VAASAN AMMATTIKORKEAKOULU
VASA YRKESHÖGSKOLA
UNIVERSITY OF APPLIED SCIENCES

TEEMU PUUKKO

INTERNETSIVUSTON KEHITYSTÄ MOBIILISELAIMELLE

Tekniikka ja liikenne

2010

VAASAN AMMATTIKORKEAKOULU

Tietotekniikan koulutusohjelma

TIIVISTELMÄ

Tekijä	Teemu Puukko
Opinnäytetyön nimi	Internetsivuston kehitystä mobiiliselaimille
Vuosi	2009
Kieli	Suomi
Sivumäärä	41 + 2 liitettä
Ohjaaja	Pirjo Prosi

Opinnäytetyö käsittelee internetsivuston optimointia mobiililaitteille käytettäväksi. Työssä perehdytään paluukuorma.fi internetsivustosta mobiililaitteille kehitettävän version kehitysprosessiin. Työn toimeksiantaja oli Vaasalainen ohjelmistotalo Waasa Zone Internet Solutions Avoin yhtiö, jonka tarkoituksena on tarjota tien päällä työskenteleville kuljettajille apuvälineenä palvelu, jolla kuljettaja kykenee paikantamaan vapaita rahteja ja varaamaan ne toimitettavikseen tai vastaavasti merkitä ne toimitetuiksi. Palvelu toteutettiin vuoden 2009 kesän aikana, käyttäen PHP-ohjelmointikieltä ja MySQL-tietokantaa avuksi.

Asiasanat mobiiliselain, PHP, MySQL

VAASAN AMMATTIKORKEAKOULU

UNIVERSITY OF APPLIED SCIENCES

Technology and Communication

ABSTRACT

Author	Teemu Puukko
Title	Developing a Web Application for Mobile Web Browsers
Year	2009
Language	Finnish
Pages	41 + 2 Attachments
Name of Supervisor	Pirjo Prosi

This thesis deals with conversion of a regular web page to be used in a mobile environment. The project focus is to optimize paluukuorma.fi page suitable for mobile web browsers. The project client is Vaasa Zone Internet Solutions General Partnership, that offers services to logistics companies, by providing the drivers easy access to freight as form of return freight. The service was implemented on summer 2009, by using PHP scripting language and MySQL relation database.

Keywords Mobile web browser, PHP, MySQL

KÄYTETYT MERKINNÄT JA LYHENTEET

AJAX on vanhojen tekniikoiden uudelleenkäyttöä. Tekniikkaa käyttäen webselain voi liikennöidä palvelimen kanssa lataamatta sivua uudestaan.

C++ on Bjarne Stroustrupin 1980-luvun alussa kehittämä ohjelmointikieli. C++ pitää sisällään lähes koko C kielen ja tukee olio-ohjelmointia.

CSS Cascading Style Sheets suomeksi kääntyy yleensä tyylitiedostoksi. W3C:n määrittelemä standardi websivujen tyylitiedostojen kuvauskielestä.

DOM on xml-dokumentista muodostettava puumalli.

FF, lyhenne tarkoittaa firefox nimistä internetselainta.

G, lyhenteellä viitataan Google Grome nimiseen internetselaimeen.

PHP, on ohjelmointikieli, jota käytetään paljon dynaamisten websivujen luonnissa.

HTML on websivujen kuvauskieli

HTTP perusverkkoprotokolla

IE, lyhenteellä tarkoitetaan Internet Explorer nimistä internetselainta.

JavaScript yleisesti webselaimissa suoritettavaa ohjelmakoodia. JavaScript on ECMAScript standardin tulkintaa.

O, lyhenteellä tarkoitetaan Opera nimistä internetselainta.

S, lyhenteellä tarkoitetaan Safari nimistä internetselainta.

XHTML, sama tarkoitus kuin HTML kuvauskielellä, erona että se toteuttaa XML:n vaatimat määritykset.

XML on W3C:n kehittämä tiedonmerkintästandardi.

SISÄLLYS

1 JOHDANTO

Tarkoituksena on luoda paluukuormat.fi sivustosta versio, jota kyettäisiin käyttämään mobiiliselaimilla. Paluukuormasivusto on tarkoitettu käytettäväksi logistiikka-alan yrityksille. Rahtarille tarjotaan kentällä työskennellessään helppo tapa varata paluurahti matkansa varrelta, jotta hänen ei tarvitse ajaa pitkää matkaa takaisin kotiinsa kuormattomana. Työssä käsitellään erityisesti huomioon otettavia asioita kehitettäessä websivustoa mobiiliselaimelle, niiden rajoitteita ja vahvuuksia. Työssä käsitellään myös palvelinpuolen ohjelmistoarkkitehtuuria, menemättä PHP-ohjelmontikielen ominaisuuksiin tarkemmin. Palvelinpuolen ohjelmointikielenä projektissa on käytetty PHP 5:sta ja tämän lisäksi internetselaimissa on käytetty ECMAScript-standardiin pohjautuvia kieliä, kuten JavaScript. Työssä kerrotaan myös ohjelmistossa käytetystä ohjelmistoarkkitehtuurista ja tekniikoista. Tämän lisäksi käsitellään työn kannalta oleellisia World Wide Web Consortiumin laatimia internetstandardeja mobiiliselainten näkökulmasta.

2 WAASA ZONE INTERNET SOLUTIONS AVOIN YHTIÖ

Waasa Zone on internetpalveluiden kehittämiseen perustettu teknologia- ja innovaatioyritys. Yritys on toiminut jo vuodesta 1997. Waasa Zone on erikoistunut Internetverkkojärjestelmien rakentamiseen ja ylläpitämiseen, asiakaspohjaisista lähtökohdista.

3 MOBIILISELAIN

Mobiili-internetselain on suunniteltu erityisesti mobiililaitteille kuten kännyköille tai kämmentietokoneille. Käytännössä monessa tapauksessa mobiiliselain on yksinkertaisesti riisuttu versio täysiverisestä internetselaimesta, kuten Firefox tai Safari. Mobiiliselain on erityisesti suunniteltu näyttämään internetsivuja pienellä näyttöresoluutiolla tehokkaasti. Kuitenkin laitteiden suorituskyvyn takia on jouduttu monesti tekemään kompromisseja selaimen tukemien tekniikoiden suhteen. Harvemmassa kännykässä on käytettävissä Flash, jota esimerkiksi käytetään näytettäessä videoita selaimen kautta. Nykypäivänä on vielä tyypillistä, että monet palvelut on toteutettu mobiililaitteille itsenäisinä ohjelmina. Tässä tilanteessa kuitenkin hävitään selaimen tarjoama etu, eli yhtenäinen toiminta-alusta mobiililaitteissa tai käyttöjärjestelmästä riippumatta tarjottavalle palvelulle.

4 WEB-STANDARDIT

4.1 W3C suosituksia mobiiliselaimille

Word Wide Web Consortium on kansainvälinen pääorganisaatio, joka luo ja muodostaa internetstandardeja. Tietotekniikkaan tutustuneelle varmasti jokin seuraavista kirjainyhdistelmistä kuulostaa tutulta: XML, XHTML, CSS, XSTL, XQuery jne. Kaikki edellä luetellut kirjainyhdistelmät ovat W3C työn tuloksena syntyneitä standardeja, joita tietotekniikassa käytetään hyvin paljon. Esimerkiksi tässä työssä käytetty XHTML-kieli joka on HTML:stä kehitetty www-sivujen merkintä kieli, joka täyttää XML-standardin muotovaatimukset.

W3C on muodostanut mobiiliselaimille tarkoitetun suosituspaketin vuonna 2008 (*Mobile Web Best Practices 1.0*), Dokumentti pitää sisällään pitkän listan eri suosituksia kuinka yleisiä web-tekniikoita tulisi soveltaa käytettäessä mobiiliselaimia. W3C on myös kehittänyt testin, jolla omaa sivuaan voi testata. Testi antaa suoria parannusehdotuksia löytämistään virheistä. Tätä testiä kutsutaan nimellä *W3C mobileOK Checker*.

4.2 HTML ja XHTML

Tiedon esitykseen tarkoitettu HTML aikaisimmista versiosta lähtien on ollut hyvin kirjava kieli, eri selainvalmistajat ovat kehittäneet siihen omia elementtejään ajan saatossa, jotka toisten on sitten täytynyt omaksua tarpeen pakosta. Selaimet tulkitsivat HTML-koodia käyttäjäystävällisesti näyttäen virheellisestikin rakennetun HTML-koodin websivuna. Vaikka XHTML ei juuri eroakaan HTML 4.01 lainkaan, on siinä kuitenkin tiettyjä tärkeitä eroja.

