokanta koostuu siis tällä hetkellä kolmesta taulusta: KAYTTAJA, TIEDOSTOT ja KOMMENTIT.

KAYTTAJA-taulussa (tästä eteenpäin taulujen nimet on kirjoitettu pienellä ja skandinaavisella kirjaimistolla) määritellään ensin id-kenttä, joka toimii taulun yksilöivänä perusavaimena ja saa arvonsa automaattisesti, kun uusi käyttäjä luodaan tietokantaan. Tämä on siis eräänlainen käyttäjän asiakasnumero.

Tunnus- ja salasanakentät saavat arvonsa samassa yhteydessä. Nämä tiedot tulevat sivuston jäsenrekisteröintisivulta.

Kun käyttäjä kirjautuu sivustolle, sivuston PHP-koodi luo istuntotunnuskenttään yksilöllisen tunnuksen. Tätä tunnusta käytetään hyväksi myöhemmin useaan otteeseen. Istuntotunnusta käsitellään tarkemmin seuraavassa kappaleessa.

Istunto_alkoi-kenttään tallennetaan kirjautumishetken ajankohta. Tätä tietoa tarvitaan uloskirjautumisen yhteydessä, jolloin kirjautumisen yhteydessä luotua evästettä muokataan.

Myös tiedostot-taulu alkaa ladattavan tiedoston yksilöivällä id-tunnuksella, joka toimii taulun perusavaimena, kuten aiemmin käyttäjä-taulussa. Kun käyttäjä tallentaa palvelimelle tiedoston, kyseinen tiedosto saa tietokannassa id-arvokseen tiedoston yksilöivän lukuarvon, jota hakemalla tietokannasta löydetään kyseinen tiedosto.

Nimi-kenttä saa arvonsa, kun käyttäjä tiedostoa tallentaessaan antaa tiedostolleen eli sivuston nykyversiossa kuvalleen, otsikon. Tämä teksti näkyy kuvan yläpuolella, kun kuva haetaan tietokannasta ja esitetään käyttäjälle. Kuvaus-kenttä toimii vastaavalla tavalla, mutta tähän kenttään tallennettava teksti toimii kuvatekstinä.

Tyyppi-kenttään tallentuu tallennettavan tiedoston tiedostotyyppi, tässä vaiheessa joko ”image/jpeg” tai ”image/gif”. Tallentaja-kenttään tulee tiedoston tallentajan tunnus ja pvm-kenttään tiedoston tallennushetki.

Käyttäjät voivat kommentoida omia ja muiden käyttäjien tallentamia tiedostoja yksinkertaisen kommentointisysteemin avulla. Tätä varten tietokantaan tulee luoda oma taulunsa kommenttien tallentamiseen. Kommentit-taulun alussa määritellään taas kommentin yksilöivä kommentti_id-kenttä. Tämä kenttä saa yksilöllisen lukuarvonsa automaattisesti, kun kommentti tallennetaan.

Kommentaattori_id-kenttään tallennetaan kommentin jättävän käyttäjän id-numero. Tiedoston_id-kenttään tallennetaan sen tiedoston id-arvo, jota

kommentti koskee. Näillä tiedoilla kommentti voidaan yhdistää oikeaan tiedostoon ja kommentin jättäjään.

Kommentti-kenttään tallennetaan itse kommentin sisältö. Jotta käyttäjät pystyvät seuraamaan, milloin kommenttiosion keskustelut on käyty, tallennetaan kommentit-tauluun myös sen hetken aikaleima, jolloin kommentti on jätetty. Aikaleimaa muokataan luettavampaan muotoon myöhemmin, kun kommentit esitetään käyttäjille.

4.4 Kirjautuminen ja tunnusten luonti

Ensimmäinen asia, jonka sivuston vierailija näkee, on kirjautumissivu. Jos vierailijalla on jo aiemmin luotu käyttäjätunnus ja salasana, hän voi kirjautua niillä sivustolle. Muussa tapauksessa hän voi rekisteröityä kirjautumissivun linkistä.

Kuvio 2. Kirjautumislomake.php ja uusitunnus.php.

Kun käyttäjä antaa kirjautumislomakkeella tunnuksensa ja salasanansa, otetaan se vastaan kirjautuminen.php –tiedostoon. Jos vähintään jompikumpi annetuista tiedoista on väärä tai tyhjä, ohjataan käyttäjä takaisin kirjautumislomakkeelle. Käytännössä tämä tapahtuu niin nopeasti, että käyttäjä ei välttämättä ehdi huomata mitään muuta kuin, että tunnus- ja salasana-kentät tyhjenevät.

Seuraavaksi kirjautuminen.php-tiedostossa luodaan istuntotunnus. Istuntotunnuksen luominen aloitetaan yhdistämällä yhteen muuttujaan seuraavat tiedot: tämänhetkinen aika, käyttäjän käyttäjätunnus, salasana ja käyttäjän tietokoneen osoite, joka selvitetään komennolla getenv(”REMOTE_ADDR”). Tiedot yhdistetään merkkijonoksi ja sijoitetaan $avain-nimiseen muuttujaan,

josta sen jälkeen muodostetaan istuntotunnus komennolla md5($avain). Tämä komento tekee muuttujasta 128-bittisen tunnisteen, jota käytetään istunnon tunnistamiseen. (Heinisuo 2004, 148.)

```
// Rakennetaan merkkijono ja luodaan uusi istuntotunnus
$avain = time() . '$tunnus' . '$salasana' . getenv("REMOTE_ADDR");
$istuntotunnus = md5($avain);
```

Kuvio 3. Istuntotunnuksen luominen.

Seuraavaksi tunniste sijoitetaan käyttäjä-tauluun ja alustetaan eväste. Eväste on eräänlainen tiedosto, joka sisältää istunnon kannalta tärkeät tiedot. Eväste lähetetään käyttäjän selaimelle, joka tallentaa sen muistiinsa. Kun käyttäjä haluaa tehdä jotain sivustolla, joka on käyttäjätunnuksin suojattu, sivusto tarkistaa jokaisen käyttäjältä tulevan pyynnön tai toiminnon yhteydessä, että eväste on voimassa. (Heinisuo 2004, 150.)

Heinisuo käyttää kirjassaan seuraavaa tapaa evästeen luomiseen: setcookie(”istuntotunnus”, ”$istuntotunnus”, 0, ”/”); (Heinisuo 2004, 149.) Kaikilla nykyisillä Internet-selaimilla tämä ei kuitenkaan toimi. Google Chrome –selain ei hyväksy Heinisuon tavalla luotua evästettä, kun taas esimerkiksi Mozilla Firefoxilla eväste toimii moitteitta. Toimivan komentoyhdistelmän rakensin käyttämällä erilaista tapaa määrittää evästeen voimassaoloaika. Eväste luodaan siis koodilla setcookie(”istuntotunnus”, ”$istuntotunnus”, time()+1800);.

Setcookie-komennolle annetaan parametriksi evästeelle haluttu nimi, tässä tapauksessa ”istuntotunnus” ja sisältö, joka on $istuntotunnus-muuttujassa. Lopuksi asetetaan time()-funktiolla evästeen voimassaoloaika. Eväste on voimassa niin kauan kuin käyttäjän selainikkuna on auki tai 30 minuuttia (1800 sekuntia) sen jälkeen, kun käyttäjä on viimeksi tehnyt sivustolla jonkin toiminnon.

