Datan visualisointi kartalla

HAMK
HÄMEEN AMMATTIKORKEAKOULU

Ammattikorkeakoulun opinnäytetyö

Tietotekniikan koulutusohjelma

Riihimäki, 31.10.2013

Joonas Koski

HAMK
HÄMEEN AMMATTIKORKEAKOULU

TIIVISTELMÄ

Riihimäki
Tietotekniikan koulutusohjelma
Ohjelmistotekniikka

Tekijä Joonas Koski **Vuosi** 2013

Työnnimi Datan visualisointi kartalla

TIIVISTELMÄ

Työn toimeksiantajana toimi Zoined Oy. Keskusteluissa tuli esille aiempi kokemukseni karttarajapinnoista. Aihe kiinnosti sekä itseä, että Zoinedia, joten yhdessä päätettiin kehitellä jokin tarkempi aihe liittyen karttarajapintoihin. Aiheeksi kehittyi lopulta asiakasdatan esittäminen graafisesti kartalla siten, että käytetään eri alueiden värittämistä datan määrää kuvaavilla väreillä.

Opinnäytetyön tavoitteena oli tutkia aluksi karttarajapintoja ja valita yksi, jolla toteutettaisiin demo-sovellus. Aiempaa kokemusta karttarajapinnoista minulta löytyi paikkatietokurssin muodossa, sekä aiemmin tekemästäni sovellusprojektista, jossa käytettiin Google Maps-rajapintaa. Työssä oli tarkoitus myös käydä läpi teoriatasolla demo-sovelluksessa käytettävät tekniikat.

Karttarajapintojen tutkimiseen käytettiin apuna SWOT -analyyseja, joilla pyrittiin esittämään rajapinnan ominaisuudet helposti luettavassa muodossa. Tietoa eri karttarajapinnoista löytyi internetistä eri lähteistä. Apuna käytettiin niin palveluntarjoajien omia sivuja, kuin käyttäjien tekemiä arvosteluja ja testejäkin.

Lopullinen demo-sovellus täytti kaikki sille asetetut tavoitteet. Suomen kartta jaettiin kunta-alueisiin ja kuntia pystyttiin värittämään tietokannasta haetun datan mukaan. Joitakin ongelmia tasojen piirtämiseen liittyvissä tyyleissä jäi kuitenkin ratkaisematta, mutta yleisesti sovellus esittää hyvin, miten alueiden värittäminen kartalle tehdään ja miltä se käytännössä näyttää. Sovelluksessa havaittiin myös pieniä suorituskykyongelmia, kun kaikki kunnat piirretään samanaikaisesti. Ratkaisuna tähän voisi olla piirrettävien kuntien rajoittaminen. Kunnat voitaisiin esimerkiksi piirtää kartalle tarkasteltavan maakunnan mukaan, jolloin sovelluksen suorituskyky paranisi.

Avainsanat Karttarajapinta, Ohjelmointi, Sovelluskehitys, JavaScript

Sivut 51 s. + liitteet 15 s.

HAMK
UNIVERSITY OF APPLIED SCIENCES

Riihimäki
Degree Programme in Information Technology

Author Joonas Koski **Year** 2013

Subject of Bachelor's thesis Visualization of data on a map

ABSTRACT

The subject of the thesis was commissioned by Zoined Oy. While discussing about possible subjects the use of map application programming interfaces came up. The author had some previous experience in programming with Google Maps and learning more about the APIs seemed interesting. Later, the subject evolved to showing customer data graphically over a map. The map would be divided into areas and the areas would be colored with different colors according to the data.

The aim of the thesis was to study different mapping APIs, choose one and create a demo application with it. Also the used programming languages and other technologies should be introduced in the theoretical section.

While studying the APIs, a SWOT analysis was used as a tool to present the features of the API in an easy to read format. Most of the information on APIs was found on the Internet from different sources. Basic information was usually found on the commissioner's own website, but user reviews and different comparisons were also a valuable resource.

The final demo application met all the goals. A map of Finland was divided into municipality areas and different municipalities were colored according to the data retrieved from the database. There were some minor problems with the styling of the drawn municipality layer that did not get solved, but overall the application shows well how to create a colored map with data and what it could look like. There were also some performance issues when drawing all of the municipalities at the same time. As a solution to this, the municipalities could be divided into regions and only the required region would be drawn. In this way the performance of the application would increase.

Keywords Programming, Software development, JavaScript, Mapping API

Pages 51 p. + appendices 15 p.

SANASTO

API	Application Programming Interface(API)	Ohjelmointirajapinta, tarjoaa käyttömahdollisuuden yleisimmille toiminnoille sovellusta ohjelmoitaessa.
Arraylist	Taulukko	Listarakenne, johon voidaan tallentaa väliaikaisesti tietoa sovelluksissa.
Attribuutti	Attribute	Arvo joka määrittää objektin, elementin tai muun kohteen ominaisuuksia.
BSD lisenssi	Berkeley Software Distribution license	Vapaa ohjelmistolisenssi, jota käytetään paljon avoimen lähdekoodin sovelluksissa.
CGI	Common Gateway Interface	Web-ympäristön tekniikka, jonka avulla selain välittää dataa palvelimella suoritettavalle ohjelmalle.
CSS	Cascadian Style Sheet	Verkkosivuilla käytettävä tyyliohje.
Elementti	Element	Esimerkiksi HTML:ssä käytettävä komponentti, jonka sisään sivulla näytettävä sisältö laitetaan.
Funktio	Function	Aliohjelma, ohjelman itsenäinen osa, joka suorittaa jonkin tietyn toiminnon.
GIS	Geographical Information System(GIS)	Paikkatietojärjestelmä, jonka avulla voidaan hallita, analysoida ja esittää paikkatietoa.
GRASS	Geographic Resources Analysis Support System(GRASS)	Ilmainen avoimen lähdekoodin paikkatieto-ohjelma.
GUI	Graphical User Interface(GUI)	Graafinen käyttöliittymä.
HTML	Hypertext Markup Language(HTML)	Kuvauskieli, jolla verkkosivuja tehdään.
Karttarajapinta	Mapping API (Application Programming Interface)	Ohjelmointirajapinta, jolla sovelluksiin voidaan lisätä karttaominaisuuksia.
kirjasto	Library	Kokoelma toimintoja, joita voidaan käyttää sovelluksessa.
KML	Keyhole Markup Language(KML)	XML:n tapainen merkintäkieli, jolla kuvataan paikkatietoa.

Latitudi	Latitude	Leveyspiiri, kuvaa maanpinnan kohdan sijaintia päiväntasaajasta pohjoiseen ja etelään.
Liitännäinen	Plugin	Tietokoneohjelma, joka toimii vuorovaikutuksessa isäntäsovelluksen kanssa ja tarjoaa jonkin uuden toiminnon.
Longitudi	Longitude	Pituuspiiri, yhdessä leveyspiirin kanssa määrittää minkä tahansa paikan sijainnin.
Merkistö	Charset	Määrittelee miten jotkin binääriluvut tulee tulkita johonkin kirjoitusjärjestelmään kuuluviksi merkeiksi.
Olio-ohjelmointi	Object-oriented programming	Ohjelmointitapa, jossa ohjelma koostuu yhteistyössä toimivista olioista, jotka sisältävät tietoa ja toiminnallisuutta.
PHP	Hypertext Processor(PHP)	Ohjelmointikieli, jota käytetään dynaamisten verkkosivujen luonnissa.
Skripti	Script	Komentosarja, jolla voidaan automatisoida tehtäviä, ilman varsinaista ohjelmointikieltä.
SQL	Structured Query Language(SQL)	Kyselykieli, jolla relaatiotietokantoja voidaan hallita.
SWOT	Strengths Weaknesses Opportunities Threats(SWOT)	Nelikenttämenetelmä, jota käytetään esimerkiksi ongelmien tunnistamisessa, arvioinnissa ja kehittämisessä.
www-sovelluspalvelu	Web service	Ohjelmistojärjestelmä, joka mahdollistaa tietokoneiden välisen vuorovaikutuksen tietoverkon yli.
XML	Extensible Markup Language	Merkintäkieli, jolla tiedon merkitys on kuvattavissa tiedon sekaan.

SISÄLLYS

1 JOHDANTO 1

2 KARTAT JA KOORDINAATIT 2

2.1 Kartta- ja geometriadataa Suomesta 3
2.1.1 Itellan perusosoitteisto 3
2.1.2 Maanmittauslaitoksen maastotietokannan tiestö osoitteilla 3
2.1.3 Maanmittauslaitoksen kuntarajat 5
2.2 Ohjelmat 6
2.2.1 OpenJUMP 6
2.2.2 Quantum GIS 6
2.2.3 GDAL - Geospatial Data Abstraction Library 7

3 OHJELMOINTI JA TIETOKANNAT 8

3.1 Yleistä 8
3.1.1 Ohjelmointi ja ohjelmistokehitys 8
3.1.2 Tietokannat 9
3.2 Tekniikat 11
3.2.1 PHP 11
3.2.2 Javascript 13
3.2.3 HTML ja CSS 14
3.2.4 SQL 16
3.2.5 MySQL 17
3.3 Ohjelmointiympäristö/Sovellukset/työkalut 18
3.3.1 Spatialite GUI 18
3.3.2 NetBeans 19

4 TOTEUTUS 20

4.1 Sovelluksen määrittely ja suunnittelu 20
4.2 Maanmittauslaitoksen ja Itellan julkisen datan yhdistäminen 20
4.2.1 Yleistä 20
4.2.2 Toteutus 20
4.2.3 Yhteenveto 24
4.3 Kuntaraja aineiston luominen Maanmittauslaitoksen materiaalista 24
4.3.1 Yleistä 24
4.3.2 Toteutus 25
4.3.3 Yhteenveto 30
4.4 Karttarajapintojen tutkiminen 30
4.4.1 Google Maps 30
4.4.2 Bing Maps 32
4.4.3 MapQuest 33
4.4.4 Leaflet 35
4.4.5 Openlayers 36
4.4.6 Johtopäätös 37
4.5 Sovelluksen ohjelmointi 38
4.5.1 Sovelluksen eteneminen 38
4.5.2 Koodiesimerkit 41

5 LOPPUSANAT 46

LÄHTEET 48

Liite 1 Maanmittauslaitoksen kuntajako – Käsittelymaksut
Liite 2 Sovelluksen lähdekoodi – Osa 1
Liite 3 Sovelluksen lähdekoodi – Osa 2
Liite 4 Sovelluksen lähdekoodi – Osa 3
Liite 5 Sovelluksen lähdekoodi – Osa 4
Liite 6 Sovelluksen lähdekoodi – Osa 5
Liite 7 Sovelluksen lähdekoodi – Osa 6
Liite 8 Sovelluksen lähdekoodi – Osa 7
Liite 9 Sovelluksen lähdekoodi – Osa 8
Liite 10 Sovelluksen lähdekoodi – Osa 9
Liite 11 Sovelluksen lähdekoodi – Osa 10
Liite 12 Sovelluksen lähdekoodi – Osa 11
Liite 13 Sovelluksen lähdekoodi – Osa 12
Liite 14 Sovelluksen lähdekoodi – Osa 13
Liite 15 Sovelluksen lähdekoodi – Osa 14

1 JOHDANTO

Toimeksiantajayritys halusi uuden tavan selata asiakasdataa graafisesti. Ideana oli luoda sovellus, jolla voitaisiin esittää tietokannasta haettua dataa erivärisininä alueina Suomen kartalla. Näin voitaisiin seurata tietylle alueelle keskittyvää myyntiä. Sovellus oli tarkoitus toteuttaa käyttämällä jotakin olemassa olevaa karttarajapintaa.

Aihe rajattiin eri karttarajapintojen tutkimiseen, joista valittaisiin tarkempaan tarkasteluun viisi eri rajapintaa. Näistä viidestä valitusta karttarajapinnasta tehtäisiin yleisen tutkimisen lisäksi SWOT-analyysit. Tämän jälkeen valittaisiin tarpeisiin parhaiten soveltuva karttarajapinta ja toteutettaisiin demo-sovellus havainnollistamaan rajapinnan mahdollisuuksia. Sovelluksesta oli tarkoitus alun perin tehdä yksinkertainen web-sivu, jossa kartta olisi jaettuna postinumeroalueisiin ja kartalla näytettävää dataa voitaisiin muuttaa helposti.

Opinnäytetyön ensimmäisessä luvussa käydään läpi karttoihin liittyvää yleistietoa, sekä Suomesta tarjolla olevaa paikkatietoa. Lopuksi esitellään vielä muutama paikkatiedon käsittelyyn liittyvä sovellus, joita opinnäytetyön aikana käytettiin.

Seuraavassa luvussa käydään läpi ohjelmointiin ja tietokantoihin liittyvää tietoa. Luvussa on esitetty opinnäytetyössä käytetyt tekniikat ja ohjelmointiin liittyvät sovellukset.

Opinnäytetyön toteutus osiossa käydään läpi kaikki sovelluksen tekoon liittyvät asiat, sekä esitellään tarkemmin valitut viisi karttarajapintaa ja tehdyt SWOT-analyysit. Viimeisenä on käyty vielä läpi itse demo-sovelluksen tekovaiheet ja sovelluksen oleellisimmat koodi-osuudet.

Opinnäytetyön lopuksi on vielä pohdittu opinnäytetyön onnistumista ja tavoitteiden täyttymistä.

2 KARTAT JA KOORDINAATIT

Ihmisillä on aikojen alusta asti ollut tarve ymmärtää ja kartoittaa elinympäristöään. Karttoja on käytetty maailman selittämiseen, kauppareittien ja sotaretkien suunnitteluun, sekä navigointiin. Vanhimmat löydetyt kartat ovat tuhansia vuosia vanhoja luolamaalauksia. Vuosien saatossa ihminen on kehittänyt karttoja hiljalleen niiden nykymuotoa kohti.

1960-luvulla sai alkunsa nykyisenkaltainen paikkatietojärjestelmä eli GIS (Geographic Information System). Järjestelmän kehitti kanadalainen Roger Tomlinson ja järjestelmän nimeksi annettiin Canada Geographic Information System eli CGIS. Sitä käytettiin tallentamaan tietoa Kanadan maastosta, maataloudesta, metsistä ja maan käytöstä. Järjestelmä käytti kansallista koordinaattijärjestelmää, joka kattoi koko maanosan. (Wikipedia 2013a.)

Paikkatietoihin liittyvät oleellisesti koordinaatit, joilla tiedon tarkka sijainti voidaan määrittää. Koordinaattijärjestelmiä on useita, esimerkiksi monilla mailla voi olla omat järjestelmänsä, kuten Suomen Kartastokoordinaattijärjestelmä KKJ tai EUREF-FIN –järjestelmä. EUREF-FIN on Euraasian mannerlaattaan kiinnitetyn ETRS89-koordinaattijärjestelmän suomalainen realisaatio, jonka tarkkuutta on kasvatettu mittaamalla noin 450 kiintopistettä Suomessa ja lähialueilla. Maailmalla yleisimmin käytössä oleva koordinaattijärjestelmä on WGS84, johon GPS-järjestelmäkin perustuu. (Wikipedia 2013b.)

Nykyisin karttoja on saatavilla paljon Internetissä. Palveluntarjoajia on useita, joista tunnetuimpia ovat Google ja Microsoftin Bing. Pelkkien karttojen lisäksi yritykset tarjoavat paikanhakua osoitetiedon perusteella, reitin suunnittelua, sekä 360 astetta kattavaa panoraamakuvaa teiden varsilta. Omien palveluiden lisäksi yritykset tarjoavat työkalut kehittäjille omien karttasovelluksien tekemiseen, jolloin karttoja voidaan integroida eri sovelluksiin ja näin karttojen käyttömahdollisuudet parantuvat.

Nämä karttapalvelut perustuvat usein karttojen käyttöön suoraan palveluntarjoajan palvelimelta. Karttatietoa on toki saatavilla myös digitaalisessa muodossa eri lähteistä. Suomen karttatietoa hallinnoi Maanmittauslaitos, jonka kautta on mahdollista hankkia karttatietoa Suomesta eri muodoissa. Karttoja on saatavilla eri kuvatiedostoformaateissa, sekä muokattavissa paikkatietoformaateissa, kuten ESRI Shapefile–pakettina.

ESRI Shapefile on Geospatiaalinen vektoridataa sisältävä tiedostomuoto, jota käytetään yleisesti GIS-sovelluksissa. Vaikka Shapefile:sta puhutaan tiedostomuotona, on se käytännössä joukko tiedostoja, joista pakollisia ovat .shp, .shx ja .dbf -päätteiset tiedostot. Shp -tiedosto sisältää varsinaisen geometriadatan, shx–tiedosto sisältää indeksointi tiedon, jonka avulla voidaan hakea tietoa nopeammin ja dbf-tiedosto sisältää geometriaan liittyvät attribuutit. (Wikipedia 2013c.)

2.1 Kartta- ja geometriadataa Suomesta

2.1.1 Itellan perusosoitteisto

Itella julkaisi tammikuussa 2013 avoimena datana Suomen perusosoitteiston pois lukien Ahvenanmaan. Tietokanta sisältää postinumerot nimitietoineen, katujen nimet sekä kuntatiedot. Kadun jakautuessa useaan postinumeroalueeseen, on käytettävä tietokannasta löytyviä talonumeroita. Näillä tiedoilla voidaan määrittää mihin postinumeroalueeseen kukin tienpätkä kuuluu. Tietokantaa ylläpidetään jatkuvasti ja uudet tiedot päivitetään viikoittain.

"Tiedostot ovat vapaasti ladattavissa palvelukuvauksessa esitetyssä muodossa ja siinä kuvatuissa palvelukanavissa. Tietoja voi luovuttaa edelleen, mutta aineistoja luovutettaessa on huolehdittava siitä, että luovutuksensaajalla on tieto palvelun käyttöehdoista sekä tietojen latauspäivämäärästä". (Itella 2013.)

2.1.2 Maanmittauslaitoksen maastotietokannan tiestö osoitteilla

Maanmittauslaitos julkaisi toukokuussa 2012 saataville ilmaiseksi koko Suomen kattavan maastotietokannan. Maastotietokannasta on saatavilla eri osa-alueita kattavia versioita. Seuraavassa taulukossa on selitetty nämä versiot.

