Markus Kuula

Tietolähteitä yhdistävä karttapalvelu

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Ohjelmistotekniikka

Insinöörityö

16.04.2013

Metropolia

Tekijä(t) Otsikko Sivumäärä Aika	Markus Kuula Tietolähteitä yhdistävä karttapalvelu 37 sivua + 1 liite 16.04.2013
Tutkinto	insinööri (AMK)
Koulutusohjelma	Tietotekniikka
Suuntautumisvaihtoehto	Ohjelmistotekniikka
Ohjaaja(t)	Lehtori Vesa Ollikainen Yliopettaja Erja Nikunen Tekstinohjaaja Jussi Alhorinne

Internetin yleistymisen myötä on syntynyt monia sähköisiä palveluita, joista hyödyllisimpiä ovat karttapalvelut. Niiden tehtävänä on korvata perinteinen kartta. Karttapalveluiksi kutsutut palvelut mahdollistavat kartan vapaan käsittelyn ja halutun matkareitin määrittämisen. Monet karttapalvelut on tarkoitettu pelkästään loppukäyttäjille, mutta monissa on mahdollisuus hyödyntää sitä myös omilla sivuilla tai omissa sovelluksissa.

Työssä tutustutaan yleisimpiin karttapalveluihin ja niiden toimintaan sekä yleisellä tasolla että syvemmin paneutuen myös reitinmuodostuksen tekniikkaan. Tässä yhteydessä tehdään Agilo Partners -yritykselle karttapalvelu, joka mahdollistaa sijaintitietojen näyttämisen kartalla. Palvelu tehdään asuntoilmoituksia sisältävälle sivustolle, jolla näytetään asuntojen sijainnit kartalla. Asiakas päätyi valinnassaan Google Mapsiin, jonka käskytys tapahtuu JavaScriptillä. Tarvittavat koordinaatit saatiin tietokannasta Rubylla.

Karttapalvelu rakennettiin toimivaksi ja sijoitettiin asuntojen sijaintitietojen yhteyteen. Tällöin sivulla vierailija näkee asunnon osoitetiedot paitsi tekstinä myös selkeänä karttakuvana. Lisäksi hän voi helposti tehdä hakuja esimerkiksi oman asuinpaikkansa ja kohteen väliltä.

Avainsanat	Google Maps, karttapalvelu, reitinmuodostus, reitti

Author(s) Title Number of Pages Date	Markus Kuula Map service utilizing data sources 37 pages + 1 appendice 16 April 2013
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Software engineering
Instructor(s)	Vesa Ollikainen, senior lecturer Erja Nikunen, principal lecturer Jussi Alhorinne, text supervisor

The purpose of this final project is to explore common map services and their behavior. The main aim is to create a map service for an Internet site featuring housing advertisements. This study was carried out for Agilo Partners.

This final project is based on a review of the most common map services and the technology of route creation. The client decided on the specifications of the map service that displays location data and decided on Google Maps which includes this opportunity. They wanted the map service to be placed where the address of the residence is showing on the site. The idea is that the visitor may then see the address in the form of a picture. It also allows doing a search between the home and the destination. The map service uses Ruby to retrieve the required coordinates.

The final project was successful in creating the map service according to the specifications. The service is functioning well and could be further developed in a separate project, which would include some more features that were not included in this version.

Keywords	Google Maps, map service, routing, route

Sisällys

Lyhenteet

API	Application Interface. Mahdollistaa sovelluksen toimintojen hyödyntämisen toisesta ohjelmasta käsin.
ERB	Embedded Ruby. Ruby-kielen koodia sijoitettuna tavallisen HTML-kielen sekaan.
GM	Google Maps. Googlen tarjoama karttapalvelu.
GPS	Global Positioning System. Satelliittipaikannustekniikka jolla voi paikallistaa esimerkiksi oman sijaintinsa.
HTML	HyperText Language. Sivunkuvauskieli joka kuvaa Internetsivulla olevat rakenteet.
JS	JavaScript mahdollistaa vuorovaikutuksen Internet-sivuilla asiakaspäässä.
JSON	JavaScript Object Notation. Tietorakenne joka sisältää rakenteisessa muodossa olevaa tietoa (avain/arvopareina).
MVC	Model View Controller erottaa tiedon ja sen esitystavan toisistaan.
PHP	HyperText PreProcessor (alkup. Personal HomePage tools). Tekniikka joka mahdollistaa muuttuvien ja vuorovaikutteisten sivujen tekemisen palvelinpäässä.
XHTML	Extended HyperText Language. HTML-kielen laajennus joka on XML-yhteensopiva.
YTV	Yhteistyövaltuuskunta. Huolehti aiemmin pääkaupunkiseudun alueella jätehuollosta ja liikenteestä (nykyisin tehtäviä hoitavat HSL ja HSY).

1 Johdanto

Kuva kertoo enemmän kuin tuhat sanaa. Tämä pätee erityisen hyvin reitin selostamisessa, jossa tavallinen kartta on ylivoimainen. Suunnistaminen pelkän ääni- tai tekstiselostuksen varassa on hankalaa. Autoilijoiden suosimissa navigaattoreissa on tämän takia lisänä pieni karttakuva lähiympäristöstä, mikä helpottaa olennaisesti navigointia kaupungissa ja maalla.

Usein tällainen kartta löytyy yritysten ja laitosten yhteystietoja sisältäviltä sivuilta. Kartta hyödyttää suunnistajaa pelkästään kuvanakin, mutta tietotekniikka antaa siihen paljon uusia mahdollisuuksia, joita perinteiset, painetut kartat eivät pysty hyödyntämään. Tällainen on esimerkiksi kartta, johon merkitty määränpää voi vaihdella käyttäjän antamien tai tietokannasta haettujen lähtötietojen perusteella. Siihen voi vaikkapa helposti tehdä merkintöjä, jotka saa sormen näpsäytyksellä pois.

Tämän opinnäytetyön aiheena on eri tietolähteitä hyödyntävän karttapalvelun toteuttaminen asunnonvuokrausilmoituksia välittävälle sivustolle nimeltä Vuokraforum. Palvelu näyttää asunnot kartalla sekä haluttaessa matkareitin. Työssä perehdytään palvelun rakentamisen lisäksi karttapalveluissa käytettäviin tekniikoihin ja vaihtoehtoisiin toteutustapoihin. Lisäksi luodaan katsaus jo toteutettuihin palveluihin.

Työhön liittyvän ohjelmistotuotteen toteutus etenee siten, että ensin laaditaan tarpeeksi kattavat suunnitelmat sovelluksen rakenteesta ja toiminnasta. Näiden perusteella laaditaan itse karttasovellus, jonka jälkeen se testataan. Testausta on toki tehtävä myös itse toteutusvaiheen aikanakin. Työn toteutuksesta vastaa yksin tämän tekstin kirjoittaja, minkä takia Scrum- tai vastaava ketterä menetelmä ei ole tarkoituksenmukainen.

Raportti jakautuu karkeasti ottaen kahteen osaan, joista ensimmäisessä tutustutaan saatavilla oleviin karttapalveluihin (luku 3) sekä niiden toteutustapoihin ja toisessa itse työn suunnitteluun ja toteutukseen (luku 6). Teoriaosuus koostuu taustojen selvittämisestä, tavoitteiden asettamisesta (luku 2) ja käytettävien tekniikoiden kuvailemisesta (luvussa 6 kohta 6.5). Toteutusosuus on jaettu määrittelyyn, toimintakuvaukseen sekä käyttöliittymän kuvailemiseen ja teknisen rakenteen selvittämiseen. Sen jälkeen katsastetaan projektin toteutusta ja lopputuloksen onnistumista ja siitä vedettäviä johtopäätöksiä.

2 Tausta ja tavoitteet

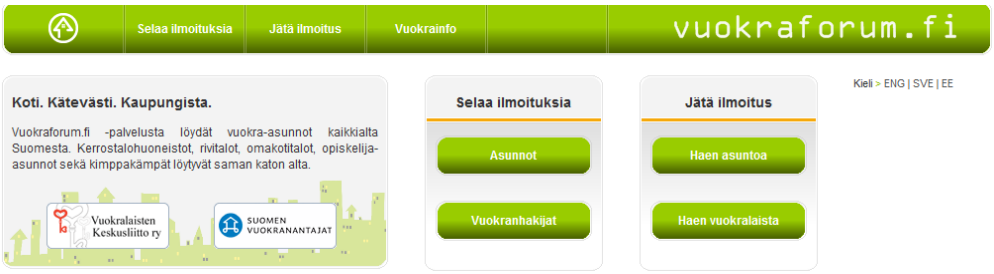
Nopeutuneiden verkkoyhteyksien ja tietokoneiden myötä ovat erilaiset karttapalvelut ilmaantuneet verkkoon. Nämä palvelut tarjoavat karttoja erilaisiin tarkoituksiin samaan tapaan kuin karttakirjat tekevät. Pelkkä kartasta oleva kuva ei kuitenkaan yksinään edusta karttapalvelua kuin lähinnä silloin, kun kartat edustavat eri ajanjaksoja (esim. historialliset kartat tai merikortit). Tällöin palvelussa voi valita kartan haluamaltaan ajanjaksolta.

Karttoja tarjotaan sekä perinteisen karttakirjan tavoin katseltavina (ja samalla myös tulostettavina) että muuttuvina (dynaamisina). Jälkimmäisissä karttakuva muodostuu käyttäjän antamien tietojen perusteella. Esimerkiksi palvelussa, joka muodostaa matkareitin haluttuun paikkaan, karttakuva muodostuu lähtö- ja kohdepisteestä, matkareitistä sekä karttakuvasta. Karttakuvassa on valmiina katujen ja paikkakuntien nimet. Esimerkkinä tällaisesta on myöhemmin esiteltävä pääkaupunkiseudulle suunniteltu Reittiopas. Esimerkkinä voisi vielä ottaa Helsingin kaupungin lumipalvelun, jonka kautta voi ilmoittaa aurausta kaipaavista teistä (http://kerrokartalla.hel.fi/lumipalaute). Jälkimmäinen on lopettanut toimintansa, koska se todettiin epäonnistuneeksi.

Karttapalvelun tarjoamaa karttaa voi paperimuotoisesta kartasta poiketen käsitellä eri tavoin. Sitä voi lähentää (tarkentaa) tai loitontaa, vierittää eri suuntiin tai joissain tapauksissa jopa kallistaa (kartta näkyy viistosti eli kyseessä on isometrinen kuvakulma). Karttapalvelut käyttävät usein satelliitin tarjoamaa kuvaa pohjana, joten kartasta on saatavana perinteisen viivakartan lisäksi satelliitti- tai ilmakuvaversio. Joissain tapauksissa käyttäjä voi sukeltaa kolmiulotteisesti mallinnettuun näkymään tai katsoa määrätyssä kohdassa valokuvaa, joka on otettu kadulta. Tästä käytetään tässä dokumentissa nimitystä katunäkymä.

Karttapalvelut voidaan jakaa kahtia sen mukaan, voiko niitä hyödyntää omissa sovelluksissa. Ensimmäiset on tarkoitettu pelkästään loppukäyttöä varten. Toisen ryhmän toimintoja käyttävät monet nettisivustot ja -palvelut, joissa karttaa käytetään tavalla tai toisella (esimerkiksi yhteystietoja käsittävällä alasivulla). Esimerkiksi Reittiopas on esimerkki edellisen ryhmän karttapalvelusta ja Google Maps jälkimmäisen.

Opinnäytetyössä tehdään karttapalvelu Vuokraforum-nimiselle sivustolle (http://www.vuokraforum.fi, kuva 1) käyttäen Googlen Maps -karttapalvelua. Kyseistä Googlen tarjoamaa palvelua voi käyttää epäkaupalliseen toimintaan ilman maksua erinäiset rajoitukset huomioiden.



Kuva 1: Vuokraforumin etusivu.

Vuokraforum on Agilo Partners Oy:n perustama sivusto asuntoilmoituksia varten, jossa kävijä voi erilaisten ehtojen perusteella etsiä haluamiaan asuntoja. Asunnoista kerrotaan sijainti, pinta-ala, vuokra, ajanjakso yms. tiedot. Tietojen ohella on myös kartta, josta näkee asunnon sijainnin. [Vuokraforum 2011.]

Palvelu ei enää ole kuitenkaan toiminnassa, koska tekijöiden resurssit eivät riittäneet palvelun ylläpitoon. Tämän takia opinnäytetyönä syntynyttä ohjelmistoprojektia ei voi testata varsinaisessa tehtävässään.

Opinnäytetyön ensimmäinen tavoite on esitellä lukijalle tarjolla olevia karttapalveluita ja kertoa, mitä karttapalvelut ovat ja miten niitä voi hyödyntää sekä käyttäjän että kehittäjän näkökulmasta.

Toisena tavoitteena on rakentaa Vuokraforumille karttapalvelu, joka hyödyntää eri tietolähteitä muodostaessaan lopputuloksen. Siitä näkee eri kohteiden (asuntojen) sijainnin. Sijaintitiedot haetaan tietokannasta, jota käyttäen muodostetaan puolestaan Google Maps -palvelua käyttäen kartta, missä asunnot on merkitty selkeästi. Palvelu mahdollistaa matkareitin muodostamisen ja piirtämisen kartalle haluttujen asuntojen välille.

Henkilökohtaisena tavoitteena on tämän valmiin karttasovelluksen myötä kartuttaa kokemusta uusista, itselle enemmän tai vähemmän tuntemattomista tekniikoista (lähinnä JavaScript, Ruby / PHP ja git) sekä vesiputousmallin mukaisesta sovelluskehityksestä. Lisäksi lopputyön tavoitteena on kartuttaa kokemusta versionhallinnasta ja suunnitelmien laatimisesta.

Lukijalle, oli se sitten toinen insinööri (opiskelija) tai vain kiinnostunut, opinnäytetyö tarjoaa läpileikkauksen karttapalveluista yleisellä tasolla sekä esimerkin sellaisen toteuttamisesta.

3 Karttapalveluja

3.1 Suomalaisia karttapalveluja

Nyt toteutettava karttapalvelu ei ole ainoa laatuaan. Esimerkkejä toteutetuista karttapalveluista ovat Oikotie [Oikotie 2013], YTV:n reittiopas [HSL 2013] sekä Eniro [Fonecta 2012]. Kaikkia näitä yhdistää kartta, jota voidaan vierittää ja suurentaa tai pienentää vapaasti. Kartalla näkyy maastonmuotojen ja katujen seurana paikkakuntien ja teiden nimet. Perinteisen viivakartan sijalla voi olla myös ilmakuva Eniroa lukuun ottamatta. Kaikilla kolmella on myös toisistaan poikkeavia ominaisuuksia.

