TAMPEREEN
AMMATTIKORKEAKOULU

OPINNÄYTETYÖ

AJANVARAUS- JA ASIAKASHALLINTAJÄRJESTELMÄ WWW-SOVELLUKSENA

Janne Kuula

Tietojenkäsittelyn koulutusohjelma
kesäkuu 2008
Työn ohjaaja: Petri Heliniemi

TAMPERE 2008

TAMPEREEN
AMMATTIKORKEAKOULU

Tekijä(t)	Janne Kuula
Koulutusohjelma(t)	Tietojenkäsittely
Opinnäytetyön nimi	Ajanvaraus- ja asiakashallintajärjestelmä WWW-sovelluksena
Työn valmistumis-kuukausi ja -vuosi	kesäkuu 2008
Työn ohjaaja	Petri Heliniemi

Sivumäärä: 44

TIIVISTELMÄ

Tämän opinnäytetyön lähtökohtana oli tarve sellaisen WWW-sovelluksen kehittämiselle, joka toimisi asiakashallintaominaisuuksia sisältävänä ajanvarausjärjestelmänä. Työn toimeksiantaja on perustamisvaiheessa oleva hierontayritys, jolla ei vielä ole nimeä. Yritys tulee tarvitsemaan järjestelmää asiakas- ja aikavaraustietojen sähköiseen säilytykseen sekä Internetissä toimivan ajanvarauspalvelun tarjoamiseen asiakkailleen. Järjestelmän on tarkoitus vähentää puhelimitse tehtävien aikavarausten määrää ja helpottaa hierojien työtä myös asiakastietojen ja aikavarausten hallinnan osalta.

Tässä raportissa käsitellään järjestelmän rakentamisessa käytettyjä tekniikoita ja niihin liittyviä erityistä huomiota vaativia asioita, kuten standardeja ja tietoturvaa sekä yleisesti, että kyseisen järjestelmän osalta. Myös sovelluksen toiminnallisuutta ja käyttöliittymän elementtejä esitellään omassa luvussaan.

Toimeksiantona tuotetun sovelluksen toteutuksessa on käytetty PHP:tä varsinaisen toiminnallisuuden luomiseen olio-ohjelmointia hyödyntäen, MySQL-tietokantaa tietovarastona sekä JavaScriptiä ja sitä käyttävää Ajax-tekniikkaa käytettävyyden parantamiseen. Järjestelmässä käyttäjät tunnistetaan käyttäjätunnuksen ja salasanan avulla, ja asiakkaat voivat järjestelmään kirjauduttuaan varata ja perua hieronta-aikoja sekä muuttaa itsestään tallennettuja tietoja. Työntekijöille on annettu mahdollisuus käyttäjätilien, omien työaikojen sekä aikavarausten lisäämiseen, poistamiseen ja muokkaamiseen.

Järjestelmä ei kuitenkaan ole tällaisenaan aivan valmis käyttöönotettavaksi, sillä ulkoasusuunnittelu on päätetty toteuttaa erikseen, ja järjestelmä tullaan vasta myöhemmässä vaiheessa integroimaan osaksi yrityksen tulevaa WWW-sivua.

Avainsanat PHP Tietokannat Standardit Tietoturva

TAMPERE
POLYTECHNIC

Author(s)	Janne Kuula
Degree Programme(s)	Business Information Systems
Title	Web application for appointment and customer management
Month and year	June 2008
Supervisor	Petri Heliniemi

Pages: 44

ABSTRACT

The starting point for this thesis was the need for developing a Web application that would serve as an appointment management system and contain certain customer management features. The client in need of the system is a yet-to-be-founded massage business, which as of yet does not have a name. The system is needed for storing information about customers and appointments electronically and for offering an online appointment-making service. The application is intended to reduce the amount of appointments made by phone and ease the workload of the massage therapists also for the part of managing the customer and appointment information.

This report describes the techniques used to build the system, as well as the matters related to them that require special attention, such as standards and software security, both universally and in the case of the particular system. The functionality of the application and the elements of the user interface are also presented in a chapter of their own.

The implementation is made using PHP and its object-oriented programming features for the actual functionality, MySQL database for data storage and JavaScript and Ajax for improving the usability. The system users are identified by usernames and passwords and upon signing in customers may make and cancel appointments as well as view and change the information that is stored about them. The employees are given tools for creating, deleting and modifying user accounts, own work hours and customers' appointments.

The system cannot be put into use right away, however, because the layout design was decided to be carried out separately from the technical development, and ultimately the system will be integrated as a part of the future website of the client.

Keywords PHP Databases Standards Software security

Sisällysluettelo

Lyhenteet ja käsitteet

XML	Extensive Markup Language on kehitetty tiedon organisoiduksi merkintätavaksi, ja on yleinen tiedonsiirtomuoto eri ohjelmien välillä, koska on standardinmukainen ja yksiselitteinen tiedon esitysmuoto.
HTML/XHTML	Hypertext Markup Language ja Extensible Hypertext Markup Language ovat sivunkuvauskieliä, joilla voidaan luoda WWW-sivuja. XHTML on HTML:n seuraaja, ja se noudattaa XML:n muotomäärityksiä.
UTF-8	Unicode-merkistöstandardin tavumäärältään vaihtelevanmittainen koodausmuoto.
CSS	Cascading Style Sheets on dokumenttien ulkoasun määrittelyyn kehitetty standardoitu merkintätapa.
Skriptauskieli	Ohjelmointikieli, jota käytetään ohjaamaan varsinaista sovellusohjelmaa, kuten palvelinohjelmistoa tai Internet-selainta. Skriptit voivat olla lyhyitäkin koodinpätkiä, ja poiketen varsinaisista sovelluksista niitä ei tarvitse erikseen kääntää ennen suorittamista.
PHP	Vuonna 2008 maailman yleisimpien ohjelmointikielten joukkoon kuuluva palvelinpuolen skriptauskieli.
JavaScript	Selaimissa käytetty skriptauskieli.
Ajax	JavaScriptiin liittyvä tekniikka, jota käytetään tiedon hakemiseen palvelimelta selaimeen ilman kokonaan uuden sivun lataamista.
Muuttuja	Ohjelmointikielissä käytetty viittaustapa tilapäisesti tallennettuun tietoon. Muuttujaan voi olla tallennettuna esimerkiksi numero, merkkijono, taulukko tai olio.
Funktio	Ohjelmointikielissä käytetty rakenne, joka pitää sisällään tietyn toistuvasti käytettävän toiminnallisuuden.
Metodi	Luokan sisältämä funktio.
Parametri	Funktiokutsussa funktion käyttöön annettava muuttuja.
SQL	Structured Query Language on tietokantoja varten kehitetty kyselykieli, jota käyttäen voidaan tietokannan tietoja lisätä, muuttaa ja poistaa.
MySQL	Ilmainen tietokantaohjelmisto, joka on yleinen etenkin kevyessä WWW-käytössä.
SSL/TLS	Secure Sockets Layer ja Transport Layer Security ovat tietoliikenteen suojaamiseen käytettyjä protokollia.

1 Johdanto

Sähköisestä asioinnista Internetissä on jo ainakin monille suomalaisille tullut jokapäiväistä tietokoneiden ja laajakaistayhteyksien lisääntyessä. Useat palvelut ovat käytettävissä verkkosivuilta käsin ympäri vuorokauden ja ilman jonottamista. Myös aikavarausten tekeminen yritysten palveluihin Internetin kautta on yleistymässä, ja etenkin aloittavan yrityksen, joka aikoo pitää kirjaa asiakkaistaan ja tarjota näille aikavarauksia palveluihinsa, on syytä vakavasti harkita jonkinlaista sähköistä järjestelmää tietomäärän ylläpitoon ja aikavarausten vastaanottoon.

Kuvaus sopii opinnäytetyöni toimeksiantajaan, joka on perustamisvaiheessa oleva hierontayritys. Perustajana toimii vuonna 2007 valmistunut koulutettu hieroja Leevi Suominen. Yritystä ei kuitenkaan ole vielä tätä kirjoitettaessa perustettu, eikä sillä näin ollen ole vielä nimeä. Yritys tulee ainakin aluksi olemaan pieni, mutta Internetissä käytettävän ajanvaraus- ja asiakashallintajärjestelmän on arveltu olevan yritykselle hyödyllinen. Tällaisen järjestelmän on ajateltu helpottavan hierojien työtä etenkin siinä mielessä, ettei heidän tarvitse vastaanottaa kaikkia aikavarauksia puhelimitse. Tämä on olennainen asia, koska varsinkaan asiakkaita hierottaessa ei puhelimeen vastaaminen onnistu. Ympärivuorokautinen Internetissä toimiva varaus- ja peruutuspalvelu tuo myös joustavuutta asiakkaan näkökulmasta. Lisäksi järjestelmän on tarkoitus auttaa työntekijöitä saamaan selkeä kuva tulevista varauksistaan sekä eri asiakkaiden käyntihistoriasta.

Toimeksiantona siis suunnittelin ja tein käytännön toteutuksen yrityksen tulevien WWW-sivujen kautta käytettävästä järjestelmästä, jonka avulla yrityksen asiakkaat voivat varata itselleen hieronta-aikoja sekä työntekijät hallinnoida asiakastietoja ja työaikojaan. Tässä raportissa käsitellään järjestelmän ohjelmointia koskevia asioita teknisestä näkökulmasta, kerrotaan järjestelmän käyttämistä tekniikoista yleisesti sekä pohditaan standardien ja tietoturvan merkitystä ja niiden huomioonottomahdollisuuksia käytännössä WWW-sovellusten kehittämisessä. Rakennetun järjestelmän perustoiminnallisuuden esittely sisältyy myös opinnäytetyön sisältöön, mutta sen ulkopuolelle on rajattu yrityksen muun WWW-sivuston tekemisen sekä käyttöliittymän käytettävyyteen ja ulkoasuun liittyvän suunnittelun kuvaaminen. Järjestelmää ei tähän työhön liittyvän toimeksiannon puitteissa myöskään liitetä tulevalle WWW-sivustolle, vaikka tämä onkin tarkoitus toteuttaa lähitulevaisuudessa.

Järjestelmä on toteutettu käyttäen palvelinpuolen ohjelmointikielenä suosittua olio-ohjelmointinsa uusinutta PHP 5:ttä, ja tämän yhteydessä on hyödynnetty JavaScriptin kautta käytettävää vasta viime vuosina nopeasti yleistynyttä Ajax-tekniikkaa, jonka avulla voidaan tuoda uudenlaista mukautuvuutta WWW-sivujen toiminnallisuuteen. Olennaisena osana järjestelmää on käytetty myös MySQL-tietokantaa. Työ tarjoaa näin ollen yhden käytännöllisen esimerkin siitä, mitä näillä ilmaisilla avoimen lähde-

koodin ohjelmilla ja tekniikoilla voidaan saada aikaiseksi yhden opiskelijan rajallisin resurssein.

Aiheen valitsin siksi, että kyseiselle järjestelmälle oli oikea tarve ja sen tekeminen vaikutti kiinnostavalta haasteelta sekä mahdollisuudelta syventää tuntemustani WWW-sovellusten kehittämisessä käytettävistä tekniikoista. Taustalla oli myös ajatus ajanvarausjärjestelmän markkinoinnista ja soveltamisesta muidenkin yritysten käyttöön, mikäli itse perustaisin oman yrityksen tulevaisuudessa.

Opinnäytetyön tekemisessä olen käyttänyt useita aihepiiriä käsitteleviä sekä suomen- että englanninkielisiä kirjoja. Käyttökelpoisia kirjoja löytyi suhteellisen helposti, mutta valtaosa niistä oli tarkoitettu tekniikoiden yleiseen opiskeluun, joten materiaalin soveltaminen käytännön toteutukseen oli työlästä. Pyrin mahdollisimman uusien kirjojen käyttöön, ja vanhin työn lähteenä käytetty kirja on vuodelta 2002. Joka aihepiiristä olen kuitenkin käyttänyt teoksia, jotka ovat vuodelta 2004 tai uudempia, ja vanhemmat kirjat ovat olleet vain näitä soveltuvin osin tukemassa.

Yhden tekniikoita käsittelevän suomeksi kirjoitetun kirjan jouduin hylkäämään epäluotettavana, koska havaitsin sen sisältävän yleisistä käytännöistä poikkeavaa termistöä, epätieteellistä tyyliä ja jopa suoranaisia virheitä. Tämä kirja oli Kauko Kolehmaisen kirjoittama PHP & MySQL – Teoriasta käytäntöön vuodelta 2006, ja esimerkkeinä kyseisistä seikoista mainittakoon toisiaan vastaamattomat HTML-elementin aloitus- ja lopetustunnisteet esimerkissä heti sivulla 4, hymiö sivulla 375 sekä epäyhtenäiset koodiesimerkit, joista suuri osa ei lainkaan noudata standardeja eikä suosituksia, kuten esimerkki kirjan sivulla 298. Tätä kirjaa siis ei ole käytetty lähteenä tässä opinnäytetyössä. Etenkin käyttämissäni englanninkielisissä teoksissa laatutaso oli kuitenkin mielestäni korkea.

Toimeksiantona tehtyyn järjestelmään voi kirjautua sisään käyttäjätunnuksilla. Yrityksen asiakkaat saavat hieronnassa käydessään omat tunnukset, minkä jälkeen tämä voi varata itselleen aikoja Internetin kautta. Työntekijä siis rekisteröi asiakkaita järjestelmään, ja heistä voidaan tallentaa järjestelmään erilaisia tietoja. Tiedot asiakkaiden käynneistä säilyvät järjestelmässä, ja tällä tavoin asiakkaille muodostuu hoitohistoria, jota hierojat sekä asiakas itse pääsevät halutessaan tutkimaan. Järjestelmän oletuskielenä on suomi, mutta toteutus on tehty myös englanninkielisenä, ja käyttäjälle tarjotaan kielenvaihtomahdollisuus.

Opinnäytetyön tavoitteena on selvittää toimeksiannon mukaisen ajanvaraus- ja asiakashallintajärjestelmän rakentamisessa tarvittavien tekniikoiden oikeaoppisia käyttötapoja tietoturvan sekä yhteensopivuuden optimoimiseksi.

2 Käytetyistä tekniikoista

Tässä luvussa käsitellään toimeksiantona kehitetyn WWW-sovelluksen toteutuksessa käytettyjä tekniikoita yleisellä tasolla sekä niihin liittyviä standardeja. WWW-kehitykseen liittyviä tietoturva-asioita tuodaan myös esille omassa alaluvussaan.

2.1 WWW-merkintäkielet

WWW-sivujen tekemiseen on kehitetty tietyt merkintätavat, joita selainohjelmat tulkitsevat näyttäen käyttäjälle koodin perusteella luodun visualisoidun näkymän. Tämä alaluku käsittelee staattisten WWW-sivujen tekemisessä käytettyjä merkintäkieliä.

2.1.1 Sivunkuvauskielet ja standardit

Hypertext Markup Language (HTML) on merkintäkieli, josta WWW-sivut muodostuvat. Internet-selaimet tulkitsevat tätä kieltä näyttäen käyttäjälle sivun graafisessa muodossa. HTML on pohjimmiltaan suhteellisen yksinkertainen kieli, mutta ilman WWW-sivujen tekijän perehtymistä oikeaoppisiin toteutustapoihin voi seurauksena olla monilla käyttäjillä toimimaton sivusto.