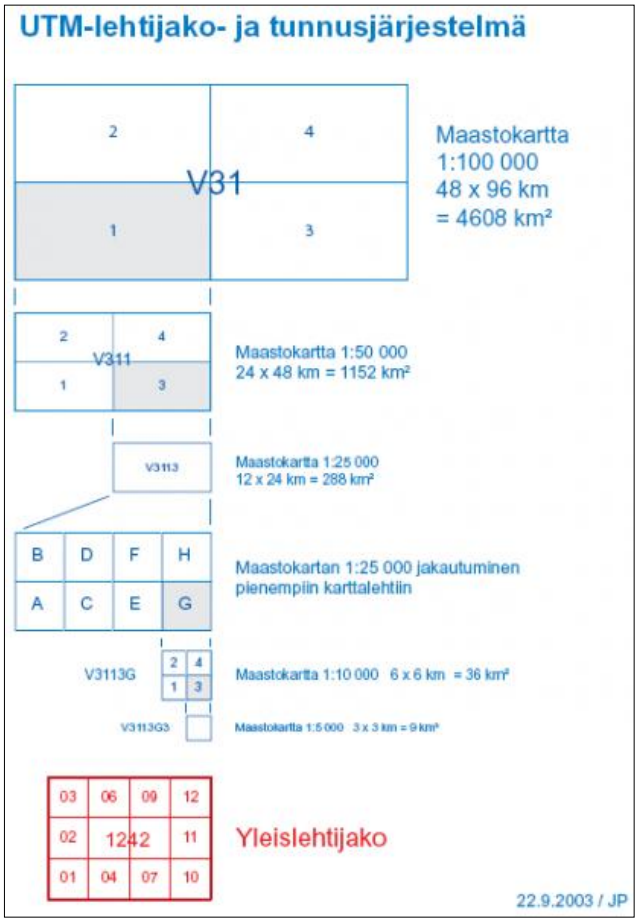
Taulukko 1. Maastotietokantaan kuuluvat aineistot

Aineiston nimi	Selite
Liikenneverkko	Liikenneverkko kattaa mm. kaikki Suomen tiet, kadut ja väylät, sekä rautatiet ja vesiväylät
Tiestö osoitteilla	Tiestö osoitteilla aineisto kattaa mm. kaikki Suomen tiet ja kadut, sekä niiden ominaisuustietona katujen nimet
Rakennukset	Rakennukset tietokanta sisältää mm. tiedot rakennuksista, kuten vesitornit, kellotapulit, lähestymisvalot ja mastot
Johtoyhteydet	Johtoyhteydet aineistoon kuuluu mm. sähkölinjat, muuntajat, suurjännitelinjojen pylväät, sekä merkittävät maakaasu- ja vesiputket
Hallintorajat	Hallintorajoihin kuuluu mm. valtakunnanraja, aluemeren ulkoraja, maakunnan raja, sekä kunnan raja
Kalliot ja kivennäismaat	Aineistoon sisältyvät mm. kalliot, jyrkänteet, kivikot ja sorakuopat
Korkeussuhteet	Mastotietokannan korkeussuhteisiin sisältyvät korkeus- ja syvyyskäyrät
Nimistö	Nimistöön sisältyvät kaikki maasto-, asutus-, ja yksittäisten kohteiden nimet sekä kartografiset tiedot
Pellot	Pellot aineisto sisältää kaikki kartoitushetkellä viljelykäytössä olleet pellot ja niityt
Suot	Soihin sisältyvät suot, soistumat ja eloperäisen aineksen

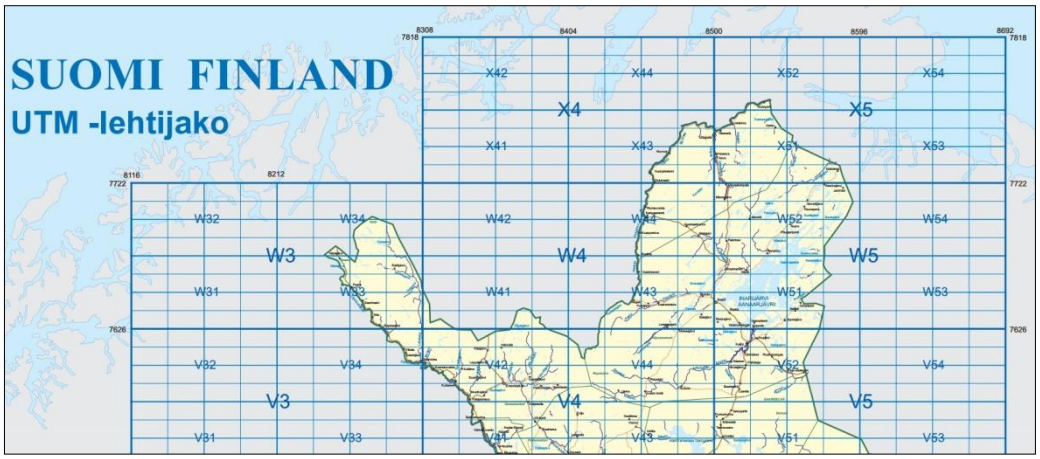
	ottoalueet
Vedet	Vesiin kuuluvat mm. merialueet, järjet, lammet ja joet

Näistä aineistoista opinnäytetyön kannalta oleellisin on ”Tiestö osoitteilla”-tietokanta, joka sisältää Suomen tieosuudet geometriatietoineen.

Maanmittauslaitos käyttää aineistoissaan TM35-lehtijakoa. Suomi on jaettu kuvan 1 mukaisesti alueisiin, jotka on edelleen jaettu pienempiin alueisiin ja niin edelleen. Kuvassa 2 on esitetty pieni pala Maanmittauslaitoksen suorittamasta jaosta. (Maanmittauslaitos1 2013.)



Kuva 1. Maanmittauslaitoksen käyttämä UTM-lehtijako. (Maanmittauslaitos1)



Kuva 2. Esimerkki Maanmittauslaitoksen toteuttamasta lehtijaosta.

Tiestö osoitteilla aineistoa voidaan käyttää erilaisissa logistiikan sovelluksissa tai paikkatietosovelluksissa. Maanmittauslaitoksen ilmaisten aineistojen käyttöehdoissa mainitaan, että aineistoa käytettäessä on palvelun yhteyteen liitettävä maininta alkuperäislähteestä ja aineiston vuosiluvusta. Aineistoa voidaan vapaasti kopioida ja levittää, muokata ja hyödyntää kaupallisesti, yhdistellä muihin tuotteisiin, sekä käyttää osana sovellusta tai palvelua. (Maanmittauslaitos2 2013.)

2.1.3 Maanmittauslaitoksen kuntarajat

Maanmittauslaitos tarjoaa Suomen kuntajakoa kuvaavan aineiston, josta luodaan uusi versio kerran vuodessa. Aineistoa on saatavilla eri mittakaavoilla lähtien 1:10 000 ja päättyen 1:4.5 miljoonaan. Tässä opinnäytetyössä käytetään 1:1 miljoonaan aineistoa. Kuntajako-aineisto on saatavissa vektorimuotoisina karttoina. Aineisto sisältää kuntanumerot, kuntien nimet suomeksi ja ruotsiksi, sekä kuntien rajat ja alueet koordinaatteina. Kuntarajat aineisto toimitetaan koko Suomen kattavana pakettina. (Maanmittauslaitos3 2013.)

Kuntajako aineisto käyttää Maanmittauslaitoksen ilmaisen aineiston käyttöehtoja, joten sitä voidaan käyttää ja muokata vapaasti, kunhan lopullisen palvelun yhteyteen laitetaan maininta alkuperäisestä aineistosta, sekä aineistoversion vuosiluvusta. (Maanmittauslaitos2 2013.)

Materiaali toimitetaan normaalisti Maanmittauslaitoksen ilmaisen aineiston latauspalvelun kautta, jolloin aineistosta tai sen käsittelystä ei peritä maksua. Jos aineisto toimitetaan muulla tapaa, peritään siitä liitteen 1 mukaisesti käsittelymaksu.

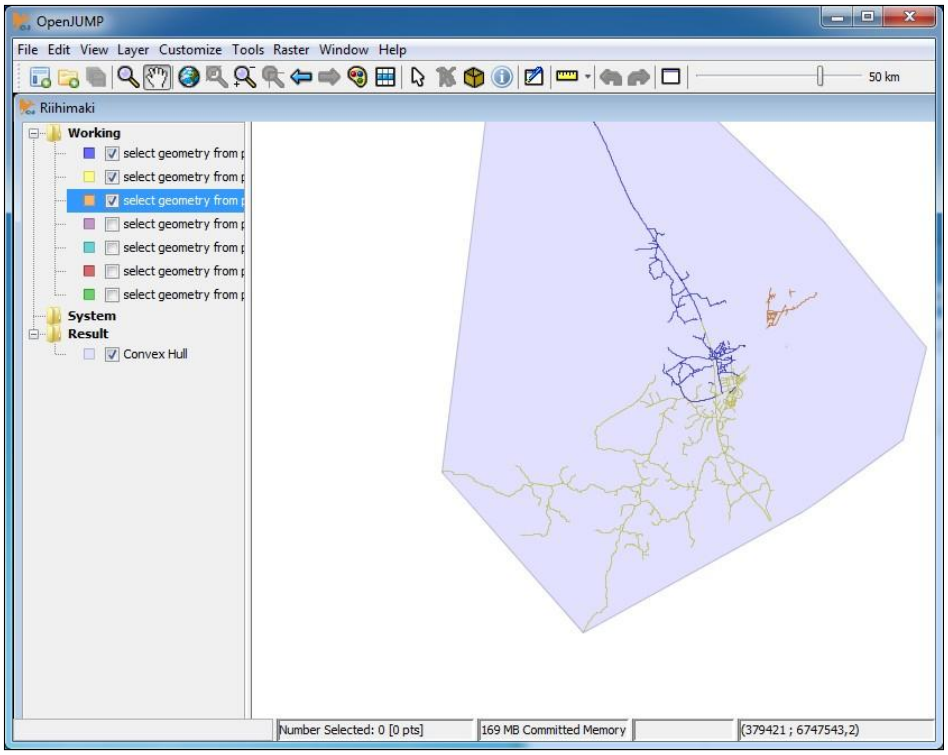
2.2 Ohjelmat

2.2.1 OpenJUMP

OpenJUMP GIS (Geographic Information System) on Javalla kirjoitettu avoimen lähdekoodin sovellus, jolla voidaan avata geometria tiedostoja ja tietokantoja.

OpenJUMP on työkalu geometriaa sisältävän datan työstämiseen. Ohjelma tallentaa väliaikaisesti kaiken käsiteltävän datan tietokoneen muistiin, joten käytettävissä olevan muistin määrä rajoittaa ratkaisevasti käsiteltävän aineiston kokoa. Pienempiin aineistoihin OpenJUMP soveltuu mainiosti.

Ohjelmaan on saatavilla lisäosia, joilla voidaan tehdä SQL-kyselyitä Oracle Spatial-, MySQL- ja SpatiaLite -tietokannoista. SpatiaLite -tietokantojen lukemiseen tarvittava lisäosa on DB Query Plugin. Lisäosan avulla ohjelma osaa piirtää tietokannan geometria tiedot pelkällä SQL -lauseella. Kuvassa 3 on esitetty ohjelman peruskäyttöliittymä. (Latuviitta 2013.)



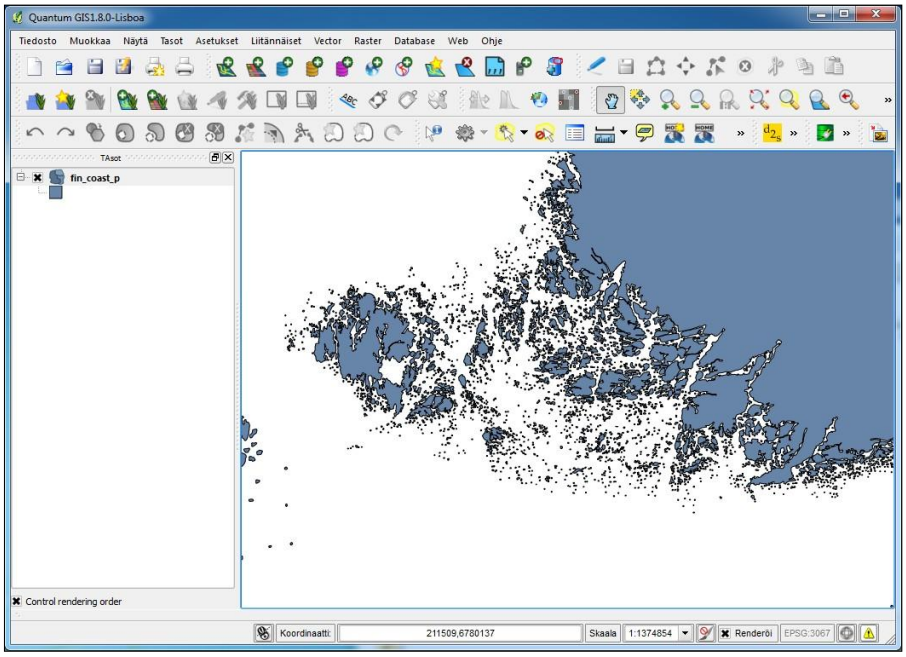
Kuva 3. OpenJump ohjelmalla voidaan avata eri karttatiedostoja.

2.2.2 Quantum GIS

Quantum GIS (Geographic Information System) on tehokas avoimen lähdekoodin sovellus, jolla voidaan avata erilaisia geometriaa sisältäviä tiedostoja.

Ohjelma on lisensoitu GNU GPL v3 (General Public License) alaisuuteen. Quantum GIS on saatavilla sekä Windowsille, Linuxille, että Mac OSX:lle. Ohjelman ensimmäinen Alpha-versio ajoittuu aina heinäkuulle 2002 asti. Versio 1.0.0 julkaistiin tammikuussa 2009 ja tällä hetkellä uusin saatavilla oleva versio on 1.8.0, joka julkaistiin kesäkuussa 2012.

Ohjelmaa käytetään maantieteellisten tietojen lukemiseen ja muokkaamiseen. Ohjelma luo omalla tiedostomuodollaan projektin, johon voidaan tuoda eri lähteistä tietoa. Tuettuja tiedostomuotoja on paljon, näihin kuuluvat muun muassa eri vektoriformaatit, kuten ESRI shapefile, MapInfo, SDTS ja GML, sekä rasteriformaatit, jotka voivat sisältää esimerkiksi korkeuseromalleja, ilmakuvia ja satelliittikuvia. Kuvassa 4 on esitetty ohjelman peruskäyttöliittymä.



Kuva 4. Quantum GIS ohjelmalla voidaan suorittaa monipuolisia toimintoja eri kartta tiedostoille.

2.2.3 GDAL - Geospatial Data Abstraction Library

GDAL, eli Geospatial Data Abstraction Library on kirjasto geospatiaalisille rasteritiedostoille. Ohjelmaa käytetään komentokehotteessa käyttäen tekstimuotoisia käskyjä. Ohjelmaa ylläpitää Open Source Geospatial Foundation ja se on julkaistu ilmaisella lisenssillä. Versiosta 1.9.0 eteenpäin ohjelma tukee yli 120 eri rasteritiedostomuotoa. Opinnäytetyötä ajatellen ohjelman oleellisin toiminto on ogr2ogr, jolla maanmittauslaitoksen tarjoamat rasterikartat voidaan viedä spatialite -tietokantaan.

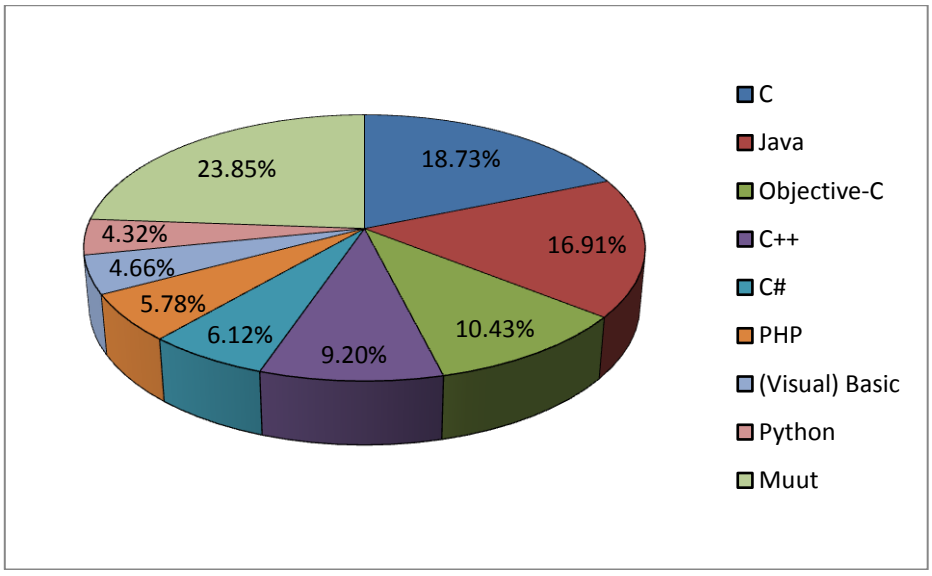
3 OHJELMOINTI JA TIETOKANNAT

3.1 Yleistä

3.1.1 Ohjelmointi ja ohjelmistokehitys

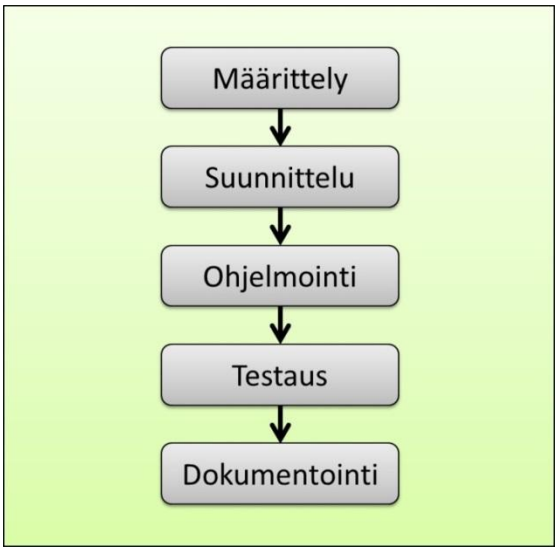
Tietokoneilla pystytään nykyään suorittamaan tehtäviä laidasta laitaan aina videoiden katselusta tekstin käsittelyyn, kuvan muokkaamiseen tai vaikka videopelien pelaamiseen. Tästä huolimatta tietokone itsessään on siinä mielessä tyhmä, että se ei osaa tehdä mitään ilman tarkkaa ohjeistusta. Tässä apuun tulevat tietokoneohjelmat.

Tietokoneohjelmat ovat siis tarkkoja ohjeita, joiden mukaan tietokone osaa suorittaa tarvittavat laskelmat ja toiminnot. Ohjelmia tehdään kirjoittamalla koodia jollakin tietyllä ohjelmointikielellä. Ohjelmointikieliä on useita, joista jokainen on suunniteltu omaan tarkoitukseensa. Tällä hetkellä suosituimpia ohjelmointikieliä ovat C ja Java. Kuvan 5 kaaviossa on esitetty tarkemmin suosituimmat ohjelmointikielet ja niiden osuudet.



Kuva 5. Kaavio eri ohjelmointikielien suosiosta. (TIOBE 2013)

Itse ohjelmointi on kuitenkin vain osa ohjelmistokehitysprosessia. Ohjelman tekoon kuuluu yleensä viisi vaihetta, jotka on esitetty kuvassa 6.



Kuva 6. Sovelluskehitysprojektin eri vaiheet

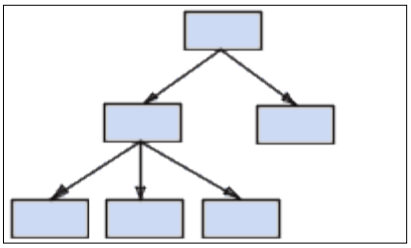
Projekti aloitetaan määrittelyllä, jossa kartoitetaan ongelma ja kerätään tarvittava taustatieto ennen sovelluksen koodaamista. Määrittelyn jälkeen suunnitellaan sovellus mahdollisimman tarkkaan, tähän kuuluu mm. koodin rakenteen, sovelluksen ulkoasun ja mahdollisten tietokantojen suunnittelu. Suunnittelun jälkeen päästään tekemään itse sovellusta. Vaikka sovelluksen aikanakin koodia täytyy testata, niin yleensä projektin loppuvaiheilla suoritetaan vielä laajempi testaus suljetussa ympäristössä, ennen sovelluksen varsinaista julkaisua. Viidentenä vaiheena sovelluskehityksessä tulee dokumentointi. Dokumentoinnilla mahdollistetaan myöhemmin tarpeen vaatiessa sovelluksen päivittäminen ja muutosten teko, sillä tekijä saattaa vaihtua ajan saatossa ja ilman kunnollista dokumentointia koodin ymmärtämiseen palaa turhaa aikaa.

3.1.2 Tietokannat

Tietokannat ovat tietotekniikassa käytettävän tiedon varastoja. Niihin voidaan tallentaa tietoa organisoidusti ja hakea sitä tarvittaessa. Tämä kaikki toimii vielä nopeasti ja tehokkaasti, vaikka dataa olisikin erittäin paljon.
Sovelluksissa tiedon tallentamiseen voidaan käyttää ohjelmointikielen omia tapoja tallentaa tietoa, esimerkkinä ArrayList, joka on listarakenne, johon voidaan tallentaa eri arvoja. Tällainen lista ei kuitenkaan säilytä dataa, jos vaikka sovellus joudutaan sulkemaan. Tietokannat tuovatkin mukanaan avun pitkäaikaiseen tiedon tallentamiseen. Tiedot siis tallennetaan ulkopuoliseen tietokantaan, joka ei varsinaisesti ole kytköksissä ohjelmaan millään tavalla.

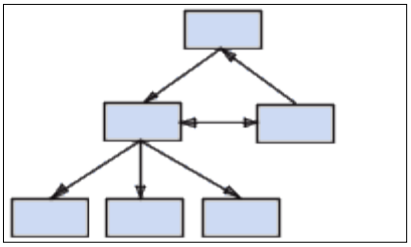
Käytössä olevia tietokantamalleja on neljää eri tyyppiä, joista jokaisella on oma tapansa esittää dataa. Malleja ovat hierarkkinen-, verkosto-, relaatio- ja oliomalli. Näistä eniten suosiota saanut on relaatiomalli, jota jo 90-luvun loppuun mennessä käytettiin kolmanneksessa tietokannoista. (Kioskea.net 2013.)

- **Hierarkkinen malli**: Mallissa data on järjestetty hierarkkisesti ylhäältä alaspäin. Datan välillä navigoidaan käyttämällä suuntimia ja hyppimällä joko ylös- tai alaspäin puussa. (Kioskea.net 2013.)