Reittiopasta lukuun ottamatta voi omilta verkkosivuilta linkittää karttapalveluun kartan valmiin reitin kanssa, mikä mahdollistaa esimerkiksi yhteystietojen esittämisen karttakuvan muodossa. Kaupallinen käyttö ei ole sallittua yhdenkään palvelun kohdalla, joten Vuokraforumin tapauksessa kaikki seuraavassa esitetyt palveluntuottajat voidaan unohtaa. Käsittelen ne kuitenkin tässä, koska niiden avulla saadaan karttapalveluista ja niiden toteutuksesta parempi kuva.

3.1.1 Oikotie

Oikotie on Sanoma Newsin omistama palvelu, jolla voi etsiä asuntoja, autoja ja työpaikkoja. Näistä asuntohaku tarjoaa pelkän kuvilla ryyditetyn näkymän lisäksi karttanäkymän. Paikkakunnan kohdalla näkyy ympyrän sisällä asuntojen lukumäärä ja tarkentamalla ilmaantuu näkyville punaiset talot, jotka edustavat asuntoa. Painamalla talon kuvaa siitä saa tarkempia tietoja.

3.1.2 Reittiopas

YTV:n alueella on käytettävissä Reittiopas matkan teon suunnittelua varten. Lähtö- ja kohdepaikan lisäksi ei muita tietoja tarvita, paitsi jos sama paikka löytyy useammalta alueelta tai tiedot ovat muuten puutteellisia.

Kun tiedot ovat oikein, palvelu piirtää kartalle viivan lähtöpisteestä kohdepisteeseen. Viivan väri kertoo kulkuneuvon (raitiovaunu = vihreä, metro = oranssi, bussi = sininen ja juna = punainen), ja pysäkit merkitään ympyröin. Kohdepisteet näkyvät neliöinä.

Varsinaisen karttakuvan lisäksi reitin näkee myös listana, jossa on lueteltu pysäkit ja pysähtymisajat sekä ajoneuvojen vaihdot.

Toteutuksen pääongelmana on löytää lyhin reitti haluttuun kohteeseen. Reittioppaan kohdalla ei riitä, että huomioidaan matka linnuntietä tai ”oikeita” teitä pitkin vaan sen on huomioitava käytettävissä olevat kulkuneuvot ja niiden aikataulut. Mahdolliset poikkeustilanteet liikenteen ja sään osalta on myös huomioitava.

3.1.3 Eniro

Eniron kartan erikoisuus on mahdollisuus mitata välimatkoja. Tämä tapahtuu napsauttamalla haluttua kohtaa (lähtöpaikkaa) ja jatkamalla samaa rataa, kunnes käyttäjä on saavuttanut kohdepisteen. Tämän kohdalla kaksoisnapsautus lopettaa piirtämisen ja näyttää kokonaismatkan.

GPS-tilassa näkyvät koordinaatit hiiren osoittamassa kohdassa. Napsauttamalla jotakin kohtaa kartalla pysyvät sen koordinaatit näkyvillä.

Vuonna 2010 Fonecta osti Eniron, jolloin myös Internet-palvelu siirtyi heidän omistukseensa.

3.2 Ulkomaisia karttapalveluja

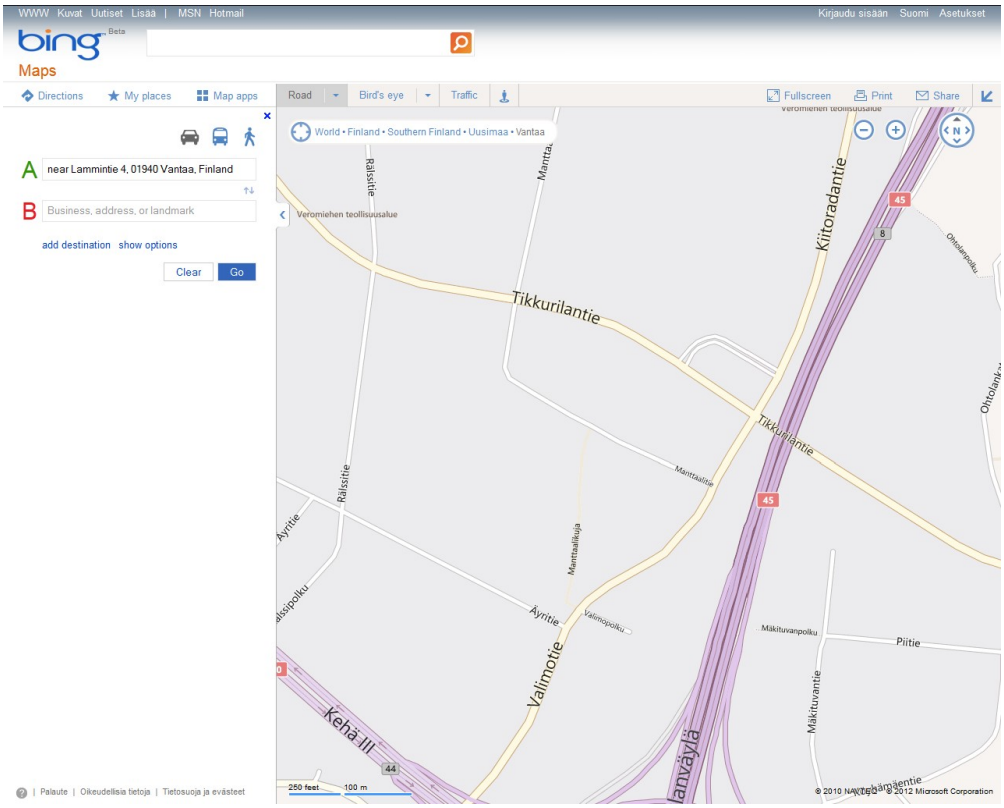
Seuraavassa on joitakin ulkomaisia karttapalveluja. Kaupallinen käyttö on kielletty kaikkien alla esitettyjen kohdalla Google Mapsia lukuun ottamatta. Mitä se Vuokraforumin käytössä tarkoittaa, on harmaata aluetta. Asuntoilmoituksia voi foorumilla selailla vapaasti, mutta niiden jättäminen on maksullista. Siksi ennen karttapalvelun käyttöönottoa on hyvä tarkistaa juridinen puoli asiakaspalvelusta, mikäli se ei ole kristallinkirkas sopimuksen lukemisen jälkeen. Jos tämä ei onnistu, on varminta siirtyä seuraavaan palveluun.

Kuten otetuista ruutukaappauksista näkyy, muistuttavat esitellyt palvelut suuresti toisiaan. Kun osaa yhtä käyttää hyvin, osaa käytännössä kaikkia muitakin.

3.2.1 Microsoft MapPoint / Bing Maps

Tietokoneohjelmistoja valmistavan Microsoftin oma näkemys karttapalvelusta on heidän MapPoint -ohjelmistonsa [Microsoft 2012]. Monet Microsoftin palvelut käyttävät kyseistä järjestelmää paikallistamiseen (kuten MSN Search ja Streets and Trips). Se tukee monia karttapalveluiden peruspiirteitä (osoitteiden paikallistaminen, koordinaattien muuttaminen osoitteiksi ja välietappien määrittely). [Gosnell 2005 : 50 – 51.]

MapPointista oli myös verkkopohjainen versio, joka oli ilmainen. Sittemmin sen on korvannut Bing Maps (kuva 2). Tämä tarjoaa tutut satelliitti- ja ilmakuvaversiot kartoista, kuten myös 3D- ja katunäkymän. Karttakuvan saaminen omille sivuille on pyritty tekemään mahdollisimman helpoksi. Näkyvillä olevan kartan voi liittää helposti omalle sivulle yhtä nappia painamalla, jolloin generoidaan kartan upottamiseksi tarvittava koodi.



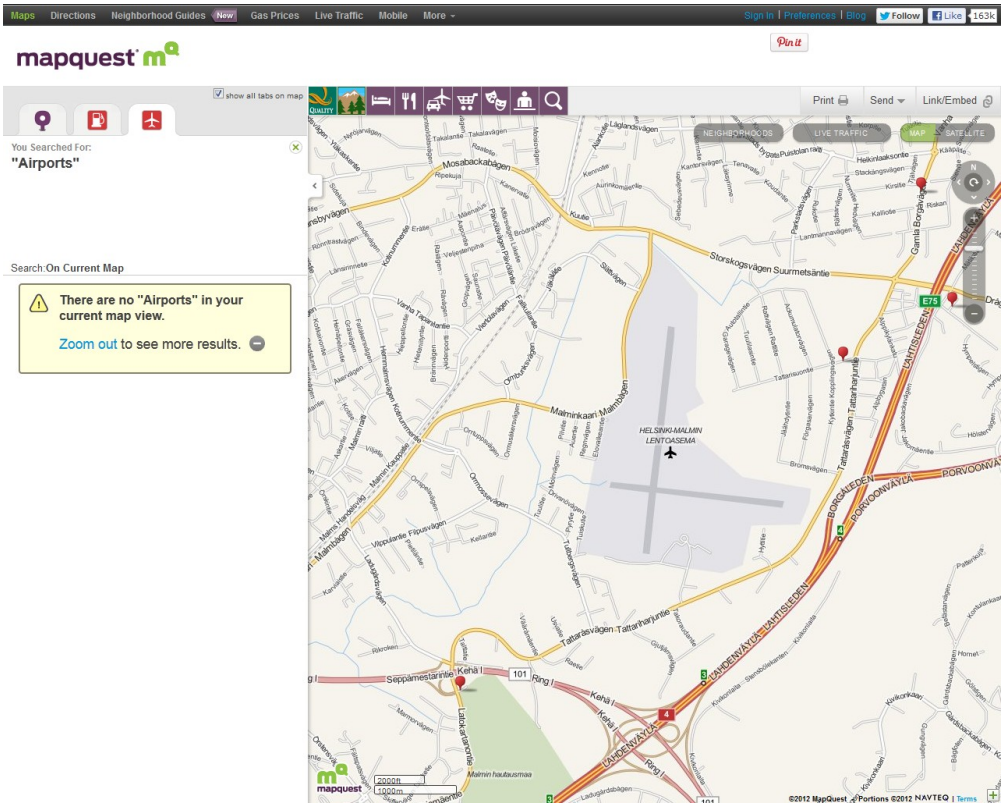
Kuva 2: Bing Maps.

3.2.2 Mapquest

Mapquest oli yksi ensimmäisiä karttapalveluita, joka tarjosi reitityspalveluita. Sen toteuttaja, AOL (American Online) -Internet-palveluntarjoaja, on Time Warnerin omistama. [Mannings 2008 : 139.] Palvelussa on mahdollista saada kartalle yleisiä tai yksityisiä palveluita, kuten ravintolat, bensa-asemat ja kirjastot. Nämä näkyvät kartan lisäksi vasemmalla olevassa luettelossa (kuva 3).

Karttakuvan upottaminen omalla sivustolle onnistuu helposti painamalla oikeassa yläkulmassa olevaa painiketta. Se tuottaa sekä linkitykseen että upotukseen tarvittavat koodit.

Ohjelmistokehittäjiä ajatellen palvelua voi hyödyntää sekä JavaScriptillä että Flashilla. Se tukee tavallisimpia toimintoja, joita ovat osoitteen geokoodaus (tarkemmin luvussa 4), optimireitin muodostaminen sekä poikkeusliikenteen huomioiminen.



Kuva 3: MapQuest.

Painiketta painamalla palvelu muodostaa käytettävän upotuskoodin:

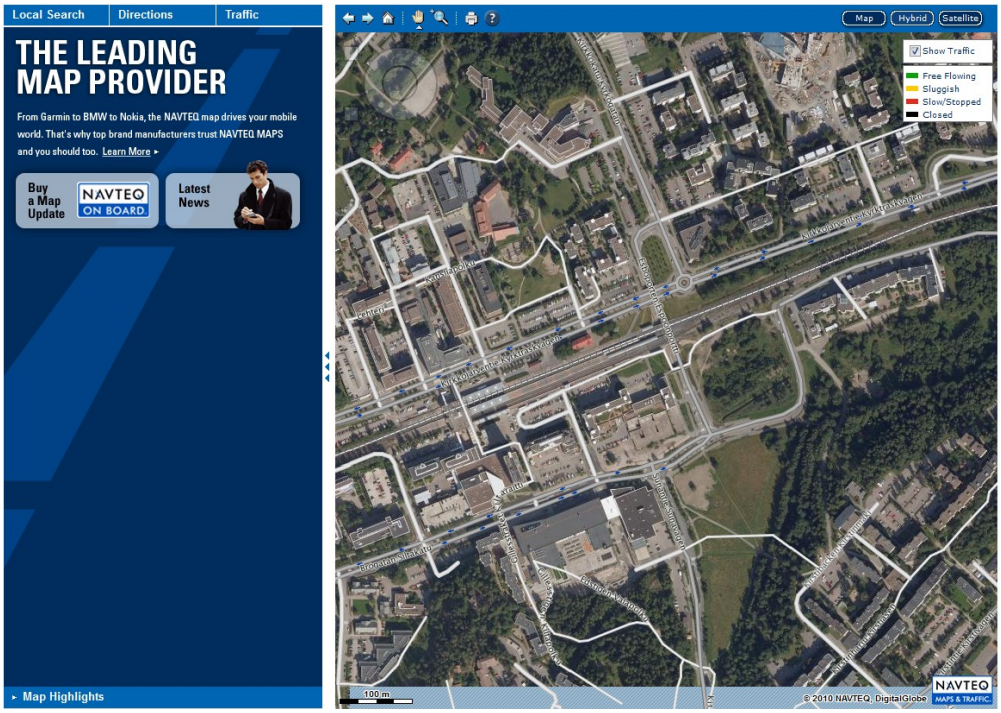
```
<iframe style="height: 270px; width: 450px;"
src="http://www.mapquest.com/embed?hk=JI6r3E" marginwidth="0"
marginheight="0" frameborder="0" scrolling="no"></iframe>
```

Ensin asetetaan korkeus ja leveys. Toisen rivin alussa oleva hk-määre sisältää koordinaatit palvelun omassa, koodatussa muodossa. Lopuksi asetetaan ulkoasuun liittyviä määritteitä.

3.2.3 Navteq

Navteq on Nokia-matkapuhelinvalmistajan vastaisku Tele Atlasille (Tele Atlas muodostaa Googlen ja monien muiden karttapalveluiden satelliittikuvat) [Mannings 2008 : 139]. Sen päänäkymä on kuvassa 4. Kartalle voi lisätä senhetkisen liikenteen, jolloin sen solmukohdat on helpompi välttää. Palvelu voi myös muodostaa reitin kahden pisteen välille (kuten lopputyönä oleva karttapalvelukin), mutta kohdepisteinä voivat toimia vain isommat alueet, kuten Suomessa kasvitieteellinen puutarha ja olympiastadion.

