ZEND FRAMEWORK & GOOGLE MAPS

Case Mastonet

LAHDEN AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma
Ohjelmistotekniikka
Opinnäytetyö AMK
Jari Nietula
Kevät 2009

Lahden ammattikorkeakoulu
Tietotekniikan koulutusohjelma

NIETULA, JARI:

Zend Framework ja Google Maps
case: Mastonet

Tietotekniikan opinnäytetyö, 48 sivua, 1 liitesivu

Kevät 2009

TIIVISTELMÄ

Tämä opinnäytetyö käsittelee Lahden ammattikorkeakoululle toteutetun Mastonet sivuston toteutusta sekä Zend Framework:ä ja web-selain ohjelmointitekniikoita.

Teoria osuudessa kerrotaan Zend Frameworkistä ja yleisesti käytettävistä selaintekniikoista sekä Google Maps-sovelluksesta. Työn toteutus osuus koostuu Mastonet-projektille toteutettavasta internetsivustosta. Sivustolla esitetään langattomientukiasemien sijaintitietoja ja uutisia. Sivusto toteutetaan käyttäen PHP5- ja Ajax-tekniikoita. Sivuston pohjana käytetään Zend Framework:ä. Toteutukseen kuuluvat Zend Framework:n tutustuminen ja sivuston luominen frameworkin avulla. Lisäksi sivustolle suunnitellaan tietokanta. Tietokantaan tallennetaan Google Maps-sijaintitietoja sekä sivuston sisältötekstejä.

Sivusto toteutetaan osittain avoimena, jolloin sivuilla oleva tieto on kaikkien käytettävissä. Tukiasemien sijaintitietojen näyttämiseen käytetään Google Maps -sovellusta. Lisäksi sivuilla on oma hallinta osionsa, josta voidaan hallinnoida tukiasemien sijaintia ja tukiasemien tietoja, mutta myös sivuston sisältöä. Sivuston hallintaosiossa käyttäjät on jaettu kahteen tasoon. Sivuston hallinta osion käyttöliittymästä pyritään tekemään mahdollisimman käytettävä ja selkeä.

Avainsanat: PHP 5, Ajax, Google Maps, Mastonet, Zend Framework, Web- selainohjelmointi, CSS, JavaScript, JSON

Lahti University Faculty of technology

NIETULA, JARI:

Zend Framework and Google Maps
Case: Mastonet

Bachelor´s Thesis in Information Technology, 48 pages, 1 appendix

Spring 2009

ABSTRACT

This project deals with the Mastonet web site, Zend Framework and webbrowser programming techniques.

The aim of the theory part is to study Zend Framework, web browser techniques and Google Maps. The implementation part consists of the web site implemented for Lahti University of applied sciences. The web site consists of information about the Mastonet wireless network, shown with Google Maps.The web site was implemented using PHP5 and Ajax techniques. Zend Framework is used as the basis for the web site. Because the web site consists of dynamic information, that can be changed, a database for the system was designed and implemented fro the system.

The web site is partially open to all users, but it has a password protected section. In the admin section, authorised users can control information about the wireless network, news and general information shown on pages. Users are divided into two groups: Admins and Superadmins. The user interface of the admin section is designed so that it is easy to use and understand.

Key words: PHP 5, Ajax, Google Maps, Mastonet, Zend Framework, Webbrowser programming, CSS, JavaScript, JSON

SISÄLLYS

LYHENTEET

AJAX (Asynchronous Javascript and XML) joukko web-sovelluskehityksen tekniikoita, joiden avulla web-sovelluksista tehdään vuorovaikutteisempia.

PHP5 (PHP Hypertext Preprocessor) palvelimella suoritettava, tulkattava komentosarjakieli

HTML (HyperText Markup Language) World Wide Webin sivunkuvauskieli

XML (eXtensible Markup Language) merkintäkieli, jolla tiedon merkitys on kuvattavissa tiedon seassa.

CSS (Cascading Style Sheets) putousmallinen tyylimalli

APACHE (A Patchy Serve) Web-palvelinohjelmisto

DOM (Document Object Model)- malli, sivujen esittämiseen ja datan käsittelyyn

1 JOHDANTO

Sähköisten palveluiden siirtyessä yhä enemmän verkkoon käytetään web-selainta interaktiivisena käyttöliittymänä. Modernien web-selaimien avulla voidaan toteuttaa monimuotoisia ja interaktiivisia käyttöliittymiä, jotka vastaavat ominaisuuksiltaan jo melkein työpöytäsovelluksia.

Tässä opinnäytetyössä toteutetaan internetsivusto Lahden ammattikorkeakoululle osana Mastonet-projektia. Mastonet on Lahden kaupungin tarjoama langaton verkko. Teoriaosuudessa perehdytään Zend Frameworkin toimintaan sekä syvennetään tietämystä JavaScript:stä ja Ajax:sta. Työn tavoitteena on toteuttaa Mastonet-sivusto.

Mastonet-verkon hallinta siirtyi Lahden ammattikorkeakoululle vuoden 2009 alusta, ja tässä yhteydessä internetsivut on päätetty uusia. Sivustolla on Mastonettiin liittyviä uutisia ja tukiasemien sijaintitietoja. Tukiasemien näyttämiseen käytetään Google Maps -sovellusta. Yksittäisen tukiaseman sijainti kartalla määritellään, joko laskemalla sen koordinaattipiste tai käyttämällä kolmannen osapuolen tarjoamaa osoitteen muunnospalvelua. (Koordinaatio pisteenlaskemisesta, eli geokoodauksesta, lisää luvussa 3.1.)

Mastonet-sivustolla on oma hallintaosionsa, joka vaatii autentikoitumisen ennen käyttöä. Kaikki tukiasematiedot, uutiset ja muu sivuston niinsanottu dynaaminen tieto tallennetaan tietokantaan. Tukiasemien tiedot (uptime ja liikennemäärät) generoidaan käyttäen apuna RRDtool:ia.

Sivuston toteutukseen käytetään PHP5- ja AJAX-tekniikoita. Sivuston alustana toimii koulun tarjoama palvelinympäristö (UNIX) ja tietokanta (MYSQL). Työssä keskitytään PHP:hen ja natiiveihin selaintekniikoihin (HTML, JavaScript ja CSS). Työn visuaalisen ilmeen toteukseen on valittu ilmainen CSS-sivupohja. Työssä ei perehdytä ja tutkita syvällisesti palvelin- ja tietokantatekniikoihin.

2 INTERNETIN TILANNEKATSAUS

2.1 Yleistä

Internetin kehityksen alussa verkko oli täynnä staattisia HTML-dokumetteja. Internet verkon laajentuessa maailmanlaajuiseksi, sivustojen dynaamisuutta lisättiin. Dynaamisuutta saatiin aikaa palvelinohjelmoinnin avulla. Palvelinohjelmoinnissa web-selain lähettää http-kyselyitä palvelimelle prosessoitavaksi. Vastauksena palvelin palauttaa dynamisesti generoitua dataa tai kokonaisen HTML-sivun. Dynaamisesti generoitu HTML-tieto yleisesti tarkoittaa tietoa, joka on kerätty tietokannasta ja käsitelty palvelinohjelmoitikielellä.

Palvelinteknologiat mahdollistavat monimutkaisten web-sovellusten luomisen. Yleisimmin käytetty palvelinteknologia on PHP, mutta muita kilpailevia teknologioita ovat muun muassa ASP.NET ja JSP (Java Server Pages). Palvelimella voidaan hyödyntää tietokantoja, soveltaa olio-ohjelmoinnin periaatteita ja suorittaa vaativia laskutoimituksia. Tietokantapalvelin ohjelmistoja on useita, tässä lueteltuna kolme eri valmistajien vaihtoehtoa MySQL, Microsoft SQL server ja PostgreSQL. Eri tietokantojen välillä voi olla eroja muun muassa arkkitehtuurissa ja ominaisuuksissa.

HTML-sivujen käytettävyyttä ja interaktiivisuutta on vuosien mittaan parannettu selaimissa suoritettavilla teknologioilla. Tunnetuin ja eniten käytetty teknologia on JavaScript. JavaScript on skiptikieli, ja sitä käytetään pääsääntöisesti sovellusten toimintojen laajentamiseen. JavaScript suoritetaan aina selaimessa tai JavaScript-tulkin sisältämässä sovelluksessa. Nykyisin eniten käytetty JavaScript-tekniikka on sivujen interaktiivisuutta lisäävä AJAX (Asynkronin JavaScript ja XML). Muita vastaavia teknologioita ovat mm. Adobe Flash ja Java Appletit.

2.2 Hyvin toteutettu Internet-sivusto

Internet-sivustojen tärkein tehtävä on viestittää, mitä palveluja on tarjolla. Sivuston tulee tarjota olennainen tieto palveluista ja tuotteista. Suunnitteluvaiheessa tulee ottaa huomioon asiakaskunta, heidän tarpeensa ja toteuttaa niitä vastaava sisältö (WDS 2009). Suunniteltaessa internet-sivustoa tulee miettiä, mitä sivustolla tavoitellaan ja miten sivustosta saadaan persoonallinen ja kiinnostava, jotta se erottuisi massasta (Suominen 2002).

Sivuston ulkoasu kiinnittää kävijöiden huomion, ja visuaalisella näyttävyydellä kävijä saadaan kiinnostumaan ja tutustumaan sivuihin tarkemmin. Sisällön ja ulkoasun ammattimainen toteutus lisää uskottavuutta ja vakuuttaa sivustolla kävijän (WDS 2009). Hyvin toteutettu sivusto on selkeä ja yksinkertainen. Käytettävyyttä ja ymmärrettävyyttä voidaan lisätä opasteilla ja selityksillä, hakumahdollisuuksilla, sisältökartoilla ja sisällysluettelolla. (Suominen 2002.)

Sivuston latautuminen vie aikaa, jos sivuille on sijoitettu suuria kuvia tai useita sivuja tekstiä, jolloin käyttäjä saattaa kyllästyä odotteluun. Sivuston luomisessa tulisikin pyrkiä tiivistämään ja optimoimaan sisältöä. Tärkeää on päivittää sivuja säännöllisesti ja huolehtia tiedon ajankohtaisuudesta sekä linkkien toimivuudesta. (Suominen 2002.) Nykyaikainen internetsivusto on rakenettu käyttämään Ajax-tekniikoita. Ajax-tekniikka nopeuttaa käyttäjän saamaa palautetta sivuilla tehdyistä toimista, kun sivun uudelleenlataukset vähenevät.

3 TEKNIIKAT, KIELET JA ALUSTAT

3.1 Leveys ja pituus asteet

Maapallon on nimensä mukaisesti pallon muotoinen ja tarkan pisteen määrittelemiseksi maapallon pinnalla käytetään leveys- ja pituus asteita. Leveys- ja pituus asteet kuvaavat pisteen kulmaetäisyyttä pallon keskustasta. Jokainen pituus- ja leveys aste voidaan jakaa myös minuuteiksi ja sekunneiksi. Yhdessä asteessa on 60 minuuttia ja minuutissa 60 sekunttia. Tässä työssä käytetään desimaali ilmaisua pituus- ja leveys asteen kertomiseen. Leveys- ja pituus asteita käytetään Google Maps -sovelluksessa. Niiden avulla määritellään näytettävä karttaruutu. Koordinaatiopisteiden avulla määritellään käytettävässä Google Maps -sovelluksessa eri Mastonettukiasemien sijainnit. Laskentakaava desimaalikäännökselle on seuraavanlainen:

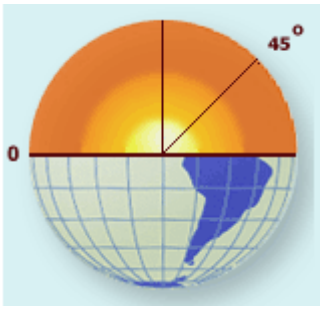
$$\left(\frac{\frac{s}{60} + m}{60} + d \right) * 1 \; tai - 1$$

KUVIO 1. Koordinaatiopisteen laskentakaava

Kuvion 1 kaavan kerroin riippuen siitä kummalla pallonpuoliskolla ollaan (pohjois- tai etelä). Laskentakaavassa ”d” kuvastaa astelukua, ”m” minuuttilukua ja ”s” sekuntilukua. Esimerkiksi 65 astetta 32 minuuttia ja 15 sekunttia pohjoista leveyttä on desimaaliarvona laskentakaavaa käyttäen (((15/60+32)/60+65)*1 = 65.5375 (Nationalatlas 2009).

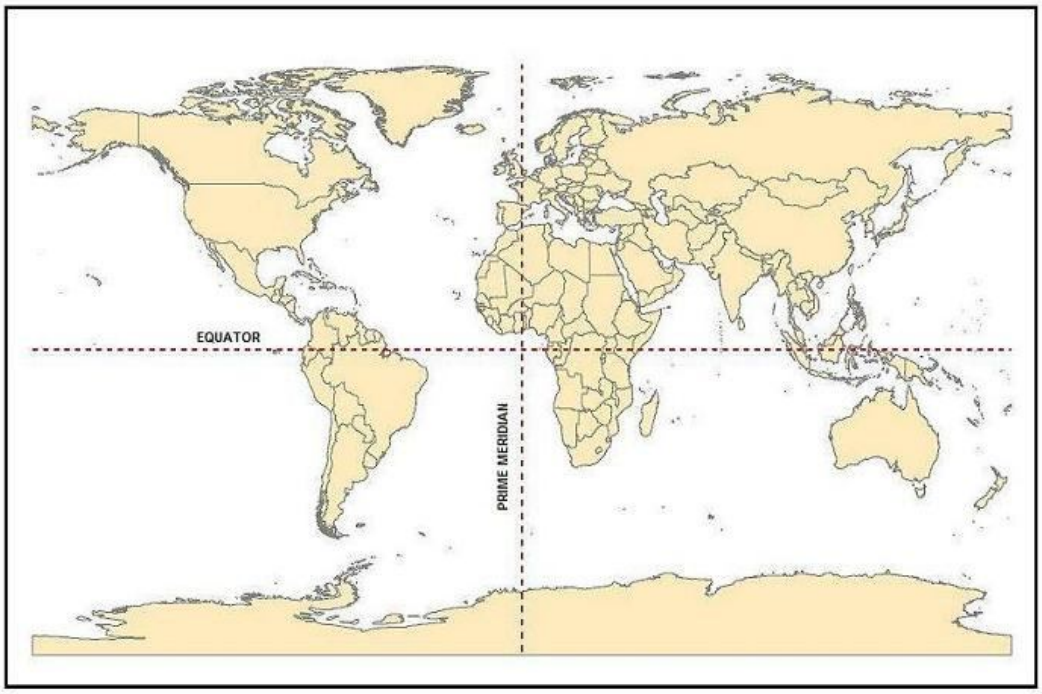
Leveysasteen keskilinja eli nolla aste linja on päiväntasaajalla. Leveysasteen luvut kertovat etäisyyden päiväntasaajaan, eli päiväntasaaja on leveysasteiden mittauksen lähtöpiste, kuten havainnolistetaan seuraavassa kuviossa (KUVIO 2). Leveysaste tarkoittaa pisteen kautta kulkevan ellipsoidipinnan vektorin ja päiväntasaajan välistä kulmaa. Leveys aste voidaan kertoa kahdella eri tavalla geodeettisenä tai geosentrisenä. Näiden kerrontatapojen välinen poikkeama on suurimmillaan 11 kaari-

minuuttia (1 kaariminuutti on asteen 60 osa). Päiväntasaajan etelä- ja pohjoispuolella leveys asteilla on eri etumerkit. Pohjoiset asteet ilmoitetaan positiivisina ja eteläiset negatiivisina. Positiiviset ja negatiiviset arvot ovat käytössä vain, kun asteluvun yhteydessä ei käytetä ilmansuunta määrettä. Molemmat maapallon navat (Pohjois- ja Etelänapa) sijaitsevat 90 asteen kulmassa päiväntasaajaan nähden. (Nationalatlas 2009.)