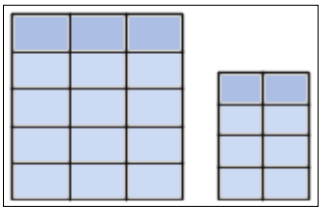
Kuva 7. Hierarkinen tietokantamalli. (Kioskea.net 2013)

- **Verkostomalli**: Hyvin samanlainen hierarkkisen mallin kanssa, mutta datan välillä voidaan myös siirtyä sivuttaissuunnassa. (Kioskea.net 2013.)



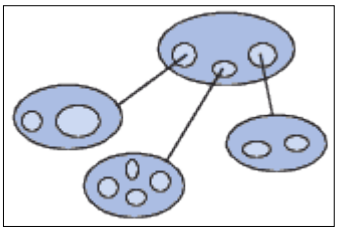
Kuva 8. Verkostomallissa datan välillä voidaan navigoida myös sivuttaissuunnassa. (Kioskea.net 2013.)

- **Relaatiomalli**: Relaatiomallissa data on säilötty tauluihin, joissa on sekä rivejä, että sarakkeita. Taulukoiden välille voidaan tehdä yhteyksiä, joiden avulla eri tauluissa olevaa dataa voidaan yhdistellä.



Kuva 9. Relaatiomallissa data on tallennettu tauluihin. (Kioskea.net 2013)

- **Oliomalli**: Oliotietomallissa oliot on jaettu luokkiin, joihin on liitetty funktioita ja proseduureja. Oliomallia käytetään vain erikoissovelluksissa. (Teuhola 2002.)



Kuva 10. Oliomallin tietokannassa data on tallennettu luokkiin. (kioskea.net 2013)

Pitkän aikaa relaatiotietokannat ovat olleet se ainoa oikea vaihtoehto, sekä web-sovelluksiin, että yrityksen sisäisessä verkossa toimiviin tietokanta-sovelluksiin. Ainoa vakavasti otettava haastaja relaatiomallille on oliomalli. Vaikka oliomalli on ollut olemassa jo 70-luvulta asti, se on vasta viimeisen kymmenen vuoden aikana nostanut suosiotaan. Relaatiomallissa data tallennetaan tauluihin, joissa jokaisella tietueella on taulussa oma avain, jonka avulla tietoa voidaan hakea. Oliomallissa data tallennetaan koodissa luotuun olioon, joka taas lähetetään oliotietokannanhallintatyökalulle, joka osaa prosessoida olion talteen. Relaatiomalli on kuitenkin huomattavasti yleisempi web-sovelluksissa, lähinnä suorituskykynsä takia. Relaatiotietokannan ja sovelluksen välissä käytetään usein Object Relational Mapper (ORM)-kirjastoa, joka muuntaa relaatiotietokannan sisältämän tiedon olioiksi ja toisinpäin. (McLaughlin 2013, 95.)

3.2 Tekniikat

3.2.1 PHP

PHP (Hypertext Preprocessor) on web-ohjelmoinnissa yleisesti käytetty skripti-kieli, jolla voidaan luoda dynaamisia verkkosivuja ja lisätä jo olemassa oleville sivuille uusia toimintoja. PHP on ohjelmointikielenä muuttujien ja funktioiden kanssa työskentelyä kuten JavaScript, mutta myös tulostus-elementtien kanssa työskentelyä kuten HTML (McLaughlin, 2012, 2.)

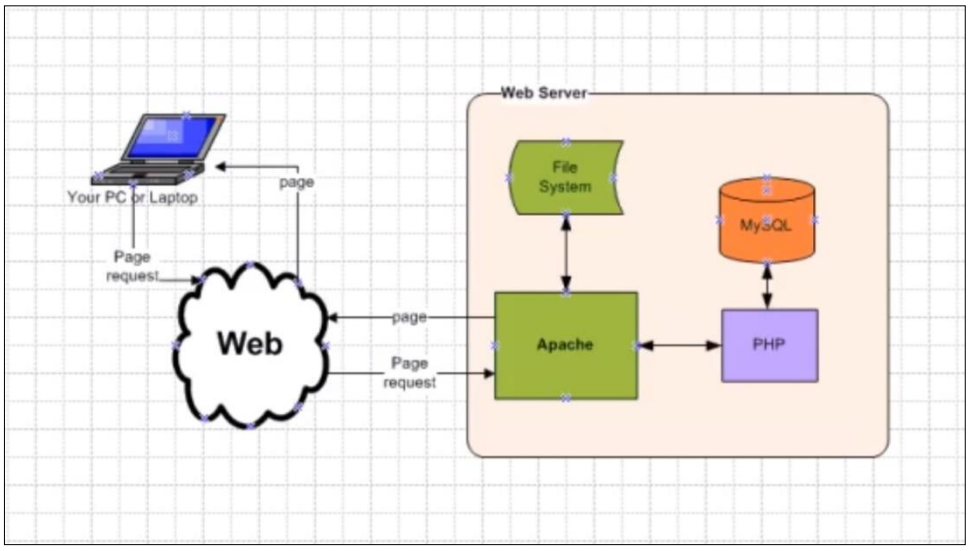
PHP:n kehitti Rasmus Lerdorf vuonna 1994. Alun perin PHP oli vain sarja C-ohjelmointi kielellä kirjoitettuja Common Gateway Interface (CGI) binäärejä, jotka seurasivat hänen ansioluettelonsa kävijämäärää. Nopeasti kuitenkin oli tarvetta suuremmalle määrälle toimintoja, kuten tietokantayhteyksille. (php.net 2013.)

Sittemmin PHP on kehittynyt jo viidenteen versioon. PHP 5 julkaistiin heinäkuussa 2004 ja sisältää nykyään tuen mm. tiedostojen käsittelyyn sekä monenlaisten tietokantojen käsittelyyn, kuten MySQL, SQLite ja Oracle. Lisäksi mukana on myös toimintoja luokkien ja olioiden käsittelyyn. (Wikipedia 2013d.)

PHP eroaa HTML, CSS ja JavaScript -ohjelmoinnista siinä, että toisin kuin muut kielet, PHP:ta ei ajeta selaimessa, vaan PHP-koodi ajetaan kokonaan palvelimella. Palvelimelle on asennettu PHP-tulkki, joka muuttaa

PHP-koodin käyttäjän selaimen ymmärtämään muotoon. (McLaughlin, 2012, 6.)

Käyttäjän avatessa web-sivun, palvelin hakee ensin halutun sivun. Jos sivulla on PHP-koodia, tarvitaan PHP-tulkin apua. PHP-tulkki kääntää PHP-koodin ja suorittaa koodissa olevat toiminnot. Tämän jälkeen PHP-tulkki palauttaa sivun web-selaimen ymmärtämässä muodossa palvelimelle, joka vastaavasti esittää sivun käyttäjälle. Tämä on esitetty kuvassa 11. (webvideobytes 2009.)



Kuva 11. PHP-koodia sisältävän web-sivun näyttö käyttäjälle. (webvideobytes 2009)

PHP-koodissa muuttujat tunnistetaan käyttämällä dollari-merkkiä $ muuttujan nimen alussa. Lisäksi PHP:ssa ketjuttamisessa käytetään pistettä plus-merkin sijaan. Kuvassa 12 esitetyssä esimerkkikoodissa haetaan käyttäjän syöttämät etu- ja sukunimi, sekä sähköposti-osoite, minkä jälkeen nämä tulostetaan näytölle.

Koodin alussa tiedot haetaan edellisen sivun lomakkeesta ja tallennetaan muuttujiin. Muuttujat tulostetaan sen jälkeen käyttämällä echo -komentoa. Etunimi ja sukunimi on ketjutettu samaan tulostus-komentoon käyttämällä piste-merkkiä. Rivinvaihdon jälkeen näytölle tulostetaan vielä $email -muuttujan sisältö.

```
1  <?php
2  $first_name = $_REQUEST['first_name'];
3  $last_name = $_REQUEST['last_name'];
4  $email = $_REQUEST['email'];
5
6  echo $first_name . " " . $last_name;
7  echo "<br>";
8  echo $email;
9  ?>
```

Kuva 12. PHP-koodi esimerkki, jossa tulostetaan käyttäjän edellisellä sivulla syöttämät tiedot.

3.2.2 Javascript

JavaScript on skriptaus -kieli, jolla voidaan lisätä toiminnallisuutta web-sivulle. Sillä voidaan tarkistaa esimerkiksi, että sivulla olevan lomakkeen kenttien sisältö on halutunlaista. JavaScriptillä voidaan myös vaihtaa sivuston tyyliä lennosta tai luoda animoituja nappeja. JavaScriptiä käytetään myös monien erillisten rajapintojen kielenä, kuten esimerkiksi tässä opinnäytetyössä käytetyissä karttarajapinnoissa. (w3.org 2012a.)

JavaScript sai alkunsa vuonna 1995 Netscapella työskennelleen Brendan Eichin toimesta. JavaScriptiä ei suinkaan aina ole kutsuttu JavaScriptiksi. Alkuperäinen nimi kielelle oli Mocha, jonka Netscapen perustaja Marc Andreessen sille valitsi. Myöhemmin syyskuussa 1995 nimi vaihdettiin LiveScriptiksi, mutta saman vuoden joulukuussa Sun Microsystemsin tavaramerkkilisenssin myötä nimi vaihdettiin JavaScriptiksi. Nimestään huolimatta JavaScriptiä ei tule sekoittaa Sun Microsystemsin Java-ohjelmointikieleen. Nimiyhteys on vain markkinointikikka. (w3.org 2012b.)

Lokakuussa 1996 Netscape toimitti JavaScriptin Ecma International standardisointi organisaatiolle arvioitavaksi. Tästä seurannut työ johti standardisoituun versioon ECMAScriptiin. Kesäkuussa 1997 Ecma International julkaisi ensimmäisen version JavaScriptin sisältäneestä spesifikaatiosta. Vuotta myöhemmin julkaistiin parannettu toinen versio. Viimeisin, eli kolmas versio julkaistiin joulukuussa 1999. Kyseinen versio on edelleen käytössä suurimmassa osassa selaimista. (Wikipedia 2013e.)

JavaScriptiä ei tarvitse asentaa tietokoneelle erikseen, sillä jokainen selainasennus sisältää myös JavaScript-tulkin. JavaScript toimii siis automaattisesti käyttäjän selaimessa, jos sivu sisältää JavaScriptiä, osaa selain näyttää sen oikealla tavalla käyttäjälle.

JavaScriptissä muuttujat määritellään käyttämällä var-tunnusta ennen muuttujan nimeä. Muuttujille ei ole myöskään määritelty mitään tyyppiä, joten niihin voidaan tallentaa minkä tyyppisiä arvoja tahansa. Kuvan 13 esimerkkikoodissa on funktio, joka tulostaa käyttäjälle joko hyvää päivää tai hyvää iltaa, riippuen sen hetkisestä kellon ajasta.

```
function myFunction()
{
var x="";
var time=new Date().getHours();
if (time<20)
  {
  x="Good day";
  }
else
  {
  x="Good evening";
  }
document.getElementById("demo").innerHTML=x;
}
```

Kuva 13. JavaScript funktio, jolla tulostetaan käyttäjälle tervehdys kellonajan mukaan.

3.2.3 HTML ja CSS

HTML eli HyperText Markup Language on merkkauskieli, jolla suurin osa verkkosivuista on toteutettu. HTML ei ole varsinaisesti ohjelmointikieli, vaan Internet-sivulla näytettävä teksti tai muu materiaali merkataan sopivan elementin sisään, jotta nettiselain osaa näyttää sisällön oikein. Elementit koostuvat yleensä aloitus- ja lopetusmerkeistä, joiden sisään näytettävä sisältö ja asetukset laitetaan. Nettiselaimella verkkosivua selatessa, käyttäjä ei näe HTML-koodia, vaan selain näyttää automaattisesti koodin lopputuloksen.(Bellis 2013.)

HTML:n kehittämisen aloitti vuonna 1989 Tim Berners-Lee, joka silloin työskenteli CERN:lle. Ajatus lähti liikkeelle, kun hän turhautui ainaiseen uudelleen kirjautumiseen eri tietokoneille eri aineistoa etsiessä. Berners-Lee ajatteli, että on oltava keino hypätä informaatiosta toiseen vaikka ne ovatkin eri tietokoneilla. Tämän kaltainen hyper-text järjestelmä olisi perusta Internetin kielelle, eli HTML:lle. (Landofcode.com 2013.)

HTML sivusto koostuu elementeistä, joiden sisällä on taas uusia elementtejä aina tarpeen mukaan. Esimerkiksi kuvan 14 HTML-dokumentissa on html-elementti, jonka sisällä on sekä head, että body–elementit. Head-elementin sisälle on laitettu sivuston otsikko tieto ja body-elementin sisään itse sivuston sisältö. Jokaisella HTML-elementillä tulisi olla sekä aloitus, että lopetus merkinnät selainvirheiden välttämiseksi.

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4      <title>HTML esimerkki</title>
5      </head>
6      <body>
7          <div id="Otsikko_div"><h1>Otsikko teksti</h1></div>
8          <div id="Sisalto_div"><p>Kappale tekstiä.</p></div>
9      </body>
10 </html>
```

Kuva 14. Yksinkertainen HTML-sivu, jossa on käytetty div-elementtejä sivuston rakenteen jakamiseen.

Nykyään käytössä on, vielä suhteellisen uusi HTML5, joka tuo mukanaan paljon uusia ominaisuuksia. Uuden standardin avulla selaimessa pyöritettävät videot eivät enää vaadi liitännäisiä kuten Flash. Mukana tulee myös liuta JavaScript-toimintoja, joilla saadaan verkkosivuille yhtä monipuolinen interaktiivinen tarjonta, kuin työpöytäsovelluksissakin. (MacDonald 2011, 1.)

HTML:n alkuperäinen tarkoitus oli kuvata vain verkkosivun rakennetta. Tämä ei kuitenkaan riittänyt verkkosivujen kirjoittajille, vaan he halusivat myös mahdollisuuden vaikuttaa sivuston ulkoasuun. Senhetkiset selainvalmistajat vastasivat pyyntöön ja esittelivät HTML-standardiin kuulumat-

tomia elementtejä, joilla voitiin muuttaa esimerkiksi tekstin ulkoasia. Nykyisin tämä sivuston ulkoasun määrittely tehdään erillisessä CSS-tyylitiedostossa, johon HTML-koodissa vain viitataan. (Wikipedia 2013f.)

CSS eli Cascadian Style Sheets kehitettiin helpottamaan verkkosivujen ulkoasun muokkaamista. Erillisen CSS-tiedoston käyttö verkkosivun tyylien hallinnassa helpottaa myös ylläpitäjän työtä, sillä samoja tyyliasetuksia voidaan käyttää kaikilla sivuston sivuilla. Samalla myös varmistetaan, että sivusto näyttää yhtenäiseltä ja vältytään mahdollisilta virheiltä, verrattuna siihen, että kaikki tyylit kirjoitettaisiin HTML-koodin sekaan. (McFarland 2012, 39.)

CSS:n kehitti Bert Bosin ja Håkon Lien vetämä ryhmä World Wide Web Consortiumin työntekijöitä. CSS on ollut jo suunnittelu tasolla lähes yhtä pitkään kuin HTML. Kun HTML julkaistiin, tietotekniikka ei ollut vielä sillä tasolla, että CSS-tyylejä olisi voitu hyödyntää. Vasta kun tietokoneen näytöt sallivat monipuolisemman median näytön pelkän tekstisisällön lisäksi, CSS-standardi voitiin sisällyttää web-ohjelmointiin. Ensimmäinen versio CSS:stä julkaistiin vuonna 1996. (Kantor 2003.)

CSS on suhteellisen helppolukuista. CSS-tiedosto on vain lista sääntöjä joista jokainen sääntö koostuu valitsijasta, sekä yhdestä tai useammasta asetuksesta. Valitsijoita on monen tyyppisiä. Näistä yleisimpiä ovat normaalin HTML-elementin nimen lisäksi id- ja luokka-valitsija. Tietyntyyppisen HTML-elementin tyylien muokkaamisen valitsijana käytetään sen nimeä. Esimerkiksi <h1> -elementin valitsijaksi riittää pelkkä h1. (w3schools.com1 2013.)

Id-valitsijaa käytetään, kun halutaan määrittää tyyliasetukset yhdelle tietylle elementille. Id-valitsijan tunnuksena toimii ”#”-merkki nimen edessä. Valitsija on kokonaisuudessaan tässä tapauksessa #-merkki ja HTML-koodissa elementille määritetty Id-tunnus. Luokka-valitsijaa taas käytetään määrittämään tyylit tietylle ryhmälle elementtejä. Jokaiselle halutulle elementille on annettu luokkatunnus, jota sitten CSS-säännöissä käytetään. Luokka-valitsijan tunnuksena CSS-tyylitiedostossa on ”.”-merkki.

Valitsijan jälkeen itse tyyliasetukset määritetään hakasulkeiden sisään. Kuvassa 15 on esitetty nämä kolme yleisintä tapaa määrittää tyylejä.

```css
h1
{
    color: red;
    font-family: Arial;
}

#id
{
    background-color: white;
    text-align: left;
}

.luokka
{
    text-align: center;
}
```

Kuva 15. CSS-tyyliasetukset, jossa on määritetty omat tyylit otsikolle, id -nimiselle elementille, sekä luokka-nimisen luokkatunnuksen omaaville elementeille.

3.2.4 SQL

SQL eli Structured Query Language on kieli jota käytetään yleisesti tietokantojen hallintaan. Kielellä voidaan hakea tietoa kannasta, lisätä, päivittää tai poistaa tietoa, sekä muokata tietokannan rakennetta. Kieli perustuu eri komentojen käyttöön, joilla kerrotaan mitä tietokannalle on tarkoitus tehdä. Näitä komentoja ovat esimerkiksi ”INSERT”, ”UPDATE” ja ”DELETE”.

SQL sai alkunsa 70-luvun alussa IBM:llä Donald D. Chamberlinen ja Raymon F. Boycen toimesta. Ensimmäistä versiota kutsuttiin nimellä SEQUEL, joka tuli englannin kielen sanoista Structured English Query Language. Tämä versio suunniteltiin muokkaamaan ja hakemaan dataa IBM:n tietokantajärjestelmästä. Lyhenne SEQUEL jouduttiin pian muuttamaan SQL:ään kun ”SEQUEL” oli jo rekisteröity tuotemerkki. (Wikipedia 2013g.)

Tietokantaohjelmalle annettavat SQL-lauseet sisältävät SQL-kielen komennon, joka määrittää mitä ollaan tekemässä ja tämän jälkeen halutut parametrit. Käytettävät komennot voidaan jakaa neljään kategoriaan. Kyselyihin ja datan hakemiseen käytetään SELECT -lauseita. Tietokannan muokkaamiseen käytetään INSERT, UPDATE ja DELETE -komentoja. Tietokannan määrittämiseen käytetään CREATE, ALTER ja DROP -komentoja. SQL-kielessä on myös hallinta käskyjä, joilla voidaan muokata esimerkiksi tietokannan käyttäjien oikeuksia. Näitä komentoja ovat esimerkiksi GRANT ja REVOKE. (wikibooks 2013.)

Kuvan 16 koodiesimerkissä luodaan ensin uusi taulu käyttäen CREATE TABLE-komentoa, minkä jälkeen tauluun lisätään uusi rivi INSERT INTO komennolla. Lopuksi taulusta haetaan kaikki taulun sisältö käyttäen SELECT * komentoa.