Historia

Toimimattomuuteen on syynä se, että kun Internetin suosio kasvoi räjähdysmäisesti 1990-luvun puolivälissä, keskenään kilpailevat selainvalmistajat Microsoft ja Netscape kehittivät HTML:ään uusia ominaisuuksia omilla toteutustavoillaan, jotka eivät toimineet kilpailijan selaimessa. Yhteensopivuuden turvaamiseksi perustettiin vuonna 1994 World Wide Web Consortium (W3C), kansainvälinen organisaatioiden yhtymä, joka alkoi kehittää Internetissä käytettäviä tekniikoita suunnittelemalla niille ohjeistoja, joita sekä sisällöntuottajien että ohjelmistovalmistajien tulisi noudattaa. Vasta kun vuonna 1998 perustettu Web Standards Project (WaSP) alkoi kutsua näitä ohjeistoja standardeiksi ja suostutteli selain- ja muut ohjelmistovalmistajat tukemaan niitä tarkasti, tuli mahdolliseksi rakentaa sivustoja, jotka toimivat tarkoitetulla tavalla useilla eri selaimilla ja laitteilla (Zeldman 2003: 16–17, 21).

Suositukset

WWW-kehittäjän on kuitenkin edelleen mahdollista olla noudattamatta standardeja ja silti luoda näennäisesti toimiva sivusto käyttämällä tuettuja, mutta standardista poikkeavia toteutustapoja tai jopa kokonaan virheellistä koodia, sillä useimmat selaimet myös korjaavat automaattisesti koodissa olevia virheitä. Tällaisten toteutustapojen käyttö ei ole lainkaan harvinaista, mutta seurauksena voi olla toimivuusongelmat joidenkin selainten ja laitteiden kohdalla. Tämän vuoksi kaikkien tulisi kirjoittaa vain standardeja noudattavaa koodia ja luottaa selainvalmistajien pyrkimykseen huolehtia yhteensopivuudesta. (Schafer 2005: xxiv–xxv.)

Tällä hetkellä suositellaan sivustojen merkintäkielenä käytettäväksi XML-rakenteeseen pohjautuvaa XHTML:ää, koska sen yhteensopivuus nykyisten ja etenkin tulevien selainsovellusten kanssa on parempi kuin sitä edeltävän HTML:n, joka sallii epäyhtenäisen koodin ja hankalien ulkoasuelementtien käyttämisen (Zeldman 2003: 145–147).

XHTML 1.0:n merkintätapa on tietyin edellytyksin taaksepäin yhteensopiva, eli vanhoillakin selaimilla XHTML-sivut voidaan saada toimimaan normaalisti. W3C:n XHTML 1.0 -määritysdokumentin liitteessä C (HTML Compatibility Guidelines) on erityisesti kerrottu keinoista, joita voidaan käyttää yhteensopivuuden parantamiseen (XHTML™ 1.0 The Extensible HyperText Markup Language (Second Edition) 2002).

HTML- ja XHTML-dokumenttien tulee alkaa dokumenttityyppimäärityksellä, josta käy ilmi, mitä merkintäkieltä ja versiota dokumentti noudattaa. Dokumenttityyppimäärityksessä tulee myös antaa polku merkityn kieliversion sallitun rakenteen tarkasti määrittelevään DTD-tiedostoon (Document Type Definition), jota käytetään, jos sivu halutaan validoida eli ohjelmallisesti tarkistaa, että se noudattaa määritellyn merkintäkielen version sääntöjä. XHTML-dokumenttien DTD:t määräävät lisäksi, että **html**-juurielementissä tulee aina määritellä nimiavaruus **xmlns**-attribuutilla seuraavan esimerkin näyttämällä tavalla.

Esimerkki: Dokumenttityypin ja nimiavaruuden määrittely

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
```

Nimiavaruuden tehtävä on erotella eri elementtiryhmät toisistaan, sillä XML mahdollistaa useiden eri nimiavaruuksien elementtien käytön samassa dokumentissa, ja XHTML:n nimiavaruusmääritys ilmaisee **html**-elementin sisällään pitämien elementtien kuuluvan juuri XHTML-nimiavaruuteen. Vaikka nimiavaruus on ilmaistu WWW-osoitteena, ohjelmat tulkitsevat sen vain yksilöivänä merkkijonona, eikä kyseinen WWW-sivu pidä sisällään määrittelyjä kuten dokumenttityyppimäärityksen vaatima DTD-tiedosto.

Merkistö

Eräs asia, joka on myös syytä huomioida WWW-sivuja tehtäessä, on käytetyn merkistön ilmoittaminen, jotta sivulla olevat kirjaimet näkyisivät selaimessa oikein. Yleisessä käytössä olevia merkistöjä on useita, ja osa merkeistä koodataan eri tavoin eri merkistössä. Tämän vuoksi merkistöongelmat eivät ole lainkaan harvinaisia. Aluekohtaiset merkistöt ovat kuitenkin pikku hiljaa korvautumassa yhdellä yleisellä merkistöllä.

Unicode on universaali merkistöstandardi, joka pyrkii korvaamaan kaikki eri alueiden omat merkistöt tarjoamalla yhden laajan merkistön maailman kaikille nykyisin yleisesti käytetyille kielille. Bittitasolla Unicodella on kuitenkin erilaisia tallennustapoja, joista UTF-8 on varsinkin Internet-

käytössä yleinen. UTF-8 on osittain yhteensopiva länsimaisten merkistöjen kanssa, koska se käyttää samaa yhden tavun tallennustapaa perusmerkeille, eikä länsimaisilla kirjaimilla kirjoitettu teksti tämän vuoksi vie tallennettuna juuri enempää tilaa. Aluekohtaiset merkit – kuten esimerkiksi **ä** ja **ö**, tai **黒** – vievät kuitenkin UTF-8:ssa merkistä riippuen kahdesta neljään tavua.

Palvelin voidaan periaatteessa konfiguroida lähettämään tieto käytetystä merkistöstä lähetettävien sivujen otsaketietojen (header) mukana, mutta palvelimen asetusten muuttaminen ei välttämättä ole joka tilanteessa mahdollista. Käytettäessä palvelinpuolen skriptauskieliä voidaan otsaketiedoissa ilmoitettava merkistö määritellä myös skriptistä käsin. Käytetty merkistö voidaan kuitenkin myös lisätä erikseen WWW-sivujen koodiin, mikä on yleensä yksinkertaisin ja selkein tapa varmistaa oikean merkistön käyttö sivuja näytettäessä.

Tämä voidaan periaatteessa tehdä kahdella eri tavalla. XML-tiedostoissa, mitä siis myös XHTML-tiedostot ovat, voidaan tiedoston ensimmäiseksi riviksi antaa XML-prologi.

Esimerkki: XML-prologi

```
<?xml version="1.0" encoding="utf-8"?>
...
```

Tämä ei kuitenkaan ole kaikkien selainten tukema merkintätapa, eikä myöskään standardien vaatima käytäntö silloin, kun XHTML-dokumentti lähetetään palvelimelta **text/html**-sisältötyypillä merkittynä. Kaikki selaimet eivät myöskään tue XHTML:n suositeltua sisältötyyppimäärittelyä **application/xhtml+xml**, minkä vuoksi tätä ei tule toistaiseksi käyttää Internet-sivujen yhteydessä. Yleisesti tuettu tapa ilmoittaa merkistö onkin lisätä merkistömäärittelyn sisältävä meta-elementti XHTML-tiedoston **head**-elementin sisään ja niin alkuun kuin mahdollista.

Esimerkki: merkistön ilmoittaminen meta-elementillä

