TAMPEREEN
AMMATTIKORKEAKOULU

TUTKINTOTYÖRAPORTTI

REKISTERÖINTIPALVELUN LUOMINEN WWW-YMPÄRISTÖÖN
CASE: WWW.KIPPIS.FI

Akimatti Katainen – Robert Mast

Tietojenkäsittelyn koulutusohjelma
joulukuu 2005
Työn ohjaaja: Petri Heliniemi

TAMPERE 2005

Tampereen

ammattikorkeakoulu

Tekijä(t)	Akimatti Katainen ja Robert Mast
Koulutusohjelma(t)	Tietojenkäsittely
Tutkintotyön nimi	Rekisteröintipalvelun luominen www- ympäristöön Case: www.kippis.fi
Työn valmistumis-kuukausi ja -vuosi	joulukuu 2005
Työn ohjaaja	Petri Heliniemi

Sivumäärä: 67

TIIVISTELMÄ

Työn taustalla on toimeksiantajalle (Pirkanmaan Osuuskauppa) rakennettu Internet -sivusto www.kippis.fi, johon käyttäjiä sitouttaakseen haluttiin rakennettavan rekisteröintipalvelun. Sivustoon liitettiin ulkoisesti yhtenevä, moduuleista koostuva rekisteröintijärjestelmä.

Tämän tutkintotyön tavoitteena on selvittää, mitä www-ympäristöön rakennetavan rekisteröintijärjestelmän tekeminen vaatii ja kuinka siitä saadaan mahdollisimman toimiva ja käyttäjäystävällinen. Samalla selvitimme, mitä tekniikoita käyttämällä järjestelmästä saadaan mahdollisimman tehokas.

Työn pohjana käytettiin nykyaikaisen web-suunnittelun standardeja ja ilmaisia teknologioita (PHP, MySQL, XHTML, CSS), joiden avulla projekti oli mahdollista toteuttaa ja jotka takasivat palvelun päivitettävyys pitkälle tulevaisuuteen.

Rekisteröintipalvelu toteutettiin onnistuneesti ja ensimmäisen viikon aikana sivustolle oli rekisteröitynyt noin viisikymmentä luonnollista henkilöä.

Tätä työtä voidaan hyödyntää kaikissa niissä www-projekteissa, joissa on tarvetta liittää modulaarinen rekisteröintijärjestelmä työn osaksi.

Avainsanat PHP MySQL XHTML CSS Rekisteröinti

Tampere

Polytechnic

Author(s)	Akimatti Katainen and Robert Mast
Degree Programme(s)	Business Information Systems
Title	Registration service creation in a www –environment - the www.kippis.fi case
Month and year	December 2005
Supervisor	Petri Heliniemi

Pages: 67

ABSTRACT

The background of this thesis was an internet site www.kippis.fi created for the cooperative retail society of Pirkanmaa. They wanted a registration service to be added to the site. A modular registration system was therefore built.

This final thesis aims to clarify what a registration service needs and how it becomes effective and userfriendly. It also clarifies the techniques that will be used and their utilization in the project.

The up-to-date web design standards and free technologies formed the basis of our thesis.
It also goes through the techniques (PHP, MySQL, XHTML, CSS) which enabled projects success and ensured updating the service in the future.

Registration service was succesfully executed and first week's number of registrated people was about 50.

This thesis can be utilized in every www -project, which requires modular registration service as a part of their work.

Keywords PHP MySQL XHTML CSS Registration

Sisällysluettelo

1 Johdanto

Tutkintotyö perustuu www.kippis.fi-sivustoon, joka sai alkunsa Pirkanmaan Osuuskaupan toimeksiantona vuoden 2004 syksyllä. Tästä alkoi suunnittelu yhdessä Pirkanmaan Osuuskaupan kanssa sivuston toiminnasta ja sisällöstä. Alun perin ideana oli toteuttaa varsin erilainen sivusto, joka olisi keskittynyt drinkkireseptien esittelyyn ja niiden lisäämiseen käyttäjien toimesta. Tämä idea kuitenkin kariutui lakiteknisiin seikkoihin drinkkien sisältäessä yli 22-tilavuusprosenttisia alkoholijuomia, joiden epäsuorakin mainonta on lain mukaan kielletty. Lopulta päädyttiin kokonaisuuteen, jossa informoitaisiin Pirkanmaan Osuuskaupan omistamien ravintoloiden tuotteista ja niiden tapahtumista.

Toimeksiannon tavoitteeksi muodostui siis toimivan, helppokäyttöisen sekä kiinnostavan sivuston luominen, joka tarjoaisi selkeää informaatiota POK:n ravintoloiden asiakkaille. Tärkeätä oli myös käyttäjien sitouttamiseen liittyvät asiat, jolla takaisimme verkkoliikenteen myös tulevaisuudessa.

Aikaisemmissa palavereissa keskusteltiin myös tulevien käyttäjien rekisteröitymismahdollisuudesta, jonka suunnittelu ja toteutus sopivat aiheeksi tutkintotyölle. Projekti oli mahdollista jakaa kahteen osaan siten, että ensimmäinen osa toteutettiin projektiopintoina ja jälkimmäinen tutkintotyönä.

Projektina rakennettiin sivuston yleisilme, rakenne, toiminnallisuus sekä hallinnointiin liittyvät työkalut. Tutkintotyöhön sisältyy rekisteröintipalvelun luominen käyttäjille.

Rekisteröityminen oli aiheena erittäin mielenkiintoinen ja se tarjosi sopivasti haasteita, johtuen järjestelmän meille tuntemattomasta toimintatavasta ja sen vaatimuksista teknologian ja ympäristön suhteen.

Toimeksiannon kokonaistavoitteen muodosti Kippis.fi-sivuston toteuttaminen sekä käyttäjien sitouttamiseen liittyvät palvelut. Tutkintotyön kannalta oleellisena toimeksiannon osatavoitteena oli rekisteröintiin liittyvien ongelmien selvittäminen ja ratkaiseminen sekä toimivan rekisteröintipalvelun luominen.

Seuraavassa tarkastellaan rekisteröintipalveluun liittyviä ongelmia hieman tarkemmin.

- **Toimivuus**. Kuinka suunnitella palvelusta tarpeeksi yksinkertainen, helppokäyttöinen ja “riisuttu”, ettei mitään ylimääräistä laiteta mukaan. Kuinka saada skaalautuva sivusto, joka mukautuu eri käyttöolosuhteisiin, ja jonka sekä rakenne että sisältö ovat ymmärrettävissä ilman kuvamateriaalia.

- **Tehokkuus**. Kuinka toteuttaa mahdollisimman vähistä elementeistä kasattu palvelu, joka vaatisi tietokoneelta tai kaistan leveydeltä vähän resursseja.

- **Käyttäjäystävällisyys**. Kuinka toteuttaa visuaalisesti tyylikäs, mutta kevyt (nopea) ulkoasu selvillä opastuksilla ja kuinka tehdä rekisteröinnistä helppoa ja miellyttävää.

- **Muokattavuus**. Kuinka taata järjestelmä, joka on helposti muokattavissa ja laajennettavissa tulevien palveluiden jatkokehityksen kannalta.

Ongelmien ratkaisemiseksi päädyttiin käyttämään aikaisempaa tuntemusta ihmisten käyttäytymisestä rekisteröitymistilanteessa, omia kokemuksia rekisteröintipalveluista, lähdekirjallisuutta sekä tutkimustyössä tarkasteltuja tekniikoita. Teknisesti palvelu toteutettiin MySQL:n, PHP:n, XHTML:n sekä CSS:n avulla. Teknologioiden ilmaisuus, vaikutti suurilta osin niiden valintaan. Käytetyistä teknologioista kerrotaan tarkemmin seuraavassa luvussa.

Sivuston pohjaksi rakennettiin tietokanta MySQL-kielellä. MySQL on ilmainen Open Source-pohjainen tietokantakieli, joten se sopi loistavasti käytettäväksi tietokannan luomiseen ja kyselyiden tekemiseen. Tietokantaa ja palveluun liittyviä dynaamisia toiminnallisuuksia ohjataan PHP (Hypertext Preprocessor)-skriptikielellä. Sivuston varsinainen rakenne rakennettiin XHTML -kielellä sen alusta- ja päätelaiteriippumattomuuden vuoksi. XHTML on tulevaisuuden standardi, joten yhteensopivuus pitkälle eteenpäin on turvattu. Rakenteen eri elementit muotoiltiin CSS (Cascading Style Sheet) kielellä. Ulkoisen CSS -tiedoston avulla elementtien muotoilu tarvitsee tehdä vain kerran, jonka jälkeen ne näkyvät samalla tavalla joka sivulla. Näin säästetään paljon aikaa ja vaivaa. Elementit muotoiltiin tavalla, jolla suurin osa selaimista osaa näyttää sivun yhtenäisesti, eikä merkittäviä eroavaisuuksia esiinny.

Tutkintotyössä keskitytään kattavasti vain yhteen tapaan rakentaa rekisteröintipalvelu. Tietokannat tai -rakenteet voidaan toteuttaa monella eri tavalla, samoin PHP-skriptit on mahdollista koostaa monilla eri tavoin. Lisäksi kaikki rekisteröintipalvelut eivät käytä lainkaan email-varmennusta, joka omaan versioomme on lisätty. Tämä ei siis ole se yksi ja ainut tapa.

Tutkintotyön päätavoitteena oli luoda kattava kuvaus koko siitä prosessista, jota rekisteröintipalvelun luominen vaatii niin teknologialta kuin yleiseltä suunnittelulta. Tavoitteena oli myös selvittää vaatimuksia, jotka tulee ottaa huomioon suunniteltaessa. Sivutavoitteeksi muodostui Kippis.fi-sivuston luomisessa käytettyjen teknologioiden esittely sekä yleiseen suunnitteluun liittyvät näkökohdat. Sivutavoitteeseen kuului myös selvittää käyttäjien sitouttamiseen liittyviä toimintatapoja, jotka rekisteröintipalvelu mahdollistaa ja niiden käyttöä Kippis.fi-sivustolla. Tutkintotyö voi toimia taustamateriaalina rekisteröintipalveluja ja suuria dynaamisia sivustoja rakennettaessa.

Tutkintotyön päätavoitteen ratkaisemiseksi työssä on paneuduttu järjestelmän vaatimuksiin ja selvitetty rakennetta. Tutkintotyön teoriaosuus tarjoaa keinon ymmärtää järjestelmän prosessien toimintaa helposti omaksuttavassa muodossa ja käytetyt tekniikat antaa viitteitä ja tietoa toiminnoista, joita on kuvattu rekisteröitymisprosessin teknisessä kuvauksessa. Tekninen kuvaus sisältää tarkan kuvauksen prosessin etenemisestä kooditasolla.

Sivutavoitteen ratkaisemiseksi tutkintotyö esittelee perusteknologiat, joita Kippis.fi-sivustolla on käytetty sisältäen PHP:n, MySQL:n, XHTML:n ja CSS:n. Työ tarjoaa myös pintapuolisen, mutta kattavan kuvauksen sivuston rakenteesta, ulkoasusta sekä toiminnasta. Sitouttamiseen liittyvä tavoite ratkaistaan esittelemällä yleisesti käytettyjä toimintatapoja ja kuinka niitä on sovellettu Kippis.fi -sivustolla.

Tutkintotyön lähteinä on käytetty paljon kirjallisuutta, Internet-sivuilla esiintyvää materiaalia sekä omiin kokemuksiin perustuvaa tietoutta. Pääasiallisesti käytetyt lähteet ovat olleet hyviä. Yleisestä web-suunnittelun teoriasta löytyi hyviä teoksia ja materiaalia,

joka vaikutti suunnitteluun jo ruohonjuuritasolta lähtien. Muistakin teknologioista löytyi kattavaa ja varsin informatiivista materiaalia. Ainut asia, josta emme juuri löytäneet materiaalia, oli rekisteröinnin teoria ja eettiset periaatteet. Olisimme kaivanneet lisää tietoa rekisteröinnin oikeaoppisesta rakentamisesta mutta, mielestämme onnistuimme rakentamaan varsin käyttökelpoisen palvelun.

Tänä päivänä kentällä vaikuttavilta suunnittelijoilta, kuten Jeffrey Veen:ltä, saimme tuoretta näkökantaa asioihin suunnittelullisista lähtökohdista alkaen. Kun selailee verkossa web-suunnittelua käsitteleviä sivustoja, voi huomata hänen nimensä useissa eri suunnitteluun liittyvissä yhteyksissä. Olemme muutenkin saaneet käsityksen, jonka mukaan häntä pidetään korkeassa arvossa web-suunnittelun saralla. Neuvoja ja oppeja kannattaa siis soveltaa kohtuullisen laaja-alaisesti.

Lähteenä on käytetty hänen kirjoittamaa kulttiteosta Inside Web Design. Kirja painotti nykyajan suunnittelun perustuvan rakenteen ja esitystavan erotteluun. Esitetyt teoriat ovat tämän päivän standardeja, joten oikeassa Veen oli muutama vuosi takaperin. Suunnittelu pilkotaan kolmeen osaan: Rakenteeseen, esitystapaan sekä käytökseen. Tästä voidaan muodostaa parit: rakenne - xhtml, esitystapa - css sekä käytös - skriptit. Näillä asioilla on mahdollista taata standardoitu suunnittelu pitkälle tulevaisuuteen mahdollisimman monille päätelaitteille.

Rakenteen muodostamiseksi kävimme läpi useita verkkolähteitä ja kirjan, joka käsitteli XHTML-kieltä. Pääsääntöisesti lähteet olivat erittäin hyödyllisiä ja ne kävivät läpi kieltä erittäin laajasti. Omassa työssämme rakenne muodosti kohtuullisen pienen, mutta sitäkin tärkeämmän osan. Sivusto rakennettiin yhden rakenteellisesti oikein muodostetun (XHTML 1.0 Transitional) taulukon pohjalle. Taulukkoon sijoitellut elementit muotoiltiin tämän jälkeen CSS:llä.

CSS:ää käsittelevät kirjat ja muut lähteet olivat mielestämme aika pitkälti eri sanoilla samoista asioista kertovia. Jotkin teokset olivat enemmän pintaraapaisua kaikesta mutta myös kattavampaa selontekoa löydettiin esim. Keith Schengili-Robertsin kirjasta Core CSS – Cascading Style Sheets (2nd edition).

PHP alati muuttuvana Open Source-kielenä omaa parhaat lähteet oikeastaan verkossa. www.php.net on myös kehittäjien aktiivisesti päivittämä sivusto, joten sisältöä sivustolla on erittäin paljon. Sieltä löytyikin kaikki mitä tarvittiin. Verkkolähteiden pienoisen epäluotettavuuden vuoksi tukeuduimme vielä Julie Melonin SAMS Teach Yourself PHP, MySQL and Apache-kirjaan, johon kaikki tarvittavat teknologiat oli laitettu sanoihin kansiin. Kirja on hyvä perusteos.

MySQL oli hivenen tuntemattomampi alue, joten kiinnostus aihetta kohtaan oli suuri. Tarjolla olevat materiaalit olivat erittäin hyödyllisiä. Julie Melonin SAMS Teach Yourself MySQL in 24 Hours-kirja tarjosi hyvän pohjatiedon ja verkosta löytyi reilusti tukevaa materiaalia.

Yleisesti lähteinä käytettiin mm. Wikipediaa eli vapaata tietosanakirjaa verkossa ja se tarjosi luotettavaa tietoa niin PHP:stä kuin MySQL:stä.

2 Käytetyt teknologiat

Kippis.fi -sivuston suunnittelu ja toteutus ovat vaatineet useiden teknologioiden samanaikaista käyttöä. Teknologioiksi on valittu vapaaseen lähdekoodiin ja web-standardeihin kuuluvia ratkaisuja. Standardeista on käytettyinä CSS:ää ja XHTML:ää.
Web-standardien etuina voidaan saavutettava.fi -sivuston (…) mukaan pitää:

- web-dokumenttien pitkäikäisyyttä,
- yksinkertaisempaa ja luettavampaa koodia,
- mahdollisuutta luoda sivustoja, jotka ovat suurten ihmisjoukkojen ja, laitealustojen tavoitettavissa,
- oikeatoimisuutta riippumatta selaimesta tai sen versiosta.

Vapaan lähdekoodin teknologioista on edustettuna PHP ja MySQL. Vapaan lähdekoodin ohjelmistojen etuina ovat Open Options -sivuston (…) mukaan:

- mahdollisuus muokata tuotetta omien tarpeiden mukaan,
- ilmaisuus tai edullisuus,
- eri ohjelmien yhteistoiminnallisuus ja yhdisteltävyyden helppous,
- mahdollisuus modulaarisuuden avulla valita halutut toiminnallisuudet ja maksaa vain niistä,
- vapaa lisenssipolitiikka (mahdollisuus asentaa mikä tahansa määrä kopioita),
- verkko ystävällisyys.

2.1 PHP

PHP (Hypertext Preprocessor) on nykyään osittain Apache Software Foundationin (www.apache.org) kehittämä yleiskäyttöinen skriptikieli, joka soveltuu erityisesti web-sovelluskehitykseen. Kieli on suunnattu dynaamisten web-sivustojen toteuttamiseen eli sivustojen toiminnallisuuden rakentamiseen. PHP:n syntaksi muistuttaa C-kieltä, Javaa ja Perliä. PHP:llä voi dynaamisten web-sivustojen lisäksi tehdä myös komentorivisovelluksia ja jopa graafisia käyttöliittymiä, mutta parhaimmillaan se on web-sovelluskehityksessä.

PHP on lisenssivapaa Open Source-tuote, jota kuka tahansa voi käyttää maksutta myös kaupallisiin tarkoituksiin. PHP-tuki on saatavissa kaikkiin yleisimpiin web-palvelimiin. PHP on myös saatavissa useisiin eri käyttöjärjestelmiin.

Kielenä PHP on aloittelijoille helppo oppia. Se soveltuu kuitenkin hyvin myös ammattimaiseen käyttöön. Se sisältää mm. tuen useimmille yleisesti käytetyille tietokannoille. (Kollanus...)

PHP:n historia alkaa vuodesta 1995, jolloin tanskalais-kanadalainen ohjelmoija Rasmus Lerdorf kirjoitti muutaman Perl-skriptin jäljittääksen kävijöitä online-ansioluotteloonsa. Skriptien nimeksi muodostui Personal Home Page Tools. Hän tarvitsi kuitenkin enemmän ominaisuuksia ja kirjoitti C-kielellä implementaation, jonka avulla pystyi käsittelemään tietokantoja ja rakentamaan pienimuotoisia verkkosovelluksia.

Tämän PHP/FI (Personal Home Page / Forms Interpreter) -version lähdekoodin hän laittoi verkkoon kaikkien nähtäväksi, käytettäväksi ja korjattavaksi.

Vuonna 1997 PHP/FI:n versio 2.0 implementaatiolla arvioitiin olevan tuhansia käyttäjiä ympäri maailmaa ja noin 50 000 domainia (1 % kaikista domaineista), jotka ilmoittivat sen olevan palvelimella asennettuna. Vaikka useammat ihmiset osallistuivat koodaukseen, PHP/FI 2.0 oli paljolti yhden miehen projekti. PHP/FI 2.0 julkaistiin virallisesti marraskuussa 1997 useiden beta-versioiden jälkeen.

Seuraava versio oli PHP3, joka syntyi Andi Gutmansin, Zeev Suraskin ja Rasmus Lerdorfin yhteistuloksena. Andi Gutmans ja Zeev Suraski aloittivat koodin uudelleenkirjoittamisen, koska tunsivat PHP/FI 2.0:n liian tehottomaksi verkkomainonta-ohjelmien rakentamiseen, joita he tarvitsivat yliopistoprojektissaan. Nämä kolme miestä tekivät yhteistyötä ja julkaisivat PHP3:n virallisena PHP/FI 2.0:n seuraajana, jonka kehitys lakkasi siihen. PHP3 oli jo paljon edistyksellisempi. Se osasi käsitellä useita eri tietokantoja ja protokollia.

PHP3:n laajennettavuus houkutteli mukaan useita koodaajia, jotka kehittivät lisää moduuleja. Näitä auttavia käsiä pidetään (ehkä jopa kiistellysti) merkittävä PHP:n menestyksen kannalta. PHP nimettiin tuolloin uudestaan ja on pysynyt sen jälkeen samana. Vuoden 1998 loppuun mennessä PHP:tä käyttivät arvioilta sadat tuhannet verkkosivut, joka vastasi 10 %:ia kaikista WWW-palvelimista. Yhdeksän kuukauden beta-testauksen jälkeen virallinen versio PHP3:sta julkistettiin kesäkuussa 1998.

Myöhemmin samana vuonna Andi Gutmans ja Zeev Suraski alkoivat kirjoittaa jälleen PHP:n ydintä uudelleen. Tavoitteena oli nyt parantaa yhteensopivuutta monimutkaisten ohjelmien kanssa ja parantaa modulaarisuutta. Uusi moottori saavutti asetetut tavoitteet menestyksekkäästi ja se julkaistiinkin ensimmäisen kerran vuoden 1999 puolivälissä. Tämän moottorin (Zend Engine) päälle lisättiin paljon uusia ominaisuuksia ja näin uusi virallinen versio PHP 4.0 näki päivänvalonsa toukokuussa 2000. Se sisälsi tuen useille eri palvelintyypeille, paremman tietoturvan ja useita kielikäännöksiä, sekä osasi käsitellä http-sessioita, joista lisää luvussa 4.6.2: Muita käytettyjä tekniikoita.

Tänä päivänä PHP:tä käyttää sadat tuhannet kehittäjät ja useat miljoonat palvelimet (20 % Internetin domaineista).

Useiden väliversioiden ja pitkän kehityksen tuloksena uusin versio on PHP 5, joka julkistettiin heinäkuussa 2004. Ytimenä toimii sama Zend Engine 2.0, joka tukee nyt täydellisesti olio-ohjelmointia sekä sisältää esimerkiksi sisäänrakennetun tietokantamoottorin (SQLite). (PHP.net 2001a)

2.2 CSS

CSS (Cascading Style Sheets) on standardi ulkoasun määrittelykieli, joka tuli webmaailman tietoisuuteen joulukuussa 1996. Se pitää huolen sivuston ulkoasusta, fonteista, väreistä, elementtien sijoittelusta ja ylipäätään kaikesta siitä, miltä web-sivu näyttää. Termillä 'cascading' (verb. 'ryöpytä') on kaksi merkitystä. Yhdellä tyylitiedostolla voit hallita satojen tai jopa tuhansien sivujen ulkoasua yhdenaikaisesti, toisaalta yhdelle sivulle voidaan linkittää monta tyylitiedostoa. (HTMLSource 2000)

Linkittämällä kaikki sivut yhteen ulkoiseen css-tiedostoon, saadaan ylläpidosta huomattavasti helpompaa ja verkkoliikenteestä huomattavasti kevyempää. Sivujen latausajat saadaan minimoitua, jolloin saatetaan säästää satoja megabittejä verkkoliikenteessä ja täten säästää kustannuksissa. Huolimatta siitä, että CSS on erittäin tarkka ja tehokas kieli, sitä on helppo kirjoittaa käsin.

Selaimet alkoivat tukea CSS:n ensimmäistä versiota kunnolla vasta versioista 4.x eteenpäin. Internet Explorer 3.0 implementoi sitä ensimmäisenä, Netscape seurasi toisena ja lopulta seurasivat pienemmät selaimet. Nykyään molemmat alkavat olla täysin yhteensopivia kahden ensimmäisen standardin kanssa (CSS1 ja CSS2), vaikkakin muutamia ongelmia esiintyy vielä tänä päivänä.

W3C (The World Wide Web Consortium) esitteli CSS2:n toukokuussa 1998. Se laajensi esitysmahdollisuuksia ja toi tarkempia määrittelyjä elementeille. Selainten tuki oli aluksi aika kehnonlaista. Nykyään selaimet tukevat CSS2:sta entistä paremmin. Näihin lukeutuvat mm. Firefox, Mozilla, Opera sekä Safari. Selainten paremman tuen myötä CSS2:sta julkaistiin paranneltu versio CSS2.1 elokuussa 2002. Se sisältää oikeastaan vain asiat, jotka jäivät puuttumaan CSS2:sta. Näin CSS 2.1 tiukentaa virallista täsmennystä ja toimii hyvänä pohjana kehitettävälle CSS 3:lle. (Schengili-Roberts 2004: 7 - 9)

Jos selain ei ymmärrä CSS-koodia, jatkaa se sivun näyttämistä ilman. Tällöin sivuston rakenteella on erittäin tärkeä merkitys. Sivua suunniteltaessa tulee huomioida, että sivu on luettavissa ja navigoitavissa myös ilman tyylitiedostoja.

CSS:n hyödyt

Muotoilut on tarpeen määritellä vain kerran, mikä tekee CSS:stä tehokkaan. Tällöin sivut latautuvat huomattavasti nopeammin, koska koodia on vähemmän. Näin savutetaan etuja koodin lyhyyden, siisteyden ja tehokkuuden suhteen. Käytettäessä vain yhtä CSS-tiedostoa, sivut näyttävät aina samalta ja niiden ylläpito on helpompaa ja virheitä esiintyy vain harvoin.