```
CREATE TABLE Taulu
(
ID int,
Nimi varchar(55)
);

INSERT INTO Taulu
VALUES ('Jaakko');

SELECT *
FROM Taulu;
```

Kuva 16. SQL komennot taulun luontiin, datan lisäämiseen ja hakemiseen.

3.2.5 MySQL

MySQL on avoimen lähdekoodin relaatiotietokantaohjelmisto, joka on laajalti käytössä web-palveluiden tietokantana. Tyypillisesti MySQL-tietokantoja käytetään sivustoilla, jotka on tehty PHP-, Python- tai Perl-ohjelmointikielillä. Sivut sitten julkaistaan Apache-web-palvelimella, joka on taas asennettu Linux-palvelimelle. Näin saadaan sivut julkaistua kokonaan avoimen lähdekoodin järjestelmälle. (Wikipedia 2013h.)

MySQL-tietokannan kehittivät vuonna 1995 suomalainen Michael Widenius ja ruotsalainen David Axmark. Ensimmäinen valmis versio julkaistiin toukokuussa 1996 ja nimeksi valittiin tuolloin MySQL, joka arvioiden mukaan juontaa juurensa Wideniuksen My-tyttären mukaan. (Moisio 2008.)

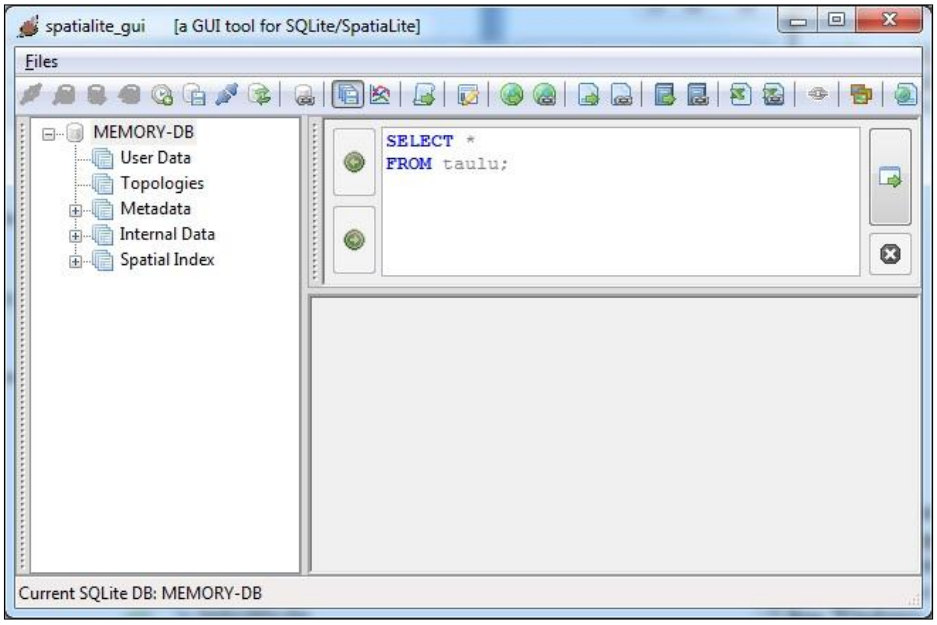
MySQL-tietokantoja hallitaan usein komentorivillä tai jollain muulla tekstipohjaisella asiakasohjelmalla. MySQL tarjoaa myös graafisia käyttöliittymiä tietokantojen hallintaan, kuten MySQLAdministrator ja MySQL-QueryBrowser. Ehkä kuitenkin suosituin vaihtoehto on phpMyAdmin, joka on ilmainen PHP:lla kirjoitettu selaimessa toimiva avoimen lähdekoodin sovellus.

3.3 Ohjelmointiympäristö/Sovellukset/työkalut

3.3.1 Spatialite GUI

Spatialite GUI (Graphical User Interface) on avoimen lähdekoodin työkalu, joka sisältää tuen SpatiaLite-toiminnoille. Ohjelmalla voi selata, muokata ja luoda SQLite -tietokantatiedostoja. Ohjelman ensimmäinen versio 1.0 julkaistiin heinäkuussa 2008. Uusin julkaistu versio on 1.6.0, joka julkaistiin marraskuussa 2012. Ohjelmasta on saatavilla omat versionsa ainakin Windowsille ja Linuxille. Ohjelma on lisensoitu GNU GPL v3 (General Public License) alaisuuteen ja on käyttäjälle täysin ilmainen.

Spatialite on SQLite-tietokantamoottorin päälle rakennettu laajennus, johon on lisätty vektoripohjaisen geotietokannan tuki. Spatialiten avulla tietokantaan voidaan tallentaa geometriatietoa ja tehdä niistä kyselyjä spatiaali -funktioilla. Kuvassa 17 on esitetty ohjelman käyttöliittymä.



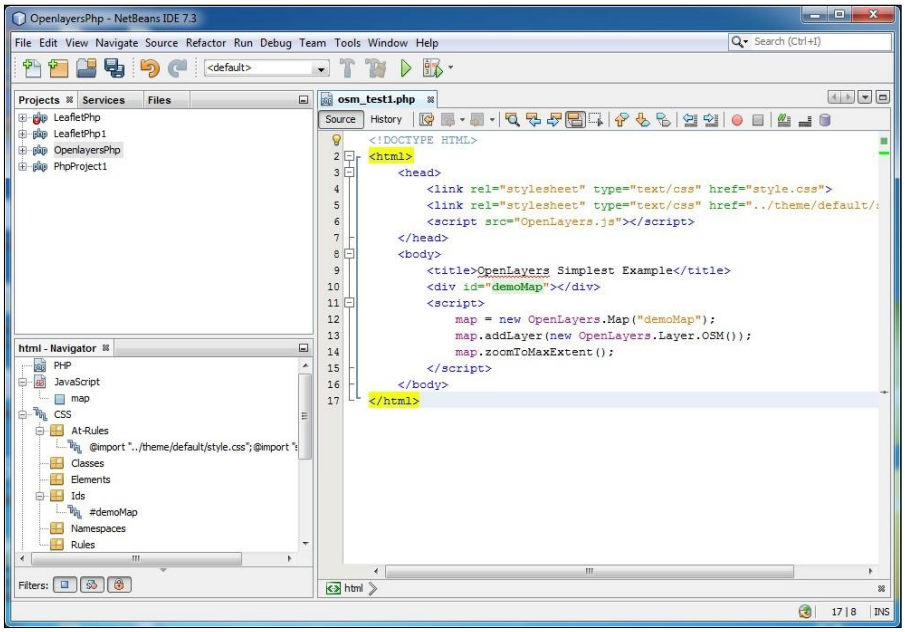
Kuva 17. Spatialite GUI ohjelmalla voidaan selata ja hallita tietokantoja.

3.3.2 NetBeans

NetBeans on integroitu ohjelmointiympäristö eli IDE, englanniksi integrated development environment. Ohjelmaa käytetään pääasiassa Java-sovellusten kehittämiseen, mutta ohjelma tukee myös muitakin kieliä, kuten PHP, C/C++ ja HTML5. Itse sovellus on kirjoitettu Javalla ja sitä voidaan ajaa Windows-, Linux- ja OS X ympäristöissä, kunhan laitteelle on asennettu Java. (Wikipedia 2013i.)

NetBeans ohjelman kehitys alkoi vuonna 1996 Java-opiskelijoiden projektina Kaarlen yliopistossa Prahassa. Seuraavana vuonna Roman Staněk perusti yrityksen projektin ympärille ja kehitti kaupallisia versioita NetBeansista kunnes Sun Microsystems osti yrityksen vuonna 1999. Vuoden 2000 aikana Sun Microsystems julkaisi ohjelman avoimen lähdekoodin projektina. 2010 Sun Microsystemsin ja siten NetBeansin omistus siirtyi Oraclelle. (Wikipedia 2013i.)

Opinnäytetyössä ohjelmaa käytetään demo-sovelluksen tekemiseen. NetBeans auttaa pitämään koodin siistinä automaattisten sisennysten ja värillisen tekstin avulla. Muitakin vaihtoehtoja PHP-sivujen kehittämiseen on, kuten PhpStorm, Eclipse ja Aptana Studio PHP Editor. Valitsin kuitenkin NetBeansin, koska olin käyttänyt ohjelmaa jo aikaisemmin ja se sisälsi kaikki tarvitsemani ominaisuudet, lisäksi se on vielä täysin maksuton. NetBeansin käyttöliittymä on esitetty kuvassa 18.



Kuva 18. NetBeans ohjelman käyttöliittymä.

4 TOTEUTUS

4.1 Sovelluksen määrittely ja suunnittelu

Sovelluksen määrittely- ja suunnitteluvaihe jäi ehkä tarkoituksellakin taka-alalle, koska käytettävä karttarajapinta olisi kuitenkin itselle tuntematon ja halusin saada kokemusta sen käytöstä ensin. Alussa ei myöskään ollut varmuutta siitä, minkälaisen sovelluksen opinnäytetyön yhteydessä ehtisi toteuttaa, joten tarkoituksena oli suunnitella samalla kun sovellusta teki. Alussa tiedettiin kuitenkin, että sovelluksen tulisi toimia avoimen lähdekoodin ympäristöissä, joten kätevintä olisi toteuttaa sivusto PHP:lla. Sovelluksessa käytettäisiin tietokannasta haettua tietoa kartan värittämiseen, ja PHP:lla kaikki tarvittavat tietokantayhteydet olisi helppo tehdä.

Sovelluksen ulkoasun suunnittelu tulisi siis vasta myöhemmässä vaiheessa, kun sovelluksen toiminnallisuutta on ehditty kokeilemaan. Ajatuksena oli kuitenkin laittaa kartta keskeiseen osaan, lisätä muutama nappi tarvittaville toiminnoille, sekä varata tila eri tietojen näyttämistä varten.

4.2 Maanmittauslaitoksen ja Itellan julkisen datan yhdistäminen

4.2.1 Yleistä

Opinnäytetyön alussa tarkoituksena oli tehdä karttasovellus, jossa Suomi olisi jaettu postinumeroalueisiin. Asian tutkimisen jälkeen kävi ilmi, että valmista aineistoa Suomen postinumeroalueista rajojen koordinaatteineen ei ollut olemassa. Internetistä kuitenkin löytyy ohjeita Itellan ja Maanmittauslaitoksen materiaalin yhdistämiseen, jolloin aikaiseksi saataisiin karkeat postinumeroalueet.

4.2.2 Toteutus

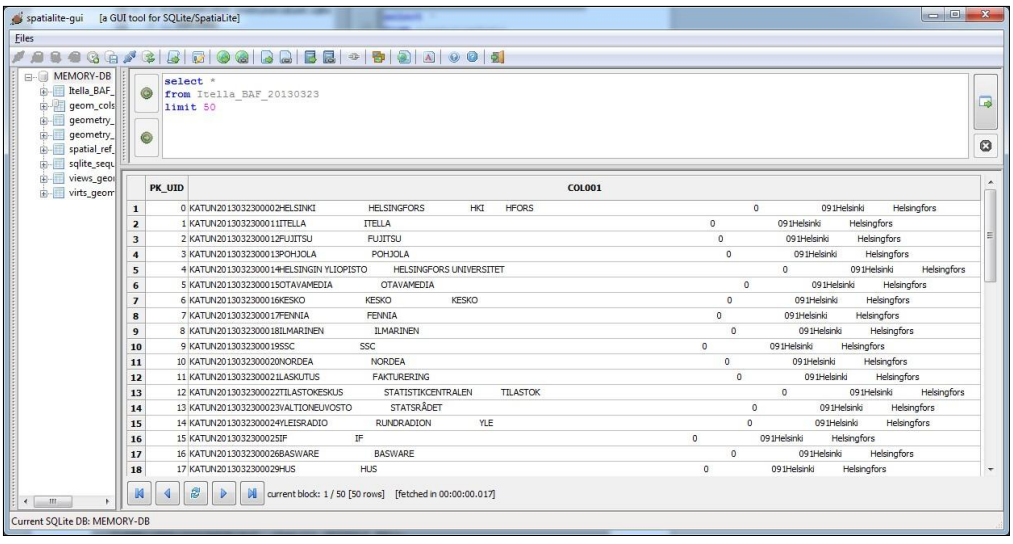
Itellan perusosoitteisto on tekstimuotoisena, joten se täytyy ensiksi muuttaa tietokantamuotoon. Perusosoitteisto -tietokantaan on myös tarkoitus lisätä paikkatietoaineistoa, joten tietokannan tulisi sisältää paikkatietoominaisuuksia. Kuvassa 19 on esitetty esimerkki Itellan tekstimuotoisesta aineistosta.

```
KATUN20130323000002HELSINKI                          HELSINGFORS
HKI          HFORS
0                              091Helsinki             Helsingfors
KATUN20130323000011ITELLA                            ITELLA
0                              091Helsinki             Helsingfors
KATUN20130323000012FUJITSU                           FUJITSU
0                              091Helsinki             Helsingfors
KATUN20130323000013POHJOLA                           POHJOLA
0                              091Helsinki             Helsingfors
KATUN20130323000014HELSINGIN YLIOPISTO               HELSINGFORS
UNIVERSITET
0                              091Helsinki             Helsingfors
KATUN20130323000015OTAVAMEDIA                        OTAVAMEDIA
0                              091Helsinki             Helsingfors
KATUN20130323000016KESKO                             KESKO
KESKO
0                              091Helsinki             Helsingfors
KATUN20130323000017FENNIA                            FENNIA
0                              091Helsinki             Helsingfors
KATUN20130323000018ILMARINEN                         ILMARINEN
0                              091Helsinki             Helsingfors
KATUN20130323000019SSC                               SSC
0                              091Helsinki             Helsingfors
KATUN20130323000020NORDEA                            NORDEA
0                              091Helsinki             Helsingfors
KATUN20130323000021LASKUTUS                          FAKTURERING
0                              091Helsinki             Helsingfors
KATUN20130323000022TILASTOKESKUS
STATISTIKCENTRALEN             TILASTOK
0                              091Helsinki             Helsingfors
```

Kuva 19. Itellan perusosoitteisto on tekstimuotoisena

SpatiaLite GUI -ohjelmalla luodaan ensin uusi tyhjä tietokanta, johon tuodaan Itellan perusosoitteisto. Itellan aineisto ei sisällä otsikkoriviä, joten tietoja tuotaessa pitää muistaa tehdä tarvittavat valinnat, jotta aineisto saadaan tuotua kokonaisuudessaan.

Tietojen tuonnin jälkeen tietokanta näyttää kuvan 20 kaltaiselta, jossa kaikki tiedot on sijoitettu yhteen tekstikenttään. Seuraavaksi kentän tiedot on jaettava Itellan tekstitiedoston rakenteen kuvauksen mukaan omiin sarakkeisiinsa. Tähän tarvittavat SQL-lauseet löytyvät valmiina Jukka Rahkosen luomasta Itellan perusosoitteiston haltuunotto -aineistosta

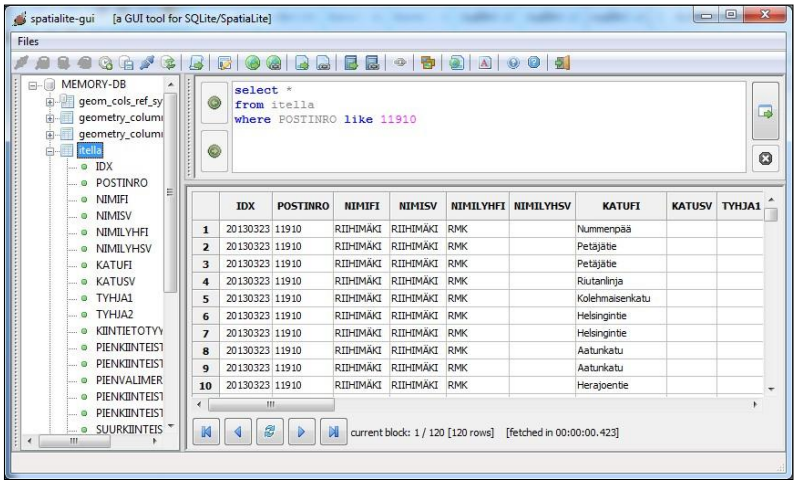


Kuva 20. Itellan perusosoitteisto tuotuna tietokantaan.

Ensiksi luodaan uusi, datalle soveltuva tyhjä tietokantataulu Itellan määrittämän rakenteen mukaan. Seuraavaksi tuodaan perusosoitteiston tuontitaulusta yhtenä jonona oleva tekstisisältö pilkottuna niille sopiviin sarakkei-

siin äsken luotuun Itella-tauluun. Lopuksi luodaan Itella-tauluun vielä tarvittavat indeksit suoritettavien hakujen nopeuttamiseksi.

Kuvassa 21 on esitetty valmis Itella-taulu, johon on tehty testihaku käyttämällä 11910 postinumeroa. Kannasta saadaan haettua kaikki tietylle postinumerolle kuuluvat tieosuudet. Kun aineistoon liitetään paikkatietomateriaalia, voidaan piirtää postinumerolle kuuluvat tieosuudet ja näin saada karkea postinumeroalue.



	IDX	POSTINRO	NIMIFI	NIMISV	NIMILYHFI	NIMILYHSV	KATUFI	KATUSV	TYHJA1
1	20130323	11910	RIIHIMÄKI	RIIHIMÄKI	RMK		Nummenpää		
2	20130323	11910	RIIHIMÄKI	RIIHIMÄKI	RMK		Petäjätie		
3	20130323	11910	RIIHIMÄKI	RIIHIMÄKI	RMK		Petäjätie		
4	20130323	11910	RIIHIMÄKI	RIIHIMÄKI	RMK		Riutanlinja		
5	20130323	11910	RIIHIMÄKI	RIIHIMÄKI	RMK		Kolehmaisenkatu		
6	20130323	11910	RIIHIMÄKI	RIIHIMÄKI	RMK		Helsingintie		
7	20130323	11910	RIIHIMÄKI	RIIHIMÄKI	RMK		Helsingintie		
8	20130323	11910	RIIHIMÄKI	RIIHIMÄKI	RMK		Aatunkatu		
9	20130323	11910	RIIHIMÄKI	RIIHIMÄKI	RMK		Aatunkatu		
10	20130323	11910	RIIHIMÄKI	RIIHIMÄKI	RMK		Herajoentie		

Kuva 21. Itellan aineisto jaettuna omiin sarakkeisiinsa. Tauluun on tehty testihaku käyttämällä Riihimäen 11910 postinumeroa.

Ennen kun postinumeroalueita voidaan piirtää. täytyy Itellan aineiston lisäksi muuntaa Maanmittauslaitoksen "Tiestö osoitteilla" -aineisto sopivaan muotoon.

Maanmittauslaitoksen materiaali on jaettu UTM-lehtijaon mukaisesti osiin, joten muunnettavaa materiaalia on reilusti. "Tiestö osoitteilla" -aineisto toimitetaan shapefile -muodossa ja se täytyy viedä spatialite -tietokantaan. Shapefile:n muuntaminen onnistuu GDAL -apuohjelman avulla käyttämällä "ogr2ogr" -muunnostyökalua. Jotta koko Suomi saadaan katettua, on muunnettavia shapefile -tiedostoja hieman yli sata kappaletta. Kuvassa 22 on esitetty esimerkki ogr2ogr -muunnostyökalun käytöstä.

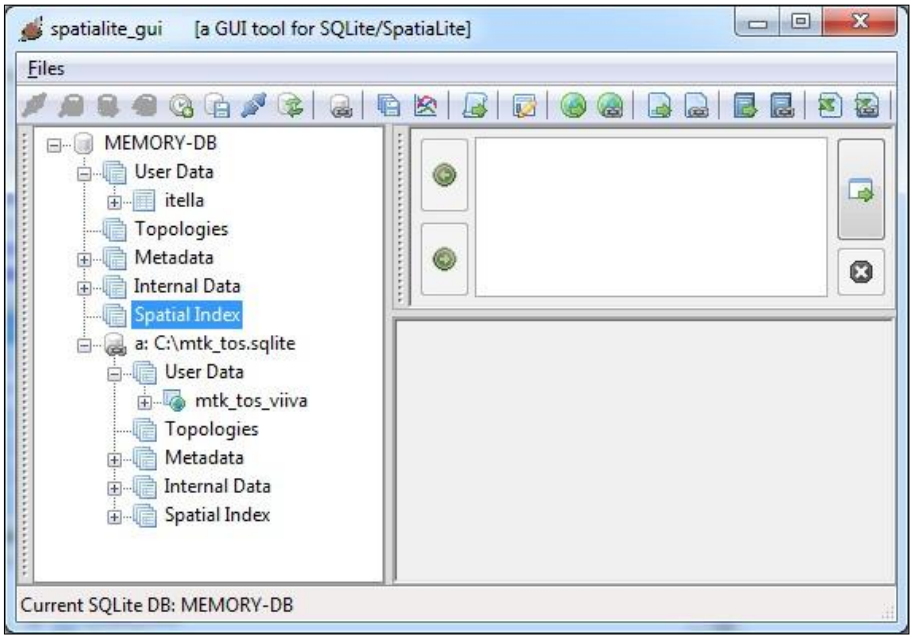
```
SET SHAPE_ENCODING=ISO-8859-1
ogr2ogr -f SQLite -dsco spatialite=yes -dsco init_with_epsg=yes -a_srs epsg:3067
mtk_tos.sqlite  -nlt linestring -nln mtk_tos_viiva -gt 20000  uK34xxxv.shp