KUVIO 2. Leveysasteiden jakautuminen (Nationalatlas 2009)

Pituusasteilla ei ole yhtä selvää 0-linjaa kuin leveysasteilla. Kansainvälisen sopimuksen mukaan 0-linja on määritelty kulkevaksi Greewich:n kautta. Nollalinjan sijainti havainnollistetaan seuraavassa kuviossa (KUVIO 3). Pituusasteet määritellään itään tai länteen nollalinjasta. Maapallon on jaettu 360 pituusasteeseen. Nollalinjan itäpuolella on 180 astetta, ja niille on määrätty positiivinen arvo. Samalla tavalla nollalinjan länsipuolella on 180 pituusastetta, ja niillä on negatiivinen arvo. Positiiviset ja negatiiviset arvot ovat käytössä vain, kun asteluvun yhteydessä ei käytetä ilmansuunta määrettä. Pituusaste kertoo pisteen sijaintipaikan ja nollameridiaanin välisen kulmaeron. (Nationalatlas 2009)



KUVIO 3. Maapallon jakautuminen pituusasteisiin

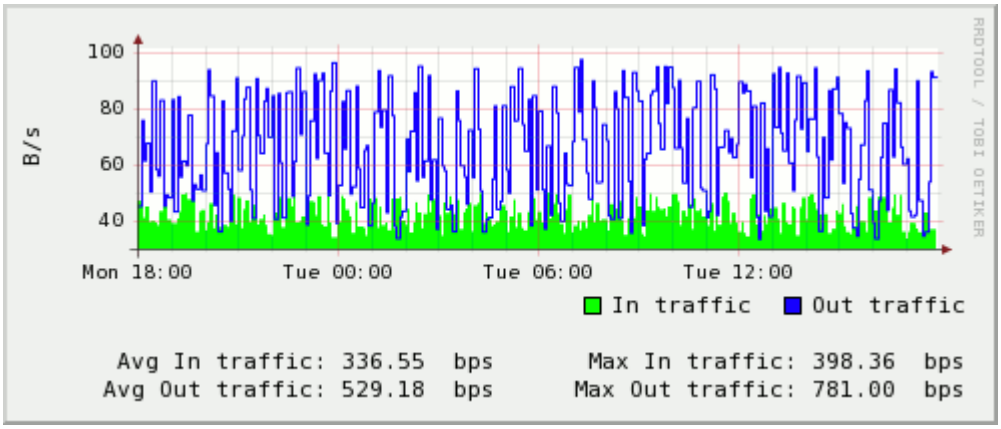
3.2 RRDtool

RRDtool on lyhenne sanoista ”Round Robin Database tool”. RRDtool on vapaan-lähdekoodin datan kirjaamiseen ja analysointiin keskittynyt ohjelma. RRDtool:n avulla on mahdollista nopeasti generoida graafisia esityksiä tallennetusta datasta (RRDtool 2009). Tässä opinnäytetyössä sitä käytetään tukiasema tietojen käsittelyyn. Tukiasemilta haetaan tietoa liikennemääristä ja käytettävyysajasta. Nämä tiedot käsitellään php komennoilla. Komentoja havainnollistetaan seuraavassa kuviossa (KUVIO 4). Lopputuloksena generoidaan graafi liikennemääristä, joka sitten voidaan liittää sivuille.

```
1  <?php
2  $opts = array( "-start", "-1d", "-vertical-label=B/s",
3                 "DEF:inoctets=net.rrd:input:AVERAGE",
4                 "DEF:outoctets=net.rrd:output:AVERAGE",
5                 "AREA:inoctets#00FF00:In traffic",
6                 "LINE1:outoctets#0000FF:Out traffic\\r",
7                 "CDEF:inbits=inoctets,8,*",
8                 "CDEF:outbits=outoctets,8,*",
9                 "COMMENT:\\n",
10                "GPRINT:inbits:AVERAGE:Avg In traffic\: %6.2lf %Sbps",
11                "COMMENT:  ",
12                "GPRINT:inbits:MAX:Max In traffic\: %6.2lf %Sbps\\r",
13                "GPRINT:outbits:AVERAGE:Avg Out traffic\: %6.2lf %Sbps",
14                "COMMENT: ",
15                "GPRINT:outbits:MAX:Max Out traffic\: %6.2lf %Sbps\\r"
16              );
17
18   $ret = rrd_graph("net_1d.gif", $opts, count($opts));
19
20   if( !is_array($ret) )
21   {
22     $err = rrd_error();
23     echo "rrd_graph() ERROR: $err\n";
24   }
25 ?>
```

KUVIO 4. PHP koodi esimerkki RRD graafin luontiin

Kuviossa 4 määritellään riveillä 2 - 16 taulukko, joka sisältää graafin generointiin käytettäviä parametrit. Lisäksi rivillä 2 määritellään graafin piirtoon käytettävä aika jakso ”-1d”, eli piirretään graafi yhden päivän keskiarvo liikennemääristä. Riveillä 3 ja 4 määritellään käytettävät keskiarvot, jotka otetaan ”net.rrd” tiedostosta. Riveillä 5 – 15 määritellän kuvioon tulevat kommentit ja käytettävät viivan värit. Rivillä 18 on php komento, jolla itse graafi luodaan. Komennolla välitetään parametreina kuvan nimi, määritellyt parametrit ja niiden lukumäärä. Seuraavassa kuviossa (KUVIO 5) havainnollistetaan yhden vuorokauden liikennedatasta generoitua graafia.



KUVIO 5. Esimerkki graafi liikennemääristä RRDtool:lla generoituna

Kuviosta 5 voidaan havaita sisäänpäin tulevan liikenteen määrät vihreällä merkittynä ja ulospäin suuntautuneen liikenteen määrä sinisellä merkittynä. Lisäksi kuvassa ilmoitetaan liikennemäärät myös tekstimuotoisena.

3.3 CSS

HTML:n alkuaikoina tyylimäärittelyillä ei ollut minkäälaisia ohjesääntöjä. Kaikki tyylimäärittelyt kirjoitettiin HTML- elementti määritysten sekaan atribuutteina. Vuoden 1995 loppupuolella pidetyssä WWW Consortium –kokouksessa päätettiin kehittää tyylisivut, joiden avulla määritellään asiakirjan ulkoasu. Ensimmäiset nykyaikaiset CSS- tyylisivut julkaistiin vuonna 1996. CSS eli Cascading Style Sheet on tyylimalliltaan putousmallinen. CSS-mallissa useampi tyylisivu voi toimia yhdessä muodostaen lopullisen asiakirjan ulkoasun. Tyyli määrittelee, miltä HTML -elementti näyttää, ja siinä otetaan huomioon edeltävät määrittelyt. CSS tukee suhteellisia ja absoluuttisia mittoja, joiden avulla sivun ulkoasu voidaan määritellä tarkasti. Tyylimallien avulla on mahdollista sovittaa HTML-sivu näkymään samalla tavalla niin internetselaimessa kuin mobiiliselaimessakin. Muita mainittavia etuja tyylisivujen käytöstä ovat:

- asiakirjan ulkoasun helppo muokattavuus
- saman tyylisivun käyttö useassa asiakirjassa
- sisällön helpompi ylläpidettävyys
- asiakirjojen nopeampi laadittavuus.

(Boumphrey & Greer & D. Raggett & J. Raggett & Schnitzenbaumer & Qugofski 2000. 234 – 238.)

Seuraavassa kuviossa (KUVIO 6) muutetaan HTML <a> tagin ominaisuuksia hiiren päälleviennin osalta.

```
1  <style type="text/css">
2  <!--
3  a:hover {
4      font: 10px;
5      font-family: Verdana, Arila, sans-serif;
6      color: green;
7      background: transparent;
8      }
9  -->
10 </style>
```

KUVIO 6. <a> tagin CSS- tyylimäärittely

Kuviossa 6 rivillä määritellään käytettävän fontin koko, kirjasinperhe, väri ja taustantyyli. Nämä uudet tyylimäärittelyt (hover) tulevat voimaan, kun hiiri viedään HTML-linkkitagin päälle. Muutokset poistuvat näkyvistä, kun hiiri viedään pois linkin päältä, jolloin linkki palaa alkuperäisen näköiseksi.

Jos CSS- tyylimäärittely kirjoitetaan HTML- koodin yhteyteen alkaa, se aina <style>- tagilla, jota seuraa kommenttimerkit. Samoin tyylimäärittely päättyy aina </style> merkkiin. Ulkoisissa tyylisivuissa edellä mainittujen tageja ei tarvitse käyttää.

Selaimen tulkitessa asiakirjaa, se oletusarvoisesti sijoittaa tekstin ja kuvat omille paikoilleen käyttäen sisäistä tyylisivuaan. Tyylisivuja käytetään kertomaan selaimella yksityiskohtaisesti, mihin kohtaan ja minkä näköisinä elementit sijoitetaan näytöllä tai paperilla. (Boumphrey & Greer & D. Raggett & J. Raggett & Schnitzenbaumer & Qugofski 2000. 74 – 80.)

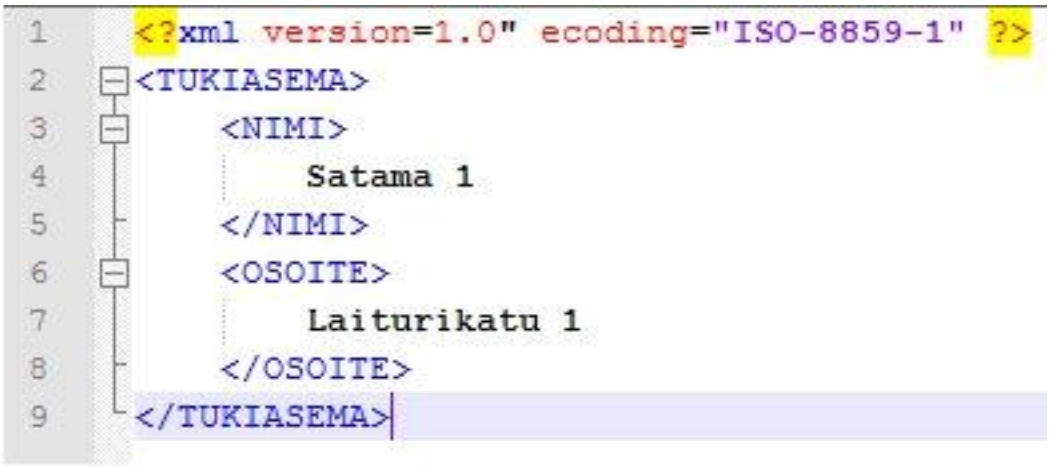
Tyylisivujen prioriiteetin päättelemiseksi on olemassa 4 yleistä sääntöä

- selaimen oletus
- ulkoiset tyylisivut
- sisäiset tyylisivut
- HTML- määrityksiin sisälletty (Inline)

Edellä luetellun listan mukaan aina viimeiseksi määritelty tyyli tulee voimaan, kun asiakirjaa tulkitaan. Tämän tulkinnan mukaisesti, jos asiakirjaan on linkitetty useita ulkoisia tyylisivuja, viimeisenä linkitetty tyylisivu on korkeimmalla prioriteetilla. Tähän kaikkeen on kuitenkin olemassa poikkeus. Poikkeusmäärittely ”!important”, kumoaa edellisen listan esittämän tärkeysjärkestyksen. Poikkeuksella on kuitenkin poikkeus, jos samalle elementille määritellään ”!important” useannab kerran samalla tyylisivulla, kumoaa se kaikki tärkeysmäärittelyt. Tällaisessa tapauksessa järjestys palautuu edellä mainitun listan mukaiseksi (CSS 2009).

3.4 XML

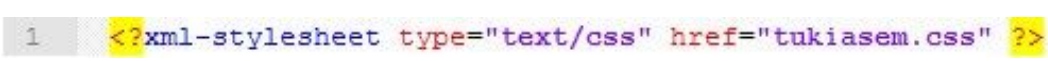
XML (eXtensible Markup Language) on W3C:n kehittämä rakenteellinen kuvauskieli. Sillä kuvataan tiedon rakennetta. XML käsitellään aina selaimen toimesta puhtaana tekstinä. XML-asiakirja koostuu tageista. Tagien nimille ei ole mitää pakottavia määreitä kuten HTML kielessä vaan käyttäjä voi vapaasti valita nimet. Aloittavan ja lopettavan tagin tulee olla samanlaisia (havainnollistettuna kuviossa 7), paitsi että lopettavan tagin ensimmäisen merkin on oltava kauttaviiva. Seuraavassa kuviossa (KUVIO 7) esimerkki XML- dokumentista. (Boumphrey & Greer & D. Raggett & J. Raggett & Schnitzenbaumer & Qugofski 2000. 277- 285.)



KUVIO 7. XML-rakenne-esimerkki

Kuviossa 7 määritellään uusi tukiasema tietue, jolla on kaksi parametria nimi ja osoite.

XML-dokumentti alkaa aina prosessointikäskyllä, ja siinä määritellään käytettävä versio ja merkkikoodaus, kuten kuviossa 7 rivillä 1. Tunnisteille voidaan valita mielivaltaisia nimiä, kunhan se alkaa kirjaimella tai alaviivalla, ja seuraavat merkit ovat kirjaimia, lukuja, alaviivoja, yhdysmerkkejä tai pisteitä. Tyhjämerkki on aino jota ei hyväksytä. Edellä olevassa kuviossa on esitetty yhden langattoman verkon tukiaseman perustiedot. XML- dokumenteille voidaan määritellä tyylejä samalla tavalla kuin perinteisille HTML- asiakirjoille. Tyylin määrittelyyn on kaksi päätapaa: CSS-tyylitiedostot, jotka esiteltiin aiemmin ja XLS (Extensible Style Sheet Language). Css-tyylitiedosto liitetään XML- dokumenttiin komennolla, jota havainnollisteaan seuraavassa kuviossa (KUVIO 8).



KUVIO 8 XML- tyylisivun liittämiseen käytettävä lause
(Holzner. 2001 19 – 24)

XSL-tyylistiedostot on tehokkaampi tapa määritellä XML dokumentin tyyli, sillä se on itsessäänkin XML-dokumentti. XML- dokumenttia tulkitessa selain tunnistaa

XSL- tiedostossa määritellyn elementin ja soveltaa siihen XSL- tiedostossa määriteltyjä sääntöjä. XSL:n avulla voidan generoida normaaleja HTML-asiakirjoja, jolloin selaimen ei tarvitse osata käsitellä XML-tiedostoja. XSL:n avulla voidaan myös järjestää, muuttaa, näyttää osin tai jopa piilottaa dokumentin elementtejä (Holzner. 2001 46). XSL-prosessoinnin avulla samaa XML-dokumenttia voidaa katsella käyttäen montaa eri päätelaittetta. XSL koostuu kolmesta osasta XSLT, XPath ja XSL-FO. XSLT:stä ja XPath:sta kerrotaan tarkemmin seuraavissa kappaleissa. XSL-FO on XML- dokumenttien muotoiluun tarkoitettu kieli. Siitä voidaan käyttää myös lyhyempää nimeä eli XSL. XSL-FO tyyleillä määritellään, miltä XSL prosessin läpi käynyt tuotos näyttää. (XSL-FO 2009.)

3.5 XSLT & XPath

XSLT ei ole varsinainen tyylisivukieli. Se on kieli, jolla voidaan muuntaa ja generoida elementtejä ja atribuutteja. Varsinaisesti XSLT:tä käytetään XML- dokumnettien muuntamiseen selainluettaviksi. XSLT muuntaa XML-elementin XHTML elementiksi. XSLT on tarkoitettu palvelimella ajettavaksi. XSLT käyttää XPath-avukseen eri elementtien ja atribuuttien etsimiseen XML dokumentista. (XSL 2009.)