4.5 Tietokantayhteys ja etusivu

Kun käyttäjä on kirjautunut onnistuneesti sivustolle, hänet ohjataan etusivulle. Etusivulla käyttäjä näkee omat ja muiden käyttäjien tallentamat tiedostot aikajärjestyksessä, uusin tiedosto ylimpänä. Jotta tiedostot saadaan esitettyä, tulee tarvittavat tiedot noutaa tietokannasta.

Jotta tietokantayhteyttä avatessa ei tarvitsisi joka kerta kirjoittaa yhteyden avaavaa funktiota uudestaan, luodaan erillinen PHP-tiedosto, josta funktiota voidaan kutsua tarvittaessa. Luodaan siis funktiot.php, joka sisältää OpenDB()-funktion:

```
function OpenDB(){

$yhteys = mysql_connect('palvelin','käyttäjätunnus','salasana');

mysql_select_db('tietokannan nimi');

return $yhteys;

}
```

Opinnäytetyön tietoturvasyistä yllä oleva koodi ei ole kirjoitettu sellaisena, kuin se todellisuudessa funktiot.php-tiedostossa on, vaan tunnukset ja muut tiedot on korvattu niiden yleisillä tietotyypeillä. Heinisuo on myös jättänyt yksinkertaisemmassa esimerkissään tietokannan salasanan pois käytöstä (Heinisuo 2004, 121). Salasana tulee kuitenkin tietoturvasyistä olla Internetiin kytketyssä sovelluksessa, joten salasana on yhtenä mysql_connect()-funktion parametrina.

Aina, kun luomaamme OpenDB()-funktiota tarvitaan, sitä voidaan kutsua PHP-tiedoston alussa komennolla require ”funktiot.php”.

Jokaisella kirjautumista vaativalla sivulla tulee myös tarkistaa, että sivun lataavalla käyttäjällä on voimassa oleva eväste selaimellaan. Tässäkin tapauksessa on järkevää luoda erillinen tiedosto, jota kutsutaan PHP-tiedostojen alussa. Luodaan aputoiminnot.php, joka aluksi kutsuu funktiot.php-tiedostoa tietokantayhteyden avaamista varten ja sitten tarkistaa tietokannasta, että käyttäjän tietokanta-taulussa oleva istuntotunnus vastaa evästeen sisältöä. Jos näin ei ole, käyttäjä ohjataan kirjautumislomakkeelle. (Heinisuo 2004, 143-144.)

Index.php eli sovelluksen varsinainen etusivu, alkaa perinteisesti HTML-koodilla, joka tuottaa sivun staattisen sisällön riippumatta käyttäjistä tai heidän tuottamastaan sisällöstä. Etusivulla, kuten kaikilla muillakin sivuilla kirjautumisen jälkeen, ylimpänä esitetään sivuston navigointipalkki, jossa on linkit

”Etusivu”, ”Tallenna uusi tiedosto”, ”Omat postaukset” ja ”Kirjaudu ulos”. Seuraavaksi esitettävä sisältö koostuu käyttäjien lataamista tiedostoista. Ilman käyttäjiä ja heidän lataamiaan tiedostoja sivu olisi tyhjä jo navigointipalkin jälkeen. Navigointipalkki on toteutettu kokonaan HTML/CSS-yhdistelmällä.

Navigointipalkin alla listataan käyttäjien tallentamat tiedostot aikajärjestyksessä, uusin ylimpänä. Tiedostosta näytetään sen saama otsikko, tiedoston tallennushetki (kellonaika ja päivämäärä) ja tallentajan käyttäjätunnus.

Kuvio 4. Etusivu ja kolme uusinta postausta listattuna.

Jotta tiedostot voidaan esittää halutulla tavalla, on tarvittavat tiedot haettava tietokannasta. Luodaan aluksi muuttuja, joka pitää sisällään haussa tarvittavan SQL-kyselyn.

```
if(!$kysely = mysql_query("SELECT tiedostot.id, nimi, tiedostonimi,
    tallentaja, pvm, kayttaja.id , istuntotunnus
        FROM tiedostot, kayttaja
            WHERE tallentaja=kayttaja.tunnus
                ORDER BY pvm DESC", $conn))
```

Kuvio 5. SQL-kyselymuuttujan luonti.

Tietokannasta haetaan tietoja kahdesta taulusta. Nämä taulut ovat ”tiedostot” ja ”käyttäjä”. Tiedostot-taulusta haetaan tiedoston id-numero, nimi (kuvan otsikko), tiedoston nimi, tallentajan tunnus ja ajankohta, jolloin tiedosto on tallennettu. Käyttäjä-taulusta haetaan käyttäjän id-numero ja istuntotunnus. Kyselylausekkeessa haetut tiedot rajataan ja yhdistetään toisiinsa WHERE-ehdolla. Tiedostot-taulussa oleva kenttä ”tallentaja” yhdistetään käyttäjä-taulussa olevaan ”tunnus”-kenttään, jolloin saadaan tunnistettua ja haettua käyttäjän tallentamat tiedostot. ORDER BY pvm DESC -komento järjestää haun tulokset niiden tallentamisjärjestyksessä viimeisimmäksi tallennettu tiedosto ensimmäisenä.

```
while($tiedosto = mysql_fetch_row($kysely)){
// Tämä koodi listaa uusimmat tiedostot.
    echo "<table><tr>";
    echo "<td width=\"365\">
        <a href=\"post.php?id=$tiedosto[0]\">$tiedosto[1]</a></td>";
    echo "<td width=\"180\">".date("H:i M d", strtotime($tiedosto[4])).
        "</td>";
    if($tiedosto[6] == $istuntotunnus)
    {
        // Jos klikkaat omaa tunnusta,
        // näet omat postauksesi ja voit muokata tietoja...
        echo "<td width=\"200\"><a href=\"listaa.php\">" .
            $tiedosto[3] . "</a></td>";
        echo "</tr></table>";
}
else{
    // ... ja jos klikkaat jonkun toisen käyttäjän tunnusta,
    // saat hänen tiedostolistauksensa.
echo "<td width=\"200\"><a href=\"userposts.php?id=" . $tiedosto[5]
    . "\">$tiedosto[3]</a></td>";
echo "</tr></table>";
}}
```

Kuvio 6. Hakutulosten esittäminen.

Haetut tiedot sijoitetaan $tiedosto-nimiseen muuttujaan. WHILE-ehtolause käy läpi hakutuloksia niin kauan kuin tallennettuja tiedostoja löytyy. Tulokset esitetään kolmisarakkeisessa taulukossa, joka on kuitenkin käyttäjälle näkymätön siten, että taulukon reunoja ei näytetä.

Taulukon vasemmassa sarakkeessa oleva tiedoston otsikko on tallennettuna $tiedosto-muuttujan sisäiseen taulukkoon, toiseen indeksiin: $tiedosto[1]. Indeksit alkavat nollasta, joten toinen indeksi on merkinnältään ”1”. Tiedoston otsikosta muodostetaan linkki itse tiedostoon.