```
<head>
    <meta http-equiv="Content-Type"
        content="text/html; charset=utf-8" />
    ...
</head>
```

Entiteetit

Entiteetit tarkoittavat HTML- ja XHTML-kielistä puhuttaessa tiettyjä merkkisarjoja, jotka WWW-selain näyttää aina määrättyinä merkkeinä. Merkeillä **<**, **>**, **"** ja **&** on erityismerkitys näissä kielissä, ja nämä merkit tulee merkitä entiteetein silloin, kun niitä käytetään sivun sisällössä, eikä kielen syntaksissa. Myös välilyönnin tulostava entiteetti *** *** on erittäin yleisesti käytetty, koska selaimet eivät näytä useita peräkkäisiä sivun koodiin kirjoitettuja välilyöntejä, mutta entiteetin avulla merkityt välilyönnit tulostuvat sivulle aina. Entiteettejä on näiden lisäksi olemassa monille erikoismerkeille, joiden kirjoittaminen ei onnistu helposti kaikilla näppäimistöillä.

2.1.2 CSS

Suositusten mukaan WWW-sivujen rakenne ja ulkoasumääritykset tulisi pitää erillä toisistaan (Zeldman 2003: 53). Ulkoasumäärityksiä varten onkin kehitetty oma merkintätapansa, Cascading Style Sheets (CSS). Sen avulla kokonaisen sivuston kaikki ulkoasumääreet on mahdollista keskittää yhteen tai muutamaan organisoituun tyylitiedostoon. Tämän ansiosta laajojenkin muutosten tekeminen ulkoasuun on helppoa, sivusto pysyy yhtenäisenä, eikä sisällön toteuttajan tarvitse edes olla sama henkilö kuin ulkoasun tekijän (Schafer 2005: 163).

XHTML-elementeille voidaan haluttaessa määritellä yksilöivänä tunnisteena toimiva **id**-attribuutti sekä rajaton määrä eri luokkia **class**-attribuutilla. Näitä arvoja käyttämällä tiettyihin sivun elementteihin tai elementtiryhmiin voidaan CSS-tiedostosta viitata suoraan. Ulkoasumäärityksiä voidaan antaa myös pelkän elementin nimen perusteella, jolloin kaikki kyseiset elementit noudattavat näin annettuja määritteitä.

CSS-tyylejä voidaan määritellä kolmessa eri paikassa. Yksi vaihtoehto on lisätä tyylejä jokaiselle elementille erikseen **style**-attribuuttia käyttäen. Tämä tekee kuitenkin koodista vaikeaselkoista ja erittäin hankalasti päivitettävää, joten tätä tapaa on normaalisti järkevää välttää. Toinen vaihtoehto on lisätä jokaisen WWW-sivun **head**-elementin sisään **style**-elementti, jossa määritellään kaikki kyseisen dokumentin käyttämät tyylit. CSS-tyyleistä saadaan kuitenkin suurin hyöty irti vasta, kun tyylit tallennetaan omaan erilliseen tiedostoonsa ja linkitetään jokainen WWW-sivu käyttämään tätä ulkoista tiedostoa, koska tällä tavalla riittää, että sivuja päivitettäessä muutoksia tehdään vain yhteen paikkaan.

W3C on kehittänyt CSS:stä eri tasoja, joista CSS 2.1 on tämänhetkinen suositus. Kaikki selaimet eivät kuitenkaan tue koko CSS 2.1 määritystä, joten WWW-kehittäjän on tiedettävä, mitkä ominaisuudet toimivat valtaosassa selaimia. Internetin hakukoneita käyttämällä voi helposti löytää sivustoja, joilla selainten eroavaisuuksia tällä alueella on dokumentoitu.

2.2 Palvelinpuolen ohjelmointi

Palvelinohjelmistot antavat yleensä mahdollisuuden käyttää ohjelmointikieliä WWW-sisällön luomiseen. Tällä tavalla saadaan WWW-sivuille ohjelmoitua sellaisia toiminnallisuuksia, mitä pelkillä staattisilla merkintäkielillä ei pystytä toteuttamaan. Ohjelmointikielten käyttö palvelimilla vaatii kuitenkin sen, että tuki kielelle on asennettu. Palvelimilla voidaan käyttää joko skriptauskieliä, jotka käännetään automaattisesti ajettaessa, tai erikseen käännettäviä kieliä, jotka kuluttavat vähemmän palvelimen resursse-

ja ajettaessa. Osaa olemassa olevista kielistä voidaan käyttää kummalla tahansa tavalla.

2.2.1 Dynaaminen sivusto

Palvelinympäristössä toimivia ohjelmointikieliä käyttämällä on mahdollista tuottaa dynaamisesti toimivia sivustoja: valmiiseen sivupohjaan voidaan ladata sisältöä eri lähteistä tilanteen vaatimalla tavalla. Yksi tekniikan suurista hyödyistä on se, ettei tiettyjä toistuvia sivun osia tarvitse sisällyttää jokaiseen HTML-tiedostoon erikseen, mikä tekisi varsinkin laajojen sivustojen hallinnasta erittäin työlästä, vaan sivut voidaan jakaa osiin, joita palvelimella sijaitseva ohjelmakoodi yhdistelee kooten niistä käyttäjälle lähetettävän sivun.

Palvelinpuolen ohjelmointi mahdollistaa myös muita hyödyllisiä toiminnallisuuksia, kuten esimerkiksi käyttäjien tunnistamisen sisäänkirjautumisen avulla. Kun tiedetään, mikä käyttäjä sivustolla liikkuu, voidaan tälle tarjota personoitua sisältöä, kuten vaikka tämän omia asiakastietoja. Myös useakielisessä käyttöliittymässä voidaan kielivalinta pitää muistissa ja lähettää oikeankieliset tekstit samaan sivupohjaan liitettynä. Käyttäjälle voidaan myös antaa mahdollisuus tehdä merkintöjä järjestelmään, esimerkiksi varaus yrityksen palveluun.

2.2.2 PHP

PHP on yksi monista palvelimilla käytettävistä skriptauskielistä, ja se sai alkunsa vuonna 1995 yksinkertaisena WWW-ohjelmoinnin helpottajana. Lyhenne tuli alun perin sanoista Personal Home Page, mutta uudistusten myötä viralliseksi nimeksi vaihdettiin PHP: Hypertext Preprocessor. (Hudson 2005: 1.)

PHP on alusta asti perustunut vapaaseen lähdekoodiin, ja kieltä kehitetään avoimesti. Sen käyttö on ilmaista, ja se tukee myös laajennusosia, joita kuka tahansa voi tehdä. Osittain näistä syistä PHP:n suosio on kasvanut suureksi ja Hudson (2005: xi) luonnehtiikin sitä WWW-ohjelmointikielten kiistattomaksi kuninkaaksi. Sen yhteensopivuus eri palvelinalustojen kanssa on erittäin hyvä ja PHP onkin valmiiksi asennettuna useilla kaupallisesti WWW-sivuja ylläpitävillä palvelimilla, joten sen käyttöönotto on helppoa. Internetistä voi myös löytää valtavan määrän ohjeita kielen opetteluun.

Muista tällä hetkellä runsaasti käytetyistä palvelinpuolen ohjelmointitekniikoista olennaisimpina voidaan mainita Microsoftin kehittämät Active Server Pages (ASP) ja uudempi .NET-teknologiaa käyttävä ASP.NET, alunperin Sun Microsystemsin kehittämä JavaServer Pages (JSP) sekä Allairen, Macromedian ja Adoben vuoron perään kehittämä ColdFusion. Niillä ei toiminnallisuutensa osalta ole merkittävää eroa toisiinsa tai

PHP:en, tosin uusimmat teknologiat saattavat tarjota tehokkaampia kehitystyökaluja tai paremman suorituskyvyn. Aiemmin mainituista syistä johtuen ilmainen, kevyt ja tuettu PHP on kuitenkin luonnollisin vaihtoehto toimeksiannon toteutuksessa käytetyksi palvelinpuolen ohjemointikieleksi.

2.2.3 Olio-ohjelmointi

Lyhyesti ilmaistuna olio-ohjelmoinnilla tarkoitetaan ohjelman eri toimintojen ja tietojen loogista jakamista ja paketoimista osioihin, joita kutsutaan luokiksi. Olioksi kutsutaan ohjelmointikielen muuttujaa, joka voidaan luoda tarpeen mukaan valmiin luokan pohjalta. Tällöin luokan sisältämät toiminnot ovat olion kautta ohjelman käytettävissä.

Babin, Good, Kromann ja Stephens (2005: 21) toteavat, että PHP:lla voidaan tehdä toimivia ja käyttökelpoisia ohjelmia ilman, että käytetään lainkaan oliopohjaista lähestymistapaa, mutta muistuttavat olio-ohjelmoinnista olevan tiettyjä etuja, koska se on astetta kehittyneempi tapa tuottaa ohjelmia. Tekniikan selviksi eduiksi voidaan mainita ensinnäkin se, että monimutkaisten järjestelmien rakentaminen on helpommin hallittavissa, mikäli kokonaisuus jaetaan luokkiin. Toiseksi myös ohjelman osat ovat helposti uudelleenkäytettävissä muissa yhteyksissä, sillä luokat toimintoineen ovat periaatteessa itsenäisiä kokonaisuuksia, vaikka saattavatkin tarvita myös muita luokkia toimiakseen.

Suunnittelutyön osuus korostuu huomattavasti olio-ohjelmointia käytettäessä, koska tällöin mahdollisten toteutustapojen määrä on huomattavasti suurempi kuin muussa tapauksessa. Onnistunut olio-ohjelmointi yleensä vaatiikin runsaasti aikaisempaa tietoa ja kokemusta aiheesta. Varsinkin ohjelman laajentaminen tulevaisuudessa voi olla hyvin vaikeaa huonosti rakennetussa järjestelmässä.

PHP:n olio-ohjelmointituki uusittiin kokonaan vuonna 2004 julkaistuun PHP 5:een, sillä edellisen version tuki oli puutteellinen (Hudson 2005: 128). Tätä kirjoitettaessa PHP 5 on edelleen kielen uusin versio, joten se on valittu kuvatun toimeksiannon toteutuksessa käytetyksi PHP-versioksi. Kehitteillä oleva PHP 6 on otettu ohjelmoinnissa huomioon niin, ettei sovelluksessa ole käytetty sellaisia ominaisuuksia, joiden tiedetään olevan tulevan PHP-version kanssa epäyhteensopivia.

2.3 Selainpuolen ohjelmointi

Palvelimella sijaitsevien skriptien lisäksi on mahdollista myös ohjelmoida skriptejä, jotka suoritetaan käyttäjän selaimessa. Samankaltaisia toimintoja on mahdollista toteuttaa kummalla tahansa skriptaustavalla, mutta palvelimella sijaitsevat skriptit ovat luotettavampia, koska niiden suoritus ei

riipu käyttäjän selaimen ominaisuuksista ja asetuksista, eikä käyttäjällä ole mahdollisuutta ohittaa palvelimella suoritettavia skriptejä, vaikka selainskriptien kohdalla se onkin yleensä mahdollista.

Palvelinpuolen skriptauksella ei pystytä kuitenkaan suoraan vaikuttamaan WWW-sivun sisältöön sivujen lataamisten välillä, vaan niiden toiminta rajoittuu käyttäjän selaimen lähettämiin sivupyyntöihin. Selainta ohjaavilla skripteillä voidaan kuitenkin muuttaa käyttäjän näkemää sivua ilman, että palvelimeen ollaan yhteydessä, ja niiden avulla voidaan saada aikaan erilaisia interaktiivisia efektejä.

2.3.1 JavaScript

JavaScriptiä käytetään yleisesti WWW-sivuilla selainta ohjaavana skriptauskielenä. Vaikka JavaScript ei ole ainoa WWW-sivuja varten kehitetty skriptauskieli, on se käytännössä ainoa yleisessä käytössä oleva skriptauskieli selainten ohjelmointiin, koska muilla kielillä ei ole laajaa tukea eri selainvalmistajien keskuudessa (Duffy 2003: 14).

Historia

JavaScript-tuki ilmestyi aluksi Netscapen selaimeen joulukuussa 1995, minkä jälkeen kielen kehitys tapahtui HTML:n tavoin: kilpailevat selainvalmistajat kehittivät omia versioitaan JavaScriptistä ja sen käyttämästä Document Object Modelista (DOM), joka tarkoittaa viittaustapaa WWW-dokumentin eri osiin. Vuonna 1997 JavaScriptistä ja DOM:sta ilmestyivät European Computer Manufacturers Associationin (ECMA) ja W3C:n kehittämät standardit, joita selaimet alkoivat pääosin tukea standardista poikkeavien vanhojen toteutustapojen rinnalla. Eri selainten DOM-rakenteissa on kuitenkin edelleen pieniä eroja. (Duffy 2003: xvii, 4–8, 14.)

JavaScriptiä käytettäessä on tärkeää tiedostaa, että kaikki WWW-sivuja näyttävät selaimet ja laitteet eivät sitä välttämättä tue. Käyttäjillä on yleensä myös mahdollisuus tietoisesti estää JavaScriptin toiminta selaimessaan osittain tai kokonaan. Tämän vuoksi on syytä olla rakentamatta mitään olennaista toiminnallisuutta WWW-sivustolla yksin JavaScriptin varaan. Sitä voidaan kuitenkin käyttää esimerkiksi käyttäjäystävällisyyden parantamiseen tai muun lisäarvon tuomiseen niille käyttäjille, joiden selaimessa se toimii. (Goodman 2003: xiii.)

2.3.2 Ajax

JavaScriptiä käyttämällä voidaan WWW-sivu ohjelmoida myös vaihtamaan tietoja isäntäpalvelimen kanssa ilman, että joudutaan lataamaan uutta sivua. JavaScriptillä palvelimelta haettu tieto voidaan saada sivulle heti näkyviin JavaScriptin muita komentoja käyttämällä. Tätä toiminnallisuutta on alettu kutsua nimellä Ajax, joka on johdettu sanoista Asynchronous JavaScript and XML.

Ajax ei siis ole itsessään mikään uusi tekniikka, vaan tapa käyttää tiettyjä tekniikoita uudella tavalla. Ajaxin käyttämät tekniikat ovat olleet olemassa jo pitkään, mutta vasta vuoden 2005 alkupuolella käyttötapa alkoi nopeasti yleistyä ja termi Ajax syntyi. Nimenomaan jo yleisesti olemassa oleva tuki sen käyttämille tekniikoille mahdollisti Ajaxin nopean leviämisen, ja vain hyvin pieni osa käyttäjistä käyttää enää selaimia, joista tuki puuttuu. (Asleson, Schutta 2007: 14–16, 25.)

Ajaxin käyttö on merkittävä edistysaskel Internetin kehityksessä, sillä se mahdollistaa tavallaan työpöytäohjelmistojen kaltaisen toiminnallisuuden WWW-sivuilla. Käyttäjän ei Ajaxia hyödynnettäessä tarvitse odotella uuden sivun latautumista jokaisen toiminnon jälkeen, vaan hän voi sen sijaan jatkaa WWW-sovelluksen käyttöä samalla, kun selain vaihtaa taustalla tietoja palvelimen kanssa. (Asleson, Schutta 2007: 15–16.)

Selaimen XMLHttpRequest-pyyntöihin, joita Ajaxia käytettäessä normaalisti lähetetään, voidaan palvelimelta lähettää vastaus erilaisissa muodoissa. Vastaus voidaan antaa yksinkertaisesti tekstimuodossa, tulkita JavaScriptissä ja suorittaa sen perusteella jokin ennalta ohjelmoitu toiminto. Vaihtoehtoisesti vastauksena voidaan lähettää jokin sivun osa valmiiksi HTML-muodossa, joka JavaScriptillä vain päivitetään näkyviin käyttäen HTML-elementtien innerHTML-ominaisuutta. Kyseistä ominaisuutta ei ole standardoitu, mutta se on kuitenkin yleisesti tuettu nykyaikaisissa selaimissa. (Asleson, Schutta 2007: 41–42.)

On myös mahdollista lähettää palvelimelta JavaScriptille tietorakenteita XML- tai JavaScript Object Notation -muodossa (JSON), jolloin tietoja voidaan käsitellä JavaScriptissä paljon monipuolisemmin. Jos tarvetta todella on monimutkaisten tietorakenteiden siirtämiseen, niin näitä tietorakenteita on järkevää käyttää. Monissa tilanteissa tämä on kuitenkin epäkäytännöllinen tapa välittää tietoja selaimelle, sillä JSON- ja XML-muotoisen tiedon käsittely JavaScriptissä on monimutkaisempaa ja siirrettävän tiedon määrä voi olla moninkertainen näitä käytettäessä rakennemääritysten viedessä ylimääräistä tilaa. JSON on näistä vaihtoehdoista kuitenkin selkeästi tiiviimpää. Näitä tietorakenteita voidaan käyttää myös lähetettäessä selaimelta tietoa palvelimelle. (Asleson, Schutta 2007: 44–45, 64, 69–71, 75–76.)

2.4 Tietokanta