XPath on syntaksi, jonka avulla määritellään XML elementtejä. Se käyttää polkumääritelmiä eri XML tietueiden tai tietue ryhmien valintaan. XPath:ssa on yli 100 funktiota erityyppisten tietuetyyppien käsittelyyn. Xpath käsitteleen XML dokumenttia tietuepuuna. XPath tunnistaa seitsemän eri tietuetyyppiä:

- elementti
- atribuutti
- teksti
- nimiavaruus
- kommentti
- dokumentin juuri
- prosessointi käsky

(XPath 2009).

3.6 PHP5

PHP on dynaamisten internetsivujen luontiin käytettävä C- ohjelmointikielen kaltainen komentosarjakieli. Komentosarjakieli tarkoittaa, että koodi tulkitaan vasta, kun palvelin suorittaa koodin. PHP ei ole alusta tai käyttöjärjestelmä riippuvainen ohjelmointikieli. (Rantala, 2005, 9.)

PHP:n kehitys alkoi vuonna 1995, kun tanskalainen Rasmus Lerdorf kirjoitti kokoelman C-kielisä CGI-skriptejä. Vuosien saatossa PHP:n suosio kasvoi, ja siitä julkaistiin uusia versioita. Nykyinen versio (5.2.9) on julkaistu helmikuussa 2009. Uusi versio on suunnitteilla, ja julkaisupäiväksi on kaavailtu kuluvan vuoden toista neljännestä. Uudessa versiossa pitäisi olla mukava seuraavat uudistukset: tuki pysyville tietokantayhteyksille, rajoitettu goto, rosklien keräys ja tuki nimiavaruuksille. (Wikipedia, php.)

PHP5 eroaa edeltäjistään muunmuassa sillä, että se tukee olio-ohjelmointia. Tähän versioon on lisätty esimerkiksi julkisten/yksityisten muuttujien ja funktioiden käsittely, abstraktit luokat, periyttäminen, poikkeuksien käsittely ja funktiot. Keskeisin osa tätä PHP-kehitysympäristöä on Zend Engine. (Rantala, 2005, 14.)

```
34        //generating navigation
35        $menu = new NaviList();
36        $this->view->naviName = "/index";
37        $this->view->naviList = $menu->menuAction($this->lang);
38
```

KUVIO 9. PHP:n olion luonti ja olion funktion kutsu

PHP5:ssä jokaiselle luokalle tarvitsee määritellä _construct-funktio, jotta siitä voidaan luoda instanssi käyttöä varten (Babin, Good, Kromann, Stephens. 2005 24, 27). Edellä olevassa kuviossa (KUVIO 9) luodaan uusi navikaatio-olio ja kutsutaan

sen *menuAction* funktiota. Esimerkissä kutsuttu funktio ottaa vastaan parametrina kielimuuttujan arvon ja hakee sen perusteella navikaation tietokannasta. Tämän valinnan avulla navikaation kieli on joko suomi tai englanti. Kutsuttu funktio palauttaa näkymämuuttuja *naviList*:lle navigaatio rakenteen.

3.7 JavaScript

Javascript on tulkattava sciprtikieli. JavaScript-tulkin voi upottaa mihin tahansa sovellukseen, kuten esimerkiksi web-selaimeen. Kun tulkki käy läpi JavaScript–koodia, se suoritetaan rivi riviltä. Käännettyyn kieleen verrattuna JavaScriptiä on huomattavasti helpompi muokata, ja tehdyt muutokset astuvat heti voimaan, kun dokumentti ladataan uudestaan. JavaScriptissä merkkijonot, numerot ja funktiot ovat olioita ja JavaScript muistuttaa syntaksiltaan C-perheen ohjelmointikieliä. JavaScriptin suurimpia vahvuuksia on sen mahdollistama funktionaalinen ohjelmointityyli. Funktioita voidaan tallentaa muuttujiin, välittää parametreina ja palauttaa arvoina. Javascript käyttää prototyyppien perintämekanismia, jolloin jokaisella oliolla on mahdollisuus periä ominaisuuksia toisilta olioilta. Jokaisella funktiolla on prototyyppi. Prototyyppi on kenttä, mihin voidaan määritellä uusia olioita eli ilmentymiä. Uusia olioita luokasta luodaan kutsumalla *construct* funktiota *new* komennolla. Uusien ilmentymien luontiin voidaan käyttää myös aaltosulkeita. (Wikipedia, JavaScript.)

JavaScript ohjelmoinnin keskeisin osa on tapahtumankäsittely. Tapahtumankäsittelijä funktioita kutsutaan, kun käyttäjä tai selain aiheuttaa jonkin määritellyn tapahtuman. Tapahtumankäsittelijöitä voi myös käyttää tapahtumakahvoina. Tapahtumakahvat ovat skriptejä, jotka käsittelevät objekteja. Suurin osa JavaScript tapahtumista on käyttäjälähtöisä, mutta ne voivat olla myös muotoa ” kun sivu ladataan”. Seuraavassa on lueteltuna JavaScriptin 12 yleisintä tapahtumaa ja niiden selitykset:

- onabort (sivun lautuksen keskeytys)
- onblur (oliosta poistutaan)
- onchange (olion tilaa muutetaan)

- onclick (oliota painetaan)
- onerror (virhe suorituksessa)
- onfocus (olio aktivoidaan)
- onload (olion lataaminen)
- onmouseover (kursori on olion päällä)
- onmouseout (kursori siirtyy pois olion päältä)
- onselect (olion valinta)
- onsubmit (lomakkeen lähetys)
- onunload (sivuilta poistuminen)

(Moncur 2000.)

3.8 DOM

W3C julkaisi vuonna 2000 ensimmäisen määrityksen siitä, kuinka selainten tulisi tulkita ja käsitellä dokumenttien oliomalleja. Document Object Model on alusta ja ohjelmointikieli vapaa rajapinta. DOM-mallissa internetsivu jaetaan olioihin, joiden avulla sivua on helpompi muokata ja käsitellä. Olioden ominaisuuksien ja metodien avulla on mahdollista muokata sivua sen latauksen aikana. Sen avulla voi lisätä, muokata tai jopa poistaa elementtejä sivulta. Olioiden ominaisuudet ja metodit vaikuttavat sivun sisällön, rakenteen ja tyylin muokkaamiseen. DOM on rajapinta sovelluskehittäjille. Se määrittää tarkasti, kuinka asiakirja esitetään ja kuinka sitä voi muokata. (Boumphrey & Greer & D. Raggett & J. Raggett & Schnitzenbaumer & Qugofski 2000, 537 - 538)

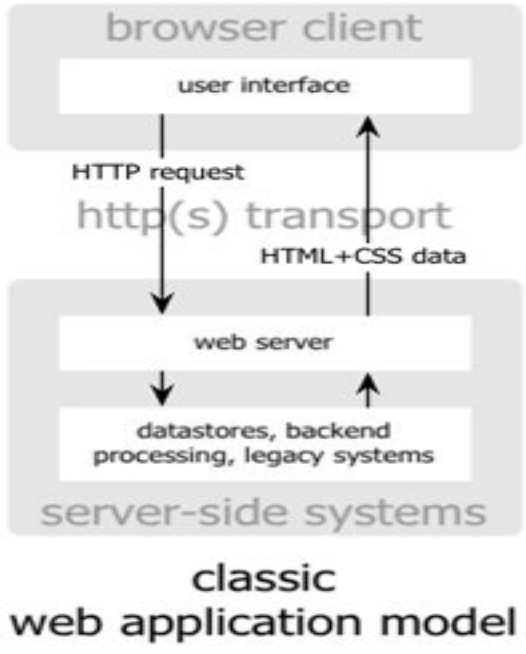
3.9 Ajax

Ajax:n kehitti James Garrett vuonna 2005. Tarkalleen ottaen Ajax on osa JavaScriptiä. Ajax (Asynchronous JavaScript and XML) on kokoelma tekniikoita.

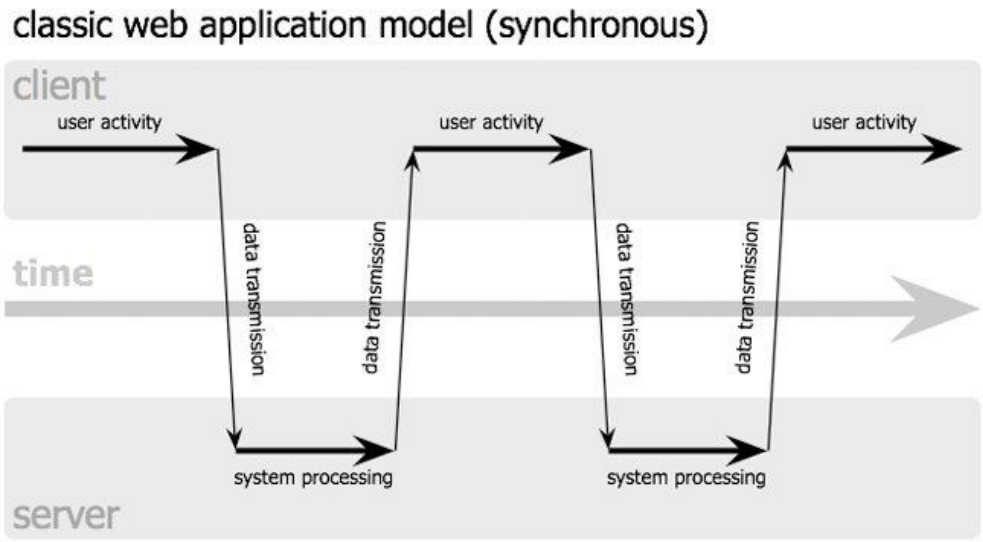
XHTML-merkkauskieltä, CSS-tyylisivuja, DOM: ia, XML:ää ja XMLHttpRequest kutsuja. Kahta ensimmäiseksi mainittua käytetään dokumentin kuvailemiseen, DOM:ia dynaamiseen tietojen näyttämiseen ja vuorovaikutukseen datan kanssa, XML:ää käytettän datan käsittelyyn ja viimeisenä mainittua itse datan asynkroniseen siirtoon. (Wikipedia, Ajax, 2009.)

Ajax:n käytön suurimpia etuja on turhien sivulatausten poistuminen. Ajax-sovelluksessa vain muuttunut data tarvitsee ladata uudelleen. Näin liikenteen määrä palvelimelle vähenee. Asynkroninen tiedonsiirto mahdollistaa nopean reagoimisen käyttäjän kehotteeseen ja tekee palvelusta interaktiivisen. (Wikipedia, Ajax, 2009.)

AJAX:n käytön haittapuolia ovat seuraavat: selaimen takaisin -nappia ei voi käyttää eikä sivua voi lisätä ”kirjanmerkiksi”. Sivun sisältö on generoitu käyttäen asynkronista tiedonsiirtoa, jolloin ei ole tapahtunut uutta sivun latausta. Seuraavissa kuvioissa (Kuvio 10 ja 11) kuvataan perinteistä sivunhaku mallia (Wikipedia, Ajax, 2009.)

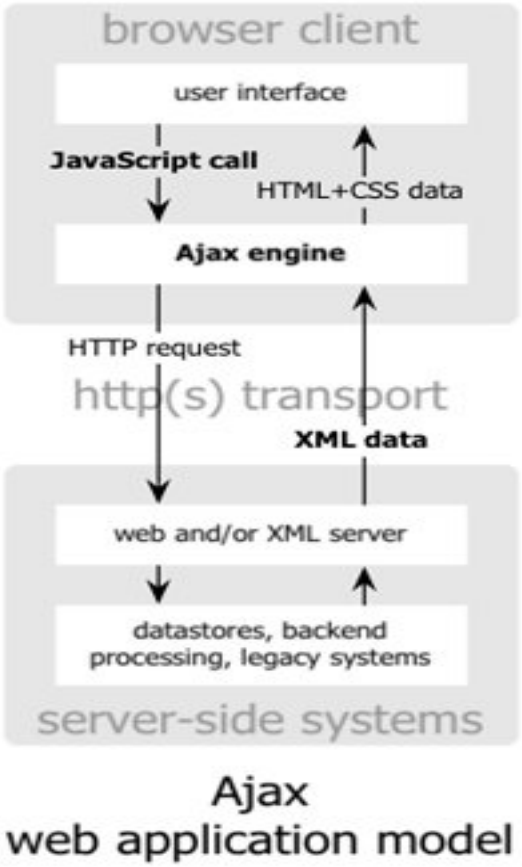


KUVIO 10. Perinteinen sivunhakumalli



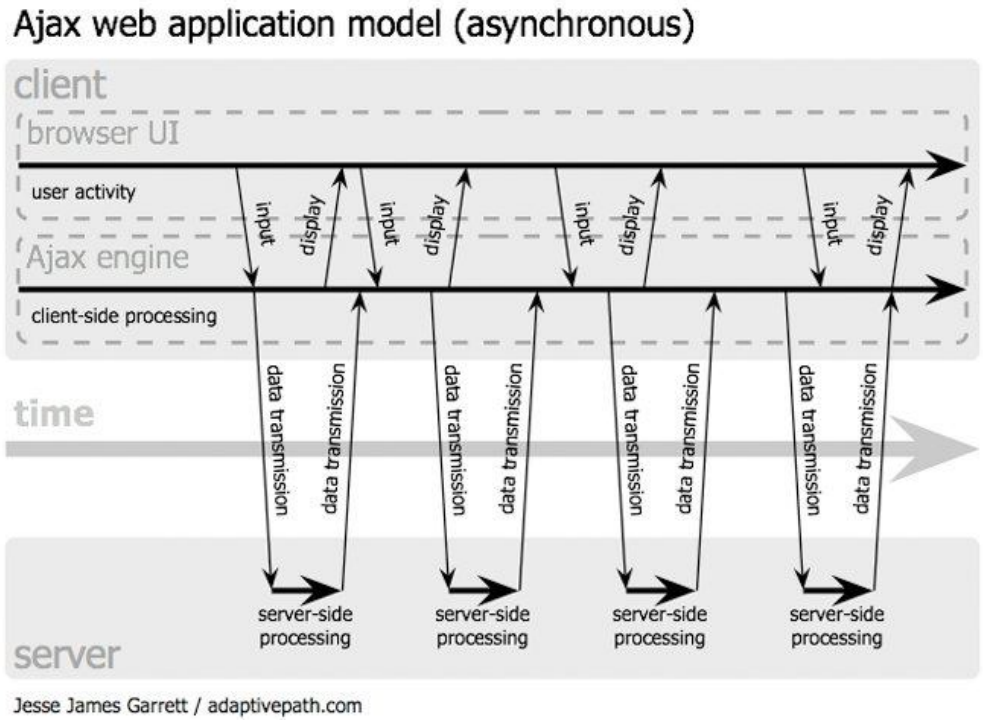
KUVIO 11. Aikaan verrannollisesti kuvattu perinteinen sivunlataus

Kuvioista 10 ja 11 voidaan huomata kuinka, ennen vanhaan tapahtuneessa sivu latauksessa selain kommunikoi suoraan palvelimen kanssa. Tässä sivun latausmallissa käyttäjä saattoi joutua odottamaan pitkiä aikoja vastausta palvelimelta. Tätä ajan kulumista on tarkemmin kuvatta kuviossa 11.



KUVIO 12. Ajax sivunhakumalli

Kuviossa 12 havainnollistetaan Ajaxia apuna käyttäen toteutetun sivuston sivulataus. Ajax sivustolla selain kommunikoi Ajax-moottorin kautta palvelimelle. Seuraavassa kuvios (KUVIO 13) havainnollistetaan ajan kulumista, kun selataan Ajaxin avulla toteutettua sivua.



KUVIO 13. Aikaan verrannollisesti kuvattu Ajax sivunlataus

Kuviossa 13 käyttäjällä pysyy kontrolli seurattavaan sivuun, koska sivun lataukset pysyvät taustalla. Tällöin käyttäjän ei tarvitse odotella pitkiä aikoja.