ogr2ogr -append -f SQLite  -a_srs epsg:3067 mtk_tos.sqlite  -nlt linestring -nln
mtk_tos_viiva -gt 20000  uK42xxxv.shp

ogr2ogr -append -f SQLite  -a_srs epsg:3067 mtk_tos.sqlite  -nlt linestring -nln
mtk_tos_viiva -gt 20000  uL23xxxv.shp
......
```

Kuva 22. Esimerkkejä tarvittavista komennoista shapefile -tiedoston muuntamiseksi spatialite -tietokantaan käyttäen "ogr2ogr" -muunnostyökalua.

Kun kaikki aineiston shapefile -tiedostot on saatu muunnettua ja vietyä spatialite -tietokantaan, voidaan Itellan perusosoitteisto- ja "Tiestö osoitteilla" -aineistojen yhdistäminen aloittaa.

GDAL -ohjelmalla luotu mtk_tos.sqlite -tietokanta tuodaan aikaisemmin luotuun spatialite -projektiin käyttämällä Spatialite GUI -ohjelmassa "Attach Database" -toimintoa.

Kun tietokanta on liitetty projektiin, voidaan siihen viitata käyttämällä sille liitoksessa syntynyttä etuliitettä a. Kokonaisuudessaan liitetyn tietokannan taulu, joka sisältää maastotietokannan tiet, on nimeltään "a.mtk_tos_viiva". Liitettyjen tietokantojen tauluista voidaan tehdä kyselyitä samaan tapaan kuin projektin omista tietokannoista, mutta tämä ei kuitenkaan ole yhtä tehokasta. Kuvassa 23 on esitetty tietokantaprojektin rakenne ja juuri liitetty maastotietokanta.



Kuva 23. Tietokantaprojektin rakennepuu, kun maastotietokantaan on tehty liitos.

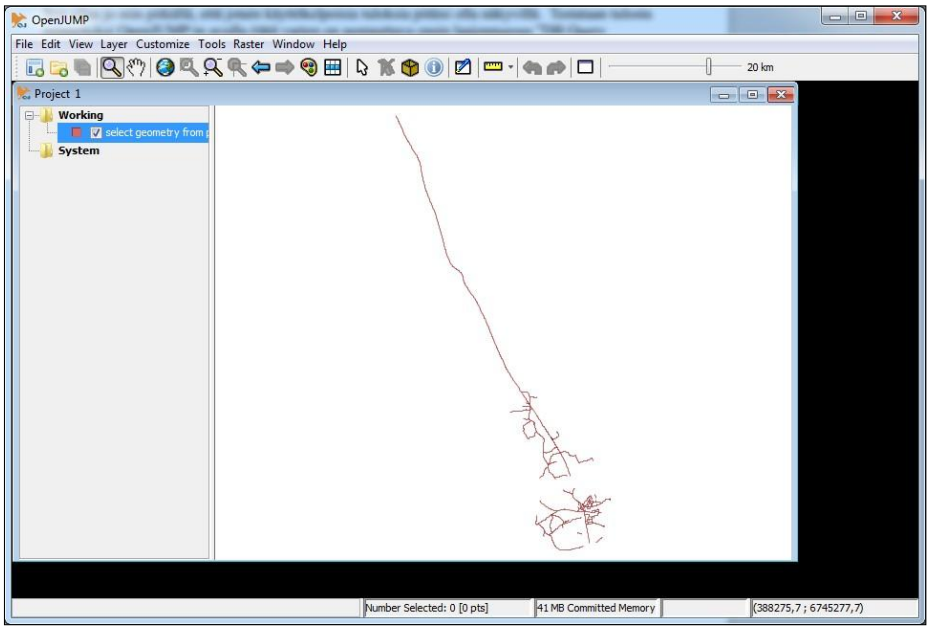
Itellan perusosoitteiston haltuunotto-ohjeessa on esitetty SQL-komennot, joilla tietokanta saadaan tuotua projektiin omaksi uudeksi tietokantatauluksi. Uuden taulun luonnin jälkeen tietokantaliitos voidaan poistaa, koska sitä ei enää tarvita.

Uudessa taulussa ei ole ainuttakaan indeksiä, joten luodaan tarvittavat indeksit ohjeesta löytyvillä SQL-komennoilla. Kun indeksit on luotu, seuraava vaihe onkin yhdistää Itellan ja Maanmittauslaitoksen data keskenään.

Ongelmaksi liitoksessa tulee se, että maastotietokannan teiden osoitenumerot on liitetty tieosuuksiin siten, että niiden yhdistäminen Itellan aineiston osoitenumeroihin ei onnistu helposti, vaan vaatisi paljon työtä ja osaamista. Vaihtoehdoksi jää olla huomioimatta osoitenumeroita lainkaan, jolloin samalla tieosuudella olevia postinumeroalueita ei voida erottaa toisistaan. Tämä tarkoittaa sitä, että joissakin tapauksissa sama tieosuus piirtyy kahdelle, tai useammalle postinumeroalueelle.

Tehdään vielä muutama valmistelu ohjeen mukaan, luomalla kaksi uutta taulua indekseineen. Kun "postitie"-tietokantataulu on luotu, voidaan tietokanta avata OpenJump ohjelmalla.

Openjump-ohjelma tarvitsee "DB Query Plugin" -lisäosan, jotta sillä voidaan suorittaa kyselyitä juuri tehtyyn tietokantaan. Kun tarvittavat lisäosat on asennettu, voidaan suorittaa kysely tietokantaan. Kuvassa 24 on esitetty alue joka on piirretty hakemalla tietokannasta kaikki tieosuudet jotka kuuluvat 11910 postinumerolle.



Kuva 24. Riihimäen postinumero 11910 alueelle kuuluvat tieosuudet

4.2.3 Yhteenveto

Koordinaattipohjaista postinumeroalue tietokantaa ei ollut valmiiksi saatavilla ja lisäksi sen tekeminen alusta lähtien todettiin liian työlääksi. Täten päätettiin perehtyä vaihtoehtoisiin kartanjako menetelmiin. Yhtenä hyvänä vaihtoehtona esille tuli kuntarajat.

4.3 Kuntaraja aineiston luominen Maanmittauslaitoksen materiaalista

4.3.1 Yleistä

Postinumeroalue kokeilun jälkeen päädyttiin tutkimaan sitä, miten kuntarajojen piirto kartalle pystyttäisiin toteuttamaan. Tähän on olemassa lähes valmis materiaali Maanmittauslaitoksella. Vapaasti ladattavissa oleva aineisto sisältää kuitenkin myös rannikkokuntien merialueet, joten aineistosta on jotenkin rajattava merialueet pois, jos alueiden halutaan kattavan vain maa-alueet.

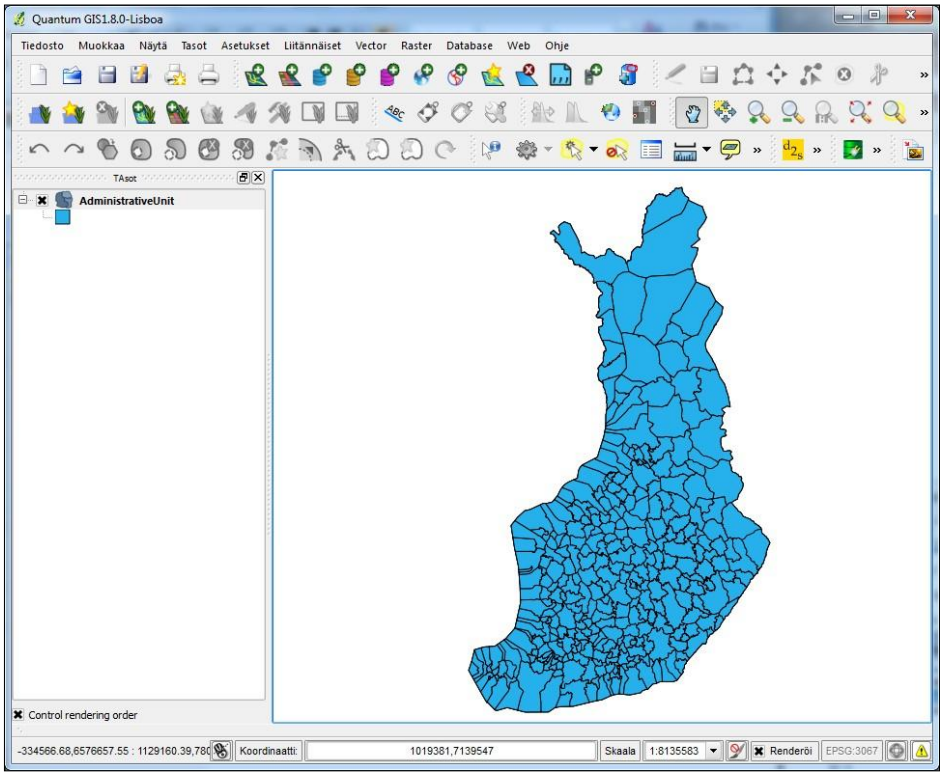
Kuntaraja-aineiston muokkaamiseen löytyy Internetistä kattavat ohjeet, joiden avulla aineisto on helppo muokata. (Lehtomäki 2012.)

Lopuksi aineisto on vielä muutettava muotoon, jota karttarajapinta osaa lukea. Tähän soveltuva tiedostomuoto on KML-tiedosto, johon paikkatietoaineisto on helppo muuntaa. (Sarkola 2012.)

4.3.2 Toteutus

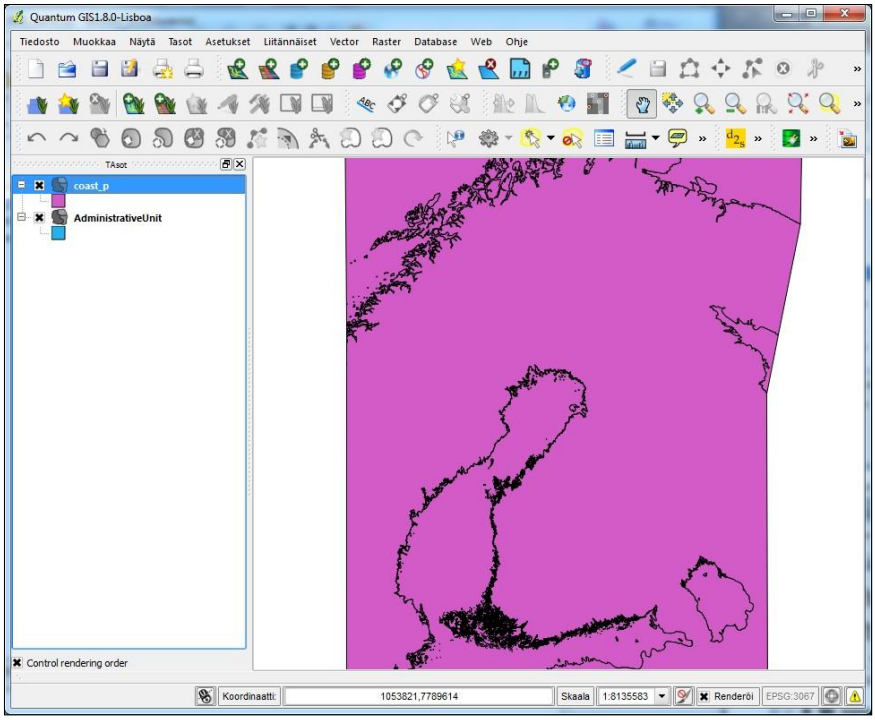
Maanmittauslaitoksen kuntajakoaineisto on ladattavissa MML:n tiedostopankista XML-muodossa. Lisäksi on ladattava aineisto Suomen rantaviivasta, joka löytyy yleiskartta-aineistosta. Saatavilla on kuitenkin valmiiksi esikäsitelty tietokanta, joka on ladattavissa Latuviitta-sivustolta. Molempien aineistojen tulisi olla samalla tarkkuudella. Tässä tapauksessa on käytetty 1:1 miljoonaan aineistoja.

Aineiston muokkaaminen tapahtuu Quantum GIS -ohjelmalla, joka on tarkemmin esitelty karttasovellukset -osiossa. Ensimmäiseksi ohjelmaan tuodaan uudeksi vektori-tasoksi juuri ladattu kuntajakoaineisto, joka on vielä XML-muodossa (Kuva 25).

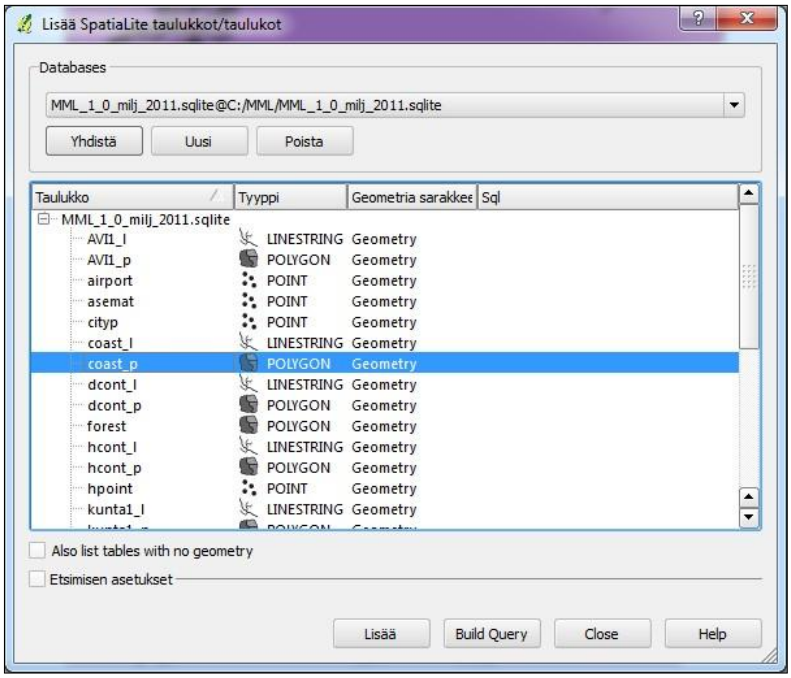


Kuva 25. Suomen kuntarajat tuotuna Maanmittauslaitoksen aineistosta Quantum GIS -ohjelmaan. Mukana ovat myös rannikkokuntien merialueet.

Tämän jälkeen projektiin tuodaan Latuviitta-sivulta ladattu sqlite-tietokanta, joka sisältää Suomen ja lähialueiden rantaviivat (Kuva 26). Latuviitta-sivuston aineistosta halutaan tuoda vain rantaviivat, joten tietokannasta valitaan vain coast_p kentän sisältö (Kuva 27).



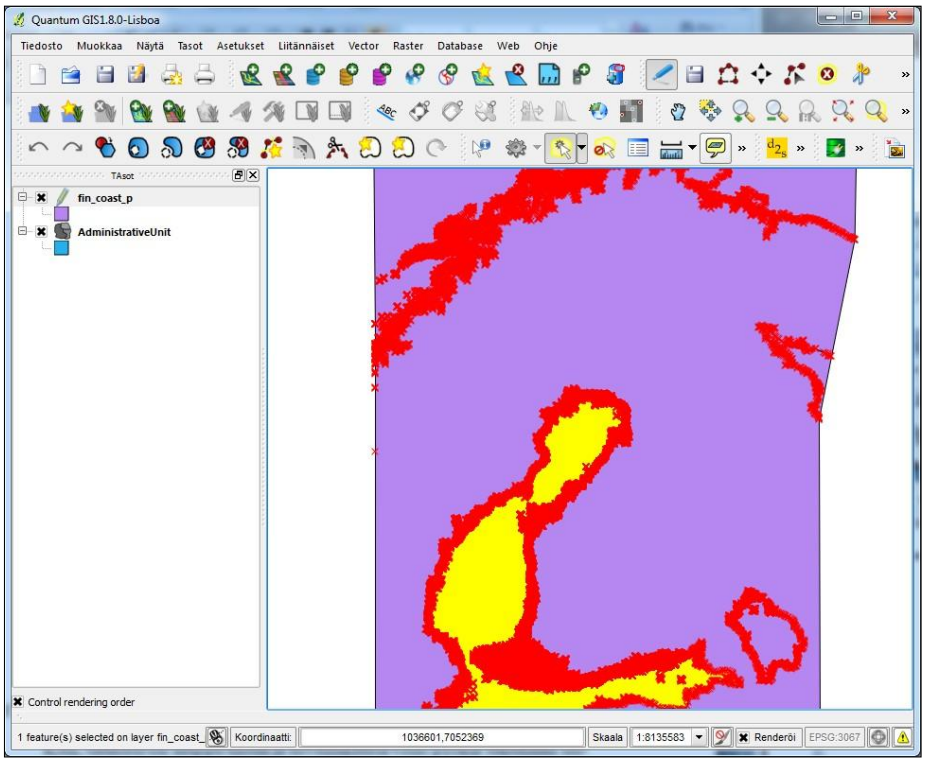
Kuva 26. Suomen ja lähiympäristön rantaviivat tuotuna Quantum GIS -ohjelmaan.



Kuva 27. Rantaviivoja tuotaessa, on tietokannasta valittava vain coast_p sarakkeen sisältö.

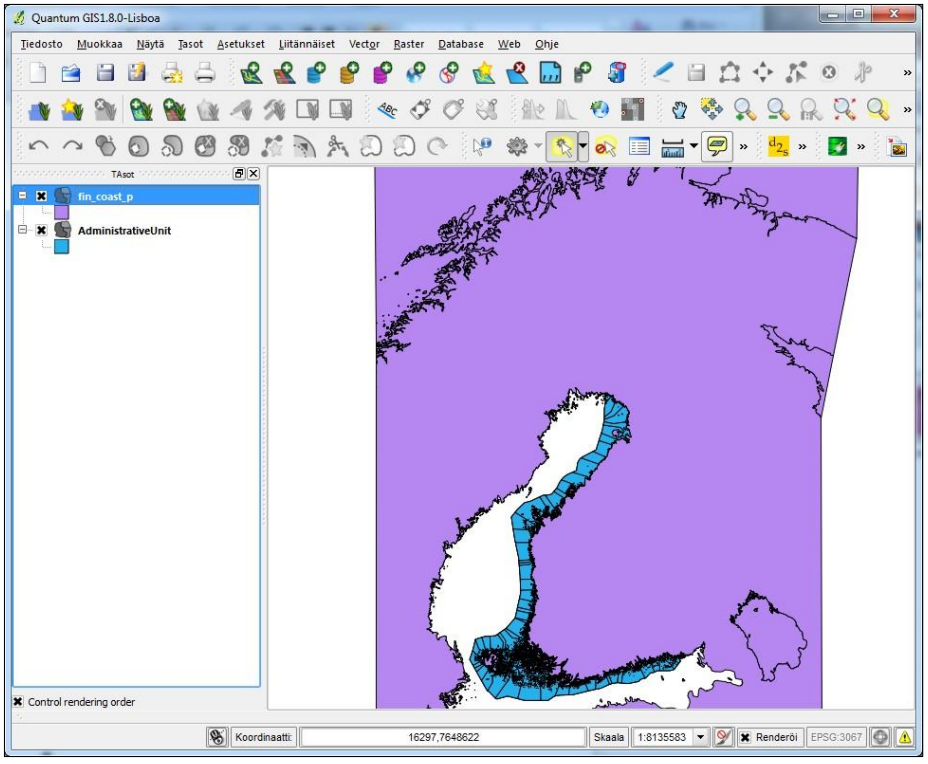
Koska tarkoituksena on muokata rantaviiva vektoria, on suositeltavaa tallentaa taso uutena tiedostona ennen muokkauksia. Kun rantaviiva taso on tallennettu ESRI-shapefile -muodossa, voidaan aiemmin tuotu spatiaalitaso poistaa projektista ja tuoda juuri tallennettu uusi vektoritaso projektiin. Seuraavaksi rantaviiva tasosta poistetaan Suomen merialueet. Tämä

tapahtuu laittamalla tason muokkaustila päälle ja valitsemalla "Select Single Feature"-työkalulla Itämeri (Kuva 28).



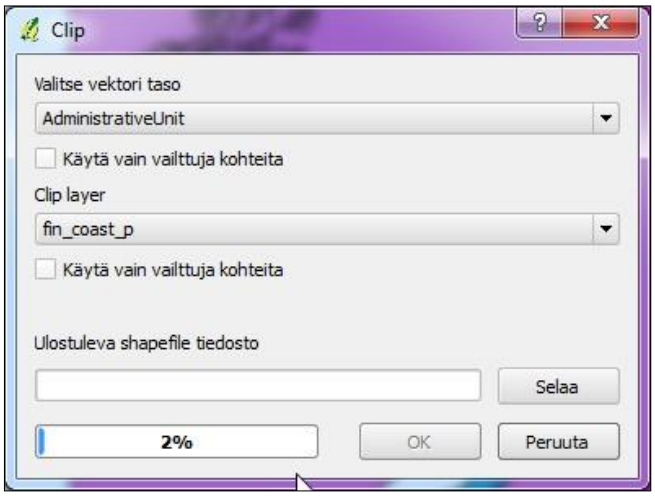
Kuva 28. Itämeri valittuna rantaviiva tasossa.

Valinnan jälkeen valitaan muokkaa -valikosta poista valitut, jonka jälkeen voidaan sulkea muokkaustila. Nyt huomataan, että kuvan 29 mukaisesti osa kunta-alueista näkyy Suomen rantaviiva tason alta.



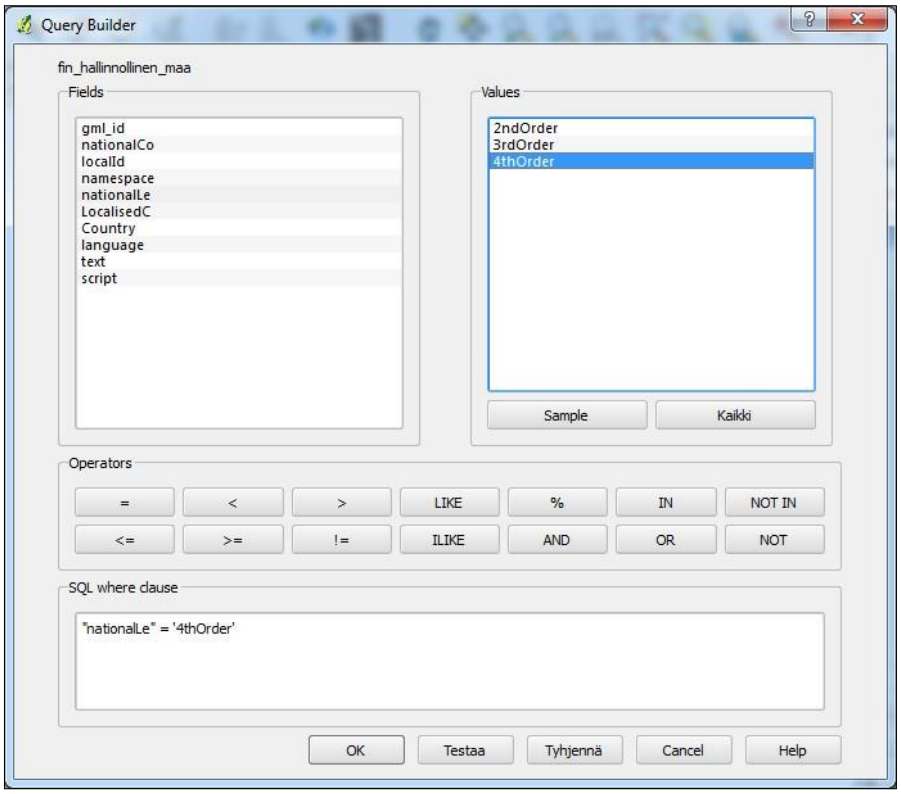
Kuva 29. Kun Itämeri on poistettu rantaviiva tasolta, nähdään kuntaraja aineistosta alueet, jotka pitäisi poistaa.

Kuntien merialueet saadaan leikattua pois käyttämällä Vektori tason Geoprosessointi työkalua nimeltä Clip. Työkaluun tarvitsee valita kaksi tasoa joiden mukaan leikkaus suoritetaan, eli tässä tapauksessa Kuntaraja- ja rantaviiva -tasot (Kuva 30). Työkalu prosessoi dataa jonkin aikaa, itsellä muunnos kesti reilut 18 minuuttia.



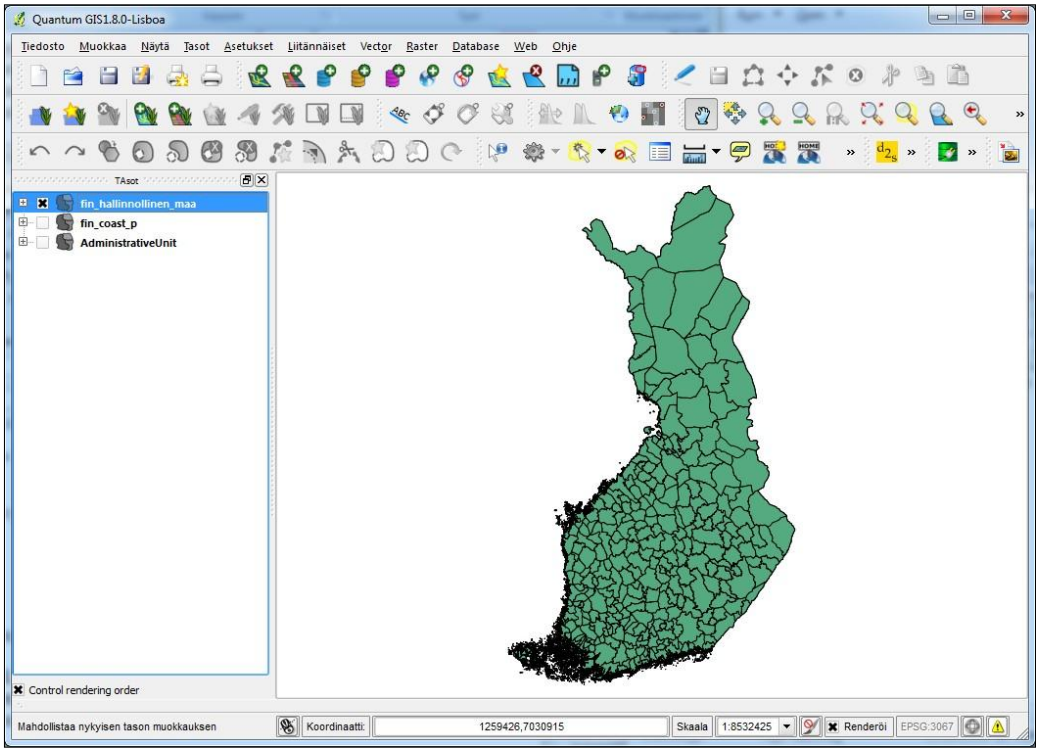
Kuva 30. Quantum GIS -ohjelman Clip -työkalu, jossa leikattavat tasot valittuna ja muunnosprosessi on käynnissä.

Kun prosessi on valmis, nähdään kuntarajat Suomen maa-alueella. Aineistossa on kuitenkin mukana kolmen eri tason mukaisia rajoja, aluevirastokeskuksen, maakuntien ja kuntien mukaisia alueita. Näistä halutaan vain kunnat, joten tasolle tehdään vielä Query Builder -työkalulla SQL-kysely, jolla vain kunnat valitaan (Kuva 31).



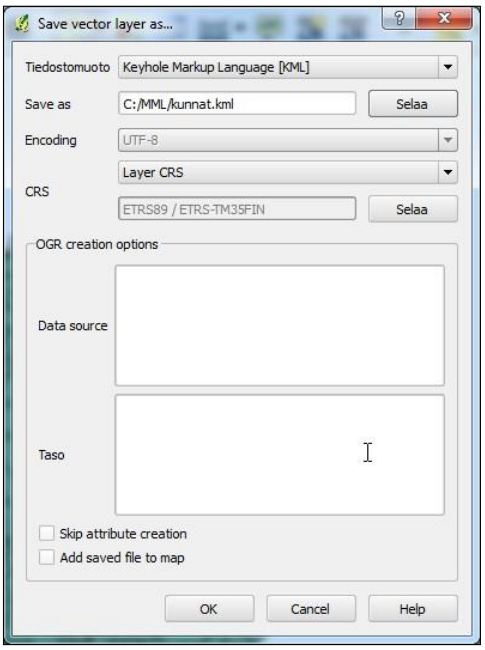
Kuva 31. SQL -valintakysely, jolla saadaan valittua vain kunnat piirrettäväksi.

Kun kysely on suoritettu, on kuntaraja aineistosta valittuna vain kunta-alueet. Valinnan ollessa päällä, tallennetaan taso uudelleen eri nimellä. Nyt käytettävissä on kuvan 32 mukainen aineisto, joka kattaa kaikki Suomen kuntien maa-alueet.



Kuva 32. Suomen kuntarajat pois lukien merialueet.

Data on tosin vielä ESRI-shapefile muodossa, joten ennen kuntarajojen käyttöä web-sovelluksessa, pitää tiedosto tallentaa KML-muotoon. Vielä, kun taso on avoinna Quantum GIS -ohjelmassa, voidaan taso tallentaa KML-muotoon, klikkaamalla tasoa listasta hiiren oikealla napilla ja valitsemalla "save as". Avautuvasta ikkunasta valitaan tiedostomuodoksi KML ja annetaan tiedostolle nimi ja polku (Kuva 33). Koordinaattimuutokset menevät automaattisesti oikein.



Kuva 33. Tason tallentaminen KML-muotoon. Taso tallennetaan automaattisesti WGS84-koordinaattijärjestelmän mukaisesti.

4.3.3 Yhteenveto

KML-tiedosto oli helppo luoda ja suurimman osan ajasta veikin merialueiden leikkausprosessin odottelu. Sovelluksesta voisi pyrkiä tekemään sellaisen, että pelkkää KML-tiedostoa vaihtamalla saataisiin kartalle päivitettyä uudet kuntarajat

4.4 Karttarajapintojen tutkiminen

4.4.1 Google Maps

Google maps on ollut markkinoilla jo pitkään ja lähes kaikki ovat siitä kuulleet. Google Maps oli alun perin kahden tanskalaisen veljeksen tekemä erillinen C++-sovellus, joka käyttäjien oli tarkoitus ladata erikseen omalle koneelle. Pian kuitenkin veljekset olivat yhteydessä Googleen ja ajatus täysin verkkopohjaisesta sovelluksesta vei voiton. Vuoden 2004 lopulla Google osti veljesten yrityksen ja sovelluksesta tuli virallisesti Google Maps. (Wikipedia 2013j.)

Google Mapsin suosio perustuu sen pitkään historiaan ja Googlen hyvään maineeseen. Alun perin Google Maps oli kehittäjille ilmainen, jolloin käyttäjäkunta kasvoi nopeasti. Lisäksi Googlen palveluihin luotetaan niiden korkean laadun ja jatkuvan kehittämisen takia. Palvelun jatkuva kehittäminen tuo mukanaan myös haasteita, sillä Googlen tiheään julkaisemat päivitykset voivat rikkoa jo valmiin ja toimivan karttasovelluksen. Yleensä kuitenkin yleisimpiä perustoimintoja sisältävät sovellukset toimivat päivitystenkin jälkeen normaalisti.

Google Maps rajapinta on toteutettu JavaScriptillä. Siitä oli tarjolla myös Adobe Flash-versio, mutta se on sittemmin lopetettu. Flash-rajapinnalla tehdyt karttasovellukset toimivat vielä syyskuuhun 2014 asti. Google Mapsin valttikorttina on pitkään kehitetyt ominaisuudet, joista varmasti tunnetuin on Street View, jossa käyttäjä voi katutasolla katsella ympärilleen. Lisäksi Googlella on laadukkaat reitinhaku-toiminnot niin, autolla, julkisilla, pyörällä kuin kävellenkin liikkuville.

Rajapinnan ominaisuuksia, heikkouksia ja vahvuuksia on pyritty pohtimaan kuvassa 34 olevassa SWOT -analyysissa.

Vahvuudet	Heikkoudet
• Paljon ominaisuuksia • Kattavat ohjeet API:n käyttöön • API:t myös mobiilialustoille (iOS ja Android) • Laadukkaat karttapohjat • Kehitetään jatkuvasti (tulevaisuus turvattu) • Street View	• Maksullinen, kalliit käyttökustannukset tarpeellisiin ominaisuuksiin nähden
Mahdollisuudet	**Uhat**
• API:a kehitetään jatkuvasti, tulevaisuudessa julkaistaan mahdollisesti uusia hyödyllisiä ominaisuuksia • Taustalla iso yritys, tulevaisuudessa tuen saaminen pitäisi olla helppoa	• Monet avoimen lähdekoodin API:t kehittyvät nopeasti ja tulevat tosissaan haastamaan maksulliset kilpailijansa

Kuva 34. SWOT-analyysi Google Maps rajapinnasta.

Google on määrittänyt käyttörajoituksia rajapinnalle riippuen siitä, onko sovellus kaupallinen vai voittoa tavoittelematon kaikille avoin sovellus. Kaupallisissa sovelluksissa katto on 25 000 latausta päivässä. Kartan lataukseksi lasketaan, kun kartta alustetaan web-sivulla. Vastaavasti kartan liikuttelu, tarkentaminen, tai vaihtaminen satelliitti- ja tiekartan välillä, ei kuluta latauskertoja. Kaupallisista sovelluksista, joissa ilmaantuu ajoittaisia piikkejä latauskerroissa, laskutetaan vasta, kun latauskertojen katto on ylitetty 90 peräkkäisenä päivänä. Avoimilla sovelluksilla ei taas ole kattoa latauskerroille ollenkaan ja lisäksi Google tarjoaa rajapinnan Business-lisenssin ilmaiseksi. Kaupallisten sovellusten yli menevistä latauskerroista laskutetaan kuvan 35 mukaisesti. (Google 2013a.)

Service	Usage limit (per day)	1,000 excess map loads (in U.S. dollars)
JS Maps API v3	25,000	$0.50
JS Maps API v2 (Deprecated)	25,000	$1.00
Static Maps API	25,000	$0.50
Street View Image API	25,000	$0.50

Kuva 35. Kaupallisista Google Maps sovelluksista laskutetaan ylimenevistä latauskerroista. (Google 2013b.)

Vaikka Googlen palvelut ovatkin tunnettuja ja suosittuja, sekä yleensä myös laadukkaita, kannattaa harkita minkälaiseen käyttöön kartat tulevat kaupallisissa sovelluksissa. Jos tarkoituksena on käyttää vain rajapinnan perusominaisuuksia, on tarjolla monta muutakin vaihtoehtoa jotka toimittavat samat ominaisuudet ilman kustannuksia. Jos taas tavoitteena on tehdä monimutkainen karttasovellus, jossa käytettäisiin reititystä ja Googlen Street Viewiä, on Google Maps rajapinta varteenotettava vaihtoehto. Tämän opinnäytetyön aiheena olleen kartan luontiin Google tarjoaa kuitenkin liikaa ominaisuuksia, joten jos käyttömäärät ylittyisivät, maksua kertyisi turhasta.

4.4.2 Bing Maps

Toinen karttarajapintojen suurista nimistä on Microsoftin Bing Maps. Bing Mapsin historia ei ole aivan yhtä pitkä kuin Google Mapsin. Ensimmäinen Microsoftin karttapalvelu julkaistiin kesällä 2005 ja vasta kesäkuussa 2009 tuotteen nimi vaihdettiin Live Search Mapsista Bing Mapsiin. (Wikipedia 2013k.)

Bing Maps sisältää pitkälle samoja ominaisuuksia kuin Google Maps, kuten Streetside, joka on Microsoftin oma versio Googlen suositusta Street View-toiminnosta. Tosin näitä kahta ei voi edes verrata keskenään, sillä Google on ehtinyt jo kuvaamaan tieosuuksia huomattavasti enemmän kuin Microsoft. Esimerkiksi Suomesta ei löydy Streetside-materiaalia lainkaan, kun Googlella on kuvattuna jo pikkuteitäkin. Bing Mapsissa on myös perinteisen satelliittikuvan lisäksi kuvamateriaalia lintuperspektiivistä. Useimmissa paikoissa lintuperspektiivi on kuitenkin toteutettu vain taittamalla satelliittikuvaa, mutta joissakin suuremmissa kaupungeissa ja nähtävyyksissä kuvamateriaali on hankittu ilmasta käsin, joko helikopterilla tai lentokoneella, jolloin kuvanlaatu on huomattavasti tarkempi.

Microsoft tarjoaa karttarajapinnasta JavaScript-, Silverlight- ja työpöytäversiot, joiden lisäksi myös Android, iOS ja Windows Phone -mobiilialustoille on omat kehitystyökalut. Kuvassa 36 tehdyssä SWOT-analyysissa on pyritty kartoittamaan Bing Mapsin ominaisuuksia. (Georelated 2013.)

Kuva 36. SWOT-analyysi Bing Maps karttarajapinnasta.