Kun käyttäjä klikkaa linkkinä toimivaa tiedoston otsikkoa, lähettää koodi post.php-nimiselle PHP-tiedostolle sen arvon jonka $tiedosto[0]-indeksi sisältää. Se on tässä tapauksessa siis tiedoston yksilöllinen id-numero. Näin voimme post.php-tiedoston avulla muodostaa WWW-sivun, jolla näytetään haluttu tiedosto ja sen kommentit.

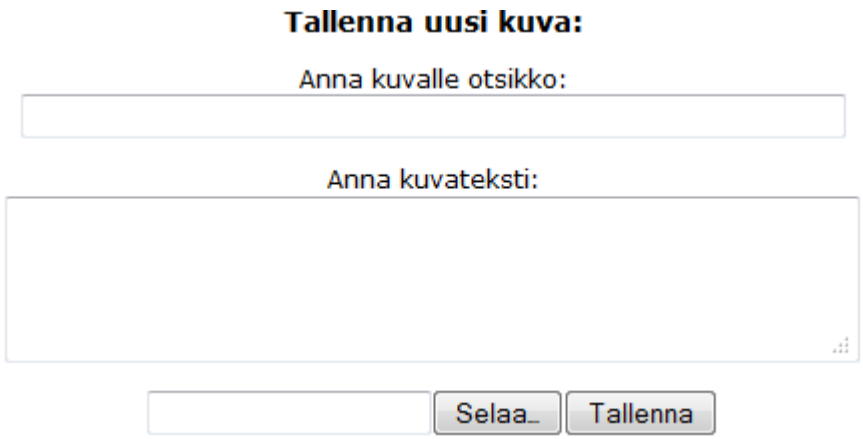
SQL-kyselyssä haetaan käyttäjä-taulusta istuntotunnus, jotta voimme erottaa, milloin käyttäjä klikkaa etusivulla omaa tunnustaan. Istuntotunnusta verrataan evästeessä olevaan istuntotunnus-tietoon ja jos nämä tiedot vastaavat toisi-

aan, tiedämme, että käyttäjä on klikannut omaa tunnustaan. Silloin käyttäjä ohjataan listaa.php-sivulle, joka nimensä mukaan listaa käyttäjän henkilökohtaiset tallennukset omana listanaan. Käyttäjä voi myös muokata tallentamansa tiedoston otsikkoa ja kuvatekstiä, mutta ei itse tiedostoa. Käyttäjä ei siis voi (ainakaan tällä hetkellä) poistaa tallentamiaan tiedostoja. Tämä siksi, että poistetun tiedoston mukana lähtisi mahdollisesti muidenkin käyttäjien tuottamaa sisältöä, kuten tiedoston saamat kommentit.

Jos käyttäjä klikkaa toisen käyttäjän tunnusta, hänet ohjataan userposts.php-sivulle. Tämä sivu listaa valitun käyttäjän tallentamat tiedostot samalla tavalla, kuin post.php-sivulla, mutta tietenkin ilman muokkausmahdollisuutta.

4.6 Uuden tiedoston tallentaminen

Uudet tiedostot tallennetaan sivustolle tallennuslomakkeen avulla. Tallennuslomake.php toimii kahdella eri tavalla: sillä voidaan tallentaa kokonaan uusi tiedosto tai sillä voidaan muokata olemassa olevan tiedoston otsikkoa ja kuvatekstiä. Toimintaperiaate riippuu siitä, mistä tallennuslomakkeelle tullaan.



Kuvio 7. Tallennuslomake.

Jos tallennuslomakkeelle tullaan linkistä ”Tallenna uusi tiedosto”, voidaan lomakkeella tallentaa uusi tiedosto palvelimelle. Jos sen sijaan lomakkeelle tullaan listaa.php-sivulta, joka on siis käyttäjän itsensä tallentamien tiedostojen listaussivu, voidaan olemassa olevan tiedoston tietoja muuttaa.

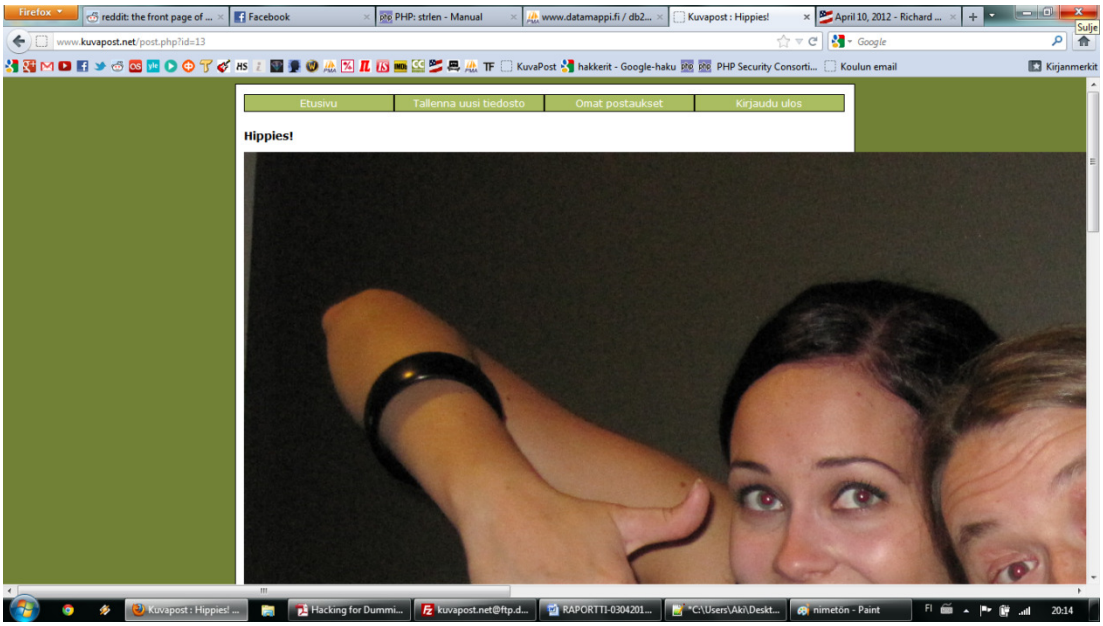
```
if(!isset($_GET['id']))
{
    echo "<input type=file name=\"tiedosto\">";
    echo "<input type=submit value=\"Tallenna\">";
}
else
{
    echo "<input type=submit value=\"Päivitä kuvaus\">";
}
```

Kuvio 8. Käyttäjän päivittäessä tietoja uutta tiedostoa ei voi tallentaa.

Siinä tapauksessa, että lomakkeelle tullaan muokkaamaan olemassa olevan tiedoston otsikkoa ja kuvatekstiä, selaimen osoiteriviltä saadaan tiedoston id-arvo komennolla $_GET[’id’]. Kuvassa seitsemän olevassa if-koodilohkossa tarkistetaan id-arvon olemassaolo osoiterivillä. Siinä tapauksessa, että id:tä ei ole annettu, ”Tallenna”-nappia ei näytetä. Jos id-arvoa ei ole, tiedämme tulleemme sivulle tallentamaan uuden tiedoston.