XHTML-elementtien täytyy olla oikealla lailla sisäkkäin seuraavasti:

```
<div><h1>Terve</h1></div>
```

Elementit eivät saa mennä ristiin seuraavasti:

```
<div><h1>Moro</div><h1>
```

XHTML-elementtien tulee olla myös suljettuja, olivatpa ne tyhjiä tai eivät. Elementit tulee kirjoittaa aina pienillä kirjaimilla. Dokumentilla tulee myös olla ainoastaan yksi juurielementti. Elementtien attribuuttien tulee aina olla lainausmerkkien sisällä.

```
<img src=”pallo.jpg” alt=”Kuva pallosta” />
```

Suurin ero HTML 4.01 ja XHTML välillä on, että XHTML käyttää validoinnissaan DTD ja XML scheemaa. Tämä määrittää XHTML:n tarkat tavat dokumentin lähdekoodin muodostamiselle. XHTML 1.1 versio ei enää pidä ollenkaan sisällään ulkoasuun vaikuttavia ominaisuuksia. Kaikki ulkoasuun vaikuttavat asiat on hoidettava CSS-tyylitiedostojen avulla.

4.3 XHTML-dokumentti

Varsinaista XHTML-dokumenttia ennen annetaan tätä dokumenttia koskevia tietoja kuten merkistökoodaus, kehittäjä, kuvaus, kieli ja muita merkintöjä hakukonetta varten. Metatietoa kutsutaan tiedoksi, joka antaa tietoa itse dokumentin tiedoista.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fi" lang="fi>
<head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
        <meta name="description" content="Sivuston kuvaus" />
        <meta name="robots" content="noarchive,index,follow" />
</head>
<body></body>
</html>
```

Yllä oleva dokumentti on esimerkki tyypillisestä XHTML-dokumentista.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

XML-ilmoitus on pakollinen ainoastaan silloin, jos merkistökoodauksena käytetään jotain muuta kuin utf-8 tai utf-16 koodausta.

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

Ensimmäinen rivi sisältää dokumentin tyypin julistuksen, se voi sisältää viittauksen DTD:stä, joka määrittelee dokumentin rakenteen. DTD-määrittely sisältää tiedon kaikista dokumenteissa käytettävissä olevista tageista ja niiden järjestyksestä.

```
<meta name="robots" content="noarchive,index,follow" />
```

Hakurobotit indeksoivat sivua automaattisesti. Niille voidaan kertoa meta-elementissä kuinka usein sivu päivittyy. Hakurobottien toimintaa voi myös ohjata sivuston juurihakemistoon sijoitettavalla robots.txt tiedostolla.

```
<meta name="description" content="Sivuston kuvaus" />
```

Tämäkin kenttä on tarkoitettu käytännössä hakuroboteille.

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

Yhteensopivuussyistä on myös hyvä ilmoittaa merkistökoodaus uudestaan meta-elementissä. Meta-elementtejä yleensäkin käytetään kuvaanmaan dokumentissa esiintyvää tietoa hakumoottoreille. Ne voivat myös sisältää tietoa mobiiliseilaimelle siitä kuinka tieto tulisi esittää sivustolla.

4.4 CSS

CSS-tyylitiedostoja käytetään erityisesti XHTML-dokumenttejen kanssa. Mutta ne soveltuvat myös muiden arkkitehtuuristen dokumenttien tyylin kuvaamiseen, kuten XML tai MathML. CSS:llä voidaan muokata monipuolisesti dokumentin esitystapaa, sillä voidaan myös antaa puhesyntetisaattoreille ohjeita puheen äänenkorkeudesta ja äänensävystä. Muokatessa dokumentin esitysasua käsitellään

jokaista elementtiä suorakulmiona. Suorakulmion sisällä on tavallaan kolme suorakulmiota muodostavaa elementtiä: Marginaali, ulkoviiva ja sisennys.

4.5 ECMAScript

ECMAScript pohjautuu alunperin moniin eri tekniikoihin, mutta kaksi tunnetuinta näistä on Netscapen kehittämä JavaScript ja Microsoftin kehittämä JScript. JavaScriptin kehitti alunperin Brendan Eich Netscapella, jonka Netscape julkaisi maaliskuussa 1996 Netscape Navigator 2.0-selaimensa mukana. Microsoft julkaisi JScriptin Internet Explorer 3.0-selaimen mukana saman vuoden elokuussa. (ECMAScript 3rd Edition specification 2009.)

Standardin kehittäminen aloitettiin marraskuussa 1996. Ensimmäinen valmis versio ECMA-262-standardista hyväksyttiin ECMA General Assembly kokoontumisessa kesäkuussa 1997. Joulukuussa 1999 julkaistiin ECMA-262 standardin kolmas ja tällä hetkellä uusin virallinen versio. (ECMAScript 3rd Edition specification 2009.)

Voidaan sanoa, että JavaScript sekä JScript ovat ECMA-262 standardin tulkintaa. Ominaista ECMAScriptille on sen tukema dynaaminen tyypitys ja prototyyppipohjainen ohjelmointitapa.

Dynaaminen tyypitys tarkoittaa, että muuttujien tyyppiä ei sidota itse muuttujaan vaan paremminkin sen sisältöön. Esimerkiksi muuttuja a voisi olla aluksi määritelty numeroksi ja tämän jälkeen muuttua merkkijonoksi. Kieli tukee olio-ohjelmointia vaikka siitä puuttuu Javalle tai C++:lle tyypillinen luokkarakenne täysin. ECMAScript tukee moninperintää ja perintä voi tapahtua ajonaikaisesti kahdella tavalla, joko yksittäinen olio perii toisen olion ominaisuudet tai luokka perii toisen luokan ominaisuudet. Kielen syntaksi on tarkoituksella tehty muistuttamaan Javaa ja C++:saa, esimerkiksi while, for ja switch-rakenteet ovat samankaltaisia.

5 PALVELINPUOLEN TEKNIIKOITA

5.1 MySQL

Nimensä MySQL on saanut yhtiön perustajan, suomalaissyntyisen Michael Wideniuksen My tyttären mukaan. SQL osa tulee sanoista Structured Query Language. MySQL:aa kehittää ruotsalainen yritys MySQL AB, jonka osti tammikuussa 2008 Sun Microsystems. (MySQL 5.0 reference manual. 2009.)

MySQL on relaatiotietokantajärjestelmä, joka on kirjoitettu käyttäen C/C++:ssa, se on avoimen lähdekoodin ohjelmisto. Sitä jaellaan kahtena eri versiona, joko eikaupallisena MySQL Community Server versiona, joka on ilmainen ja käyttää GNU General Public License ehtoja tai kaupallisena MySQL Enterprise versiona, joka tuo asiakkaan kattavan tuotepiiriin. Se on suunniteltu monen käyttäjän tietokannaksi, ja sille löytyy ohjelmointirajapinta hyvin monelle ohjelmointikielelle, joihin kuuluvat C/ C++, Eiffel, Java, Perl, PHP, Python, Ruby ja TLC. (MySQL 5.0 reference manual. 2009.)

5.2 JSON

JSON tulee sanoista JavaScript Object Notification, tekniikka on kevyt tiedonsiirtomenetelmä. Tekniikan idea on sama kuin monilla web-palveluilla, mutta erona on, että tekniikka ei käytä ollenkaan XML:lää tiedon koodaamisessa. JSON on huomattavasti tehokkaampi tapa siirtää tietoa verkon yli kuin SOAP tai XML RCP web-palvelutekniikat. JSON ei rajoitu ainoastaan JavaScriptiin, vaan on huomattavasti yleiskäyttöisempi, voit esimerkiksi siirtää Java oliota suoraan verkon yli PHP:lle tai päinvastoin. JSON käy erinomaisesti pienen tietomäärän siirtämiseen.

5.3 PHP

PHP on lyhenne sanoista Hypertext Preprocessor, se on avoimen lähdekoodin ohjelmointikieli ja se on kehitetty erityisesti web-ohjelmointiin. PHP:n syntaksi muistuttaa paljolti C, JAVA ja Perlin syntaksia.

Ensimmäisen version PHP:stä tai PHP/FI:stä kehitti Rasmus Lerdorf intenetsivulleen seuratakseen kävijämääriä sivustollaan. Versiolla oli tuki joillekin yksinkertaisille funktiolle, lomakkeelta tulevien tietojen käsittelyyn ja mSQL tietokannalle. (Vaswani, Vikram. 2005 5.)

PHP/FI koostui pienestä määrästä Perl skriptejä, joita Lerdorf kutsui nimellä Personal Home Page Tools (History of PHP and related projects 2009.)

Vuoteen 1997 mennessä, PHP/FI 2.0 oli PHP uudelleen kirjoitettu käyttämällä C:tä. Käyttäjiä PHP:llä oli tässä vaiheessa jo useita tuhansia ympäri maailmaa. Tässä vaiheessa se oli käytössä 50000 verkko-osoitteessa. Tämä vastasi noin 1%:a internetin silloisista verkko-osoitteista. (History of PHP and related projects 2009.)

Samana vuonna julkaistiin PHP 3.0, joka oli täysin uudelleen kirjoitettuna Andi Guntmansin ja Zeev Suraski toimesta. Heidän todettuaan PHP 2.0 suorituskyvyn riittämättömyyden suuren luokan kaupallisiin ohjelmistoihin. PHP:n nimenkin merkitys muuttui PHP:Hypertext Preprocessor. PHP 3.0 sisälsi rajoittuneen olio-ohjelmoinnin, parantuneen resurssien hallinnan ja tietoturvan sekä HTTP-istunnot (History of PHP and related projects 2009.)

PHP 5.0 julkaistiin vuonna 2004, se sisälsi parannetun olio-ohjelmointituen, poikkeusten hallinnan, parannetun merkkijonojen hallinnan, XML- ja web-service tuen ja monia muita ominaisuuksia. (History of PHP and related projects 2009.)

5.4 phpDocumentator

PhpDocumentatoriin viitataan yleensä PHPdoc tai PHPdocu nimityksillä. PHPdoc on PHP:n nykyinen standardi autodokumentaatiotyökalu PHP-lähdekoodille. Työkalu itsessäänkin on toteutettu PHP:llä ja sitä voidaan käyttää joko komentoriviltä tai selainkäyttöliittymän kautta. Työkalulla voidaan luoda ammattimainen dokumentaatio PHP-lähdekoodista. PHP:llä kirjoitetun lähdekoodin dokumentaatio on monella tapaa samankaltaista kuin Javalla kirjoitetun lähdekoodin, sillä PHPdoc dokumentointityökalu ottaa mallia paljon

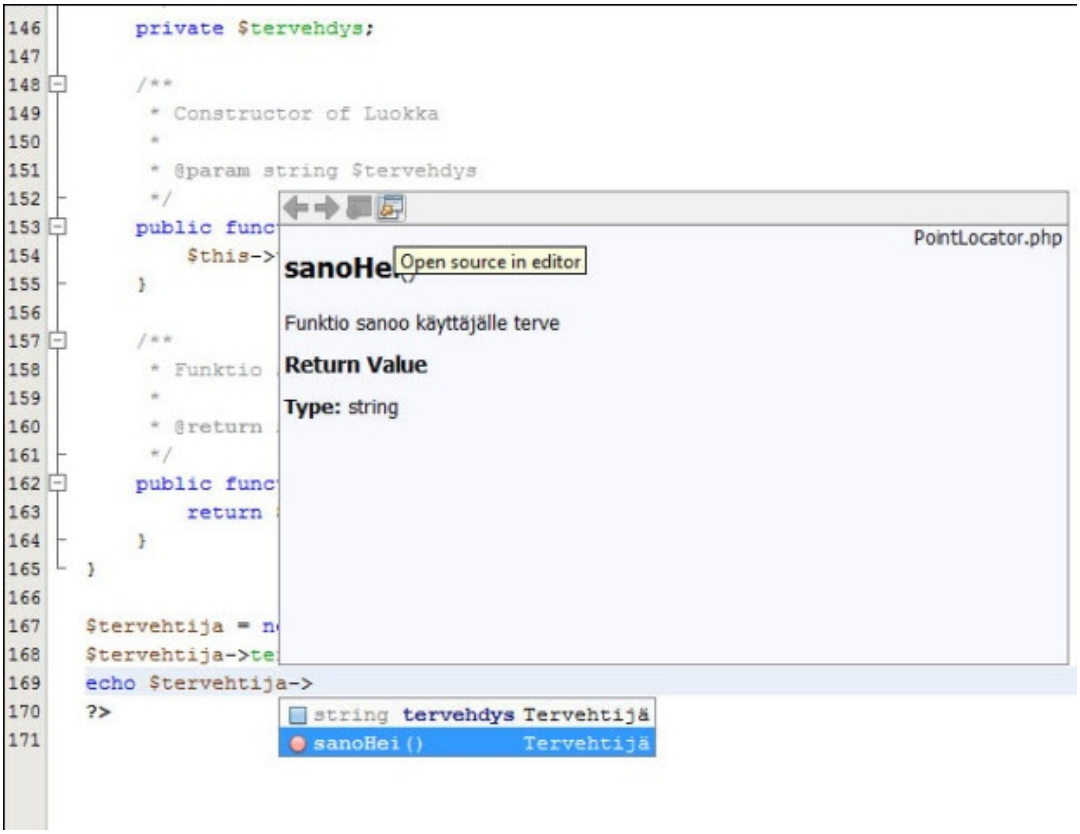
Sun microsystemsin Javalle kehittämästä dokumentaatiotyökalusta, nimeltään Javadoc. Lähdekoodista voidaan luoda dokumentaatiokooste automaattisesti, kunhan lähdekoodin dokumentoinnissa on käytetty PHPdoc:sin vaatimaa syntaksia. Useat PHP-kehitystyökalut, kuten Netbeans ja Eclipse, osaavat indeksoida koodirakenteita tutkimalla lähdekoodin dokumentaatiota. Se on suuri apu kehitystyössä. Monet sovelluskehitystyökalut hyödyntävät ennalta määrätyllä tavalla dokumentoitua ohjelmakoodin rakennetta. PHP on ohjelmointikielenä hyvin löyhästi tyypitetty, joten sovelluskehitystyökalut eivät muuten osaisi löytää esimerkiksi funktioiden parametrien tietotyyppejä, joita ohjelmoija haluasi funktiossa käyttää.