Palvelun hyödyntäminen omilla verkkosivuilla tapahtuu Navteqin omalla rajapinnalla, jota käytetään JavaScript-koodista käsin.



Kuva 4: Navteq.

Seuraavassa lyhyt skripti, joka esittelee palvelun liittämistä omille verkkosivuille (script-elementtiä käytetään erilaisten toimintojen eli komentosarjojen liittämiseksi sivulle):

```
1)   <script type="text/javascript">
2)       //Add the deCarta credentials
3)       deCarta.Mobile.Configuration.clientName = '*****';
4)       deCarta.Mobile.Configuration.clientPassword = '*****';
5)       deCarta.Mobile.Configuration.configuration = '*****';
   // Specify your Navteq data set
   //Register a function that will be executed once the document has
         completed loading.
6)       window.onload = function(){
7)       //Get the size of the window we are working with.
8)       var size = deCarta.Window.getViewport();
9)       //Resize body and map element to perfectly fit the window.
10)              document.body.style.width = size.width + 'px';
11)              document.body.style.height =  size.height + 'px';
12)              document.getElementById('mapContainer').style.width =
                     size.width + 'px';
13)              document.getElementById('mapContainer').style.height =
                     size.height + 'px';
14)              //Now instantiate a map!
15)              new deCarta.Mobile.Map({id: "mapContainer"});
16)      }
17)      </script>
```

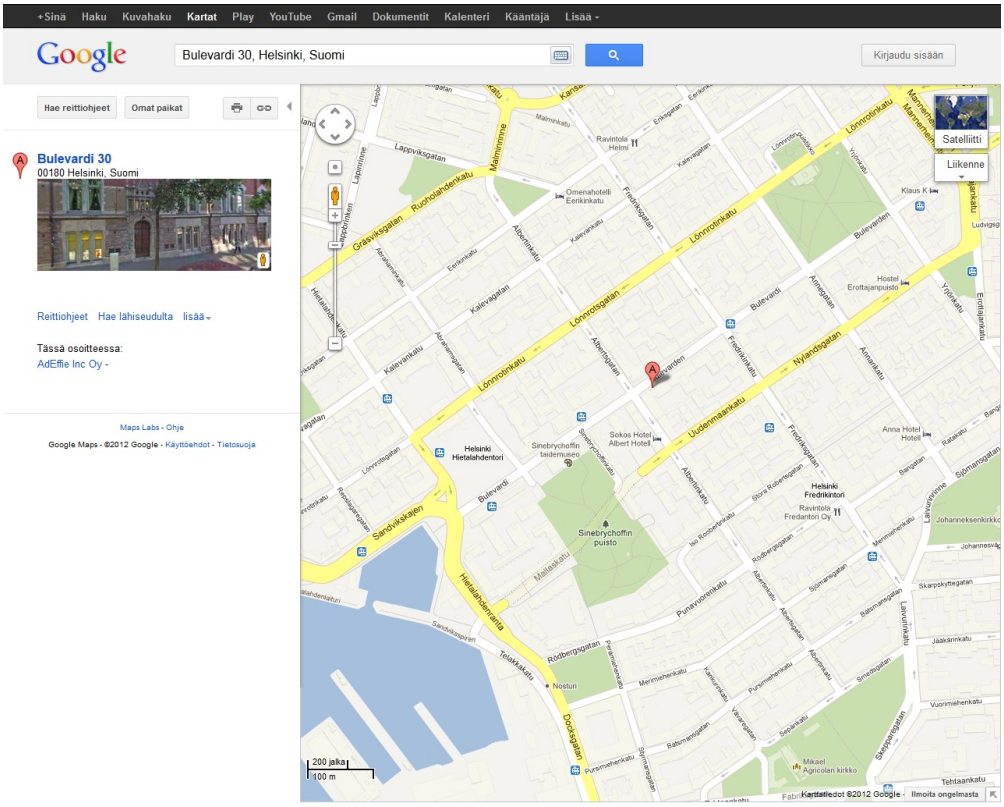
Ensimmäisillä riveillä hoidetaan kirjautuminen palveluun (rivit 2-5). Riveillä 10-13 asetetaan karttaobjektin koko sopivaksi. Rivi 15 varsinaisesti luo kartan mapContainer-elementin sisälle.

3.2.4 Google Maps

Google tarjoaa pelkän www-sivujen haun lisäksi useita muita hakupalveluja. Eräs näistä on paikkahaku, joka tunnetaan nimellä Google Maps [Google 2010]. Se näyttää kartalla pyydetyn kohdepaikan, katujen ja kaupunkien nimet sekä tarvittaessa ajo-ohjeet, kuten kuvasta 5 näkyy.

Google Maps -toiminnallisuuden voi helposti upottaa omille verkkosivuille, kunhan, käy ensin rekisteröitymässä palveluun. Se on ilmaista, mutta rajoitukset kannattaa palvelun käytössä tarkistaa, sillä sivunlatausten määrä on rajoitettu. Lisäksi palvelua ei voi käyttää, mikäli sivusto ei ole vapaassa jakelussa (käytöstä peritään maksua).

Asiakas oli päätynyt valinnassaan Google -yhtiön Maps-palveluun. Eräs painava tekijä valinnassa oli ko. palvelun ilmaisuus ja mahdollisuus hyödyntää sitä myös kaupallisessa käytössä (Vuokraforum tuottaa rahaa ilmoituskohtaisilla lisäpalveluilla).



Kuva 5: Google Maps.

Google Maps -palvelun käyttöönottoa tarkastellaan lähemmin luvussa 6.5.3.

4 Paikkatieto karttapalvelussa

Paikkatieto voidaan ilmoittaa joko selväkielisenä osoitteena tai satelliittikoordinaatteina. Tekstimuotoinen osoite on tarkoitettu pelkästään ihmisiä varten ja koostuu katuosoitteesta, postinumerosta ja postitoimipaikasta. Joissakin maissa voi olla tästä poikkeavia osoitteen ilmaisutapoja. Karttapalvelu puolestaan käyttää paikan hakemiseen satelliittikoordinaatteja. Sen takia osoite ja koordinaatit on yhdistettävä toisiinsa.

Tätä varten karttapalvelussa on oltava tiedot haettavista osoitteista, jotta se pystyy niiden perusteella muodostamaan leveys- ja pituusasteet sisältävän koordinaattiparin. Osoitteen ja sijaintipaikan yhdistämistä kutsutaan geokoodaukseksi. Tällöin käyttäjä antaa haluamansa katuosoitteen, ja palvelu muuntaa sen koordinaateiksi, joilla sijainti voidaan paikallistaa kartalla.

Päinvastaista toimintaa, kohdepisteen muuntamista osoitemuotoon, kutsutaan käänteiseksi geokoodaukseksi. Toiminnot ovat suhteellisen raskaita suorittaa ja niinpä niitä on luvallista hyödyntää pääsääntöisesti palveluiden käyttöehtojen mukaan vain saman palvelun kanssa.

Karttapalvelujen tietokannoissa voi olla yksilöllisiä eroja, mutta pääsääntöisesti ne rakentuvat siten, että taulussa on ainakin osoite ja sitä vastaavat koordinaatit. Myös muuta kohteeseen liittyvää tietoa, kuten kohteen tyyppi (baari, ravintola, hotelli yms.), voi olla tallennettu tietokantaan.

Kohteet esitetään kartalla yleensä jonkinlaisen nastaa esittävän symbolin avulla ja reitti lähtöpaikasta kohteeseen viivalla. Palvelusta riippuen muitakin kohteita voi olla merkittynä, kuten ravintolat, nähtävyydet ja julkisten kulkuneuvojen pysäkit. Joskus reitti voidaan antaa myös tekstimuotoisena, reittiselostuksen muodossa.

5 Reitin määritys laskennallisena ongelmana

Reitin hakeminen ja siihen käytettävä algoritmi (menettelytapa) ovat karttapalvelun kannalta ehkä keskeisimmät toiminnot, jonka takia niitä käsitellään tässä tarkemmin.

Yksinkertaisin ratkaisu reitinhakuun on niin kutsuttu leveyssuunnattu haku. Se tutkii kaikki samalla etäisyydellä olevat kohteet ensin ennen kuin jatkaa eteenpäin. Ongelmana on, että algoritmissa pitää tallentaa varsinaisen kohteen lisäksi kaikki välissä olevat välietapit eli polku kohteeseen. Muistinkäyttö on siis lineaarista.

Monipuolisempia ovat A* ja Dijkstran algoritmit. Dijkstran algoritmi etenee lähtöpisteestä tasaisesti kaikkiin suuntiin tallentaen samalla tiedon yksittäiseen solmuun johtavasta lyhimmästä polusta. A* puolestaan huomioi myös maalin ja lähtöpaikan välisten solmujen hyvyyden ja valitsee sitten lyhimmän reitin. Se on siis Dijkstran algoritmi, johon on lisätty jäljellä olevaa matkaa laskeva osuus. Kun se saavuttaa loppupisteen, algoritmi suorittaa vertailun lyhimmästä reitistä. Tätä sanotaan myös pienimmän kustannuksen poluksi. [Kokkarinen 2002: 97-105.]

5.1 Dijkstran algoritmi

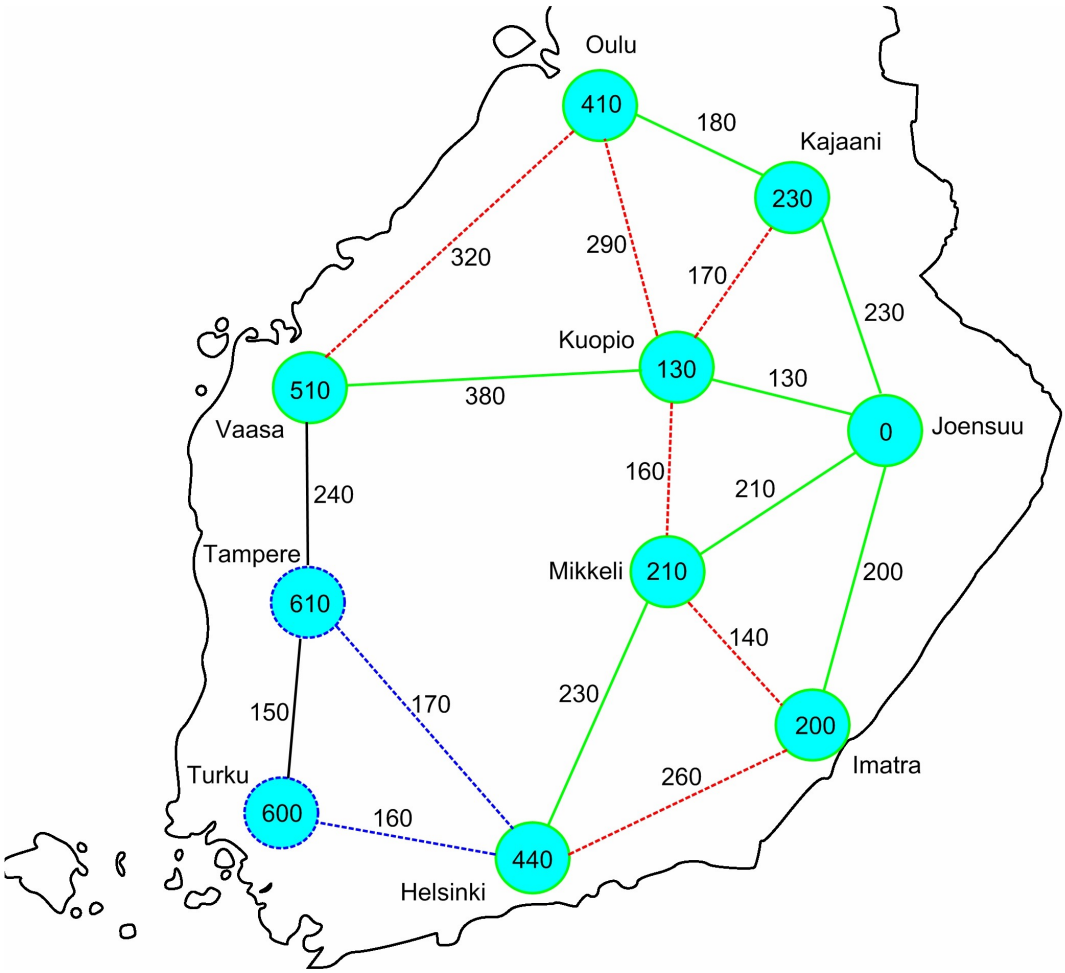
Aloitetaan merkitsemällä aloitussolmu vaikkapa K-merkinnällä (K = kyllä). Tässä K-merkintä tarkoittaa optimaalista reittiä alku- ja loppupisteen välillä. Sitten tarkistetaan, ovatko kaikki reitit aloitussolmusta eli juurisolmusta samanarvoisia (kaikki reitit yhtä hyviä tai huonoja). Jos on, lopetetaan hommat siihen. Muuten merkitään aloitussolmuun (tai juurisolmuun) suoraan kytkeytyvät polut ja niiden päässä olevat solmut merkinnällä H eli harkinnanvarainen. Samalla laitetaan H merkinnän oheen ketjun pituus kyseisestä solmusta aloitussolmulle asti.

Tarkistetaan, onko jokin H-merkinnän omaava solmu useamman kuin yhden harkinnanvaraisen polun päässä. Merkitään lyhyempi reitti ja siihen liittyvä solmu K-merkinnällä, muuten lyödään merkintä E (E = ei). [Kotilainen 2011.]

Jatketaan näin, kunnes kaikki reitit on käyty läpi. Tulokseksi jää optimireitti (muissa E-merkintä). Joskus optimireittejä voi olla useitakin.

Seuraavassa on Jussi Kotilaisen tekemä esitys Dijkstran algoritmista. Esimerkissä valitaan algoritmisesti lyhin reitti eri kaupunkien välillä. Kuvassa on yhtenäisellä viivalla merkitty vahvistetut reitit.

Otetaan esimerkiksi reitti väliltä Joensuu – Helsinki. Sinne menee useita reittejä, mutta suorimmat näyttäisivät kuvan 6 perusteella olevan Imatran tai Mikkelin kautta. Imatran kautta näyttäisi olevan hieman lyhyempi. Kun päästään seuraavaan vaiheeseen, huomataan että reitti Mikkelin kautta onkin lyhyempi, kun lasketaan koko matkan yhteispituus (Imatra = 200 + 260 = 460 ja Mikkeli = 210 + 230 = 440).