Ajaxin avulla sivuille voidaan toteuttaa kehittyneimpiä käyttöliittymäkomponentteja, kuten drag-and-drop ja puu-navit. AJAX:n avulla luodaan rikkaita internet sovelluksia, joiden tarkoituksena on toimia enemmän työpöytäsovellusten tapaan. (Eichorn 2006.)

Sivun ensimmäinen lataus kestää pidempään kuin HTML-sivun lataus, jolloin ladataan koko javascript-moottori selaimeen. Ladattu moottori säilyy selaimen muistissa koko istunnon ajan. Tämän jälkeen sivuston kommunikointi palvelimen kanssa on kuitenkin huomattavasti nopeampaa ja tehokkaampaa. Ajax sovelluksen nopeaa ja tehokasta kommunikointia on kuvattu edellä olevassa kuviossa (KUVIO 13). Tietoa selaimelle voidaan lähettää muunmuassa JSON (JavaScript Object Notation) avulla tai vaihtoehtoisesti puhtaasti tekstinä tai XML:nä.

3.10 JSON

JSON (Javascript Object Notation) on yksinkertainen merkintäkieli tiedonsiirtoon. Se on tekstipohjainen, selkokielinen ja sisältää olioita. JSON on ohjelmointikieli riippumaton, mutta se noudattaa C-kielisiä käytäntöjä. JSON:ia voidaan käyttää muun muassa JavaScripti-tiedonsiirtoon. JSON-syntaksi koostuu kahdesta eri vaihtoehdosta: Ensimmäinen vaihtoehto on avain-arvo parit, joita voi verrata assosatiiviseen taulukkoon tai tietorakenteeseen. Toinen vaihtoehto on järjestetty lista, jota voi verrata vektoriin. Seuraavassa esimerkkikuviossa (KUVIO 14) selvennetään JSON-syntaksia. Perusmuuttujatyyppejä ovat numerot, boolean, merkkijono, taulukko, objekti tai null- arvo. (WIKIPEDIA, JSON, 2009.)

```
1  {
2      "Name": "Satama 1",
3      "address": {
4          "streetAddress": "Laiturikatu 1",
5          "postalCode": "15100",
6          "city": "Lahti"
7          },
8      "Transfer": [
9          "1 212",
10         "6 460"
11         ]
12 }
```

KUVIO 14. JSON objekti avain-arvo-pareina

Kuviossa 14 kuvaillaan yhtä tukiasema objektia. Objektilla on merkkijonomuotoinen nimikenttä, taulukkomuotoinen siirtomääräkenttä ja objekti, jolla kuvataan tukiaseman osoite.

JSON ei vaadi erillistä tulkintaprosessia, sillä se on jo itsessään syntaksiltaan JavaScriptiä. Koodi voidaan muuttaa käytettäväksi tietorakenteeksi käyttämällä JavaScriptin eval-funktiota. Eval on JavaScriptin sisäänrakennettu funktio, joka ottaa

vastaan parametrina merkkijonon. Funktio laskee tai suorittaa sen JavaScriptinä. Sen avulla merkkijono muunnetaan JavaScript objektiksi. JavaScriptin eval- funktion käyttö tuo mukanaan tietoturvaongelmia. Koska funktio ei tarkista saamaansa koodia millään tavalla, ainoastaan suorittaa sen, on sen avulla mahdollista suorittaa haitallista JavaScript-koodia sivustolla. Eval-funktion käyttö ei ole suositeltavaa, vaan paremman turvallisuuden kannalta tulisi käyttää parseJSON()- funktiota. (WIKIPEDIA, JSON, 2009.)

3.11 XMLHttpRequest

XMLHttpRequest on alun perin Microsoftin kehittämä geneerinen HTTP-asiakasolio JavaScriptille. JavaScript voi suorittaa GET- ja POST HTTP-pyyntöjä käyttäen hyväkseen XMLHttpRequest:ia. XMLHttpRequest käytöllä on muutamia turvallisuus tekijöistä johtuvia rajoituksia. Se voi suorittaa HTTP kutsuja vain siihen domainiin, missä se on ladattu. Vaikka XMLHttpRequest oliolla on vain muutama funktio, sen käytössä pitää tarkistaa, mitä selainta käyttäjä käyttää. Selaimen tarkistus pitää suorittaa siksi, että Microsoftin selaimet käsittelvät JavaScriptiä eritavalla. Microsoftin selaimissa XMLHttpRequest oliota käsitellään ActiveX objektina, kun taas modernit selaimet käsittelevät XMLHttpRequest- oliota natiivisti. (Eichorn 2006.)

Seuraavassa on lueteltuina yleisimmin käytetyt XMLHttpRequest- objektin funktiot:

- abort (pyynnön keskeytys)
- getAllResponseHeaders (palauttaa ”täydellisen” otsikon)
- getResponseHeader (palauttaa pyydetyn otsikon)
- open (yhteyden avaus)
- send (lähettää pyynnön)
- setResquestHeaders (asettaa avain arvo parin lähetettävään pyyntöön).

Näitä funktioita tukevat kaikki uudemmat selaimet. Luettelosta käytetyimmät metodit ovat open() ja send(). Kun objekti on luotu onnistuneesti, siltä voi kysellä seuraavia ominaisuuksia: onreadystatechange, readyState, responseText, responseXml, status ja statusText. Ensimmäisenä mainittu onreadystatechange on tapahtumakahva, joka käynnistää muutoksen. ReadyState on integer arvo väliltä 0-4. Se kertoo objektin statuksen seuraavasti:

- 0 (ei alustettu)
- 1 (ladataan)
- 2 (ladattu)
- 3 (interaktiivinen)
- 4 (valmis)

ResponseText on tekstimuotoinen vastaus palvelimelta ja vastaavasti responseXml on dokumentti objekti. Status viesti kertoo selkeästi internetliikenteen yleisillä koodeilla kyselyn tilan. Vastaus voi esimerkikisi olla 200 ”OK” tai 404 ”Ei löydy”. StatusText kertoo tekstinä saman kuin edellinen numeroina. (XMLHttpRequest 2009.)

3.12 Google Maps

Google Maps on ilmainen internetin karttasovellus. Ensimmäinen versio siitä julkaistiin vuonna 2005. Sovelluksen käyttö edellyttä rekisteröitymistä, käyttösivuston internet-osoitteen tietämistä ja ”Google Maps API”- avaimen anomista. (Purvis, Sambells & Turner 2006, 13)

Google Maps:n asennus ja käyttöönotto on tehty helpoksi. Rekisteröinnin yhteydessä saa valmiin yksinkertaisen JavaScript-koodin sähköpostiin. Tässä koodissa on oma Google Maps-avaimesi sulautettuna. Sen käyttöönotto on yksinkertaista: kopioit saamasi koodin HTML-sivulle. Lisäominaisuuksien implementoimista varten Google tarjoaa selkeän ja helppolukuisen sivuston, jolta löytyvät ohjeet ja esimerkit. (Purvis, Sambells & Turner 2006, 14).

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtm'
11/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:v="urn:schemas-microsoft-com:vml">
  <head>
    <meta http-equiv="content-type" content="text/html; charset=utf-8"/>
    <title>Google Maps API Sample</title>
    <script src="http://maps.google.com/maps?file=api&amp;v=2&amp;sensor=false&amp;
    key=ABQIAAAAyAp3rafBeQ0pOEvzb1K7RhQBQsUirap_pBj9RC5aWETVy9P1QRQyYB-MZMUGjPkEb">
    </script>
    <script type="text/javascript">

    function initialize() {
      if (GBrowserIsCompatible()) {
        var map = new GMap2(document.getElementById("map_canvas"));
        map.addControl(new GSmallMapControl());
        var location = new GLatLng(60.98160191607567, 25.645179748535156);
        map.setCenter(location, 12);

      }
    }

    </script>
  </head>
  <body onload="initialize()" onunload="GUnload()"
  style="font-family: Arial;border: 0 none;">

    <div id="map_canvas" style="width: 500px; height: 500px">
    </div>
  </body>
</html>
```

KUVIO 15. Perus Google maps sovelluksen koodi

Esimerkkikuviosta (KUVIO 15) voidaan havaita, kuinka yksinkertaisimmillaan Google Maps implementaatio vaatii vain muutaman rivin koodia. Kuviossa luodaan HTML– dokumentti. Dokumentissa määriteltyyn div-elementtiin generoidaan JavaScriptin avulla karttanäkymä. Karttanäkymän generointi on helppoa, sillä kartta voidaan liittää JavaSciptin avulla suoraan Googlen palvelimelta omaan sovellukseen. Kartan aloitusnäkymäksi voidaan määritellä haluttu navigaatiopiste, kun vain tiedetään pisteen koordinaatit (pituus- ja leveysasteet), havainnollistettuna esimerkissä rivillä 16 ja 17.

Karttanäkymään voidaan lisätä kartan kontrollipainikkeet ja markkereita. Seuraavassa kuviossa (KUVIO 16) on lisätty karttaan yksi esimerkkimarkkeri ja kontrollipainikkeet.

```
12        map.addControl(new GSmallMapControl());
13        var location = new GLatLng(60.98160191607567, 25.645179748535156);
14        map.setCenter(location, 12);
15        var marker = new GMarker(location);
16        map.addOverlay(marker);
```

KUVIO 16. Karttakontrollin ja yhden markkerin lisääminen

Esimerkki kuviossa 16 lisätää rivillä 12 karttakontrollipainikkeett karttasovellukseen. Kontrollipainikkeiden avulla käyttäjä voi tarkentaa, loitontaa karttanäkymää ja liikuttaa kartaa. Rivillä 15 ja 16 luodaan uusi markkeri, ja sen oletuskohdaksi laitetaan sama piste, kuin kartan keskikohdaksi on laitettu.

KUVIO 17. Edellä olevien esimerkkikoodien lopputulos

Kuviossa 17 esitetään edellisten kuvioiden (KUVIO 15 & 16) koodin lopputulos, kun se suoritetaan selaimessa.

4 ZEND FRAMEWORK

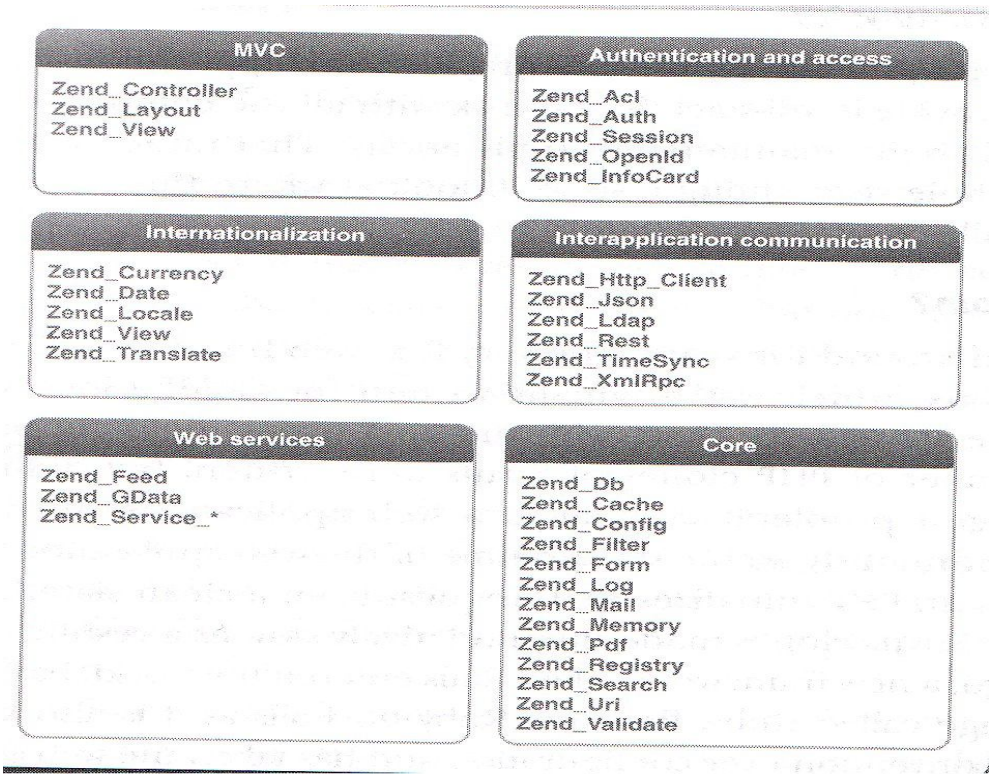
4.1 Yleistä

Zend Framework (ZF) sai alkunsa vuoden 2005 alussa, kun monet muut ohjelmistokehykset kasvattivat suosiotaan. Niinsanotut sivustokehykset ovat vanha ilmiö ja ensimmäisiä kehyksiä olivat Goldfusionille kirjoitettu ”Fusebox” ja Javapohjainen ”Struts”. Seuraava merkittävä kehys oli ”Rails”. Rails- kehys pohjautuu Ruby on Rails -kieleen. Se mullisti internetkehityksen uudella ajattelutavallaan. (Allen, Lo & Brown 2009.)

Zend Framework on php-koodi kirjasto. Sen avulla on helppo rakentaa toimivia, moderneja ja helposti ylläpidettäviä php-sovelluksia. ZF on suunniteltu niin, että eri komponenteilla on mahdollisimman vähän riippuvuuksia toisistaan. Zend Framework on vapaan lähdekoodin projekti, joka tähtää kehyksen vakinaistamiseen yhtenä ”defacto” kehyksenä sivustokehityksessä. Zend Framework koostuu kuudesta kategoriasta, jotka havainnollistetaan seuraavassa kuviossa (KUVIO 18):

- MVC
- Authentication and access
- Internationalization
- Interapplication communication
- Web services
- Core.

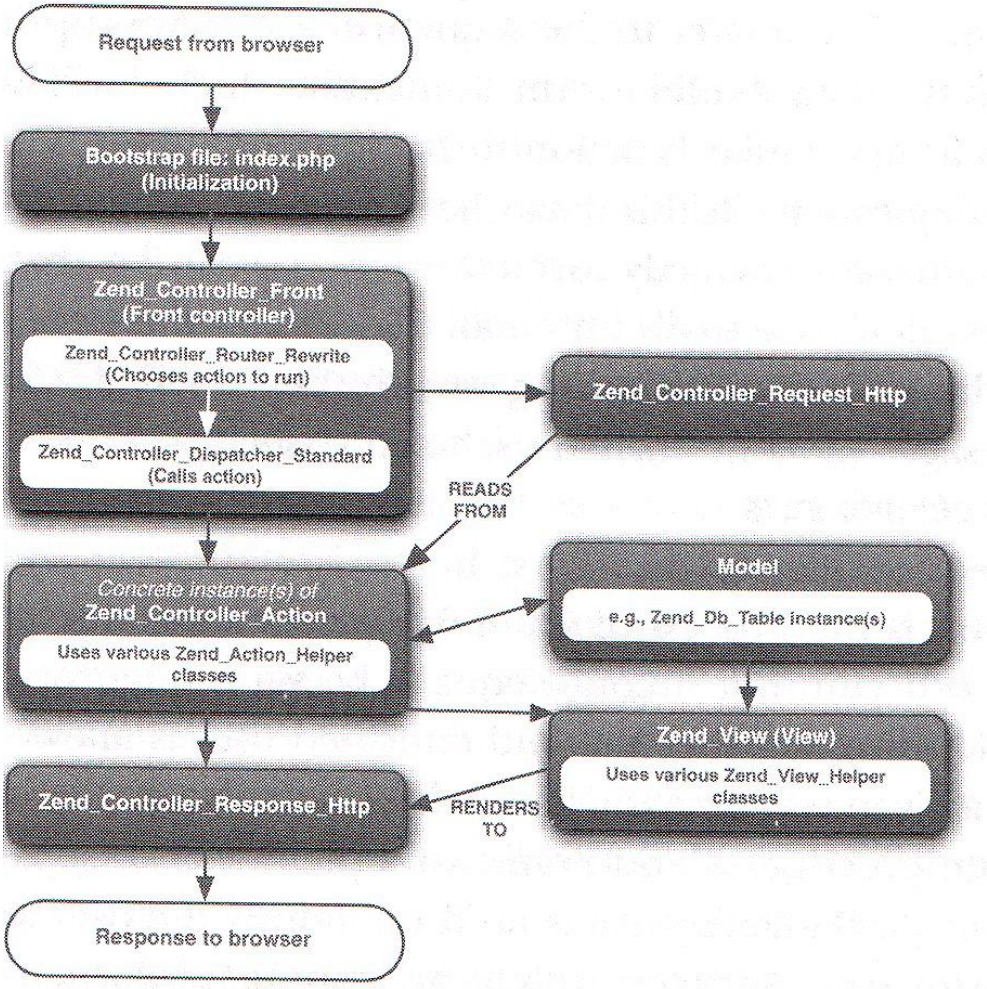
(Allen, Lo & Brown 2009). Seuraavissa kappaleissa tarkastellaan muutamia tärkeimpiä funktioita jokaisesta kategoriasta.



KUVIO 18. ZF:n komponentit

4.2 MVC

Ohjelmistot tarvitsevat hyvin suunnitellun arkkitehtuurin, jotta niiden laajennettavuus ja ylläpidettävyys olisi mahdollisimman helppoa. MVC-suunnittelumalli jakaa ohjelman kolmeen eri osaan: malli, näkymä ja ohjain. Jaottelulla pyritään erottamaan toimintalogiikka näkymästä. ZF:ssä suunnittelumallin mukainen rakenne ei toteudu automaattisesti, vaan se pitää itse rakentaa. Seuraavassa kuviossa (Kuvio 19) havainnollistetaan ZF:n toimintaa pääpiirteittäin. Siinä Zend_Controller_Front saa sivupyynnön ja välittää sen eteenpäin. ZF käsittelee URL:ia eritavalla kuin perinteinen PHP sovellus. Esimerkki perinteisestä URL-muodosta: *index.php?controller=news &action=list*. ZF:ssa on sisään rakennettu esikäsittelijä, jonka avulla URL voi olla helpommin muistettavaa muotoa: *www.mastonet.fi/news/list*.



KUVIO 19. MVC mallin mukainen toiminta ZF:ssä

Zend_controller on koko ZF:n sydän. Se on suunniteltu mahdollisimman helpoksi ja laajennettavaksi. Käyttäjän syöttäessä selaimeen osoitteen Zend_Controller tarkistaa onko siinä määritelty domainin jälkeisiä osia. Jos domain nimen jälkeen on määritelty esimerkiksi *news/list.* Päättelee ZF siitä, että käyttäjä haluaa nähdä uutisosion. Jokainen domainnimen jälkeen määritelty osio on oma luokkansa ja metodinsa. Tässä tapauksessa Zend_Controller hakee luokkaa nimeltä news ja siitä luokasta funktiota list. Jos luokan nimen jälkeen ei ole määritelty funktiota, ZF olettaa funktioksi index- funktion. (Programmer's Reference Guide: Zend Framework 2008).

4.3 Authentication and access

ZF:ssä on sisään rakennettu kaksi käyttäjän tunnistamiseen ja verifiontiin tarkoitettua luokkaa (Zend_Acl ja Zend_Auth). Zend_Acl on kevyt tapa toteuttaa roolipohjainen käyttäjän hallinta. Objekti, joihin käyttäjänhallinta halutaan implementoida, rekisteröidään resurssiksi käyttämällä valmista luokkaa Zend_Acl_resource_Interface ja tämän funktiota getResourceId(). Zend_acl luo resursseista puumaisen rakenteen, jolloin rakenteessa oleville resursseille voidaan luoda periytyviä sääntöä. Jokaiselle resurssille voidaan erikseen määritellä oikeudet (luku, kirjoitus, päivitys ja poisto), tai vastavuoroisesti koko oikeudet voidaan määritellä koskemaan jotain tiettyä sääntöä. Rooleja luodaan kutsumalla Zend_Acl_Role_Interface ja tämän ainoata funktiota getRoleId(). Roolit voivat periä oikeuksia toisilta rooleilta.

Zend_Auth:ia käytetään käyttäjän tunnistamiseen. Sen käyttöön liitetään usein Zend_Session:in käyttö. Zend_Authin avulla kontrolloidaan ainoastaan käyttäjän tunnistautumista, ei pääsyn valvontaa. Zend_auth luokka implementoidaan Zend_Auth_Adapter_Interface luokasta. Luokalla on yksi metodi authenticate. Seuraavassa kuviossa (KUVIO 20) kuvataan authenticate funktion kutsua. (Programmer's Reference Guide: Zend Framework 2008).