Bing maps tarjoaa eri lisensointi-vaihtoehtoja eri tarkoituksiin. Jokainen karttasovellus on kuitenkin rekisteröitävä jollekin lisenssille. Tarjolla on 90 päivän kokeilu versio, peruslisenssi ja yrityksille suunnattu Enterprise-lisenssi. Perusavain on tarkoitettu ilmaisille sovelluksille ja opetuskäyttöön, sekä Windows Phone sovelluksille. Peruslisenssilläkin on joitain rajoituksia, kuten Windows Phone-sovellusten 50 000 käyttökertaa vuorokaudessa ja julkisten nettisivujen karttasovellusten käyttökerrat, jotka on rajoitettu 125 000 lataukseen vuodessa. Enterprise-version laskutuksesta joutuu sopimaan erikseen Microsoftin kanssa tapauskohtaisesti. (Bing Maps 2013.)

Microsoft on yrittänyt parhaansa mukaan haastaa Googlea karttabisneksessä ja onkin ominaisuuksiensa puolesta erittäin lähellä. Bing Mapsissa on kuitenkin samat ongelmat, kuin Google Mapsissa. Yrityskäytössä täytyy kartoittaa tarkkaan, minkälaiseen käyttöön karttasovellusta tehdään. Simppelimmässä sovelluksessa joutuisi todennäköisesti maksamaan turhasta, joten suositeltavampaa on käyttää avoimen lähdekoodin rajapintoja.

4.4.3 MapQuest

MapQuest on vertailluista karttapalveluista vanhin. Se on ollut markkinoilla vuodesta 1996 asti. MapQuest on tarjonnut myös paperiversioita kartoista, mutta myi kyseisen osaston pois 2006 lokakuussa. Nykyisin yritys keskittyy internet- ja mobiilikarttoihin. MapQuest on keskittynyt lähinnä Yhdysvaltojen markkinoille. Euroopassa käyttäjistä kilpailevat Google ja Microsoft, jotka tarjoavat enemmän ominaisuuksia palveluissaan. (Wikipedia 2013l.)

Ominaisuuksiltaan MapQuest jää auttamatta kilpailijoidensa Googlen ja Microsoftin jalkoihin. MapQuestin tarjonta perustuu lähinnä tiekarttoihin ja reittien suunnitteluun. Palvelun tarjoamat satelliittikartat ovat vanhoja ja

epätarkkoja Yhdysvaltojen ulkopuolella. Yrityksellä ei myöskään ole tarjota vastinetta Googlen ja Microsoftin katutason kuville, eli Street View ja Streetside -palveluille.

MapQuest-kartoille löytyy rajapinnat JavaScriptille, Flashille, iOS- ja Android-mobiilialustoille, sekä useampi Web Service-rajapinta. Kuvassa 37 olevassa SWOT-analyysissa on kiteytetty MapQuest-rajapinnan ominaisuudet. (MapQuest 2013a.)

Kuva 37. SWOT-analyysi MapQuest karttarajapinnasta.

MapQuest tarjoaa kahta eri versiota palvelustaan. Tarjolla on rajapintoja joko lisensoidulla datalla tai avoimella datalla. Avoimella datalla käytetään OpenStreetMapin karttapohjia, jolloin rajapinnan käytöstä ei peritä mitään maksuja ja käyttöä ei ole rajoitettu mitenkään. Lisensoidulla datalla tarjolla on kaksi eri versiota. Maksullinen Enterprise Edition ja ilmainen Community Edition. Community Editionissa on käyttörajoituksia reitityksen ja hakujen suhteen ja lisäksi palvelua ei saa käyttää maksullisissa verkkosovelluksissa. Vastaavasti Enterprise Editionissa ei ole mitään rajoituksia ja MapQuest tarjoaa ympärivuorokautisen tukipalvelun. Enterprise Editionista joutuu maksamaan noin 2500 dollaria vuodessa. (MapQuest 2013b.), (MapQuest 2013c.)

MapQuest ei oikein pysty kilpailemaan Euroopan markkinoilla, koska kilpailevat karttapalveluita tarjoavat yritykset tarjoavat laadukkaampaa dataa ja enemmän ominaisuuksia rajapinnoillaan. Vaikka MapQuest tarjoaakin avoimen datan lisenssin, jolloin sen käytöstä ei tulisi mitään kustannuksia, tarjolla on myös aidosti avoimen lähdekoodin karttarajapintoja, joihin löytyy huomattavasti helpommin koodiesimerkkejä ja yhteisön apua. Täten MapQuest on varteenotettava vaihtoehto ainoastaan Yhdysvaltojen markkinoilla.

4.4.4 Leaflet

Leaflet on avoimen lähdekoodin karttarajapinta, sekä vertailluista karttarajapinnoista tuorein. Ensimmäisen version julkaisu ajoittuu toukokuulle 2011. Tällä hetkellä rajapinnan kanssa ollaan m enossa versiossa 0.6.3, joten täysin valmiista tuotteesta ei vielä voida puhua. Leaflet rajapintaa kuitenkin markkinoidaan kevyimpänä tarjolla olevana karttarajapintana, jota se varmasti onkin, mutta välttämättömiä ominaisuuksia lukuun ottamatta rajapinta on varsin rajoitettu. Tästä syystä Leaflet luottaakin erillisiin liitännäisiin eli plugineihin. Kaikki erikoisemmat ominaisuudet vaativat erillisen liitännäisen toimiakseen. Esimerkkinä KML-tiedostotuki, jota ei rajapinnassa ole suoraan, vaan KML-tiedostojen luku vaatii erillisen liitännäisen. (Leaflet 2013a.), (Leaflet 2013b.)

Ominaisuuksiltaan Leaflet on aika simppeli. Karttapohjina Leaflet rajapinnan kanssa käytetään yleisesti OpenStreetMapin vapaasti muokattavia karttoja. Mahdollista on myös käyttää Leafletin taustalla olevan Cloudmade yrityksen maksullisia karttapohjia, tai minkä tahansa muun karttapohjia tarjoavan yrityksen karttoja, kuten Googlen. Näissä tapauksissa tulee kuitenkin varmistua kyseisten karttapohjien käytön laillisuudesta karttapohjien tarjoajalta. Leaflet rajapinta antaa työkalut reittien tai alueiden piirtämiseen kartalle, sekä kartan liikuttamiseen. Lisäosilla kartalle saadaan toiminnallisuutta esimerkiksi osoitehaun muodossa. Leaflet tarjoaa rajapinnan JavaSriptille, jolla voidaan tehdä karttasovelluksia sekä mobiilialustoille, että Internet-sivuille työpöytäkäyttöön. (Leaflet 2013c.)

Rajapinnan ominaisuuksia on pohdittu kuvan 38 SWOT-analyysissa.

Vahvuudet	Heikkoudet
• Perustuu avoimeen lähdekoodiin • Ilmainen • Laajennettavuus plugineilla • Tuki mobiililaitteiden selaimille • Erittäin kevyt ja simppeli • Html5 ja css3 • Valmiita ohjeita interaktiiviseen koropleettikarttaan • Drupal moduuli	• Ei omia karttapohjia • Suhteellisen uusi kartta API (ensimmäinen versio 05/2011)
Mahdollisuudet	**Uhat**
• Paljon kolmansien osapuolten tekemiä plugineja	• Vähän resursseja API:n kehittämiseen verrattuna kaupallisiin kilpailijoihin

Kuva 38. SWOT-analyysi Leaflet karttarajapinnasta.

Leaflet käyttää vapaata BSD-lisenssiä, joten rajapinnalla voi tehdä huoletta minkälaisia sovelluksia tahansa ja käyttää niitä kaupallisesti ilman huolia ongelmista. Ainoat maksut joita Leafletin käytössä voi ilmetä, ovat maksullisten karttapohjien käytöstä. (Switch2Osm 2013.)

Leaflet vaikuttaa nuorekkaalta ja kiinnostavalta sen tuoreuden ja keveyden vuoksi, mutta joissakin tapauksissa rajapinnan kevyen luonteen vuoksi käytettävät kolmansien osapuolien liitännäiset tuottavat harmia. Ongelmia tuotti myös Leafletin oletuksena käyttämä koordinaattijärjestelmä, joka ymmärtää koordinaatit latitudi-longitudi järjestyksessä, jolloin longitudi-latitudi datan lukemisessa tulee ongelmia. Plussaa Leaflet saa ilmaisesta lisenssistä, joten jos tarkoituksena on tehdä peruskarttasovellus, tai on valmis työskentelemään joissain tapauksissa keskeneräisten liitännäisten kanssa, on Leaflet ihan hyvä vaihtoehto karttarajapinnaksi.

4.4.5 Openlayers

Openlayers on vertailluista karttarajapinnoista toinen avoimeen lähdekoodiin perustuva rajapinta. Se on ollut markkinoilla lähes yhtä pitkään kuin Googlen karttapalvelu. Openlayersin ensimmäinen versio julkaistiin kesäkuussa 2006. Tällä hetkellä käytössä on rajapinnan toinen versio. Kolmatta Openlayers versiota ollaan jo kehittämässä ja tarkoituksena on kirjoittaa kirjasto kokonaan uusiksi. (Openlayers 2013a.)

Openlayersilla on siis jo pitkä historia ja kehitystyötä sen takana on paljon. Rajapinnasta löytyy perusominaisuuksien lisäksi tuki navigoinnille ja esimerkiksi KML-tiedostojen lukuun rajapinnasta löytyy työkalut suoraan. Tietenkin rajapinnasta puuttuu suurten kaupallisten kilpailijoiden valttikortit kuten Googlen Street View, mutta muuten rajapinnalla pystyy toteuttamaan varmasti vastaavanlaisia karttoja, kuin Googlen rajapinnalla. Openlayersilla ei myöskään ole omia karttapohjia lainkaan, joten yleisesti rajapinnan yhteydessä käytetään kaikille avoimia OpenStreetMapin karttapohjia.

Openlayers rajapinta on JavaScript kirjasto, jolla tehdyt sovellukset toimivat moderneissa internet selaimissa. Version 2.10 jälkeen rajapintaan on sisällytetty myös mobiilialustoilla käytettävät kosketuskomennot, eli kartat toimivat myös monilla mobiililaitteilla.

Kuvassa 39 on pohdittu rajapinnan ominaisuuksia SWOT-analyysin muodossa.

Kuva 39. SWOT-analyysi Openlayers karttarajapinnasta.

Openlayers käyttää avoimen lähdekoodin sovelluksille tuttua BSD-lisenssiä, eli rajapintaa voidaan käyttää vapaasti sekä kaupallisiin että ilmaisiin sovelluksiin. Jos karttasovelluksessa käytetään jonkin muun tarjoajan karttapohjia, kuin OpenStreetMapin, tulee varmistaa niiden käytön laillisuus. (Openlayers 2013b.)

Openlayers on hyvä vaihtoehto karttajapinnaksi, jos tarkoituksena on tehdä peruskarttasovelluksia. Rajapinnasta löytyy tuki kartalle piirtelyyn ja reittien suunnitteluun. Lisäksi yhteisö rajapinnan takana on suhteellisen vankka, joten sovellusta tehdessä eteen tuleviin ongelmiin löytyy varmasti apua eri keskustelupalstoilta. Openlayersin kanssa käytetään yleensä Openstreetmapin karttoja, jotka ovat jo suhteellisen tarkkoja, varsinkin kaupunkialueilla, joten karttapohjissakaan ei häviä paljoa kaupallisille kilpailijoille.

4.4.6 Johtopäätös

Opinnäytetyön ja sovelluksen luonteen takia kaupalliset karttarajapinnat eivät pysty kilpailemaan vertailussa ilmaisten karttarajapintojen kanssa. Suunniteltuun sovellukseen tarvittaviin ominaisuuksiin kuuluu lähinnä ulkoisesta tiedostosta alueiden piirtäminen kartalle ja tähän pystyvät myös ilmaiset rajapinnat. Maksullisten karttajättien Street View ja Streetside toiminnot ovat siis tässä tapauksessa turhia.

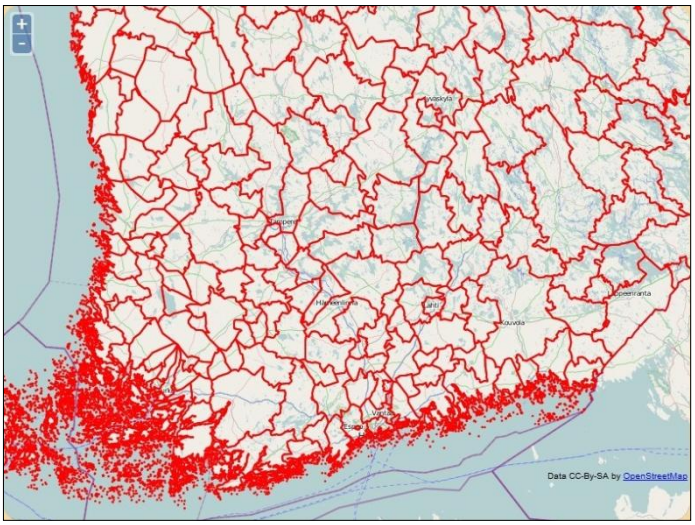
Alun perin käytettäväksi karttarajapinnaksi valittiin opinnäytetyöhön Leaflet, joka vaikutti modernilta ja nuorekkaalta verrattuna kilpailijaansa Openlayersiin. Sovelluksen teon alkuvaiheessa kuitenkin ilmeni ongelmia, sekä Leafletin käyttämän koordinaattijärjestelmän, että kolmansien osapuolten liitännäisten kanssa, joten käytettävä rajapinta päätettiin vaihtaa Openlayersiin. Openlayersillä on myös huomattavasti laajempi yhteisö, jo-

ten mahdollisissa ongelmatilanteissa avun saaminen on todennäköisempää.

4.5 Sovelluksen ohjelmointi

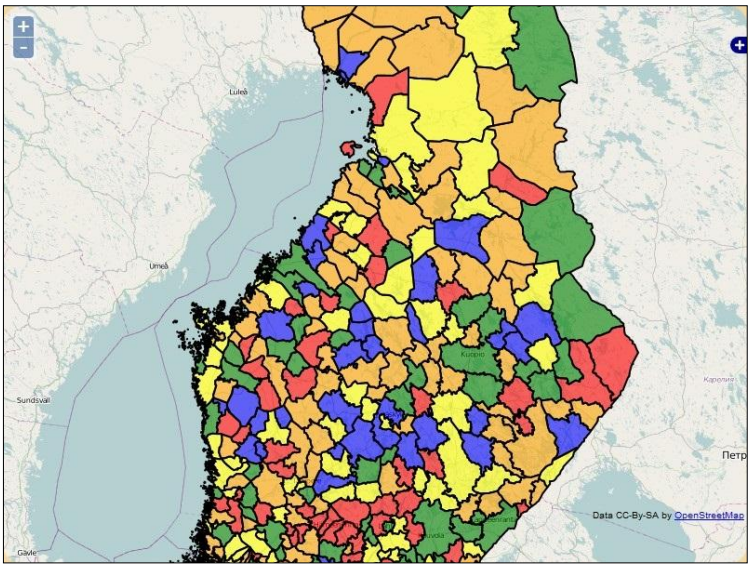
4.5.1 Sovelluksen eteneminen

Sovelluksen ohjelmointi lähti etenemään perusvisiosta, miltä ohjelma voisi näyttää, koodiesimerkkien kokeilemiseen. Aluksi tavoitteena oli saada kartta piirrettyä ja ladattua siihen tasoksi KML-tiedostosta kuntarajat. Tässä ei juuri ongelmia ilmennyt, sillä kartan piirtäminen Openlayersilla on yksinkertaista ja KML-tiedoston lukemiseen löytyy valmis esimerkki rajapinnan arkistosta. Kuvassa 40 on kuvankaappaus kartasta jolle on tuotu kuntarajat KML-tiedostosta.



Kuva 40. Kuntarajat tuotuna kartan päälle omana tasonaan Openlayers karttarajapinnassa.

Seuraavaksi piti pystyä tunnistamaan yksittäiset kunnat tasolta. Tiedossa oli, että jokainen kunta on erikseen piirrettynä ja siten valittavissa, mutta esimerkiksi kunnan nimen saaminen ulos tasosta osoittautui hankalaksi, koska tason rakennetta oli hankala tietää. Tason rakenne ja siihen tallennetut arvot eivät noudattaneet suoraan KML-tiedoston rakennetta, joten ongelman kanssa sai pitkään taistella. Lopulta ratkaisu löytyi tason tulostamisesta konsoliin, jolloin tason rakennetta pystyi selaamaan suoraan selaimen konsolista. Kun tason rakenne oli selvillä, pystyttiin eri kunnat erottamaan toisistaan ja siten antamaan niille ominaisuuksia, esimerkiksi eri täyttövärejä. Seuraava vaihe olikin värittää kunnat eri väreillä niiden ID-numeron perusteella. Tämä on esitetty kuvassa 41.

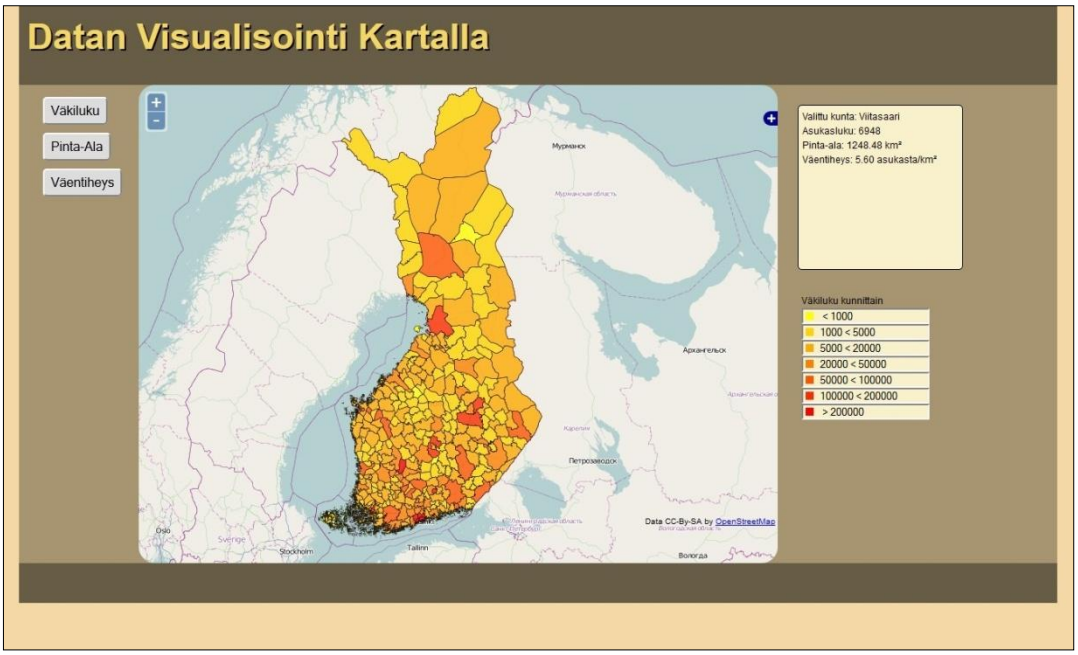


Kuva 41. Kunnat on väritetty eri väreillä niiden ID-numeron perusteella

Tässä vaiheessa oli vielä ongelmia tason tyylien säilyttämisessä ja kuntien värittämisen jälkeisessä kartan päivittämisessä. Kunnat eivät nimittäin värittyneet suoraan funktion ajon jälkeen, vaan uudet värit päivittyivät vasta karttaa liikuteltaessa. Tämä ongelma kuitenkin korjaantui myöhemmin kartan päivitys-funktiolla.

Seuraava vaihe sovelluksen teossa oli hakea dataa tietokannasta ja värittää kunnat tietokannan datan mukaan. Ongelmia tässä aiheutti lähinnä ääkkösiä sisältävien kuntien nimet. Kunnat jotka sisälsivät ääkkösiä, eivät värittyneet lainkaan sovelluksessa. Ongelman syyksi ilmeni php:n käyttämä väärä merkistö. Tämä korjaantui määrittämällä PHP:n käyttämä merkistö UTF-8:aan.

Kun tietokantayhteys oli saatu toimimaan, voitiin jo keskittyä demosovelluksen hienosäätöön ja ulkoasun muokkaamiseen. Lopulliseen sovellukseen tuli kolme eri datahakua, joiden mukaan kunnat väritettiin, selite eri väreille, sekä tulostusalue johon haettiin valitun kunnan tiedot kannasta. Sovelluksen pääaiheena ei ollut sivuston näyttävyys, joten ulkoasu jätettiin pelkistetyksi. Kuvassa 42 on kuvankaappaus valmiista sovelluksesta.



Kuva 42. Valmis demo-sovellus, jossa on esitetty kolme eri datahakua.

Sovelluksesta jäi vielä uupumaan joitakin visuaalisia elementtejä, kuten kuntien valintaan liittyvät tyylit, joilla määritettiin klikatulle kunnalle eri tyyliasu, sekä mahdollistettiin hiirtä liikutettaessa hiiren alla olevan kunnan tyylimuutokset. Nämä toimivat kyllä alkuperäisessä kunta-tasossa, mutta kun kunnat väritetään, katoavat tyyliasetuksetkin.

4.5.2 Koodiesimerkit

Sovellus koostuu PHP- ja JavaScript-koodista, sekä ulkoasun määrittelyyn käytetystä HTML- ja CSS-koodista. PHP-koodilla on toteutettu ohjelman tietokantahaut. Tässä luvussa käydään läpi sovelluksen tärkeimpiä JavaScript-funktioita, jotka liittyvät kartan piirtämiseen. Pääasiassa kartta rakentuu kolmesta eri funktiosta. Ensimmäinen funktio piirtää kartan, toinen piirtää kartalle kuntarajat ja kolmas piirtää kartalle halutun datan mukaisesti väritetyt kunnat.

Ensimmäisenä vuorossa on Init-funktio, jolla itse kartta piirretään sivulle. Kuvassa 43 olevan funktion alussa alustetaan kartta ja määritetään kartan käyttämän div-elementin nimi. Tämän jälkeen määritetään käytettävät karttapohjat, jotka ovat tässä tapauksessa OpenStreetMapin kartat, eli OSM. Kartalle myös määritetään koordinaattipiste ja zoom-taso, johon karttanäkymä on keskitetty ensimmäisellä latauskerralla. Lopussa ajetaan vielä funktio, jolla kartalle piirretään kuntarajat.