CSS tekee sivuista luettavat myös muilla laitteilla, kuin pelkillä tietokoneilla ja selaimilla. PDA -laitteet, kännykät yms. osaavat lukea XHTML/CSS -yhdistelmää tehokkaasti. Lisäksi vammautuneet (esim. sokeat) ihmiset pystyvät ottamaan vastaan XHTML/CSS -teknologioilla tehtyä materiaalia.

2.3 XHTML

SGML (Standard General Markup Language) on HTML-kielen äiti. SGML:n tarkoituksena oli toimia pohjana asiakirjatyöstämiselle, ja se on toiminut kansainvälisenä standardina jo vuodesta 1986. Se määrittelee lähes kaikki tunnetut tavat kuvata kieltä ja sen, mitä tieto on (esim. kappaleet määritellään kappaleina ja otsikot otsikoina). Se oli kuitenkin epäkäytännöllistä. Lukuiset kuvausmahdollisuudet tekivät SGML:stä web-kirjoittajan näkökulmasta erittäin monimutkaisen. Epäkäytännöllisyyden lisäksi jokaisen dokumentin tuli sisältää tarkka DTD (Document Type Definition), jotta tagit voitaisiin tulkita oikein.

Niinpä olikin luonnollista kehittää SGML:n pohjalta verkkokirjoittamiseen paremmin sopiva kieli, HTML. Se on itse asiassa SGML:n sovellus, joissa DTD on valmiiksi upotettu selaimeen.

Kompastuskiveksi muodostuivat selaimet. Alkuaikoina selaimet tulkitsivat jokainen eri tavalla epätarkkaa HTML-koodia, joka aiheutti ihmisten innostuksen omien websivujen tekemiseen koodauksen näennäisestä helppoudesta johtuen. Riitti kun otti copy Paste -metodilla toisen sivuista lähdekoodin (todennäköisesti jo valmiiksi virheellisen), ja muokkasi tekstit omia ajatuksiaan vastaaviksi. Internet alkoikin pian olla virheellisiä sivuja pullollaan.

Nykyään ongelmana on, että vaikka selaimet näyttävät virheelliset sivut oikein, muut päätelaitteet eivät tee samoin. Palm-laitteet, kännykät, Web -TV jne. tarvitsevat validin rakenteen näkyäkseen niin kuin suunnittelija on niiden tarkoittanut näkyvän. Toinen iso ongelma on hakukoneissa. Virheellisellä koodilla tehdyistä sivuista hakokoneet voivat hakea vain yksittäisiä sanoja, joka tekee hausta valtavan työlästä. Oikein rakennetusta sivusta haku on mahdollista tagien kuvauksen mukaan, mikä nopeuttaa ja tehostaa hakutyötä. Oli siis suuri tarve tarkkaan ja täydelliseen kieleen.

XML (eXtensible Markup Language) tehtiin myös SGML:n pohjalta, mutta tavoitteena oli luoda helppo rakennekieli, joka pystyy kehittyneeseen tiedon esittämiseen ja manipulointiin. XML siis määrittelee, mitä jokin on, ei miltä jonkin tulisi näyttää. XML on laajennettavissa käyttäjän tarpeiden mukaan (omia tageja voi keksiä niin kauan kuin noudatetaan kielen perussääntöjä), eikä DTD:tä tarvita ollenkaan.

Viimeisenä askeleena yhdistettiin XML- ja HTML-kielet, jotta saatiin entistä parempi kieli. XHTML käyttää HTML:n sanastoa ja XML:n syntaksia. XHTML on yhteensopiva sekä XML:n että HTML:n kanssa ja se näkyy kaikissa selaimissa versioista 2.0 ylöspäin. Näin se kaventaa kuilua käyttäjien välillä, joilla on käytössään uudempia ja vanhempia selaimia. Toinen suuri XHTML:n etu on siirrettävyys. Yhden koodijoukon avulla kaikki selaimet ja verkkoon kytkettävät päätelaitteet esittävät asiakirjat samalla tavalla. (Boumphrey, Greer, Raggett, Raggett, Schnitzenbaumer, Wugofski, 2000: 25 - 33)

Tällä hetkellä XHMTL-versioita on kolme: 1.0, 1.1 ja Basic. XHTML2 on työn alla. Sen ensimmäinen julkinen luonnos julkistettiin elokuussa 2002. XHTML2:sta povataan laajamittaisinta muutosta HTML:n ja XHTML:n historiassa kautta aikojen.

XHTML 1.0 versio on jaoteltavissa kolmeen osakokonaisuuteen: 1.0 Strict, 1.0 Transitional sekä 1.0 Frameset. Strict noudattaa tiukimmin varsinaisia XML:n periaatteita estämällä mm. muotoiluelementtien käytön. Transitional on taaksepäin yhteensopiva ja sisältää jonkin verran muotoiluelementtejä. Frameset hyödyntää nimensä mukaisesti kehyksiä. XHTML 1.0 on sama kuin HTML 4.01, paitsi että se kirjoitettiin uudelleen XML:n sääntöjen mukaiseksi. Rajoitukset koskivat lähinnä isojen ja pienien kirjaimien käyttöä sekä tagien sulkemista.

XHTML Basic kehitettiin pääasiassa mobiililaitteita varten modulaarisena kuvauskielenä. Tarkoituksena oli muodostaa ydinmoduuli, jota kaikki päätelaitteet osaisivat käyttää. Tarvittaessa voitaisiin ladata haluttavat moduulit laitekohtaisesti. Tämän vuoksi XHTML Basic on erittäin pelkistetty.

XHTML 1.1 on viimeisin ns. virallinen versio. Pidemmälle viety moduulijako sekä standardista kiinni pitäminen ovat 1.1:n tunnusmerkkejä. Ulkoasu on muotoiltava tyylimäärittelyillä (CSS), koska 1.1 ei sisällä niitä ollenkaan. Tämä mahdollistaa tulevaisuudessa järkevämmän laajennettavuuden ja helpottaa erikoisemmille päätelaitteille rakennettavien sovelluksien toteuttamisessa. (2k mediat.com 2000 ja Wikipedia Vapaa tietosanakirja c 2001)

Toisin kuin HTML, XHTML vaatii, että DTD -esittelyn on oltava tiedoston alussa. Aiemmin tätä ei tarvittu, koska uusimmat selaimet osasivat käsitellä kaiken tyyppistä HTML:ää. Juurielementin on aina oltava <html> ja siihen on lisättävä viittaus XML-nimiavaruuteen. XML tunnistaa isot ja pienet kirjaimet, joten XHTML -tagit on kirjoitettava pienin kirjaimin. Tagit tulee myös laittaa sisäkkäin oikealla tavalla ja oikeaan järjestykseen. Esimerkiksi normaalissa HTML:ssä kaksi tekstikappaletta voitaisiin muodostaa seuraavalla tavalla:

```
<p>Jotain tekstiä esimerkiksi.
<p>Toinen kappale.</p></p>
```

HTML tulkitsee edellä olevan normaaleina tekstikappaleina. XHTML puolestaan vaatii myös lopetustagit oikeisiin paikkoihin ollakseen validia koodia:

```
<p>Validia XHTML- koodia.</p>
<p>Toinen kappale.</p>
```

XHTML:n tyhjät elementit täytyy lopettaa kauttaviivan avulla. Esimerkiksi HTML:n <hr> -tagi, joka piirtää ruudulle vaakasuoran viivan, voidaan lisätä sivulle aivan sellaisenaan, mutta XHTML:ssä elementtiin on lisättävä kauttaviiva <hr />.

Lisäksi XHTML:ssä on oltava elementit <head> ja <body> sekä head -osan ensimmäisenä elementtinä tulee olla <title>. (Boumphrey ym. 2000: 25 - 33)

2.4 MySQL

MySQL on suosituin avoimeen lähdekoodiin perustuva tietokantahallintajärjestelmä, jonka on kehittänyt (ja kehittää edelleen) ruotsalainen MySQL AB. Tietokannan loivat vuonna 1995 Suomesta kotoisin olevan Michael ”Monty” Widenius sekä ruotsalainen David Axmark. Ensimmäinen virallinen versio julkaistiin vuonna 1996. Sen jälkeen varsinaisia tietokantapalvelinversioita on ollut kuusi (3.20, 3.21, 3.22, 3.23, 4.0 ja 4.1) ennen tuoreinta versiota 5.0. MySQL levitetään GNU GPL -lisenssillä tai kaupallisella lisenssillä, jos GPL ei ole sopiva. Alun perin ohjelman tarkoituksena oli käsitellä suuria taulukoita muita olemassa olevia järjestelmiä huomattavasti nopeammin. (Wikipedia Vapaa tietosanakirja b 2001)

Kirjaimet SQL tulevat englannin kielen termeistä Structured Query Language. Se on yleisin standardoitu kyselykieli, joita relaatiotietokannoissa käytetään. SQL jakautuu kahteen osaan: tietokannan rakenteen määrittelyyn (Data Definition Language [DDL]) sekä tietokannan sisällön käsittelyyn (Data Manipulation Language [DML]). SQL-kieltä on käytetty vuodesta 1986, ja siitä on rakenneltu lukuisia versioita.

Tietokanta on loogisesti yhtenäinen kokoelma tietoa, jolla on jokin merkitys. Se on suunniteltu, rakennettu ja täytetty tiedolla jotain tiettyä tarkoitusta varten. Se voi olla lähes mitä tahansa aina pienestä ostoslistasta kuvagalleriaan tai erittäin isoja määriä tietoa yrityksen verkossa. Jotta voitaisiin lisätä, päästä käsiksi tai prosessoida dataa, koneella täytyy olla asennettuna hallinnointijärjestelmä, esim. MySQL server. Koska tietokoneet osaavat nykyään käsitellä hyvin suuria määriä tietoa, tietokannat ovat hyvin olennainen osa tietokoneen käyttöä, joko yksittäisinä ohjelmina tai jonkun muun sovelluksen yhteydessä. (Lahtonen 2002: 2)

”MySQL on relaatiotietokantojen hallintajärjestelmä, RDBMS (relational database management system). MySQL varastoi kaikki tietokannat, taulut, sarakkeet ja rivit (sekä niiden sisältämän datan), ja lisäksi se käsittelee niitä yhtenä kokonaisuutena”. (Meloni 2003: 11)

Tieto tallennetaan useampiin tauluihin yhden massiivisen taulun sijaan. Tämä nopeuttaa ja tuo järjestelmään lisää joustavuutta. MySQL:n voit ladata verkosta ilmaiseksi osoitteesta www.mysql.com.

MySQL on erittäin nopea, luotettava ja helppokäyttöinen. Se toimii useilla eri alustoilla (cross-platform). Ei liene siis ihme, että MySQL:n suosio on lisääntynyt huimasti muutamien viime vuosien aikana, ja sitä on käytetty vaativissa kehitysympäristöissä menestyksekkäästi jo useita vuosia. Kolmen viikon aikana uusimman version (MySQL 5.0) lataukset ylittivät jo miljoonan rajan. MySQL:n logon on suunnitellut suomalainen mainostoimisto.

3 Kippis.fi-sivusto

Kippis.fi-sivusto on syytä käydä lävitse ennen siirtymistä rekisteröintipalvelun kuvaamiseen. Näin saadaan kuva siitä, minkälaisessa ympäristössä palvelu toimii ja miten sivusto on vaikuttanut palvelun toteuttamiseen ulkonäöllisesti ja rakenteellisesti. Tämän osion aikana käsitellään graafiset, rakenteelliset ja toiminnalliset osiot.

3.1 Grafiikat

Sivuston graafinen osuus on toteutettu pääosin valokuvilla, joita on käsitelty Adoben Photoshop-ohjelmalla. (http://www.adobe.com). Grafiikan työstäminen aloitettiin valokuvaamalla jokaisessa Pirkanmaan Osuuskaupan ravintolassa. Näin saatiin kerättyä paljon työstettävää materiaalia, jota sivustolla käytettiin. Sivuston ulkoasu, väriteemat ja kaikki graafisuuteen liittyvät elementit omasivat suuren painoarvon. Näyttävyys houkuttelee asiakkaita ja sitouttaa heidät tehokkaammin. Näitä asioita mietittiin moneen kertaan palavereissa, kunnes saimme lopullisen rakenteen ja värimaailmat yms. selville.

Yläpalkki-kuvia (kuvat 1a, 1b, 1c ja 1d) teimme useita erilaisia, joista lopulliseen käyttöön valitsimme neljä kappaletta, yhden jokaista teemaa kohden. Vaihtuvat yläpalkkikuvat olivat suosiossa vielä joitakin vuosia sitten, mutta käytettävyyden kannalta ne tuovat nykyään enää harmeja.

Yläpalkki-kuvat tehtiin rajaamalla valokuvista mielenkiintoisia yksityiskohtia. Rajauksen koko määriteltiin 700 x 150 pikseliin (sivuston pohjana toimivan taulukon leveys). Värejä ei liiemmin muokattu. Kuvan päälle teimme suorakulmion, josta leikkasimme osan pois ja lopuksi pienensimme peittävyyttä 58 %:iin. Hieman normaalia monimutkaisempi ja osaksi läpinäkyvä palkki toimii hyvänä pohjana tekstille. Lisäsimme palkkiin vielä KIPPIS.FI-tekstin ja tehostimme sitä heittovarjolla. Kyseiset toimenpiteet ovat kuvankäsittelyn perusjuttuja, mutta yksinkertaisia ja toimivia. Samaa palkkia ja tekstiä käytimme kaikissa yläpalkki-kuvissa. Käytetyt kuvat on esitelty kuvissa 1a - 1d.

Kuva 1a Alkoholittomien juomien yläpalkki

Kuva 1b Breezereiden ja siidereiden yläpalkki

Kuva 1c Oluiden yläpalkki

Kuva 1d Viinien yläpalkki

3.2 Palkit ja Logot

Sivustolle tehtiin eri osioita kuvaamaan otsikko palkit Photoshopilla. Palkkien muoto on sama kuin yläpalkki-kuvan läpinäkyvässä osassa. Näin sivuun saatiin yhdenmukaisuutta ja kokonaisuus on tyyliltään sama. Palkkien pohjalla on kahden eri punaisen sävyn liukuväri, joka tuo palkkeihin elävyyttä. Palkkien otsikkotekstit käyttävät samaa fonttia kuin KIPPIS.FI-teksti (fontin nimi). Tekstiin lisättiin jälleen heittovarjo, joka parantaa tekstin luettavuutta ja elävöittää kuvaa.

Jokaiseen palkkiin tehtiin myös kutakin otsikkoa hyvin kuvaava logo. Logot on esitelty alla olevassa kuvassa 2.

Kuva 2 Otsikkopalkkien logot

Uutiset-logosta tehtiin sanomalehden sivun / paperiarkin näköinen, joka sisältää kuvan ja tekstiä. Uutisille mielestämme se ainut ja oikea logo.

Kirjaudu-logosta tehtiin hieman raollaan olevan ovi. Avoimesta ovesta on helppo mennä sisään, aivan kuin sivustollemmekin.

Uudet tuotteet-logon on tarkoitus herättää huomiota. Sarjakuvamainen logo on hauska ja jää mieleen.

Ravintolat saivat osakseen rakennuksen, joissa ravintolat yleensä fyysisesti sijaitsevat. Aika luonnollinen valinta ravintoloiden logoksi.

Polttopisteessä sai omakseen suurennuslasin, jonka merkityksen ymmärtää jokainen. Tämän tarkoituksena on ottaa tietokannan uusin tuote lähempään tarkasteluun.

Kilpailuun teimme pixel art -periaateella arpakuution. Viivat ovat symmetriset toisiinsa nähden ja kokonaisuudessaan arpakuutio on erittäin sopiva tarkoitukseensa.

Rekisteröityminen sai tytön ja pojan päät. Rekisteröityvät ovat luonnollisia henkilöitä, joten päät ovat erittäin kuvaava logo. Lisäksi kumpaakaan sukupuolta väheksymättä logoon sijoitettiin molempien sukupuolien edustajat.

Gallup on kysely, jossa tiedustellaan, mitä mieltä ihminen kulloinkin esillä kysymyksestä on. Niinpä ei kuvaavampaa logoa kuin kysymysmerkki, voisi olla.

Taustat

Taustat päätettiin tehdä kolmiulotteisiksi. Taustalla oli ajatus pöydästä, jota katsotaan suoraan ylhäältä käsin. Olut-teeman tausta oli mahdollista tehdä täysin Photoshopilla, mutta muissa teemoissa jouduimme turvautumaan CINEMA 4D ohjelman (http://www.maxon.net) tarjoamiin mahdollisuuksiin.

Olut-teema toteutettiin kokonaan Photoshopilla. Punertavalle pohjalle puupinnan aikaansaamiseksi käytimme Motion Blur -suodinta (kulma 90 astetta ja etäisyys suureksi). Peittävyyttä vähennettiin sen verran, että pöytä näytti luonnolliselta. Pöydälle lisättiin vielä valaistus Lightning Effects -suotimella. Valon tulosuunta on vasemmalta ylhäältä, jonka mukaan teimme taustan loput elementit.

Pöydälle tehtiin seuraavaksi kaksi erimuotoista lasinalusta. Väreiksi valitsimme neutraaleja sävyjä, etteivät ne erottuisi taustasta liikaa. Lasinalusille teimme vielä erikseen varjot, jotka noudattavat valon tulosuuntaa. Näin kuvasta tuli realistinen. Pöydän pinta näytti vielä turhan pelkistetyltä, joten teimme pöydälle vielä kuriositeetiksi tahran. Tahran muoto saatiin kastamalla pullon pohja vedessä ja pyöräyttämällä sitä pöydällä (tällä kertaa aivan oikealla pöydällä). Kun pöydän pintaan jäi haluttu kuvio, otimme siitä valokuvan. Kuva siirrettiin Photoshop:iin ja muoto tallennettiin valintana. Muoto täytettiin 50% harmaalla ja peittävyys laskettiin 25%:iin. Tahrasta ja koko taustasta tuli mielestämme realistinen ja juuri sellainen mitä halusimmekin.

Loput teemoista olikin sitten toteutettava CINEMA 4D:llä. Mallinnuksen vaativat pullonkorkit, viinilasi, kahvikuppi sekä lusikka. Tekniikat vaihtelivat mallien mukaan. Erilaiset spline -käyrät, Lathe NURBS, Hyper NURBS sekä tekstuurit tulivat tutuiksi. Mallinnettujen objektien taustalle tehtiin erilaisia ”pöytiä”. Varsinaiseksi pöytälevyksi laitettiin tummaa lankkua alkoholittomat ja viinit teemoihin, jonka päälle lisättiin erilaiset ”pöytäliinat” kutakin teemaa varten. Breezer teema sai nuorekkaan, aavistuksen heijastavan rosterimaisen pinnan johon laitettiin mallinnetut pullonkorkit.

Taustoista tuli lämminhenkiset ja kodikkaan oloiset. Toivottavasti sivustolla kävijöistä tuntuu samalta.

Taustatapetti

Koko sivuston taustalle tehtiin 1800-luvun loppupuolen tapetti. Hillitty kukkakuviointi neutraalin ruskealla pohjalla antaa tilaa sisällölle ja on samanhenkinen.

S- Etukortti

S- Etukortin pohjana käytimme valmista kuvaa, johon lisättiin teippi. Teipin tarkoituksena oli kiinnittää kortti suoraan tapettiin. Tämä oli yksinkertainen, mutta toimiva ratkaisu.

Post -it

Post-it -lappu tehtiin myös itse. Kellertävälle pohjalle tehtiin tummemmat rajat sekä liukuvärillä realistisempi ulkoasu. Lisäksi oikea alakulma taitettiin realistisuuden lisäämiseksi. Formaatiksi valittiin läpinäkyvyyttä tukeva gif, jolloin muistilapun reunat eivät näy häiritsevästi taustalla. Muistilappuun vaihdettava teksti toimii datamanagement -puolelta. Tekstin tulisi näkyä handwriting -fontilla ja luoda käsinkirjoitettu efekti (aivan kuin oikeassa muistilapussa).

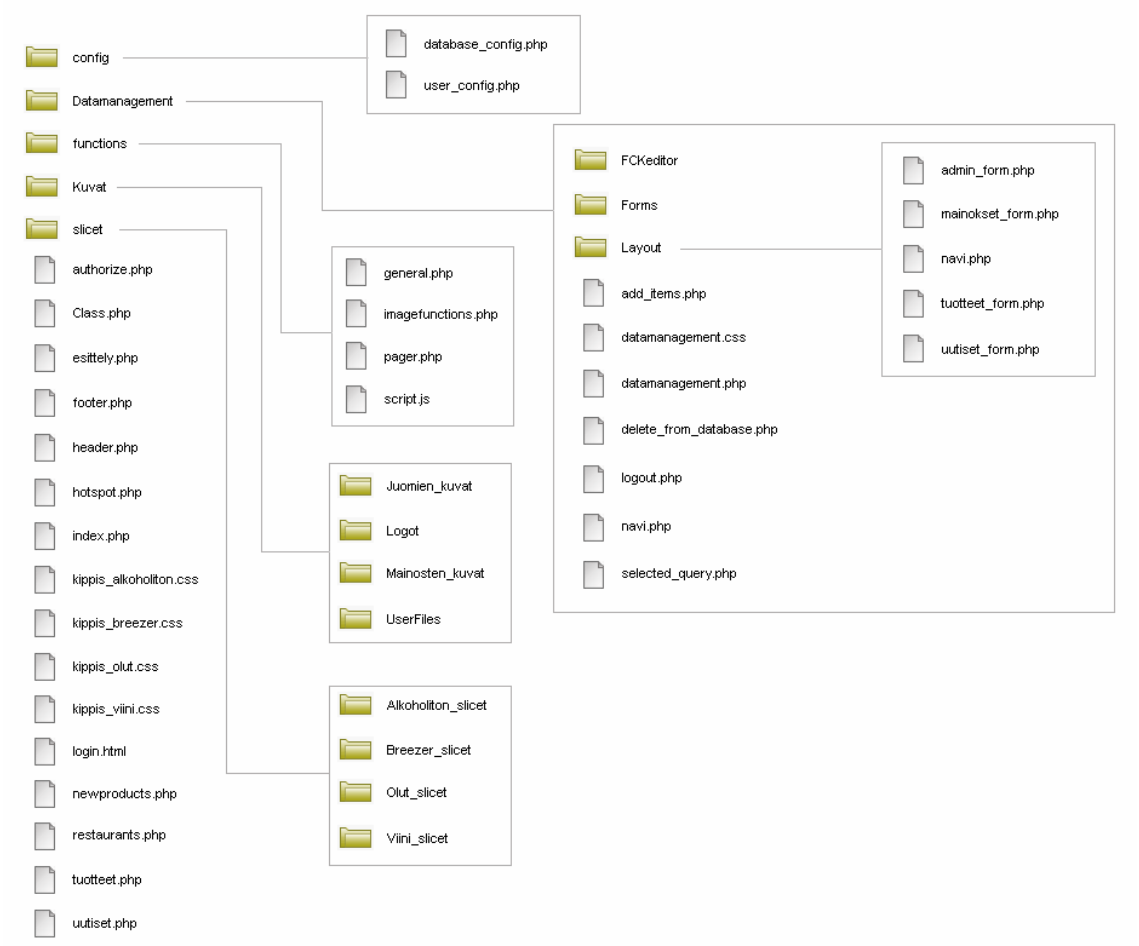
Taulujen kehykset

Taulujen kehykset editoitiin taulusta otetun valokuvan pohjalta. Kehyksiin lisättiin tummanruskea keskiosa, jotta niihin tulisi lisää selkeyttä ja ne näyttäisivät pienentämisen jälkeen vielä kehyksiltä, eivätkä suttuisilta reunoilta.

3.3 Tiedostorakenne

Suurien web-projektien läpivienti ja sivustojen toimivuus vaativat myös sivuston rakenteelta selkeää jäsentelyä ja rakennetta. Kippis.fi -sivuston monet toiminnallisuudet asettivat omat rajoitteensa niiden sijoittelulle kokonaisrakenteessa (kuva 3). Varsinkin sivujen hallinnointiin liittyvät toiminnot tuli sijoittaa omaksi kokonaisuudekseen, jotta ne olisivat turvattuna ulkopuolisilta tunkeutumisilta. Muissa toiminnallisuudet sijoitettiin rakenteeseen niin, että ne palvelisivat parhaalla mahdollisella tavalla sivuston toimintaa.

Sivusto toimii ns. moduulitasolla, eli erilliset toiminnallisuudet lisätään erillisinä palikoina tarpeen mukaan. Pääsivuina toimivat index.php, uutiset.php, esittely.php ja tuotteet.php. Muut toiminnallisuudet on lisätty php:n include -funktion avulla, jolla voidaan upottaa erillisiä tietoja ja kokonaisuuksia, muista tiedostoista. Tällä menettelyllä saadaan nopeasti sekä joustavasti rakennettuja ja helposti tiettyyn käyttötarkoitukseen muokattavia sivustoja.



Kuva 3 Kippis.fi-sivuston tiedostorakenne

3.4 Juurirakenne

Pää- eli ns. root -taso tiedostorakenteessa sisältää kaikki peruskäyttäjälle tapahtuvat ja näkyvät toiminnot ja niihin liittyvät tiedostot. Myös grafiikkaan ja sivuston ulkonäköön sekä sivuston hallinnointiin liittyvät tiedostot ja kansiot löytyvät tältä tiedosto tasolta.