```
<?php
/**
 * Description of Luokka
 *
 * @author TeeMuki
 * @property string $tervehdys
 *
 * @package mallipaketti
 */
class Tervehtijä {

    /**
     * Constructor of Luokka
     *
     * @param string $tervehdys
     */
    public function __construct($tervehdys) {
        $this->tervehdys = $tervehdys;
    }

    /**
     * Funktio sanoo käyttäjälle hei
     *
     * @return string
     */
    public function sanoHei() {
        return $this->tervehdys;
    }
}

$tervehtija = new Tervehtijä("Terve!");
```

```
$tervehtija->tervehdys = "Moi!";
echo $tervehtija->sanoHei();
?>
```

Esimerkki PHP-lähdekoodin dokumentoinnista.

Kun koodille kirjoitetaan edellä mainitulla tavalla, helpottaa se ohjelma koodin uudelleenkäyttöä. Kuvassa 1 esimerkki kuinka netbeans osaa tuoda esille dokumentaation edellisestä Tervehtijä luokasta.



Kuva 1 Kuvankaappaus netbeanssin avuliaasta dokumentaation luvusta

5.5 Käytetyn selaimen tunnistaminen palvelinpuolella

Laitekohtaisesti suoritettavaa ohjelmakoodia ei yleensä haluttaisi joutua kirjoittamaan silloin, kun ollaan tekemässä ohjelmistoa selaimelle. Mutta jos halutaan tukea ominaisuuksia, joita vain osa laitteista tukee. Päädytään välttämättä tilanteeseen, jossa laitteistokohtaista koodia syntyy. Tästä syntyy myös tarve tunnistaa laite luotettavasti, jotta vältytään mahdollisilta virhetilanteilta.

Jotta käyttäjälle voidaan tarjota mahdollisimman hyvä käyttökokemus sivusta voidaan palvelinpuolella yrittää tunnistaa käyttäjän käyttämä mobiiliselain. Tunnistus ei ole koskaan sataprosenttisen varma, koska selain saattaa lähettää muunneltuja tunnistetietoja ja esittäytyä toisena selaimena. Tarkan tunnistamisen tekeminen on suhteellisen monimutkaista mutta mahdollista. Selain lähettää jokaisella pyynnöllään HTTP-standardin määrittelemän mukaiset tunnisteet.

Tärkein tunnistetieto kohdasta "User Agent" näyttää seuraavalta:

```
Mozilla/5.0 (Windows; U; Windows NT 6.1; fi; rv:1.9.1.3) Gecko/20090824
Firefox/3.5.3 (.NET CLR 3.5.30729) FirePHP/0.3
```

Tunnistetiedon muoto poikkeaa joka selaimella ja käyttöjärjestelmällä melkoisesti. Mutta siitä voidaan selvittää yleensä mitä käyttöjärjestelmää ja selainta käyttäjä käyttää. Edellisestä esimerkistä saadaan selville, että käytössä oli Firefox 3.5 selain. Gecko viittaa firefoxin käyttämään renderöintimoottoriin. Käytössä oli Windows 7 käyttöjärjestelmä.

Esimerkiksi Opera-selain kykenee naamiomaan itsensä IE-selaimeksi lähettämällä palvelimelle IE:n käyttämän tunnistetiedon.

Selaimen ja laitteen tunnistaminen saattaa olla työlästä, mutta tähän on olemassa valmiiksi rakennettuja tietolähteitä ja ohjelmointirajapintoja, joita voidaan ottaa käyttöön tarpeen tullen. Yksi hyvä esimerkki on DeviceAtlaksen tarjoama maksullinen tietokanta. Kanta tarjoaa tiedot yli tuhannesta laitteesta. Laitteesta listataan 88 ominaisuutta. Palvelu toimii omalla palvelimellasi, johon DeviceAtlas tekee päivityksiä joko kuukauden, viikon tai päivän välein käytetyn lisenssityypin mukaan. (Device Atlas, Mobile Device Intelligence 2009.)

6 KÄYTTÖLIITTYMÄ

6.1 Käyttöliittymän suunnittelu mobiiliselaimelle

Käyttöliittymä on se mikä on loppukäyttäjälle itse tuote. Käyttäjää ei kiinnosta millä tekniikalla palvelu on toteutettu. Ainoa mistä käyttäjä käytännössä välittää on käyttöliittymä. Sivun suunnittelussa täytyy sen erityistä tarkkuutta kohdistaa itse käyttöliittymän suunnitteluun. Käyttöliittymä käytännössä ratkaisee kuinka helppokäyttöiseksi järjestelmä saadaan. Mobiiliselainten käyttöliittymät voidaan jakaa kolmeen pääkategoriaan: (Mobile Web Best Practices 1.0 2009.)

Valintapohjainen käyttöliittymä, jossa valittavat elementit voivat olla esimerkiksi tekstinsyöttölaatikoita tai linkkejä. Valintojen välillä siirrytään hyppien valinnasta toiseen, kun valinta tulee valituksi tuo elementti korostetaan. Yleensä tällainen käyttöliittymää pohjautuu painikeohjaukseen (Mobile Web Best Practices 1.0 2009.)

Kursoripohjainen käyttöliittymä on tuttu työpöytäkoneen työpöydän käyttötavasta, kun käyttäjä siirtää hiirtä fyysisesti työpöydällä niin kursori siirtyy tietokoneen työpöydällä haluttuun suuntaan. Ainoana erona yleensä mobiililaitteille on se, että kursoria ohjataan painikkeilla, eikä hiirellä haluttuun suuntaan. Tällaista käyttöliittymää käytettäessä on valittavat elementit syytä korostaa silloin kun kursori on viety niiden päälle. Tämä onnistuu helposti käyttäen CSS-tiedostoa (Mobile Web Best Practices 1.0 2009.)

Kosketuspohjainen käyttöliittymä, jota käytetään kosketusnäytöllä. Näyttö toimii yleensä sormen tai kynänkosketuksen avulla. Tällaista käyttöliittymää käytettäessä on hyvä pitää valittavien elementtien välissä tarpeeksi suuri rako, että käyttäjä ei tule valinneeksi turhaan väärää elementtiä (Mobile Web Best Practices 1.0 2009.)

Käyttöliittymä voidaan suunnitella jokaiselle tyypille erikseen tai tehdä vaadittavia kompromisseja käyttöliittymän suhteen, jolloin kaikkia laitteita palvellaan samalla tavalla. Mobiililaitteille on omaista se, että näyttö on hyvin

kapea, jopa ainoastaan 120 pikseliä leveä. Tämä pitää ottaa erityisesti huomioon, koska ei ole hyvä, että käyttäjä joutuu rullaamaan näkymää vaakasuunnassa.

Sivun ulkoasua suunniteltaessa kannattaa myös ajatella miltä sivu näyttää päivänvalossa mobiililaitteella. Riittävä värikontrasti helpottaa sivun selaamista vaihtelevissa valaistusolosuhteissa ja monesti mobiililaitteiden näyttöjen värierottelukyky on suhteellisen rajoittunut. Lisäksi on hyvä välttää värikombinaatioita, jotka estävät värisokeita lukemasta tekstiä.

6.2 Käyttöliittymän lokalisointi

Lokalisointi tarkoittaa ohjelman tai tuotteen muokkaamista vieraaseen ympäristöön tai kulttuuriin sopivaksi. Ohjelmistokehityksen kannalta lokalisointi on heti alusta asti huomioon otettava asia. Lokalisoinnin takia olisi hyvä, että kaikki sisältö erotettaisiin ohjelmakoodista erilleen. Näin päästäisiin tilanteeseen, jossa ohjelmiston kääntämisessä vieraalle kielelle ei tarvitsisi kajota lainkaan ohjelmakoodiin, vaan koko käyttöliittymän käännösprosessi hoituisi muuntelemalla erillisiä lokalisaatioon liittyviä asetustiedostoja.

Teknisesti lokalisoinnissa täytyy ottaa huomioon merkistökoodaus. Merkistökoodaus on yksinkertaisuudessaan sovittu tapa järjestää numerot, kirjaimet ja muut merkit taulukkoon siten, että jokaista numeroa vastaa jokin merkki. Vielä tänäkin päivänä käytössä on 7-bittinen merkintästandardi ASCII, johon mahtuu 128 merkkiä. Tämä ei käytännössä riitä kovin pitkälle lokalisointitarkoituksissa. Tähän merkistöön ei edes mahdu skandinaaviset ä, ö, ja å kirjaimet. Tätä varten ASCII standardia laajennettiin 256 merkkiseksi. Varsinkin monet aasialaiset kielet, jotka käyttävät kirjoituksessaan tavu-, tai sana-merkistöä, saattavat sisältää kymmeniätuhansia merkkejä. Tämän vuoksi on kehitetty yhteiset UTF-8 ja UTF-16 standardit. Molemmat standardit ovat yhteensopivia ASCII-koodauksen kanssa siten, että ensimmäisten 256 merkin järjestys on sama kuin ASCII-standardissa.

Lokalisoinnissa merkistökoodaus ei ole ainoa huomioon otettava asia vaan on myös huomioitava ajan, lämpötilan, valuutan ja eri suureiden mittaustavat. Lokalisointi on monitahoinen haaste.

6.3 Semanttiset internet-osoitteet

Semaattisella internet-osoitteella tarkoitetaan tavallista internet-osoitetta, joka on muotoiltu käyttäjäystävälliseen muotoon. Monesti dynaamisesti generoitu internet-osoite on vaikeaselkoinen lukuisten eri ohjausparametrien takia. Vaikka parametreja ei voida välttämättä poistaa voidaan internet-osoite muotoilla käyttäjäystävällisesti, jonka palvelin osaa muotoilla takaisin koneen ymmärtämään muotoon. Käyttämällä tällaista tekniikkaa on myös helppo piilottaa palvelinalustassa käytetty tekniikka mahdollisilta urkkijoilta, joten tekniikka tarjoaa myös tietoturvaa. Seuraavaksi esimerkki:

Vaikealukuinen www-osoite:

http://blogi.com?page=321

Voidaan esittää muodossa:

http://blogi.com/sivu/321/

Tämä antaa käyttäjälle huomattavasti paremman mahdollisuuden "arvata" haluamansa internet-osoitteen.

Apatchen palvelimella on mahdollista tehdä muutos, joka muuttaa helppolukuisen internet-osoitteen palvelimen ymmärtämään muotoon, käyttäen sen mod_rewrite moduulia. Tekniikka käyttää hyväkseen säännöllisiä lausekkeita kun uudelleenkirjoitussääntöjä luodaan.

Taulukko 1 Malliesimerkki .htacsess tiedostosta

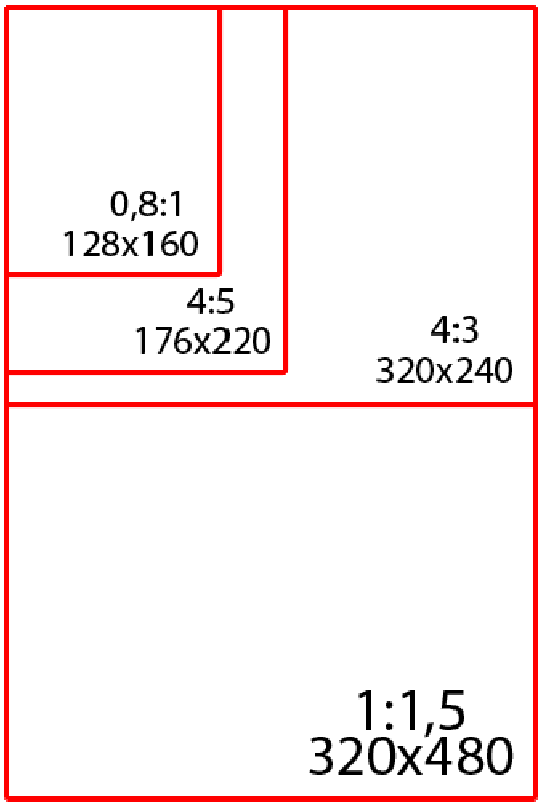
```
1. RewriteEngine On