```
42          $authAdapter = $this->_getAuthAdapter($formData);
43          $auth = Zend_Auth::getInstance();
44
45          $result = $auth->authenticate($authAdapter);
46          if(!$result->isValid())
47          {
48              $this->_flashMessage('Login failed');
49          }
50          else
51          {
52              $data = $authAdapter->getResultRowObject(null,'password');
53              $auth->getStorage()->write($data);
54
55              $this->_redirect($this->_redirectUrl);
56              return;
57          }
```

KUVIO 20. Authentikaation käyttö

Kuviossa 20 riveillä 42 ja 43 määritellään uusi auth- olio ja auth adapteri. Rivillä 45 kutsutaan auth- olion authenticate funktiota, jolle välitetään parametrina adapteri.

4.4 Internationalization

Zend Framework tarjoaa tuen monikielisten sivustojen luontiin. ZF:n internationalization komponentti tarjoaa laajan kirjon eri funktioita monikielisyyden toteuttamiseen. Esimerkiksi: oikean valuutta symbolin näyttämiseen ja vallutan laskentaan (Zend_Currency & Zend_Date) ja sivujen sisältötekstin muuntamiseen toiselle kielelle (Zend_Translate).

Tarkastellaan hieman lähemmin Zend_Translate:n toimintaa ja rakennetta. Zend_Translate on ZF:n versio monikielisestä sovelluksesta. ZF:n mukaan monikielisen sivuston rakentaminen voidaan jakaa seuraaviin neljään vaiheeseen:

- Valitse, mitä sovitinta haluat käyttää
- Luo kielinäkymät ja implementoi Zend_Translate koodiisi
- Luo lähdekoodi
- Käännä tämä lähdetiedosto halutuille kielille.

Sovittimia on monta erilaista, mutta yleisimmin käytetty on Zend_Translate_Adapter_Gettext. (Programmer’s Reference Guide: Zend Framework 2008).

4.5 Interapplication communication

Interapplication communication kategorian luokat ovat tarkoitettu sivustojen väliseen kommunikointiin. Kategorian luokat toimivat monin osin samalla tavalla kuin PHP:n curl-lisäosa (curl-lisäosa on kirjasto jonka avulla voi luoda yhteyden moniin erityyppisiin palvelimiin käyttäen eri protkollia). Kategorian luokat tarjoavat tuen muun muassa JSON-protokollan käytölle Zend_Json:n avulla.

Zend_Json tarjoaa helpon tavan kerätä dynaamista dataa palvelimelta, ennen kuin se siirretään Ajax palvelulle selaimeen. Zend_Json tarjoaa funktiot natiivin PHP:n kääntämiseen JavaScriptiksi ja päinvastoin. Kääntöfunktioita käytettäessä Zend_Json ei hyväksy olio viittauksia. JSON objektia käänettäessä takaisin natiiviksi PHP:ksi palautettava arvo on yleensä assosiatiivinen taulukko.

Zend_Json:ssa on myös monta muuta sisäänrakennettua funktiota. Yksi näistä on ::fromXml. Sen avulla voi kääntää minkä tahansa XML- syötteen JSON: ksi. (Programmer's Reference Guide: Zend Framework 2008).

4.6 Web services

Zend tarjoaa joukon toimintoja, joiden avulla voi kytkeytyä käyttämään muiden tarjoamia palveluita. Esimerkiksi Zend_Feed Rss:ää voidaan käyttää syötteiden käsittelyyn. Zend tarjoaa komponentteja Yahoo:n, Google:n ja monen muun julkisten API:en liitäämiseksi omaan sovellukseen. Googlen palveluiden käyttämiseen ZF on luonut oman Zend_Gdata komponentin. Zend_Gdata on PHP 5 rajapinta. Sen avulla voi käyttää muunmuassa seuraavia Googlen palveluita: blogi, kalenteri, youtube, koodihaku ja Picasa. Googlen data palvelut pojautuvat AAP (Atom Publishing Protocol) protokollaan ja Atom syndikaatti formaattiin. Zend_Gdata komponentin tarjoamien funktioiden avulla on mahdollista hakea, lisätä, päivittää ja tuhota merkintöjä Google palveluista. Näiden luokkien avulla kommunikointi Googlen suuntaan tapahtuu syötteinä. Monet Googlen palvelut vaativat käyttäjän tunnistautumisen ennen käyttöä, Zend_Gdata tarjoaa tähän kaksi luokkaa AuthSub ja ClientLogin. (Programmer's Reference Guide: Zend Framework 2008).

4.7 Core

Core on kokoelma eri komponentteja, mitä ei voinut ryhmittää muihin kategorioihin. Tässä ryhmässä on muun muassa seuraavat luokat luokkatietojen hakemiseen (Zend_Search_Lucene), välimuistin käyttöön (Zend_Cache), pdf-luontiin

(Zend_Pdf) ja sähköpostin lähettämiseen (Zend_Mail). Eräs tärkeimmistä tämän kategorian luokista on Zend_Config.

Zend_Config:n avulla voi koko sovelluksen konfiguroinnin hoitaa yksinkertaisesti ja keskitetysti. Konfiguraatiotiedot voivat olla monessa eri formaatissa. Yleisimmin käytettyjä formaatteja ovat Xml ja Ini tiedostot. (Programmer's Reference Guide: Zend Framework 2008.) Seuraavassa kuviossa (KUVIO 21), on esimerkki Ini-muotoisesta konfiguraatio tiedostosta.