- authorize.php → Tiedostossa tarkistetaan salasanan ja käyttäjätunnuksen löytyminen tietokannasta ja oikeudesta siirtyä hallinnointi alueelle. Myös käyttäjätaso määrätään kyseisessä tiedostossa.

- Class.php → Oliopohjaisten toimintojen sijaintipaikka. Sisältää tietokannan käsittelyyn ja sivujen sivutukseen (pagination) liittyvät luokat.

- esittely.php → Tulostaa tuotteen id:n perusteella kaikkine tietoineen (kuva 4).

- footer.php → Sisältää copyright-palkin ja muistilapun. Tulostaa lisäksi kannassa olevia mainoksia ja ravintoloiden logoja vierailijan nähtäväksi (kuva 4).

- header.php → Navigaatio tiedosto, jossa ovat myös css-tiedosto linkitykset (kuva 4).

Kuva 4 Sivuston modulaarinen perusrakenne

- hotspot.php → Polttopisteessä ikkunan tuotetietojen tulostukseen tarvittavat toiminnot (kuva 4).

- index.php → Aloitussivu, joka sisältää polttopisteessä, uutiset, uudet tuotteet, ravintolat, kirjautumis, gallup ja kilpailu -ikkunat.

- css-tiedostot → Sivuston muotoiluun liittyvät tiedostot, jotka ovat kippis_alkoholiton.css, kippis_breezer.css, kippis_olut.css ja kippis_viini.css. Tiedostot ovat muuten identtisiä, mutta niiden kuvamaailma vaihtelee. Nämä tiedostot mahdollistavat teemojen vaihdon JavaScriptin avulla.

- login.html → Kirjautumis ikkuna (kuva 4).

- newproducts.php → Uusien tuotteiden ikkuna. Tulostaa viisi uusinta tuotetta, jotka on valittu kannasta (kuva 4).

- restaurants.php → Ravintolat ikkuna. Tulostaa kaikkien Kippis.fi -sivustoon kuuluvien ravintoloiden nimet ja linkit niiden kotisivuille (kuva 4).

- tuotteet.php → Tulostaa ja lajittelee tuotteet navigaation valintojen mukaan (kuva 4).

- uutiset.php → Tulostaa yksittäiset ja kaikki uutiset (kuva 4).

Config -kansio

Sivuston tietokanta- sekä datamanagement -puolen käyttäjäasetuksiin liittyvät tiedostot, joita ovat seuraavat tiedostot:

- database_config.php → Tarpeelliset asetukset otettaessa yhteyttä tietokantaan. Tiedoston asetukset vaikuttavat koko sivustoon.

- user_config.php → Vaikuttaa sivuston hallinnoinnin asetuksiin. Määrää mitä tietoja esitetään eri käyttäjäoikeuksia omaaville henkilöille.

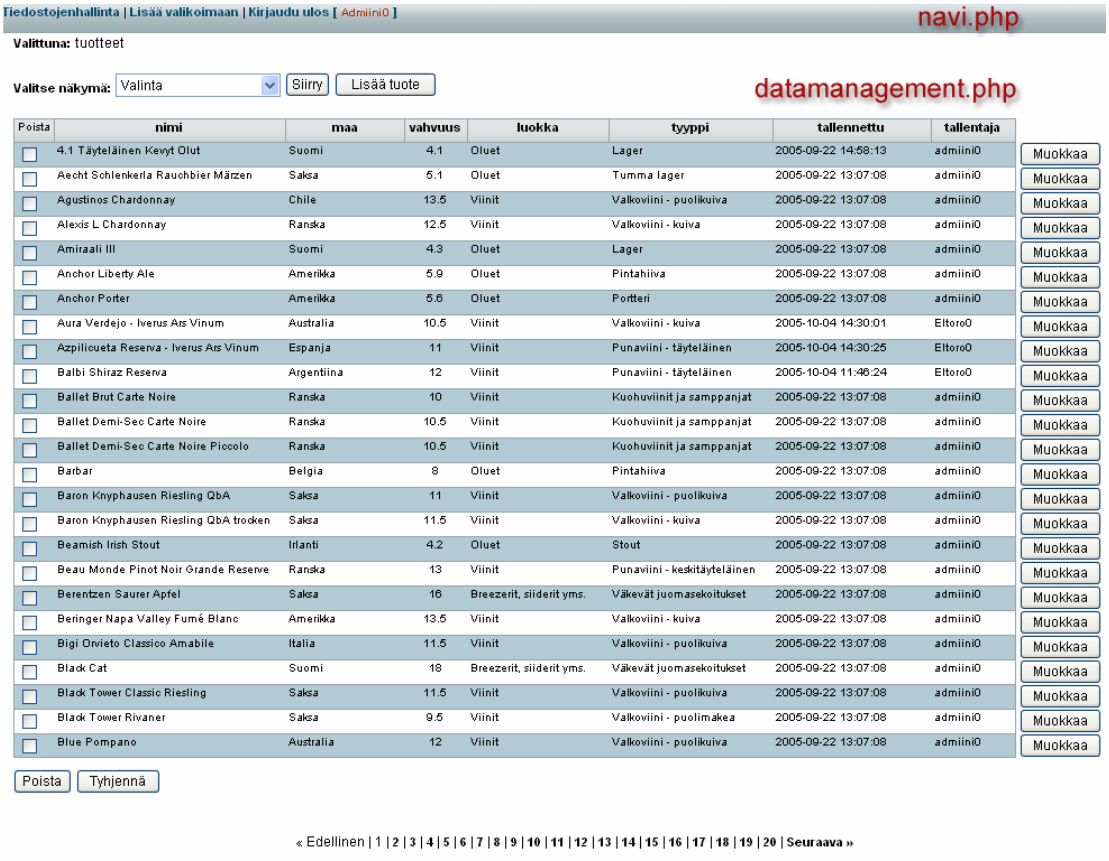
Datamanagement-kansio

Kansio sisältää tiedostot ja kansiot sivuston hallinnointiin. Pääsy kansioon on suojattu käyttäjätunnuksen ja salasanan avulla.

- datamanagement.css → Hallinnointiosion ulkonäön muotoilu.

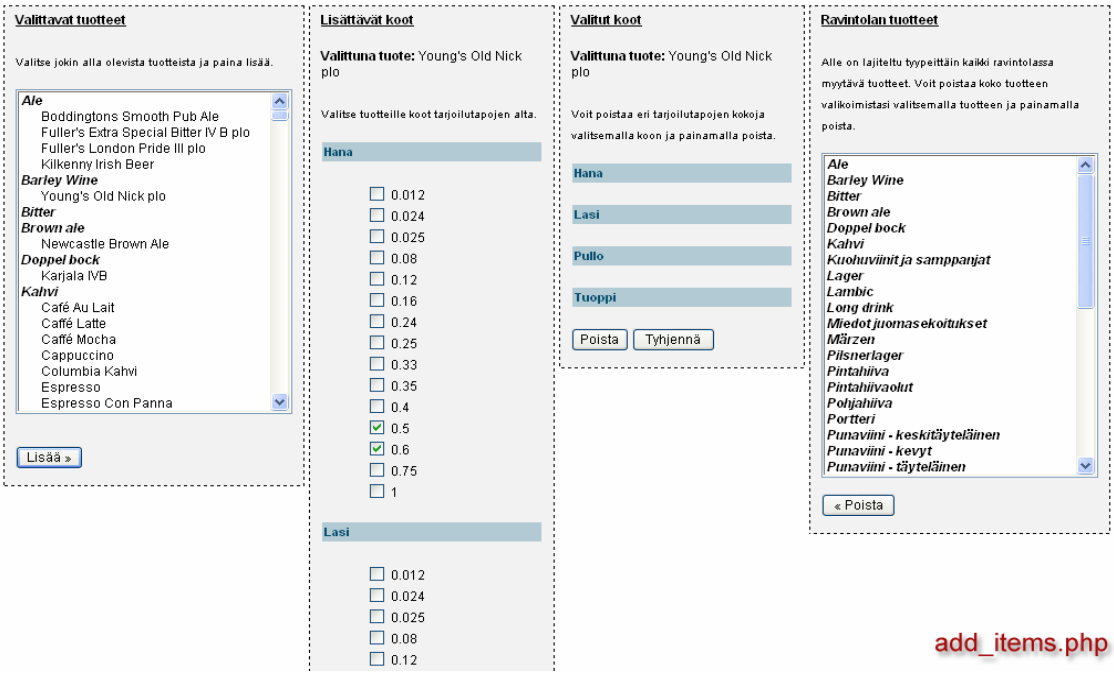
- datamanagement.php → Varsinaiseen tiedostonhallintaan liittyvät toiminnot. Tulostaa taulukkomuodossa halutut tiedot (esim. tuotteet) ja mahdollistaa niiden poistamisen sekä lajittelun haluamalla tavalla **(**kuva 5).

- add_items.php → Mahdollistaa tuotteiden lisäämisen ja poistamisen ravintoloiden valikoimasta (kuva 6**)**.



Kuva 5 Datamanagement.php ja tuotteiden listaus

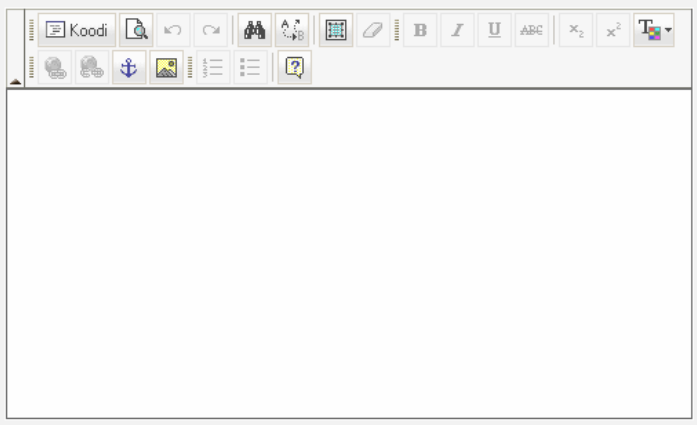
- delete_from_database.php → Sisältää tietokannasta poistamiseen liittyvät toiminnot. Liitetty include -komennolla datamanagement.php:en.
- logout.php → Kirjaa käyttäjän ulos.
- navi.php → Navigaatio tiedosto. Liitetty include -komennolla datamanagement.php:en.
- selected_query.php → Käsittelee käyttäjän valintoihin liittyvää tietoa(pitää muistissa edellisen valinnan ja luo uuden). Liitetty include -komennolla datamanagement.php:en.



Kuva 6 Ravintoloiden tuotteiden valitsemiseen ja poistamiseen käytettävä lomake

FCKeditor -kansio

FCKeditor on Open Source html-editori, joka mahdollistaa nopean ja vaivattoman tavan luoda html-koodia normaalista tekstistä (**kuva 7**).

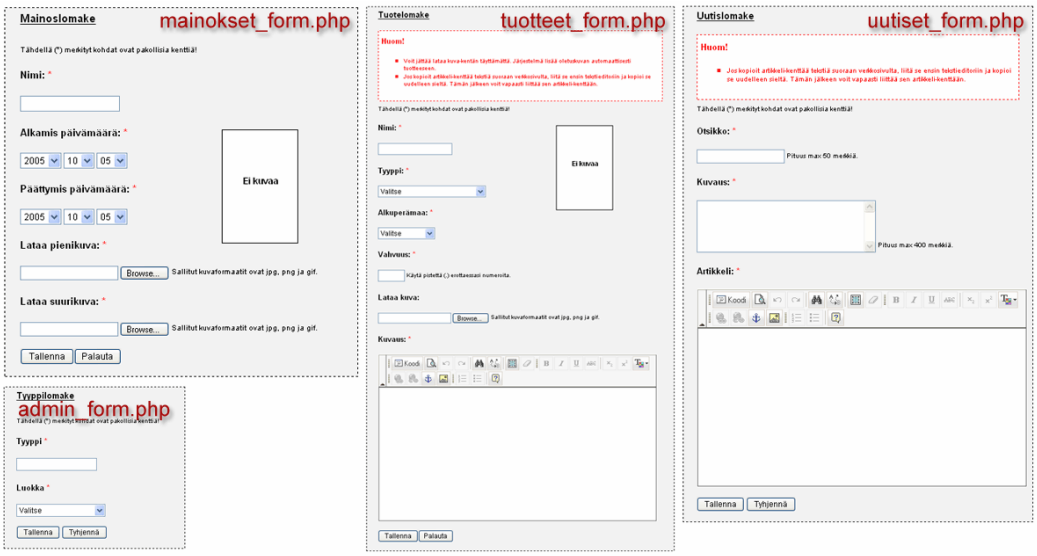


Kuva 7 FCKeditor

Forms -kansio

Kansio on Sivuston hallintaan käytettävien lomakkeiden sijaintipaikka. Sisältää tuotteiden, uutisten jne. lisäämiseen sekä muokkaamiseen tarvittavat tiedostot.

- admin_form.php → Pääkäyttäjän lomake, jolla hallinnoidaan mm. ravintoloiden käyttäjätunnuksia ja salasanoja, juoma tyyppejä, kokoja jne. (kuva 8).
- mainokset_form.php → Mainoksien lisäämiseen ja muokkaamiseen liittyvä lomake (kuva 8).
- navi.php → Navigaatiotiedosto.
- tuotteet_form.php → Tuotteiden lisäämiseen ja muokkaamiseen liittyvä lomake (kuva 8).
- uutiset_form.php → Uutisten lisäämiseen ja muokkaamiseen liittyvä lomake (kuva 8).



Kuva 8 Tietojen hallinnoinnin lomakkeita

Layout -kansio

Kansio sisältää Datamanagement -osion graafiset elementit, kuten ylänavigaation graafisen gradient -palkin.

Functions -kansio

Useasti käytettävät toiminnot sijaitsevat tässä kansiossa. Toiminnot on luotu funktioiksi,

jotta niitä voitaisiin käyttää nopeasti ja helposti.

- general.php → Sisältää päivien oikeellisen laskemisen kalenteri vuoden ja kuukauden avulla, kuvan nimen toistuvuuden tarkastamisen ja kuvakentän arvon tarkastamisen (onko kuvakenttä tyhjä).

- imagefunctions.php → Kuvakoon muuttaminen ja oikean kuvasuhteen säilytäminen toteutetaan tämän tiedoston funktioilla.

- pager.php → Sivutuksen (pagination) arvojen siirtäminen olioista muuttujiin.

- script.js → Javascript funktiot, jotka huolehtivat teemojen vaihtamisesta eri css-tiedostojen välillä.

Kuvat -kansio

Kansio sisältää kuvat, jotka eivät sisälly oleellisesti Kippis.fi:n rakenteeseen. Yleensä tämä tarkoittaa halinnointiosiosta tallennettuja kuvia, kuten logoja tai tuotteiden kuvia.

Juomien_kuvat-kansio
Tuotteiden kuvat tallennetaan tänne. Sisältää myös Thumbs-kansion, johon tallennetaan pienkuvat.

Logot-kansio
Kansio on ravintoloiden logojen tallennus kansio.

Mainosten_kuvat-kansio
Tähän kansioon tallennetaan mainosten kuvat. Sisältää myös Thumbs-kansion, johon tallennetaan pienkuvat.

UserFiles-kansio
FCKeditorin luoma kansio, johon tallennetaan kaikki editorissa koneelta ladatut tiedostot. UserFiles-kansio sisältää file-, flash-, image- ja mediakansiot.

Slicet-kansio
Sisältää kuvat, jotka liittyvät Kippis.fi-sivuston rakenteeseen.

Alkoholiton_slicet-kansio
Kippis_alkoholiton.css:än tarvitsemat graafiset elementit.

Breezer_slicet-kansio
Kippis_breezer.css:än tarvitsemat graafiset elementit.

Olut_slicet-kansio
Kippis_olut.css:än tarvitsemat graafiset elementit.

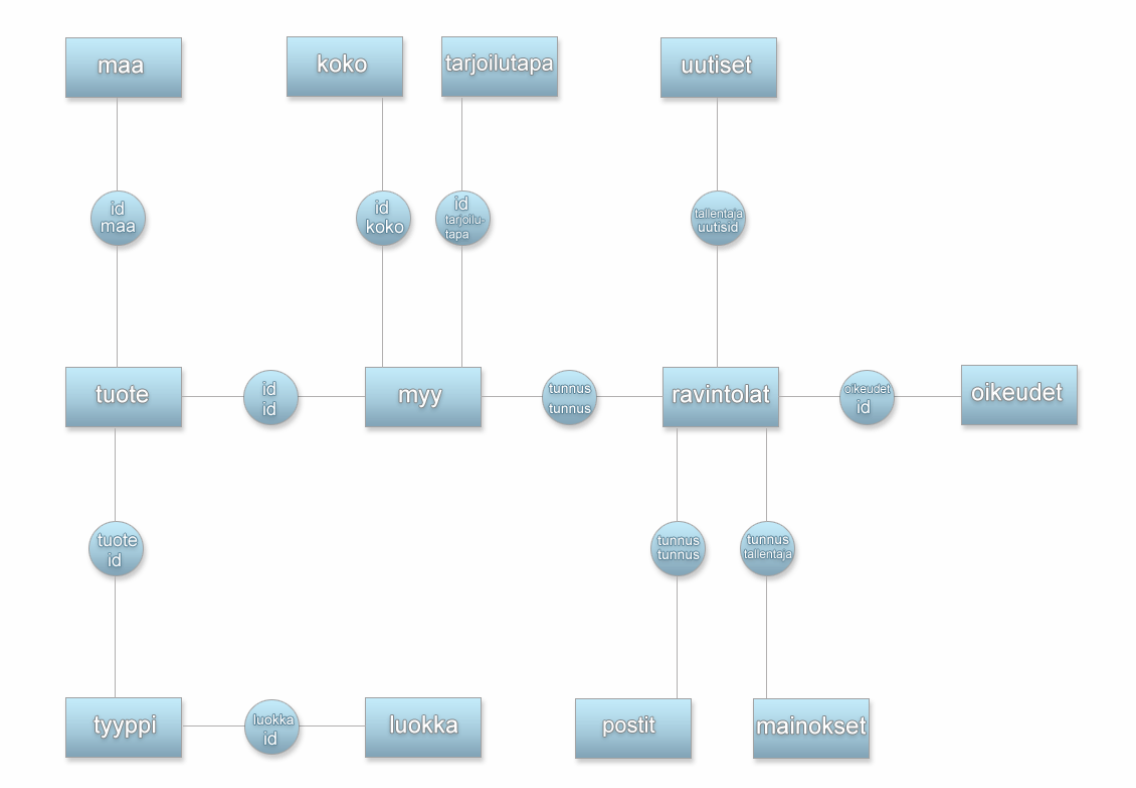
Viini_slicet-kansio
Kippis_viini.css:än tarvitsemat graafiset elementit.

3.5 Tietokannan rakenne

Suunniteltaessa Kippis.fi:tä tuli heti selväksi tietokannan laaja-alainen käyttö sekä sen tarpeellisuus. Tietokannan rakenteessa lähdettiin liikkeelle hahmotelmasta, jossa tieto sijaitsee vain yhdessä paikassa ja näin päästäisiin toistuvista arvoista eroon kolmannen normalisointimuodon mukaisesti. Tämä on johtanut tilanteeseen, jossa tieto on ns. hajautettu useampiin tauluihin. Taulujen tietoihin pääsee käsiksi ns. liitoksilla, joita voidaan kuvata kahden erillisen taulun sarakkeiden yhdistämisellä. Kummassakin sarakkeessa tulee olla yhtenevät arvot. Alla pieni esimerkki liitoksesta, joka löytyy lauseen WHERE -osiosta.

```
SELECT t.nimi, m.maa FROM tuote AS t, maa AS m WHERE t.maa=m.id
```

Tietokannan graafisessa kuvaajassa (kuva 9) näkyvät taulujen suhteet toisiinsa liitoksessa yhdistetyt sarakkeet niiden välillä.



Kuva 9 Tietokannan rakenne liitoksineen

tuote -taulu

Tuotetaulu sisältää kaiken tarpeellisen informaation sivustolla olevista (juoma)tuotteista. Lisäksi se sisältää liitokset tauluihin tyyppi, maa ja myy. Taulu sisältää alla olevat sarakkeet:

- id → Tuotteen identifioiva tunnus
- nimi → Tuotteen nimi
- tyyppi → Liitos tyyppitauluun
- maa → Liitos maatauluun
- vahvuus → Tuotteen alkoholiprosentti
- alkoholiton → Onko tuote alkoholiton vai ei
- luonnehdinta → Tuotteen kuvaus
- kuvanimi → Tuotteen kuvan nimi
- kuvatyyppi → Kuvan tyyppi (esim. jpg)
- isokuva → Ison kuvan hakemistopolku
- thumbnailkuva → Pienen kuvan hakemistopolku
- tallentaja → Tuotteen tallentaneen ravintolan tunnus
- tallennettu → Tallennus aika

maa -taulu

Sisältää maiden nimet sekä id:n liitokseen tuotteen kanssa.

- id → Liitos tuotetaulun kanssa
- maa → Maan nimi

tyyppi -taulu

Taulu on juomatyyppien tallennuspaikka (esim. Siideri). Taulu sisältää myös liitokset tuotetauluun sekä luokkatauluun.

- id → Liitos tuotetaulun kanssa
- tyyppi → Tyypin nimi
- luokka → Liitos luokkataulun kanssa

luokka -taulu

Taulu on juomaluokkien tallennuspaikka (esim. Alkoholittomat). Liitos tyyppitauluun.

- id → Liitos tyyppitaulun kanssa
- luokka → Luokan nimi

myy -taulu

Tauluun tallennetaan ravintoloiden myymät tuotteet sekä niiden tarjoilutavat (esim. pullo) ja koot. Tässä taulussa tietyissä sarakkeissa olevat arvot voivat olla samoja, kuten ravintola tunnus. Taulun suhde on ns. monen suhde moneen, eli monilla ravintoloilla voi olla monia tuotteita ja eri kokoja ja tarjoilutapoja näistä. Lisäksi taulussa on liitokset tuote-, ravintolat-, koko- ja tarjoilutapatauluihin.

- myyid → Yksittäisen ravintolassa myytävän tuotteen identifioiva tunnus
- id → Liitos tuotetaulun kanssa
- tunnus → Liitos ravintolat taulun kanssa
- tarjoilutapa → Liitos tarjoilutapataulun kanssa
- koko → Liitos kokotaulun kanssa

koko -taulu

Juomien kokojen tallennuspaikka. Liitos myytaulun kanssa.

- id → Liitos myytaulun kanssa
- koko → Koon nimi

tarjoilutapa -taulu

Tarjoilutapojen tallennuspaikka (esim. pullo). Liitos myytaulun kanssa.

- id → Liitos myytaulun kanssa
- tarjoilutapa → Tarjoilutavan nimi

ravintolat -taulu

Ravintoloihin liittyvän tiedon tallennustaulu. Liitokset myy-, oikeudet-, uutiset-, postit- ja mainokset taulun kanssa.

- tunnus → Käyttäjätunnus
- nimi → Ravintolan nimi
- osoite → Ravintolan www-osoite
- salasana
- oikeudet → Käyttäjätaso (esim. admin)
- logonimi → Ravintolan logon nimi
- logotyyppi → Ravintolan logon tyyppi (esim. jpg)
- logohakemisto → Ravintolan logon sijaintipaikka tiedostojärjestelmässä
- logonmuoto → Onko logo vertikaalisen vai horisontaalisen muotoinen

uutiset taulu

Ravintoloiden julkaisemien uutisten tallennustaulu. Liitos ravintolat taulun kanssa.

- uutisid → Uutisen identifioiva tunnus
- otsikko → Uutisen otsikko
- kuvaus → Lyhyt kuvaus julkaistavasta uutisesta
- artikkeli → Uutinen kokonaisuudessaan
- tallennettu → Tallennus päivämäärä
- tallentaja → Liitos ravintolat taulun kanssa

oikeudet -taulu

Ravintoloiden käyttöoikeuksien tallennuspaikka. Liitos ravintolat taulun kanssa.

- id → Liitos ravintolat taulun kanssa
- taso → Tason nimi

mainokset -taulu

Ravintoloiden tallennettujen mainoksien tallennuspaikka. Liitos ravintolat taulun kanssa.

- id → Mainoksen identifioiva tunnus
- tunnus → Liitos ravintolat taulun kanssa
- nimi → Mainoksen nimi
- kuvaiso → Ison kuvan nimi
- kuvapieni → Thumbnail kuvan nimi
- tyyppi → Thumbnail kuvan tyyppi (esim. jpg)
- hakemisto → Ison kuvan hakemistopolku
- thumbnail → Thumbnail kuvan hakemistopolku
- tallennettu → Tallennus päivämäärä
- alkamispvm → Mainoksen alkamis päivämäärä
- paattymispvm → Mainoksen päättymis päivämäärä

3.6 Sivuston toiminta ja sen dynaamisuus

Kippis.fi -sivuston suuresta koosta, monipuolisesta toiminnallisuudesta ja melko tiheästä päivitystiheydestä johtuen normaali web-sivun teknologia ja sen suomat toiminnot eivät olleet riittäviä kyseisen sivuston luomiseen. Ratkaisuksi löytyi dynaaminen julkaiseminen.