Tietokantaohjelmistoja käytetään yleisesti sähköisessä muodossa olevan tiedon organisoituun tallentamiseen. Tarvittaessa tietokantoihin voidaan tallentaa monimutkaisiakin tietorakenteita, ja tiedon hakeminen on muihin tallennustapoihin nähden suhteellisen helppoa.

2.4.1 Syitä tietokannan käyttöön

Tietokantoihin voidaan tehdä kyselyitä käyttäen tarkoitukseen kehitettyä kyselykieltä. Tämä mahdollistaa tietokannassa säilytettävän tiedon hakemisen, lisäämisen, poistamisen ja muokkaamisen ohjelmakoodista käsin. Structured Query Language (SQL) on selvästi yleisin tietokantojen muokkaukseen käytetty kieli, mutta vaikka sen ensimmäinen virallinen standardi julkaistiin jo vuonna 1986, eri tietokantaohjelmistot käyttävät edelleen hieman toisistaan poikkeavia merkintätapoja ja ominaisuuksia (Lahtonen 2002: 38).

Tietokantaa ja SQL:ää käyttäen saadaan tarvittavat tiedot haettua hyvin yksinkertaisesti ja tehokkaasti verrattuna siihen, että tiedot olisi tallennettu jollakin muulla tavalla, esimerkiksi XML-tiedostoihin. Tietokantaan voidaan tehdä monimutkaisiakin kyselyitä yhdistellen useita eri tauluja ja käyttäen vertailuja tai tietokannan tukemia funktioita, mikä lisää tehokkuutta entisestään. Tietokantaohjelmistoihin on myöskin sisäänrakennettu ominaisuuksia samanaikaista käyttöä varten, eli ainakaan vakavia ongelmia ei synny, vaikka useat käyttäjät yrittäisivät tehdä muutoksia tietoihin samanaikaisesti.

2.4.2 MySQL

Tietokantaohjelmistoja on olemassa monia, mutta ilmainen avoimeen lähdekoodiin perustuva MySQL on tullut erittäin yleiseksi kevyessä Internet-käytössä. Kaikilta ominaisuuksiltaan se ei välttämättä pärjää vertailussa parhaiden kaupallisten tietokantajärjestelmien kanssa, mutta sitä pidetään kuitenkin suorituskykyisenä ja sitä kehitetään jatkuvasti (Hovi 2004: 4). Ilmaisuutensa ja helpon saatavuutensa vuoksi MySQL on yleensä järkevin ratkaisu ainakin pienimuotoisten WWW-sovellusten käyttämäksi tietokannaksi.

Kuten PHP, myös MySQL löytyy asennettuna monilta WWW-sivutilaa tarjoavilta palvelimilta, ja erityisesti niiden yhteiskäyttö on yleistä. PHP:stä löytyy valmiita MySQL-yhteyttä varten tehtyjä funktioita, joten käytön aloittaminen on vaivatonta.

2.5 Tietoturva

Suurin osa Internet-palveluiden käyttäjistä toimii kuten järjestelmien suunnittelijat ovat tarkoittaneet, eivätkä sen seurauksena yleensä aiheuta ongelmia, mutta on olemassa myös haitallisia käyttäjiä, jotka yrittävät kohdistaa erilaisia hyökkäyksiä järjestelmiin. Osa haittakäyttäjistä tekee tätä vain huvin vuoksi, toiset puolestaan saavuttaakseen itselleen rahallista hyötyä (Hudson 2005: 190). Tämän vuoksi palvelinpuolen ohjelmointia

ja tietokantaa käyttävien sivustojen suunnittelijoiden täytyy ottaa huomioon monia tietoturvaseikkoja. Näitä asioita käsitellään tässä luvussa.

2.5.1 Järkevän tietoturvatason määrittäminen

Tietoturvan saatetaan helposti ajatella olevan asia, joka on tarpeellista aina maksimoida. Tietoturvamurroista uutisoidaan, ja ehkä yleinen mielipide on se, ettei turvaa tunnu olevan tarpeeksi. Tietoturvan on kuitenkin tarkoitus olla käyttäjälle mahdollisimman näkymätön asia. Tietoturvallisuutta lisättäessä seuraa siitä todennäköisimmin jonkinlaista epämukavuutta käyttäjälle. Tämä voi näkyä esimerkiksi pidempinä latausaikoina, useampina varmennuksien antamisina, nopeampina aikakatkaisuina ja/tai suurempina palvelumaksuina. Tietoturvan kohdalla oikean tasapainon löytäminen jokaiseen käyttökohteeseen on olennaista.

Sivustolla tarjottavan palvelun tyyppi osaltaan vaikuttaa tarvittavan tietoturvan määrään: esimerkiksi pankkien verkkopalveluiden tietoturvan täytyykin olla ehdottoman luotettava, koska niiden kautta käsitellään asiakkaiden omaisuutta, ja kohteena ne kiinnostavat rikollisia. Toisaalta taas palvelussa, jossa rekisteröidyistä käyttäjistä ei edes kerätä henkilökohtaisia tietoja, voi liiallinen tietoturva olla turha investointikohde tai jopa palvelun käyttöä häiritsevä tekijä.

Toinen asia, jonka voidaan ajatella vaikuttavan vaadittavan tietoturvan määrään, on palvelun käyttäjämäärä. Mitä enemmän käyttäjiä sivustolla liikkuu, sitä suurempi on palveluun kohdistuvien hyökkäysten todennäköisyys kuin myös haitta palvelun keskeytymisestä tai tietojen katoamisesta. Hyvin pienen palvelun ei välttämättä ole mielekästä rakentaa erityisen tehokasta suojausta, mikäli palvelussa liikkuva tieto ei sitä erityisesti vaadi. Tällöin käyttäjämäärän mahdollisesti kasvaessa voidaan tietoturvaa kuitenkin joutua tehostamaan.

2.5.2 Järjestelmän perussuojaus

Jokaisessa järjestelmässä on tärkeää ottaa huomioon mahdolliset tietoturva-aukot ja -riskit jotka voivat syntyä ohjelmointivaiheessa, jotta tyypillisimmät hyökkäykset palvelua vastaan voidaan estää. Esimerkiksi kaikki tietokantaan tallennettava käyttäjiltä saatu syöte täytyy prosessoida niin, etteivät mahdollisesti vahingollista koodia sisältävät syötteet pääse aiheuttamaan vahinkoa (Howard, LeBlanck & Viega 2005: 60). Monissa tapauksissa oikeanmuotoinen syöte voidaan tarkistaa säännöllisillä lausekkeilla (regular expression), mutta kun käyttäjien syötteitä käytetään tietokantakyselyissä, tulee vielä tietyt merkit muuntaa sellaiseen muotoon, että tietokantaohjelmisto ymmärtää merkkien olevan syötettävää tietoa, eikä osa suoritettavaa komentoa. PHP:ssa MySQL-tietokantaa käytettäessä tämä voidaan tehdä funktiolla **mysql_real_escape_string**.

Sama pätee silloin, kun käyttäjän antama syöte tulostetaan WWW-sivulle, tietokannan välityksellä tai ilman. Tällöin on syytä aina käyttää PHP:n **htmlspecialchars**-funktiota, joka muuttaa HTML-syntaksin käyttämät erikoismerkit entiteeteiksi, jotta kyseiset merkit tulostuisivat sivulle sen sijaan, että ne vaikuttaisivat sivun rakenteeseen.

Palvelimen skriptitiedostojen tiedostotunnisteen pitää olla sellainen, että palvelin prosessoi kaiken ohjelmakoodin ennen lähettämistä, jottei kukaan ulkopuolinen pääse näkemään, kuinka ohjelma toimii. Toisin sanoen kaikilla PHP-tiedostoilla – myös muihin tiedostoihin suorituksen aikana liitettävillä koodia sisältävillä osilla – tulee siis olla tiedostopääte **.php**, eikä esimerkiksi **include**-komentoon viittaava **.inc**. Toinen vaihtoehto on muuttaa palvelimen asetuksia niin, että se prosessoi myös muunpäätteiset tiedostot PHP:na. (Hudson 2006: 191.)

Tietokantaa käytettäessä tietokantayhteyden käyttöön tarvittavat tunnukset täytyy olla PHP-skriptin käytettävissä. Tällaiset salassa pidettävät tiedot on järkevintä kuitenkin tallentaa kokonaan palvelimen julkisen WWW-kansion ulkopuolelle, mistä niitä ei voi ladata ilman kirjautumista palvelimelle, vaikka samalla palvelimella sijaitseva ohjelma pystyykin tiedot lataamaan. Virheviestien poistamista käytöstä sekä palvelimella olevien skriptitiedostojen tiedostotunnisteiden piilottamista voi myös harkita, jolloin hyökkääjä ei välttämättä saa selville palvelimella käytettävää ohjelmointikieltä. (Hudson 2006: 191–192.)

Järjestelmää koskevien tarkkojen tietojen antamista tulee myös välttää, koska mahdollinen hyökkääjä voi saada niistä apua hyökkäyksiinsä. Käyttäjille näytettävien virheviestien ei siis tule olla liian yksityiskohtaisia. Myös palvelinohjelmiston tarkka versionumero kannattaa piilottaa, jotta sen mahdollisten tunnettujen ohjelmointivirheiden hyödyntäminen hyökkäyksissä ei onnistuisi. (Howard ym. 2005: 186.)

2.5.3 Salasanojen ja tietoliikenteen suojaus

Salasanan suojaus

Salasanaa ei ole käyttäjien kannalta turvallista tallentaa selkokielisenä tietokantaan, koska ainakin kehittäjillä ja ylläpitäjillä on mahdollisuus nähdä suoraan tietokannan sisältämät tiedot hallintaohjelman avulla. Jos joku ulkopuolinen onnistuisi murtautumaan tietokantaan, saisi tämäkin haltuunsa kaikkien käyttäjien salasanat. Vaikka kyseisessä järjestelmässä tietokantaan murtautuja ei saisikaan salasanoilla enää ylimääräistä tuhoa aikaan, ihmiset usein käyttävät samoja salasanoja eri palveluissa ja tämän johdosta murtautuja saattaisi päästä helposti kirjautumaan käyttäjien sähköpostitileihin sekä muihin palveluihin. (Howard ym. 2005: 144–145.)

Salaus ja hash

Käyttäjien salasanat tulee siis suojata muuntamalla ne eri merkkijonoksi. Yksi tapa on käsitellä salasana jollain monista yleisesti käytetyistä sa-

lausalgoritmeista. Näin se voidaan vielä kääntää takaisin alkuperäiseen muotoonsa, kun tiedetään salausavain. Howard ym. (2005: 104) muistuttavat, ettei itse kehitettyjä salausalgoritmeja tule koskaan käyttää. Salasanaa ei kuitenkaan tarvitse koskaan pystyä muuntamaan takaisin alkuperäiseen muotoon, joten turvallisinta on kääntää salasana sellaiseksi merkkijonoksi, ettei sen perusteella pysty alkuperäistä salasanaa selvittämään. Tähän voidaan käyttää hash-funktioita, kuten **md5** (Message-Digest Algorithm) tai **sha1** (Secure Hash Algorithm 1), jotka muodostavat aina tietynmittaisen tunnistemerkkijonon annetun merkkijonon perusteella.

Tietty merkkijono ajettuna saman hash-funktion läpi palauttaa aina saman merkkijonon, joka koostuu heksadesimaalinumeroista, mutta pienikin muutos syötetyssä merkkijonossa palauttaa täysin erilaisen hash-arvon. Saman hash-arvon syntyminen kahdesta eri syötteestä on hyvin epätodennäköistä, mutta ei mahdotonta. Sisäänkirjautuminen varmennetaan niin, että käyttäjän kirjautuessaan antama salasana ajetaan saman hash-funktion läpi kuin tietokantaan tallennettu salasana ajettiin ennen tallentamistaan, ja verrataan hash-arvoja keskenään. Koska hash-arvosta ei enää voida muodostaa alkuperäistä salasanaa, on käyttäjän unohtaessa salasanansa ainoa mahdollisuus generoida tälle kokonaan uusi salasana.

Vaikka alkuperäistä merkkijonoa ei pystytäkään enää selvittämään hash-arvon perusteella, on tietomurtautujan mahdollista niin sanotulla sanakirjahyökkäyksellä yrittää saada tietoonsa alkuperäinen merkkijono. Hyökkäyksessä muodostetaan suuri määrä hash-arvoja ennalta määrätyistä sanoista ja verrataan niitä murrettavaan hash-arvoon. Koska monet ihmiset käyttävät salasanoinaan selkokielisiä sanoja, on murtautujan periaatteessa mahdollista näin saada selville tällaisia salasanoja hash-arvon perusteella.

Tämän ehkäisemiseksi voidaan käyttäjien salasanoihin vielä lisätä automaattisesti generoitu satunnaismerkkijono ennen hash-funktion käyttöä. Tätä menetelmää kutsutaan salasanan suolaamiseksi. Jotta salasana voitaisiin tämän jälkeen varmentaa joka kirjautumisen yhteydessä, pitää lisätyn satunnaismerkkijonon eli suolan olla kuitenkin tallennettuna. Sen tulee kuitenkin olla vaikeasti löydettävissä. Se voi esimerkiksi olla sulautettuna hash-arvon alkuun tai loppuun tai siroteltuna sen sekaan erillistä funktiota käyttäen, tai jopa olla tallennettuna kokonaan tietokannan ulkopuolelle.

Liikenteen suojaus

Jos sivustolla käsitellään luottamuksellista tietoa, on perusteltua tarjota käyttäjälle suojattu yhteys palvelimelle. Yhteyden suojaus WWW-sivuilla tarkoittaa sitä, että kaikki palvelimen ja käyttäjän välillä tapahtuva tiedonvaihto salataan tietyllä salausalgoritmilla käyttäen kertakäyttöistä salausavainta, joka on vain näiden kahden osapuolen tiedossa.

Ilman tietoliikenteen salausta jopa käyttäjän kirjautuessa antama salasana lähetetään palvelimelle selkokielisenä tekstinä. Vaikka salasana tämän jäl-

keen palvelimella salattaisiin, se voidaan pystyä kaappaamaan jo aikaisemmassa vaiheessa. Pelkän salasanan salaaminen lähetystä varten käyttäen JavaScriptillä tehtyjä salausalgoritmeja on monimutkaista ja suhteellisen turvatonta, koska salausfunktiot voidaan saada selville sivun koodia tutkimalla ja salauksen purkaminen on sen johdosta helpompaa. Suojattua yhteyttä voi siis olla syytä käyttää jo pelkästään sisäänkirjautumistietojen suojaamiseen.

Yleisin tapa suojata liikenne tietoverkoissa on käyttää Secure Sockets Layer -protokollaa (SSL) tai sen seuraajaa Transport Layer Security -protokollaa (TLS) (Howard ym. 2005: 126). Näillä on vain pieniä eroja ja perusperiaate on tekniikoissa sama, mutta ne eivät kuitenkaan ole toistensa kanssa yhteensopivia, eli salattavan tietoliikenteen molempien osapuolten tulee käyttää samaa protokollaa (What is TLS/SSL? 2003).

Internet-sivuilla SSL- tai TLS-suojattu yhteys käyttää normaalin HTTP-protokollan (Hypertext Transfer Protocol) tunnuksen sijasta WWW-sivun osoitteen alussa protokollatunnusta HTTPS (Hypertext Transfer Protocol over Secure Sockets Layer). Tämä ei kuitenkaan ole erillinen protokolla, vaan tapa ilmaista selaimelle, että käytetään SSL- tai TLS-suojattua normaalia HTTP-protokollaa. Tällöin yhteys siirtyy automaattisesti käyttämään porttia 443, kun taas suojaamaton yhteys käyttää normaalisti porttia 80. Sekä suojatussa että suojaamattomassa yhteydessä portti voidaan myös määritellä joksikin muuksi liittämällä se sivun osoitteeseen.

Liikenteen salaamisen lisäksi näitä tekniikoita käytetään osapuolten autentikointiin eli sen varmistamiseen, että taho, jonka kanssa tietoja vaihdetaan, on oikeasti se, joka väittää olevansa. Useimmiten käyttäjien ja palvelinten varmistus tapahtuu eri mekanismein. Käyttäjien kohdalla salasanojen käyttö on yleisin autentikointitapa, mutta palvelinten autentikoinnissa käytetään sertifikaatteja, joita tietyt yritykset myöntävät sivustoille. Tällä tavalla pystytään poistamaan mahdollisuus, että joku kolmas osapuoli tekeytyisi palvelimeksi ja harhauttaisi käyttäjää luovuttamaan tälle salaisia tietoja. (Howard ym. 2005: 126–127.)