4.7 Kuvakoon automaattinen muokkaaminen

Nykyaikaisten digikameroiden ja puhelintenkin tallentamat kuvat voivat olla todella suuria, turhankin suuria esitettäväksi sellaisinaan verkkosivuilla. Jos esimerkiksi Samsung Galaxy S –puhelimella maksimiasetuksilla otetun kuvan tallentaa sellaisenaan tälle sivustolle (ilman minkäänlaista muokkausta palvelimenkaan päässä), tuodaan käyttäjän ruudulle mitoiltaan 2560 x 1920 pikselin kokoinen kuva. Tämän kokoinen kuva ei mahdu suurinäyttöisimmänkään kannettavan tietokoneen näytölle (puhumattakaan mobiililaitteista), eikä se varsinkaan mahdu sille varattuun tilaan sivustolla.



Kuvio 9. Ladattu kuva, kun käytössä ei ole ImageMagickia.

Ei olisi käyttäjäystävällistä pyytää käyttäjää itseään muokkaamaan kuvaa, jota tämä on lähettämässä. Hän ehkä ajattelee, ettei kuvan postaaminen ole sen ajan ja vaivan arvoista, mitä kuvan muokkaaminen vaatisi. Kuvaa on siis muokattava automaattisesti palvelimella.

Apachelle on saatavilla ulkopuolisen tahon kehittämä ohjelmisto kuvien monipuoliseen muokkaamiseen. Tämä ImageMagick-ohjelma asennetaan palvelimelle erilliseksi moduuliksi. Toisin kuin useimmissa kuvankäsittelyohjelmissa, ImageMagickissa ei ole graafista käyttöliittymää, vaan ohjelma saa komentonsa joko käyttöjärjestelmän komentoriviltä tai sovelluskehittäjän koodista. (ImageMagick Studio LLC 2012.)

Siinä vaiheessa, kun tässä työssä tuli ajankohtaiseksi ottaa käyttöön kuvakoon muokkaus, oli sivustolle jo hankittu ulkoinen web-hotelli. Valittu web-hotelli tarjoaa asiakkailleen mahdollisuuden käyttää ImageMagickin moduulia kuvien muokkaamiseen. Yksi suuri (vaikkakin ongelman selvittyä pieni) ongelma työn teossa oli ImageMagickin käyttöönottaminen. ImageMagickista on hyvin niukasti kirjallisuutta ja Internetin keskustelupalstoilta oli hankala löytää ohjeistusta siihen, millä tavalla PHP:ssä tulee viitata ImageMagickiin, vai tuleeko siihen viitata ollenkaan. Pelkkä ImageMagickin komentojen liittäminen PHP-koodin sekaan antaa

virheilmoituksen. Webhotellin internetsivuilla ainoa maininta ImageMagickista lukee webhotellin yleisissä käyttöohjeissa, joissa mainitaan ImageMagickin löytyvän hakemistosta ”/usr/local/bin/” (Datamappi 2012). Tästä pystyy päättelemään, että moduuliin todellakin pitää viitata jollain tavalla PHP-koodissa, jotta sen komentoja voidaan käyttää.

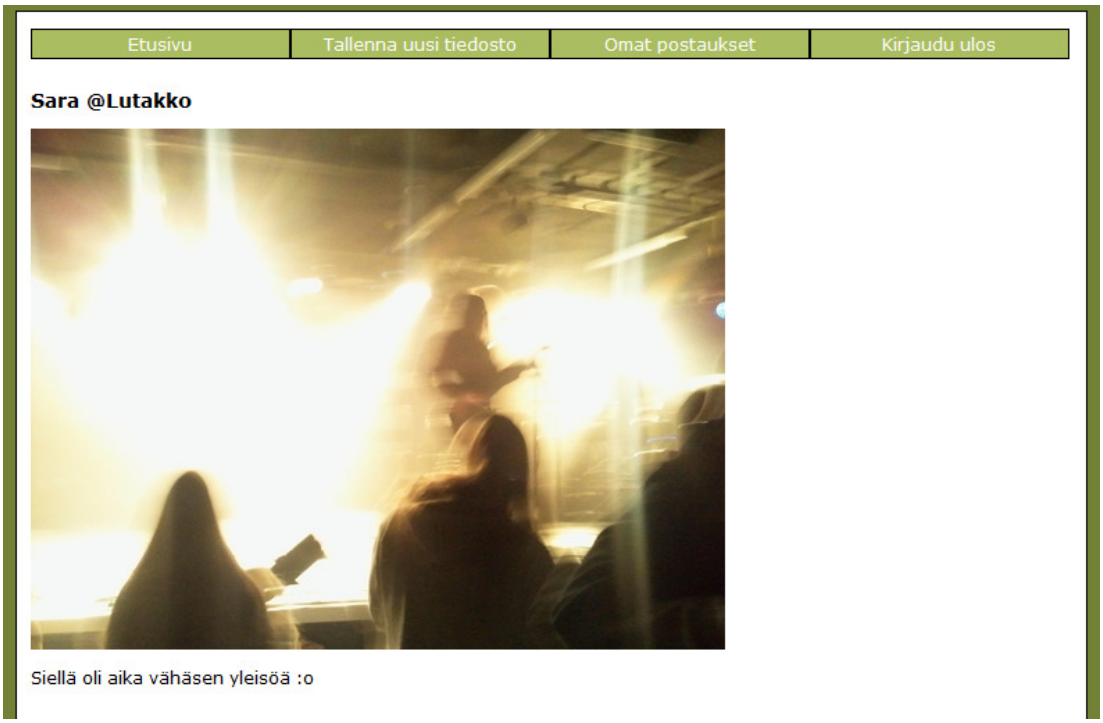
The PHP Group ylläpitää erinomaista PHP-manuaalia kotisivuillaan (The PHP Group 2012a). Tätä selaamalla ja lopulta osuvilla hakusanoilla löytyi oikea komento, jolla ulkoisen moduulin voi ottaa käyttöön. ImageMagickin vähäisestä kirjallisuudesta johtuen toimivan komennon löytäminen kuvakoon pienentämiseksi osoittautui sekin aikaavieväksi. Lopulta, kymmenien yritysten ja erehdysten jälkeen, löytyi tarkoituksenmukainen yhdistelmä komentoja: exec(’/usr/local/bin/convert uploads/’ .$tiedosto[1]. ’ –resize 500x500 uploads/’ .$tiedosto[1]);.

```
$file = $_FILES['tiedosto']['tmp_name'];
list($width, $height, $type, $attr) = getimagesize($file);

if($height > 500)
{
    exec('/usr/local/bin/convert uploads/' .$tiedosto[1]. '
        -resize 500X500 uploads/' .$tiedosto[1]);
}
header("Location: post.php?id=$tiedosto[0]");
```

Kuvio 10. $tiedosto[1]-indeksi sisältää tarkistettavan tiedoston nimen.