Sivusto noudattaa dynaamisen julkaisemisen periaatteita, joka voidaan jaotella tietokantaan (MySQL), väliohjelmistoon (PHP), ja käyttöliittymän (XHTML). Tietokannan tarkoituksena on säilöä sisältöä, jota väliohjelmiston (PHP) avulla tallennetaan lisää tietokantaan tai tulostetaan sieltä ulos. Väliohjelmistolla tietokannasta haettu sisältö on mahdollista saattaa rakenteelliseen muotoon tulostamalla se käyttöliittymään (XHTML). Sivuston sisältö muotoillaan käyttämällä tyylejä sekä tyylitiedostoja (CSS). (Veen 2002: 221)

Sisällön hallinta

POK toivoi dynaamista sivustoa, jonka sisältö muuttuisi ravintolapäälliköiden niin halutessa. Tällöin suunnittelun lähtökohdaksi tuli luoda järjestelmä, jonka avulla päälliköt voisivat helposti päivittää ja lisätä sisältöään sivustolle. Pulman ratkaisemiseksi päädyttiin rakentamaan ravintolapäälliköille oma käyttöliittymä, joka olisi suojattuna käyttäjätunnuksen ja salasanan takana.

Päälliköiden käyttöliittymä mahdollistaa helpon sisällön hallinnan ilman minkäänlaisia ohjelmointitaitoja. Käyttöliittymän toiminnot ovat jaettu tietyille käyttäjä tasoille, jotka ovat pääkäyttäjä eli admin ja normaali käyttäjä. Alla on lajiteltu kunkin käyttäjä ryhmän mahdollisuudet vaikuttaa sisältöön kuten lisätä, poistaa tai muokata.

ADMIN:

- uutiset
- kilpailut
- tuotteet
- ravintolat
- postaukset (muistilapun teksti)
- mainokset (taulut muistilapun alla)
- koko
- luokka (esim. olut)
- käyttäjät
- tyyppi (esim. valkoviini)
- tarjoilutapa (esim. pullo)
- valikoima

NORMAALI:

- uutiset
- tuotteet
- mainokset (taulut muistilapun alla)
- koko
- tyyppi (esim. valkoviini)
- tarjoilutapa (esim. pullo)

Kaikki sisällön muokkaamiseen ja lisäämiseen on toteutettu lomakepohjaisesti, jolloin hieman harjaantumattomampikin oppii käyttöliittymän käytön hetkessä. Tilanteissa, joissa sisältöä on haluttu muotoilla html-muotoon, kuten uutisten tapauksessa, on turvauduttu avoimenlähdekoodiin perustuvaan FCKeditor:iin, jonka voi ladata ilmaiseksi osoitteesta http://www.fckeditor.net. Kyseinen sovellus on upotettu lomakkeeseen, jossa sitä tarvitaan.

4 Rekisteröintijärjestelmä

Tarkasteltaessa laajasti web-sivuja ja web-maailmaa kulttuurillisessa mielessä, voidaan huomata yhteisöllisyyden kuuluvan tähän hyvin voimakkaasti. Erilaiset web-kommuunit ja keskusteluryhmät ovat lisääntyneet yhdistäen samanhenkisiä ihmisiä keskustelemaan ja tekemään yhdessä tärkeiksi pitämiään asioita. Tämän voi todeta runsaasta sivustojen määrästä, jotka sisältävät jonkinlaista yhteisöllisyyteen liittyvää, kuten keskusteluryhmiä ja palveluita, jotka ovat rajattu ns. kanta-asiakkaille. Web-maailmassa tämä lisäarvon tuominen kanta-asiakkaille tarkoittaa useimmiten sivuille kirjautumista, jolloin käyttäjä luovuttaa sivuston omistajille tarvittavan tietomäärän itsestään vastineeksi sivuston antamasta lisäarvosta. Käyttäjän hyötyessä palveluista sivuston omistajalla on mahdollisuus sitouttaa käyttäjä sivustoille, toisin sanoen edesauttaa tilannetta, jolloin hän palaisi uudelleen. Tällaista tilannetta voitaisiin kuvata symbioottiseksi tilaksi, jolloin molemmat osapuolet hyötyvät omalla tavallaan toistensa toiminnasta.

Jo alkutaipaleelta lähtien Kippis.fi -sivuston tarkoituksena oli tarjota tuotetiedon sekä uutisten lisäksi erityisiä palveluita sivustolle kirjautuneille ihmisille. Nämä palvelut sisältäisivät erinäisiä kilpailuja, kyselyitä, bilekutsujen lähettämisen sähköpostin välityksellä ja paljon muuta. Jotta kaikki nämä käyttäjään kohdistuvat toiminnallisuudet voitaisiin toteuttaa, tarvittaisiin mutkattomasti toimiva ja turvallinen rekisteröitymisjärjestelmä, joka olisi myös käytettävyydeltään hyvää tasoa.

4.1 Lomakkeiden sijoittelu ja muiden sivustojen rekisteröintipolitiikka

Lomakkeiden sijoittelu on toteutettu pitkälti samalla tavalla kuin muillakin sivustoilla. Kirjaudu -moduuli ja muut keskeiset elementit (uutiset, kilpailu jne.) ovat näkyvillä paikoilla. Viimeisenä lisättiin rekisteröitymismahdollisuus. Aikaisemmin Kirjaudu -moduuli oli sijoitettu sivun sisältösolun oikeaan alakulmaan (vain ravintolapäälliköiden oli tarvetta kirjautua sisään). Rekisteröintielementtien ollessa valmiina, Kirjaudu -moduuli siirrettiin sisältösolun oikeaan yläkulmaan, jossa näkyvyys on paras mahdollinen. Ravintolapäälliköitä varten tehtiin oma kirjautumislinkki footer -palkkiin, josta hallintatilaan kirjautuminen avautuu uuteen selainikkunaan.

Uutiset oli luonnollista sijoittaa ylös vasemmalle. Suomalainen on pienestä asti opetettu lukemaan vasemmalta oikealle ja ylhäältä alas. Tällä tavoin emme sekoita kävijää, vaan tuntuma sivustoon ja sen elementteihin säilyvät neutraaleina. Ainoastaan Kilpailulaatikko voidaan laittaa ennen uutisia, koska se ”kuumana aiheena” ajaa uutisten ohi. Myöhemmin kilpailut tullaan siirtämään kokonaan rekisteröityneille käyttäjille, joten uutiset tulevat säilymään omalla paikallaan.

Tutkittaessa muiden sivustojen rekisteröitymismahdollisuuksia, huomasimme niiden noudattelevan hyvin samantyyppisiä sijoitteluperiaatteita.

Plussan sivuilla (http://www.plussa.com) kirjautuminen ja rekisteröinti on sijoitettu vasempaan reunaan. Rekisteröinti avautuu uuteen ikkunaan ja tietojen täyttäminen näyttäisi olevan kolmivaiheinen.

Tietokone.fi (http://www.tietokone.fi) sivuston rekisteröinti on toteutettu puolestaan oikeaan reunaan. Tunnuksen ja salasanan alapuolella on Rekisteröidy -linkki, joka avaa salatun yhteyden yhteystietolomakkeeseen.

Veikkaus.fi (http://www.veikkaus.fi) on toteuttanut oman sivustonsa kehys-tekniikalla ja rekisteröityminen on sijoitettu yläkehykseen. Palkki on helppo havaita ja sisään kirjautuminen onnistuu luonnollisesti samasta kohdasta. Varsinainen rekisteröitymislomake avautuu salatulla yhteydellä uuteen selainikkunaan. Rekisteröinti on kuusivaiheinen ja täytettäviä kenttiä on erittäin monta. Tämän tosin ymmärtää, koska Veikkaus tarvitsee asiakkaastaan luotettavat tiedot (mm. pankkitilin numeron jne.)

4.2 Lakiasiat

Suomen laki vaikuttaa tehtäessä rekisteröitymispalveluita sekä järjestettäessä kilpailuja. Sivustoa tehtäessä on hyvin vahvana osa-alueena laki tekijänoikeudesta sekä esim. Sosiaali- ja terveydenhuollon tuotevalvontakeskuksen (http://www.sttv.fi) säännökset arpajaisista sekä kilpailuista. Sivusto on tehty tekijänoikeutta kunnioittaen, joten kaikki sivustolla esiintyvä materiaali on itse tuotettua eikä tekijänoikeutta ole näin ollen loukattu. Näin saimme myös tekijänoikeudet itsellemme ja toivottavasti väärinkäytöksiä ei esiinny. Lakia on sovellettu kulloinkin tarvittavaan tilanteeseen, ja esim. rekisteröinnin yhteydessä on kirjoitettu ehdot ja säännöt, joihin sitoutumalla käyttäjällä on oikeus rekisteröityä sivustolle. Teksti sisältää myös oleelliset vastuusta vapauttamisesta, korvausvelvollisuuden raukeamisesta jne.

Alla oleva teksti rekisteröintisivulta Kippis.fi -palvelusta:

> *”Kippis.fi -verkkosivujen käyttäjä sitoutuu noudattamaan jäljempänä esitettyjä ehtoja. Mikäli käyttäjä ei hyväksy näitä ehtoja kokonaisuudessaan, ei käyttäjällä ole oikeutta rekisteröityä.*
>
> *Kippis.fi:n sisältö esitetään sellaisenaan. Pirkanmaan Osuuskauppa ei vastaa sivujen sisällöstä, oikeellisuudesta, täsmällisyydestä tai luotettavuudesta, eikä tietojen käytön välittömästi tai välillisesti aiheuttamista vahingoista. Pirkanmaan Osuuskauppa varaa oikeuden ilman erillistä ilmoitusta koska tahansa muuttaa sivuja tai estää pääsyn niille.*
>
> *Käyttäjä on korvausvelvollinen Pirkanmaan Osuuskaupalle ja suoraan muille tahoille mahdollisista väärinkäytöksistä sekä lain- ja sopimuksenvaraisista toimistaan.*
>
> *Pirkanmaan Osuuskaupan sivuilla on mahdollisesti linkkejä ulkopuolisten ylläpitämille www-sivuille. Pirkanmaan Osuuskauppa ei vastaa näiden sivujen sisällöstä, eikä ko. sivuilla kerättyjen henkilötietojen rekisteröinnin lainmukaisuudesta.” (Kippis.fi 2005)*

Tekstin alapuolelle on liitetty vielä kaksi valintaruutua, joista ensimmäisessä hyväksytään säännöt ja ehdot (pakollinen rekisteröitymisen edellyttämiseksi) ja toisessa valitaan, haluaako käyttäjä vastaanottaa Pirkanmaan Osuuskaupalta ilmoituksia tulevista kilpailuista ja tapahtumista sähköpostitse.

Mielestämme tämä on toteutettu reilun pelin hengessä. Yhdenkään käyttäjän ei ole pakko vastaanottaa sähköpostia, vaan sitä on tiedusteltu reilusti etukäteen. Näin luodaan rehellinen kuva käyttäjälle, joka edesauttaa positiivisen yleiskuvan muodostamisessa.

4.3 Järjestelmän vaatimukset

Rekisteröitymispalvelu voidaan käsittää useiden erillisten toiminnallisuuksien yhdistämisenä ja niiden toimimista yhdessä, luoden yhtenäisen kokonaisuuden. Karkeasti ottaen nämä erilliset kokonaisuudet voidaan karkeasti jakaa käyttäjän antamiin syötteisiin (esim. nimi) ja ohjelmakoodin suorittamiin operaatioihin ja tulostuksiin. Nämä kokonaisuudet voidaan määrittää seuraavanlaisilla kysymyksillä:

1. Kuinka informoidaan käyttäjää rekisteröintipalvelusta?
2. Miten saadaan käyttäjältä vaaditut tiedot?
3. Kuinka suhtaudutaan käyttäjien tekemiin virheisiin ja vahingollisiin syötteisiin?
4. Kuinka varmistutaan, että käyttäjän tiedot ovat ainakin osittain oikeelliset, ja haluaako hän oikeasti sitoutua sivuille?
5. Miten ja mihin tallennetaan käyttäjän tiedot?
6. Kuinka tarjotaan käyttäjälle apua kadonneen salasanan tai käyttäjätunnuksen kadotessa?

Rekisteröintipalvelusta informoiminen

Jos palvelusta ei tiedoteta tarpeeksi tai siihen käsiksi pääseminen aiheuttaa paljon vaivaa, voidaan melkein suoralta kädeltä todeta, ettei palvelusta tule menestystä. Kaikki esitystyylistä sijaintiin vaikuttavat oleellisesti palvelun näkyvyyteen.

Käyttäjätietojen kerääminen

Lomakkeen käyttöä voidaan pitää ensisijaisena tapana vuorovaikuttaa palvelimen sekä käyttäjän välillä ja tätä tapaa käytetään käyttäjän tiedon keräämiseen rekisteröitymistä varten. Tarkoituksena on luoda lomake, joka on selkeä, käyttäjää opastava ja looginen.

Tyhmä ja vaarallinen käyttäjä

Voidaan olettaa, että peruskäyttäjä ei tiedä tai osaa toimia, vaikka ohjeistusta löytyisikin suoraan nenän edestä. Tällöin tulevat tarpeelliseksi erilaiset järjestelmät, jotka valvovat käyttäjän syötettä tarkistaen, onko kenttiin syötetty tieto oikeellista ja onko pakollisia kenttiä jätetty tyhjäksi. Käyttäjää tulee myös ohjeistaa syöttöprosessin aikana, jotta saavutettaisiin virheettömin mahdollinen tulos.

Käyttäjä voi olla myös vaarallinen aiheuttaen joka tietämättömyyttään tai tahallisesti vahinkoa järjestelmälle. Näin ollen syötetty tieto tulisi saattaa vaarattomaan muotoon ennen lopullista käsittelyä.

Sitoutumisen tarkistaminen

On hyvin yleistä, että käyttäjä antaa virheellistä tietoa ollakseen sitoutumatta sivustoon tai vaikka osallistuakseen nopeasti vain johonkin kilpailuun. Tällaisten tilanteiden vält-

tämiseksi tulee rakentaa järjestelmä, joka vaatii käyttäjän interaktiivisuutta tämän aktivoidessa oman tilinsä, ja että ainakin osa annetuista tiedoista on totta. Monessa tilanteessa tämä tarkoittaa sähköpostivarmistusta, jolloin käyttäjälle lähetetään sähköpostiviesti, jonka avulla hänen pitää aktivoida tilinsä.

Tallennustavan valinta

Oleellinen osa rekisteröitymistä on käyttäjän tietojen tallentaminen. Ilman tätä mahdollisuutta palvelu on merkityksetön. Palvelun koon ja käyttötarkoituksen mukaan tulisi määritellä sopivin tallennusmedia. Tallennus vaihtoehtoina ovat mm. tietokanta ja tiedosto niin teksti, kuin xml-pohjaisena.

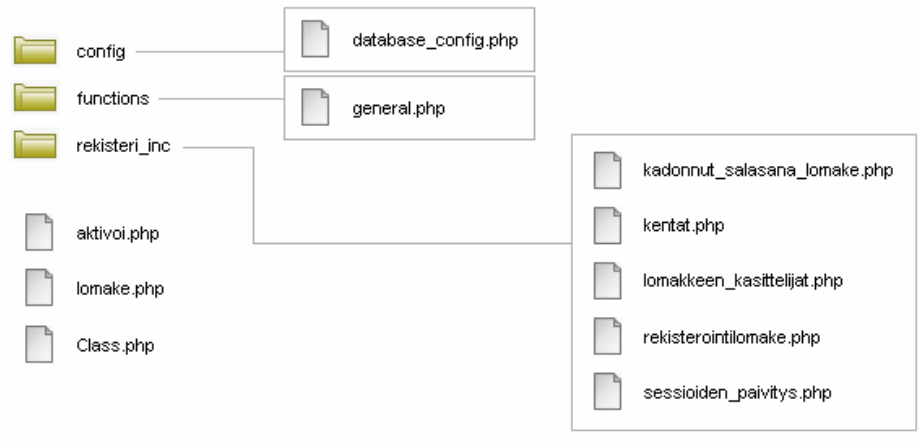
Käyttäjätietojen hallinta jatkossa

Käyttäjämäärän kasvaessa ja ajan kuluessa tulee varmasti eteen tilanteita, jossa käyttäjä on kadottanut tunnuksensa tai salasanansa tai haluaa muuten muuttaa tietojaan. On tärkeätä luoda käyttäjälle mahdollisuus muokata tietojaan ja auttaa häntä saamaan kadonnut salasanansa takaisin.

4.4 Rekisteröintipalvelun rakenne

Tiedostorakenne

Rekisteröintipalvelu on pääsääntöisesti toteutettu käyttäen ns. modulaarista rakennetta (toiminnot sijaitsevat erillisissä tiedostoissa) (kuva 10). Kyseiseen ratkaisuun on vaikuttanut toimintojen helppo uudelleen käyttö sekä rakenteellinen yksinkertaistaminen. Näin on päästy tilanteeseen, jossa tietyt toiminnot ovat tarkkaan rajattu omaan tiedostoonsa.



Kuva 10 Rekisteröintipalvelun tiedostorakenne

Kokonaisuudessaan palvelu koostuu kahdenlaisista tiedostoista, jotka ovat ns. moduulitiedostoja eli uudelleen käytettäviä sekä yhden kerran käytettäviä rekisteröitymistä kohden. Rekisteröintipalvelun moduulitiedostot sijaitsevat kansiossa rekisteri_inc. Se sisältää seuraavat tiedostot:

- kadonnut_salasana_lomake.php → Kadonneiden salasanojen lähettämiseen tarkoitettu lomake.
- kentat.php → Lomakkeiden kenttien arvojen tallennus muuttujat. Käytetään tilanteessa, jossa lomakkeen tietojen syötössä on tapahtunut virhe ja jo täytetyt virheettömät kentät tulostetaan valmiiksi lomakkeille.
- lomakkeen_kasittelijat.php → Lomakkeiden kenttien tarkistukseen vaadittavat toiminnot. Tulostaa myös virheilmoitukset.
- rekisterointilomake.php → Rekisteröintiin liittyvä lomake. Käytetään myös käyttäjän muokatessa tietojaan.
- sessioiden_paivitys.php → Käyttäjän ollessa kirjautuneena ja hänen muokatessaan tietojaan tulee käyttäjän sessio-muuttujien arvot päivittää uusiin arvoihin, koska niitä käytetään tallentamaan käyttäjien tietoja hänen ollessaan kirjautuneena.

Palvelu sisältää myös muita moduuleita, jotka eivät suoranaisesti liity rekisteröintiin, mutta ovat välttämättömiä sen toimivuuden kannalta. Niihin kuuluvat seuraavat tiedostot:

- config/database_config.php → Tietokannan käytön välttämättömät tiedot. Sisältää tietokannan käyttäjätunnuksen, salasanan, palvelinosoitteen ja tietokannan nimen.
- functions/general.php → Sisältää satunnais salasanan luomiseen tarvittavan funktion.
- Class.php → Tietokannan sekä sähköpostin lähettämiseen liittyvät luokat löytyvät tästä tiedostosta.
- login.php → Sisältää kirjautumiseen tarvittavan lomakkeen sekä linkit käyttäjäprofiilin muokkaukseen ja uuden salasanan saamiseen.
- header.php → Sisältää ulos- ja sisäänkirjautumiseen vaadittavat toiminnot.

Moduulien ulkopuolisia tiedostoja ovat seuraavat:

- aktivoi.php → Tiedosto aktivoi käyttäjän tilin, jonka jälkeen hän on valmis kirjautumaan sisään.
- lomake.php → Sitoo moduulit yhteen. Tiedoston toimintona on tulostaa lomakkeet, tarkistaa käyttäjän syötteet sekä antaa virheilmoitukset. Tallentaa myös käyttäjän tiedot sekä lähettää hänelle aktivoimis sähköpostiviestit sekä uuden salasanan.

Tietokantarakenne

Rekisteröintipalvelun toteuttaminen ei vaadi suurta tai monimutkaista tietokantaa toimiakseen vaan sen toteuttamiseen riittää yksi taulu. Tähän yhden taulun suunnitelmaan on vaikuttanut se tosiasia, että toistuvia eli samoja tietoja ei juuri esiinny, vaan ne ovat suu-

rilta osin uniikkeja. Näin päästään tilanteeseen, jossa tietokannalta ei vaadita lukuisia tauluja liitoksineen. Rekisteröityneiden henkilöiden käyttäjätiedot tallennetaan kayttajat nimiseen tauluun (kuva 11). Taulu sisältää seuraavat kentät:

- kayttaja_id → Tallentuu automaattisesti käyttäen auto increment-toimintoa käyttäjän rekisteröityessä. Tietotyyppinä on INT, joka mahdollistaa kokonaislukujen käytön.
- kayttaja_tunnus → Käyttäjän rekisteröityessä valitsema oma tunnus, joka mahdollistaa kirjautumisen sisään. Käyttää VARCHAR-tietotyyppiä, joka on rajoitettuna kymmeneen merkkiin. Näin ei ylitetä käyttäjätunnukselle määrättyä enimmäispituutta.
- nimi → käyttäjän nimi, johon on varattu VARCHAR-tyypillä enimmillään 255 merkkiä.
- svuosi → Syntymävuosi käyttää SMALLINT-tietotyypppiä kokonaislukujen tallentamiseen ja se on rajoitettu neljään merkkiin.
- postinumero → Postinumeron tallennus kokonaisluku muotoon.
- toimipaikka → Postitoimipaikan tallennus vaihtuvaan merkki eli VARCHAR-tyyppiin, joka on rajoitettu 255 merkkiin.
- puhelinnumero → Puhelinnumero tallennetaan vaihtuvaan merkki tyyppiin, joka rajoitettu 20 merkkiin. Kyseisessä kohdassa on käytetty merkki tyyppiä, koska halutaan antaa mahdollisuus käyttää erilaisia merkintätapoja kuten väliviivaa numeroiden välissä.
- email → Sähköpostiosoitteen tallennus vaihtuvaan merkki tyyppiin.
- salasana → Salasanan tallennus vaihtuvaan merkki tyyppiin, jolla enimmäispituus 255 merkkiä.
- skortti → S-kortin olemassa olon tallennus. Käytetään ENUM-tyyppiä, jolla voidaan määrittää soluun mahdollisesti tallennettavien tietojen vaihtoehdot. Kyseiseen soluun saa sijoittaa vain 'Ei' tai 'On'.
- tiedotteet → Sama tilanne kuin skortissa, mutta voidaan sijoittaa vain '0' (ei saa lähettää) tai '1' (saa lähettää).

kayttaja_id	INT
kaytta_tunnus	VARCHAR(10)
nimi	VARCHAR(255)
svuosi	SMALLINT(4)
osoite	VARCHAR(255)
postinumero	INT
toimipaikka	VARCHAR(255)
puhelinnumero	VARCHAR(20)
email	VARCHAR(255)
salasana	VARCHAR(255)
skortti	ENUM('Ei', 'On')
tiedotteet	ENUM('0', '1')
aktivoitu	ENUM('0', '1')
rekisteroitumis_pvm	DATETIME
kirjautunut_viimeksi	DATETIME

Kuva 11 Käyttäjät-taulun tietotyypit

Kentän tarkoitus on tallentaa vaihtoehto, haluaako käyttäjä hänelle lähetettävän sähköpostin avustuksella viestejä tulevista tapahtumista jne.

- aktivoitu → Sama tilanne kuin tiedotteet kentässä. Tili on aktivoitu, kun tallennettuna on numero yksi.

- rekisteroitumis_pvm → Aika jolloin käyttäjä on rekisteröitynyt. Käytetään tietotyyppiä DATETIME, joka muokkaa tallennettavan ajan muotoon vuosi-kuukausi-päivä tunnit:minuutit:sekunnit.

- kirjautunut_viimeksi → Tallennetaan jokaisen kirjautumisen yhteydessä käyttäjän aika. Käytetään samaa tietotyyppiä, kuin rekisteroitymis_pvm:ssä.