```
1  [db]
2  database.host = 127.0.0.1
3  database.port = 3306
4  database.dbname = mastonet
5  database.username = local_masto_user
6  database.password = "MrR496#0P"
7
8  ;Inherits data from db
9  [development : db]
10 developer = true
11
12 [production : db]
13 developer = false
14
```

KUVIO 21. Ini muotoinen konfiguraatio tiedosto

Kuviossa 21, konfiguraatio tiedostossa määritellään rivillä 1 osio nimeltä ”db” ja sille tietokannan konfiguraatio tietoja. Riveillä 9 ja 12 määritellään kaksi erinimistä ajoympäristöä, jotka perivät määrittelyinformaatiot osiolta ”db”.

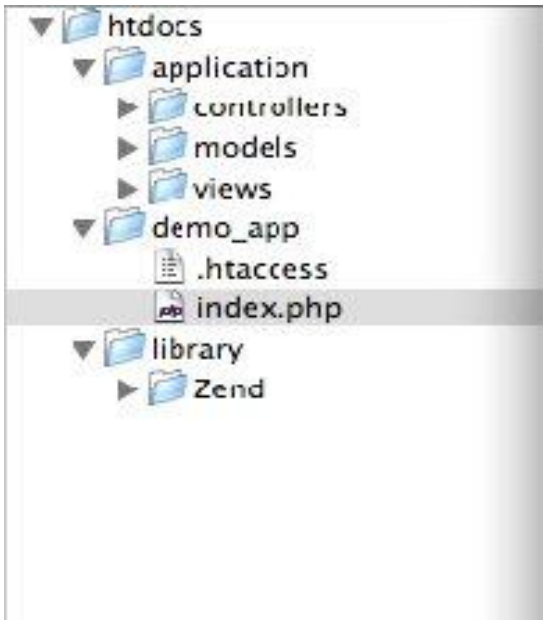
```
13  $config = new Zend_Config_Ini('../application/config.ini', 'development');
14  Zend_Registry::set('config',$config);
15  // Error reporting
16  error_reporting(E_ALL|E_STRICT);
17  if($config->developer)
18      ini_set('display_errors','on');
19  else
20      ini_set('display_errors','off');
```

KUVIO 22. Konfiguraatio tiedon käyttö PHP:ssä

Kuviosta 22 voidaan huomata, kuinka rivillä 13 määritellään käytettävä konfiguraatiotiedosto ja sen osio. Riveillä 17 hyödynnetään Ini- tiedostossa määriteltyä ”developer” parametria. Parametrin perusteella, joko näytetään tai piilotetaan sivuilla tapahtuvat virheet.

4.8 Zend Framework:n asennus ja hakemistorakenne

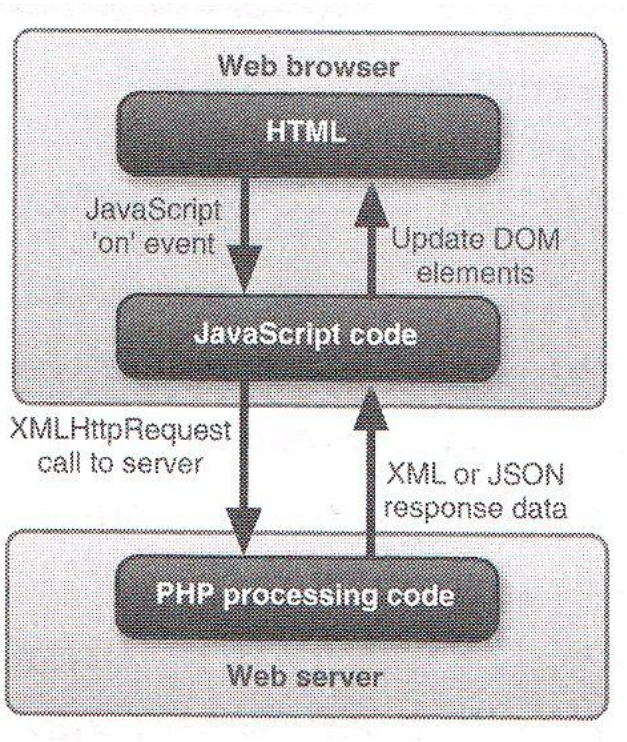
Luotaessa sovellusta tarvitsee ensimmäisenä asentaa ZF. Asennus on kaikessa yksinkertaisuudessaan yhden hakemiston (library) kopiointi sovelluksen juureen. Library-hakemistossa on kaikki ZF:n toiminnalle tärkeät tiedostot, joten se kannattaa pitää varsinaisen sovelluksen juuren ulkopuolella. Seuraavasta kuviosta (KUVIO 23) näemme miten tyypillisen ZF -sovellusen hakemistorakenne koostuu. Sovelluksen juureksi on Apachessa määritelty ”demo_app”. Tällöin library ja application hakemistot ovat varsinaisen ”Sovelluksen juuren” ulkopuolella. Näin ne ovat suojassa väärinkäytöksiltä, eli niihin ei ole mahdollista päästä käsiksi nykyisillä selaintekniikoilla. (Allen, Lo & Brown 2009.)



Kuvio 23. ZF sovelluksen tyypillinen hakemistorakenne.

4.9 ZF ja Ajax

ZF:n kannalta Ajax kutsu muistuttaa mitä tahansa muuta toimintokutsua, ja se vaatii controllerin ottamaan vastaan kutsun. Ainoa ero on, että kutsu palauttaa JSON snippetin. Seuraavassa kuviossa (KUVIO 24) selvennetään lisää Ajax-kutsutoimintaa. Ajax-kutsujen käsittelyjä varten ZF:ään voi rakentaa apufunktioita, jotka käsittelevät pyynnön ja päivittävät näkymän. (Allen, Lo & Brown 2009 95-96.)



KUVIO 24. Ajaxin toimintaa

4.10 Helpers

ZF sisältää paljon valmiita apufunktioita ja metodeja. Avustaja luokat ovat hyvä tapa toteuttaa yleisesti käytettyjä funktioita ja koodia. Yleisimmät funktiot ovat yhdessä paikassa, ja tällöin sovelluksen ylläpidettävyys on huomattavasti helpompaa. (Allen, Lo & Brown. 2009. 33 – 34.)

Apuluokkien avulla voi esimerkiksi käsitellä helposti JSON kutsuja, Ajax pyyntöjä ja Flash viestejä. Eräs tärkeimmistä apuluokista on ViewRenderer. Sen päätehtäviä ovat:

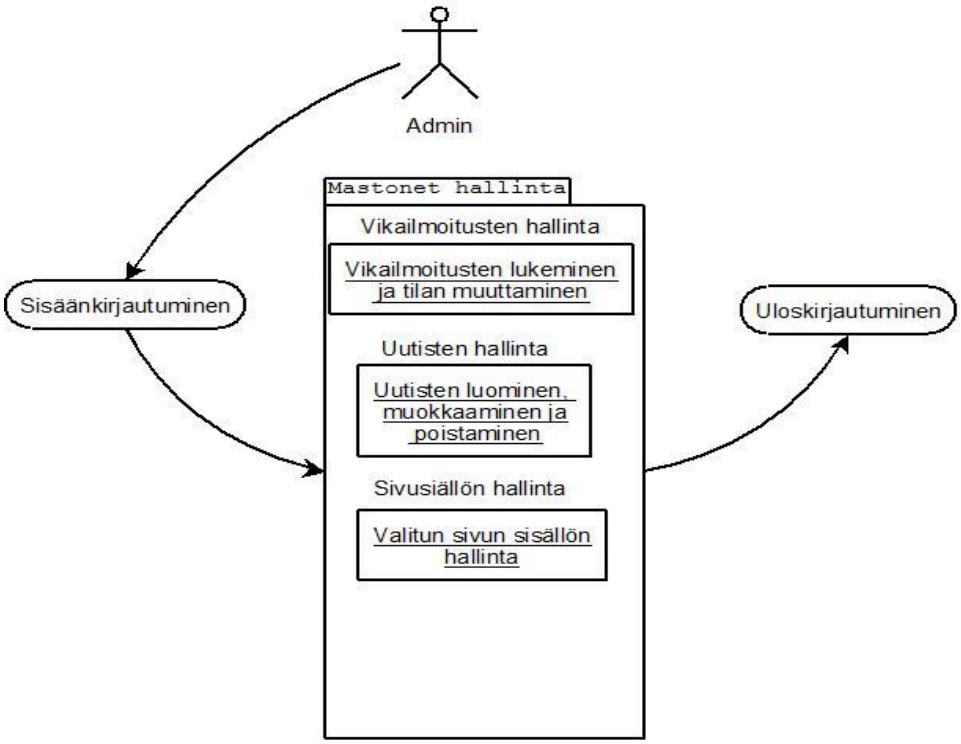
- luoda globaali ”view” olio
- mahdollistaa vakionäkymä kaikille controllerille
- automatisoida ”view” olion rekisteröiti controllerille.

(Programmer’s Reference Guide: Zend Framework 2008.)

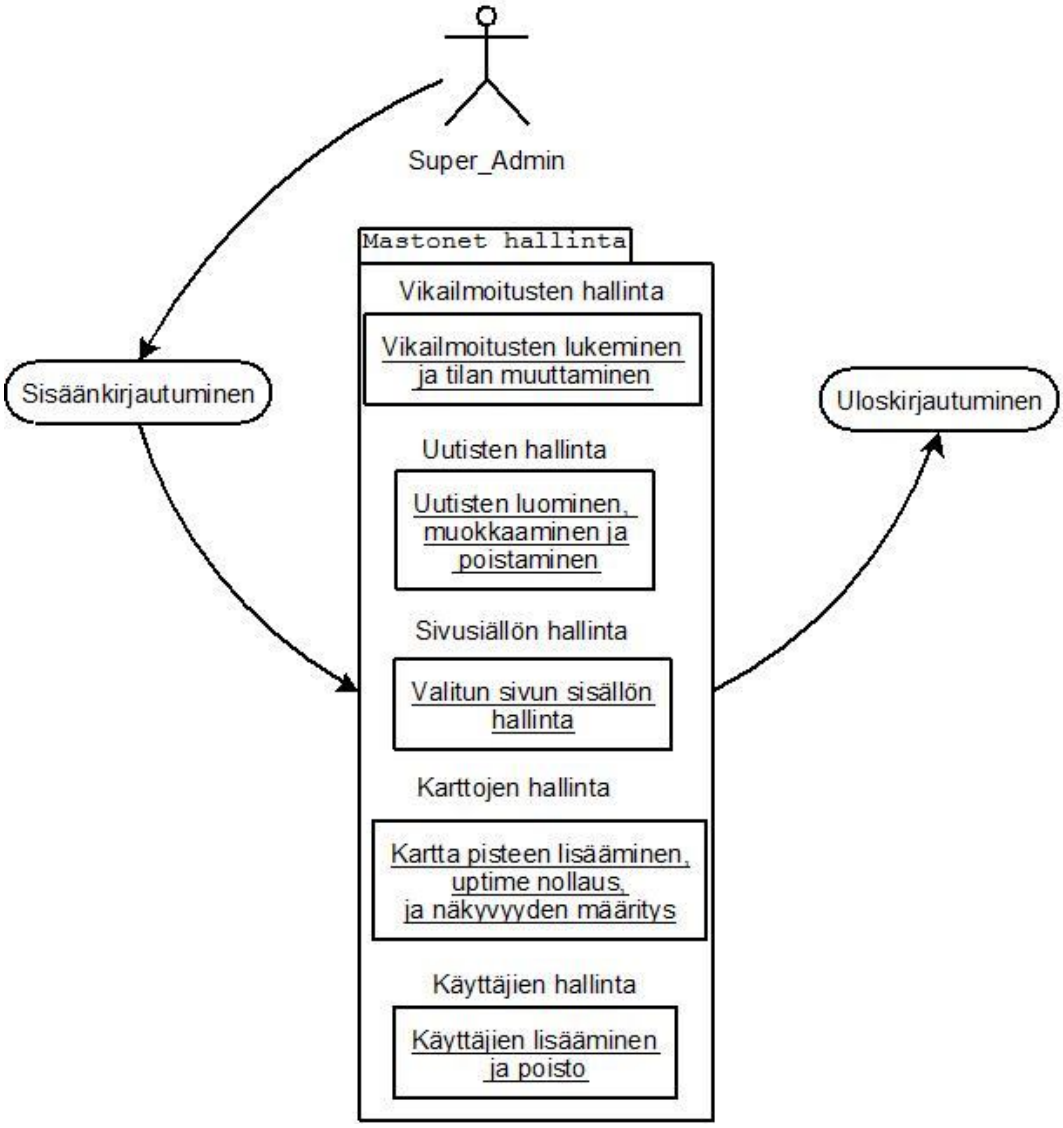
5 TIETOKANTASUUNNITTELU

Nykyaikaisten web-sivustojen tärkeän osan muodostaa tietokanta. Jos tietokanta ei ole hyvin suunniteltu, saattaa se näkyä käyttäjälle sivuston hitautena ja jossain tapauksissa jopa toimimattomuutena. Suunniteltaessa tietokantaa, on otettava huomioon laaja kirjo asioita, jotka määrittelevät vaatimukset tietokannan mallinnuksesta ja fyysisestä suunnittelusta. Hyvin suunnitellun tietokannan keskeisiä ominaisuuksia ovat kattavuus, selkeys, ymmärrettävyys, eheys, muutosjoustavuus, ohjelmointimukavuus ja tehokkuus. (Hovi, Huotari & Lahdenmäki 2003, 20-21.)

Työssä toteutetun tietokannan asiakasvaatimuksia kuvataan seuraavissa kuvioissa (KUVIO 25 & 26). Kuviossa esitellään käyttötapauskaaviot eri käyttäjille. Toiminnot edellyttävät sisäänkirjautumisen, joten sisään- ja uloskirjautuminen on kuvattu käyttötapauksena. Järjestelmällä on kahden tasoisia käyttäjiä:
Admin – Opiskelija käyttäjä
SuperAdmin – Lahden ammattikorkeakoulun opettaja, järjestelmän ylläpitäjä.

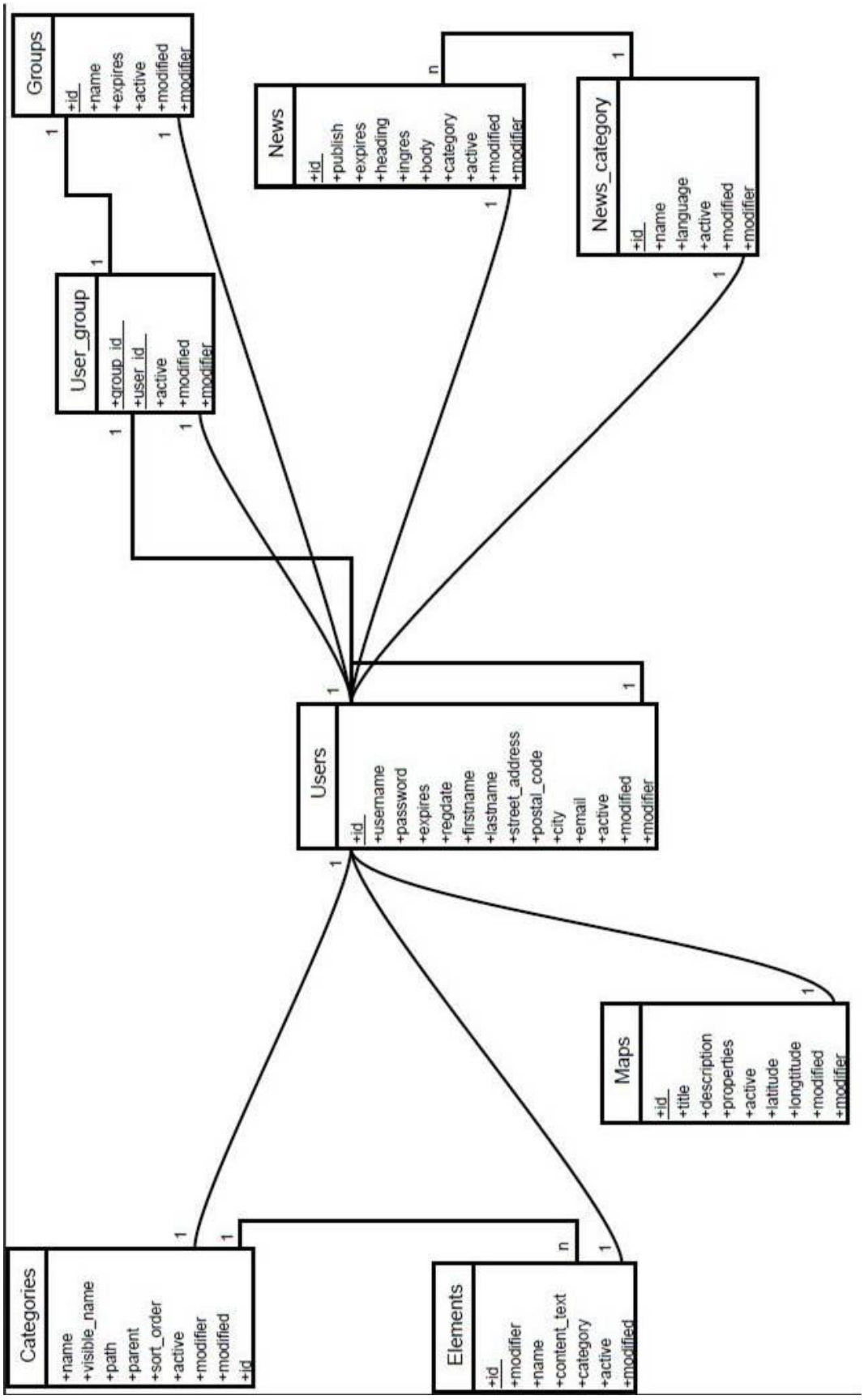


KUVIO 25. Admin-käyttötapaus



KUVIO 26. SuperAdmin-käyttötapauskaavio

Tässä työssä totetutettua tietokantaa voi tarkastella liitteenä olevasta tietokanta-kaaviosta (LIITE 1). Tietokannan relaatioita talujen välillä on kuvattu seuraavassa kuviossa (KUVIO 27). Kuviosta käy ilmi, että tietokanta on suunnitelty niin, että jokainen käyttäjän tekemä muutos on jäljitettävissä. Tietokannan keskeisin taulu on käyttäjätaulu (Users), kaikki tietokannan muut taulut viittaavat tähän tauluun. Tietokannan taulut ovat jaettavissa kahteen ryhmään. Sivuston sisältöön vaikuttaviin tauluihin (Categories, Elements, News, News_category ja Maps) ja käyttäjien hallinta tauluihin (Users, User_group ja Groups).



KUVIO 27. Tietokannan relaatiomalli

Kuviossa 27 kuvatusta relaatiomallista voidaan huomata, että tietokannan keskeisin taulu on Users. Kaikilla tauluilla on 1-1:n suhde tämän taulun kanssa. Tällä viittauksella voidaan jäljittää jokaisen tietokantaan tallennetun tietueen muokkaaja (modifier). Lisäksi kaikissa tauluissa on modified-kenttä, jonka avulla voidaan seurata, milloin muutokset on tallennettu. Käyttäjät linkitetään ryhmään user_group linkkitaulun avulla. Jokaisessa taulussa on ”id” kenttä tallennettavien tietueiden yksilöimistä varten ja active kenttä tietueiden aktiivisuuden ilmentämiseksi. News ja News_category taulujen välillä 1-n viittaus. Uutiskategorialla voi olla useita uutisia, mutta uutinen voi kuulua vain yhteen kategoriaan. Saman tapainen yhteys on myös Categories- ja Elements-taulujen välillä. Kategorialla voi olla useita elementtejä, mutta yksi elementti voi kuulua vain yhteen kategoriaan. Jokaisella kategorialla on niin sanottu isäntäkategoria, jota ilmaistaan parent arvolla. Categories- ja elementstauluilla hallitaan sivuilla näkyvää sisältötekstiä.

6 MASTONET

6.1 Historia ja nykytila

Mastonet on Lahden kaupungin perustama avoin WLAN-teknologiaan perustuva langaton verkko, jonka käyttö on kaikille ilmaista. Verkon muodostaminen aloitettiin vuonna 2005, ja se on edennyt vaiheittain. Verkko on Suomen toiseksi suurin avoin langaton verkko, vain Oulun panOulu on suurempi. Tällähetkellä verkkoon kuuluu 82 toiminnassa olevaa tukiasemaa. (Wikipedia, mastonet.) Verkonhallinta siirtyi Lahden ammatikorkeakoululle vuoden 2009 alusta.