Ennen kuvan pienentämistä kannattaa tarkistaa, tarvitseeko kuvaa muokata. On siis selvitettävä ladatun kuvatiedoston koko. Sopiva komento tähän on getimagesize(), jolle annetaan parametrina ladatulla kuvalla alustettu muuttuja (The PHP Group 2012b). Komento antaa taulukkolistauksena tiedot, jotka sijoitetaan $list()-taulukkomuuttujaan: $list($width, $height, $type, $attr) = getimagesize($file);. Näistä tiedoista hyödyllinen tässä työssä on kuvan korkeus eli $height. Tarkistetaan if-lausekkeella, onko $height-arvo yli 500 pikseliä. Jos korkeus on yli 500 pikseliä, käytetään kuvaa ImageMagickissa ja pienennetään se matalammaksi eli 500 pikselin korkuiseksi. Kuvan vääristymisestä ei tarvitse olla huolissaan, sillä ImageMagickin *resize*-komento skaalaa kuvan automaattisesti oikeisiin mittasuhteisiin.



Kuvio 11. ImageMagickin skaalaama kuva. Korkeus: 500px.

Sivuston testausvaiheessa tuli esiin kaksi erikoista ongelmaa, joita en halunnut jättää käyttäjän ratkaistavaksi. Kun käyttäjä latasi sovellukseen kuvan, jonka tiedostonimessä oli skandinaavisia kirjaimia, ei ladattua kuvaa löytynyt tietokannasta, vaikkakin kuva tallentui palvelimelle. Kuvaa ei löytynyt, vaikka sen URL-osoitteen kirjoitti selaimeen.

Syy skandinaavisten kirjainten ongelmaan on kokemuksen perusteella ilmeinen: palvelimen merkistökoodaus on eri, mitä tässä työssä käytetään. Koska seuraavan ongelman ilmetessä oivalsin, että voin ratkaista molemmat ongelmat samalla menetelmällä, en kiinnittänyt merkistökoodauksen eroihin sen enempää huomiota.

Toinen ongelma ilmeni, kun sivustolle yritti tallentaa kuvan, joka oli hyvin suuri ja sisälsi yhden tai useamman välilyönnin tiedostonimessään. Tällöin kuva kyllä tallentui palvelimelle ja sen esittäminenkin oikealla paikallaan onnistui, mutta ImageMagick ei onnistunut skaalaamaan sitä pienemmäksi. Tarkkaa rajaa sille, kuinka suuri kuvan tuli olla ohittaakseen kuvakoon muokkauksen, en selvittänyt, koska sillä ei ollut varsinaista merkitystä. Kuvan muokkaus ei toiminut ja se oli ongelma.

Ratkaisu löytyi PHP:n prep_replace()-funktiosta, joka yksinkertaisesti etsii syötteestä halutut merkit ja korvaa ne toisilla (The PHP Group 2012e). Skandinaaviset merkit korvattiin väliviivalla. Seuraava koodi on tiivistelmä työssä käytetystä koodista (syötteentarkistukset karsittu selkeyden vuoksi):

```
$tiedostonimi = $_FILES['tiedosto']['nimi'];

$uusi_tiedostonimi = preg_replace("/[äöå ]/","-", $tiedostonimi);
```

Funktiolle annetaan hakasulkeiden sisällä etsittävät merkit ja niiden perässä, pilkun jälkeen lainausmerkeissä korvaava merkki. Huomaa esimerkissä skandinaavisten kirjainten jälkeen oleva välilyönti, joka siis oli myös yksi etsittävistä ”merkeistä”.

4.8 Tiedostonkatselusivu

Käyttäjien lataamia kuvia pitää tietysti voida myös katsella. Tätä varten sivustolla on post.php-niminen sivu. Kun linkkinä toimivaa kuvan otsikkoa klikataan muilla sivuilla, tuodaan käyttäjä post.php-sivulle.

Navigointipalkin alapuolella esitetään kuvan otsikko, itse kuva ja sen kuvateksti. Tieto siitä, mikä tiedosto käyttäjälle pitää näyttää, saadaan jälleen osoiteriviltä id-muuttujan arvona $_GET['id']-komennolla.

Käyttäjät voivat kommentoida tai käydä keskustelua kuvien alapuolella yksinkertaisella lomakkeella. Kommentit ja niiden jättäjien tunnukset esitetään kommenttilomakkeen yläpuolella, vanhin kommentti ylimpänä.

Kommenttilomake on HTML-koodilla toteutettu kolmeosainen lomake, joka kooditasolla koostuu neljästä osasta. Aluksi määritellään käyttäjällekin näkyvä tekstinsyöttökenttä. Se on kuusi riviä korkea ja 50 merkkiä leveä tekstialue, mutta käyttäjä voi muokata sen kokoa hiirellään. Seuraavat kaksi osaa lomakkeen koodista eivät tule käyttäjälle näkyviin, vaan ne lähettävät kommentin käsittelevälle PHP-tiedostolle tiedot ”kommentoija” ja ”tiedosto”. Viimeinen, neljäs, osa koodista on lomakkeen tiedot käsiteltäväksi lähettävä nappula. Lomakkeelta eteenpäin lähetettävät tiedot ovat kommentoijan id-numero, tiedoston id-numero ja käyttäjän kirjoittama kommentti.

```
<form name="kommenttilomake" method="post" action="postkomment.php">
    <textarea name="kommentti" rows=6 cols=50></textarea>
    <input type="hidden" name="kommentoija" value="<?php echo $kommentoija[0] ?>">
    <input type="hidden" name="tiedosto" value="<?php echo $id ?>">
    <input type="submit" value="Kommentoi">
</form>
```

Kuvio 12. Kommentointilomakkeen <form>-koodi.

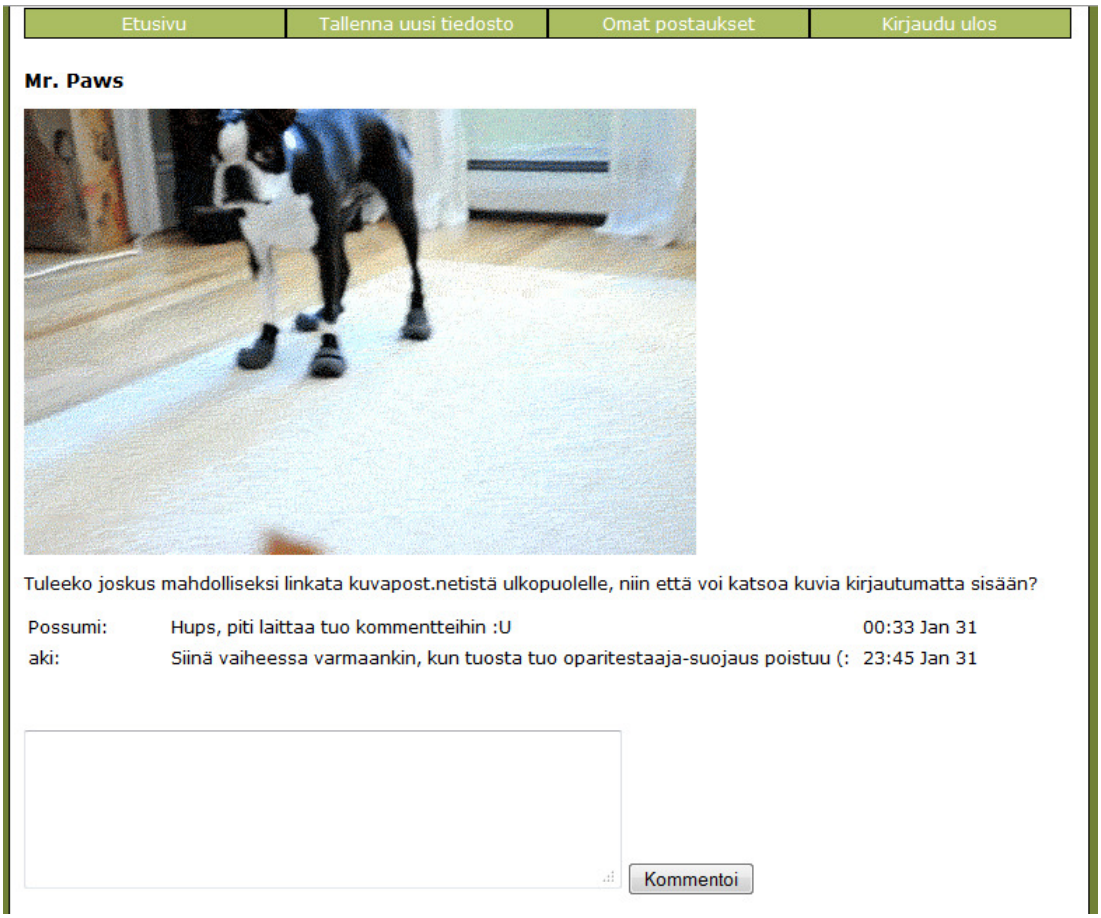
Lomakkeen lähettämät tiedot otetaan vastaan postkomment.php-tiedostoon $_POST['…']-komennolla ja tallennetaan tietokantaan. Kommentit haetaan tietokannasta taas SQL-kyselyllä. On tarkasteltava käyttäjä- ja kommenttitauluja ja keksittävä jokin kommentteja ja käyttäjiä yhdistävä asia, joita verrata SQL:n WHERE-lausekkeessa. Tämä haluamamme yhteys on tietenkin kommentit-taulukkoon tallennettu tieto kommentin jättäjästä eli hänen id-numeronsa.