Yksi kansainvälisesti tunnettu sertifikaatteja myöntävä yritys on VeriSign, joka on varmentanut esimerkiksi Nordean, Sampo Pankin ja Osuuspankin verkkopankkien suojatun yhteyden käyttämät sertifikaatit. Monet yritykset perivät suuriakin maksuja sertifikaattien myöntämisestä, mutta sertifikaatteja on mahdollista saada myös ilmaiseksi. Olennaista on, että sertifikaatin myöntänyt taho kuuluu niin sanottuihin luotettuihin sertifiointiauktoriteetteihin. Muussa tapauksessa selain luultavimmin varoittaa käyttäjää epäluotettavasta sertifikaatista sivua ladattaessa.

Sertifikaatin voi nimittäin myöntää vaikka itselleen, mikäli suojatun yhteyden käytön syynä on ainoastaan tietoliikenteen salaaminen. Tällä tavoin saadaan kyllä suojattua tietoliikenne palvelimen ja käyttäjän välillä, mutta palvelimen identiteettiä ei voida varmistaa. Joidenkin selainten

kohdalla tällaisten epäluotettavien sertifikaattien käyttö voi aiheuttaa ongelmia, koska varoitusikkuna saattaa avautua aina sivustolle tultaessa, eikä sertifikaatin hyväksyminen asentamalla se selaimeen ole välttämättä kaikille käyttäjille helppo tehtävä.

3 Tekninen toteutus käytännössä

Tässä luvussa kerrotaan toimeksiantona tehtyä järjestelmää koskevista valinnoista ja toimintatavoista WWW-kehittäjän teknisestä näkökulmasta. Työtä tehtäessä on koodieditorina käytetty Adobe Dreamweaveria.

3.1 Menettelytavat yleisen toimivuuden varmistamiseksi

Standardimääritykset

Ajanvarausjärjestelmän käyttämäksi dokumenttityypiksi valitsin XHTML 1.0 Strictin, koska se on tällä hetkellä määrittelyltään tiukin ja tulevien tekniikoiden kanssa yhteensopivin selainten yleisesti tukema dokumenttityyppi. Strict-määritys tarkoittaa sitä, että puhtaasti esitystapaa määrittäviä elementtejä ja attribuutteja ei dokumentissa sallita lainkaan. Strictin lisäksi on olemassa määritykset Transitional ja Frameset, jotka on tarkoitettu vanhoille WWW-sivuille, joilla vielä on käytössä mainittuja poistumassa olevia ominaisuuksia. Näitä dokumenttimäärityksiä käyttäen myös vanhat HTML-dokumentit voidaan kääntää XHTML-muotoon ilman, että sivuja on pakko muuttaa radikaalisti.

Uudessa järjestelmässä ei ole mitään syytä ottaa käyttöön näitä poistuvia ominaisuuksia, mutta myöskään uusinta XHTML 1.1:tä ei voi vielä käyttää puutteellisen selaintuen vuoksi. Järjestelmän tuottama XHTML-koodi on kuitenkin pyritty pitämään myös XHTML 1.1:n kanssa yhteensopivana, jotta mahdollinen siirtyminen siihen tulevaisuudessa olisi helppoa.

Järjestelmän tuottamat sivut on validoitu W3C:n validaattorilla osoitteessa http://validator.w3.org virheettömyyden varmistamiseksi.

CSS:n käyttö ei varsinaisesti kuulu tämän opinnäytetyön piiriin muuten kuin yleisesti standardien näkökulmasta, koska ulkoasun tekeminen on rajattu aiheen ulkopuolelle. Järjestelmään on silti tehty joitakin CSS-määrityksiä ulkoasun siistimiseksi, mutta nämä määritykset korvataan siinä vaiheessa, kun järjestelmä liitetään osaksi varsinaista WWW-sivustoa. Järjestelmän käyttämät CSS-tyylit on tarkoitettu tehtäväksi ulkoiseen tiedostoon, ja XHTML-koodin elementeille on järjestelmässä annettu luokka- ja id-arvoja, jotta eri elementteihin olisi helppo viitata tulevassa tyylitiedostossa.

Merkistö

Staattisen sivuston käyttämään merkistöön ei WWW-kehittäjän välttämättä tule kiinnitettyä erityistä huomiota, sillä ongelmia ilmenee vain, jos sivuston koodiin tai palvelimelta lähetettyihin otsaketietoihin merkitty merkistö eroaa tiedoston fyysisestä tallennusmuodosta ja eroavia merkkejä käytetään sivulla. Tämän ongelman pystyy periaatteessa kiertämään käyttämällä sivuston koodissa ongelmallisten merkkien kohdalla HTML-entiteettejä.

Oikea tapa välttää tämä ongelma on kuitenkin tallentaa tiedosto sitä merkistöä käyttäen, joka tiedostoon on merkitty käytettäväksi. Näin tiedoston sisältämät merkit oikeasti myös vastaavat tarkoitettuja merkkejä. Kaikilla editoreilla ei pysty valitsemaan tallennettavan tiedoston merkistökoodausta, joten on tärkeää käyttää sivuja tehdessä ohjelmaa, jolla tämä onnistuu.

Kun rakennetaan dynaamisia verkkosovelluksia, käytettävän merkistön huomiointi on tärkeää joka tilanteessa, koska tekstimuotoista tietoa siirretään selaimen, palvelinskriptien ja tietokannan välillä. Samaa merkistöä ei ole pakko käyttää järjestelmän jokaisessa osassa, mutta merkistömuunnosten täytyy silloin olla kunnossa, jotta merkit säilyvät samoina. Jos valinnanmahdollisuus on olemassa, niin saman merkistön käyttö koko järjestelmässä on kuitenkin varmasti järkevin vaihtoehto. Tässä järjestelmässä on käytetty merkistönä pelkkää UTF-8:aa.

3.2 Ohjelmointiratkaisut

Koodin uudelleenkäytettävyyden sekä hallittavuuden helpottamiseksi järjestelmä on tehty käyttäen etupäässä oliopohjaista ohjelmointia. Järjestelmässä on eroteltu omiin luokkiinsa navigointi, kalenteri, käyttäjähallinta sekä tietokantayhteys. Osien sisäiset toiminnot tapahtuvat luokkien ja niistä tehtyjen olioiden metodeita käyttämällä.

Järjestelmän kaksikielisyys on toteutettu niin, että kaikki sivuilla näkyvä teksti tulostetaan olioiden kautta, ja aina käyttäjän kielivalinnan mukaiset tekstit tulevat näkyviin. Tekstit on tallennettu tässä järjestelmässä taulukoina tulostavien olioiden ominaisuuksiin tai metodeihin riippuen siitä, käyttääkö niitä yksi vai useampi metodi, ja kielivalinta välitetään olioille parametrein.

Sessiot ja evästeet

PHP mahdollistaa käyttäjäkohtaisten tilapäistietojen yksinkertaisen tallentamisen palvelimelle käyttäen globaalia **_SESSION**-muuttujaa. Tietoja voidaan PHP:lla tallentaa myös käyttäjän selaimen evästeisiin, mutta tietoturvan näkökulmasta on parasta säilyttää sellaiset käyttäjäkohtaiset tiedot vain palvelimella, jotka halutaan säilyttää suojassa mahdollisilta väärentämisyrityksiltä, koska osaavalla käyttäjällä on mahdollisuus manuaalisesti muuttaa evästeiden tietoja haluamakseen. Evästeet voivat kuitenkin olla hyödyllisiä sivulle myöhemmin palaavan käyttäjän asetusten lataamiseksi, ja rakennettu järjestelmä käyttääkin evästeitä kielivalinnan tallentamiseen (Liite 1: Ohjelmointiesimerkkejä, Esimerkki 1: Sessio- ja evästetietojen käyttö). Jos valinta tallennettaisiin vain sessioon, joutuisi englanninkielinen käyttäjä muuttamaan kielen aina sivulle palatessaan, koska sessiotiedot säilyvät palvelimella vain tietyn ajan. Jos asetus on puolestaan käyttäjäkohtaisissa tiedoissa tietokannassa, pitää käyttäjän kirjautua sisään en-

nen kuin asetukset voidaan ottaa käyttöön. Evästeet säilyvät yleensä selaimen käytettävissä, ellei käyttäjä niitä erikseen poista, ja asetukset voidaan ladata automaattisesti evästeen tietojen perusteella heti käyttäjän palatessa sivustolle samalla tietokoneella ja selaimella kuin aikaisemmin.

Sessioita hyödyntämällä voidaan järjestelmään puolestaan rakentaa sisäänkirjautuminen, jonka ansiosta käyttäjä pysyy tunnistettuna aina uloskirjautumiseen tai aikakatkaisuun saakka. Koska sessiotiedot sijaitsevat vain palvelimella, ei käyttäjä pysty väärentämään käyttöoikeuksiaan. Hierontayrityksen järjestelmään tehtiin kaksi käyttäjätasoa kirjautuneille käyttäjille: asiakas ja työntekijä. Oikea käyttäjätaso määräytyy, kun annetut käyttäjätunnukset varmennetaan käyttäjän kirjautuessa sisään (Liite 1: Ohjelmointiesimerkkejä, Esimerkki 2: Sisäänkirjautumisen käsittely). Näiden lisäksi myös kirjautumaton käyttäjä voi käyttää ajanvaraussivua. Eri ryhmien käyttäjille näytetään eri navigointivalikko, koska työntekijöillä on käytössään ylimääräisiä sivuja hallintaominaisuuksia varten ja kirjautumattomilla ei ole lainkaan navigointia kirjautumista käyttävässä osiossa. Työntekijällä voi olla myös korotettu hallintaoikeus, mikä mahdollistaa uusien työntekijöiden lisäämisen. Tämä on oletusarvoisesti vain yhdellä pääkäyttäjällä.

Uusia käyttäjätilejä voidaan lisätä järjestelmään käyttäjähallintasivulta, ja aikaisemmin annettuja käyttäjätietoja voidaan muokata jälkikäteen. Osa tiedoista tarkistetaan palvelimen päässä sen varmistamiseksi, että syötetty tieto mahtuu sen käyttämään tietokantakenttään, ja että se olisi oikeassa muodossa, eikä sisältäisi tahattomia merkkivirheitä. Tiedon oikeellisuutta ei kuitenkaan käytännössä voida ohjelmallisesti varmistaa, koska kuka tahansa voi syöttää valheellista tietoa oikeassa muodossa.

Merkistön vaikutus

Kun käytettynä merkistönä on UTF-8, pitää se muistaa huomioida PHP:n merkkijonofunktioita käytettäessä. Esimerkiksi merkkijonon pituuden palauttava funktio **strlen** antaa **ä**-kirjaimen pituudeksi 2, koska se merkitään UTF-8:ssa kahdella tavulla, kun taas useatavuista merkistöä varten tarkoitettu funktio **mb_strlen** palauttaa kirjainten todellisen määrän.

Työntekijäkäyttäjillä on mahdollisuus generoida uusi salasana kenelle tahansa käyttäjälle. Tämä voi olla tarpeellista, mikäli käyttäjä unohtaa salasanansa. Uusi salasana voidaan valita lähetettäväksi sähköpostitse kyseiselle käyttäjälle, tai tarvittaessa se voidaan näyttää suoraan ruudulla työntekijälle.

JavaScriptiä sekä Ajaxia on järjestelmässä käytetty erityisesti kalenterinäkymien yhteydessä käytettävyyden parantamiseen, mutta näiden tekniikoiden turhaa käyttöä on pyritty välttämään. Ajaxin ladatessa sivulle uutta sisältöä näkyy sivun otsikon vieressä latausanimaatio aktiivisuuden osoittamiseksi käyttäjälle.

3.3 Tietokantarakenne

Ajanvarausjärjestelmä käyttää relaatiotietokantaa käyttäjätilien sekä aikavaraustietojen tallentamiseen. Lahtonen (2002: 30) kertoo, että tietokannoissa tulee välttää tiedon turhaa toistoa, ja tämän takia tietokantarakenteen pitäisi noudattaa tiettyjä normaalimuotoja. Tietokantarakennetta on tässä järjestelmässä kuitenkin tarkoituksella hieman yksinkertaistettu ja normalisointia ei ole kaikin osin kirjaimellisesti noudatettu.

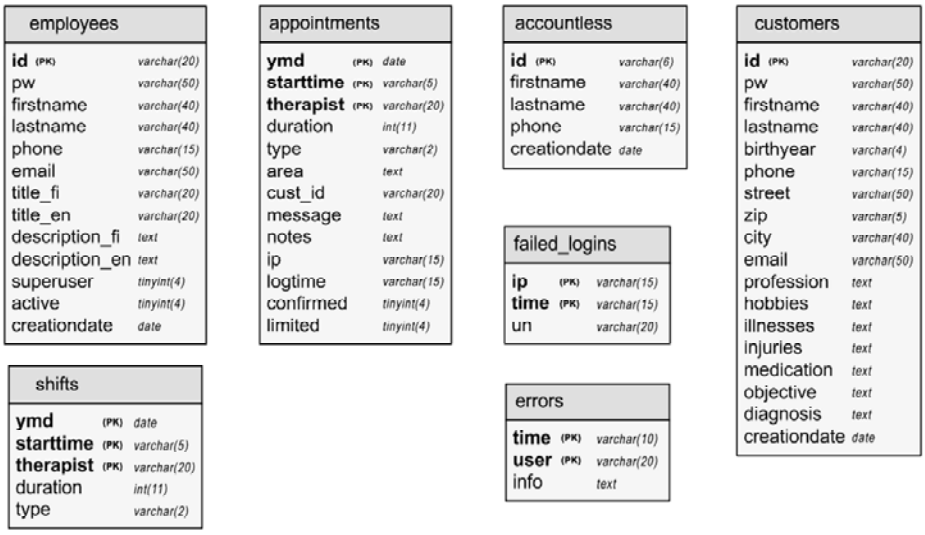
Tämä johtuu siitä, että järjestelmässä käytettävän tiedon rakenne on suhteellisen yksinkertainen, ja tarkoituksena oli pitää myös tietokannan rakenne selkeänä ja kyselyt nopeina. Tietojen hakeminen kerralla useista tauluista on raskaampaa kuin yhdestä (Lahtonen 2002: 31). Toistuvia tietoja ei eri tauluissa esiinny, mutta esimerkiksi asiakkaiden paikkakunta on tallennettu suoraan asiakastauluun, eikä tätä varten ole tehty omaa postinumeroon perustuvaa taulua. Myös aikavaraustauluun hierottavat alueet on tallennettu ensimmäisen normaalimuodon vastaisesti vain yhteen kenttään yksinkertaisena pilkuilla erotettuna listana, mistä ohjelmallisesti erotetaan eri alueet tarpeen mukaan. Näiden kohtien erottaminen erillisiksi tauluiksi ei tuntunut lainkaan hyödylliseltä tai järkevältä. Sen sijaan tietokannan taulut ovat nyt loogisia kokonaisuuksia.

Tietokantataulut

Jokaiselle käyttäjäryhmälle on tietokannassa oma taulunsa (Kuvio 1), koska erityyppisistä käyttäjistä tallennetaan osittain eri tietoja ja tyhjiä kenttiä taulujen riveissä tulee välttää. Rekisteröimättömistä käyttäjistä tallennetaan vain nimi ja puhelinnumero sekä järjestelmän generoima asiakkaan yksilöivä id-numero ja tunnuksen luontipäivä. Rekisteröityjen käyttäjien taulusta löytyy puolestaan yhteensä jopa 18 kenttää eri tietoja varten ja työntekijätaulussa on sekä samoja että eri kenttiä kuin asiakkailla.

Tietokannassa on myös taulut työntekijöiden työaikoja sekä asiakkaiden aikavarauksia varten. Työaikataulussa on vain ajankohta- ja tyyppitiedot sekä työntekijän tunnus, kun puolestaan varaustaulussa on näiden lisäksi myös tieto hierottavasta asiakkaasta, hierottavista alueista sekä kentät asiakkaan viestiä ja hierojan merkintöjä varten. Teknisen toteutustavan vuoksi varaustaulussa on myös pari ylimääräistä tietokenttää, mitä käytetään varausmäärän rajoituksissa ja varausten vahvistuksessa.

Oma taulunsa on myös epäonnistuneilla kirjautumisyrityksillä, mihin tallentuu vain käyttäjän IP-osoite, ajankohta sekä käyttäjätunnus, jolla yritettiin kirjautua. Tätä taulua käytetään suojaamaan käyttäjätilejä murtautumiselta, mistä kerrotaan seuraavassa alaluvussa.



Kuvio 1 Järjestelmän käyttämät tietokantataulut

Koska sivusto ja PHP-tiedostot käyttävät merkistönä UTF-8:aa, on se luonnollisesti asetettu myös tietokannan merkistöksi. Kun tietokantayhteys avataan PHP:n kautta, tulee yhteys valmistaa vielä UTF-8-merkistöllä koodatun tiedon lähetykseen erillisellä tietokantakomennolla **SET NAMES 'utf8'**, jotta tietokanta ei enää yrittäisi kääntää merkkejä ennen tauluihin tallentamista.

Tietokantaa käytettäessä on myös järkevää varautua virhetilanteisiin niin, että jos yhteyttä ei syystä tai toisesta saada, ei käyttäjälle kuitenkaan näytetä PHP:n tuottamaa virheilmoitusta. Tietokannan antamia virheilmoituksia ei ole myöskään syytä näyttää sivuston käyttäjälle sellaisenaan. Näissä tilanteissa ilmoitetaan asiasta käyttäjälle tilanteeseen sopivalla selkokielisellä ilmoituksella häiriöstä.

3.4 Tietoturvan huomiointi

Käyttäjältä saadut syötteet tarkistetaan säännöllisiä lausekkeita käyttämällä. Näin voidaan varmistua, että syöte on sallitun mittainen ja sisältää vain sallittuja merkkejä. Tämän lisäksi tietokantakyselyissä käytettävät syötteet ajetaan vielä **mysql_real_escape_string**-funktion läpi ennen kyselyn lähettämistä, jotta ne olisivat varmasti turvallisia. Syötteiden tarkistusfunktio on järjestelmässä tehty tietokantayhteydestä vastaavan luokan julkiseksi metodiksi, koska tietokantakyselyiden yhteydessä syötteentarkistusta yleensä tarvitaan. Metodissa on määritelty useita eri säännöllisiä lausekkeita, ja parametrien avulla metodille kerrotaan, mitä niistä tulee käyttää. Metodi palauttaa arvon **true** tai **false** sen mukaan läpäiseekö syöte tarkistuksen vai ei.