Kuva 6: Dijkstran algoritmi toiminnassa (välillä Joensuu - Helsinki).

Otetaan sitten toisena tarkempana esimerkkinä vaikka matka välille Oulu – Tampere. Koska Oulusta lähdetään, sen etäisyydeksi tulee nolla. Merkitään se myös vierailluksi. Muiden kuin lähikaupunkien etäisyyksiä ei vielä tiedetä, joten niitä ei tässä vaiheessa määritetä. Lähinaapureina ovat Vaasa, Kuopio ja Kajaani. Kajaaniin näyttäisi olevan lyhin matka (kolminumeroinen luku viivan vieressä).

Merkitään Kajaanin kohdalle 180. Tämän jälkeen tutkitaan sen lähikaupungit. Koska Oulu on merkitty vierailluksi, sitä ei oteta lukuun. Näin ollen lähikaupungeiksi jäävät

Kuopio ja Joensuu. Lisätään Joensuu listalle ja sen etäisyydeksi matkat väleiltä Oulu – Kajaani ja Kajaani – Joensuu eli 180 + 230. Sama tehdään Kuopion kohdalla (180 + 170). Merkitään Kajaani vierailluksi, jotta se ei sotke myöhemmässä vaiheessa matka-vertailuja.

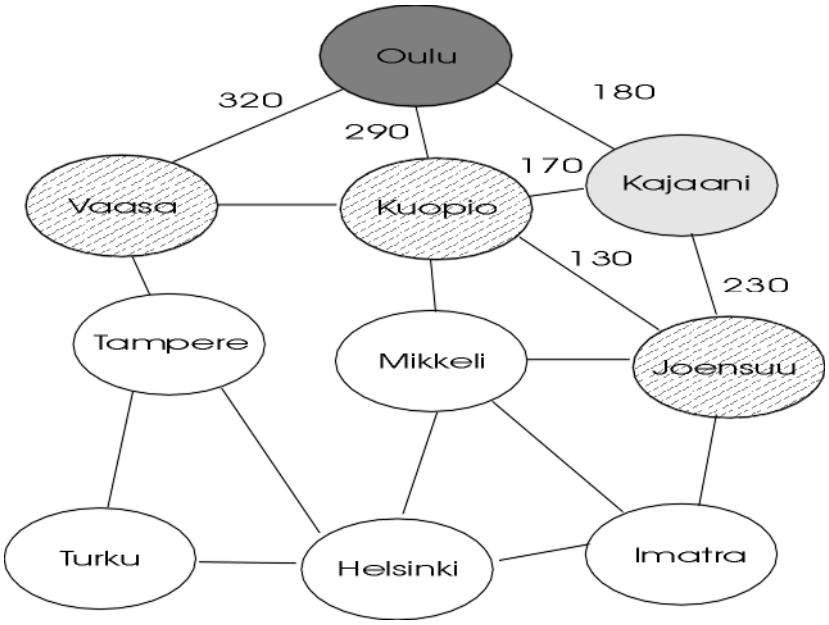
Nyt listalla ovat siis Oulun lähikaupungit Vaasa, Kuopio ja Kajaani sekä Joensuu (Kajaanin kautta). Kajaanin kautta on laskettu reitit Joensuun ja Kuopion kohdalle. Koska Kajaanin kautta kulkeva reitti Kuopioon on pidempi kuin suora reitti, ei sitä huomioida, vaan vanha etäisyys (Oulu – Kuopio) jää voimaan.

Siirrytään seuraavaan lähikaupunkiin, jossa ei ole vierailtu. Valitaan toiseksi lähin, joka tässä tapauksessa on Kuopio. Sen lähinaapureita ovat Oulu, Kajaani, Joensuu, Mikkeli ja Vaasa. Koska ensimmäiset kaksi on jo läpikäytyjä (vierailtuja), ei niihin palata enää tässä vaiheessa. Joensuu on lähin (Oulu → Kuopio – Joensuu eli 290 + 130), sitten tulee Mikkeli (Oulu → Kuopio – Mikkeli 290 + 160) ja viimeisenä Vaasa (Oulu → Kuopio – Vaasa 290 + 380).

Sitten siirrytään viimeiseen Oulun lähikaupunkiin eli Vaasaan, jonka lähikaupunkeja ovat Oulu, Kuopio ja Tampere. Koska Tampere on ainoa ryhmän vierailematon paikka, kartoitus voitaisiin periaatteessa lopettaa tähän, sillä se on päätepiste. Lasketaan reitti sinne (Oulu – Vaasa ja Vaasa – Tampere eli 320 + 240). Vaasan kautta on lyhin reitti, minkä voi varmistaa laskemalla muiden kaupunkien kautta kulkevat reitit yhteen (etäisyydet).

Varmistetaan kuitenkin vielä, että kyseessä on todella lyhin mahdollinen reitti. Mikkelin ja Joensuun kautta kulki vielä pari mahdollista reittiä (eli reitit Oulu → Kuopio – Mikkeli – Helsinki – Tampere ja Oulu → Kuopio – Joensuu – Imatra – Helsinki - Tampere). Selvitetään niiden kautta kuljetut matkat. Jos menisimme Mikkelin kautta, matka olisi 290 + 160 + 230 + 170 = 750. Vastaavasti Joensuun kautta kuljettu matka olisi 290 + 130 + 200 + 260 + 170 = 1050.

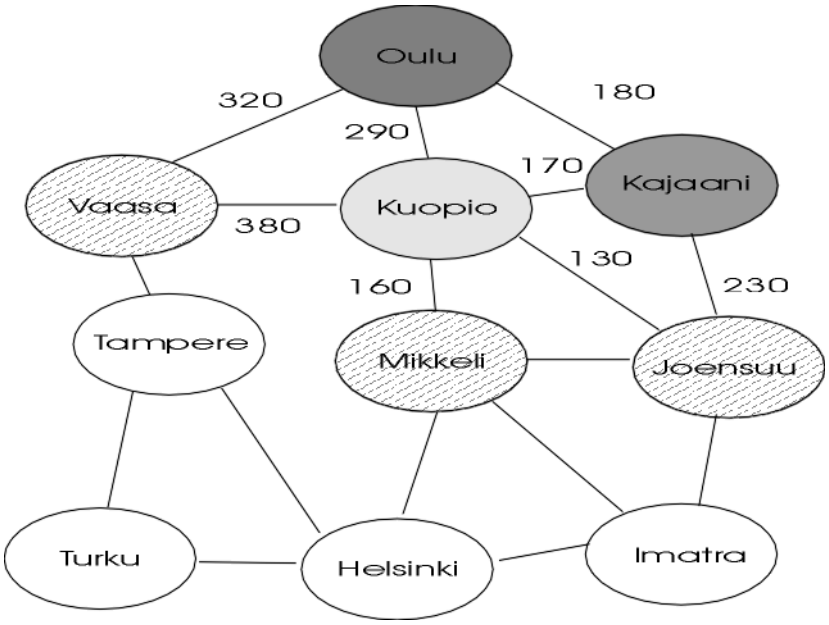
Kuvien 7, 8 ja 9 kautta on yllä oleva kuvaus kuvamuodossa. Lähtökohtana Oulu on siis jo valmiiksi vierailtu ja sieltä lyhin reitti on Kajaaniin. Kajaani otetaan siis ensimmäiseksi käsittelyn alle ja tutkitaan sen lähinaapurit (Ouluun ei enää mennä, joten jätän sen jatkossa mainitsematta).



Kuva 7: Alkutilanne, jossa Kajaani on käsiteltävänä.

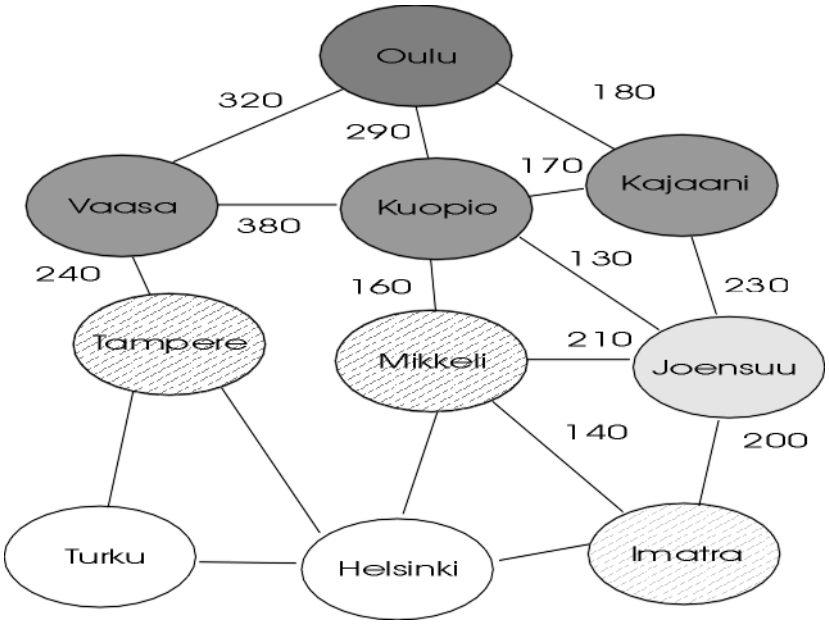
Kajaanilla on Oulu pois lukien kaksi rajanaapuria: Kuopio ja Joensuu. Nämä on merkitty raidoituksella (Vaasa ei ole Kajaanin vaan Oulun rajanaapuri, mutta se käsitellään myöhemmin). Tästä myös huomaa selvästi sen, että käsittelyssä oleva kaupunki ei tiedä naapureidensa lähinaapureiden etäisyyksiä.

Käydään seuraavaksi Kuopion kimppuun, jolla on Kajaanin kanssa yhteinen rajanaapuri, Joensuu. Muina naapureina ovat Mikkeli ja Vaasa. Lyhin reitti on Joensuuhun, joten siihen palataan kohta.



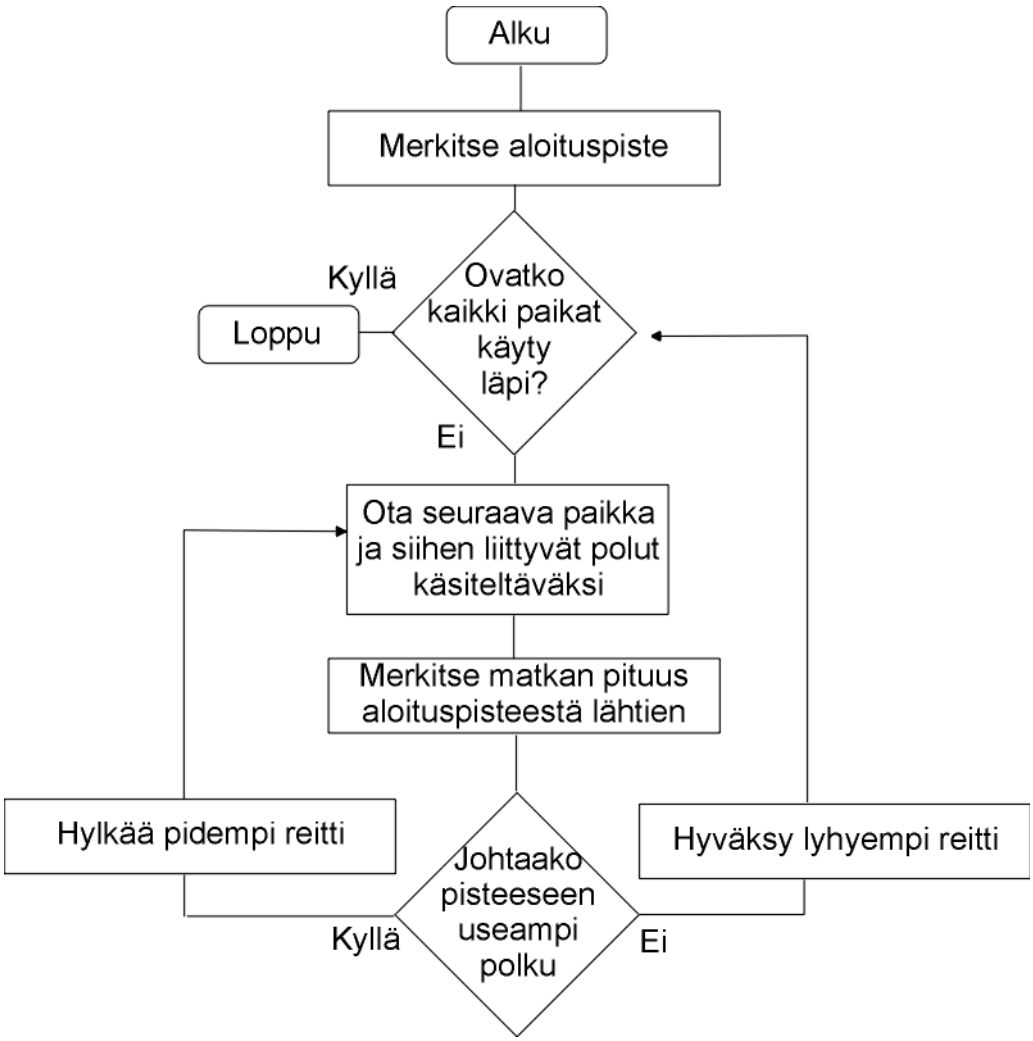
Kuva 8: Kuopio ja sen kolme käsittelemätöntä naapuria.

Otetaan sitä ennen Vaasa Oulun rajakaupungeista käsiteltäväksi. Siitä onkin suora reitti Tampereelle. Jotta voitaisiin olla varmoja, että kyseessä on suorin (lyhin) reitti Tampereelle, olisi vielä laskettava reitit Joensuun ja Mikkelin kautta Tampereelle (tällöin Helsinki on reitillä). Tästä kuitenkin toivottavasti selvisi algoritmin toiminta pääpiirteittäin.



Kuva 9: Kuopiosta on siirrytty Joensuuhun.