Poimitaan käyttäjä-taulusta kommentin jättäjän id-numero ja käyttäjätunnus. Kommentit-taulusta haetaan itse kommentti ja sen jättäneen käyttäjän id-numero. Kun tiedot on haettu, suoritetaan niille WHERE-ehto, jolloin saadaan verrattua käyttäjän id_numeroa kommentin jättäneen käyttäjän id-numeroon. AND-vertailulla varmistetaan, että haettava kommentti on sille kuuluvan tiedoston kommentti. Tiedoston id-numerolla, joka saatiin edelliseltä lomakkeelta, haetaan sitä vastaava tieto kommentit-taulusta.

```
if(!$sql = mysql_query("SELECT kayttaja.id, kayttaja.tunnus,
    kommentit.kommentti, kommentit.kommentaattori_id
    FROM kayttaja, kommentit
        WHERE kommentaattori_id = kayttaja.id
            AND tiedoston_id ='$id'", $conn))
```

Kuvio 13. Kommenttitietojen hakeminen tietokannasta.

Kommentit esitetään WWW-sivulla tulostamalla ne taulukkoon samalla tavalla, kuin etusivulle tulostettiin tiedostolistaus.



Kuvio 14. Tällä sivulla käyttäjät voivat katsella ja kommentoida kuvia.

4.9 Uloskirjautuminen

Sivustolta uloskirjautuminen tapahtuu klikkaamalla valikon ”Kirjaudu ulos”-linkkiä. Tällöin kutsutaan poistu.php-sivua, joka tyhjentää evästeen. Tämän jälkeen käyttäjän selaimelta puuttuvat ne tarvittavat tiedot, joita se tarvitsee näyttääkseen sivuston suojattua sisältöä.

```php
<?php
    require "aputoiminnot.php";
    $istuntotunnus = $_COOKIE['istuntotunnus'];

    setcookie("istuntotunnus", "");
    header("Location: http://localhost/kuvapost/index.php");
    exit;
?>
```

Kuvio 15. Uloskirjautumisen toteuttava koodi.

Evästeen tyhjentämisen jälkeen käyttäjä ohjataan index.php-sivulle, josta ilman voimassa olevaa evästettä olevat käyttäjät ohjataan kirjautumislomakkeelle. Toki käyttäjän voi ohjata suoraan kirjautumislomakkeellekin.

5 TIETOTURVA

5.1 SQL-injektiot ja niiden torjuminen

Ilman minkäänlaista käyttäjältä saadun datan tarkistusta verkkoportaali olisi avoinna hakkereiden hyökkäyksille. Lähiaikoina on ollut useita uutisointeja tapauksista, joissa hakkeroimalla on saatu ja sitten levitetty tuhansien ihmisten luottokortti- ja käyttäjätietoja salasanoineen (CERT-FI 2012). Portaalia kehitettäessä tulee olla mahdollisimman tarkka siitä, että käyttäjiltä vastaanotettavat tiedot ovat turvallisia. Ne eivät saa sisältää tietokannan vaarantavia komentoja, kuten SQL-injektioita. (PHP Security Consortium 2012a.)

Kun Internetiin kytketyt WWW-sovellukset ottavat käyttäjältä vastaan syötteitä, ovat ne alttiita hakkereiden vihamielisille tiedonhakuyrityksille. Käyttäjien tallentamat – usein luottamukselliset – tiedot on suhteellisen helppo tuhota tai kopioida, jos sovelluksen kehittäjä ei ole ennakoinut tällaisia yrityksiä. (Heinisuo-Rauta 2007, 399-400.)

SQL-injektio (engl. SQL-injection) on hakkereiden käyttämä tekniikka, jolla he sovelluksen tietoturva-aukkoja hyväksikäyttäen syöttävät tietokantakomentoja päästäkseen tietokantaan käsiksi tavalla, jota sovelluskehittäjä ei ole tarkoittanut. Tällaisia tietoturva-aukkoja voivat olla mm. tarkistamatta jätetyt syötteet käyttäjiltä tai palvelimen antamat virheilmoitukset. (Beaver 2010, 311.)

Datamappi-webhotelli on poistanut käytöstä palvelimen antamat virheilmoitukset, joten niistä ei tässä työssä tarvitse huolehtia. Työssä kuitenkin otetaan vastaan käyttäjien vapaasti kirjoittamaa tekstiä ja se on mahdollinen tietoturvariski.

Kun oppikirjallisuudesta ja Internetistä hakee tietoa SQL-injektioiden torjumisesta, yksi asia toistuu niissä jokaisessa: datan suodattaminen (engl. data filtering). Käyttäjän syöttämät tiedot on ehdottomasti tarkistettava ja puhdistettava mahdollisista SQL-lausekkeista ennen niiden syöttämistä tietokantaan (Heinisuo-Rauta 2007, 400).