2. RewriteBase   C:/xampp/htdocs

3. RewriteRule   ^rahti/([0-9]+)/$  index.php?rahti=$1
```

Ensimmäisellä rivillä tiedostossa kytketään URL:ien uudelleenohjauksen päälle. Kolmannella rivillä asetetaan absoluuttinen tiedostopolku, johon palvelupyyntö tullaan ohjaamaan. Vihdoin viidennellä määrittelemme uudelleen kirjoitussäännön, joka alkaa merkistä '^' ja loppuu merkkiin '$'. Dollari merkin jälkeen määritellään palvelimen fyysinen osoite, jonne pyyntö ohjataan. Sulkujen sisälle jäänyt kohta erotetaan säännöllisestä lausekkeesta, eli tästä kohdasta '([0-9]+)' ja siirretään lausekkeen kohtaan '$1'-tilalle.

7 SIVUSTON OPTIMOINTI MOBIILISELAIMILLE

7.1 Mobiililaitteiden kuvasuhteet

Perinteisesti tietokoneen näytöltä katsottaessa näkymä on ollut matala ja leveä. Suurin osa mobiililaittteista on varustettu kapealla, mutta korkealla näytöllä, tämä olisi myös otettava huomioon sivuston ulkoasussa. Esimerkiksi iPhonen kuvasuhde pystyasennossa on 1:1,5, kun taas tietokoneen näytön kuvasuhde voi olla jopa 16:10. Tämän lisäksi asiaa hankaloittaa, että monet laitteet kykenevät kääntämään kuvaa laitteen asennon mukaan, jolloin myös kuvasuhde muuttuu.



Kuva 2 Tyypillisiä resoluutiota ja kuvasuhteita

Yllä mainituista resoluutiosta yleisin on 320x240 resoluutio. Se on hyvin nopeasti yleistymässä kosketusnäytöllisten laitteiden tullessa yhä yleisimmiksi.

7.2 Navigointitavan huomioiminen ulkoasun suunnittelussa

Mobiiliselaimella navigoidessa internetsivustolla on olemassa kolme päätapaa, joilla navigointi suoritetaan: kosketus-, valinta- ja kursoripohjainen navigointi. Jokaisen käyttötavan käyttöä voidaan helpottaa ottamalla navigointitavat huomioon sivuston ulkoasun suunnittelussa.

Käytettäessä sivustoa kosketuspohjaisesti on hyvä, että sivuston elementit on sijoitettu tarpeeksi kauaksi toisistaan eli jokaisella elementillä on jätetty riittävä marginaali, jotta vältytään virhepainalluksilta.

Kun käyttäjä navigoi sivustolla käyttäen valintapohjaista navigointitapaa, täytyy sillä hetkellä valittua komponenttia korostaa riittävän selkeästi, jotta käyttäjä

tietää missä valinta sijaitsee. Tämä tapahtuu esimerkiksi värittämällä valitun elementin reunat kirkkaalla ja korostavalla värillä.

Erityisesti, kursoripohjaista navigointitapaa käyttävää käyttäjää silmälläpitäen on hyvä miettiä linkkien sijoittelua sivustolla. Voidaan esimerkiksi miettiä, olisiko järkevää sijoittaa päälinkit sivuston ylä- että alaosaan. Tällä saatetaan välttyä turhalta kursorin liikuttelulta.

7.3 Tiedonsiirron optimointi

Sivuston latautumiseen kuluvaan aikaan vaikuttavat tekijät koostuvat monesta osatekijästä, mutta merkittävin niistä on yleensä tiedostojen siirtämiseen palvelimelta selaimelle kuluva aika. Itse sivuston XHTML-koodin generointi paluukuormatapauksessa kestää alle 50-100 ms, mutta tiedoston siirrosta syntyy muutamien sekunnin latausaika käyttäessä 3g-verkkoa.

Latausaikaa saadaan kuitenkin lyhennettyä monella eri tavalla. Esimerkiksi siirrettävä JavaScript-koodi voidaan pakata ennen siirtoa poistamalla koodista turhat välimerkit ja turhat rivinvaihdot. Tällä voidaan vähentää keskimäärin 30%:a siirrettävän JavaScript-tiedoston kokoa. Tähän tarkoitukseen on kehitetty monia sovelluksia.

Käytettävät kuvat voidaan pakata tehokkaasti käyttäen PNG bittikarttagrafiikan tallennusformaattia, joka on W3C-standardi. PNG:n etuna, JPEG-formaattiin nähden on sen tukema läpinäkyvyys, jota voidaan käyttää hyväksi sivuston ulkoasussa ja sen lisäksi PNG-formaatti sisältää kehittyneemmän kuvanpakkausalgoritmin. (Portable Network Graphics Specification 15.12.2009.)

Toisin kuin normaalisti toimiessa, on mobiiliselaimella toimiessa perusteltua pitää kaikki tyylimäärittelyt yhdessä CSS-tiedostossa ja luonnollisesti sekin pidetään mahdollisimman lyhyenä ja yksinkertaisena.

Nykyiset mobiililaitteet kykenevät täysin hyvin mallintamaan XHTML-dokumentteja, joten sen käyttäminen on hyvin suositeltavaa.

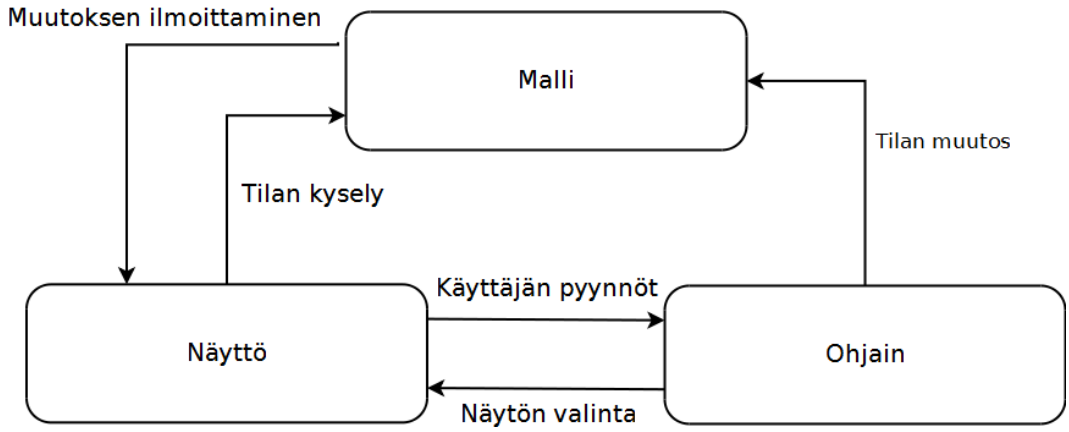
8 MODEL VIEW CONTROLLER

Hyvän ohjelmistoarkkitehtuurin käyttäminen edistää lähdekoodin uudelleenkäytettävyyttä ja ohjelmiston ylläpitämisen helpottamista. Siksi on hyvin suositeltavaa miettiä ohjelmistoa suunniteltaessa, minkälaista ohjelmistoarkkitehtuuria kannattaisi käyttää.

MVC on ohjelmistosuunnittelussa käytetty ohjelmistoarkkitehtuuri. Arkkitehtuurin ideana on erottaa käyttöliittymä sovellustiedoista. Mallia käytetään erityisesti graafisten käyttöliittymien suunnitteluun ja niiden ohjelmointiin. MVC soveltuu myös hyvin interaktiivisin web-sovelluksiin, joissa käyttäjät ovet vuorovaikutuksessa sovelluksen kanssa.

MVC koostuu kolmenlaisista luokista: malleista, jotka määrittelevät sovelluksen tietorakenteet, näkymistä jotka esittävät tiedon ja kontrollereista jotka pitävät sisällään sovelluksen yleisen toimintalogiikan

Ennen kuin MVC-malli yleistyi, nämä oliot niputettiin käyttöliittymän suunnittelussa yhteen. MVC irrottaa ne toisistaan, jotta ohjelmiston muunneltavuus ja uudelleenkäytettävyys paranisi. (Gamma, Helm, Johnson & Vlisides 2001, 5.)



Kuva 3 MVC mallin toimintaperiaate