Yllä olevissa kuvissa 7-9 oli esimerkkinä näytetty reitin muodostuksen alkua, mutta Dijkstran algoritmi voidaan esittää myös vuokaaviomuodossa:

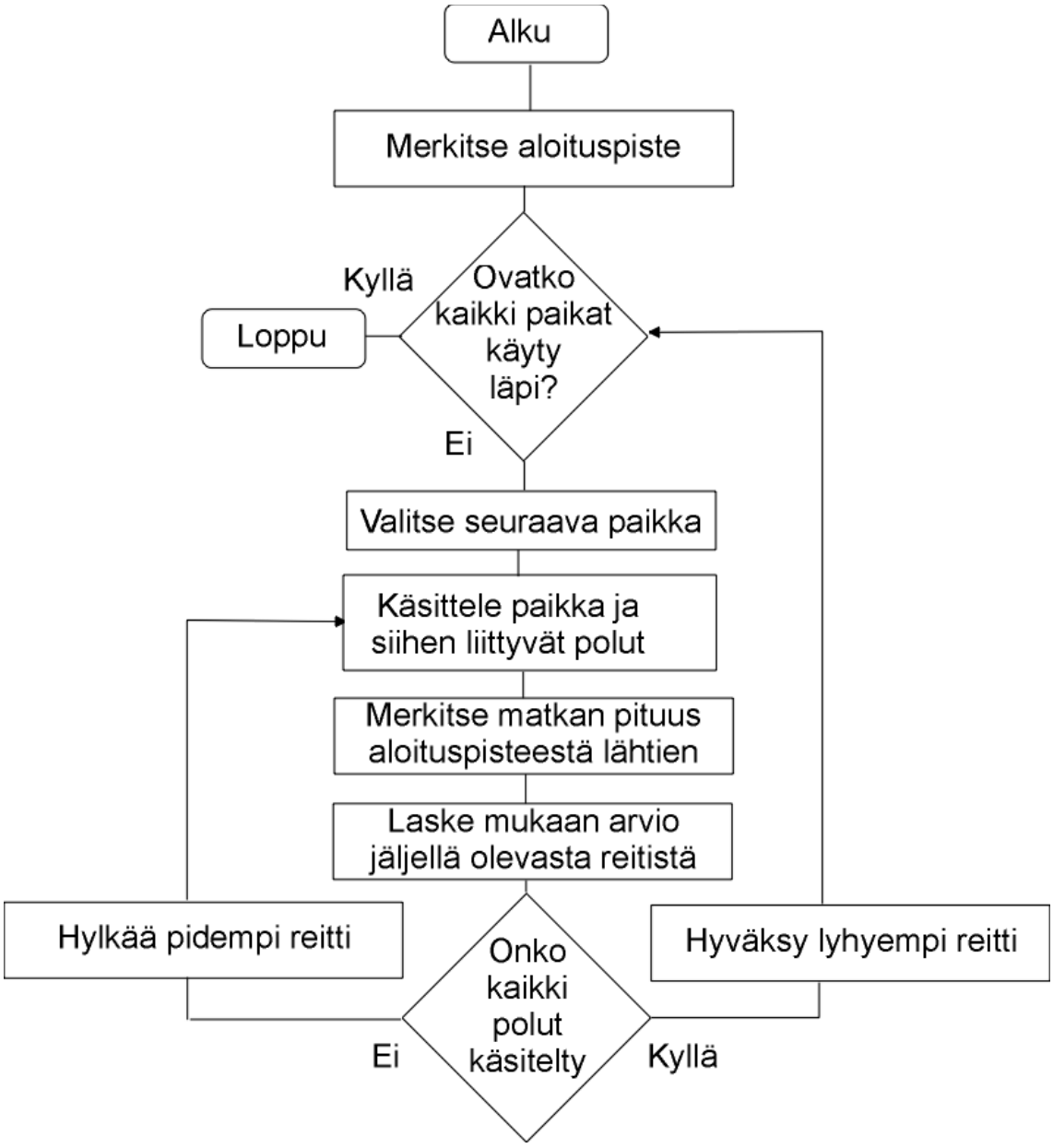


Kuva 10: Dijkstran algoritmi vuokaaviomuodossa.

5.2 A-tähti-algoritmi

A*-algoritmi toimii lähes samalla tavalla kuin yllä esitetty Dijkstran-algoritmikin, mutta sillä erotuksella, että siinä reitin hyvyys ei määräydy pelkästään yksittäisen solmun ja juurisolmun etäisyyden perusteella, vaan se lasketaan summasta, joka saadaan laskemalla yhteen juurisolmun ja sen hetkisen solmun etäisyys sekä arvio solmun ja kohdesolmun välisestä etäisyydestä [Siren 2007].

A*-algoritmi voidaan niin ikään esittää vuokaaviona, jossa on yllä esitettyyn algoritmiin tehty muutamia lisäyksiä:



Kuva 11: A* algoritmi vuokaaviomuodossa.

6 Ohjelmistoprojekti

6.1 Toteutettava karttapalvelu

Karttapalvelun rakentaminen vaihe vaiheelta tapahtui karkeasti ottaen seuraavalla tavalla:

1. Tutustuminen aiheeseen (mitä karttapalvelut ovat ja mitä toimintoja käytössä olevissa palveluissa on).
2. Tutustuminen Google Maps -palveluun.
3. Tutustuminen siihen, mitä Vuokraforum-sivustolla olevaan karttapalveluun tarvitaan (määrittelyvaihe). Ensimmäiset vaatimukset tulevat tässä vaiheessa asiakkaan suunnalta.
4. Omalla koneella suoritettavan luonnoksen tekeminen, jotta Googlen rajapinta ja toiminnallisuus tulevat tutummiksi.
5. Vuokraforumin siirtäminen omalle koneelle käyttövalmiiksi, jotta palvelun kehitys voidaan aloittaa.
6. Toiminnallisuuden tekeminen sivustolle. Tämä vaihe sisältää sekä koodaamista että testaamista.
7. Valmiin, testatusti toimivan sovelluksen lähettäminen asiakkaalle.

Vesiputousmallissa (ks. esim. Tervakari 2011) tehdään ensin suunnitelmat, laaditaan tuote, testataan ja otetaan se käyttöön. Näin teoriassa. Käytännössä virheiden takia joudutaan testausvaiheesta hyppäämään suunnitelmavaiheeseen useastikin, mikä maksaa aikaa ja samalla rahaa. Tämän projektin pienuuden vuoksi päädyttiin asiakkaan kanssa kuitenkin vesiputousmalliin (osaltaan siksikin, koska varsinaista keskitettyä kokoontumispaikkaa ei ole).

Projekti toteutettiin käyttämällä vesiputousmallia, joka on eräs vanhimpia ohjelmistonkehitysmalleja. Siinä ohjelmistoprojektista tehdään ensin kattava määrittely, joka tehdään yhdessä asiakkaan kanssa. Määrittelyssä kerrotaan olennaiset seikat ja ominaisuudet, jotka on oltava mukana projektissa.

Määrittelystä siirrytään suunnitelmien laatimiseen, jossa tarkemmin kuvataan ohjelmiston toiminnat, tekniikkaa sekä tietokanta.

Tämän jälkeen alkaa projektin toteutus eli koodaus. Projekti yritetään toteuttaa mahdollisimman hyvin suunnitelmien mukaisesti. Toteutusvaiheen jälkeen seuraa projektin testaus ja koulutus. Työ ei vielä pääty tähän vaan usein sitä on vielä ylläpidettävä sopimusten mukaisesti.

Ongelmana vesiputousmallissa on lineaarisuus. Oletus, että kaikki toimii heti, on virheellinen. Testausvaiheesta siirtyminen takaisin toteutukseen ja usein vielä suunnitelmavaiheeseen vaatii paljon aikaa ja rahaa. Niinpä on kehitetty useita muita menetelmiä, joita kutsutaan ketteriksi menetelmiksi. Näitä ei kuitenkaan käsitellä tässä tämän enempää.

Projektin määrittelyvaiheessa päätettiin ohjelmistotuotteen visio (päätavoite), toiminnallisuus ja käyttöliittymä. Nämä kuvataan lyhyesti, jonka jälkeen esitellään suunnitteluvaiheessa valitut tekniikat.

6.2 Visio

Projekti alkoi visiosta, joka tehtiin yhdessä asiakkaan kanssa. Asiakastapaamisen lisäksi käytiin sähköpostinvaihtoa, jossa selvisi tarkemmin projektin yksityiskohtia.

Tehtävänä on tehdä Oikotien tapainen palvelu, jossa haetut asunnot näkyvät kartalta. Näkymän voi vaihtoehtoisesti muuttaa listamuotoon, jolloin asuntojen osoitteet näkyvät listana. Asuntojen näyttöaikojen perusteella lasketaan optimaalinen reitti eri asuntojen välille. Viimeisenä ajatuksena palvelussa näkyvän kartan voi valita väliltä kartta / ilmakuva / lintuperspektiivi.

Myöhemmin projekti muuttui määrittelyn osalta siten, että optimireitti lasketaankin ”asuntokierrokselle” lisättyjen asuntojen väliltä. Kävijä voi tällöin suunnitella vapaasti kiertokäyntinsä.

6.3 Toiminnallisuus

Vuokraforumille tehtävä karttapalvelu toimii seuraavassa kuvatulla tavalla.

Kun kävijä saapuu sivustolle ja siirtyy siellä asuntoilmoituksia käsittelevään osioon, hän näkee vasemmalla puolella asunnot listana. Selailemalla listaa alemmas asuntoja ilmestyy lisää. Valitsemalla jonkin kohteen ilmestyvät sen tiedot sivun oikealle puolelle, kartan kera.

Karttanäkymän yläosassa olevilla painikkeilla voi näkymän muuttaa normaalista viivastokartasta ilmakuvaksi.

Näkymän voi muuttaa normaalista karttakuvasta katunäkymään, jossa näkyy kameralla otettu näkymä kohteen kohdalta. Molemmat toiminnot ovat Google Mapsin vakiokalustoa.

Käyttäjän antaessa oman kotiosoitteensa ohjelma muodostaa optimaalisen reitin halutuille kohteille. Reitti piirretään kulkemaan asuntokierrokselle lisättyjen asuntojen kautta. Kävijälle on annettu vapaus ja vastuu valita asunnot haluamassaan järjestyksessä.

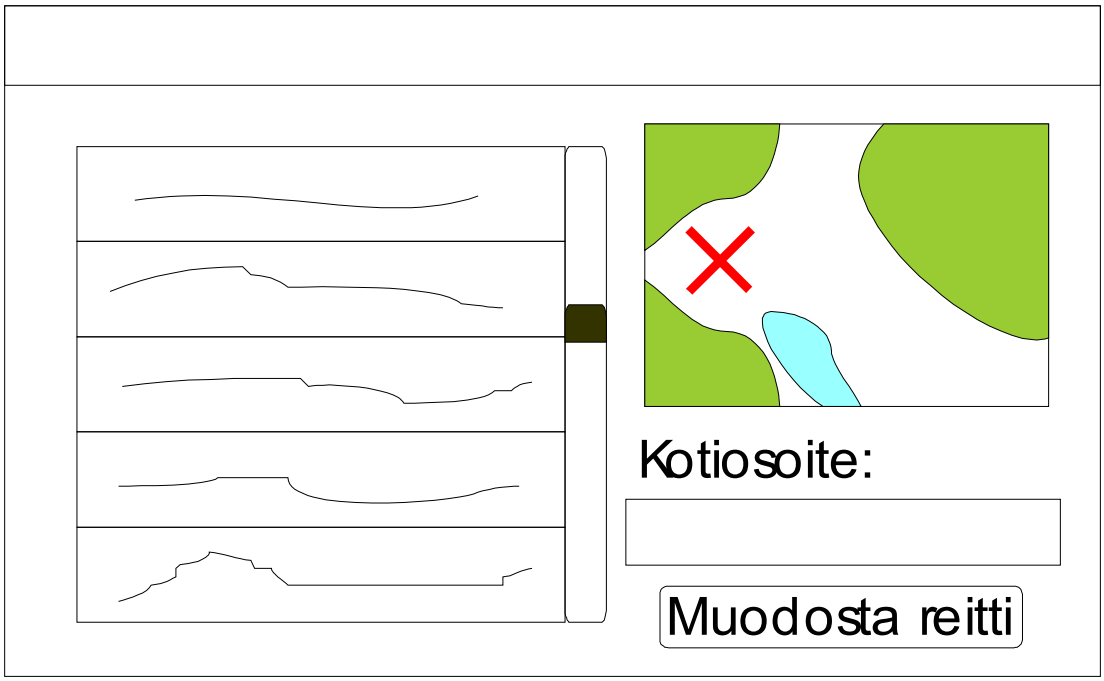
6.4 Käyttöliittymä

Vuokraforum-sivuston perusrakenne noudattaa kaavaa, jossa vasemmalla sijaitsevat asunnot osoitetietoineen ja oikealle avautuvat valitun asunnon tiedot, karttakuvan kera. Karttakuva ja sen päällä olevat kontrollit ovat käytännössä se käyttöliittymä, jonka parissa tässä työssä operoidaan. Nämä kontrollit (ohjaimet) ovat Googlen omia eli niitä ei tarvitse itse koodata.

Ainoa varsinainen käyttöliittymään liittyvä tehtävä onkin laittaa asetukset niin, että tarpeelliset ohjaimet ovat näkyvissä karttakuvan päällä (näitä ovat liikkuminen ja karttatyypin vaihto).

Lisäksi tarvitaan valintapainikkeet karttanäkymän ja listamuodon vaihtamista varten, painikkeet karttakuvan ja katunäkymän vaihtamiseksi sekä tarvittava lomake osoitteen kirjoittamista varten (mikäli kävijä haluaa selvittää optimireitin).

Näistä voisi tulla esimerkiksi seuraavan kuvan 9 kaltainen liittymä:



Kuva 12: Käyttöliittymän luonnos.

6.5 Tekniikat ja työvälineet

Työ nojautuu sekä palvelimella olevaan tietokantaan että Googlen Maps -palveluun. Googlen valintaa puolsi se, että asiakasyritys on käyttänyt sitä ennestään ja kyseessä on ilmainen palvelu. Googlella on myös hyvä ohjeistus palvelun käytöstä. Sitä käytetään JavaScript-koodista käsin, joka muodostetaan joko PHP- tai Ruby-tekniikalla. JavaScript on selainten ohjelmointikielistä tuetuin, joten tämän takia toteutus pohjautuu siihen.

Työvälineenä käytetään tekstinmuokkausohjelmaa, joka kykenee tallentamaan tiedot puhtaassa tekstimuodossa (valitsin ohjelmaksi Notepad ++, koska se on suosittu ja sisältää monia käteviä toimintoja, joista vähäisin ei ole kirjoitustyötä helpottava värikoodaus). Linux-puolella tekstinmuokkaukseen käytetään puolestaan Ubuntu-jakelupaketin mukana tullutta tekstieditoria. Versionhallintaohjelmana käytetään git-ohjelmaa, koska se on asiakkaallakin käytössä.

Käytettävä relaatiotietokanta on MySQL-pohjainen. Tärkein sen tauluista työtä ajatellen on eri asuinkohteiden sijaintitiedot sisältävä taulu, jossa ovat koordinaatit Google Map-sin ymmärtämässä muodossa.