```
Esimerkki: säännölliset lausekkeet

public function checkInput($type, $string) {


    $regex['id'] = "/^[a-zA-Z0-9_]{4,20}$/";
    $regex['birthyear'] = "/^(19|20)\d\d$/";
    $regex['phone'] = "/^[0-9]{6,15}$/";
    $regex['name'] = "/^[ \pL',.-]{1,40}$/";
    $regex['zip'] = "/^[0-9]{5}$/";
    $regex['time'] = "/^([01][0-9]|2[0-3]):([0-5][0-9])$/";
    ...

    $check = false;
    if (isset($regex[$type]))
        $check = preg_match($regex[$type], $string);

    return $check;


}
```

Tietokantatunnukset

Tietokantayhteyden muodostukseen tarvittavat tunnukset on sijoitettu palvelimella julkisen kansion ulkopuolelle, jotta niihin ei olisi mahdollista päästä käsiksi ilman, että kirjautuu sisään palvelimelle. Tunnukset sisältävä tiedosto on vielä koodattu 64-bittisesti, jotta tunnukset eivät näkyisi suoraan selkokielisenä kirjautuneillekaan ihmisille. Koodaus on kuitenkin helppo purkaa, jos tietää mitä tekee, ja tietokantayhteydestä vastaavasta luokasta muodostettu olio suorittaa kyseisen operaation aina avatessaan uuden tietokantayhteyden.

Käyttäjien salasanat

Salasanan uudelle käyttäjälle järjestelmä luo automaattisesti. Salasanasta tulee kahdeksanmerkkinen, ja järjestelmä arpoo merkit isojen ja pienten kirjainten sekä numeroiden joukosta, eikä kaksi peräkkäistä merkkiä voi olla samoja. Ensisijaisesti järjestelmä lähettää luodun salasanan suoraan käyttäjän sähköpostiin, mutta erikoistapauksia varten on hierojalle myös annettu mahdollisuus näyttää salasana ruudulla tunnusten luonnin yhteydessä, mikäli käyttäjällä ei ole sähköpostiosoitetta.

Käyttäjät voivat vaihtaa salasanansa kirjauduttuaan järjestelmään. Salasanan minimipituudeksi on määritelty neljä merkkiä, eikä sille tehdä muita tarkistuksia. Näin käyttäjät voivat halutessaan käyttää salasanana vaikka pankkikorttinsa tunnuslukua. Sen vahvempaa salasanaa ei tässä järjestelmässä ole käyttäjiltä syytä vaatia, sillä järjestelmässä säilytettävät tiedot eivät ole erityisen arkaluontoisia, ja tärkeämpää onkin se, että käyttäjät muistaisivat salasanansa.

Salasanojen suojaus

Ennen tallentamista tietokantaan salasanaan liitetään satunnaisia heksadesimaaleja sisältävä merkkijono, ja se ajetaan **sha1**-funktion läpi. Näin

saatuun hash-arvoon liitetään vielä salasanaan lisätty merkkijono, ja tämä tallennetaan tietokantaan. Sama menettely toistetaan käyttäjän sisäänkirjautumisen yhteydessä antamalle salasanalle sillä erolla, että satunnaisen merkkijonon lisäämisen sijasta käytetään tietokannasta löytyvää aikaisemmin käytettyä merkkijonoa. Mikäli lopputulos vastaa tietokannasta löytyvää merkkijonoa, on käyttäjä syöttänyt oikean salasanan.

Järjestelmä pitää kirjaa epäonnistuneista sisäänkirjautumisyrityksistä, ja mikäli samasta IP-osoitteesta tulee lyhyen ajan sisällä useita yhteen tunnukseen kohdistuneita epäonnistuneita kirjautumisia, järjestelmä lakkaa sallimasta sisäänkirjautumisen kyseisellä tunnuksella tästä IP-osoitteesta vähäksi aikaa. Oletusarvoisesti viidentoista minuutin aikana saa tulla enintään viisi epäonnistunutta kirjautumista.

Liikenteen suojaus

Järjestelmässä on tarkoitus ottaa käyttöön tietoliikenteen salaava SSL- tai TLS-tekniikka, koska salasanaa ei voi luotettavasti ja yksinkertaisesti suojata muilla tavoin sen siirtyessä selaimelta palvelimelle. Järjestelmä ei kuitenkaan ole vielä Internet-käytössä, eikä edes tulevan domainin nimi ole vielä selvillä, joten sertifikaattiakaan ei voida vielä hakea niitä myöntävältä yritykseltä ja SSL/TLS:n käyttöönotto jää näin ollen toteutettavaksi myöhempänä ajankohtana.

4 Toiminnallisuuden esittely

Ajanvarausjärjestelmä on tarkoitettu liitettäväksi yrityksen tulevalle WWW-sivustolle Ajanvaraus-linkin taakse, mutta varsinaisen sivuston ulkoasu ja navigointi puuttuu vielä tehdystä käyttöliittymästä. Järjestelmän ulkoasu on tässä vaiheessa siis hyvin pelkistetty, ja ulkoasusuunnittelu tapahtuu vasta varsinaisen sivuston tekemisen yhteydessä, mikä ei kuulu tämän opinnäytetyön aihepiiriin. Järjestelmä sisältää oman navigointirakenteensa, mikä aukeaa varsinaisen sivuston navigoinnin rinnalle sen jälkeen, kun käyttäjä kirjautuu järjestelmään. Tässä luvussa kuvaillaan järjestelmän oman navigoinnin sisältämien sivujen toiminnot käyttäjän perspektiivistä.

4.1 Navigointi

Navigointialue on mallikäyttöliittymässä sivun vasemmassa reunassa, ja eri käyttäjätyypeille tämä alue tulostuu erilaisena (Kuvio 2). Kirjautumaton käyttäjä näkee vain kentät, joihin voi kirjoittaa käyttäjätunnuksen ja salasanan, mutta onnistuneesti järjestelmään kirjautuneet käyttäjät saavat käyttöönsä linkit sivuille, joita voi kirjautuneena käyttää järjestelmässä navigoimiseen.

Kuvio 2 Kirjautumattoman käyttäjän, asiakaskäyttäjän ja henkilökuntakäyttäjän navigointialue

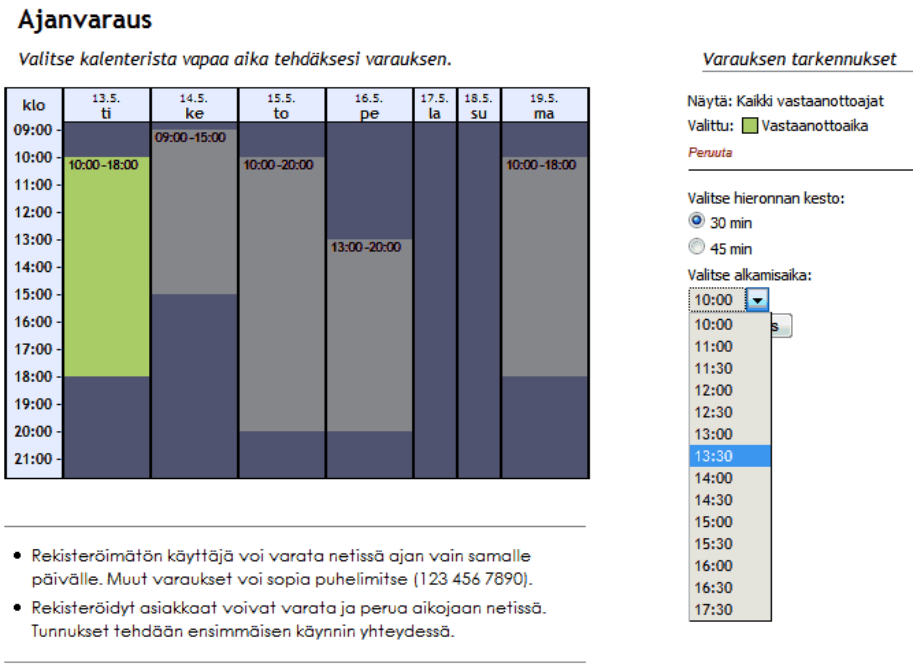
Mikäli käyttäjän kirjautumisyritys epäonnistuu tai tämän yhteys aikakatkaistaan käyttämättömyyden vuoksi, tulee tästä ilmoitusteksti kirjautumiskenttien yläpuolelle.

Mikäli kirjautumaton tai väärän käyttäjätason käyttäjä yrittää ladata sivun, jonka katselu vaatii tietyn käyttäjätason sisäänkirjautumista, ohjautuu tä-

mä suoraan ajanvarauksen etusivuna toimivaan viikkokalenteriin, joka näkyy kaikille käyttäjille.

4.2 Varausten tekeminen viikkokalenterista

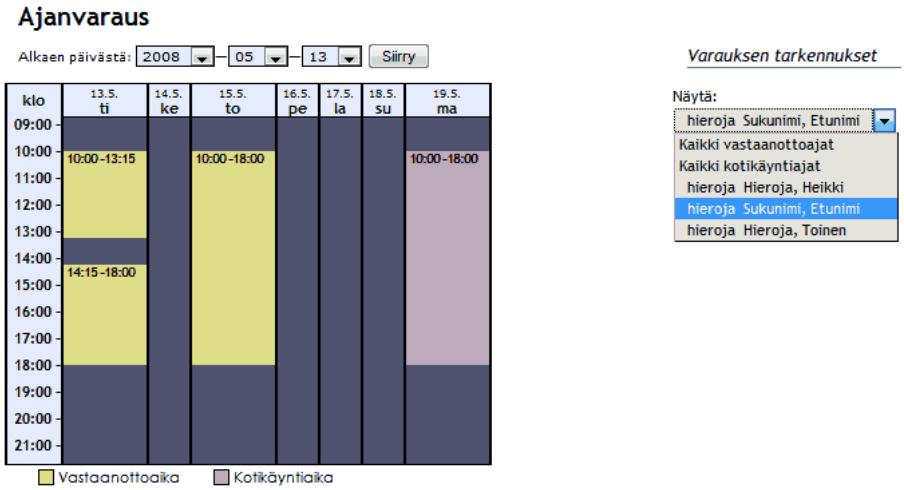
Viikkokalenteri on järjestelmän olennaisin osa, ja myös ensimmäinen sivu, joka aukeaa, kun käyttäjä saapuu ajanvarauspalveluun (Kuvio 3). Kalenterissa näkyy seitsemän päivän jakso ja se alkaa aina kuluvasta päivästä. Vapaat ajat näkyvät värillisinä palkkeina kalenterissa. Kirjautumattomat käyttäjätkin näkevät varattavissa olevat ajankohdat, mutta eivät voi kuitenkaan varata WWW-sivun kautta aikoja muille kuin samalle päivälle. Samalle päivälle tehtävä varaus voi alkaa aikaisintaan tunnin päästä varaushetkestä. Toimeksiantajan toivomuksesta kirjautumattomille käyttäjille järjestelmä sallii korkeintaan kaksi varausta päivää kohti mahdollisista väärinkäytöksistä koituvan haitan minimoimiseksi.



Kuvio 3 Varauksen tekeminen viikkokalenterista (kirjautumaton käyttäjä)

Varattavissa olevilla ajoilla ja varauksilla on kaksi mahdollista tyyppiä – vastaanottoaika ja kotikäyntiaika. Kirjautumattomat käyttäjät näkevät ainoastaan vapaat vastaanottoajat, eivätkä voi eritellä näkymää hierojien mukaan. Kun asiakas tai työntekijä kirjautuu sisään, voi tämä valita näkymäksi kaikki vastaanottoajat, kaikki kotikäyntiajat tai minkä tahansa yksittäisen hierojan aikakalenterin, missä eri varaustyypit on erotettu toisistaan värein (Kuvio 4). Työntekijällä on viikkokalenterissa myös mahdollisuus siirtyä päivämäärässä eteen tai taaksepäin. Myös kirjautuneille asiak-

kaille on järjestelmään tehty kahden varauksen rajoitus viikon aikaväliä kohti. Ylimääräisiä varauksia voi kuitenkin sopia puhelimitse.



Kuvio 4 Työntekijän viikkonäkymä

Varauksen teko

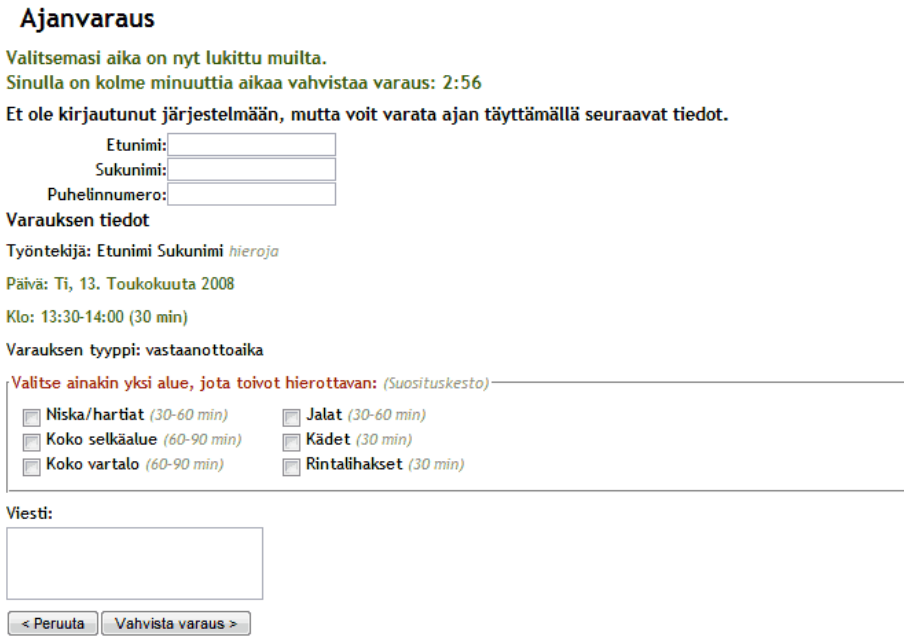
Varauksen tekeminen tapahtuu vapaata aikajaksoa klikkaamalla. Kalenterin yläpuolella on tästä maininta, mutta myös aikapalkit vaihtavat väriä interaktiivisuuden osoittamiseksi hiiren siirtyessä niiden päälle. Kun vapaata aikajaksoa klikataan, muuttuu valittu aikajakso kirkkaammaksi, kun taas muut ajat vaihtavat väriä harmaammaksi valinnan selkiyttämiseksi käyttäjälle. Kalenterin viereen avautuu myös tieto varauksen tyypistä, Peruuta-painike ja lyhyen latauksen jälkeen Ajaxin avulla tietokannasta haetut kyseiselle käyttäjäryhmälle, varaustyypille sekä aikajakson pituudelle sopivat kestovaihtoehdot.

Käyttäjän valittua hieronnan keston, hakee sovellus uudelleen mahdolliset alkamisajat tietokantaa ja matemaattisia laskutoimituksia hyödyntäen. Varaus voi aina alkaa joko heti vapaan aikajakson alusta tai päättyä sen loppuun. Jos vapaa aikajakso on tarpeeksi pitkä, voi varaus alkaa sen keskeltä niin, että sekä alkuun että loppuun jää tilaa vähintään 45 minuutin vastaanottoajalle. Jos kyseessä on kotikäyntiaika, niin aikaa on jäätävä vähintään 60 minuutin hieronnalle. Toimeksiannon mukaisesti järjestelmä laskee myös vastaanottoaikavarausten väliin automaattisesti aina 15 minuutin tauon sekä kotikäyntien väliin 60 minuutin tauon. Näin ollen vapaan kotikäyntiaikajakson reunoille on jäätävä vähintään kahden tunnin väli, jos uusi aika varataan aikajakson keskeltä.

Aikavalinnan vahvistamiseksi käyttäjän tulee klikata painiketta, josta pääsee varauksen seuraavaan vaiheeseen. Kalenteri häviää näkyvistä, kun järjestelmä lataa käyttäjälle uuden sivun. Tässä vaiheessa varaus tallennetaan järjestelmään varmistamattomana, jotta kukaan ei voisi samanaikaisesti

yrittää varata päällekkäistä aikaa. Jos varausta ei varmisteta kolmen minuutin sisällä, vapauttaa järjestelmä varauksen automaattisesti. Sama tapahtuu myös välittömästi, mikäli käyttäjä klikkaa seuraavalla sivulla olevaa Peruuta-painiketta.

Kolmen minuutin aikalaskuri näytetään käyttäjälle uuden sivun yläosassa (Kuvio 5). Jäljellä oleva aika päivittyy palvelimelta Ajaxin kautta kaksi kertaa sekunnissa. Näin saadaan varmasti todellinen jäljellä oleva aika, ja vaikka samanlaisen laskurin olisi voinut tehdä pelkällä JavaScriptillä, ei sen täsmällisyys olisi välttämättä ollut taattu joka selaimessa.



Kuvio 5 Kirjautumattoman käyttäjän ajanvaraussivu

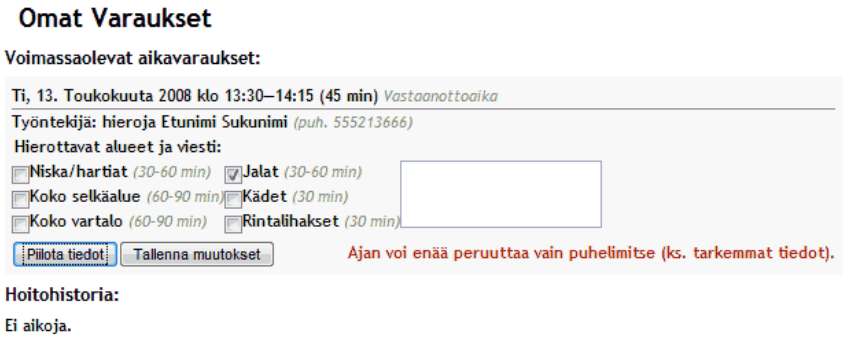
Seuraavaksi sivulla pitää kirjautumattoman käyttäjän syöttää nimensä sekä puhelinnumeronsa. Jos käyttäjä on kirjautunut sisään, näyttää järjestelmä sen sijaan kirjautuneen käyttäjän nimen ja puhelinnumeron, jotka järjestelmään on tallennettu. Sivulle tulostuu myös valitun hierojan nimi, ja mikäli kalenterissa näkymänä on ollut kaikki ajat ja usealla hierojalla on vapaata työaikaa käyttäjän valitsemana ajankohtana, järjestelmä arpoo hierojan, jolle varaus tehdään. Lisäksi valittu ajankohta ja varauksen tyyppi näytetään käyttäjälle tässäkin vaiheessa.

Käyttäjä voi seuraavassa kohdassa valita alueet, joita toivoo hierottavan. Vaihtoehtoja on kuusi, ja niiden perään on merkitty ohjeelliset kestosuositukset. Aluetta ei ole kuitenkaan välttämätöntä valita aikavarauksen hyväksymiseksi, joskin se on toivottavaa. Viimeiseksi sivulla annetaan käyttäjälle vielä mahdollisuus jättää vapaamuotoinen viesti hierojalle. Vahvistuspainiketta klikkaamalla järjestelmä varaa ajan käyttäjälle pysyvästi ja lisää annetut tiedot varaukseen.

Käyttäjä näkee tämän jälkeen vielä yhteenvedon varauksen tiedoista, ja mikäli kyseessä on kirjautunut asiakas, näkee tämä myös ohjeen varauksen perumisesta.

4.3 Omien varausten ja käyntien tarkastelu

Kirjautunut asiakaskäyttäjä näkee aktiiviset aikavarauksensa sekä aikaisemmat käyntinsä Varaukset/Käynnit-sivulta (Kuvio 6). Sivun yläosassa näytetään aikajärjestyksessä tehdyt varaukset, ja mikäli varaukseen on vielä tarpeeksi aikaa, sen voi perua kyseisen toiminnon painiketta klikkaamalla. Oletusarvoisesti aikaraja on edellisen vuorokauden kello 16. Varauksen ylimääräisiä tietoja, eli hierottavia alueita sekä viestiä hierojalle, voi myös muuttaa aina hieronnan alkamiseen asti. Selkeyden vuoksi nämä tiedot on varausriveiltä piilotettu, mutta ne saa klikattua esille varauskohtaisesti. Näiden joukossa näytetään myös hieroja, jolle aika on varattu. Aikavarauksen alkamisen jälkeen vain hieroja voi tehdä käyntitietoihin muutoksia, mutta käyttäjä näkee tiedoista edelleen hierojan sekä hierotut alueet.



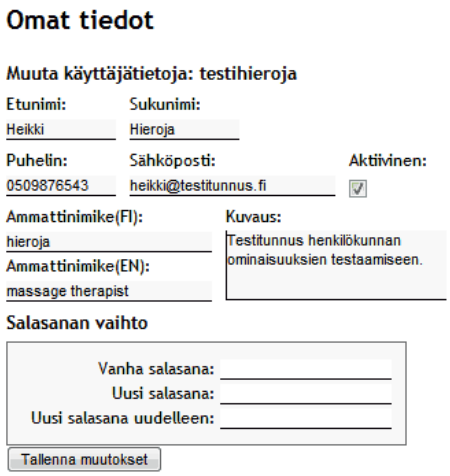
Kuvio 6 Asiakaskäyttäjän Varaukset/Käynnit-sivu, josta ainoan varauksen tarkemmat tiedot on klikattu näkyviin

4.4 Omien tietojen muuttaminen järjestelmään

Sekä työntekijät että asiakaskäyttäjät voivat nähdä itsestään järjestelmään tallennetut tiedot. Näitä tietoja pystyy käyttäjä itse vapaasti muuttamaan. Tämä tapahtuu Omat tiedot -sivulta, missä käyttäjälle näytetään tietokentät lomakemuodossa (Kuvio 7). Käyttäjä voi suoraan tehdä muutoksia tietoihin ja tallentaa ne lopuksi Tallenna muutokset -painikkeella, jolloin tiedot päivittyvät järjestelmään automaattisesti.

Ainoat asiakaskäyttäjien pakolliset kentät ovat nimi ja syntymävuosi, koska nämä taatusti löytyvät jokaiselta käyttäjältä. Työntekijöillä ei ole edes syntymävuosi-tietoa. Muut kentät voi jättää tyhjäksi, mutta mikäli niihin

jotain täytetään, syötteiden merkkimuotoinen oikeellisuus tarkistetaan palvelimella siinä vaiheessa, kun tiedot yritetään tallentaa. Jos jossakin kentässä oli vääränlaisia merkkejä, pyydetään käyttäjää vielä korjaamaan kyseinen kenttä ennen kuin tiedot tallennetaan järjestelmään.



Kuvio 7 Työntekijän Omat tiedot -sivu

Salasanansa voi käyttäjä myös muuttaa tällä sivulla. Voimassa oleva salasana täytyy täyttää ensin, jotta voidaan olla suhteellisen varmoja siitä, ettei luvaton henkilö yritä muuttaa salasanaa. Käyttäjähän on voinut esimerkiksi julkiselta koneelta poistuessaan jättää selaimen auki kirjautumatta ulos ja joku muu näin päästä muuttamaan käyttäjän tietoja. Uusi salasana pitää myös antaa kahteen kertaan, jotta voidaan tarkistaa, ettei se sisällä tahattomia kirjoitusvirheitä.

Hierojalla on tiedoissaan myös merkintä siitä, onko tämä aktiivinen. Hierojat, jotka eivät ole aktiivisia, eivät näy asiakkaille viikkokalenterissa. Hierojan käyttäjätiliä ei siis tarvitse poistaa, vaikka tämä jäisi pitkälle lomalle tai lopettaisi työt kokonaan.

4.5 Työntekijän työaikakalenteri