Heinisuo ja Rauta käyttävät kirjassaan PHP:n omaa mysql_real_escape_string()-funktiota torjuakseen SQL-injektioita. Tämä funktio tarkastaa käyttäjältä saadun datan SQL-koodin varalta ja muokkaa sen

vaarattomaan muotoon. Yleisin funktion tekemä toimenpide on yksittäisten heittomerkkien korvaaminen turvallisemmalla merkintätavalla. Kun SQL-komennoissa hyvin yleisen ’-merkin eteen lisätään mysql_real_escape _string()-funktion avulla kenoviiva, ei hakkerin kirjoittama SQL-lauseke enää toimi halutulla tavalla. (Heinisuo-Rauta 2007, 400.) Tässä opinnäytetyössä tämä tarkoittaa sitä, että jos hakkeri yrittää kirjoittaa esimerkiksi kuvatiedostojen kommenttikenttiin SQL-komentoja, nämä komennot muokataan tehottomiksi ja tallennetaan kommenttitauluun kuin mikä tahansa, normaali kommentti. Tässä työssä on suodatettu kaikki lomakkeilta tai selaimen osoiterivil-tä saatadut tiedot mysql_real_escape_string()-funktion läpi.

5.2 Cross-Site Scripting (XSS)

SQL-injektion lisäksi yksi tunnetuimmista WWW-sovelluksien haavoittuvuuk-sista on cross-site scripting, XSS (Beaver 2010, 290). Aikaisemmassa kap-paleessa puhdistimme käyttäjiltä tulevat syötteet SQL-komentojen varalta, mutta se ei riitä XSS-hyökkäyksen torjumiseksi. Cross-site scripting mahdol-listaa mm. luvattoman JavaScript- ja VBScript (Visual Basic Scripting Edition) –koodien syöttämisen ja ajamisen verkkosivuilla. Tämä on vaarallista varsin-kin hyökkäyksen kohteena olevan sovelluksen käyttäjille. Onnistunut XSS-hyökkäys voi antaa hyökkääjälle pääsyn sivuston käyttäjän koneelle, koska tällöin hyökkäys tulee luotettavaksi oletetusta lähteestä eli verkkoselaimesta. (Beaver 2010, 290.)

Hyvä keino estää XSS-hyökkäys on suodattaa syötteet vielä yhden funktion läpi. Tähän soveltuu PHP:n funktio htmlentities() (PHP Security Consortium 2012b). Toinen vaihtoehto on htmlspecialchars() (The PHP Group 2012d). Molemmat funktiot korvaavat tarkistettavasta syötteestä XXS-hyökkäyksessä tarvittavia erikoismerkkejä niiden turvallisemmilla vastineilla, mutta htmlenti-ties() on kattavampi (The PHP Group 2012c).

Tarkastellaan esimerkin kautta, mitä htmlentities()-funktio tekee. Kuvapostin kommentointilomakkeeseen voi syöttää 300 merkin pituisen tekstin. Siihen määrään mahtuu jo kohtuullinen määrä mahdollisesti vahingollista skriptaus-ta. On siis suodatettava lomakkeelta tuleva kommentti htmlentities()-funktion avulla näin: $kommentti = htmlentities($kommentti);.

Testataan lomakkeen toimintaa suodatuksen jälkeen perinteisellä ”Hello World” -pätkällä kirjoittamalla kommenttikenttään JavaScriptillä seuraava koodi:

<script> document.write('Hello World'); </script>

Suodatusfunktio käy $kommentti-muuttujan sisällön läpi ja tallentaa HTML- ja skripti-tagit tietokantaan vaarattomin merkein. Edellinen koodi tallentuu tietokantaan seuraavasti:

<script> document.write(\'Hello World\'); </script>

Kun edellinen syöte haetaan tietokannasta ja esitetään käyttäjälle kommenttina, näkyy se melkein sellaisena, kuin se kommenttikenttään kirjoitettiinkin. Aikaisemmin käyttöönotettu mysql_real_escape_string() on lisännyt yksittäisten heittomerkkien eteen kenoviivat, eikä koodi muutenkaan toimi enää XSS-hyökääjän yrittämällä tavalla.

5.3 Salasanan suojaaminen

Salasanan turvallisuuden varmistamiseksi tässä työssä käytetään kahta tietoturvamenetelmää. Käyttäjän luodessa uuden käyttäjätunnuksen hänen salasanaansa ei tallenneta tietokantaa sellaisenaan. Valittuun salasanaan lisätään palvelimella ennalta määrätty, vaikeasti arvattavissa oleva merkkijono. Heinisuo ja Rauta käyttävät tällaisena merkkijonona ”kalenteri”-sanaan perustuvaa ”#k@lenter!”-merkkijonoa (Heinisuo-Rauta 2007, 268). Tietoturvasyistä en paljasta tässä raportissa, mikä Kuvapostissa käytetty merkkijono on. Tarkastellaan tapausta siis kuvitteellisen merkkijonon kanssa. Valittu merkkijono lisätään käyttäjän henkilökohtaisen salasanan perään PHP:n funktiolla concat(), jolloin käyttäjän salasana on muodossa ”salasana#k@rv0n3n”. Näin salasana on huomattavasti turvallisempi (Heinisuo-Rauta 2007, 279).

Kun merkkijono on lisätty käyttäjän antaman salasanan perään, on viesti vielä syytä salata niin sanotuksi viestitiivisteeksi ennen tietokantaan tallentamista. Tietokantaan ei missään vaiheessa tallenneta salasanamerkkijonoa selkokielisenä. (Heinisuo-Rauta 2007, 265.) Käytin salasanan viestitiivisteen luomiseksi SHA-1-algoritmia. Sen tuottama viestitiiviste on aina 40 merkkiä

pitkä riippumatta käyttäjän valitseman salasanan pituudesta ja tietokantaan tallennetaan tämä versio salasanasta (Heinisuo-Rauta 2007, 266). Tarkemmin sanottuna tallennettava tiiviste ei ole edes versio salasanasta vaan algoritmin siitä tuottama tarkistussumma (Heinisuo-Rauta 2007, 265). Merkkijonosta ”salasana#k@rv0n3n” muodostettu SHA-1-tiiviste näyttää seuraavalta: add695566c5ec9cfd429eb115057f1cfa53898a3.

```
if(!$kysely = mysql_query("INSERT INTO kayttaja
    (tunnus, salasana)
        VALUES
        ('$tunnus', sha1(concat('$salasana','[redacted]')))", $conn))
```

Kuvio 16. Viestitiiviste muodostetaan salasanan ja salaisen merkkijonon yhdistelmästä.

Kun käyttäjä tunnusten luomisen jälkeen kirjautuu sivustolle, suoritetaan saman algoritmin avulla tarkistus, jossa verrataan tietokantaan tallennetun viestitiivisteen yhteensopivuutta sen salasanan kanssa, jolla käyttäjä nyt yrittää kirjautua sisään. Jos tiivisteet vastaavat toisiaan, käyttäjä on antanut oikean salasanan.

```
if(!$kysely = mysql_query(
"update kayttaja set istuntotunnus='$istuntotunnus'
    where md5(tunnus) = '$tunnus' and
    salasana=sha1(concat('$salasana','[redacted]')) ", $conn))
```

Kuvio 17. Tarkistetaan, että tietokannan salasanasarake vastaa $salasana-muuttujasta luotua tiivistettä.

5.4 Muita tietoturvaratkaisuja

Datan suodattamisen lisäksi tämän työn teon aikana, teknisen ymmärryksen karttuessa, tuli useita pieniä ideoita tietoturvan parantamiseksi. Tallennettavan tiedoston oikeellisuus eli varmistus siitä, että käyttäjältä vastaanotettava tiedosto on sallittu kuvatiedosto, on yksi tärkeistä oivalluksista. Tällä varmistetaan se, ettei käyttäjä lataa palvelimelle esimerkiksi haittaohjelmaa tai virusta.