6.5.1 Ruby

Ruby on olio-ohjelmointiin suuntautunut tulkattava ohjelmointikieli. Tällaisia ohjelmia saatetaan joissakin yhteyksissä nimittää myös skripteiksi (fiktiivisen kirjan tai elokuvan käsikirjoitus = a manuscript).

Tulkattavassa kielessä lähdekoodi tulkataan konekielelle suorituksen aikana rivi kerrallaan. Käännettävässä kielessä ohjelma käännetään konekielelle kerran, jonka tuloksena on ajokelpoinen ohjelma (ajokelpoinen tiedosto). Näiden yhdistelmää sanotaan puolikäännettäväksi (tai -tulkattavaksi) kieleksi, jossa lähdekoodi käännetään välikielelle ja siitä konekielelle, kun ohjelma käynnistetään. Ruby kuuluu ensimmäiseen eli tulkattavien kielten ryhmään.

Koska Ruby-ohjelma käännetään aina suoritusvaiheessa, tarvitaan tulkkiohjelma. Tämä suorittaa tekstimuodossa olevat käskyt eli lähdekoodin. Ruby-tulkkeja on olemassa ainakin Linux- ja Windows-ympäristöihin. Ruby on käytännössä paremmin kotonaan Linux-ympäristössä, joten kokeiluja varten saattaa olla kannattavaa asentaa virtuaalikone ja siihen Linux, mikäli sitä ei vielä ole. Hyviä vaihtoehtoja ovat esimerkiksi Oraclen VirtualBox ja VMwaren tuotteet. [Addison 2001.]

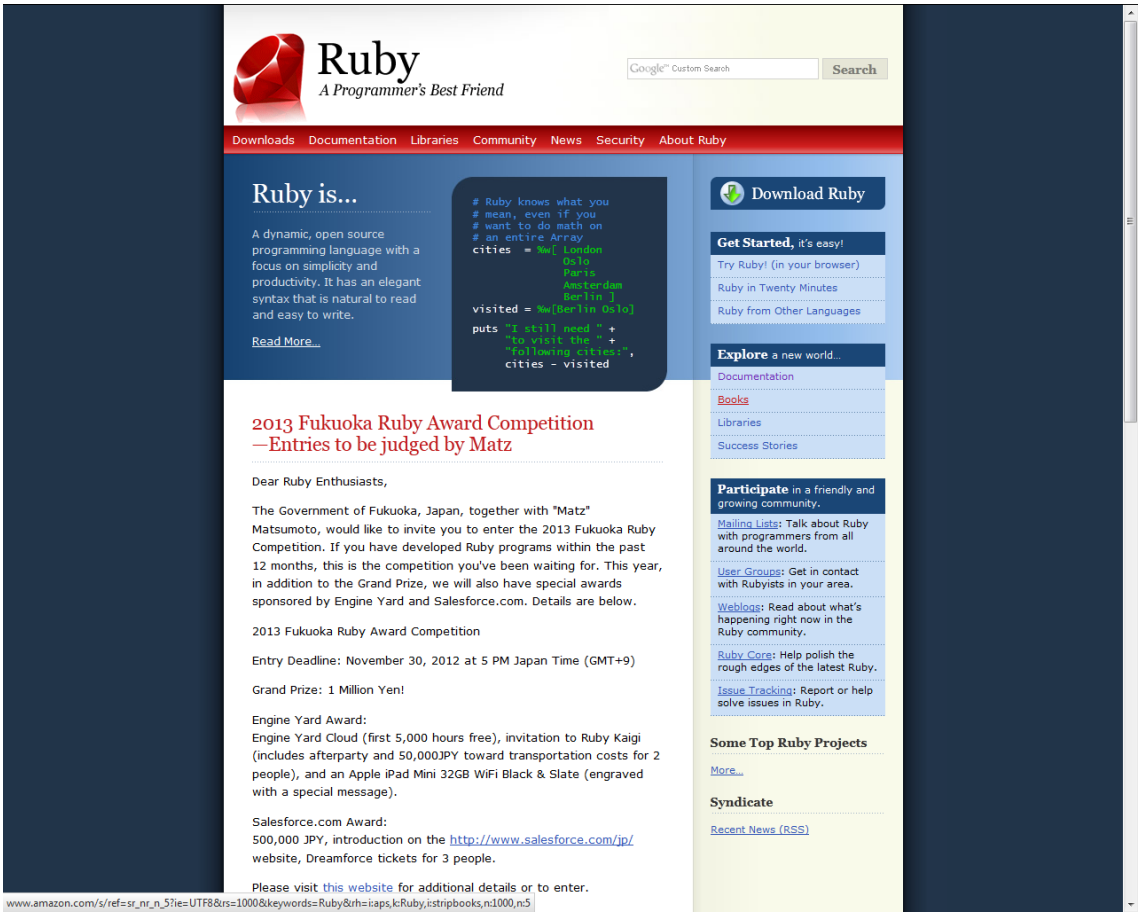
Olio-ohjelmointi tarkoittaa lyhyesti sitä, että koodissa ei keskitytä liikaa tekniseen rakenteeseen (kuinka tiedot tallennetaan tietokoneen muistiin ja käsitellään siellä) vaan itse tietoon. Oliot edustavat jokapäiväisessä elämässä olevia asioita, joita ohjelmassa käytetään (niistä luetaan tietoa tai niihin kirjoitetaan tietoa). Otetaan esimerkiksi Kaukon Liikenne -niminen kuvitteellinen demo-ohjelma. Ohjelmassa olevia olioita voisivat olla matkalippu, kortinlukija ja liikenneväline (bussi, laiva, juna yms.). Kun matkalippu annetaan kortinlukijalle, se ensin muuttaa kortin tilan (mitätöi sen) ja antaa merkin vaikkapa bussioliolle, että maksu on suoritettu. Tämän jälkeen matka voi jatkua seuraavalle pysäkille.

Rubyssa on menty tavallista pidemmälle ja siinä kaikki ovat objekteja eli olioita. Näitä ovat numerot, funktiot, rakenteet, taulukot ja itse oliot. Tyypillisesti tehokkuussyistä esimerkiksi ns. alkeistietotyypit eli erilaiset luvut ja yksittäiset merkit eivät ole olioita. Ru-

byssa tavoitteena on kuitenkin ollut kieli, jota voisi käyttää ihmisten eikä tietokoneiden ehdoilla. [Sagell 2008.]

Rubyn yhteydessä mainitaan usein Ruby On Rails, joka on tunnetuin Ruby-pohjainen sovelluskehys web-pohjaiseen ohjelmointiin. Web-puolen ohjelmoinnissa nämä kaksi liittyvätkin usein yhteen.

Kuvassa 10 on Ruby-kielen virallinen sivusto:



Kuva 13: Ruby-kielen sivusto.

6.5.2 Ruby On Rails

Ruby On Rails on yleisin Ruby-ohjelmoinnissa käytettävä rajapinta- tai sovelluskehys. Se helpottaa monien rutiiniluonteisten tai muuten hankalasti tehtävien toimintojen laatimista sovelluksiin. Kehys pohjautuu MVC-suunnittelumalliin, jossa kantavana ajatuksena on erotella tiedon näyttäminen, välittäminen ja käsittely toisistaan. Näistä käytetään

nimityksiä näkymä, ohjain (kontrolleri) ja logiikka (business logiikka) tai malli. Jälkimmäinen on yleisempi nimitys Rails-maailmassa.

Rails toimii tavallisimpien palvelinohjelmistojen kanssa. Tuettuina ovat esimerkiksi Apache, Mongrel tai Railsin mukana tuleva pieni WEBrick -palvelin, joka soveltuu erinomaisesti testauskäyttöön, jos muuta palvelinta ei ole parhaillaan asennettuna ja toiminnassa.

Kun nettikäyttäjä pyytää selainta siirtymään Railsillä toteutettuun sivustoon, hän kirjoittaa osoiteriville ohjaimen nimen (ohjaimena toimii sivuston oletustiedosto, joka useimmiten on nimeltään ”index” tai ”default”). Käytännössä riittää, että kirjoittaa pelkän hakemistoviittauksen. Ohjain puolestaan lataa halutut näkymät esille ja lähettää käyttäjän antamat tiedot (syötteet) edelleen logiikkapuolen käsiteltäväksi. Tämän tavan ehdoton etu on se, että osoiterivi säilyy ”puhtaana”. Toisaalta myöskään pahantekijät eivät saa turhan helposti selville yksittäisten dokumenttitiedostojen nimiä.

Tässä ei käydä sen tarkemmin läpi asentamista, sillä sen voi jokainen tehdä, mikäli kiinnostusta riittää, Railsin sivuilla olevien ohjeiden mukaan. Tässä käydään ainoastaan pikaisesti läpi peruskomennot eli miten luodaan uusi sovellus, määritetään sen käyttämä tietokanta ja käynnistetään se.

Uuden sovelluksen luonti tapahtuu komennolla

```
rails new sovellus
```

Tuloksena syntyy reilun kymmenen hakemiston rakenne, jossa app/ -hakemisto sisältää MVC-mallin mukaiset palaset. Sovelluksessa käytetään oletusasetuksia ja oletustietokantaa (SQLite).

Sovelluksesta puuttuu vielä toiminnallisuus. Se tapahtuu luomalla ohjain ja näkymä komennolla

```
rake generate controller home index
```

Sitten tarvitaan vielä tietovarasto. Tietokannan muodostaminen tapahtuu komennolla

```
rake db:create
```

Sovelluksen testaamiseksi on se käynnistettävä, jotta selaimella voidaan nähdä valmis sovellus. Siksi viimeinen komento on

```
rake server
```

Sovellus sijaitsee oletuksena portissa 3000, joten suuntaamalla osoitteeseen http://localhost:3000 pitäisi ilmaantua Railsin oletussivu.

6.5.3 Google Maps

Maps on Googlen oma versio karttapalvelusta, jota voidaan Googlen oman karttahaun (http://maps.google.fi/) lisäksi käyttää omista sovelluksista käsin käyttäen JavaScript-koodia, jota lähes kaikki nykyisin käytössä olevat verkkoselaimet tukevat. Jatkossa käytän lyhennettä GM tarkoittamaan Google Mapsia.

GM tukee monia karttapalvelun perustoimintoja, kuten kartan vierittämistä, lähentämistä, loitontamista ja viiva / ilmanäkymää. Kartalla voi myös koodista käsin piirtää viivoja, ympyröitä ja kuvakkeita mutta ei tekstiä. Kohteet merkitään oletuksena nuppineulan kuvalla ja näiden välillä voi olla välietappeja. GM tukee optimireitin muodostamista, jossa lasketaan lyhin reitti kahden pisteen välillä, mahdollisesti välietapit huomioiden.

Lisäksi GM tukee geokoodausta. Se tarkoittaa osoitteen muuntamista karttakoordinaateiksi, leveysasteiksi (pohjois-etelä) ja pituusasteiksi (itä-länsi). Osoite voi olla täydellinen katuosoite tai koostua pelkästä postinumerosta.

Google Maps JavaScript API versio 3 on tämänhetkinen versio Mapsin rajapinnasta. Sen vanhempi kakkosversio on yhä käytettävissä, mutta sitä ei enää suositella käytettäväksi. Molemmat löytyvät Googlen kehitysalueelta (http://code.google.com), josta löytyy myös paljon muitakin palveluja, joita kehittäjät voivat hyödyntää, kuten mainostusohjelma AdSense ja YouTube-videopalvelu.

Tässä yhteydessä sopii myös esitellä lyhyesti Google Earth. Tämä on selaimessa toimiva lisäosa, mikä mahdollistaa kävelyn kolmiulotteisessa ympäristössä, joka on mallinnettu todellista ympäristöä huomioiden. Oletusnäkymä on maapallo, jota napsauttamalla näkymä siirtyy lähemmäksi. Sovelmalle voi myös syöttää halutun pisteen koordinaatit, ja näkymä siirtyy kuvaamaan kyseistä paikkaa.

Google Earth -palvelua käytetään Mapsin tavoin JavaScriptillä. Se mahdollistaa saman kuin Maps mutta kolmiulotteisena. Tämä vaatii kuitenkin tehoa, ja ennen kaikkea se edellyttää lisäpalikan asentamista selaimeen, minkä takia sen käyttö kannattaa jättää valinnaiseksi (käyttäjä voi esimerkiksi linkistä siirtyä kolmiulotteiseen näkymään). Vaikka maapalloa voi liikutella hiirellä, se voi erityisesti modeemikäyttäjille olla turhan raskas.

Jotta Googlen palvelut eivät ruuhkautuisi, on Mapsin käyttö rajoitettua. Jokainen karttapalvelun käyttäjä voi ladata kartan enintään 25 000 kertaa vuorokauden aikana. Käyttäjä tarkoittaa tässä sivulla vierailijaa. Poikkeuksena ovat ensimmäiset kolme kuukautta, jonka aikana ei ole rajoituksia.

Karttakuva latautuu div-elementin sisälle, joten sille on annettava jokin tunniste (id-määre). Tämä on XHTML-kielen elementti, jolla sivun voi jaotella osiin. Div-elementin sisälle voi kirjoittaa tekstiä, kuvan tai jotain muuta joka näkyy kun karttakuva ei näy. Lisäksi se näkyy hetken, kun karttakuva latautuu. Kannattaa kuitenkin miettiä, onko sopivaa kirjoittaa esim. ”Ladataan, odota hetki...”, sillä jos JavaScript on pois käytöstä, ei kuva lataudu koskaan ja käyttäjä voi jäädä hämmennyksiin. JavaScriptin osalta ongelman voi korjata siten, että latausteksti tulostetaan JavaScript-kielen omalla tulostuskomennolla eikä sijoiteta suoraan XHTML-tiedostoon. Sen sijaan jos Google Mapsin toiminnassa on jokin ongelma, saattaa teksti jäädä ruudulle. Tämä ei kuitenkaan ole kovin suuri ongelma.

Google Mapsin rajapinta ladataan Script-tunnisteen avulla seuraavasti:

```
  <script type="text/javascript"
                src="http://maps.googleapis.com/maps/api/js?
key=YOUR_API_KEY&sensor=SET_TO_TRUE_OR_FALSE">
    </script>
```

Key edustaa avainta, jolla pyritään ehkäisemään palvelujen väärinkäyttöä ja voidaan tehokkaammin valvoa erityisesti ilmaiskäyttäjille langetettujen latausrajojen noudattamista. Sensor-parametri voidaan ohittaa. Se kertoo, käytetäänkö navigaattorissa olevaa paikannussensoria käyttäjän sijainnin paikallistamiseen.