```
function init() {
    map = new OpenLayers.Map('map');
    var osm = new OpenLayers.Layer.OSM();
    map.addLayer(osm);
    map.addControl(new OpenLayers.Control.LayerSwitcher({}));
    //Karta keskuskohdan määrittäminen
    map.setCenter(
    new OpenLayers.LonLat(26.987915, 65.138033).transform(
        new OpenLayers.Projection("EPSG:4326"),
        map.getProjectionObject()
    ), 5        //Kartan zoom-taso, arvolla 5 koko suomi näkyy kerralla
    );
    //Ajetaan showKunnat funktio, jolla piirretään kuntien rajat kartalle.
    showKunnat();
}
```

Kuva 43. Init-funktio, jolla itse kartta piirretään, lopussa ajetaan myös kuntarajat piirtävä showKunnat-funktio.

Kuvassa 44 olevaa funktiota käytetään kuntarajojen piirtämiseen ja kuntien tietojen hakemiseen. Alussa tuodaan PHP:n puolelta kaikkien kuntien tiedot sisältämä arraylist JavaScriptiin. Arraylistaa käytetään myöhemmin valitun kunnan tietojen tulostamiseen näytölle. Seuraavassa osassa luodaan taso KML-tiedostosta tuotaville kuntarajoille. Tasolle määritetään eri asetuksia, sekä polku käytettävään KML-tiedostoon. Lopussa taso lisätään kartalle.