Jo lomakkeiden suunnitteluvaiheessa jokaisella lomakekentällä oli jokin tietynmittainen rajoitus sille, kuinka monta merkkiä kenttään syötettävä teksti

voi enintään sisältää. On petollisen helppoa tyytyä johonkin tietoturvaratkaisuun ajattelematta asiaa syvemmin. Voisi pinnallisesti ajatella, että lomakkeen HTML-koodiin kirjoitettu maxlength=”20” riittäisi varmistamaan, ettei käyttäjä todellakaan pysty lähettämään lomakkeelta kahtakymmentä merkkiä pidempää syötettä. Lähemmällä tarkastelulla huomaa kuitenkin sen tosiasian, että yleisesti jokaiselle käyttäjälle tarjolla oleva, HTML-koodilla toteutettu lomake, on vain tietokannasta täysin erillään oleva käyttöliittymä. Lomakkeen lähdekoodi on jokaisen Internet-käyttäjän nähtävillä, eikä hakkerin tarvitse tehdä muuta kuin kopioida lähdekoodi omaan tekstieditoriinsa. Siinä hän voi muuttaa maxlength-arvoa ja action-osoitetta ja sen jälkeen käynnistää muokattu lomaketiedosto nettiselaimessa. Tämän muokatun lomakkeen kautta hakkeri voi ohittaa rajoitukset, jotka sovelluskehittäjä on asettanut.

On siis viisasta tarkistaa lomakkeelta vastaanotettava syöte myös sen pituuden varalta, vaikkakin muita varotoimia on otettu käyttöön

6 POHDINTA

Opinnäytetyön valmistuttua tähän vaiheeseen huomasin, että olen saavuttanut ne tavoitteet, jotka työlleni asetin. En ollut aiemmin käyttänyt MySQL-tietokantaa tai tehnyt verkkosivuja, jotka hyödyntäisivät tietokantaa, mutta nyt minulla on niiden tekemiseen vaadittavat taidot. Näitä taitoja haluan myös kehittää jatkossa.

Työn edistymisen ohella tietoturvanäkökulma korostui hyvin selkeästi. Lähdemateriaaleja lukiessa tuli selväksi, että sovelluskehittäjän tulee olla hyvin skeptinen jokaista käyttäjää ja toimintoa kohtaan. Jokaisen uuden kehitysidean kohdalla on pohdittava mahdollisia keinoja, joilla hyökkääjä voisi käyttää sovellusta ja sen ominaisuuksia vihamielisesti. Koskaan ei voi olla, eikä kannata olla, täysin varma sovelluksen turvallisuudesta, mutta johonkin on myös vedettävä raja. On luotettava omaan harkintakykyyn, mutta myös pysyttävä ajan tasalla mahdollisista tietoturvauhkista.

Toivon, että tästä raportista olisi hyötyä jokaiselle, joka haluaa toteuttaa tietokantapohjaisen verkkoportaalin. Olen pyrkinyt raportoimaan tärkeimmät työn osa-alueet siinä järjestyksessä, kuin ne on järkevintä ottaa huomioon ja toteuttaa. Tietoturvan käsittelin omana kappaleenaan, mutta tietoturva tulee olla mielessä kehitystyön jokaisessa vaiheessa suunnittelusta valmiiseen tuotteeseen saakka.

Työtä tehdessäni aloin keskittyä enemmän muiden käyttämieni WWW-sivustojen tekniikoihin. Web 2.0 sisältää erittäin paljon mahdollisuuksia kehittää aina vain parempia yhteisöllisiä ominaisuuksia WWW-sivuille. Aikomukseni on jatkaa työn kehittelyä oman ammattitaitoni kehittämiseksi ja ylläpitämiseksi. Jos sivusto alkaisi menestyä, olisi se eräänlaisen unelman täyttyminen.

LÄHTEET

Apache Software Foundation 2011. Apache HTTP Server Project. The number one HTTP server on the Internet. Osoitteessa httpd.apache.org. 6.11.2011

CERT-FI 2012. Vuosikatsaus 2011. Osoitteessa http://www.cert.fi/katsaukset/2011/vuosikatsaus2011.html 7.3.2012.

Datamappi 2012. Ohjeita webhotellin tehokkaaseen käyttöön. http://www.datamappi.fi/kayttoohjeita.php 8.4.2012.

Heinisuo, R. 2004. PHP ja MySQL. Tietokantapohjaiset verkkopalvelut. Jyväskylä: Gummerus Kirjapaino Oy.

Heinisuo, R. – Rauta, I. 2007. PHP ja MySQL. Tietokantapohjaiset verkkopalvelut. Gummerus Kirjapaino Oy.

Hovi, A. – Huotari, J. – Lahdenmäki, T. 2005. Tietokantojen suunnittelu & indeksointi. Jyväskylä: Docendo Finland Oy.

ImageMagick Studio LLC 2012. About ImageMagick. Osoitteessa http://www.imagemagick.org/script/index.php 9.4.2012.

Negrino, T. – Smith, D. 2007. JavaScript. Rakenna dynaamisia verkkosivuja. Jyväskylä: Gummerus kirjapaino Oy.

Oikarinen, J. IRC History by Jarkko Oikarinen. Osoitteessa http://www.ircnet.org/History/jarkko.html 6.11.2011.

O'Reilly, T. 2011 What is Web 2.0. Design Patterns and Business Models for the Next Generation of Software. Osoitteessa http://oreilly.com/pub/a/web2/archive/what-is-web-20.html?page=1 6.11.2011.

PHP Security Consortium 2012a. PHP: Security Guide: Overview. Osoitteessa http://phpsec.org/projects/guide/1.html 11.4.2012.

PHP Security Consortium 2012b. Form Processing: Cross-Site Scripting. Osoitteessa http://phpsec.org/projects/guide/2.html 11.4.2012.

The PHP Group 2012a. PHP Manual. Osoitteessa http://php.net/manual/en/index.php 8.4.2012.

The PHP Group 2012b. PHP Manual. Getimagesize. Osoitteessa http://php.net/manual/en/function.getimagesize.php 8.4.2012.11

The PHP Group 2012c. PHP Manual. Htmlentities. Osoitteessa http://www.php.net/manual/en/function.htmlentities.php 11.4.2012.

The PHP Group 2012d. PHP Manual. Htmlspecialchars. Osoitteessa http://php.net/manual/en/function.htmlspecialchars.php 11.4.2012.

The PHP Group 2012e. PHP Manual. Preg_replace. Osoitteessa http://fi.php.net/manual/en/function.preg-replace.php 11.4.2012.

The PHP Group 2011. What is PHP? Osoitteessa php.net/manual/en/intro-whatis.php. 6.11.2011.

World Wide Web Consortium 2011. What is CSS? Osoitteessa http://www.w3.org/Style/CSS/. 6.11.2011.

World Wide Web Consortium 2011. What is HTML? Osoitteessa http://www.w3schools.com/html/html_intro.asp. 6.11.2011.

LIITEET

Sivukartta Liite 1