Kun API on ladattu, ovat Google Mapsin toiminnot käytettävissä.

Karttakuvan muodostaa Map-objekti, joka luodaan sopivassa yhteydessä. Useimmiten tämä tapahtuu webbisivun latauksen yhteydessä. Lataushetkeen pääsee kiinni luomalla funktio, joka kytketään OnLoad-tapahtumaan:

```
function init() {
          map = new gm.Map(document.getElementById("map"));
}
<body onload="init()">
<div id=”map”>
</div>
</body>
```

```
1) <html>
2) <head>
3) <meta http-equiv="Content-Type" content="text/html; charset=UTF-
   8" />
4) <style type="text/css">
5) <!--
6)       html { height: 100% }
7)       body { height: 100%; margin: 0; padding: 0 }
8)       #map { height: 100% }
9) -->
10)        </style>
11)
12)        <script type="text/javascript"
   src="http://maps.google.com/maps/api/js?sensor=false"></script>
13)        <script type="text/javascript">
14)        var gm = google.maps;
15)
16)        function init() {
17)                  var myOptions = {
18)                  zoom: 8,
19)                  center: new gm.LatLng(60.1664865, 24.9432303),
20)                  mapTypeId: gm.MapTypeId.ROADMAP,
21)                  streetViewControl: true
22)        };
23)
24)        new gm.Map(document.getElementById("map"), myOptions);
25)        }
26)        </script>
27)        </head>
28)
29)        <body onload="init()">a
30)        <div id="map" style="width:66%; height:100%;">
31)        <noscript>
32)        <img  src="http://maps.google.com/maps/api/staticmap?cen-
   ter=60.1664865,24.9432303&zoom=8&size=300x400&sensor=false" />
33)        </noscript>
34)        </div>
35)        </body>
36)        </html>
```

Kun nämä paketoidaan, saadaan seuraavalla sivulla oleva XHTML-koodi. Varsinainen JS-koodi on sijoitettu tiedostoon ”script.js”, jonka koodi löytyy liitteistä.

Seuraavassa esimerkissä on staattisen karttakuvan lataava koodi, sillä vaikka sitä ei voikaan käsitellä millään tavalla (siirtää, zoomata tai vaihtaa näkymää), on se parempi kuin ei mitään. Lisäksi kävijä näkee kohteen (loitompana) samalla, kun varsinainen karttaobjekti latautuu.

Asetukset ovat myOptions-nimisen rakenteen sisällä (rivit 17-22). Näistä *center*, *mapTypeId* ja *zoom* ovat pakollisia. Karttakuva piirretään center-kohdassa kerrottujen koordinaattien ympärille. Alustusvaiheessa (kun karttakuva ladataan sivulle) käytettävä karttanäkymä määräytyy mapTypeId-kohdassa ja zoomaus zoom-kohdassa. Street-

ViewControl määrittää, onko vasemmassa yläkulmassa oleva katunäkymän ohjain näkyvissä. Tämä toimii käytännössä pohjana, jonka päälle voi rakentaa edistyneempiä ratkaisuja.

Rivillä 24 luodaan karttaobjekti map-elementin sisälle. Rivillä 29 skripti määrätään suoritettavaksi, kun sivu latautuu. Rivillä 32 on sisältö, joka latautuu, mikäli JavaScript on pois päältä (tai selain ei tue sitä). Siinä yksinkertaisesti näytetään staattinen sivu, jossa kysymysmerkin jälkeen kerrotaan kartan näkymäasetukset.

Kaiken kaikkiaan Mapsin käyttö on suhteellisen yksinkertaista. Kyseessä on normaali HTML-sivu, jonka sisälle laitetaan div-elementti ja JavaScript-koodilla tehdään tarvittavat alustustoimenpiteet. Asetukset kerrotaan JSON-objektien muodossa.

JSON-objekti (JavaScript Object Notation) tarkoittaa käytännössä esimerkiksi C-kielestä tuttua rakennemuuttujaa (structure). Muuttuja siis ikään kuin sisältää useita erilaisia arvoja, jotka erotetaan toisistaan avaimilla. Avain/arvoparien erottimena toimii pilkku.

Esimerkkinä JSON-objektista toimii seuraava mapOptions-objekti:

```
var mapOptions = {
            zoom : 8,
            center: new gm.LatLng(lat, lng),
            mapTypeId: gm.MapTypeId.ROADMAP,
            streetView: panorama,
            streetViewControl: false
}
```

Esimerkissä on käytetty arvoina lukua, objekteja sekä totuusarvoa (tässä järjestyksessä). JSON-objekteissa on huolehdittava siitä, että avainten arvot vastaavat juuri API-dokumentaatiossa mainittuja. Isot ja pienet kirjaimet ovat eriarvoisia. Merkkijonojen ympäriltä ei myöskään tule unohtaa lainausmerkkejä. Nämä ovat varmaankin tuttuja seikkoja kaikille, jotka ovat joskus ohjelmoineet jollakin lausekielellä (Java, C#, Basic, C, C++, Pascal...).

6.6 Matkareitin määritys

Matkareitin määrittämisessä ja näyttämisessä karttakuvan päällä käytetään Google Mapsiin kuuluvia DirectionsService- ja DirectionsRenderer-palveluita, joista ensimmäinen hoitaa reitin laskemisen ja jälkimmäinen sen näyttämisen. Ennen kuin näitä voidaan käyttää, on objektit luotava samoin kuin tehtiin Map-objektin kanssa.

DirectionsService saa parametrikseen DirectionsRequest-objektin, joka sisältää muun ohella tiedot lähtö- ja kohdepaikasta. Lisäksi palvelulle voi kertoa välietapit, joiden kautta reitti muodostuu. Toisena parametrina on vastaajafuntio (callback), jonka parametreina on oltava DirectionsResult ja DirectionsStatus-oliot. Ensimmäinen palauttaa reitin (taulukko, jossa on enemmän kuin yksi alkio jos Mapsia pyydetään palauttamaan vaihtoehtoisia reittejä) ja toinen onnistumisen. Alkio on DirectionsRoute-olio, jonka sisältämästä tiedosta tärkein lienee legs, jossa on tieto reitin muodostavista askelista (lähtöpistettä ei lasketa mukaan). Tätä ei kuitenkaan tarvita lopputyössä, joten lopetan sen käsittelyn tähän.

Mikäli reitin määritys (reititys) onnistui, voi DirectionsResult-olion antaa DirectionsRenderer-olion käsiteltäväksi komennolla setDirections. Tämä piirtää reitin karttakuvaan näkyville. Jos reitin haluaa pois näkyvistä, käy se asettamalla setMap-ominaisuuden arvoksi null.

Ohjelmakoodin osa, jossa selvitetään reitti ja näytetään se kartalla:

```
1)      var waypoints = [];
2)      waypoints = waypoints.concat(tourPositions);
3)      waypoints.shift();
4)
5)      var directorOptions = {
6)         origin: document.getElementById("address").value,
7)         destination: tourPositions[tourPositions.length - 1],
8)         travelMode: gm.DirectionsTravelMode.DRIVING,
9)         waypoints: waypoints
10)                }
11)
12)     director.route(directorOptions, function (result, status){
13)                if (status == gm.DirectionsStatus.OK) {
14)                           renderer.setDirections(result);
15)                }
16)     });
```

Rivillä 12 lasketaan reitti directorOptions-rakenteen mukaisesti. Matka kuljetaan autolla (rivi 8), koska tällöin pois jätetään sellaiset kohdat jotka edellyttäisivät jalkautumista. Kaikkialle, minne pääsee autolla, pääsee myös pyörällä tai kävellen (maantieteellisesti, poikkeuksia ovat esimerkiksi moottoritiet). Rivillä 9 oleva waypoints sisältää reittipis-

teet, joiden kautta muodostettava reitti kulkee. Rivillä 3 tiputetaan lähtöpaikka pois, koska maksimietappien määrä on kahdeksan ilmaiskäyttäjille eikä lähtöpaikka ole välttämätön tässä yhteydessä. Rivillä 14 reitti piirretään kartalle.

6.7 Tekninen rakenne

Jokainen webissä oleva verkkosovellus koostuu periaatteessa samoista osista. Näkyvin osa on käyttöliittymä, jonka kautta kävijä voi antaa halutut tiedot (syötteet) ja nähdä lopputuloksen. Logiikka huolehtii tiedonhausta, tietojen hyödyntämisestä ja erinäisistä tarkistuksista. Lisäksi sovellukset käyttävät jotain tietokantaa tietojen säilyttämiseen. Sillä ei ole karttasovelluksen kannalta suurtakaan merkitystä, sillä kaikista löytyy toiminnot, joilla haetaan tietokannasta tietoja (SELECT-lauseet).

Tämä sovellus rakentuu kahdesta MVC-mallin mukaisesti erotetusta osuudesta: toisella hoidetaan logiikka ja toisella tulosten näyttäminen ruudulle. Sovelluksessa logiikka on keskitetty yhteen paikkaan ja tulosten näyttäminen puolestaan useampaan kohtaan XHTML-koodin sekaan käyttäen pääosin parin rivin mittaisia koodinpalasia.

Sovellus toimii sekä selaimessa (asiakaspäässä) että palvelimella (palvelinpäässä). Asiakaspäässä käytetään JavaScriptiä, jolla saadaan aikaan karttanäkymä. Palvelinpäässä haetaan kohteiden tiedot tietokannasta ja asetetaan ne JavaScript-koodin sekaan, joka sitten lähetetään XHTML-muodossa asiakkaalle. JS-funktiot sijoitetaan erilleen XHTML-dokumentista.

Fyysisesti ohjelma koostuu noin kymmenestä eri hakemistosta, joista ”\forum\app” on tärkein. Forum on vain sopivaksi katsottu nimi karttapalvelun kotihakemistolle. Ruby-sovellukset tallentavat tiedot app-hakemiston alle, jossa sijaitsee näkymät, ohjaimet ja logiikka. Tässä muutokset ovat kohdistuneet lähinnä näkymäpuolelle, koska JS-koodi tulkataan selainpäässä ja se ladataan normaalin XHTML-sivun mukana. Ruby-projektissa olevat Ruby-kieliset tiedostot varustetaan rb-päätteellä (tarkentimella) tai erb-päättellä (Embedded Ruby). Jälkimmäiset mahdollistavat Ruby-koodin sijoittamisen normaalin XHTML-koodin sekaan, joten näkymät koostuvatkin tämäntyyppisistä tiedostoista.

Suurin osa opinnäytetyöstä liittyykin logiikkapuoleen, sillä karttakuvan näyttämisestä huolehtii Google. Ainoat varsinaiset työt karttakuvan osalta ovat erinäisiä säätöjä, joilla saadaan tarvittavat toiminnallisuudet mukaan. Selaimessa toimivaan JavaScript-koo-

diin on tarkoitus tehdä toiminnot alustukseen, paikkatietojen hakemiseen ja listaamiseen, näkymienhallintaa varten sekä optimireitin muodostamiseen.

6.7.1 Karttapalvelun toiminnot

Alustuksessa muodostetaan karttanäkymä ja laitetaan kuntoon oletusasetukset. Täällä luodaan katunäkymän ilmentymä sekä reititykseen ja reitin näyttämiseen käytettävät objektit.

Taulukko 1: Karttapalvelun ohjaamiseen käytettävät komennot.

Funktion nimi ja parametrit	**Käyttötarkoitus**
initializeMap(lat, lng)	Muodostaa kartta- ja katunäkymän. Asettaa nastan oikeaan kohtaan kartalla sekä muodostaa optimireitityksen vaatimat objektit.
setPanorama(pitch, yaw, zoom)	Asettaa katunäkymän kameran koordinaatit.
showStreet(viewType)	Vaihtaa karttakuvan paikalle katunäkymän tai päinvastoin.
calculateRoute()	Reitin muodostaminen ja näyttäminen.
alterRoute(lat, lng)	Kohteiden lisääminen tai poistaminen reitiltä.
updateRouteButton(lat, lng)	Muuttaa karttakuvan yhteydessä olevan reitinmuokkauspainikkeen tilaa.
isOnRoute(lat, lng)	Apufunktio, joka tarkistaa onko kohde jo lisätty reitille.

6.8 Tietokannan rakenne

Tässä käsitellään vain työn kannalta olennaisinta taulua Vuokraforumin tietokannassa, koordinaatit sisältävää taulua nimeltä ad_gmaps_location.

Ruby-koodissa sen käsittely tapahtuu kirjoittamalla taulun nimi ja pisteellä erotettuna halutun kentän nimi. Koordinaatit ovat lat (latitudi) ja lng (longitudi) -nimisissä taulun kentissä. Nämä tiedot tallennetaan, kun ilmoittaja lisää ilmoituksensa järjestelmään.

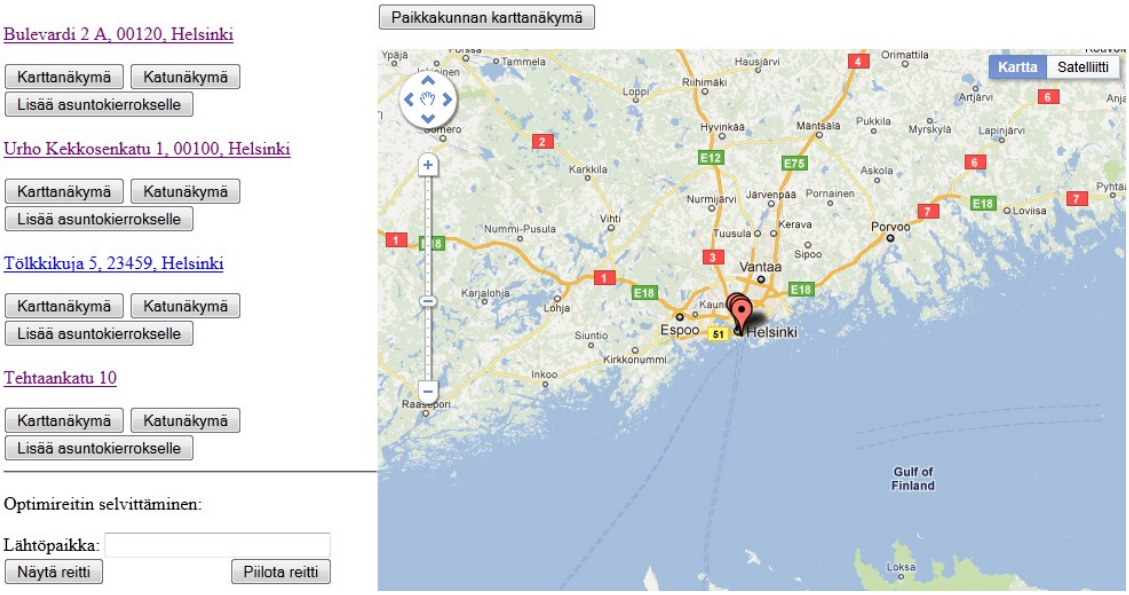
Taulukko 2: Koordinaattien tallennustaulu.