6.2 Sivuston käyttötarkoitus

Työssä toteutetaan uudet internet sivut Mastonet-verkolle. Sivuille liitetään Googlen tarjoama ilmainen karttapalvelu kertomaan tukiasemien sijaintia ja liikennemääriä. Tukiasemien liikennemäärätiedot näytetään kuvioina, jotka on generoitu RRDtool:ia apuna käyttäen. Sivustolla on oma hallintaosio, jonka toteutus kuuluu tämän työn piiriin. Koska kyseessä on laaja ja dynaamista tietoa sisältävä sivusto, jolle suunnitellaan ja toteutetaan myös tietokanta.

6.3 Tärkeimmät toiminnot

Sivuston tärkeinpiä toimintoja ovat Google Maps-sovellus ja sivustonhallinta-osio. Hallintaosiossa voi hallita tukiasema tietoja, luodaan tai muokataan uutisia, käsitellä vikailmoituksia ja hallita sivujen sisältötekstejä. Hallinta-osio on jaettu kahteen eri käyttäjätasoryhmään.

6.4 Käyttöliittymävaatimukset

Sivuston käyttöliittymästä on tarkoitus tehdä helposti omaksuttava ja ylläpidettävä. Käyttöliittymän vaatimuksina ovat selkeys, esteettisyys, käytettävyys ja ylläpidettävyys.

Hallintaosion käyttöliittymä tuo omat haasteensa, sillä siinä on kaksi erilaista näkymää. Näkymät vaihtelevat kirjautuneen käyttäjän oikeustason mukaisesti. Niin sanotuilla ”admin”- ryhmään kuuluvilla käyttäjillä näkyvillä on vain vikailmoitusten hallinnointi. Varsinaisilla ”SuperAdmin”- ryhmään kuuluvilla käyttäjillä on näkyvillä kaikki hallintaosion toiminnot.

6.5 Sivuston rakenne

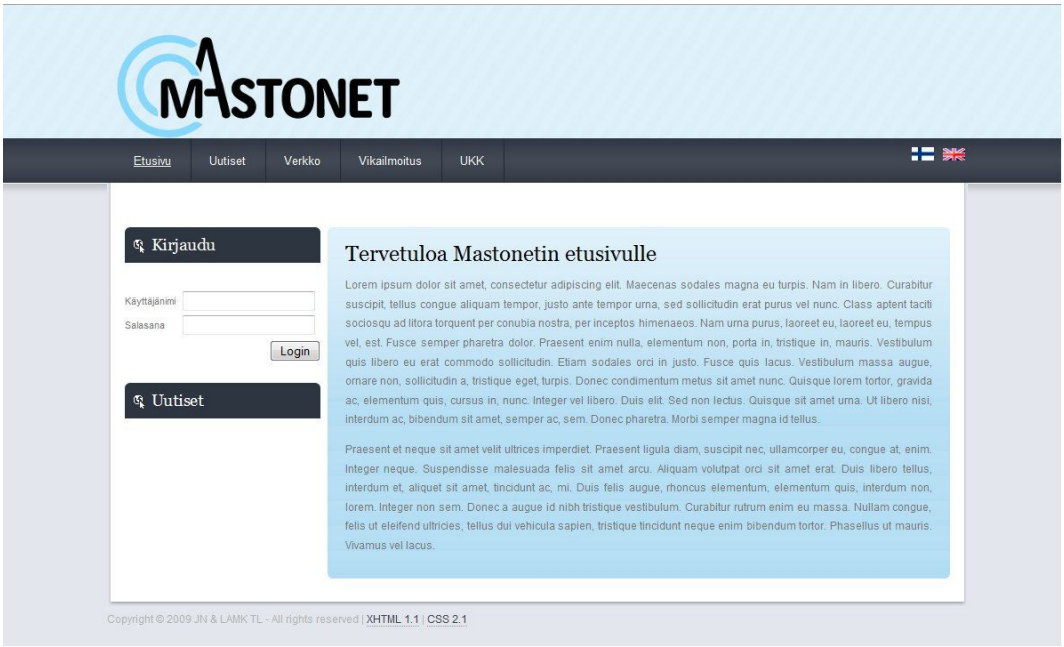
Seuraava lista kuvaa sivuston hierarkista rakennetta. Sivusto on kaksikielinen ja päätason kategorioita on kolme kappaletta (FI, EN ja Admin). Kahdella ensin mainitulla kategorialla on alakategorioita.

- FI
    - Etusivu
    - Uutiset
        - UKK
    - Vikailmoitukset
    - Tukiasematiedot
- EN
    - Frontpage
    - News
        - FAQ
    - Service advice
    - Network
- Admin

6.6 Työn toteutus

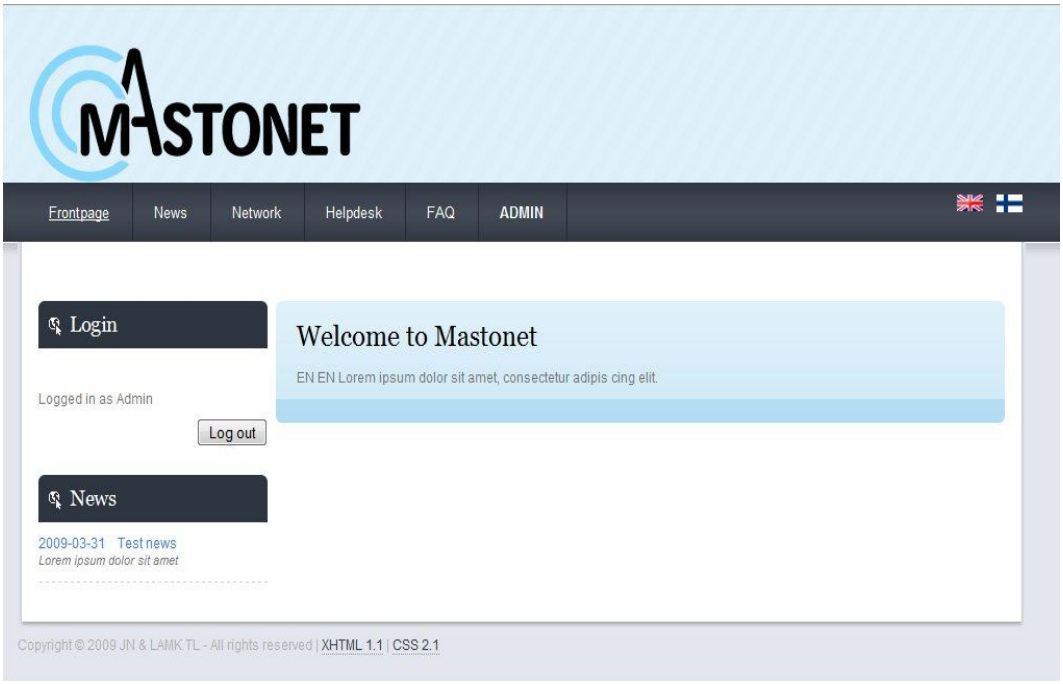
Koska työn toteutuksessa tavittavat tukiasematiedot tulevat toisesta järjestelmästä niiden hakeminen toteutetaan ajastettuna. Tätä varten palvelimelle luotiin ajastupalvelu. Palvelimen käyttöjärjestelmän ollessa Unix-pohjainen luotiin ajastus käyttäen ”cron”- palvelua. Ajastuspalvelu suorittaa konennot taustalla, kun aika- ja päivämäärämääritykset täsmäävät. Ajastuspalvelun avulla toisesta järjestelmästä haetut tukiasema tiedot tallennetaan, työssä toteutetun järjestelmän, hakemistorakenteeseen ja tiedostoista generoidaan RRDtoolia apuna käyttäen kuvat.

Tukiasema tietojen hallintaosiossa haetut tiedot yhdistetään tiettyyn tukiasemaan. Yhdistäminen tapahtuu tukiaseman muokkaustoiminnossa, valitsemalla listalta tiedosto. Tämä viittaus tallennetaan tietokantaan kunkin tukiaseman tietoihin. Tukiaseman muokkaustilassa voidaan tukiaseman ”uptime”-tieto nollata ja määritellä näkyykö kyseinen tukiasema sivustolla.



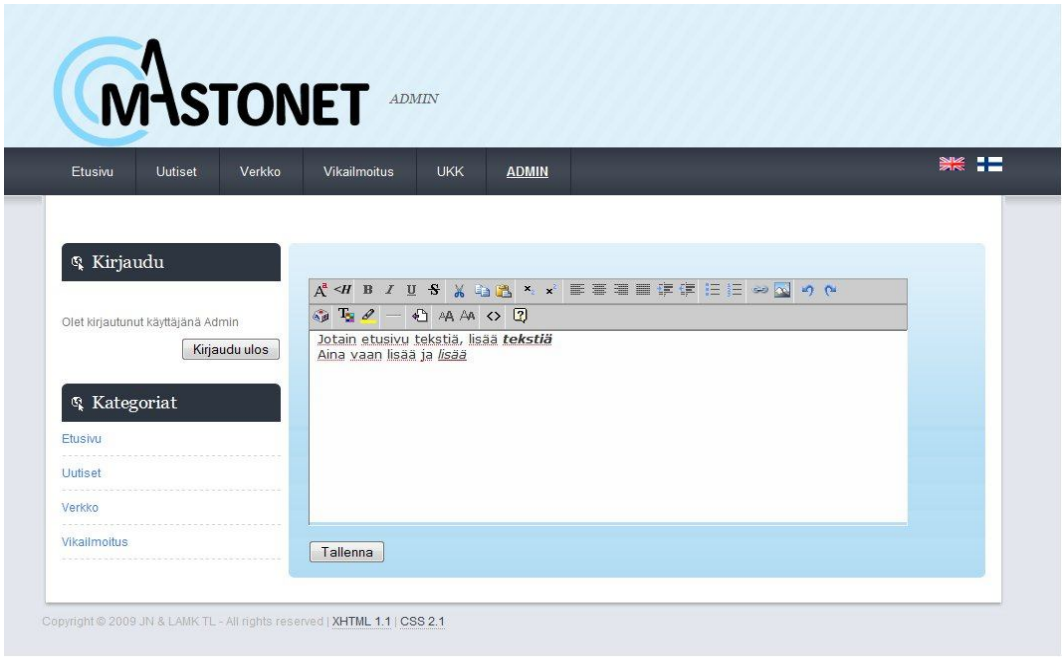
KUVIO 28. Mastonet sivut

Kuviossa 28 on esitelty työssä toteutetun sivuston suomenkielinen etusivu. Kuviossa 29 on sama sivu, kun sivustolle on kirjauduttu sisään.



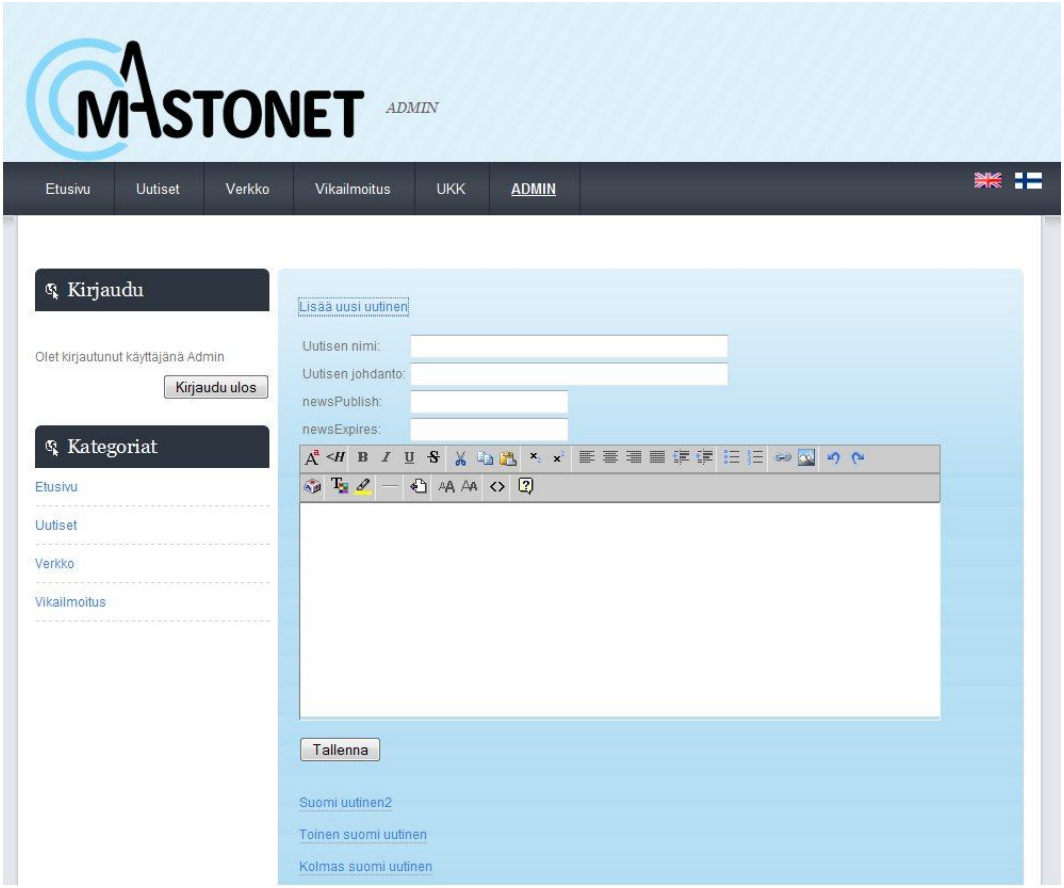
KUVIO 29. Mastonetin etusivu kun on kirjauduttu sisään

Kuvioissa 30 havainnollistetaan hallintaosion etusivun muokkaus osiota. Muokkaus osiossa käyttäjälle näkyy ”WYSIWYG”(What You See Is What You Get eli mitä näet, sitä saat)- editori. Editorin avulla tekstin muokkaus on työpöytä sovelluksen omaista. Käyttäjä voi tallentaa tekemänsä muutokset painamalla tallenna-nappia.



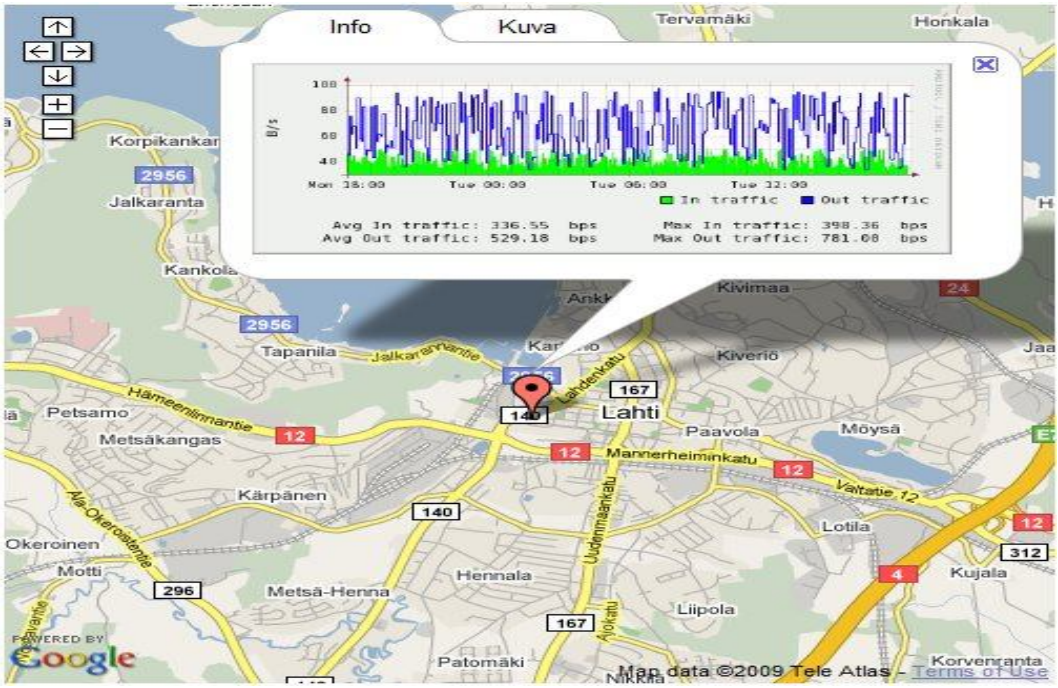
KUVIO 30. Mastonetin etusivun hallintatyökalu

Kuviossa 31 havainnollistetaan sivuston uutisten hallintatyökalu. Lisänä olevien kenttien avulla määritellään uutisen otsikko, johdanto, ilmenemispäivä ja vanhenemispäivä. Ilmenemis- ja vanhenemispäivät valitaan javascript kalenterista, joka ilmestyy ruudulle kenttää painettaessa.



KUVIO 31. Mastonetin uutisten hallintatyökalu.

Kuviossa 32 havainnollistetaan sivuille toteutettua Google Maps-implementaatiota. Markkeria painettaessa näkyviin tulee puhekuplan tapainen näkymä. Näkymässä on kaksi välilehteä. Infovälilehdellä kerrotaan muun muassa tukiaseman liikennemäärät ja nimi. Kuvavälilehdellä voidaan näyttää tukiasemasta otettuja kuvia.



KUVIO 32. Liikenne graafi Google Maps-kartalla

7 TIETOTURVA

Avoin internet on tietoturvan kannalta vaarallinen paikka, eikä käyttäjän tekemiin toimiin saa luottaa missään vaiheessa. Sivuston kaikki syötteet pitää tarkistaa ja sallia vain ne, mitä havaitaan luotettaviksi. Internetpalveluiden kaksi yleisintä ongelmaa ovat ”xss-aukko” ja ”sql-injektio”. (Kotilainen 2009, 55).

Xss-aukon hyväksikäyttämisessä käyttäjä syöttää tekstikenttään koodin, joka vaihtaa kokonaan tai osittain kyseisen sivun sisällön. Viime vuoden maaliskuussa Suomessa nähtiin laajamittainen haavoittuvuusrieha, joka johtui juuri kyseisestä xss-aukosta. Sql-injektio on ongelmista vaarallisempi, koska sen avulla käyttäjät voivat muuttaa tai jopa poistaa tietoja sivuston käyttämästä tietokannasta. (Kotilainen 2009, 55.)

Tunnollinen php koodaaja noudattaa seuraavia tietoturva muistisääntöjä aina koodatessaan:

- Muuttujat alustetaan ennen käyttöä.
- Käyttäjän syöttämä data tulee varmistaa ja puhdistaa.
- Palvelimella ajettavien komentojen kanssa tulee noudattaa erityistä varovaisuutta.
- Tallennettaessa tiedostoja palvelimelle tulee niiden nimet aina muuttaa.
- Syötetystä datasta tulee poistaa aina kaikki html ja javascript koodi.
- PHP sivujen virheitä ei tule koskaan paljastaa tuotanto sivuilla.
- SQL-injektio hyökkäys mahdollisuudet tulee mitätöidä.
- Palvelimella ei tule pitää phpinfo() scriptejä.

(Ullman 2007, 124-125).

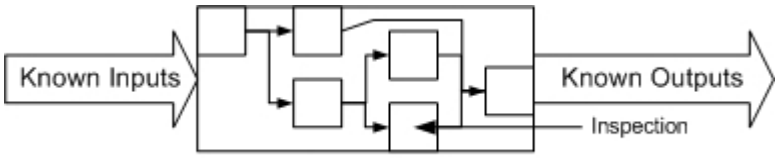
Sivuston kehityksessä tulee kiinnittää erityisesti huomiota tietoturvaan. Jokainen käyttäjän syöttämä tieto validoidaan, ja lisäksi käyttäjien tarkistukseen käytetään

ns. epäsuoraa tapaa. Tämän epäsuoran käyttäjätarkistuksen lisäksi käyttäjän oikeudet tarkistetaan sivukohtaisesti.

8 TESTAUS

Testauksella pyritään varmistamaan koodin laatu ja minimoimaan virheiden määrä. Testaus tapahtuu vaiheissa. Testuksen eri vaiheita ovat suunnittelu, testiympäristön luonti, suorittaminen ja tulosten tarkastelu. (Haikala, 2004, 283-300.)