Työntekijöillä on oma kalenteri työaikojen valintaa varten (Kuvio 8). Kalenterissa näkyy kerralla kuluva viikko sekä kaksi seuraavaa viikkoa, mutta siinä voi liikkua viikko kerrallaan eteen ja taaksepäin nuolinappien avulla. Työntekijät näkevät toistensa työajat, ja aktiivinen työntekijä vaihdetaan kalenterin yläpuolella olevasta pudotusvalikosta.

Kuvio 8 Työajat-sivu: tyhjiä päiviä on valittuna

Päivät, joina valitulla hierojalla on jo vähintään yksi työaikaväli merkittynä, näytetään kalenterissa värillä ja reunuksella korostettuna. Tällaisen päivän klikkaaminen lataa kalenterin alapuolelle tiedon valitulle hierojalle merkityistä päivän työajoista. Aiemmin merkittyjä työaikoja voi muuttaa tai poistaa vastaavien painikkeiden avulla. Uuden työajan lisääminen tapahtuu täyttämällä ajan tiedot ja klikkaamalla Lisää-painiketta. Työaikajaksot eivät voi kuitenkaan mennä toistensa kanssa päällekkäin.

Vapaata päivää kalenterista klikkaamalla se saa ympyrän numeronsa ympärille, ja kalenterin alapuolelle latautuu ajanlisäysmahdollisuus. Vapaita päiviä voi kuitenkin valita kerralla useita, ja näin saman työajan voi kerralla asettaa usealle eri päivälle. Valittua päivää uudelleen klikkaamalla valinta sen päivän kohdalta poistuu. Kalenterin oikeasta reunasta löytyy myös joka viikon kohdalta **M-F**-tekstillä merkitty pikavalintapainike, jota klikkaamalla kyseisen viikon vapaiden arkipäivien valinnat muuttuvat käänteisiksi. Tätä käyttämällä voi siis helposti valita tyhjän viikon kaikki arkipäivät yhdellä klikkauksella. Useita päiviä valittaessa kalenterin oikealle puolelle tulee lista kaikista valittuna olevista päivistä, koska kalenterissa liikkuminen on mahdollista, eivätkä kaikki päivät välttämättä näy siinä kerralla.

Kun tyhjille päiville lisätään uusi aika, päivittyy kalenteri Ajaxia käyttäen heti ajan tasalle. Sama tapahtuu myös silloin, kun työaikoja poistetaan.

4.6 Työntekijän aikavarauskalenteri

Työntekijöiden sivu aikavarausten hallintaa varten sisältää samannäköisen kalenterin kuin työaikojen valinnassa käytetty sivu (Kuvio 9). Tässä kalenterissa aikavarauksia sisältävät päivät on korostettu samalla tavalla kuin työaikoja sisältävät päivät työaikakalenterissa, mutta mahdollista on valita kuitenkin vain yksi päivä kerrallaan. Kun kalenterista valitaan päivä, kyseisen päivän varaukset näkyvät listana kalenterin alapuolella. Myös hierojalle mahdollisesti merkitty työaika kyseiselle päivälle näytetään listan yläpuolella.

Jos päivälle ei ole varauksia, näytetään listan paikalla vain linkki "ei varauksia", josta klikkaamalla hieroja voi lisätä uuden varauksen päivälle. Jos varauksia kuitenkin on, näytetään niistä kellonaika ja pituus, hierottavan asiakkaan nimi ja käyttäjätunnus, ja mikäli asiakas on varatessaan jättänyt viestin, näkyy nimen perässä myös asiasta ilmoittava kirjekuori-ikoni, joka toimii linkkinä viestiin. Päivän varausten viestit latautuvat palvelimelta jo varauslistan mukana, mutta ne ilmestyvät näkyviin vasta linkkiä klikattaessa, jolloin kyseinen viesti aukeaa JavaScriptin avulla sivun oikeassa reunassa olevaan alueeseen.

Kuvio 9 Varaukset-sivu, jolla näkyy yksi varaus päivälle ja sen tarkemmat tiedot

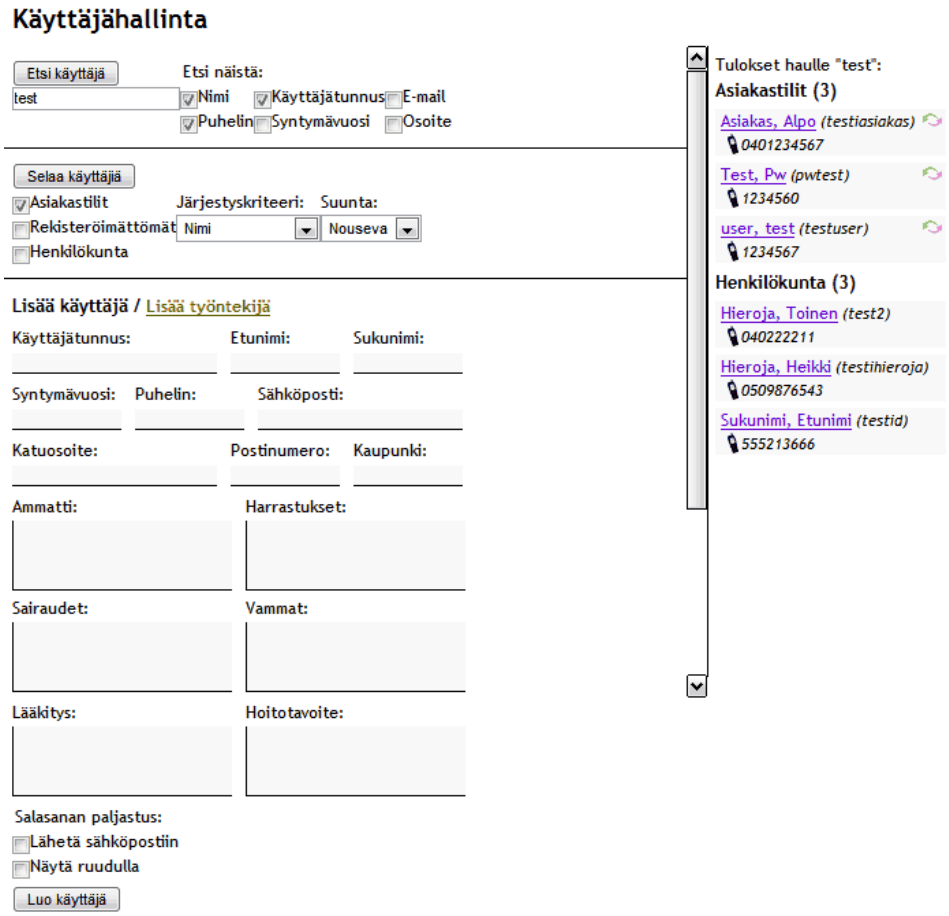
Varausrivien lisäksi varauslistassa näytetään rivit väliin jääville aikajaksoille. Näillä riveillä näkyy vain kellonaika ja jakson kesto. Sekä varauksien että varaamattomien aikajaksojen kellonajat toimivat linkkeinä, joita klikkaamalla voi varauksen tietoja muuttaa tai tehdä kokonaan uuden varauksen. Varausten muutokset ja lisäykset tehdään varauslistan alapuolella olevassa alueessa, johon varauksen tiedot tai varaamattoman aikavälin ajankohta latautuu. Hieroja voi tätä kautta vaihtaa valmiista varauksesta ajankohtaa, asiakasta tai hierottavia alueita. Jos asiakkaalla, jolle varaus tehdään, ei vielä ole käyttäjätunnusta, voidaan uusi tilapäiskäyttäjä luoda varauksen teon yhteydessä täyttämällä nimi ja puhelinnumero. Varaukseen voi hieroja liittää myös oman tekstimuotoisen muistiinpanon, joka ei näy asiakkaalle. Varauslistassa oleva käyttäjän nimi on myös linkki, josta klikkaamalla käyttäjän tiedot voidaan avata sivun oikean reunan alueeseen.

4.7 Työntekijän käyttäjähallintaominaisuudet

Työntekijöillä on käytössään sivu, jonka kautta he voivat etsiä olemassa olevia käyttäjiä sekä tarkastella ja muuttaa heidän tietojaan (Kuvio 10). Myös uusien käyttäjien luominen tapahtuu tältä sivulta.

Käyttäjähaku

Kun käyttäjää etsitään hakusanoilla, voidaan valita tietokentät, joista sanoja etsitään. Haku kohdistetaan automaattisesti kaikkiin käyttäjäryhmiin turhien valintavaihtoehtojen karsimiseksi, sillä sanahaulla on tarkoitus löytää vain pieni joukko ehdot täyttäviä käyttäjiä. Tulokset on kuitenkin aina jaoteltu käyttäjäryhmien mukaan. Sanahaku on tarkoitettu ensisijaisesti tietyn käyttäjän etsimiseen nimen tai puhelinnumeron perusteella. Täsmällisten osumien lisäksi haku palauttaa myös käyttäjät, joiden tiedoista hakusana löytyy osana pitempää merkkiriviä. Jos käyttäjää haetaan usealla sanalla, pitää kaikkien hakusanojen löytyä valittujen kenttien joukosta.



Kuvio 10 Käyttäjähallintasivu, jossa on suoritettu haku sanalla "test". Tässä näkyy myös asiakaskäyttäjien tietokentät.

Jos hakutuloksia on paljon, vain kymmenen ensimmäistä käyttäjää haetaan näkyviin, ja löytyneiden käyttäjien kokonaismäärä näytetään hakutulosten yhteydessä. Tämä pitää tietokantahaun ja sivun latautumisen nopeana vaikka käyttäjätilien määrä järjestelmässä kasvaisi suureksi. Tässä tilanteessa sivulle tulostuu myös linkki seuraavalle tulossivulle, minne siirryttäessä järjestelmä lataa Ajaxia käyttäen seuraavat kymmenen käyttäjää näkyviin edellisten tilalle.

Käyttäjien selaus

Käyttäjiä voidaan myös selata niin, että rajataan pois yksi tai kaksi käyttäjäryhmää. Selaus näyttää aina kaikki valittuihin ryhmiin kuuluvat käyttäjät. Tulokset voi selatessa kuitenkin järjestää käyttäjän nimen tai muun tiedon perusteella nousevaan tai laskevaan järjestykseen. Edellisessä kappaleessa mainittu kymmenen käyttäjän kerralla näkyminen on käytössä myös selaustoiminnossa.

Käyttäjän lisäys

Käyttäjähaun ja -selauksen alapuolella sijaitsee tyhjä käyttäjätietolomake uuden käyttäjätilin lisäämistä varten. Kenttien tarkistus tapahtuu palvelimella, mutta vasta kun tiedot lähetetään. Tietojen lähetys kuitenkin tapahtuu Ajaxin välityksellä, joten koko sivu ei lataudu uudelleen kun käyttäjä luodaan. Jos pakollisia tietoja puuttuu tai kentistä löytyy virheellisiä merkkejä, ei tietoja vielä tallenneta, vaan käyttäjälle palautuu lomake korjattavat kohdat värillä korostettuna.

Käyttäjien tietojen katselu ja muuttaminen

Käyttäjät, jotka on tuotu sivun oikeaan reunaan haulla tai selaustoiminnolla, näytetään linkkeinä, joita klikkaamalla kyseisen käyttäjän tiedot latautuvat niihin kenttiin, joilla muussa tapauksessa tehdään uuden käyttäjän lisääminen. Tämän jälkeen työntekijä voi halutessaan muuttaa käyttäjän tietoja ja tallentaa muutokset. Tällä lomakkeella voidaan myös käyttäjän salasana generoida uudelleen, mikäli tämä on unohtanut sen. Uusi salasana voidaan joko lähettää käyttäjän sähköpostiin tai näyttää suoraan ruudulla.

5 Yhteenveto

Opinnäytetyöhön liittyvän toimeksiannon tavoitteena oli kehittää WWW-sovellus, jonka kautta hierontayrityksen asiakkaat voivat varata aikoja hierojalle. Opinnäytetyön tavoitteena oli selvittää toimeksiannon kaltaisen WWW-sovelluksen tekemisessä tarvittavien tekniikoiden oikeaoppisia käyttötapoja tietoturvan sekä yhteensopivuuden optimoimiseksi. Tavoitteet toteutuivat siihen nähden hyvin, että aihe oli työläs sekä toimeksiannon että raporttisisällön osalta.

Työtä aloittaessani tunsin tietäväni perusteet PHP-kielestä sekä sen käyttämisestä MySQL-tietokannan kanssa. WWW-merkintäkielet ja niihin liittyvät standardit olivat myös pääosin tuttuja. Laajuudeltaan toimeksiannon kaltaista sovellusta en kuitenkaan ollut aikaisemmin tehnyt, ja työtä tehdessä tuli vastaan paljon uusia olennaisia asioita WWW-sovellusten ohjelmointiin liittyen, mitä tässä raportissa on tuotu esille.

Olio-ohjelmoinnista minulla oli työtä aloittaessani hyvin vähän käytännön kokemusta, minkä johdosta luokkien ja metodien suunnittelu ennalta oli vaikeaa. Pyrin hahmottelemaan oppimani teorian perusteella luokkarakennetta, mutta työn edetessä huomasin, etteivät valinnat varmasti johtaneet parhaaseen mahdolliseen toteutustapaan. Useat metodit olisivat järjestelmässä voineet olla staattisia, jolloin niitä olisi voinut käyttää ilman, että luokasta tehdään ensin olio. Näin olisi voinut välttyä olioiden lähettämiseltä parametrien mukana niitä käyttäville metodeille (Liite 1: Ohjelmointiesimerkkejä, Esimerkki 3: Varausten määrän palauttaminen). Myös luokkia olisi voinut tehdä useampia, koska ainakin kalenteriluokasta tuli turhan massiivinen.

Järjestelmän kaksikielisyyden vuoksi käyttäjälle näytettävät tekstit piti jakaa taulukoihin, joista kielen perusteella oikea teksti haetaan. Tekstien sijoittamisen niitä käyttäviin metodeihin olisi voinut olla järkevää korvata tekemällä oma luokkansa pelkästään tekstien säilytykseen ja hakemiseen. Tietojen tallentamista sessiomuuttujaan olisi myös voinut käyttää tehokkaammin hyödyksi sen sijaan, että lähettää metodeille paljon tietoa parametreinä. Järjestelmän toiminnallisuuteen ei näillä seikoilla silti ole suoranaista vaikutusta.

Lisäksi huomasin, että työhön tarvittavan ajan arviointi on näin suuressa projektissa äärimmäisen vaikeaa, ja jopa mahdotonta ilman aikaisempaa kokemusta. Vaikka pyrin kiirehtimään, ylittyi aika-arvio huomattavan paljon, enkä saanut järjestelmää viimeisteltyä niin valmiiksi kuin alkuperäinen tarkoitus oli. Se on kuitenkin perustoiminnallisuudeltaan kunnossa ja vastaa toimeksiantajan esittämiä toivomuksia. Osa ylimääräisistä ominaisuuksista vaatii vielä hieman lisätyötä. Myös järjestelmän toiminnallisuus niissä tapauksissa, että JavaScript ei ole käytössä, on vielä puutteellinen.

Järjestelmä ei olisi kuitenkaan aivan suoraan mennyt käyttöön, vaikka se olisikin tullut täysin valmiiksi, koska tätä ennen on vielä suunniteltava ja

toteutettava yrityksen graafinen ilme sekä varsinainen WWW-sivusto, johon järjestelmä integroidaan. Tulevaisuudessa järjestelmä siis liitetään osaksi yrityksen tulevaa WWW-sivustoa niin, että se noudattaa samaa ulkoasua kuin muukin sivusto sekä säilyttää myös varsinaisen sivuston navigoinnin oman navigointirakenteensa lisäksi. Näin käyttäjä ei tunne joutuneensa sivuston ulkopuolelle, kuten monissa muissa Internetistä löytyvissä ajanvarauspalveluissa saattaa tapahtua.

Järjestelmä tulee tämän vuoksi käyttämään samoja CSS-tiedostoja kuin muukin sivusto, ja sivuston ulkoasun suunnittelussa otetaan huomioon myös ne järjestelmän osat, mitä ei muulla sivustolla esiinny. Dokumenttityypiltään muun sivuston tulee noudattaa samaa määrittelyä kuin mitä järjestelmän ohjelmoinnissa on käytetty, eli XHTML 1.0 Strictiä, ja merkistön on suositeltavaa olla järjestelmän käyttämä UTF-8, jotta ongelmia ei merkistön vuoksi ilmaannu. Koska muustakin sivustosta sivustosta on tarkoitus tulla kaksikielinen, on todennäköisesti järkevintä tehdä myös se palvelinpuolen ohjelmointia hyödyntäen.

Koodin optimointi saattaa tulla uudelleenkäytettävyyden kannalta kysymykseen, mutta tämän projektin osalta sillä ei ole varsinaista merkitystä, sillä se ei tuo lisäetua järjestelmään loppukäyttäjän kannalta. Mikäli järjestelmää kuitenkin kehitetään laajempaa tuotantoa ajatellen, vaatii se varmasti ainakin luokkarakenteen ja sessiotietojen käytön tehostamista.

Itselleni opinnäytetyön ja toimeksiannon tekeminen olivat erittäin hyödyllisiä prosesseja oppimisen kannalta, ja myös toimeksiantaja saa työn tuloksena Internetin kautta käytettävän ajanvaraus- ja asiakashallintajärjestelmän käyttöönsä. Työssä ei tullut vastaan suuria ongelmakohtia, mutta kokonaistyömäärä oli suurin haaste.

Lähteet

- Asleson, Ryan, Schutta, Nathaniel T. 2006/2007. Ajax – Tehokas hallinta. Jyväskylä: Gummerus Kirjapaino Oy.

- Babin, Lee, Good, Nathan A., Kromann, Frank M. & Stephens Jon 2005. PHP 5 Recipes: A Problem-Solution Approach. Berkeley: Apress.

- Duffy, Scott 2003. How to Do Everything with JavaScript. Berkeley: McGraw-Hill/Osborne.

- Goodman, Danny 2003. JavaScript and DHTML Cookbook. Sebastopol: O'Reilly & Associates, Inc.

- Hovi, Ari 2004. SQL-opas. 1. painos. Jyväskylä: Docendo Finland Oy.

- Howard, Michael, LeBlanck, David & Viega, John 2005. 19 Deadly Sins of Software Security. Emeryville: McGraw-Hill/Osborne.

- Hudson, Paul 2006. PHP in a Nutshell. 1. painos. Sebastopol: O'Reilly Media, Inc.

- Lahtonen, Tommi 2002. SQL. 1. painos. Jyväskylä: Docendo Finland Oy.

- Schafer, Steven M. 2005. Web Standards Programmer's Reference: HTML, CSS, JavaScript, Perl, Python, and PHP. Indianapolis: Wiley Publishing, Inc.

- What is TLS/SSL? 2003. [online] [viitattu 20.5.2008]. technet2.microsoft.com/windowsserver/en/library/ed5ae700-e05e-45ef-b536-45795dbb99a21033.mspx?mfr=true

- XHTML™ 1.0 The Extensible HyperText Markup Language (Second Edition) 2002. [online] [viitattu 20.5.2008]. www.w3.org/TR/xhtml1

- Zeldman, Jeffrey 2003. Designing with Web Standards. Indianapolis: New Riders Publishing.

Liitteet

Liite 1: Ohjelmointiesimerkkejä

Esimerkki 1: Sessio- ja evästetietojen käyttö