Alla on esitetty esimerkki (kuva 12) mahdollisesta rivistä käyttäjän tallentamien tietojen kera. Huomion arvoista on salasana kentässä oleva kryptinen merkkijono, joka on luotu käyttämällä php -funktiota md5 rekisteröitymisen aikana. Kyseisellä funktiolla saadaan aikaiseksi salasanasta salattu merkkijono.

kayt-taja_id	kayt-taja_tunnus	nimi	svu-osi	osoite	postinumero	toimipaikka	puhelinnumero
1	Robert1	Robert Mast	1975	Kiertotie 7 b	33001	Tampere	040-111 111

email	salasana	skortti	tiedotteet	aktivoitu	rekisteroitumis_pvm	kirjautunut_viimeksi
rob@co.fi	03cdfe1acffd1195e7c5c683822cbefd	on	1	1	2005-10-19 13:31:30	2005-10-19 13:31:30

Kuva 12 Esimerkki käyttäjät-taulun tiedoista

4.5 Palvelun toiminta teoriassa

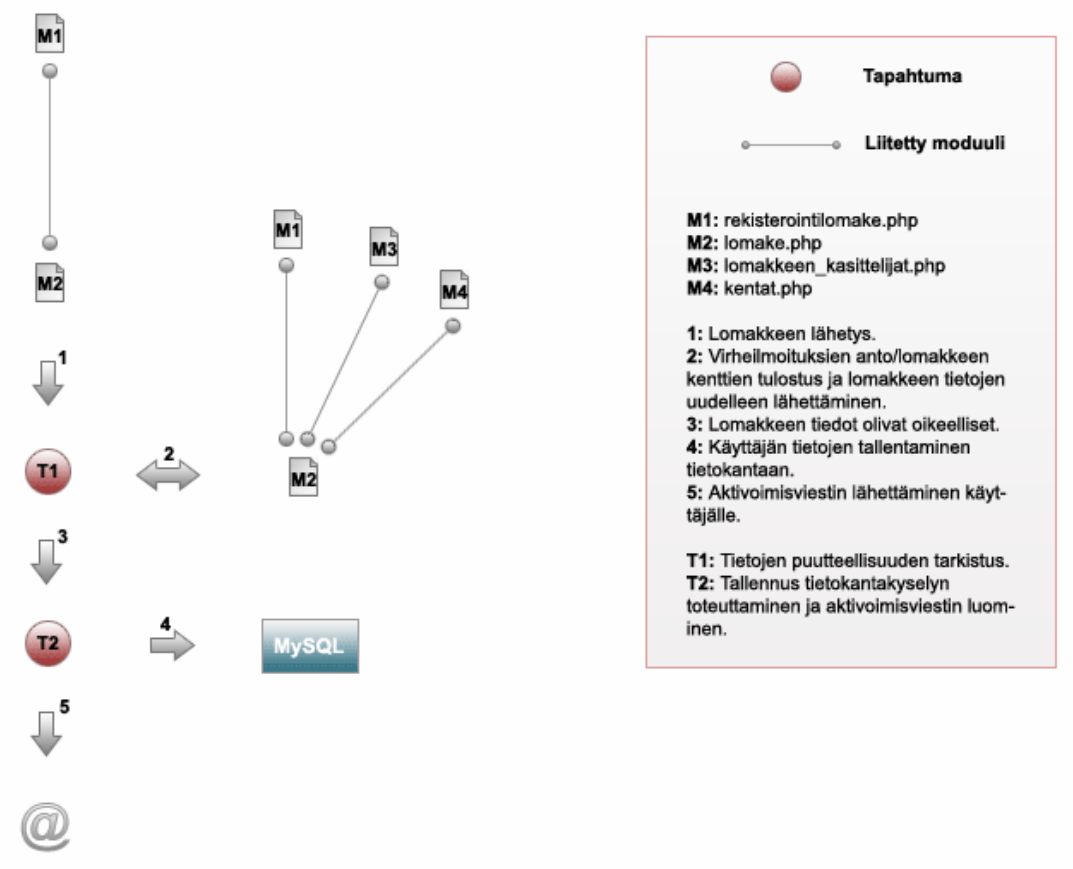
Teoriassa palvelu voidaan jakaa helpommin omaksuttaviin omiin kokonaisuuksiin, jotka esitellään seuraavissa luvuissa.

Rekisteröityminen

Lähtökohtana rekisteröitymisessä on rekisteröintilomake, jolle käyttäjä kirjaa vaaditut tiedot. Kippis.fi:n tapauksessa tunnuksen, nimen, osoitteen, postinumeron, sähköpostiosoitteen ja salasanan. Lisäksi sivuston rekisteröinnille asetetut ehdot ja säännöt tulee hyväksyä ja antaa Kippis.fi-sivustolle lupa lähettää tulevaisuudessa sähköpostia tulevista tapahtumista ja kilpailuista. Kyseinen lomake on moduuli, joka sijaitsee tiedostossa rekisteroitymislomake.php. Tämä moduuli on liitetty lomake.php-tiedostoon, joka toimii rekisteröityneen tiedon käsittelijänä.

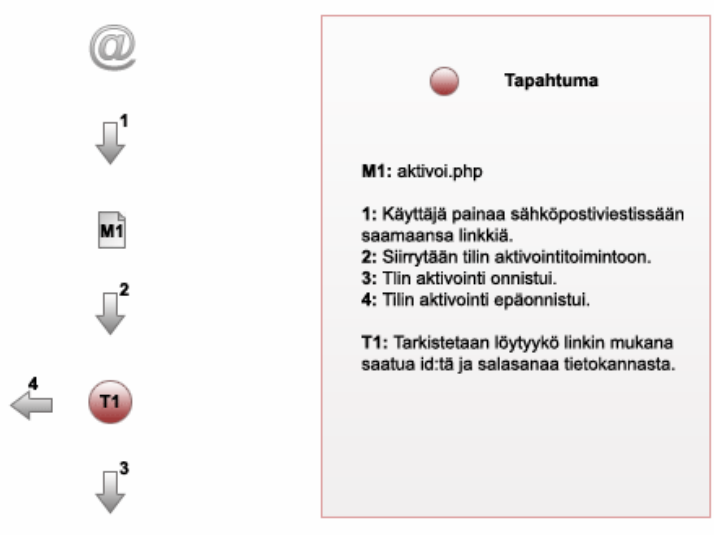
Käyttäjän lähetettyä tietonsa, käsitellään ne lomake.php-tiedostossa ns. vaarattomaan muotoon. Kun syötetty tieto on tehty vaarattomaksi, liitetään lomakkeen_kasittelijat.php-moduuli lomake.php-tiedostoon include-funktiolla tietojen tarkistusta varten. Kyseinen moduuli tarkistaa lähetettyjen kenttien tietojen oikeellisuuden sekä antaa virheilmoitukset tyhjistä sekä vääristä tiedoista. Jos käyttäjän syötteessä oli virheitä, liitetään kenttien muistina toimiva kentat.php-moduuli. Se lisää oikein täytetyt kentät säästäen käyttäjää ylimääräiseltä tiedon syöttämiseltä. Lisäksi rekisterointilomake.php liitetään uudelleen, johon jo syötetyt oikeelliset tiedot asetetaan kenttiin.

Syötteen ollessa oikein, siirrytään lomake.php-tiedostossa kohtaan, jossa käyttäjän valitsema tunnus ja sähköpostiosoite tarkistetaan tupla-arvojen poistamiseksi tietokannasta. Jos jompikumpi jo löytyi, liitetään include-funktiolla lomakkeen_kasittelijat.php, kentat.php sekä rekisterointilomake.php. Muussa tapauksessa käyttäjän syöttämät tiedot tallennetaan tietokantaan, mutta tiliä ei vielä aktivoida. Näin on saatu rekisteröitymisen ensimmäinen vaihe suoritettua (**kuva 13**).



Kuva 13 Kaavio rekisteröitymis prosessista

Tilin aktivointiosuus (**kuva 14**) alkaa lomake.php-tiedostossa sijaitsevan sähköposti toiminnon käynnistyessä. Sähköpostin lähettämiseen tarvittavat toiminnot ovat toteutettu oliopohjaisesti ja ne sijaitsevat Class.php-tiedostossa. Käyttäjälle lähetetään ilmoitus rekisteröinnin aktivoimisesta ja lisäksi se sisältää linkin, jossa on osoite, missä aktivointi tulee tapahtumaan sekä osoiterivimuuttujat, joihin kuuluvat käyttäjän id ja salasana. Myös käyttäjän salasana ja käyttäjätunnus mainitaan viestissä.

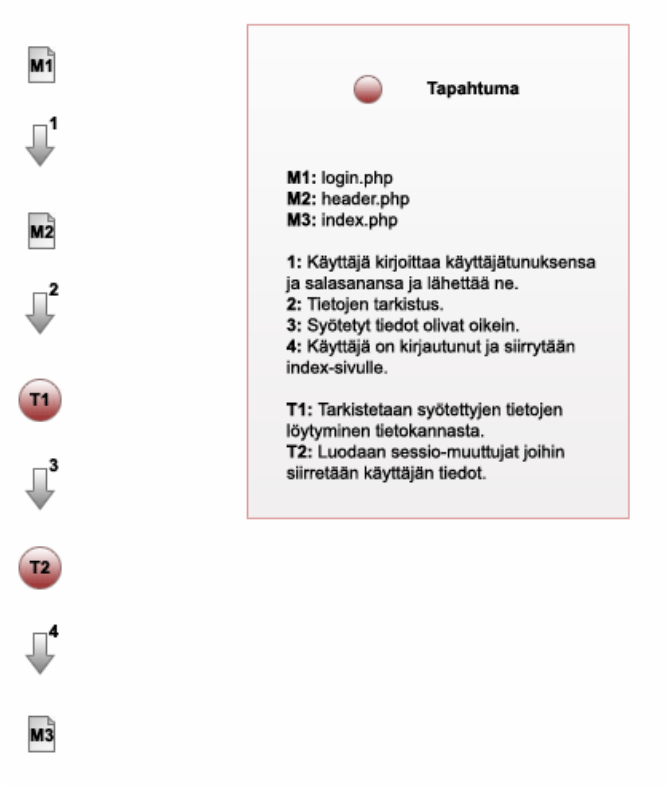


Kuva 14 Kaavio aktivoimistapahtumasta

Tilin aktivoiminen tapahtuu painamalla viestin linkkiä, jolloin käyttäjä siirtyy aktivoi.php-tiedostoon, jossa käyttäjän tili aktivoidaan hänen id:nsä ja salasanansa perusteella. Tapauksessa, jossa kyseisellä id:llä ja salasanalla olevaa tiliä ei löydy, informoidaan käyttäjää tästä. Muussa tapauksessa käyttäjä saa ilmoituksen aktivoidusta tilistä.

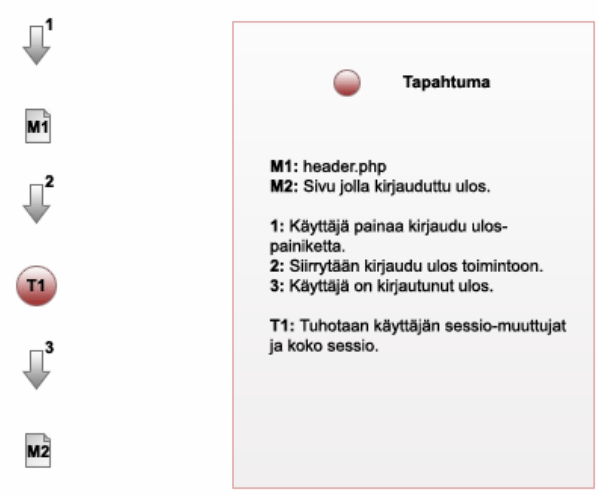
Kirjautuminen sisään sekä ulos

Kun tili on aktivoitu, käyttäjä on valmis kirjautumaan sivustolle (**kuva 15**) käyttäen login.php-moduulissa olevaa kirjautumislomaketta, joka on liitetty itse sivustoon. Tunnuksen ja salasanan lähettämisen jälkeen kirjautuminen tapahtuu header.php-moduulissa, joka sisältää myös Kippis.fi-sivuston navigaation. Käyttäjän lähettämät tunnukset tehdään ensin vaarattomiksi, jonka jälkeen tarkistetaan löytyykö tietokannasta kyseistä tiliä. Tilin löytyessä tallennetaan kirjautumisaika tietokantaan sekä siirretään kannasta haetut käyttäjätiedot sessio-muuttujiin, joissa ne säilyvät niin kauan, kuin käyttäjä on kirjautuneena. Myös sessio-muuttuja nimeltä kirjautunut luodaan, jolla käyttäjä tunnistetaan sivustolla kirjautuneeksi ja hänellä on pääsy vain kirjautuneiden alueille.



Kuva 15 Kaavio sisään kirjautumisesta

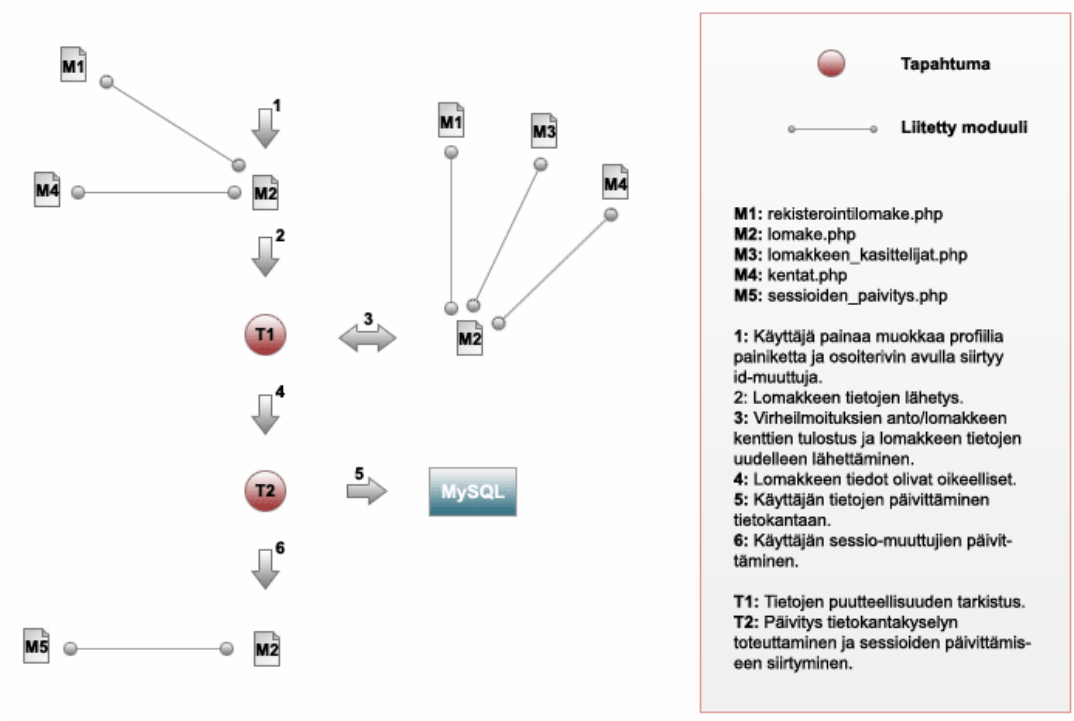
Ulos kirjautuminen (**kuva 16**) tapahtuu painamalla Kirjaudu ulos-painiketta, joka antaa osoiteriviltä käskyn header.php-moduuliin käynnistää ulos kirjautuminen. Kirjauduttaessa ulos tuhotaan jokainen sessio-muuttuja sekä tuhotaan itse sessio.



Kuva 16 Kaavio ulos kirjautumisesta

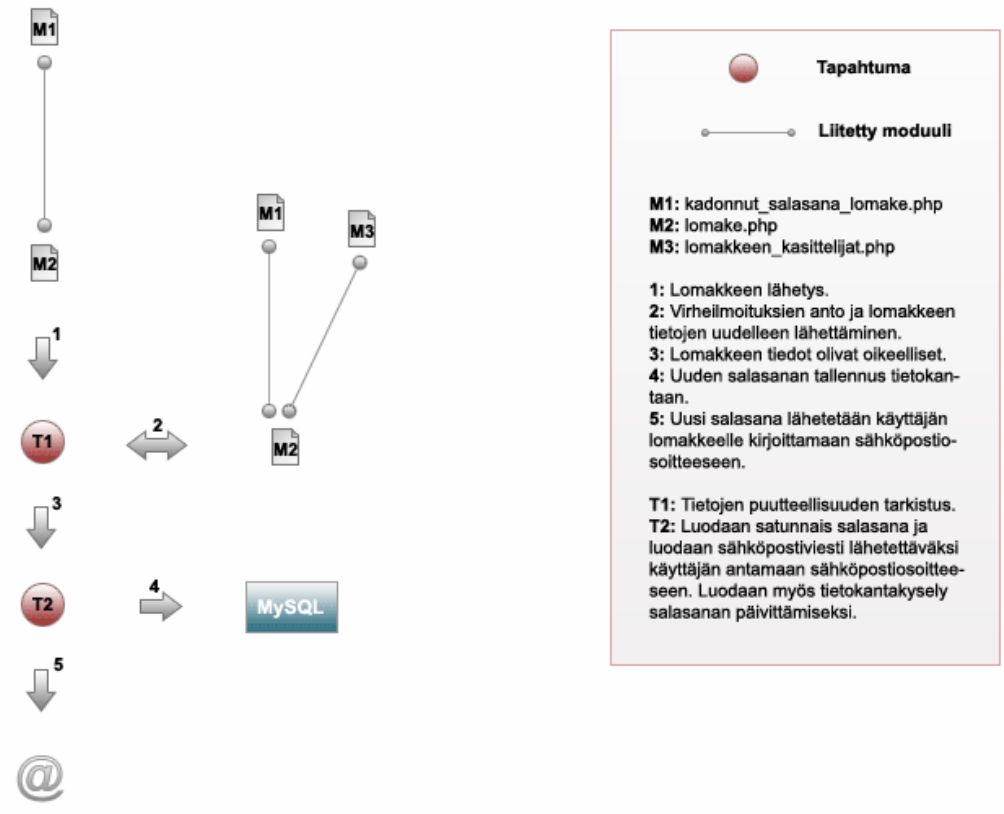
Käyttäjätietojen muokkaaminen ja kadonneet salasanat

Rekisteröintipalvelu antaa mahdollisuuden käyttäjän muokata tietojaan (**kuva 17**) vapaasti hänen ollessaan rekisteröityneenä. Tämä on mahdollista painamalla Muokkaa profiiliasi-painiketta. Tämä ohjaa käyttäjän lomake.php-sivulle jolloin siirretään selaimen osoiterivillä käyttäjän id-muuttuja. Kyseiselle sivulle liitetään rekisterointilomake.php- ja kentat.php-moduulit. Kentat.php-moduulista syötetään käyttäjän sessiomuuttujien tiedot lomakkeen kenttien muuttujiin eli käyttäjän tiedot lisätään lomakkeeseen. Muutoksien jälkeen tehdään samat tietojen oikeellisuuden tarkistukset ja vaarattomaksi tekevät toimenpiteet, kuin rekisteröinnissä. Onnistuneen profiilin päivityksen eli tietojen tallentamisen tietokantaan jälkeen tulee myös sessio-muuttujat päivittää, jotta kirjautuneen käyttäjän tiedot pysyisivät ajantasalla. Tämä tapahtuu lisäämällä sessioiden_paivitys.php-moduuli lomake.php-sivulle, joka hakee päivitetyt käyttäjätiedot tietokannasta ja siirtää ne käyttäjän sessio-muuttujiin.



Kuva 17 Kaavio profiilin muokkaamisesta

Salasana on mahdollista uusia (**kuva 18**) painamalla Unohtuiko salasana-painiketta. Tällöin siirrytään lomake.php-sivulle ja siirretään samalla osoiterivillä salasana-muuttuja, jolla identifioidaan oikea toiminto lomake.php-sivulta. Sivuun liitetään kadonnut_salasana_lomake.php-moduuli, joka sisältää lomakkeen, jossa kysytään käyttäjän sähköpostiosoitetta sekä käyttäjätunnusta. Jos käyttäjän antamat tiedot kuuluvat samaan tiliin, luodaan hänelle satunnaissalasana käyttäen functions/general.php -tiedostossa sijaitsevaa makeRandomPassword-funktiota ja lähetetään se Class.php -tiedostossa sijaitsevan sähköpostiluokan avulla käyttäjälle. Jos käyttäjän tiedot olivat virheelliset tai hänen syöttämänsä tiedot eivät kuuluneet samaan tiliin, liitetään lomakkeen_kasittelijat.php sekä kadonnut_salasana_lomake.php ja tulostetaan virheilmoitukset.



Kuva 18 Toiminta kadoneen salasanan tapauksessa

4.6 Rekisteröitymisprosessissa käytetyt tekniikat

Rekisteröitymispalvelun onnistuneeseen luomiseen vaikuttaa keskeisesti tekninen puoli ja sen osasten kitkaton toiminta. Seuraavissa luvuissa paneudutaan tarkemmin tämän teknisen puolen kuvailuun.

Ennen teknisten näkökulmien yksityiskohtaisempaa kuvausta tulee tietää projektissa käytetyistä tekniikoista enemmän, jotta voitaisiin täysin ymmärtää prosessin toimintaa.

Oliopohjaisuus ja sen käyttö prosessissa

Oliopohjainen-ohjelmointi tuli mahdolliseksi PHP-version 4 myötä. Sen käyttö ei kuitenkaan ole pakollista ja monet suuretkin sovellukset on tehty osittain tai käyttämättä ollenkaan olioita. Olioiden kätevyys perustuu koodin parempaan organisointiin, laajennettavuuteen ja uudelleenkäytettävyyteen. Nämä näkökohdat huomioiden oliot ovat tarpeellinen lisä ratkottaessa toiminnallisuuksia. (Meloni 2004: 63)

Olio voidaan yksinkertaisimmillaan kuvata teoreettisena laatikkona, joka sisältää muuttujia, toisia olioita ja funktioita. Nämä toiminnallisuudet sijaitsevat rakenteessa nimeltä class eli luokka, jonka rakenteeseen kuuluvat ominaisuudet eli muuttujat sekä metodit eli funktiot. Funktioiden avulla luokka vastaanottaa tietoa ja arvoja olioon ja tulos-

taa/syöttää tietoa ja arvoja oliosta. Luokkia voidaan myös periyttää, jolloin uudet luokat saavat (perivät) omien olemassa oleviensa toimintojen ja arvojen lisäksi toisen luokan arvoja ja ominaisuuksia. Parhaiten luokkaa ja sen toimintaa kuvaa pieni esimerkki, jossa on käytetty database luokkaa hyväksi. (Timo Tiaisen Kotimaailma)

Luokka tulee ensimmäiseksi luoda avainsanalla class, jonka perään kirjoitetaan luokan nimi. Tämän jälkeen ise luokan ominaisuudet sekä funktiot kirjoitetaan aaltosulkeiden sisään.

```
//Esitellään luokka database, jolla käsitellään tietokantoja.
class database {
	//Tänne tulee ominaisuudet ja funktiot.
}
```

Kun luokka on esitelty, tulee esitellä luokan ominaisuudet/muuttujat, joihin tallennetaan luokan ulkopuolelta sekä sisäpuolelta tulevia tietoja. Luokan ominaisuuksia luotaessa tulee käyttää var-avainsanaa niiden nimeämisessä.

```
//Luokan ominaisuuksien esittely.
var $user;
var $host;
```

Luokan sisäpuolella voi olla lukuisa määrä funktioita (metodeita), joiden avulla luokka suorittaa toimintoja. Funktion esittely ja luominen ei eroa kovinkaan paljoa normaalin funktion luonnista, mutta luokan sisällä joudutaan noudattamaan erästä sääntöä. Tämän säännön mukaan luokan sisäisiin muuttujiin ja funktioihin päästään käsiksi käyttämällä $this ja nuoli(->) -operaattoria. Esimerkiksi kuten alla on kuvattu, siirretään aiemmin luotuun $user-muuttujaan muuttujan $usr arvo käyttämällä $this-> -operaattoria.

```
//Funktion, joka avaa yhteyden tietokantaan.
function open_conn($host, $usr, $pass)
{
	//Siirretään juuri saadut arvot luokan omiksi ominaisuuksiksi käyttämällä $this-> ope-
	raattoria.
	$this->host = $host;
	$this->user = $usr;

//Kyselystä saatu boolean muuttuja olion db_link ominaisuuteen.
$this->db_link = mysql_connect($this->host, $this->user, $this->password);
```

Jotta luokkaa voidaan käyttää, tulee se sijaita samassa tiedostossa, jossa sitä tullaan käyttämään tai se pitää liittää tiedostoon, jossa sitä tullaan käyttämään, kuten alla olevassa esimerkissä on tehty. Tämän jälkeen luodaan luokan avulla olio avainsanalla new ja luokan nimi (esim. database), jota halutaan käyttää. Nyt luotu olio voi käyttää luokan muuttujia ja funktioita hyväkseen.

```
//Liitetään tiedosto, jos se sijaitsee toisessa tiedostossa.
include('Class.php');
//Luodaan olio nimeltä database.
$new_connection = new database;
//Käytetään luokan funktiota, johon sijoitetaan arvoja.
$new_connection->open_conn($host, $user, $password);
```

Rekisteröintiprosessi sisältää kaksi toimintakokonaisuutta, jotka käyttävät olioita hyväkseen ja ne ovat mime_mail- sekä database-kirjastot, jotka sijaitsevat samassa Class.php-tiedostossa. Oliopohjaisuuden käyttöön vaikuttivat osaltaan näiden toiminto-

jen toistuvuus sekä koodin saaminen lyhyempään, organisoidumpaan ja ymmärrettävämpään muotoon.

Mime_mail

Mime_mail löytyi intensiivisen etsimisen jälkeen ohjelmointiin keskittyneeltä Ohjelmointiputka sivustolta osoitteesta www.ohjelmointiputka.fi. Kyseisen funktion valintaan vaikuttivat luokan helppokäyttöisyys sekä monipuolisuus. Suurimpana vaikuttimena valintaan oli mahdollisuus lähettää liitetiedostoja, mutta tämä osa-alue ei koskettanut varsinaisesti rekisteröitymisprosessia.