MVC-malli koostuu kolmentyyppisistä luokista, jotka hoitavat omaa tehtäväänsä sovelluksessa. Seuraavaksi esimerkki siitä miten malli toimii käytännön tasolla.

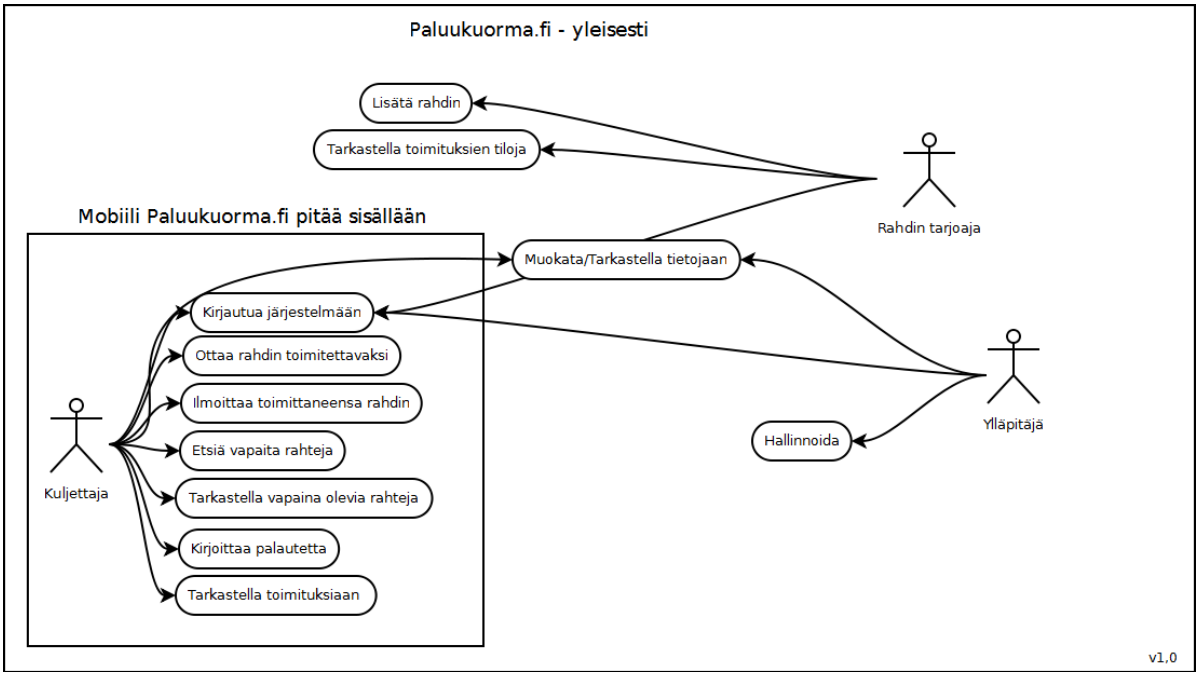
Kun käyttäjä kirjoittaa selaimeen osoitteen tämä aiheuttaa sen, että selain lähettää HTTP-kutsun, jonka palvelin antaa pyynnön kontrolleriluokan käsiteltäväksi. Kontrolleri voi tarvittaessa pyytää malliluokkaa hakemaan tietokannasta tietoa, jonka kontrolleri välittää käytettävälle näkymälle. Kun näin on tapahtunut täytyy tieto esittää, tämä tehtävä kuuluu sillä hetkellä käytössä olevalle näkymälle.

Tällä tavalla jäsennelty koodi on helppo ylläpitää ja uudelleenkäyttää tarvittaessa, koska ohjelmistologiikka on hajautettu selkeästi erilleen. Pienissä projekteissa tällaisen ohjelmistoarkkitehtuurin käyttäminen kuitenkin saattaa monimutkaistaa turhaan ohjelmistokoodia, mutta projektin koon suurentuessa malli selkeyttää ohjelmakoodin rakennetta huomattavasti. Arkkitehtuurin käyttäminen toisaalta mahdollistaa jokaisen kerroksen testaamisen erikseen, omana kokonaisuutenaan.

9 SUUNNITTELU JA TOTEUTUS

9.1 Lähtökohdat

Lähtökohtana oli toteuttaa palvelu, jolla kentällä oleva kuljettaja voi itsenäisesti tarkkailla ja seurata rahtejaan mobiililaitteella. Pyyntö paluukuorma.fi:n kehittämiseen tuli alalla toimivilta tahoilta. Tämän työn tarkoituksena oli kehittää jo toiminnasta olevasta paluukuormasta mobiiliselaimille soveltuva versio. Palvelinalustana toimi kolmannen osapuolen tarjoama palvelin, jossa oli tuki PHP:lle ja mahdollisuus käyttää MySQL-tietokantaa. Tietokantarakenne oli valmiiksi määritelty.



Kuva 4 Use Case kaavio paluukuorman toiminnoista

Mobiilipuoli on kehitetty kuljettajien käytettäväksi, joten suurin osa mobiilipuolen toiminnoista on ainoastaan kuljettajien käytössä. Kirjautuminen järjestelmään toimii käyttäjätunnuksen ja salasanan avulla. Kuljettaja kykenee myös varaamaan rahdin, mutta ei kuitenkaan perumaan varausta suoraan järjestelmän kautta väärinkäytön ehkäisemiseksi. Tämän lisäksi kuljettaja voi kuitata rahdin kuljetetuksi ja kirjoittaa palautetta suoraan järjestelmän kautta. Kaiken tämän

tehtyään kuljettaja voi halutessaan katsella raporttia kuljettamistaan rahdeista tai halutessaan kirjoittaa rohkaisevaa palautetta paluukuorma.fi:n ylläpidolle.

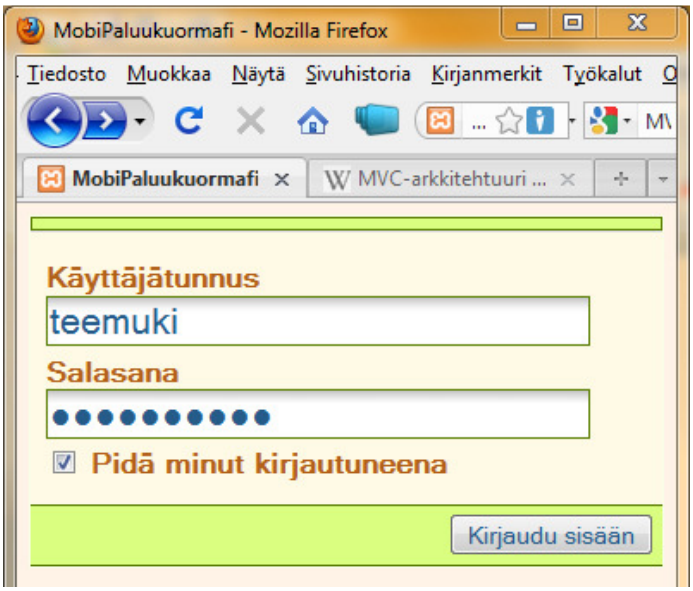
9.2 Toteutus

Nykyisen järjestelmän toteutuksen takia mobiilipuolen toteutus oli koodattava kokonaan uusiksi eli käytännössä tämä tarkoitti sitä, että mobiilipuolen suunnittelu täytyi aloittaa alusta, eikä vanhaa koodia voitu käyttää tehokkaasti uudelleen. Suunnittelun tavoitteina oli luoda helposti ylläpidettävä ja helposti jatkokehitettävä ympäristö. Melko selkeä valinta ohjelmistoarkkitehtuuriksi oli palvelin puolella ja PHP-ohjelmistoissa yleisesti käytetty MVC-arkkitehtuuri.

Ohjelmiston luokat jaetaan kolmeen kansioon MVC-arkkitehtuurin määrittelemällä tavalla: malleihin, näkymiin ja ohjaimiin. Näiden lisäksi löytyy myös ulkopuolisia kirjastoja, jotka on sijoitettu kirjastohakemistoon kuten geolokaatiotietoja käsittelevä luokka.

Sivustolle kirjaudutaan ja sitä käytetään ainoastaan yhden dynaamisesti sisältöään muuttava sivu kautta. Sivun sisältöä muutetaan dynaamisesti käyttäjän tekemien muutosten mukaan. Muutokset kuljetetaan HTTP protokollan GET ja POST parametreissa kulkevilla arvoilla.

Ensimmäisenä HTTP-pyyntö käsitellään palvelimella ohjainluokassa, joka ottaa vastaan käsiteltäväkseen pyynnön. Ainoastaan ohjainluokka käsittelee GET ja POST pyyntöjä ja suorittaa tarvittavat tietokantapyynnöt, jonka se välittää oikealle näkymäluokalle. Tämän jälkeen palvelin lähettää paluupostissa selaimelle näkymän tuottaman XHTML dokumentin. Katso kuva 6.



Kuva 5 Etusivu kirjautuessa, jonka LoginView-luokka tuottaa.

Jokainen näkymä toteuttaa ViewInterface nimisen rajapintaluokan joko perimällä BasicView nimisen luokan, joka toteuttaa ViewInterfacen-rajapinnan, tai toteuttamalla ViewInterfacessa itse suoraan rajapinnan. Kaikki tietoesittävät luokat sijaitsevat fyysisesti views nimisen kansion alla.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML Basic 1.0//EN"
   "http://www.w3.org/TR/xhtml-basic/xhtml-basic10.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
     <title>MobiPaluukuorma.fi</title>
     <link rel="stylesheet" type="text/css" href="css/varikas.css" />
     <meta http-equiv="content-type"        content="text/html;charset=utf-
8" />
     <?php DeviceController::printDeviceSpecificHeaders(); ?>
    </head>

    <body>
        <div id="frame">

            <div class="content_action" >
              <h3>Mobiili paluukuorma.fi</h3>
            </div>

              <div id="tab_content">

                  <div id="document">
                   <?php $this->renderContent(); ?>
```

```
                </div><!-- document end -->

            </div><!-- tab content end -->
        </div><!-- frame end -->
    </body>
</html>
```

Edellinen koodilainaus on "loginview.phtml"-tiedostosta, jonka LoginView-luokka hakee ja toteuttaa. Tässä tapauksessa ainoa metodikutsu, joka suoritetaan, on renderContent().