Tässä työssä testaus keskittyy lähinnä tekemisen ohella tapahtuvaan vikojen korjaamiseen ja siihen liittyvään häirötilanteiden jäljitykseen. Tätä testausmenetelmää nimitetään usein moduulitestauksena. Tämän työn toteutuksen aikana ei kehitä omaa testialustaa moduulitestauksen suorittamiseksi vaan käytetään jo edellä mainittuja keinoja yksittäisten luokkien ja funktioiden testaamiseen. Koska suurimman osan testaamisesta suorittetaan toteutuksen yhdessä, voi testaamista kutsua niin sanotuksi metodi-/luokkatestaamiseksi. Edellä mainittua testaustapaa selvennetään seuraavassa kuviossa (KUVIO 33). Työn laatua varmistetaan käytettävyystestauksella. Käytettävyys testauksessa tarkistetaan järjestelmän helppokäyttöisyys ja eri osien toimivuus.



KUVIO 33. Moduuli- ja luokkatestaus (http://www.codeproject.com/KB/cs/autp1)

9 YHTEENVETO

Opinnäytetyön tavoitteena oli tutkia ja toteuttaa Mastonet-projektille uudet internet sivut. Sivuille implementoitiin Google Maps sovellus, kertomaan tukiasemien tietoja. Sivuston alustaksi valitsin Zend Frameworkin. Työn toteutuksen edetessä ZF, JavaScript ja Google Maps tekniikat tulivat tutuiksi. Opimisen kannalta erityisen hyödyllistä oli monien kirjojen ja sivustojen lukeminen perehdyttäessä ZF:n rakenteeseen ja mahdollisiin käyttötapoihin. JavaScript ja Google maps osoittautuivat selkeiksi ja suhteelisen helpoiksi omaksua. Kaikkia ZF:n tai Google mapsin tarjoamia ominaisuuksia ei käytetty työssä hyväksi, joten sivustolle integroitua karttaa voidaan vielä monimuotoistaa. Samoin alustaksi valitun ZF:n ominaisuuksi voidaan tulevaisuudessa höydyntää laajemmin.

Tätä raporttiosuutta kirjoittaessa työ on vielä toteutusvaiheessa. Tähän mennessä suurin osa toteuttamiseen käytetystä ajasta on mennyt ZF:ään perehtymiseen. ZF on osoittautunut haastavaksi alustaksi, koska sen toteuttaminen poikkeaa suuresti niin sanotusta ”sivusto alustalle” ajatuksesta kehittää internetsivusto. Zend Framework ei ole alusta vaan enneminkin PHP:n luokkakirjasto.

Jatkokehitys ideana sivuston kaikki toiminnot voisi toetuttaa Ajaxia hyödyntäen. Ajaxia hyödynnettäessä toiminnot saavat niin sanotun työpöytävaikutuksen, muutokset näkyvät heti ja sivua ei tarvitse ladata uudelleen. Tämän suuntainen sivustojen kehitys on jo nykypäivää, mutta se tuo mukanaan jo opittujen käyttötapojen muutoksen. Käyttäjien tulee opetella uusi tapa käyttää internetsivuja, enää ei voi ”palata takaisin” käyttäen selaimen tarjoamia nappeja. Ajaxia käyttävät sivustot vaativat entistä enemmän käyttäjien keskittymistä ja paneutumista. Sivustolla kaikki muutokset ja toiminnot tapahtuvat välittömästi, jolloin käyttäjällä on suurempi vastuu.

LÄHTEET

Allen & Lo & Brown. 2007. Zend Framework in Action. Manning Publications

Babin, Lee & Good, Nathan & Kromann, Frank & Stephens, Jon 2005 PHP5 Recipes APRESS

Boumphrey & Greer & D. Raggett & J. Raggett & Schnitzenbaumer & Qugofski 2000. Inside XHTML Ohjelmoijan käsikirja. Oy Edita Ab

Eichorn Joshua 2006. Understanding AJAX: Using JavaScript to Create Rich Internet Applications Prentice Hall

Haikala & Märijärvi. 2004. Ohjelmistotuotanto

Holzner, Steven. 2001 Inside XML Gummerus Kirjapaino Oy

Hovi & Huotari & Lahdenmäki. 2003. Tietokantojen suunnittelu ja indeksointi. Docendo Finland Oy

Moncur, Michael. 2000 Javascript Trainer IT Press

Purvis & Sambells & Turner. 2006. Beginning Google Maps Applications with PHP and Ajax. Apress

Programmer's Reference Guide: Zend Framework 2008

Ullman. 2007. PHP 5 Advanced. Peachpit Press

Kotilainen. 2009. Saittien tietoturva vuotaa. Tietokone 1 sivut 53-55.

CSS 2009 [verkkojulkaisu] Cascading Order and Inheritance in CSS [viitattu

20.4.2009] Saatavissa: http://monc.se/kitchen/38/cascading-order-and-inheritance-in-css

Leponiemi, Jarkko 2002 [verkkojulkaisu]. Palvelinpuolen skriptikielet. [viitattu 27.3.2009] Saatavissa: http://www11.uta.fi/~jl/php/yleista.php

Nationalatlas 2009 [verkkojulkaisu] [viitattu 10.4.2009] Saatavissa: http://www.nationalatlas.gov/articles/mapping/a_latlong.html

RRDtool 2009 [verkkokjulkaisu] RRDtool logging & graphing [viitattu 13.4.2009] Saatavissa: http://oss.oetiker.ch/rrdtool/tut/rrdtutorial.en.html

Suominen, Jaakko 2002. [verkkojulkaisu] Millainen on hyvä www-sivusto. [viitattu 27.3.2009] Saatavissa: http://www.tuug.fi/~jaakko/opetus/verkko2002/hyvat.phtml

WDS 2009.[verkkojulkaisu] Hyvä sisältä vakuuttaa. [viitattu 27.3.2009] Saatavissa: http://www.wds.fi/

Wikipedia, Internet 2009 [verkkojulkaisu] Wikipedia, the free encyclopedia [viitattu 1.3.2009] Saatavissa: http://fi.wikipedia.org/wiki/Internet

Wikipedia, JSON 2009 [verkkojulkaisu] Wikipedia, the free encyclopedia [viitattu 13.2.2009] Saatavissa: http://fi.wikipedia.org/wiki/JSON

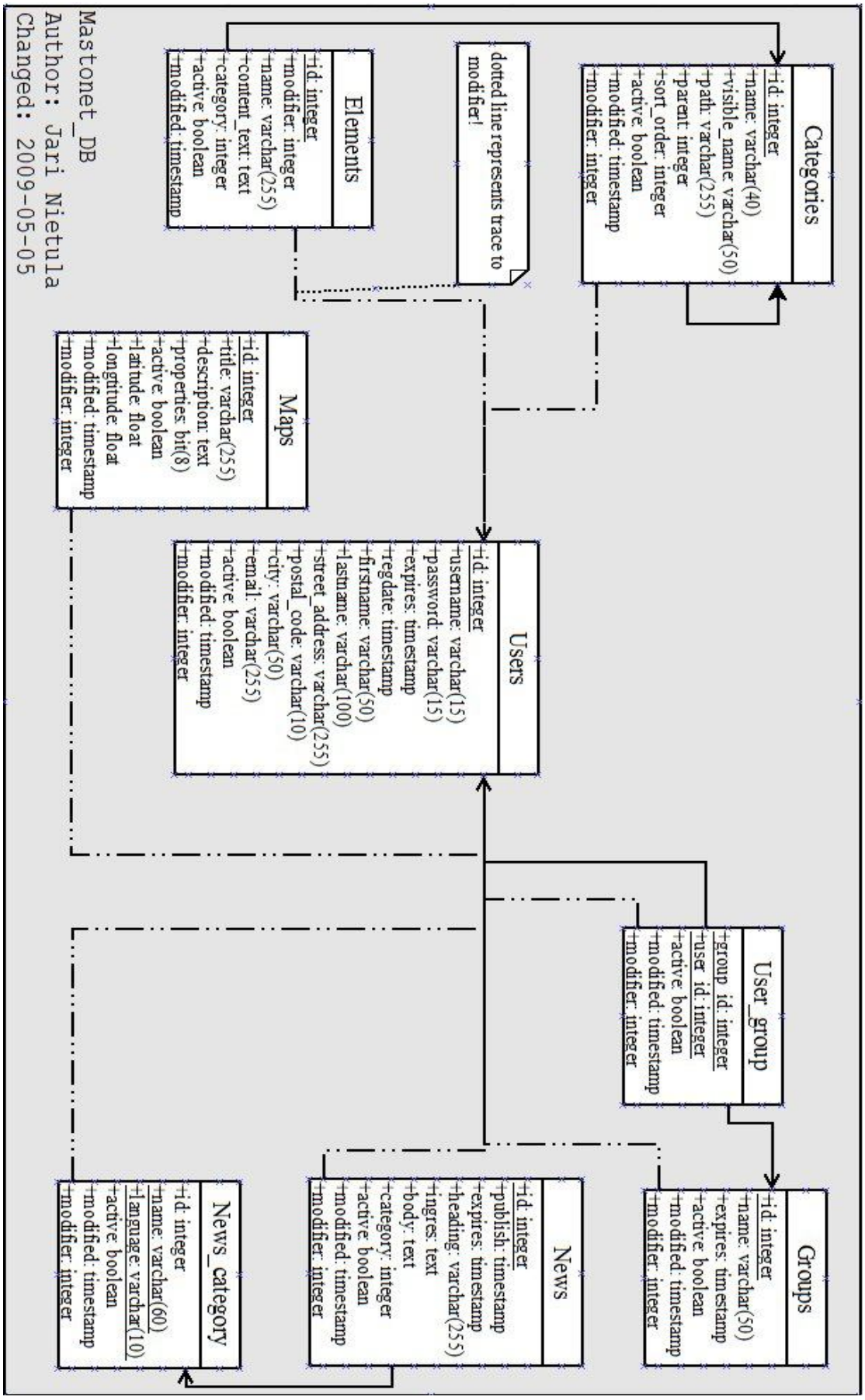
W3School, Xsl 2009 [verkkojulkaisu] XSLT Instruction [viitattu 6.5.2009] Saatavissa: http://www.w3schools.com/Xsl/xsl_intro.asp

W3School, XPath 2009 [verkkojulkaisu] XPath Tutorial [viitattu 6.5.2009] Saatavilla: http://www.w3schools.com/xpath/default.asp

W3School XSL-FO 2009 [verkkojulkaisu] XSL-FO Tutorial [viitattu 6.5.2009] Saatavilla: http://www.w3schools.com/xslfo/default.asp

XMLHttpRequest 2009 [verkkojulkaisu] Dynamic HTML and XML: The XMLHttpRequest Object [viitattu 21.4.2009] Saatavissa: http://developer.apple.com/internet/webcontent/xmlhttpreq.html

LIITE 1


Mastonet_DB
Author: Jari Nietula
Changed: 2009-05-05
Categories
+id: integer
+name: varchar(40)
+visible_name: varchar(50)
+path: varchar(255)
+parent: integer
+sort_order: integer
+active: boolean
+modified: timestamp
+modifier: integer
dotted line represents trace to modifier!
Elements
+id: integer
+modifier: integer
+name: varchar(255)
+content_text: text
+category: integer
+active: boolean
+modified: timestamp
Maps
+id: integer
+title: varchar(255)
+description: text
+properties: bit(8)
+active: boolean
+latitude: float
+longitude: float
+modified: timestamp
+modifier: integer
Users
+id: integer
+username: varchar(15)
+password: varchar(15)
+expires: timestamp
+regdate: timestamp
+firstname: varchar(50)
+lastname: varchar(100)
+street_address: varchar(255)
+postal_code: varchar(10)
+city: varchar(50)
+email: varchar(255)
+active: boolean
+modified: timestamp
+modifier: integer
User_group
+group_id: integer
+user_id: integer
+active: boolean
+modified: timestamp
+modifier: integer
Groups
+id: integer
+name: varchar(50)
+expires: timestamp
+active: boolean
+modified: timestamp
+modifier: integer
News
+id: integer
+publish: timestamp
+expires: timestamp
+heading: varchar(255)
+ingres: text
+body: text
+category: integer
+active: boolean
+modified: timestamp
+modifier: integer
News_category
+id: integer
+name: varchar(60)
+language: varchar(10)
+active: boolean
+modified: timestamp
+modifier: integer