```
// sessinit.php – jokaisen sivun alussa suoritettavia komentoja
session_start(); // otetaan sessiot käyttöön

// tässä välissä suoritetaan muitakin toimintoja
// uloskirjauduttaessa tyhjennetään sessiotiedot ja aloitetaan uusi sessio
if (isset($_POST['logout']) || (isset($_SESSION['mode']) &&
                                        $_SESSION['mode'] == 'forcelogout')) {
    $_SESSION = array();
    session_destroy();
    session_start();
}
// alustetaan käytettävät sessiomuuttujat oletusarvoilla jos niitä ei ole vielä asetettu
if (!isset($_SESSION['mode'])) $_SESSION['mode'] = 'basic';
if (!isset($_SESSION['lang'])) $_SESSION['lang'] = 'fi';
if (!isset($_SESSION['user'])) $_SESSION['user'] = NULL;
if (!isset($_SESSION['super'])) $_SESSION['super'] = 0;
if (!isset($_SESSION['time'])) $_SESSION['time'] = time();
if (!isset($_SESSION['therapist'])) $_SESSION['therapist'] = '-';

$langs = array('fi', 'en'); // mahdolliset kielivaihtoehdot

// jos kielivalinta on tehty sallitulla arvolla, tallennetaan valinta sessiomuuttujaan sekä evästeeseen
if (isset($_GET['lang']) && in_array($_GET['lang'], $langs)) {
    $_SESSION['lang'] = $_GET['lang'];
    setcookie('lang', $_SESSION['lang'], time()+8640000); // eväste voimassa 1000 päivää
}
// jos kieltä ei ole juuri valittu, haetaan kielivalinta evästeestä jos se löytyy
else if (isset($_COOKIE['lang']) && in_array($_COOKIE['lang'], $langs))
    $_SESSION['lang'] = $_COOKIE['lang'];

// jos käyttäjätunnus ja salasana on täytetty, tarkistetaan tunnusten oikeellisuus
if (isset($_POST['loginun'], $_POST['loginpw'])) {
    require_once('class/DBModule.php'); // liitetään skriptiin luokka, jotta sitä voidaan käyttää
    $dbmodule = new DBModule();         // luodaan uusi olio DBModule-luokasta
    $_SESSION['mode'] = $dbmodule->confirmSignIn($_POST['loginun'], $_POST['loginpw']);

    // jos tunnukset täsmäsivät, asetetaan käyttäjätunnus sessiomuuttujaan
    if (in_array($_SESSION['mode'], array('user', 'admin')))
        $_SESSION['user'] = $_POST['loginun'];

    // jos kirjautunut käyttäjä oli työntekijä, asetetaan ylimääräisiä muuttujia
    if ($_SESSION['mode'] == 'admin') {
        $therapists = $dbmodule->getArrayFromDB("SELECT id, active, superuser
                                                 FROM employees
                                                 WHERE id = '{$_SESSION['user']}';");
        if ($therapists[0]['active'] == 1) $_SESSION['therapist'] = $_SESSION['user'];
        if ($therapists[0]['superuser'] == 1) $_SESSION['super'] = 1;
    }

    // kirjautumisen jälkeen tehdään uudelleenohjaus kalenterisivulle, jotta kirjautumistietoja ei
    // voida lähettää uudelleen siirtymällä selaimella edelliselle sivulle
    $_SESSION['redirect'] = 1;
    $redirecturl = 'http://' . $_SERVER['HTTP_HOST'] .
                    rtrim(dirname($_SERVER['PHP_SELF']), '/\\') . '/calendar.php';
    header('Location: '.$redirecturl); exit;
}
```

Esimerkki 2: Sisäänkirjautumisen käsittely

```
// class DBModule: confirmSignIn – käyttäjän sisäänkirjautumisen prosessoiva metodi
public function confirmSignIn($username, $password) {

    $mode = 'failedlogin';                  // oletuksena palautettava arvo
    $ip = $_SERVER['REMOTE_ADDR'];          // käyttäjän ip-osoite
    $time = time();                         // aika nyt
    $starttime = time() - (60 * 15);        // aika 15 minuuttia sitten
    $salt = NULL;

    // tarkistus, että käyttäjän antama käyttäjätunnus sisältää vain sallittuja merkkejä
    if ($this->checkInput('id', $username)) {

        // haetaan muuttujaan tietokannasta samalla ip-osoitteella ja käyttäjätunnuksella tehdyt
        // epäonnistuneet kirjautumisyritykset 15 minuutin ajalta
        $failed = $this->getArrayFromDB("SELECT * FROM failed_logins
                                WHERE ip = '$ip' AND un = '$username'
                                AND time BETWEEN '$starttime' AND '$time';");

        // jos epäonnistuneita yrityksiä löytyi yli 5, hypätään tämän lohkon yli
        if (count($failed) <= 5) {

            // haetaan annetulla käyttäjätunnuksella riviä asiakaskäyttäjä-tietokantataulusta
            $users = $this->getArrayFromDB("SELECT id, pw
                                FROM customers WHERE id = '$username';");

            if (count($users) == 1) {       // jos käyttäjä löytyi
                if (isset($users[0]['pw']))  // erotetaan suola salasanakentästä
                    $salt = $this->extractSalt($users[0]['pw']);

                // jos annetun salasanan ja suolan hash vastaa kannasta löytynyttä arvoa
                if ($this->encodePWforDB($password, $salt) == $users[0]['pw'])
                    $mode = 'user';         // käyttäjä on asiakaskäyttäjä
            }
            else { // jos käyttäjää ei löytynyt
                // haetaan tunnuksella riviä työntekijä-tietokantataulusta
                $admins = $this->getArrayFromDB("SELECT id, pw
                                FROM employees WHERE id = '$username';");
                if (count($admins) == 1) {          // jos käyttäjä löytyi
                    if (isset($admins[0]['pw']))    // erotetaan suola
                        $salt = $this->extractSalt($admins[0]['pw']);

                    // jos salasana ok
                    if ($this->encodePWforDB($password, $salt) == $admins[0]['pw'])
                        $mode = 'admin';  // käyttäjä on henkilökuntakäyttäjä
                }
            }
        }
        else $mode = 'banned'; // jos yrityksiä enemmän kuin 5 palautetaan tämä

        // samaten jos meneillään on viides yritys eikä sekään onnistunut
        if ($mode == 'failedlogin' && count($failed) == 5) $mode = 'banned';

        // lisätään epäonnistunut kirjautuminen tietokantaan
        if ($mode == 'failedlogin' || $mode == 'banned')
            $this->sendQuery("INSERT INTO failed_logins
                        VALUES ('$ip', '$time', '$username');");
    }

    return $mode; // palautetaan käyttäjän kirjautumistila
}
```

Esimerkki 3: Varausten määrän palauttaminen

```
// class Calendar: countApps – käytetään tarkistettaessa onko asiakaskäyttäjän viikon
// maksimivarausmäärä jo täyttynyt tai onko kuluvalla päivällä jo maksimimäärä rekisteröimättömien
// käyttäjien tekemiä varauksia
private function countApps(DBModule $dbmod, $mode, $username = NULL) {

    $time = date('H:i');        // kellonaika (hh:mm)
    $x = $this->today['ymd'];   // kuluva päivämäärä
    $z = $this->getDateFromDay(1);
    $z = $z['ymd'];             // seuraava päivä
    // jos tarkistetaan asiakaskäyttäjän viikon varausmäärä:
    if ($mode == 'week' && $dbmod->checkInput('id', $username)) {

        $y = $this->getDateFromDay($this->week - 1);
        $y = $y['ymd'];         // viikkokalenterin viimeinen päivä
        // lähetetään tietokantakysely
        $data = $dbmod->getArrayFromDB("SELECT COUNT(ymd) AS 'count'
            FROM appointments WHERE cust_id = '$username' AND confirmed = 1
            AND (ymd BETWEEN '$z' AND '$y' OR (ymd = '$x' AND starttime > '$time'));");
    }

    // jos tarkistetaan kuluvan päivän rekisteröimättömien käyttäjien tekemiä varauksia:
    else if ($mode == 'day') // lähetetään tietokantakysely
        $data = $dbmod->getArrayFromDB("SELECT COUNT(ymd) AS 'count'
            FROM appointments WHERE ymd = '$x' AND confirmed = 1 AND limited = 1;");

    return $data[0]['count']; // palautetaan tietokannan palauttama lukumäärä
}
```