Kun käyttäjä kirjautuu sivustolle, selain lähettää lomakkeen tiedot palvelimelle post-pyynnössä. Pyynnön käsittelee EventController-luokka, joka hoitaa kirjautumisprosessin, tekee kyselyt tietokannalle ja välittää tiedot käytettävälle näkymälle. Itse näkymä ei ole koskaan suoraan yhteydessä tietokantaan vaan aina pyyntöä käsittelevän ohjainluokan kautta.

Tietorakenteita kuvaavat ja niitä käsittelevät luokat sijaitsevat fyysisesti model nimisen kansion alla. Jokaista tietokannan taulua vastaa fyysisesti yksi luokka, jonka yksi instanssi pitää kerrallaan sisällään yhden rivin tietoa. Luokka pitää sisällään taulun jokaisen sarakkeen tiedot omana jäsenmuuttujanaan.

Tietokannan kanssa keskustelevien luokkien väliin tehtiin rajapinta, tekemällä jokaisesta tiettyä taulua käsittelevästä luokasta abstraktit luokat, joita ohjaimet käyttävät. Tällä tavalla saavutetaan ohjainten ja näkymien riippumattomuus käytettävästä tietokantamoottorista. Kaikki MySQL-tietokantaa käsittelevät luokat perivät ja toteuttavat nämä abstraktit luokat.

9.3 Geokoodaus

Paluukuorman rahdille haetaan paikkatiedot rahdin osoitteen mukaan käyttäen Googlen kehittämän Google Maps ohjelmointirajapinnan kautta, rahdille haetaan leveyspiiri (latitudi) ja pituuspiiri (longitudi). Geokoodin avulla pystytään laskemaan etäisyyksiä ja hakemaan rahteja tietyn matkan päästä halutusta pisteestä. Tällaista osoitteen muunnosta kutsutaan geokoodaamiseksi. Muunnoksen voi tehdä halutessaan myös toisin päin, geokoodista osoitteeksi.

Google tarjoaa palvelun ilmaiseksi, kunhan käyttäjä hankkii Googlen palvelun käyttäjätunnukset, jota vastaan palvelu tunnistetaan koodia haettaessa. Seuraavassa koodiesimerkissä osoitetaan kuinka palvelua käytetään käyttäen JSON rajapintaa hyväksi.

```
<?php
function getGeocode( $address ) {
   $query = "http://maps.google.com/maps/geo?q=";
   $query .= urlencode($address)
         ."&output=json&oe=utf8&sensor=true_or_false&key=";
   $query .= GOOGLE_API_KEY;

   $response = file_get_contents($query);
   $resultObject = json_decode($response);

   $longitude = $resultObject->Placemark[0]->Point->coordinates[0];
   $latitude = $resultObject->Placemark[0]->Point->coordinates[1];

   return array( 'longitude' => $longitude, 'latitude' => $latitude );
}

$address ="Wolffintie 30, 65200 VAASA";

$result = getGeocode($address);

echo "Pituuspiiri: ".$result['longitude']." Leveyspiiri: ".$result['latitude']
?>
```

Skripti lähettää HTTP- pyynnön Googlen tarjoamaan palveluun, jonka käsiteltyään palvelu lähettää takaisin JSON koodatun vastauksen. Itse karttapalvelun web-käyttöliittymän hyödyntäminen jää toistaiseksi toteuttamatta mobiiliselainten puutteista johtuvien syiden vuoksi.

9.4 Vapaiden rahtien etsintä

Tarkoituksena oli toteuttaa kuljettajalle mahdollisuus varata rahteja mobiilipalvelua käyttäen. Kuljettajat pystyvät varaamaan rahteja, joko suoraan selaamalla sivu kerrallaan rahteja tai rajaamalla näytettävien rahtien määrää käyttämällä hakutoimintoa. Hakutoiminto käyttää hyväkseen älypuhelinten geolokaatiotukea siten, että käyttäjä voi nappia painaen hakea sijaintinsa hakulomakkeen kenttiin. Hakutoiminto toimii siten, että se muuttaa annetun sijaintipaikan koordinaattitiedoksi, eli geokoodaa tiedon. Tämän jälkeen

koordinaattitietoja verrataan tietokannassa oleviin vapaisiin rahteihin ja käyttäjälle näytetään haun ehdot täyttävät rahdit

9.5 Laitteen sijainnin paikannus

Joissakin uusimmissa mobiililaitteissa on mahdollisuus paikantaa laitteen sijainti. Tätä ominaisuutta voidaan hyödyntää monessa ohjelmistossa, ensimmäisenä mieleen tulee karttaohjelmisto, jolla käyttäjä voi paikantaa itsensä maailman kartalla nappia painamalla. Paluukuorman tapauksessa ominaisuutta voidaan hyödyntää rahteja etsiessä.

W3C on tehnyt oman määritelmänsä geolokaatio-ohjelmointirajapinnasta nimellä "Geolocation API Specification", joka määrittelee kuinka geolokaatiotietoihin pääsee käsiksi ja miten niitä käsitellään. Esimerkiksi iPhonen safariselain tukee tätä määritelmää. Katso kuva 7.

Kuva 6 iPhone ja sijainnin haku

iPhonin tapauksessa haku on toteutettu JavaScript koodilla, joka hakee puhelimen koordinaatit ja sijoittaa ne "mistä"-kenttään. Tämä pieni ominaisuus säästää käyttäjän sijaintinsa kirjoittamiselta.

```
var latitude, longitude;

//Registeres callback function to hande positions
navigator.geolocation.getCurrentPosition(handlePosition,handleError);

function handlePosition(position) {
    latitude = position.coords.latitude;
    longitude = position.coords.longitude;
}

function handleError(error) {
    alert(error);
}
```

```
function buttonClickHandler() {
   $('start').value = latitude + ',' + longitude;
}

//Register event listener to button, after document objec mobel is loaded
document.observe("dom:loaded", function() {
   $('getpositionbtn').addEventListener('click', buttonClickHandler, false);
});
```

Edellä esitelty koodi toimii ainoastaan iPhonen Safari-selaimessa, mikä on toteutettu vastaamaan W3C määrittelemää Geolocation API Specification määrittelyä. Tapahtumia käsitellään lähes aina funktioiden paluukutsuina.

```
function handlePosition(position) {
   latitude = position.coords.latitude;
   longitude = position.coords.longitude;
}
```

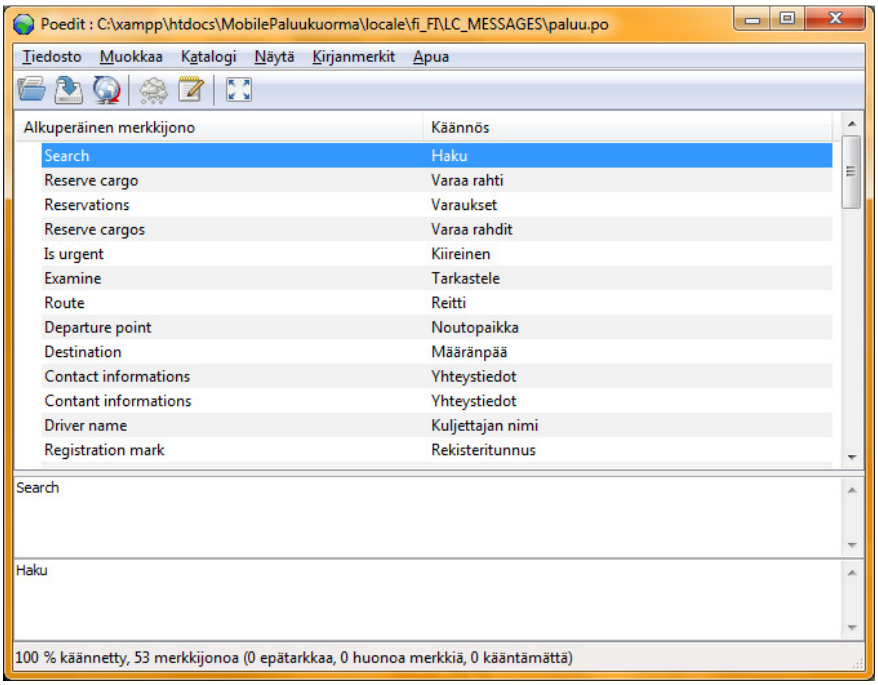
Coordinates olion mukana tulee vähintään pituuspiiri, leveyspiiri ja sijainnin tarkkuus. Sen lisäksi saattaa tulla korkeus merenpinnasta, korkeuden tarkkuus, suuntima ja siirtymänopeus ja kaikki mahdolliset tiedot käyttäjän vakoilemiseksi. (Geolocation API Specification 1.10.2009.)

```
function buttonClickHandler() {
   $('start').value = latitude + ',' + longitude;
}
```

Tämä funktio hoitaa tiedon sijoittamisen itse tietokenttään.