Luokan käyttö on hyvin yksinkertaista. Ensimmäisenä luodaan olio, jonka jälkeen voidaan käyttää tässä tapauksessa mime_mail-luokan muuttujia ja funktioita apuna viestin lähettämisessä.

```
//Sähköpostiolion luominen.
$mail_olio = new mime_mail;
//Kenelle viesti on lähetetty.
$mail_olio -> from = "lähettäjä@fi";
//Kenelle viesti lähetetään
$mail_olio -> to = "kenelle@fi";
//Viestin aihe, joka näkyy vastaanottajan aihe/subject-kentässä.
$mail_olio -> subject = "Aiheen nimi.";
//Viestin runko.
$mail_olio -> body = "Tähän kirjoitetaan itse viesti.";
//Viestin lähetys.
$mail_olio -> send();
```

Database

Tietokantojen käsittelyyn tarkoitettu database on itse luotu luokka, joka kehitettiin jo Kippis.fi-sivustoa varten. Luokan avulla voidaan suorittaa helposti yhteydenotto tietokantaan, tietokantakyselyt sekä tietojen haku tulostettavaan muotoon.

Luokan käyttö edellyttää olion luontia, jonka jälkeen voidaan luoda yhteys haluttuun tietokantapalvelimeen. Kun yhteys on luotu, valitaan vielä haluttu tietokanta.

```
//Luodaan database-olio.
$tietokanta_olio = new database;
//Avataan yhteys haluttuun tietokanta palvelimeen.
$tietokanta_olio->open_conn('palvelimenosoite', 'käyttäjätunnus', 'salasana');
//Valitaan haluttu tietokanta.
$tietokanta_olio->select_database('tietokannan nimi');
```

Yhteyden luonnin jälkeen voidaan luoda kysely tietokannasta, joka sitten sijoitetaan funktioon, joka suorittaa kyseisen haun.

```
//Esimerkki tietokantakyselystä.
$kysely = "SELECT mitä haetaan FROM mistä taulusta haetaan";
//Tehdään kysely.
$new_connection->make_query('kysely sijoitetaan tähän');
```

Kyselyn tulos voidaan tulostaa käyttämällä funktiota fetch_query, jonka hakemat tiedot sijoitetaan muuttujaan. Todellisuudessa kyseessä ei ole vain yksi arvo, vaan funktio siir-

tää kokonaisen taulukon, josta haetaan tietoja viittaamalla solun indeksinumeroon, jossa haluttu tieto sijaitsee. Toisin sanoen jos tietokanta kyselyssä SELECT-sanan jälkeen ensimmäinen haettava tieto saa indeksiarvokseen nolla.

```
//Haetaan kyselyn tulos muuttujaan.
$tulos = $new_connection->fetch_query();
//Tulostetaan haun tulos.
print "<h1>".$tulos[0]."</h1>";
```

Muita käytettyjä tekniikoita

Foreach

Foreach-ohjausrakennetta käytetään käsiteltäessä taulukoita tai haettaessa arvoja jostakin tietystä paikasta (esim. lomakkeesta). Rakenne toimii vasta PHP:n versiosta neljä ylöspäin. Vanhemmilla versioilla foreach aiheuttaa virheen. Foreach rakennetta voidaan käyttää kahdella eri tavalla: (PHP.net 2001)

```
1.foreach(taulukko as $arvo)
2.foreach(taulukko as $avain => $arvo)
```

Ensimmäinen tapa purkaa taulukon asettaen sen hetkisen indeksin arvon muuttujaan ($arvo). Jälkimmäinen tapa käy taulukon arvot läpi ja tulostaa indeksin nimen muuttujaan $avain ja sen arvon muuttujaan $arvo.

Esimerkiksi

Käydään kaikki lomakkeelta lähetetyt arvot läpi foreach-silmukalla. $key on lomakkeen kentän nimi ja $value on sen arvo.

```
foreach($_POST as $key => $value)
```

Sijoitetaan arvot $keys-taulukkoon, jonka indeksiksi määrätään kentän nimi ja indeksin arvoksi kentän arvo.

```
$keys[$key] = $value;
```

Include

Include-komento on tehokas funktio, jonka avulla voidaan liittää tietty tiedosto PHP-koodin sekaan. Tämä puolestaan mahdollistaa modulaarisen koodaamisen, joka helpottaa koodaajaa ja säästää aikaa (vrt. CSS-tiedostojen edut). Funktioita ja osia sivustosta voidaan sijoittaa eri tiedostoihin ja käyttää niitä include-komennolla. Sivustoon voi sisällytää esim. footer.php-moduulin, johon esim. sijoitetaan linkkejä. Näin linkkejä ei tarvitse vaihtaa joka sivulle, vaan muokkaamalla ainoastaan footer.php-tiedostoa, ne päivittyvät automaattisesti. (tizag.com 2003)

Syntaksi tiedoston liittämiseksi on seuraava:

```
<? include("tiedosto.php") ?>
```

Sessiot

Sessiot tunnetaan suomenkielessä istuntoina. Istunnot tulivat mukaan PHP:n versiosta 4.0 alkaen. Niiden toiminta perustuu palvelun käyttäjälle muodostettavaan yksilölliseen id-tunnukseen (session id). Sessiotunnus tallennetaan joko evästeen (cookie) avulla tai se lähetetään URL:n mukana selaimelle. Palvelimelle muodostetaan id-tunnusta vastaava tiedosto, johon tarvittavat muuttujat tallennetaan. (Ohjelmointiputka 2002)

Session tarkoituksena on säilyttää tietoa yhden session ajan, kunnes käyttäjä lopettaa session ja sulkee selaimen. Evästeisiin verrattuna sessiot eroavat siinä, että sessiotieto säilytetään aina palvelimella, eikä sitä välitetä selaimelle.

Sessiotietoon on mahdollista päästä käsiksi sessiotunnisteiden avulla. Tästä johtuen tunnisteet on jollain tavalla välitettävä selaimelle. Sessiotunnisteet kannattaakin yleensä tallettaa evästeeseen, koska URL:n mukana siirrettävissä tunnisteissa piilee tietoturvariski. (Kollanus... b)

Istunto pitää aloittaa aina session_start()-funktiolla. Jos funktion eteen laitetaan muita lauseita, tuloksena on virheilmoitus. Esimerkki alla.

```
<?php session_start(); ?>
```

Nyt PHP on aloittanut session ja käyttäjällä on mahdollisuus tallentaa tietoa sessiomuuttujiin tai tulostaa tietoa niistä. Tietoja voi nyt käyttää lukuisin eri tavoin. Otetaan yksinkertaiseksi esimerkiksi lomake, joka kysyy käyttäjän nimeä.

Normaali html-koodi näyttää tältä:

```
<form method="post" action="sivu2.php">
Anna nimesi: <input type="text" name="nimi">
<input type="submit" value="Lähetä">
</form>
```

sivu2.php koodi puolestaan näyttää tältä:

```
<?php
session_start();
echo "<strong>Session rekisteröinti</strong><br />";

// Käyttäjän antama syöte
$nimi = $_POST['nimi'];

// Rekisteröi session avain
$_SESSION[‘nimi'] = $nimi;

// Näytä sessiotiedot:
?>

Tervetuloa <strong><?php echo $_SESSION['nimi']; ?></strong>!<br />
```

(College of San Mateo…)

Eli jos käyttäjä antaa nimekseen esim. Timo, tulostuu ruutuun:

Tervetuloa **Timo**!

Tietojen väärinkäytön mahdollisuuden vuoksi sessio on muistettava aina tuhota. Se tapahtuu joko session_unset()- tai session_destroy()-funktioilla. Molemmat funktiot tuhoavat sessiomuuttujat ja tiedot, eikä väärinkäytökselle jää enää mahdollisuutta. Esimerkki alla.

```
<?
session_start();
$_SESSION = array();
session_destroy();
?>
```

$_GET ja $_POST

GET ja POST ovat HTTP-protokollan tavat siirtää dataa HTML-lomakkeesta skriptille. GET-metodilla data siirtyy osoitteen (selaimen osoiterivin) mukana ja POST-metodilla data lähetetään erillään osoitteesta. Lähetystavasta riippumatta PHP muuttaa kenttien arvot omiksi taulukoiksi $_GET tai $_POST. Työssämme on käytetty molempia metodeja niiden hieman erilaisen luonteen vuoksi. (Sivut.web… a)

GET metodia on käytetty seuraavalla tavalla:

```
if($_GET['id'] && isset($_SESSION['kirjautunut']))
```

Yllä olevassa esimerkissä tarkastetaan osoiteriviltä id:n arvo, ja onko sessiomuuttujalla kirjautunut arvoa vai ei.

POST metodia puolestaan on käytetty mm. seuraavasti:

```
foreach($_POST as $key => $value)
```

silmukalla käydään läpi $_POST taulukon arvot ja muodostetaan niistä arvopareja.

Stripslashes

Kun html-lomakkeelta otetaan vastaan muuttujien arvoja, PHP lisää takakenoviivoja tiettyjen erikoismerkkien eteen käsittelyvaikeuksien välttämiseksi. Stripslashes()-funktiolla karsitaan merkkijonosta pois etumerkit (yleensä nuo edellä mainitut takakenoviivat) ja palautetaan merkkijono siistimmässä muodossa. Stripslashes on myös tietoturvan kannalta erittäin hyvä ratkaisu. Lomakkeen kenttiin kun voi syöttää merkkejä, jotka mahdollistavat SQL-komentojen välittämisen suoraan tietokantaan. Vihamieliset käyttäjät saadaan torjutuksi Stripslashes-funktiolla, eikä sovelluksen tietoturva horju. (autaystavaa.fi…)

Ereg

Ereg on PHP:n funktio (aliohjelma), joka etsii merkkijonoa käyttäen säännöllisiä ilmauksia (regular expressions). Säännölliset ilmaukset ovat merkkijonoja, joissa on erikoismerkkejä. (Sivut.web… a)

Työssä Ereg-funktiota on käytetty mm. postinumeron syötteen tarkastamiseen.

```
(!ereg("[0-9]{5}", $keys['postinumero']) && !empty($keys['postinumero'])
```

Esimerkissä käydään läpi $keys-taulukon postinumero kenttään syötteenä tallennettu viisinumeroinen sarja ja tarkastetaan, että kenttää ei ole jätetty tyhjäksi. Ereg toimii silmukan tavoin. Se käy yhden numeron kerrallaan läpi ja tarkastaa, että annettu numero on positiivinen kokonaisluku välillä 0 ja 9. Muissa tapauksissa käyttäjälle annetaan virheilmoitus.

Md5

Md5 on algoritmi, jonka avulla on mahdollista luoda 128-bittisiä secure hash tarkistussummia eli ns. digitaalisia allekirjoituksia. Lyhenne md tarkoittaa Message Digestiä ja numero versiota (Järvinen 1996). Alla on esimerkki tarkistussummasta.

```
03cdfe1acffd1195e7c5c683822cbefd
```

Tarkistussummia on käytetty rekisteröintipalvelussa salaamaan käyttäjien salasanoja. Alla on php esimerkki siitä, kuinka md5-funktion sisään sijoitetaan haluttu sana ja luotu tarkistussumma siirretään muuttujaan. Sama tehdään toiselle salasanalle ja lopuksi vertaillaan, ovatko tarkistussummat samat. Näin saadaan rakennettua salaus järjestelmä tarkistussummien avulla.

```
$salasana1 = md5('salasana');
$salasana2 = md5('salasana');

if($salasana1 == $salasana2)
```

Tarkistussumman luominen on yksisuuntainen tapahtuma eikä summasta voida palauttaa alkuperäistä sanaa tai viestiä.

4.7 Rekisteröitymisprosessin tekninen kuvaus

Rekisteröintilomakkeen tiedon käsittely, muokkaaminen ja lähettäminen sähköpostin välityksellä

Lomake.php on sivu, joka sisältää rekisteröintilomakkeen sekä käyttäjän syötteeseen reagoivat toiminnot. Aivan ensimmäiseksi aloitetaan istunto, jolloin saadaan käyttäjän sessio-muuttujat toimintaa, jos käyttäjä on jo valmiiksi kirjautuneena sivustolle.

```
<?php
//Aloitetaan istunto.
session_start();
```

Muina sivun yläosan toimintoina liitetään lomake.php-tiedostoon tietokannan asetukset, joita ilman ei tietokantaan päästä käsiksi. Näihin kuuluvat tietokantapalvelimen osoite, tietokannan käyttäjätunnus, salasana sekä tietokannan nimi.

```
//Liitetään tietokannan asetukset ($host, $user, $password, $database).
include("config/database_config.php");
```

Tämän jälkeen liitetään lomake.php-tiedostoon Class.php-tiedosto, jossa sijaitsevat tietokannan käsittelyn ja sähköpostin lähetyksen luokat.

```
//Liitetään luokkatiedosto (mm. tietokantaluokka).
include('Class.php');
```

Kyseisen tiedoston liittämisen jälkeen voidaan luoda uusi tietokanta olio, jonka avulla luodaan yhteys haluttuun tietokantaan.

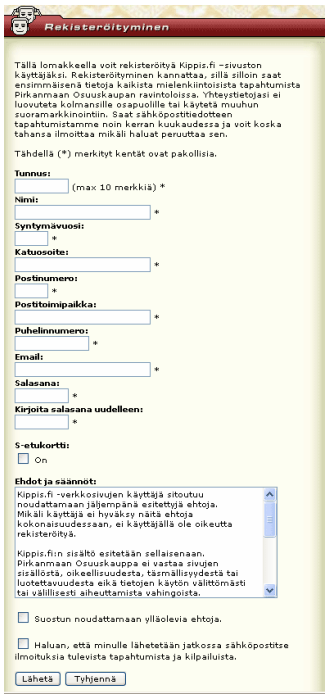
```
//Avataan yhteys tietokantaan käyttäen tietokantaluokan funktioita.
$new_connection = new database;
$new_connection->open_conn($host, $user, $password);
$new_connection->select_database($database);
?>
```

Toiminnon valinta alkuvaiheessa

Lomake.php sisältää kolme päätoimintoa, jotka ovat itse rekisteröinti tiedonkäsittely, sen muokkaus sekä kadonneen salasanan uudelleen luonti. Tästä johtuen ehtolauseilla testataan, mikä toiminnoista on kulloinkin kyseessä. Ensimmäisenä testataan, ettei osoiteriviltä olla saatu muuttujia check ja salasana. Tällä toimenpiteellä varmistetaan, ettei lomaketta tulosteta kahteen kertaan saataessa virheilmoitus käyttäjän syöttämistä tiedoista. Ehtolause poistaa myös mahdollisuuden, että toiminto sekaantuisi salasana toiminnon kanssa. Salasana uusiminen käsitellään hiukan myöhemmin tarkemmin.

```
/*Jos lomaketta ei ole lähetetty eikä saada osoiteriviltä
muuttujan tyyppi arvona salasanaa ja saadan osoiteriviltä
tyyppi muuttujana rekisteri.*/
if(!isset($_GET['check']) && $_GET['tyyppi'] != "salasa-
na" && $_GET['tyyppi'] == "rekisteri" || $_GET['id'])
{
```

Sisemmän ehtolauseen tarkoitus on testata, onko käyttäjän tarkoitus muokata profiiliaan. Käyttäjän tulee olla myös kirjautuneena voidakseen päästä käsiksi toimintoon. Käyttäjän muokatessaan tietojaan siirretään toiminto-muuttujaan arvo rekisteri_muokkaa, joka tullaan sijoittamaan lomakkeen näkymättömään kenttään nimeltä toiminto.



Kuva 19 Rekisteröintilomake

Arvolla ohjataan lomaketta lähetettäessä tietojen käsittely oikeaan toimintoon. Tämän jälkeen lisätään kentat.php-moduuli, joka lisää kirjautuneen käyttäjän sessio muuttujista arvot lomakkeen kenttiin. Lopuksi liitetään rekisteröintilomake (kuva 19). Muussa tapauksessa kuin muokkaamisessa on toimintona uusi rekisteröinti ja toiminto-muuttujaan siirretään arvo rekisteri_tallenna.

```
//Jos saadaan osoiteriviltä muuttuja id ja käyttäjä on kirjautunut. Kyseessä on tällöin käyttäjän tietojen muokkaus.
if($_GET['id'] && isset($_SESSION['kirjautunut']))
{
	/*Lomakkeen toiminto kenttään sijoitetaan arvoksi rekisteri_muokkaa. Kyseistä arvoa käytetään
	valittaessa oikea toiminto käsiteltäessä tietoja.*/
	$toiminto = "rekisteri_muokkaa";
	/*Liitetään kentat.php tiedosto, josta siirretään sessio-muuttujien avulla rekisteröintilomakkeen kenttiin muokkausta varten.*/
	include("rekisteri_inc/kentat.php");
}

//Kyseessä on uusi rekisteröityminen.
else
{
	/*Lomakkeen toiminto kenttään sijoitetaan arvoksi rekisteri_muokkaa. Kyseistä arvoa käytetään
	valittaessa oikea toiminto käsiteltäessä tietoja.*/
	$toiminto = "rekisteri_tallenna";
}
//Liitetään rekisteröintilomake.
include("rekisteri_inc/rekisterointilomake.php");
}
```

Lomakkeen tietojen läpikäynti ja käsittely vaarattomaksi

Kun käyttäjä on syöttänyt tiedot ja lähettänyt ne, testataan saadaanko jokin seuraavista arvoista lomakkeelta: rekisteri_tallenna, rekisteri_muokkaa tai salasana. Lisäksi osoiteriviltä tulee saada muuttuja check, joka toimii merkkinä lomakkeen tiedon käsittelemisestä.

```
//Jos saadaan jokin alla olevista arvoista ja osoiteriviltä muuttuja check.
if($_POST['toiminto'] == "rekisteri_tallenna" || $_POST['toiminto'] == "rekisteri_muokkaa" ||
$_POST['toiminto'] == "salasana" && $_GET['check'] == "yes")
{
```

Tietojen läpikäyntiin käytetään avuksi for-silmukan tapaista foreach-silmukkaa, joka on erittäin kätevä tapa siirtää lähetetyn lomakkeen kenttien nimet ja arvot muuttujiin. Silmukka käy läpi $_POST-muuttujasta saadut arvot eli lomakkeen lähettämät ja muodostaa niistä arvo pareja. Tässä tapauksessa muodostetaan $key-muuttujan arvoksi lomakkeen kentän nimi ja sen arvoksi kentän arvo ($value).

```
/*Käydään kaikki lomakkeelta lähetetyt arvot läpi foreach silmukalla. $key on lomakkeen kentän nimi
 ja $value on sen arvo.*/
foreach($_POST as $key => $value)
{
```

Seuraavaksi ehtolauseilla tarkistetaan, onko kyseessä jompikumpi checkbox-muuttujista. Jos kyseessä ei ollut kumpikaan, voidaan kenttien arvot tehdä vaarattomiksi

stripslashes php-funktiolla, joka poistaa kenoviivat merkkijonoista. Tämä minimoi riskin haitallisille merkkijonoille kirjoitettaessa tietokantaan.
Lopuksi muodostetaan $keys-taulukko muuttuja, jonka indeksin arvoksi sijoitetaan kentän nimi ja sen arvoksi kentän arvo. Tämä helpottaa tiedon hakua taulukosta, kun voidaan hakea haluttu arvo kentän nimen perusteella.

```
//Jos kentän nimi ei ole tiedotteet tai saannot, jotka ovat checkbox elementtejä.
if($key != "tiedotteet" || $key != "saannot")
{
        /*Tehdään lomakkeelta saadut tiedot vaarattomiksi poistamalla niistä kauttaviivat.
        Tämän jälkeen sijoitetaan arvot $keys-taulukkoon, jonka indeksiksi määrätään kentän
        nimi ja indeksin arvoksi kentän arvo.*/
        $keys[$key] = stripslashes($value);
}
//Jos saada kentän nimeksi tiedotteet tai saannot.
else
{



/*Sijoitetaan arvot $keys-taulukkoon, jonka indeksiksi määrätään kentän nimi ja
indeksin arvoksi kentän arvo.*/
$keys[$key] = $value;
}
```

Toiminta virhetilanteissa

Käyttäjän tietoja käsiteltäessä tulee varautua virheisiin ja puutteellisiin tietoihin. Alla on esimerkki tällaisesta virhesyötteeseen reagoinnista. Esimerkiksi on otettu rekisterin tallentamiseen liittyvä toiminto-osuus, mutta järjestelmä reagoi samalla tavalla myös profiilia muokattaessa ja unohtuneen salasanan tapauksessa.

Aluksi on tilanne, jossa ehtolauseella on jaettu kaikki kyseiselle toiminnolle kuuluva koodi omiin lohkoihinsa. Tämän jälkeen sisemmällä ehtolauseella testataan kenttien oikeellisuus, eli ovatko kentät tyhjiä, tai onko tieto virheellistä. Ehtolauseen toteutuessa liitetään lomakkeen käsittelijät-moduuli, joka tarkistaa kentät ja tulostaa virheilmoitukset, kentät-moduuli, joka tulostaa jo kirjoitetut oikeelliset kentät ja rekisteröintilomake-moduulin, joka tulostaa lomakkeen uudelleen.

```
//Jos lomakkeen näkymättömästä kentästä saadaan arvoksi reksiteri_tallenna otetaan käyttöön tal-
lennus toiminto.
if($_POST['toiminto'] == "rekisteri_tallenna")
{


/*Alla olevalla ehtolauseella testataan onko jokin lomakkeen kentistä tyhjä tai saako se väärän
tyyppisen arvon.*/
if(empty($keys['tunnus']) || empty($keys['nimi']) || empty($keys['postinumero']) || emp-
ty($keys['email']) || empty ($keys['salasana']) || empty($keys['saannot']) || !ereg("[0-9]{5}", $ke-
ys['postinumero']) && !empty($keys ['postinumero']))
        {
                /*Jos kentät olivat tyhjiä liitetään alla olevat moduulit, joista ensimmäinen li-
                sää virheiden käsittelijät ja tulostajat. Toinen liittää moduulin, joka tulostaa jo
                kirjoitetut oikeelliset kentät lomakkeelle. Viimeinen tulostaa rekisteröintilo-
                makkeen uudelleen.*/
                include("rekisteri_inc/lomakkeen_kasittelijat.php");
                include("rekisteri_inc/kentat.php");
                include("rekisteri_inc/rekisterointilomake.php");
        }
```

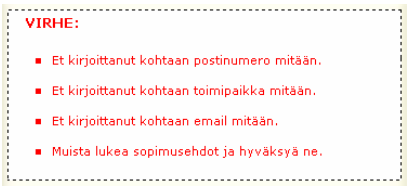
Lomakkeiden virheiden käsittely

Lomakkeen_kasittelijat.php toimii laaja-alaisesti riippumatta lomakkeesta tai toiminnosta. Tämä on mahdollista käyttämällä ehtolauseita, joilla suljetaan pois virheen tarkistukset tietyiltä toiminnoilta ja lomakkeilta. Alla on kaksi hieman erilaista esimerkkiä moduulin toiminnasta.

Ensimmäinen esimerkki testaa, onko nimi-kenttä tyhjä. Lisä ehtona on, että toimintona ei saa olla unohtuneen salasanan uusiminen. Näillä kielteisillä ehtolauseilla voidaan eristää virheentarkistusten toiminta aluetta.

```
//Jos nimi kenttä on tyhjä ja toiminto ei ole salasana.
if(empty($keys['nimi']) && $_POST['toiminto'] != "salasana")
{
        //Tulostetaan virheilmoitus.
        print("Et kirjoittanut kohtaan nimi mit&auml;&auml;n.<br/><br/>");
}
```

Toinen esimerkki on hieman eksoottisempi ja sen tarkoituksena on antaa virheilmoitus (kuva 20), jos postinumero ei sisällä viittä numeroa, joiden numerot ovat väliltä nolla ja yhdeksän. Tämän tarkistuksen toteuttamiseen on käytetty php-funktiota ereg, joka etsii haluttua merkkijonoa säännöllisten merkkijonojen avulla. Ehtolauseen lisänä on myös kaksi kielteistä ehtolausetta, jotka sulkevat pois tapaukset, joissa postinumero-kenttä olisi tyhjä ja toimintona olisi unohtuneen salasanan uusiminen.



Kuva 20 Virhetilanteessa ilmestyvä virhelaatikko

```
/*Jos postinumero kenttä ei ole viisinumeroinen (0-9 numerot) ja kenttä ei ole tyhjä ja toiminto ei
ole salasana.*/
if(!ereg("[0-9]{5}", $keys['postinumero']) && !empty($keys['postinumero']) &&
$_POST['toiminto'] != "salasana")
{
//Tulostetaan virheilmoitus.
print("Postinumero ei ollut oikean pituinen tai k&auml;ytit muita, kuin numeroita. Kirjoita postinumero uudelleen.<br/><br/>");
}
```

Tietojen lisääminen kenttiin

Kun saadaan virheilmoitus tyhjästä kentästä tai virheellisestä arvosta, helpotetaan käyttäjän lomakkeen uudelleen täyttöä lisäämällä lomakkeessa jo olleet oikeelliset arvot valmiiksi kenttiin. Myös tilanteessa, jossa käyttäjä haluaa muokata käyttäjä profiiliaan, käytetään kentat.php-moduulia lisäämään käyttäjän tiedot lomakkeen kenttiin. Alla olevat koodit kuvaavat näitä kahta erillistä tapahtumaa.