Kentän nimi	Tyyppi	Kuvaus
offer_ad_id	kokonaisluku	tunniste
lat	merkkijono	latitudi
lng	merkkijono	longitudi
yaw	merkkijono	kallistus
pitch	merkkijono	kaltevuus
zoom	merkkijono	tarkennus
created_at	päivämäärä	luontipäivä
updated_at	päivämäärä	muokkausaika

”ad_gmaps_location.lat” ja ”ad_gmaps.location.lng” ovat esimerkkejä Ruby-koodista, jolla saadaan vastaavat arvot noukittua kannasta. Nämä lausekkeet voi kirjoittaa suoraan koodin sekaan. Yaw-, pitch- ja zoom-kenttiä käytetään katunäkymätilassa, jossa ne sisältävät kameran kulman (kallistus vaaka ja -pystysuunnassa sekä zoomaustaso). Tätä toimintoa ei kuitenkaan ole toteutettu nykyisessä versiossa.

6.9 Toteutus

Kun suunnitelmat oli tehty tarpeellisilta osin, alkoi projektin työstäminen tekemällä ensin paikallisella levyasemalla toimiva demo. Tarkoituksena oli testata ja samalla opetella Google Mapsin toimintaa. Demossa on ohjelmoituna muutama osoite keskustan alueelta, joita napsauttamalla kartta siirtyy vastaavaan paikkaan (kuva 7). Lisäksi virheellisen paikkamäärityksen avulla testattiin sitä, kuinka Google Maps suhtautuu virheellisiin viittauksiin.



Kuva 14: Karttapalvelun demo.

Käyttöliittymä pyrittiin tekemään mahdollisimman paljon varsinaisen sivuston näköiseksi siten, että kohteet näkyvät listamuodossa vasemmalla puolella. Kohteiden yhteydessä on painikkeet näkymänvaihtoa ja asuntokierrosta varten.

Demo saatiin tehtyä kuntoon noin kuukauden aikana ja sitä testattiin Mozillan, Internet Explorerin ja Operan kanssa. Demoa ei tarkistettu validaattorin kautta, koska se tuli vain omaan käyttöön ja siinä pelkästään harjoiteltiin Googlen sovellusrajapinnan käyttöä. Tästä opittua tietotaitoa hyödynnettiin, kun sitä integroitiin asuntoilmoitussivustolle.

Toteutus tapahtui virtuaalikoneympäristössä, johon oli asennettu Ubuntu Linux. Kirjoitusohjelmana oli Linux-jakelun mukana tullut tekstieditori. Testaus tapahtui käynnistämällä Vuokraforumin mukana tuleva WeBrick-palvelin. Tämä sisältyy kaikkiin Ruby On Rails -rajapintaa käyttäviin sovelluksiin.

6.10 Lopullinen versio

Eteneminen oli tasaista demon valmistumiseen saakka, jonka jälkeen oli vuorossa varsinaisen karttasovelluksen tekeminen ja testaaminen. Tässä vaiheessa meni paljon aikaa erinäisten syiden vuoksi, joista yksi oli se, ettei Ruby toiminut Windows-virtuaalikoneessa, minkä takia piti asentaa Linux. Vasta sitten asiakkaan toimittama versio Vuok-

raforum-sivustosta saatiin toimimaan, minkä jälkeen varsinaisen karttasovelluksen tekeminen saattoi alkaa toden teolla.

Lopulliseen versioon tulivat lähes kaikki tarvittavat ominaisuudet. Valmiista sovelluksesta jäi puuttumaan muutama vähemmän tärkeä ominaisuus, kuten katunäkymän kameran Z-sijainnin tallentaminen muistiin. Muuten projekti tuli suurelta osin valmiiksi koodin osalta. Vaihtoehtoinen listamuoto jäi ajanpuutteen takia myös pois sovelluksesta.

Lopullinen versio ei kuitenkaan koskaan ehtinyt julkaisuun asti, sillä Vuokraforumia ylläpitävä Agilo Partners meni selvitystilaan.

7 Johtopäätökset

Projekti onnistui lähes odotusten mukaisesti lukuun ottamatta joitakin pieniä, edellisessä luvussa kuvattuja puutteita.

Työn aikana tutustuin (pintapuolisesti) Ruby-kieleen sekä karttapalveluiden toimintaan ja teoriaan. Mitään varsinaista yllättävää ei selvinnyt, mutta karttapalveluiden toiminnasta, Google Mapsista sekä reitin muodostamisesta sain uusia tietoja. Myös Ruby-kielestä oppi ainakin alkeita.

Työn tavoitteina olivat karttapalveluiden kartoittaminen sekä yleisellä että teknisellä tasolla ja ohjelmistoprojektin toteuttaminen Vuokraforum-sivustolle. Tutustuminen karttapalveluihin ja niiden toimintaan sekä karttapalvelun integrointi vuokraussovellukseen onnistui hyvin. Jälkimmäisessä jäi tosin toteuttamatta osa tarvittavista toiminnoista, kuten vaihtoehtoinen listamalli. Sovellus kuitenkin toimi odotetulla tavalla.

8 Yhteenveto

Tämän opinnäytetyön aiheena on eri tietolähteitä hyödyntävän karttapalvelun toteuttaminen asunnonvuokrausilmoituksia välittävälle sivustolle nimeltä Vuokraforum.

Palvelu näyttää asunnot kartalla sekä haluttaessa matkareitin. Työssä perehdytään palvelun rakentamisen lisäksi karttapalveluissa käytettäviin tekniikoihin ja vaihtoehtoisiin toteutustapoihin. Lisäksi luodaan katsaus jo toteutettuihin palveluihin.

Osaa karttapalveluista voi käyttää omilla verkkosivuilla, osa on vain loppukäyttäjille. Karttapalvelut mahdollistavat monia sellaisia asioita, jotka eivät ole mahdollisia perinteisillä, paperisilla kartoilla. Sitä voi lähentää (tarkentaa) tai loitontaa, vierittää eri suuntiin tai joissain tapauksissa kallistaa. Karttapalveluiden tarjoamista kartastoista on saatavana perinteisen viivakartan lisäksi satelliitti / ilmakuvaversio. Joissain tapauksissa käyttäjä voi sukeltaa kolmiulotteisesti mallinnettuun näkymään tai katsoa määrätyssä kohdassa valokuvaa, joka on otettu kadulta.

Oikotie, YTV:n reittiopas sekä Eniro edustavat suomalaisia, loppukäyttäjille suunnattuja palveluita. Ulkomaisista tunnetuimmat lienevät Google Maps ja Bing Maps. Google Mapsin toiminnallisuuden voi helposti upottaa omille sivuille.

Paikkatieto ilmoitetaan selväkielisenä osoitteena, jonka palvelu muuntaa satelliittikoordinaateiksi. Tätä kutsutaan geokoodaukseksi ja sitä varten pitää palvelulla olla tiedossaan sekä osoite että sen koordinaatit. Kartalla kohteet esitetään viivojen ja muiden geometristen symboleiden avulla.

Varsinaiseen reitin muodostamiseen käytetään ns. reitinhakualgoritmeja, joista tunnetuimmat ovat Dijkstran algoritmi sekä A* (a-tähti). Jälkimmäinen on oikeastaan edellisen paranneltu versio. Näissä pohjimmainen idea on edetä lähtöpisteestä (aloitussolmu) loppupisteeseen edeten solmu kerrallaan merkiten samalla reitin hyvyys.

Ohjelmistoprojektin tekeminen alkoi suunnitteluvaiheella ja tutustumalla Google Maps -palveluun. Näiden tietojen pohjalta laadittiin demo, eräänlainen koeversio JavaScriptillä, jonka jälkeen tehtiin varsinainen toteutus JavaScriptiä ja Rubya käyttäen. Sovellus pohjautuu Ruby on Rails -sovelluskehykseen.

Sovellus pohjautuu MVC-suunnittelumalliin, joten sen toimintalogiikka ja tulosten näyttäminen hoidetaan omissa rutiineissaan. Sovelluksen luonteesta johtuen se toimii sekä palvelimella että selainpäässä. Ohjauskomentoja on vain 7, joilla ohjataan erityisesti reitinmuodostamista.

Lähteet

Addison Wesley Longman, Inc. 2001. [verkkodokumentti]. The Pragmatic Programmer's Guide. http://www.ruby-doc.org/docs/ProgrammingRuby/.

Fonecta. 2012. http://www.fonecta.fi/ohje, (3/2/2013).

Google. 2010. [verkkodokumentti]. Google Maps API Family. http://code.google.com/intl/en/apis/maps/, (09/09/2010).

Gosnell, Denise M. 2005. Professional Web APIs : Google, eBay, Amazon.com, MapPoint, FedEx. John Wiley & Sons, Incorporated.

Helsingin Seudun liikenne. 2013. http://www.reittiopas.fi/fi/instructions/, (3/2/2013).

Kokkarinen, Ilkka & Ala-Mutka, Kirsti. 2002. Tietorakenteet ja algoritmit. Talentum Media Oy

Kotilainen, Jussi & Koponen, Mika. 2011 [verkkodokumentti]. Dijkstran algoritmi. http://www.joensuu.fi/matematiikka/kurssit/MatematiikanMestariluokka/VerkotJaRelaatiot/slideshows/painotetut_verkot_dijkstran_algoritmi/index.html, (24/11/2011).

Mannings, Robin. 2008. Ubiquitous Positioning. Artech House.

Microsoft. 2012. http://www.microsoft.com/mappoint/, (3/2/2013).

Oikotie. 2013. http://www.oikotie.fi/, (3/2/2013).

Rantala, Ari. 2005. Web-ohjelmointi. Docendo Finland Oy.

Sagell, Mark. 2008. Ruby documentation.

Sarikaya, Behcet. 2002. Geographic Location in the Internet. Kluwer Academic Publishers.

A*-reitinhaku. http://www.cs.helsinki.fi/u/jltsiren/tiralab/reitinhaku.html, (13/01/2007).

Tervakari, A-M., Silius, K. & Koro, J. 2005 – 2011. Ohjelmistotuotannon mallit. http://hlab.ee.tut.fi/hmopetus/vpsist-oppimateriaali/4-menetelmia-ja-malleja/4-3-suunnittelumalleja/4-3-1-ohjelmistotuotannon-malli, (17/12/2008).

Vuokraforum. 2011. (palvelu ei enää käytössä). http://www.vuokraforum.fi, (5/2/2013). Agilo Partners Oy.

Karttapalvelun JavaScript-lähdekoodi

```
<script     type="text/javascript"     src="http://maps.google.com/maps/api/js?
sensor=false"></script>
<script type="text/javascript">
var gm = google.maps, panorama;
var mapObject;
var geocoder;           var ds, dr;
function showStreet(viewType) {
            panorama.setVisible(viewType);
            // Reset waypoints - gumlike set-up to 'fix' the bug
            document.cookie = "waypoints=";
}
function setPanorama(pitch, yaw, zoom) {
            panorama.setPov({pitch: pitch, heading: yaw, zoom: zoom});
}
function initializeMap(lat, lng) {
            var panoramaOptions = {
                        position: new gm.LatLng(lat, lng),
                        visible: false
            };
panorama = new gm.StreetViewPanorama(document.getElementById("mapsearch"), pa-
noramaOptions);
            var mapOptions = {
                        zoom: 8,
                        center: new gm.LatLng(lat, lng),
                        mapTypeId: gm.MapTypeId.ROADMAP,
                        streetView: panorama,
                        mapTypeControl: true
            };
            mapObject = new gm.Map(document.getElementById("mapsearch"), ma-
pOptions);
            marker = new gm.Marker({position: new gm.LatLng(lat, lng), map:
mapObject, visible: true});
            geocoder = new gm.Geocoder();
            ds = new gm.DirectionsService();
            dr = new gm.DirectionsRenderer();
}
function isOnRoute(lat, lng) {
            var coordinates = lat + "," + lng;
            var dc = document.cookie;
            waypoints = dc.substr(dc.indexOf('waypoints=') + 10);
if (waypoints.indexOf(coordinates) != -1) return true; else return false;
}
function updateRouteButton(lat, lng) {
            if (isOnRoute(lat, lng))
document.getElementById("alterButton").value="Poista asuntokierrokselta";
            else
document.getElementById("alterButton").value="Lisää asuntokierrokselle";
}
function alterRoute(lat, lng) {
            var coordinates = lat + "," + lng;
            var dc = document.cookie;
            waypoints = dc.substr(dc.indexOf('waypoints=') + 10);
            // Alter waypoints
            if (waypoints.indexOf(coordinates) != -1) {
            waypoints = waypoints.replace(new RegExp(",*" + coordinates), "");
                        waypoints = waypoints.replace(new RegExp("^,"), "");
            }
            else {
            waypoints += "," + coordinates;
            if (waypoints[0] == ',') waypoints = waypoints.substr(1);
```

```
            }
            document.cookie = "waypoints=" + waypoints;
            updateRouteButton(lat, lng);
}
function calculateRoute() {
            var address = document.getElementById('origin').value;
            var dc = document.cookie;
var waypointList = dc.substr(dc.indexOf('waypoints=') + 10).split(',');
            var secCounter = 0;
            var waypoints = new Array();
            for (counter = 0; counter < waypointList.length; counter += 2) {
                        waypoints[secCounter] = {
location: new gm.LatLng(waypointList[counter], waypointList[counter + 1])
                        };
                        secCounter++;
            }
            var destination = waypoints.pop().location;
            var dsRequest = {
                        origin: address,
                        destination: destination,
                        travelMode: gm.TravelMode.DRIVING,
                        waypoints: waypoints,
                        optimizeWaypoints: true
            };
if (dr.getMap() == null) directionsShown = false; else directionsShown = true;
            if (!directionsShown) {
                        dr.setMap(mapObject);
    document.getElementById('calculateButton').value = 'Piilota reitti';
            }
            else {
                        dr.setMap(null);
            document.getElementById('calculateButton').value = 'Näytä reitti';
            }           // Calculate route ...
            ds.route(dsRequest, function (dsRoute, dsStatus) {
                        // ... and show it
            if (dsStatus == gm.DirectionsStatus.OK) dr.setDirections(dsRoute);
            });
}
function showTargets(mode) {
document.write(mode + ' | ' + adsit);
for (counter = 0; counter < 3; counter++)
            document.write(adsit[counter]);
}
</script>
```