9.6 Käyttöliittymän lokalisointi paluukuormaohjelmistossa

Käyttöliittymän lokalisointi voidaan toteuttaa PHP-ohjelmistossa käyttäen gettext PHP-lisäosaa, joka toimii rajapintana C:llä kirjoitetun GNU gettext kirjaston välillä. Ohjelmakoodi voidaan kirjoittaa siten, että ohjelmakoodista voidaan generoida .Po-tiedosto, jota erillinen varsinaiseen kääntötyöhön kehitetty graafinen ohjelmisto käyttää. Tällainen lokalisointitapa on tarkoitettu erityisesti käyttöliittymän lokalisointiin.



Kuva 7 PoEdit on ohjelma, jolla varsinainen käännöstyö tehdään.

PoEdit on yksi vaihtoehdoista kun käännöstyö halutaan tehdä. Itse kääntäjän ei tarvitse nähdä edes ohjelmakoodia käännösprosessin aikana, mikä on varmasti mielekkäämpää myös kääntäjälle.

Aluksi lähdekoodista generoidaan po-tiedosto PoEditlla, joka pitää sisällään viittaukset lähdekoodiin ja alkuperäiskäännöksiin. Tämän jälkeen tehdään käännös halutulle kielelle, josta syntyy binäärinen kielitiedosto, jonka gettext-lisäosa lataa välimuistiinsa sivun ensimmäisellä latauskerralla. Muutosten päivittäminen saattaa aiheuttaa hiukan ihmetystä, koska tiedot ladataan vain kerran kielitiedostosta välimuistiin ja näin vältytään siltä, että tiedosto ladattaisiin levyltä jokaisella sivunlatauksella. Paras tapa saada uudet päivitykset näkymään sivustolla kielitiedostosta on tehdä asia siten, että uudelleen nimeää kielitiedoston nimen joka päivityskerralla tai nimeää tiedoston uudelleen ja lataa sivun selaimessa uudelleen ja nimetään tiedoston takaisin alkuperäiseksi. Tämän jälkeen muutosten pitäisi näkyä myös sivustolla.

```
<?php
echo _("Hello World!");
```

```
?>
```

PoEdit osaa generoida PHP-lähdekoodista Po-tiedoston lukemalla yläpuolella näytetyllä tavalla tehdyt funktiokutsut.

Seuraavalla tavalla gettext-lisäosa saadaan tietoiseksi käytettävästä lokalisoinnista.

```
<?php
$locale = "fi_FI";
if (isSet($_GET["locale"])) $locale = $_GET["locale"];
putenv("LC_ALL=$locale");
setlocale(LC_ALL, $locale);
bindtextdomain("messages", "./locale");
textdomain("messages");
?>
```

9.7 Jatkokehitys

Raportin kirjoittamisen loppuvaiheessa mobiilipaluukuorma on koekäytössä ja sivuston kehitystä on tarkoitus kehittää käyttäjiltä saadun käyttökokemusten perusteella. Tämän lisäksi käyttökokemuksia on jo kerätty näyttämällä demoversiota koehenkilöille ja keskustelemalla heidän kanssansa tämän jälkeen.

Sivuston ensisijainen käyttöliittymä suunniteltiin iPhonen puhelimen näytölle, jonka näyttö on mobiililaitteiden joukossa kohtalaisen suuri. Sivuston käyttäessä saattaa olla ongelmallista käyttäessä sivustoa hyvin pieninäyttöisellä laitteella.

Mahdollisesti mobiilipuolelle voidaan kehittää myös mahdollisuus muokata sivusto käyttäjän mieltymyksien mukaiseksi. Käyttäjä voisi valita oman väriteemansa tai oletus näkymän. Tässä kohdassa on monia mahdollisuuksia parantaa käyttäjäkokemusta. Käyttöliittymän muokkausmahdollisuuden voisi suorittaa käyttäen pöytätietokonetta, jolloin käyttöliittymän personalisoimisen suorittaminen olisi helpompaa.

10 YHTEENVETO

Työn tuloksena saatiin aikaan mobiiliversio paluukuorma.fi internetsivustosta, joka on koekäytössä tämän raportin kirjoittamisen hetkellä.

Ohjelmiston koodaaminen kävi sinänsä suoraviivaisesti, koska ohjelmiston tietorakenteet oltiin jo ennalta määritelty. Täytyi vain miettiä miten ja mitä ominaisuuksia ohjelmistoon voidaan toteuttaa. Kun huomattiin puutteita, jouduttiin toisinaan tekemään isojakin rakenteellisia uudistuksia.

Sivuston ulkoasun suunnittelu kävi läpi kaksi vaihetta, ensimmäinen versio sivustosta oli harmaan kylmä, jonka jälkeen sivuton ulkoasuun päädyttiin pistämään lisää väriä. Ulkoasun fontiksi päädyttiin valitsemaan arial, koska se sopii hyvin näytöltä luettavaan tekstiin. Lisä kuorrutteena sivustolle laitettiin muutamia ikoneita, jotka helpottavat toimintojen hahmottamista.

Työhön tietoa hakiessa, varsinkin projektin alkuvaiheessa paras tietolähde oli W3C Mobile Web Best Practices 1.0 luonnos, josta löytyi hyvin kattavasti neuvoja sivuston kehittämistä varten. Tällä hetkellä netistä löytyy yllättävän vähän ja kyseenalaisia ohjelmointioppaita mobiilisivuston kehittämistä varten, joten W3C:n tarjoama luonnos osoittautui lähes korvaamattomaksi.

Yritykset ovat kehittäneet lisääntyvässä tahdissa omista internetsivuistaan mobiiliversiota. Mobiiliselainten toiminnallisuuksien yhtenäistyminen W3C-standardointityön mukana on luonut vakaan ja kestävän kehityksen perustan internetille mobiililaitteissa. Tulevaisuudessa tulemme varmasti näkemään myös mobiili-internetsivuja, jotka hyödyntävät suuremmassa määrin geolokaatiotietoja palveluissaan tai mainonnassaan.

LÄHDELUETTELO

1. *Painetut teokset*

Erich Gamma, Richard Helm, Ralph Jonson, John Vlissides: Design Patterns. Olio-ohjelmointi. Suunnittelumallit 2001. 5 p. Edita, IT Press

Vaswani, Vikram. How to Do Everything with PHP and MySQL. 2005. 26 p. Blacklick, OH, USA: McGraw-Hill Companies

2. *Elektroniset julkaisut*

Ecma International, ECMAScript 3rd Edition specification [Päivitetty 25.6.2000] [viitattu 11.8.2009] Saatavissa pdf-tiedostona <URL:http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf>.

Device Atlas, Mobile Device Intelligence [Päivitetty 13.12.2009] [viitattu 13.12.2009] Saatavana www-muodossa: <URL: http://deviceatlas.com/>.

PHP NET, History of PHP and related projects [Viitattu 7.11.2009]. Saatavissa www-muodossa <URL: http://php.net/manual/pl/history.php>.

MySQL AB, MySQL 5.4 Reference Manual [Viitattu 10.8.2009]. Saatavissa www-muodossa <URL: http://www.mysql.com/>.

PHPDocumentor, PHP documentator manual [Päivitetty 1.6.2009] [Viitattu 15.7.2009] Saatavissa www-muodossa: <URL: http://manual.phpdoc.org/>.

World Wide Web Consortium, Mobile Web Best Practices 1.0[Päivitetty 29.5.2008] [Viitattu 8.7.2009] Saatavana www-muodossa <URL: http://www.w3.org/TR/mobile-bp/>.

World Wide Web Consortium, Portable Network Graphics (PNG) Specification (Second Edition) [Päivitetty 10.11.2009] [Viitattu 15.12.2009] Saatavana www-muodossa <URL:http://www.w3.org/TR/PNG/>.

World Wide Web Consortium, Geolocation API Specification (Editor's Draft 24 August 2009) [Päivitetty 24.3.2009] [Viitattu 22.10.2009] Saatavana www-muodossa <URL: http://dev.w3.org/geo/api/spec-source.html#geolocation_interface>.

Zend Framework, Zend Framework Coding Standard for PHP [Viitattu 28.10.2009] Saatavana www-muodossa <URL:http://framework.zend.com/manual/en/coding-standard.html>.

LIITELUETTELO