Kun osoiteriviltä saadaan check-muuttuja, on kyseessä joko uusi rekisteröintitapahtuma tai olemassa olevan tilin muokkaus. Lomakkeen kenttien arvot haetaan jo aiemmin esitellystä $keys-taulukosta, johon kenttien arvot siirrettiin foreach-silmukan avulla. Muuttujat joihin $keys-taulukon arvot siirretään, sijoitetaan myöhemmin lomakkeen kenttien arvoiksi.

```
//Jos saadaan osoiteriviltä arvoksi check, joka tarkoittaa, että lomake on lähetetty.
if($_GET['check'] == "yes")
{
	//Siirretään lomakkeen kentän arvo-muuttujaksi lomakkeelta lähetetyt tiedot.
	$toiminto = $keys['toiminto'];
	$tunnus = $keys['tunnus'];
	$nimi = $keys['nimi'];
	jne...
```

Osoiteriviltä saataessa id-muuttuja kysymyksessä on profiilin muokkaus hetkellä, jolloin käyttäjä on painanut Muokkaa profiiliasi-painiketta. Muuten toimitaan samoin kuin aikaisemmassa tapauksessa, mutta tällä kertaa siirretään sessio-muuttujat lomakkeen kenttien muuttujiin. Näin saadaan lomakkeella näkymään käyttäjän tiedot esitäytettyinä.

```
//Jos saadaan osoiterivi muuttujan arvoksi id, ollaan profiilin muokkaus tilassa.
else if(isset($_GET['id']))
{
	//Siirretään lomakkeen kentän arvo-muuttujaksi kirjautuneen käyttäjän tiedot.
	$tunnus = $_SESSION['kayttaja_tunnus'];
	$nimi = $_SESSION['nimi'];
	$osoite = $_SESSION['osoite'];
	jne...
```

Kenttien arvot tulostetaan lomakkeen kenttiin käyttämällä kentat.php-moduulista saatuja muuttujia ja tulostamalla ne kentän value osioon.

```
<input type="hidden" name="toiminto" value="<?php print ($toiminto); ?>" />
Tunnus:<br/>
<input name="tunnus" type="text" size="10" maxlength="10" value="<?php print($tunnus);
?>"/><br/>
```

Käyttäjätunnuksen, sähköpostiosoitteen ja salasanan tarkistus

Ennen kuin käyttäjän tili voidaan luoda, tulee tarkistaa, ettei tietokannasta löydy tilejä, joilla on sama käyttäjätunnus ja/tai sähköpostiosoite. Tämä toimenpide parantaa käyttäjän tietoturvaa, koska sisäänkirjautuminen tapahtuu salasanan lisäksi käyttäjätunnuksella. Lisäksi käyttäjä halutaan identifioida käyttäjätunnuksen avulla. Myös sähköpostiosoitteen halutaan olevan henkilökohtainen tietylle käyttäjätunnukselle.

Aluksi tulee suorittaa kyselyt käyttäjätunnuksesta ja sähköpostiosoitteesta. Kyselyn tulosta tarkastellaan php-funktiolla nimeltä mysql_affected_rows, jolla saadaan kyselystä haettujen rivien lukumäärä. Tässä tapauksessa oleellista on se, onko saatu rivien lukumäärä suurempi kuin nolla, sillä siinä tapauksessa vastaava tunnus ja/tai sähköpostiosoite on löytynyt tietokannasta. Lopuksi saatu lukumäärä sijoitetaan kussakin tapauksessa omaan muuttujaansa.

```
//Luodaan tietokanta kyselyt käyttäjän syöttämästä tunnuksesta ja sähköpostiosoitteesta.
$tunnus_kysely = "SELECT kayttaja_tunnus FROM kayttajat WHERE kaytta-
ja_tunnus='".$keys['tunnus']."'";
$email_kysely = "SELECT email FROM kayttajat WHERE email='".$keys['email']."'";
//Suoritetaan kysely tunnuksesta.
$new_connection->make_query($tunnus_kysely);
//Haetaan kyselyssä vastauksiksi saatujen kenttien määrä.
$tunnus_testi = mysql_affected_rows();
```

```
//Suoritetaan kysely sähköpostista.
$new_connection->make_query($email_kysely);
//Haetaan kyselyssä vastauksiksi saatujen kenttien määrä.
$email_testi = mysql_affected_rows();
```

Tilanteessa, jossa tunnus, sähköpostiosoite tai molemmat löytyvät tietokannasta, tulee käyttäjää informoida tästä. Tämä tapahtuu ehtolauseiden avulla, jossa testataan kutakin aikaisemmasta kyselystä saatujen muuttujien arvoilla. Alla olevassa tilanteessa reagoidaan tilanteeseen, jossa käyttäjätunnuksen testistä saatu arvo ei ole nolla eli tietokannasta olisi löytynyt jo vastaava tunnus. Tästä johtuen luodaan $tunnus_virhe-muuttuja ja annetaan arvoksi tosi. Kyseistä muuttujaa käytetään lomakkeen_kasittelijat.php-moduulissa antamaan oikea virheilmoitus. Viimeiseksi liitetään lomakkeen_kasittelijat.php-, kentat.php- ja rekisterointilomake.php-moduulit, jotta saadaan virheilmoitukset, oikeelliset tiedot ja rekisteröintilomake tulostettua.

```
//Jos tunnus ja/tai sähköpostiosoite löytyi.
else {
	//Jos tunnus löytyi jo.
	if($tunnus_testi != 0)
	{
		/*Luodaan muuttuja, joka antaa tunnukseen liittyvän virheilmoituksen
		lomakkeen_kasittelijat.php moduulista.*/
		$tunnus_virhe = true;
	}

	sama toimenpide sähköpostiosoitteelle...
```

Muokattaessa omaa profiilia, toiminnot eroavat hieman vastaavista tallennettaessa. Suurimpana erona voidaan pitää salasanan vaihtamiseen liittyvää toimintoa, jossa kysytään vanhaa salasanaa. Ensimmäisessä ehtolauseessa testataan, ovatko molemmat vaaditut kentät täytetty.

```
//Jos salasana ja vanha_salasana kenttä eivät ole tyhjiä.
if(!empty($keys['salasana']) && !empty($keys['vanha_salasana']))
{
```

Tämän jälkeen luodaan tietokantakysely, jolla haetaan salasanaa tietokannasta, jossa vanha salasana-kentästä saatu md5-salattu tieto on käyttäjän salasana ja käyttäjän id on sessio-muuttujasta saatu id. Lopuksi suoritetaan kysely tietokannasta. Jos kysely palautti rivin (sisempi ehtolause) eli vanha salasana oli oikein, siirretään lomakkeelta saatu salasana-kentästä saatu arvo md5-salattuna muuttujaan, jota käytetään muokattavien tietojen tallennus lauseessa.

```
/*Suoritetaan tietokanta kysely käyttäjän vanhasta tämän hetken salasanasta.
Lomakkeelta saatu salasana tulee salata php-funktio md5:llä.*/
$salasana_kysely = "SELECT salasana FROM kayttajat WHERE salasa-
na='".md5($keys['vanha_salasana'])."' AND kayttaja_id='".$_SESSION['kayttaja_id']."'";
//Suoritetaan kysely.
$new_connection->make_query($salasana_kysely);
//Jos kyselyn tulos on suurempi kuin 0 eli kysely tuotti haetun rivin.
if(mysql_affected_rows() > 0)
{
	//Salataan salasana kentästä saatu muuttuja ja tallennetaan muuttujaan.
	$salasana_salattu = md5($keys['salasana']);
}
```

Tilanteessa, jossa molemmat kentät ovat tyhjiä, siirretään käyttäjän tämän hetken salasana sessio-muuttujasta. $salasana_sallittu-muuttujaa käytetään muokkaus lausekkeessa tallentamaan olemassa oleva salasana.

```
//Jos molemmat kentät ovat tyhjiä.
else if(empty($keys['salasana']) && empty($keys['vanha_salasana'])
{
	//Salasanana pidetään vanhaa salasanaa, joka on kirjautuneen käyttäjän sessio-
	muuttujana.
	$salasana_salattu = $_SESSION['salasana'];
}
```

Ennen muokattavan profiilin tallennusta, suoritetaan vielä tarkistukset käyttäjätunnuksesta ja sähköpostiosoitteesta. Tämä tapahtuu samaan tapaan kuin rekisteröityessä, mutta SQL-lause sisältää pienen eron. Muokatessa profiilia tulee ottaa huomioon, että käyttäjän jo olemassa oleva tunnus ja sähköpostiosoite eivät saa vaikuttaa tietokanta hausta saatuun tulokseen. SQL-lausekkeessa tämä on toteutettu käyttämällä <> -merkkiä, joka tarkoittaa samaa kuin NOT-vertailuoperaattori.

Alla olevassa esimerkissä tätä tapaa on käytetty käyttäjän tunnuksen poissulkemiseen. Virhetilanteessa toimitaan samoin, kuin aikaisemmin tallennuksen yhteydessä todettiin eli annetaan varoitus jo olemassa olevasta tunnuksesta.

```
/*Tehdään tietokantakysely lomakkeelta saadusta käyttäjätunnuksesta. Lause eroaa tallennuksessa
käytetystä loppuosaltaa, jossa tunnus ei saa olla käyttäjän sen hetken tunnus. Jos tunnus on sama,
kuin sen
hetken kysely ei palauta mitään tulosta.*/
$tunnus_kysely = "SELECT kayttaja_tunnus FROM kayttajat WHERE kaytta-
ja_tunnus='".$keys['tunnus']."' AND kayttaja_tunnus <> '".$_SESSION['kayttaja_tunnus']."'";
//Tehdään kysely.
$new_connection->make_query($tunnus_kysely);
//Siirretään kyselyn tulos eli saatujen rivien määrä muuttujaan.
$tunnus_testi = mysql_affected_rows();
```

Käyttäjätilin tallentaminen

Kun rekisteröityminen tapahtuu ilman virheitä, päästään tilanteeseen, jossa tallennetaan käyttäjätili. Tietokantaan syötetään käyttäjän lähettämät tiedot, mutta tiliä ei aktivoida. Tämä ns. deaktivointi toteutetaan muuttamalla $aktivoitu-muuttujan arvoksi 0. Tämän lisäksi tarkistetaan, onko käyttäjä valinnut tiedotteiden lähettämiseen valtuuttavan checkbox-valinnan. Jos valinta on jäänyt tyhjäksi, asetetaan sen arvoksi 0. Näiden toimenpiteiden jälkeen muutetaan lomakkeelta lähetetty salasana php-funktiolla md5 hash muotoon. Lopuksi luodaan kysely, jossa sijoitetaan lomakkeelta saadut tiedot, salattu salasana, deaktivointi muuttuja sekä SQL-funktio now(), joka lisää tämän tallennusajan tietokantaan rekisteroitumis_pvm kenttään.

```
//Deaktivoidaan tili. Tiliä ei voi käyttää niin kauan kuin tietokannassa kenttä aktivoitu saa arvon
1.
$aktivoitu = 0;
//Jos käyttäjä ei halunut tiedotteita.
if(empty($keys['tiedotteet']))
{
	/*Asetetaan muuttujan arvoksi 0. Tämä kenttä tietokannassa tulee olla arvossa 1, jotta
	käyttäjä voi vastaanottaa viestejä kilpailuista ja bileistä.*/
	$keys['tiedotteet'] = 0;
}
```

```
//Salasanan muuttaminen hash-muotoon php-funktiolla md5.
$salasana = md5($keys['salasana']);
//Tilin tallentaminen tietokantaan.
$uusi_kayttaja = "INSERT INTO kayttajat(kayttaja_tunnus, nimi, osoite, postinumero, email, sa-
lasana, tiedotteet, aktivoitu, rekistoroitumis_pvm) VALUES('".$keys['tunnus']."',
'".$keys['nimi']."', '".$keys['osoite']."', '".$keys['postinumero']."', '".$keys['email']."', '".$salasana."',
'".$keys['tiedotteet']."', '".$aktivoitu."', now())";
//Tehdään tietokanta kysely.
$new_connection->make_query($uusi_kayttaja);
```

Muokkaus tilassa tallennus eroaa SQL-lausekkeeltaan tallennuksesta ja insert-kyselyn sijaan käytetään update-komentoa, jolla voidaan päivittää jo olemassa olevaa riviä tietokannassa.

```
//Luodaan muokkaa kysely.
$muokkaa_kysely = "UPDATE kayttajat SET kayttaja_tunnus='".$keys['tunnus']."', ni-
mi='".$keys['nimi']."', osoite='".$keys['osoite']."', postinumero='".$keys['postinumero']."',
email='".$keys['email']."', salasana='".$salasana_salattu."', tiedotteet='".$keys['tiedotteet']."'
WHERE kayttaja_id='".$_SESSION['kayttaja_id']."'";
```

Koska käyttäjä on kirjautuneena muokatessaan profiiliaan, tulee hänen tietonsa päivittää myös sessio-muuttujiin, joita tarvitaan oltaessa kirjautuneena. Tämä tapahtuu liittämällä sessioiden päivitykseen liittyvän moduulin.

```
//Sessioiden päivitys moduulin liittäminen.
include("rekisteri_inc/sessioiden_paivitys.php");
```

Sessioiden päivitys moduuli sisältää kyselyn, jolla haetaan SQL-kyselyllä kirjautuneen käyttäjän id:n perusteella hänen tietonsa. Nämä tiedot siirretään edelleen sessio-muuttujiin, jolloin näiden arvot päivittyvät.

```
//Tehdään kysely käyttäjän tiedoista tietokannassa.
$kayttajan_tiedot = "SELECT kayttaja_tunnus, nimi, postinumero, email, tiedotteet, osoite, sala-
sana FROM kayttajat WHERE kayttaja_id='".$_SESSION['kayttaja_id']."'";
//Suoritetaan kysely.
$new_connection->make_query($kayttajan_tiedot);
//Haetaan kyselyn tulokset.
$tiedot = $new_connection->fetch_query();

//Päivitetään sessio-muuttujien arvot tietokannasta saaduilla tuloksilla.
$_SESSION['kayttaja_tunnus'] = $tiedot[0];
$_SESSION['nimi'] = $tiedot[1];
$_SESSION['postinumero'] = $tiedot[2];
jne...
```

Aktivoimisviestin lähettäminen

Kun käyttäjätili on tallennettu, käynnistyy viimeinen vaihe ennen mahdollista käyttäjän itse toteuttamaa aktivointia. Rekisteröityvän käyttäjän id siirretään erilliseen muuttujaan php-funktiolla mysql_insert_id(), joka hakee viimeiseksi MySQL AUTO_INCREMENT -operaatiolla luodun id:n arvon. Tämä tapahtuma on siitä oleellinen, että operaatiosta saatu numero tullaan sijoittamaan sähköpostiviestin aktivoimislinkkiin.

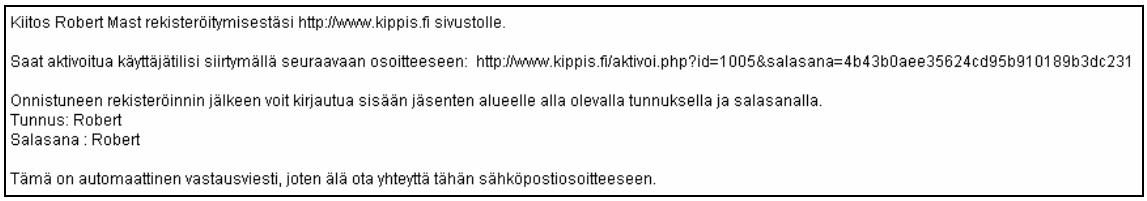
```
//Siirretään viimeisimmän tallennetun käyttäjän id muuttujaan.
$kayttaja_id = mysql_insert_id();
```

Sähköpostin lähetykseen käytetään oliopohjaista ratkaisua. Ensimmäiseksi luodaan uusi olio, jonka jälkeen luodaan sille ominaisuudet from, to, subject ja body. Viestin sisältö sijoitetaan body -osuuteen, jossa informoidaan onnistuneesta rekisteröinnistä ja annetaan linkki, minkä avulla aktivointi suoritetaan. Linkkiin syötetään juuri haettu käyttäjän id sekä salattu salasana. Tämän jälkeen ilmoitetaan vielä käyttäjän tilin salasana ja käyttäjätunnus. Lopuksi viesti lähetetään $mail->send() komennolla.

```
//Luodaan uusi mime_mail-olio.
$mail = new mime_mail;
//Keneltä sähköposti on tulossa.
$mail -> from = "rekistorointi@kippis.fi";
//Kenelle se on osoitettu (rekisteröityneen osoite).
$mail -> to = $keys['email'];
//Viestin aihe.
$mail -> subject = "Varmennus rekistöröitymisestä";
//Viestin sisältö käyttäjälle aktivointilinkkeineen.
$mail -> body = "Kiitos ".$keys['nimi']." rekistöröitymisestäsi http://www.kippis.fi sivustolle.
Saat aktivoitua käyttäjätilisi siirtymällä seuraavaan osoitteeseen:
http://www.kippis.fi/aktivoi.php?id=".$kayttaja_id."&salasana=".$salasana."
Onnistuneen rekist&ouml;r&ouml;innin j&auml;lkeen voit kirjautua sis&auml;&auml;n
j&auml;senten alueelle alla olevalla tunnuksella ja salasanalla.
Tunnus: ".$keys['tunnus']."
Salasana : ".$keys['salasana']."
T&auml;m&auml; on automaattinen vastausviesti, joten &auml;l&auml; ota yhteytt&auml;
t&auml;h&auml;n s&auml;hk&ouml;postiosoitteeseen.";
//Postin lähetys.
$mail -> send();
```

Tilin aktivointi

Käyttäjä tilin aktivointi on erittäin yksinkertainen operaatio teknisessä mielessä. Oleellista tapahtumassa on, että käyttäjä painaa sähköpostiinsa saapuneessa vahvistusviestissä (kuva 21) olevaa linkkiä, joka ohjaa hänen aktivoi.php-sivulle.


Kiitos Robert Mast rekisteröitymisestäsi http://www.kippis.fi sivustolle.

Saat aktivoitua käyttäjätilisi siirtymällä seuraavaan osoitteeseen: http://www.kippis.fi/aktivoi.php?id=1005&salasana=4b43b0aee35624cd95b910189b3dc231

Onnistuneen rekisteröinnin jälkeen voit kirjautua sisään jäsenten alueelle alla olevalla tunnuksella ja salasanalla.
Tunnus: Robert
Salasana : Robert

Tämä on automaattinen vastausviesti, joten älä ota yhteyttä tähän sähköpostiosoitteeseen.

Kuva 21 Sähköpostiin saapuva aktivointiviesti

Linkin mukana välittyy myös osoiterivi-muuttujina käyttäjän id sekä salasana. Nämä muuttujat saadaan GET-menetelmällä noukittua osoiteriviltä ja ne tehdään vaarattomiksi tietokantakyselyn kannalta käyttäen php-funktiota stripslashes().

```
//Haetaan osoiteriviltä id ja salasana ja tehdään ne vaarattomiksi.
$kayttaja_id = stripslashes($_GET['id']);
$salasana = stripslashes($_GET['salasana']);
```

Seuraavaksi toteutetaan itse aktivointikysely tietokannasta, jossa asetetaan aktivoitu kentän arvo 1:ksi sellaisessa tapauksessa, kun käyttäjän id ja salasana täsmäävät samaan käyttäjätiliin. Jos tiedot eivät täsmää, ei tiliäkään aktivoida.

```
//Aktivoimiskysely.
$aktivoi_kysely = "UPDATE kayttajat SET aktivoitu='1' WHERE kayttaja_id=".$kayttaja_id."
AND salasana='".$salasana."'";
//Tehdään kysely.
$new_connection->make_query($aktivoi_kysely);
```

Käyttäjää informoidaan ehtolauseen avulla siitä, miten aktivointi onistui. Tässä tapauksessa käytetään php-funktiota mysql_affected_rows(), joka hakee kyselyssä haettujen rivien määrän. Jotta käyttäjä saisi positiivisen vastauksen, tulee funktion tuloksen olla 1. Muussa tapauksessa tulostetaan epäonnistuneesta aktivoinnista kertova viesti.

```
//Jos aktivointikysely palautti yhden rivin tuloksena.
if(mysql_affected_rows() == 1)
{
        //Tulostetaan ilmoitus onnistuneesta aktivoinnista ja linkki etusivulle.
        print("Tilisi on aktivoitu! Siirry takaisin p&auml;&auml;sivulle <a
        href='index.php'>t&auml;st&auml; linkist&auml;</a> ja kirjaudu sis&auml;&auml;.");
}

else {
        //Tulostetaan ettei aktivointi onnistunut ja käyttäjää pyydetään rekisteröitymään uudel-
        leen.
        print("Tili&auml; ei saatu aktivoitua! Siirry takaisin rekister&ouml;intilomakkeelle
        <a href='lomake.php?tyyppi=rekisteri'>t&auml;st&auml; linkist&auml;</a>.");
}
```

Sisään- ja uloskirjautuminen

Sisään- ja uloskirjautumiseen liittyvät toiminnot on sijoitettu header.php-moduuliin, joka sisältää myös tyylitiedostojen linkitykset sekä päänavigaation. Toimintojen sijoitteluun vaikutti header.php-tiedoston sijainti globaalisti sivustolla, jolloin kirjautumiseen liittyvät toiminnot saatiin mahdollisimman laaja-alaisesti käyttöön.

Sisäänkirjautuminen on melko yksinkertainen toimenpide. Ensimmäinen vaihe kirjautumisessa on ehtolauseessa tarkistettavat kirjautumislomakkeelta (kuva 22) saadut arvot. Lomakkeelta tulee saada näkymätön kenttä login sekä kentät tunnus ja salasana eivät saa olla tyhjiä.

```
//Jos saadaan kirjautumis lomakkeelta kentät login ja
kentät tunnus ja salasana eivät ole tyhjiä.
if(isset($_POST['login']) &&
!empty($_POST['tunnus']) && !emp-
ty($_POST['salasana']))
{
```

Kuva 22 Kirjautumis laatikko

Tämän jälkeen käyttäjätunnus ja salasana-kentistä saadut arvot tehdään vaarattomiksi käyttämällä php-funktiota stripslashes ja salasana muutetaan vielä salattuun muotoon käyttämällä php-funktiota md5 ennen siirtämistä muuttujaan.

```
//Siirretään lomakkeelta saatu tunnus vaarattomana muuttujaan.
$k_tunnus = stripslashes($_POST['tunnus']);
//Siirretään lomakkeelta saatu salasana vaarattomana muuttujaan ja salataan se php-funktiolla
md5.
$k_salasana = md5(stripslashes($_POST['salasana']));
```

Seuraavaksi luodaan ja suoritetaan kysely, jolla haetaan tietokannasta käyttäjä, jonka käyttäjätunnus sekä salasana ovat lomakkeelta saatuja ja hänen tilinsä on aktivoitu.

```
//Tehdään kysely käyttäjän tiedoista missä salasana ja tunnus ovat saman tietokannassa, kuin lo-
makkeelta saadut ja tili tulee olla aktivoitu.
$kirjautumis_kysely = "SELECT kayttaja_id, kayttaja_tunnus, nimi, postinumero, email, tiedot-
teet, salasana, osoite FROM kayttajat WHERE kayttaja_tunnus='".$k_tunnus."' AND salasa-
na='".$k_salasana."' AND aktivoitu='1'";
//Suoritetaan kysely.
$new_connection->make_query($kirjautumis_kysely);
//Kyselyn tulokset.
$tulos = $new_connection->fetch_query();
```

Tilanteessa, jossa kysely palautti tuloksen, tallennetaan tietokantaan kirjautumisaika ja siirretään aikaisemmasta kyselystä saadut käyttäjätiedot sessio-muuttujiin. Näin saadaan tallennettua käyttäjän tiedot kirjautumisen jälkeistä käyttöä varten.

```
//Jos kyselyn ehdoilla löytyi käyttäjä tietokannasta.
if(mysql_affected_rows() > 0)
{
        //Luodaan kysely kirjautumisajan tallennuksesta tietokantaan.
        $kirjautumisaika = "UPDATE kayttajat SET kirjautunut_viimeksi=now() WHERE
                kayttaja_tunnus='".$tulos[1]."'";
        //Suoritetaan kysely.
        $new_connection->make_query($kirjautumisaika);
```

Palvelut, jotka vaativat kirjautumisen, edellyttävät $_SESSION['kirjautunut']-muuttujan arvoksi ”true”. Tällöin alla olevan esimerkin ehtolause toteutuu ja käyttäjä pääsee nauttimaan rekisteröityneille tarkoitetuista palveluista.

```
//Jos käyttäjä on kirjautunut.
if($_SESSION['kirjautunut'])
{
```

Uloskirjautuminen edellyttää, että Kirjaudu ulos-painiketta (kuva 23) on painettu, jolloin linkki lähettää osoiterivin kautta muuttujan logout. Tätä muuttujaa verrataan ehtolauseessa ja sen toteutuessa tuhotaan kaikki kirjautuessa luodut, käyttäjän tiedot sisältävät sessio-muuttujat ja lopuksi koko sessio. Tästä eteenpäin käyttäjä ei ole enään kirjautunut.

Kuva 23 Kirjaudu ulos -painike

```
//Jos saadaan kirjaudu ulos-painikkeen lähettämä logout-muuttuja osoiterivin kautta.
if(isset($_GET['logout']))
{
        //Tuhotaan sessio-muuttujat.
        unset($_SESSION['kayttaja_id']);
        unset($_SESSION['kayttaja_tunnus']);
        unset($_SESSION['nimi']);
```

```
jne...

//Tuhotaan koko sessio.
session_destroy();
```

Toiminta kadonneen salasanan tapauksessa

Kadonneen salasanan tilalle on mahdollista luoda uusi satunnaissalasana. Kyseinen toiminto sijaitsee lomake.php-tiedostossa, joka valitaan linkin avulla tuodun osoiterivi muuttujan avulla. Tuon muutujan nimi on tyyppi ja sen arvona on salasana. Tällöin lomakkeelle sijoitettava sen hetkistä toimintoa kuvaava muuttuja luodaan ja sen arvoksi tulee salasana. Lopuksi liitetään kadonnut_salasana_lomake.php-moduuli, joka sisältää toiminnossa käytettävän lomakkeen.

```
else if($_GET['tyyppi'] == "salasana")
{
	$toiminto = "salasana";
	include("rekisteri_inc/kadonnut_salasana_lomake.php");
```

Kun tiedot ovat lähetetty lomakkeelta, tarkistetaan uloimmalla ehtolauseella, että kumpikaan kentistä ei ole tyhjä.

```
//Jos kumpikaan lomakkeen kentistä ei ollut tyhjiä.
if(!empty($keys['tunnus']) && !empty($keys['email']))
{
```

Seuraavaksi suoritetaan tietokantakysely, jonka tarkoituksena on tarkistaa, täsmäävätkö käyttäjän syöttämä tunnus ja sähköpostiosoite samaan henkilöön.

```
//Luodaan kysely, jolla varmennetaan, että käyttäjän tunnus täsmää sähköpostiosoitteeseen.
$kayttajan_varmennus = "SELECT email FROM kayttajat WHERE email='".$keys['email']."'
AND kayttaja_tunnus='".$keys['tunnus']."'";
//Suoritetaan kysely.
$new_connection->make_query($kayttajan_varmennus);
```

Tilanteessa, jossa tiedot ovat täsmänneet, luodaan satunnaisluku-funktiolla uusi salasana ja sijoitetaan se muuttujaan. Satunnaisluvun luonnin jälkeen luodaan sähköpostiviesti samaan tapaan kuin on kuvattu aktivoimisviestin lähettämisosiossa ja tähän liitetään myös juuri luotu satunnaisluku uudeksi salasanaksi.

```
//Jos kysely nouti rivin eli tunnus ja email täsmäsivät.
if(mysql_affected_rows() > 0)
{
	//Luodaan satunnais salasana funktiolla makeRandomPassword().
	$satunnais_salasana = makeRandomPassword();
```

Lopuksi tulee juuri luotu salasana salata php-funktiolla md5 ja tallentaa alla olevalla kyselyllä tietokantaan.

```
//Luodaan kysely, jolla päivitetään uusi salasana tietokantaan.
$muokkaa_salasanaa = "UPDATE kayttajat SET salasana='".md5($satunnais_salasana)."'
WHERE kayttaja_tunnus='".$keys['tunnus']."' AND email='".$keys['email']."'";
```

5 Sitoutumista vahvistavat palvelut

Aiemmissa luvuissamme yksityiskohtaisesti esitellyn rekisteröitymispalvelun tarkoituksena on olla ensimmäinen askel sivustolle tulevien käyttäjien sitouttamisessa. Tarvitaan kuitenkin muitakin keinoja henkilöiden sitoutumisen vahvistamiseksi. Heille tulee tarjota palveluita, jotka ovat jollain tavoin arvokkaita. Jo aikaisemmin POK:n kanssa käymiemme kehityskeskusteluiden aikana nousi esille ajatus kilpailujen järjestämisestä sekä erilaisista juhlista ja tapahtumista mainostaminen.

Kävijöitä ei tulisi vaatia rekisteröitymään, vaan tehokkaampi keino on pyytää heitä liittymään jäseniksi. Pelkän rekisteröintilinkin tai -painikkeen lisäksi on hyvä luoda liittymisen eduista kertova oma sivu tai teksti, minkä vuoksi sivustolle kannattaa rekisteröityä. Millaista konkreettista hyötyä asiakkaalle sitten voidaan tarjota? Vaihtoehtoina voivat olla mm. laajemmat pääsyoikeudet tietokantoihin, kilpailut, keskustelufoorumi, Bulletinboard -tyylinen ilmoitustaulu tai sähköpostiin lähetettävät viestit tapahtumista ja ravintoloiden tarjouksista. Kaikki edellä mainitut vaihtoehdot on aiemmin todettu tehokkaiksi sitouttamiskeinoiksi.

Pääasiallisesti sitouttaminen alkaa kohderyhmän kartoittamisesta ja heille räätälöidyistä tarjouksista tai palveluista. Kippis.fi -sivuston kohderyhmänä ovat ensisijaisesti ulkona käyvät nuoret aikuiset, jotka joko entuudestaan tuntevat Pirkanmaan Osuuskaupan ravintoloita tai he, jotka muuten käyvät ulkona, mutta eivät välttämättä POK:n ravintoloissa. Vanhoja asiakkaita pyritään siis osaltaan sitouttamaan tällä tavalla ja samalla houkutellaan uusia kattavalla palvelulla.

SitePoint.com on suosittu online-julkaisu verkkokehittäjien ja -suunnittelijoiden keskuudessa. Heidän fooruminsa on toiminut heinäkuusta 1999 alkaen ja siellä vierailee kymmeniätuhansia aktiivisia jäseniä. Tämä aktiivisuus osoittaa, että sitouttaminen on heidän osaltaan onnistunut todella tehokkaasti. SitePoint.com on koonnut useista eri lähinnä kommuuneille tarkoitetuista foorumeista viisi sitouttamista edesauttavaa neuvoa, joiden huomasimme soveltuvan suurilta osin myös omaan projektiimme. Niinpä muokkasimme omat viisi neuvoamme:

1. Toivota tervetulleeksi ja neuvo uusia käyttäjiä

- Tervetulleeksi toivottaminen on ensimmäinen asia, josta täytyy lähteä liikkeelle. Tämä saa käyttäjän tuntemaan, että hänestä välitetään ja että hänet on huomattu.
- Kattavien ohjeistuksien liittäminen sivustolle edesauttaa käyttäjää löytämään haluamansa ja ongelmatilanteita esiintyy harvemmin. Tästä johtuen käyttökokemus on positiivinen ja sivustolle palataan uudestaan
- Uusien käyttäjien mahdollisiin kysymyksiin pitäisi pystyä vastaamaan mahdollisimman nopeasti. Mikäli käyttäjälle jätetään vastaamatta, hän tuskin tulee sivulle uudestaan.
- Uusien käyttäjien kysymyksiin on suhtauduttava kunnioituksella. ”Etkö lukenut sitä pientä tekstiä kymmenennellä alasivulla...” -tyyliset vastaukset eivät toimi eli vastaa nopeasti, ystävällisesti ja kohteliaasti. Näin käyttäjät tulevat varmemmin sivuille myös jatkossa. (Sitepoint 1998)

2. Pidä rekisteröityneisiin yhteyttä

- Vältä massasähköposteja käyttäjille, ellet jaa tuotteita/palveluita ilmaiseksi. Jos käyttäjä kokee joutuneensa roskapostin uhriksi, sivusto tulee suurella todennäköisyydellä kaatumaan.
- Äänestykset ja muut vastaavat interaktiivisuutta vaativat palvelut tulisi laittaa näkyville.
- Rakenna pieni uutiskirje (enewsletter), joka kertoo sivuston tuoreimmista päivityksistä. Tämä antaa positiivisen käsityksen sivuston jatkuvasta kehityksestä. (Sitepoint 1998)

3. Pidä kilpailuja

- Kilpailut, joista käyttäjällä on mahdollisuus voittaa jotain konkreettista, ovat aina tehokkaita. Jos ilmaiseksi (tai vain rekisteröitymällä) voi saada palkintoja, harva jättää tilaisuuttaan käyttämättä.
- Yritä saada palkinnoille sponsori (vastapalveluna anna esimerkiksi mainostilaa sivuilta) ja laita palkinnot tulevan tapahtuman yhteyteen. (Sitepoint 1998)

4. Pidä sivu aktiivisena

- Kirjoita etusivulle uusia mielenkiintoisia uutisia. Näin käyttäjät näkevät sivun säilyvän aktiivisena ja heidän mielenkiintonsa pysyy yllä.
- Sivun informaation päivittäminen auttaa ylläpitämään mielikuvaa aktiivisesta sivustosta (tuotteiden, uutisten ja kyselyjen päivittäminen). (Sitepoint 1998)

5. Tee rekisteröitymisestä ”pakko saada” –juttu

- Ilmaise konkreettiset syyt, mitä hyötyä rekisteröitymisestä on.
- Osoita käyttäjälle sivuston olevan hyödyllinen hänelle, jossa hänen kysymyksiinsä vastataan ja häntä arvostetaan. Kun olet saanut arvokkaita jäseniä, pidä heistä kiinni. (Sitepoint 1998)

Jo projektin alkumetreiltä lähtien halusimme taata tasaisen kävijämäärän myös tulevaisuudessa. Samaa halusivat myös Pirkanmaan osuuskaupan edustajat, sillä se olisi heillekin hyvä kanava houkutella asiakkaita. Kävijöiden sitouttamiseksi tuli keksiä mahdollisimman toimivia ratkaisuja.

Kilpailujen järjestäminen päätettiin toteuttaa kaksivaiheisesti. Ensimmäisessä vaiheessa sivuston hallinnointipuolelle rakennettiin käyttöliittymä, jonka avulla oli mahdollista luoda sisältöä etusivulla olevaan kilpailuosioon, joka toteutettiin kyseistä tarkoitusta varten. Toisen vaiheen tarkoituksena on siirtää kyseinen palvelu vain rekisteröityneiden käyttöön. Ensimmäinen kilpailu järjestettiin 10.10.2005 - 6.11.2005 välisenä aikana heti sivuston avauduttua. Käyttäjien piti vastata muutamaan kysymykseen ja viikoittain arvottiin kaksi 10 euron suuruista lahjakorttia.

Rekisteröinnin yhteydessä keräsimme kävijöiltä perustiedot (tunnus, nimi, syntymävuosi, osoite, puhelinnumero, sähköpostiosoite ja salasana), joiden avulla on mahdollista luoda käyttäjistä profiilit ja mahdollisesti tehostaa suoramarkkinointia. Sähköpostiosoitteista luotiin postituslista, joka toimii tehokkaana markkinointikanavana käyttäjille.

Rekisteröinnin yhteydessä (ennen tilin aktivoimista) sähköpostiin lähetettiin varmistusviesti, jonka yhteydessä toivotettiin käyttäjä tervetulleeksi.

Sivustolla pyritään lisäämään uutisia vähintään viikoittain, jota voidaan pitää rajapyykkinä aktiivisuuden määrittelemisessä. Siinä on onnistuttu kohtalaisesti, vaikkakin ravintolapäälliköt voisivat uutisoida enemmän.

Juhlista ja tapahtumista tiedottaminen ja muiden lisäetujen tarjoaminen on myös lisäarvo, joka on vain rekisteröityneitä varten. Tätä varten toteutettiin erillinen joukkopostin lähetystoiminto, johon käytettiin hyväksi aikaisemmin esiteltyä mime_mail-luokkaa. Kyseinen toiminto lähettää postia kaikille niille rekisteröityneille, jotka ovat ilmoittaneet rekisteröintilomakkeessa haluavansa vastaanottaa erinäisiä tiedotteita.

Tulevaisuudessa rekisteröintiin liittyviä palveluita tullaan varmastikin lisäämään ja monipuolistamaan. Valmiin ja toimivan rekisteröintipalvelun ympärille on helppo lisätä tulevaisuudessa uusia palveluita.

6 Yhteenveto

Kokonaisuutena Kippis.fi:stä tuli onnistunut ja toimiva kokonaisuus niin toteuttajien, kuin POK:in organisaatiosta mukana olleidenkin mielestä. Sivusto täyttää sille asetetut vaatimukset niin toiminnallisuudeltaan kuin tarjoamalla sisällöltään.

Sivustosta tuli ulkonäöllisesti vetoava ja sen tietosisällöstä tarpeeksi monipuolinen kiinnostaakseen niin uusia kuin vanhojakin käyttäjiä myös tulevaisuudessa. Sivuston hallintopuolesta tuli tarpeeksi yksinkertainen, jotta käyttöliittymiin tottumattomat ravintolapäälliköt osasivat sitä käyttää. Myös käyttämämme standarditeknologiat tulevat pitämään sivuston toimintakelpoisena vielä pitkälle tulevaisuudessa.

Toiminnallisuudeltaan sekä laajuudeltaan saavutimme, sen mitä olimme POK:in kanssa aikaisemmin sopineet ja päättäneet toteuttaa.

Rekisteröintipalvelun osalta asetimme tavoitteeksi luoda toimivan, tehokkaan, käyttäjäystävällisen ja muokattavan järjestelmän. Mielestämme tavoitteet saavutettiin hyvin.

Hyvä toimivuus saavutettiin tarkastelemalla muiden sivustojen vastaavia palveluita sekä tukeutumalla mahdollisimman suoraviivaiseen ja yksinkertaistettuun ulkoasuun. Myös rakenteella eli CSS:än ja XHTML:än käytöllä on vaikutusta palvelun toimivuuden kannalta eri laitteistoympäristöissä ja kyseiset tekniikat takaavatkin paremman ja vakaamman toimivuuden, kuin standardittomat tekniikat. Niiden avulla on mahdollista esittää tietoa jäsennetysti myös laitteistoissa, jotka ovat rajoitettuja esittämistapojensa suhteen.

Teknisessä mielessä toimivuudelle asettamamme tavoitteet toteutuivat omalta osaltamme eikä palvelun toiminnassa ole havaittu ongelmia. Eräs tekninen ongelma kuitenkin ilmaantui, joka oli meidän työpanoksestamme riippumaton. Tietyt sähköpostipalvelut (Suomi24.fi) eivät suostuneet vastaanottamaan aktivointiviestiä ja näin ollen rekisteröitynyt ei saanut aktivoitua tiliään eikä päässyt kirjautumaan sivuille. Ongelma ilmeisesti johtui palvelimen sähköpostiasetuksista. Kyseiseen ongelmaan etsitään ratkaisua.

Tehokkuus saavutettiin yksinkertaisella PHP:n avulla luodulla modulaarisella rakenteella, jonka avulla samoja elementtejä voitiin käyttää uudelleen. Samalla rakenteesta saatiin helposti ymmärrettävä. Yksinkertaisen tietokantarakenteen avulla oli mahdollista saavuttaa tehokkuus etuja monimutkaisten kyselyiden jäädessä pois. Palvelun tiedonsiirrolle aiheuttamat rasitteet minimoitiin käyttämällä CSS-tyylitiedostoja, jotka ovat kevyitä sivujen muotoilussa. Kaiken kaikkiaan tavoite tehokkuuden osalta toteutui.

Palvelu täytti käyttäjäystävällisyyteen liittyvät kriteerit. Lomakkeista tuli lyhyitä, ytimekkäitä ja käyttäjää opastettiin pakollisista kentistä sekä häntä informoitiin väärin täytetyistä kentistä. Häntä opastettiin myös sähköpostin välityksellä tapahtuvista toiminnoista.

Rekisteröintipalvelun käytettävyyteen liittyvät tekijät toteutuivat myös ja tästä esimerkkinä voidaan pitää melko suurta rekisteröityneiden määrää kolmen viikon aikavälillä suhteessa, että sivusto on vielä melko tuntematon (101 rekisteröitynyttä). Sivustoa testattiin usean käyttäjän voimin ja sse todettiin käytettävyydeltään onnistuneeksi.

Muokattavuus toteutui pitkälti yksinkertaisen tietokantarakenteen sekä modulaarisuuden avulla. Toiminnot sijaitsevat tarkasti rajatuissa osioissaan ja täten muokkaaminen on helposti toteutettavissa. Koodi myös kommentoitiin muokkausta ja järjestelmän laajennusta silmällä pitäen.

Muokattavuuteen liittyvät tavoitteet toteutuivat niin pitkälle, kuin ne olivat mahdollisia toteuttaa. Lopullinen onnistuminen testataan vasta, kun sivustoa laajennetaan tai muokataan.

Sitouttamiseen liittyvät palveluihin liittyvä tavoite toteutui niiltä osin, mitä toteutimme seuraten SitePoint.com -sivustolta saatuja ohjenuoria. Voidaan kuitenkin sanoa, että tavoite ei täysin onnistunut, sillä sitouttamiseen liittyviä palveluita toteutettiin melko vähän. Huolimatta vähäisestä toteutettujen palveluiden määrästä, kilpailu palvelu toimi moitteitta. Myös joukkopostin lähettäminen 24.11.2005 ravintola Emmassa järjestetyistä Kippis.fi -bileistä toimi moitteitta muiden, kuin niiden muutaman sähköposti palvelun osalta, jotka aiheuttivat ongelmia jo aikaisemmin mainitun aktivoimistoiminnon kohdalla. Paikanpäälle Emmaan ilmestyi yli 200 henkilöä, mikä oli enemmän kuin normaali torstainen väkiluku.

Kokonaisuutena voidaan sanoa, että toimeksiannolle asettamamme tavoitteet toteutuivat niissä puitteissa kuin se oli ajallisesti sekä resurssien suhteen mahdollista.

Tutkintotyölle asetettu päätavoite toteutui suuriltaosin. Rekisteröintijärjestelmän rakenne esitettiin selkeästi ja varsinkin prosessin teoria osio kuvineen selvitti tarkasti kuvina rekisteröintipalveluun liittyvät tapahtumaketjut. Työssä esiteltiin käytetyt tekniikat tarpeeksi yksityiskohtaisesti, vaikkakin kuvallisia esimerkkejä olisi voinut olla tukemassa sanomaa. Ehkäpä huonoiten päätavoite toteutui järjestelmän vaatimukset osiossa, jossa ei päästy tukeutumaan kirjalliseen lähdeaineistoon, johtuen rekisteröintipalvelua käsittelevän kirjallisuuden ja teorian puutteesta. Prosessien tekninen kuvaus antaa todella täsmällisen kuvauksen koodi tasolla, kuinka asiat toimivat ja sitä voidaan soveltaa tai käyttää suoraan luotaessa ja suunniteltaessa rekisteröintipalvelua. Varsinkin teknisenä kuvauksena tutkintotyö toimii hyvänä lähteenä luotaessa rekisteröintipalveluita, mutta teoreettisena teoksena se ei täysin onnistu eli kattavana kuvauksena tutkintotyö hieman epäonnistui.

Tutkintotyön osatavoitteet toteutuivat. Kippis.fi -sivustolla käytetyt teknologiat (PHP, MySQL, CSS, XHTML) esiteltiin tarkasti ja laaja-alaisesti tukeutuen moniin lähteisiin. Tämä esittely tarjoaa tarkan kuvan aina historiasta teknologioiden perustoimintaan. Myös Kippis.fi -sivuston rakenne ja ulkoasu kuvattiin tarkasti. Ainoastaan sivuston dynaamisuuteen liittyvät näkökohdat jäivät hieman vajavaiselle kuvailulle. Pienistä vajavaisuuksista huolimatta tutkintotyö kuvaa kokonaisvaltaisesti laajan dynaamisen sivuston vaatiman teknologian ja suunnitteluun liittyvät näkökohdat. Sitouttamisosatavoitteessa esiteltiin SitePoint -sivulta saaduin toimintatavoin niiden soveltamista Kippis.fi -sivuilla. Tutkintotyössä onnistuttiin osoittamaan SitePoint -sivustolta saamiemme tietojen onnistuneen soveltamisen Kippis.fi -sivuston tarpeisiin ja näin todettiin sitouttamiseen liittyvien näkökohtien oikeellisuus.

Tulevaisuutta ajatellen sivustoon jää vielä paljon lisättävää, kehitettävää ja varmastikin korjattavaa. Varsinkin sitoutumista vahvistavat palvelut jäivät hieman lapsen kenkiin, koska ajalliset resurssit eivät niihin yksinkertaisesti riittäneet.

Eräänä sitouttamiseen liittyvänä palveluna tulevaisuudessa saattaa olla oma keskustelu foorumi, vain rekisteröityneiden käyttöön tarjoten erittäin tehokkaan tavan sitouttaa käyttäjät sivustoon. Rekisteröintipalvelu ei tulevaisuudessa sinällään tarvitse suuria parannuksia tai lisäyksiä, vaan se täyttää sille asetetut vaatimukset. Sitä on kuitenkin helppo muokata tulevaisuuden tarpeita ajatellen.

Lähteet

Kirjat:

- Boumphrey Frank, Greer Cassandra, Raggett Dave, Raggett Jenny, Schnitzenbaumer Sebastian, Wugofski Ted. 2000. Beginning XHTML. Wrox Press. Inside XHMTL - Ohjelmoijan käsikirja. Suomentanut Kauko Kolehmainen. Helsinki: IT Press.

- Meloni, Lucie C. 2004. SAMS Teach Yourself - PHP, MySQL and Apache. SAMS.

- Meloni, Julie C. 2003. Sam's Teach Yourself MySQL in 24 Hours. Pearson Education. MySQL Trainer Kit. Suomentanut Riitta Santala-Köykkä. Helsinki: IT Press.

- Lahtonen, Tommi. 2002. SQL. Jyväskylä: Docendo.

- Schengili-Roberts, Keith. Core CSS - Cascading Style Sheets, 2nd edition: Prentice Hall PTR.

- Veen, Jeffrey T. 2002. Web Design. New Riders, IT Press. Jyväskylä: Gummerus Kirjapaino Oy.

Web:

- 2K mediat.com 2000
  http://www.2kmediat.com/xhtml [online][viitattu 11.2005].

- autaystavaa.fi http://www.autaystavaa.fi/site/php/merkkijonottaulukot.html
  [online][viitattu 8.12.2005].

- College of San Mateo
  http://smccd.net/accounts/greenm/sessions.pdf [online][viitattu 8.12.2005].

- HTMLSource 2000
  http://www.yourhtmlsource.com/stylesheets/introduction.html
  [online][viitattu 8.12].

- Kollanus, Sami. Jyväskylän yliopisto - Tietojenkäsittelytieteidenlaitos.
  http://www.cs.jyu.fi/~kolli/ITK215_05/php/ [online][viitattu 8.12.2005].

- Kollanus, Sami. Jyväskylän yliopisto - Tietojenkäsittelytieteidenlaitos.
  http://www.cs.jyu.fi/~kolli/ITK215/PHP/sessiot.html
  [online][viitattu 8.12.2005].

- Järvinen, Petteri. Internet - muutostekijä.
  http://www.pjoy.fi/kirjat/imuutos/luku07b.html [online][viitattu 16.11.2005].

- Kippis.fi 2005
  http://www.kippis.fi [online][viitattu 24.11.2005].

- Ohjelmointiputka 2002
  http://www.ohjelmointiputka.net/opas.php?tunnus=phpj7
  [online][viitattu 8.12.2005].

- Open Options
  http://www.netc.org/openoptions/pros_cons/features.htmls
  [online][viitattu 17.11.2005].

- PHP.net 2001a
  http://fi2.php.net/manual/fi/history.php [online][viitattu 30.11.2005].

- PHP.net 2001b
  http://fi2.php.net/foreach [online][viitattu 8.12.2005].

- Saavutettava.fi
  http://saavutettava.fi/artikkelit/web-standardien-edut-kavijoille-asiakkaille-ja-sinulle [online][viitattu 17.11.2005].

- SitePoint 1998
  http://www.sitepoint.com/article/turn-lurkers-posters
  [online] [viitattu 28.11.2005].

- Sivut.web/a
  http://www.sivut.org/php/oppaat/muuttujat.php [online][viitattu 8.12.2005].

- Sivut.web/b
  http://www.sivut.org/php/vinkit/paivamaarat_luettavaan_muotoon.php
  [online][viitattu 8.12.2005].

- Timo Tiaisen Kotimaailma
  http://koti.mbnet.fi/timmie/cv/php_taulukot_oliot.html
  [online][viitattu 16.11.2005].

- tizag.com 2003
  http://www.tizag.com/phpT/include.php [online][viitattu 8.12.2005].

- Wikipedia-Vapaa tietosanakirja 2001a
  http://fi.wikipedia.org/wiki/PHP [online][viitattu 29.11.2005].

- Wikipedia-Vapaa tietosanakirja 2001b
  http://fi.wikipedia.org/wiki/MySQL [online][viitattu 8.12.2005].

- Wikipedia-Vapaa tietosanakirja 2001c
  http://fi.wikipedia.org/wiki/XHTML [online][viitattu 30.11.2005].