```
function showKunnat(){

    //Haetaan PHP:n puolelta arraylist JavaScriptin puolelle.
    //Arraylist sisältää kaikkien kuntien tiedot.
    var tiedotArray = <?php echo json_encode($tiedotArray); ?>;

    //KML-tason luonti, haetaan kunnat_data.kml tiedosto ja annetaan tasolle tarvittavat asetukset.
    kmllayer = new OpenLayers.Layer.Vector("KML", {
        styleMap: styleMap,
        strategies: [new OpenLayers.Strategy.Fixed(), refresh],
        protocol: new OpenLayers.Protocol.HTTP({
            url: "data/kunnat_data.kml",
            format: new OpenLayers.Format.KML({
                extractStyles: false,
                extractAttributes: true
            })
        })
    });

    //Lisätään kartalle KML-taso
    map.addLayer(kmllayer);
```

Kuva 44. Kuntarajojen piirtämiseen käytettävän funktion ensimmäinen osa.

Loput kuntarajat piirtävästä funktiosta on esitetty kuvassa 45. Ensimmäisessä osiossa alustetaan report-muuttuja, jota käytetään HighlightCtrl kuuntelijan kanssa. HighlightCtrl kuuntelija luo KML-tasolle hiiren liikkeeseen reagoivan efektin, jossa hiiren alle jäävä kunta on korostettu eri sävyllä kuin muut kunnat. SelectCtrl kuuntelija taas mahdollistaa eri kuntien hiirellä klikkaamisen. SelectCtrl kuuntelija tarvitaan siis, jos halutaan saada kartalle kuntien valinta ominaisuus. Kuuntelijoiden alustuksen jälkeen molemmat kontrollit lisätään kartalle, jonka lisäksi niille määritetään vielä stopDown ominaisuuden arvoksi false, joka mahdollistaa kartan liikuttelun hiiren ollessa kuntien päällä. Ilman kyseistä false arvoa, kartta ei reagoi mitenkään hiirellä liikuttamiseen, jos hiiri jää piirretyn KML-tason päälle. Lopussa vielä luodaan kuuntelija KML-tasolle, jolla tulostetaan valitun kunnan tiedot näytölle.

```
//Alustetaan highlightCtrl kuuntelijassa käytettävä report muuttuja.
var report = function(e) {
OpenLayers.Console.log(e.type, e.feature.id);
};

//HighlightCtrl kuuntelija, jolla luodaan hover efekti KML-tasolle
var highlightCtrl = new OpenLayers.Control.SelectFeature(kmllayer, {
    hover: true,
    highlightOnly: true,
    renderIntent: "temporary",
    eventListeners: {
        beforefeaturehighlighted: report,
        featurehighlighted: report,
        featureunhighlighted: report
    }
});

//SelectCtrl kuuntelija,
var selectCtrl = new OpenLayers.Control.SelectFeature(kmllayer,
    {
        clickout: true
    }
);

//Lisätään Hover ja Select kontrollit kartalle ja aktivoidaan ne
map.addControl(highlightCtrl);
map.addControl(selectCtrl);
   selectCtrl.handlers.feature.stopDown = false;
   highlightCtrl.handlers.feature.stopDown = false;
highlightCtrl.activate();
selectCtrl.activate();

//Luodaan kuuntelija KML-tasolle. Kuntaa klikatessa tulostetaan kunnan tiedot näytölle.
kmllayer.events.on({
    featureselected: function(event) {
        var feature = event.feature;
        var nimi = feature.data.text.value;
        nimi = nimi.substring(3, nimi.indexOf(','));

        document.getElementById("valinta").innerHTML = "Valittu kunta: " + nimi + "<br>";
        document.getElementById("valinta").innerHTML += "Asukasluku: " + tiedotArray[nimi+"Asukasluku"] + "<br>";
        document.getElementById("valinta").innerHTML += "Pinta-ala: " + tiedotArray[nimi+"PintaAla"] + " km²<br>";
        document.getElementById("valinta").innerHTML += "Väentiheys: " + tiedotArray[nimi+"AsTiheys"] + " asukasta/km²";
    }
});
```

Kuva 45. Kuntarajojen piirtämiseen käytettävän funktion viimeinen osa.

Kolmas esitelty funktio värittää kunnat niiden väkiluvun mukaan. Kaikki ohjelmassa käytettävä esimerkkidata näytetään käyttämällä kyseisenlaista funktiota, vain käytettävät arvot muuttuvat. Kuvassa 46 on esitetty funktion ensimmäinen osa, jossa alustetaan käytettävät muuttujat, jotka sisältävät erikokoiset kunnat erottelevat väkilukurajat, sekä niitä vastaavat värit.

```
function vakiColor() {
 //Alustetaan kuntien väkiluku tiedot sisältävä ArrayList.
 //Alustetaan myös käytettävät väkilukurajat, sekä värit joilla kunnat väritetään.
 var vakiArray = <?php echo json_encode($vakilukuArray); ?>;

 var num1 = 1000;
 var num2 = 5000;
 var num3 = 20000;
 var num4 = 50000;
 var num5 = 100000;
 var num6 = 200000;

 var color1 = "#FFFF00";
 var color2 = "#FFD400";
 var color3 = "#FFAA00";
 var color4 = "#FF7F00";
 var color5 = "#FF5500";
 var color6 = "#FF2A00";
 var color7 = "#FF0000";
```

Kuva 46. Väkiluvun mukaan kunnat värittävän funktion ensimmäinen osa.

Kuvassa 47 on funktion toinen osa, jossa käydään läpi jokainen kunta, ja väritetään ne ArrayListasta löytyvän väkiluvun mukaan. Jokainen kunta tunnistetaan niiden nimen perusteella, joten ennen kuin kyseistä arvoa voidaan käyttää ArrayListan tulkitsemiseen, joudutaan nimi arvoa siistimään. Alkuperäisestä KML-tiedostosta kartan KML-tasoon siirtyvä nimitieto on nimittäin sisällytetty sulkeisiin ja sisältää, sekä suomenkielisen, että ruotsinkielisen kunnan nimen. Nämä on erotettu pilkulla, joten nimestä voidaan siistiä alusta sulkeet ja poistaa kaikki pilkun jälkeinen tieto. Näin saadaan pelkkä suomenkielinen kunnan nimi, jonka avulla voidaan hakea ArrayListaan tallennettu tieto.

```
for(var i = 0 ; i<320 ; i++)
    {
        var feature=kmllayer.features[i];
            try{
                var nimi = kmllayer.features[i].data.text.value;
                nimi = nimi.substring(3, nimi.indexOf(','));
                var vakiLuku = vakiArray[nimi];
                if(vakiLuku < num1)
                {
                    feature.style = {fillColor: color1, fillOpacity: 0.8, 'strokeWidth': 0.5};
                    kmllayer.drawFeature(feature);
                }
                if(vakiLuku >= num1 && vakiLuku < num2)
                {
                    feature.style = {fillColor: color2, fillOpacity: 0.8, 'strokeWidth': 0.5};
                    kmllayer.drawFeature(feature);
                }
                if(vakiLuku >= num2 && vakiLuku < num3)
                {
                    feature.style = {fillColor: color3, fillOpacity: 0.8, 'strokeWidth': 0.5};
                    kmllayer.drawFeature(feature);
                }
                if(vakiLuku >= num3 && vakiLuku < num4)
                {
                    feature.style = {fillColor: color4, fillOpacity: 0.8, 'strokeWidth': 0.5};
                    kmllayer.drawFeature(feature);
                }
                if(vakiLuku >= num4 && vakiLuku < num5)
                {
                    feature.style = {fillColor: color5, fillOpacity: 0.8, 'strokeWidth': 0.5};
                    kmllayer.drawFeature(feature);
                }
                if(vakiLuku >= num5 && vakiLuku < num6)
                {
                    feature.style = {fillColor: color6, fillOpacity: 0.8, 'strokeWidth': 0.5};
                    kmllayer.drawFeature(feature);
                }
                if(vakiLuku >= num6)
                {
                    feature.style = {fillColor: color7, fillOpacity: 0.8, 'strokeWidth': 0.5};
                    kmllayer.drawFeature(feature);
                }
            }
            catch(e)
            {
                console.log(e);
            }
    }
```

Kuva 47. Väkiluvun mukaan kunnat värittävän funktion toinen osa.

Kuvan 48 koodiesimerkissä on esitetty funktion viimeinen osio, jossa tulostetaan vielä näytölle selite funktiossa käytettävistä kuntien väreistä ja niitä vastaavista väkilukurajoista. Tulostukseen käytetään HTML-koodia jonka sekaan on laitettu käytetyt luku- ja värimuuttujat, jolloin selite saadaan täytettyä oikeilla tiedoilla.

```
//Tulostetaan näytölle selite kuntien väreistä ja niiden tarkoituksista.
document.getElementById("legend").innerHTML =  "Väkiluku kunnittain <br>";
document.getElementById("legend").innerHTML +=  "<ul>\n\
                                                <li>\n\
                                                    <div class='input-color'>\n\
                                                        <input type='text' value=' < "+num1+"' />\n\
                                                        <div class='color-box' style='background-color: "+color1+";'></div>\n\
                                                    </div>\n\
                                                </li>\n\
                                                <li>\n\
                                                <li>\n\
                                                    <div class='input-color'>\n\
                                                        <input type='text' value='"+num1+" < "+num2+"' />\n\
                                                        <div class='color-box' style='background-color: "+color2+";'></div>\n\
                                                    </div>\n\
                                                </li>\n\
                                                <li>\n\
                                                    <div class='input-color'>\n\
                                                        <input type='text' value='"+num2+" < "+num3+"' />\n\
                                                        <div class='color-box' style='background-color: "+color3+";'></div>\n\
                                                    </div>\n\
                                                </li>\n\
                                                <li>\n\
                                                    <div class='input-color'>\n\
                                                        <input type='text' value='"+num3+" < "+num4+"' />\n\
                                                        <div class='color-box' style='background-color: "+color4+";'></div>\n\
                                                    </div>\n\
                                                </li>\n\
                                                <li>\n\
                                                    <div class='input-color'>\n\
                                                        <input type='text' value='"+num4+" < "+num5+"' />\n\
                                                        <div class='color-box' style='background-color: "+color5+";'></div>\n\
                                                    </div>\n\
                                                </li>\n\
                                                <li>\n\
                                                    <div class='input-color'>\n\
                                                        <input type='text' value='"+num5+" < "+num6+"' />\n\
                                                        <div class='color-box' style='background-color: "+color6+";'></div>\n\
                                                    </div>\n\
                                                </li>\n\
                                                <li>\n\
                                                    <div class='input-color'>\n\
                                                        <input type='text' value=' > "+num6+"' />\n\
                                                        <div class='color-box' style='background-color: "+color7+";'></div>\n\
                                                    </div>\n\
                                                </li>\n\
                                            </ul>\n\ ";
```

Kuva 48. Väkiluvun mukaan kunnat värittävän funktion kolmas osa, jossa näytölle tulostetaan selite kuntien väreistä ja niiden tarkoituksista.

Sovelluksen lähdekoodi on esitetty kokonaisuudessa liitteissä 2-15.

5 LOPPUSANAT

Opinnäytetyössä oli tarkoitus tutkia mahdollisuuksia käyttää karttarajapintoja asiakasdatan graafiseen esittämiseen. Karttarajapinnalla voitaisiin luoda sovellus, jolla saataisiin nopeasti yleiskuva esimerkiksi yrityksen myynnistä eri alueilla. Tutkimuksen tuloksia oli sitten tarkoitus käyttää jatkokehitykseen toimeksiantajayrityksessä.

Karttarajapintoja valittiin tarkasteltavaksi viisi kappaletta. Näistä Googlen ja Microsoftin rajapinnat olivat selkeästi tunnetuimpia ja tehokkaimpia ominaisuuksiltaan, mutta myös maksullisia yrityskäytössä. Väliinputoajana oli MapQuestin karttarajapinta, joka tarjoaa palveluita sekä lisensoidulla datalla, että avoimella datalla. MapQuest keskittyy pääasiassa Yhdysvaltojen markkinoille, joten sen käyttö täällä Suomessa ei ole suositeltavaa. Avoimen lähdekoodin rajapintoja edustivat Leaflet ja Openlayers. Molemmilla on tarjota omat hyvät ja huonot puolensa. Vaikka Leaflet vaikuttaakin aluksi kätevän kevyeltä paketilta, voi vaativimpien sovellusten tekoon tarvittavien liitännäisten kanssa tulla nopeasti ongelmia. Openlayers taas omaa pitkän kehitystyön ja sisältää heti ominaisuuksia vaativaankin käyttöön.

Demo-sovelluksen toteuttamiseen valittiin Openlayersin rajapinta, koska sen käyttö on ilmaista ja se sisältää kaikki tarvittavat ominaisuudet. Sovelluksen teko koostui käytännössä kolmesta osasta. KML-tiedoston luonnista, joka sisältää tarvittavat koordinaattitiedot käytettävien alueiden rajoista. Kartan ja web-sivun ohjelmointi, sekä tarvittavat tietokantayhteydet, jotta dataa voitiin esittää suoraan tietokannasta.

Alun perin tarkoitus oli luoda postinumeroalueisiin perustuva jako, mutta tämä osoittautui liian työlääksi valmiin datan puutteen vuoksi. Täten päädyttiin käyttämään kuntajakoa, johon löytyi valmis materiaali Maanmittauslaitokselta. Kuntarajat sisältävän KML-tiedoston luonti oli helppoa ja suhteellisen nopeaa. Eniten aikaa tuhraantui odotteluun, kun ohjelma prosessoi dataa.

Itse sovelluksen ohjelmointi eteni lähinnä ongelma kerrallaan. Edellisen ongelman ratkettua ei mennyt kauaa, kun edessä oli jo uusi kompastuskivi. Parhaiten mieleen on jäänyt KML-tiedoston rakenteen kanssa tuskailu, jonka vuoksi eri kuntien tunnistaminen kartalta aiheutti paljon päänvaivaa. Ratkaisu oli lopulta hieman liiankin yksinkertainen, sillä rakenteen pystyi tulostamaan suoraan JavaScript-komennolla web-selaimen konsoliin.

Lopullinen demo-sovellus todistaa sen, että karttarajapinnoilla voidaan tehdä suhteellisen helposti näyttävä lisä asiakasdatan esittämiseen. Ulkoasussa on vielä varmasti hiomista, sillä esimerkiksi kunta-tason tyyleihin liittyvää ongelmaa en saanut ratkaistua. Uskon kuitenkin, että kuntatasolle määritetyt hiiren liikkeisiin reagoivat leijunta ja valinta efektit on mahdollista saada toimimaan myös kuntien värittämisen jälkeen. Myös suorituskyvyssä voisi olla parantamisen varaa, sillä koko Suomen kartan jako kunta-alueisiin ja niiden piirtäminen samanaikaisesti aiheuttaa pientä viivettä sovelluksessa. Postinumeroalueita käytettäessä sovellus olisi voi-

nut olla käyttökelvoton, joten kuntarajat ovat varmasti sopivampi ratkaisu siinäkin mielessä. Mahdollisena ratkaisuna kuntaraja-materiaalinkin kanssa esiintyvään viiveeseen voisi olla piirrettävien kuntien jako maakuntiin. Jos kunnat piirrettäisiin vain maakunnittain, olisi karttarajapinnalla huomattavasti vähemmän prosessoitavaa ja suorituskyky kohenisi.

Ennen opinnäytetyötä minulla oli jo jonkin verran kokemusta Microsoftin ja Googlen karttarajapinnoista, mutta lähinnä perustasolla, joten demosovellusta tehdessäni opin paljon uutta, varsinkin kartalle piirtämisestä ja sen haasteista. Lisäksi PHP:n käytöstä minulla ei juuri kokemusta ollut, joten sen saralla opin myös paljon uutta. Mielestäni opinnäytetyön alussa asetetut tavoitteet täyttyivät sekä demo-sovelluksen, että tutkimustyön osalta. Sovelluksessa käytettyjä JavaScript-funktioita pystyisi varmasti yksinkertaistamaan huomattavasti, mutta sovellusta tehtäessä taidot olivat vielä perustasolla.

LÄHTEET

Bellis M. 2013. The History of HTML.Viitattu 29.5.2013.
http://inventors.about.com/od/computersoftware/a/html.htm

Bing Maps. 2013. Bing Maps lisenssit. Viitattu 18.7.2013.
http://www.microsoft.com/maps/

Georelated. 2013. Bing Maps API reviewed. Viitattu 18.7.2013.
http://www.georelated.com/2012/02/cloud-web-map-api-services-reviewed_19.html

Google. 2013a. Google Maps - Usage Limits and Billing. Viitattu 13.7.2013.
https://developers.google.com/maps/documentation/javascript/usage

Google. 2013b. Google Maps FAQ - Usage Pricing. Viitattu 13.7.2013.
https://developers.google.com/maps/faq#usage_pricing

Itella. 2013. Postinumeropalvelut – Palvelukuvaus ja käyttöehdot.Viitattu 24.4.2013.
http://www.itella.fi/liitteet/palvelutjatuotteet/yhteystietopalvelut/postinumeropalvelut_palvelukuvaus_ja_kayttoehdot.pdf

Kantor, P. 2003. CSS History. Viitattu 30.5.2013.
http://www.daaq.net/old/css/index.php?page=css+history&parent=using+css

Kioskea.net. 2013. DBSM models. Viitattu 11.6.2013.
http://en.kioskea.net/contents/67-dbms-models

Landofcode.com. 2013.History of HTML.Viitattu 30.5.2013.
http://www.landofcode.com/html-tutorials/html-history.php

Latuviitta. OpenJUMP. Viitattu: 25.3.2013.
http://www.latuviitta.org/OpenJUMP.php

Leaflet. 2013a. Leaflet Changelog. Viitattu 22.7.2013.
https://github.com/Leaflet/Leaflet/blob/master/CHANGELOG.md

Leaflet. 2013b. Leaflet Overview. Viitattu 22.7.2013.
http://leafletjs.com/index.html

Leaflet. 2013c. Leaflet Features. Viitattu 22.7.2013.
http://leafletjs.com/features.html

Lehtomäki J. 2012. Kuntarajat leikattuna suomen rantaviivalla. Viitattu 3.6.2013.
https://github.com/kansanmuisti/datavaalit/wiki/kuntarajat-leikkatuna-suomen-rantaviivalla

Maanmittauslaitos. 2013a. TM35-lehtijako. Viitattu 3.4.2013.
http://www.maanmittauslaitos.fi/kartat/koordinaatit/maastokartat/tm35-lehtijako

Maanmittauslaitos. 2013b. Ilmaisten aineistojen käyttöehdot/lisenssi. Viitattu 14.6.2013.
http://www.maanmittauslaitos.fi/ilmaiset_aineistot_lisenssi_20120430asti

Maanmittauslaitos3. Tuotekuvaukset – Kuntajako. Viitattu 26.3.2013.
http://www.maanmittauslaitos.fi/digituotteet/maastotietokannan-tiesto-osoitteilla

MapQuest. 2013a. MapQuest Licensed Data Map APIs an Web Services. Viitattu 22.7.2013.
http://developer.mapquest.com/web/products

MapQuest. 2013b. Licensed Data vs. Open Data. Viitattu 22.7.2013.
http://developer.mapquest.com/web/tools/getting-started/platform/licensed-vs-open

MapQuest. 2013c. Licensing and Terms Overview. Viitattu 22.7.2013.
http://developer.mapquest.com/web/tools/getting-started/terms-overview

MacDonald, M. 2011. HTML5 the missing manual. Yhdysvallat: O'Reilly.
ISBN-10: 1449302394

McFarland, D. S. 2012. CSS3 the missing manual. Yhdysvallat: O'Reilly.
ISBN-10: 1449325947

McLaughlin, B. 2012. PHP & MySQL the missing manual. Yhdysvallat: O'Reilly.
ISBN-10: 1449325572

Moisio A. 2008. MySQL – Suomalainen menestystarina. Viitattu 30.5.2013.
http://www.itviikko.fi/ratkaisut/2008/01/16/mysql--suomalainen-menestystarina/20081483/7

Openlayers. 2013a. Openlayers History. Viitattu 24.7.2013.
http://trac.osgeo.org/openlayers/wiki/History

Openlayers. 2013b. Openlayers: Home. Viitattu 24.7.2013.
http://www.openlayers.org/

Php.net. 2013.History of PHP.Viitattu 23.5.2013.
http://www.php.net/manual/en/history.php.php

Sarkola P. 2012. Paikkatietoaineisto KML-muotoon. Viitattu 3.6.2013.
http://www.gispo.fi/Ohjeita/Paikkatietoaineisto-KML-muotoon

Switch2Osm. 2013. Getting started with leaflet. Viitattu 23.7.2013. http://switch2osm.org/using-tiles/getting-started-with-leaflet/

Teuhola J. 2002. Oliotietokannat. Viitattu 11.6.2013. http://staff.cs.utu.fi/kurssit/tietokannat/05/pdf/OlioKannat.pdf

TIOBE. 2013. TIOBE Programming Community Index for May 2013.Viitattu 4.6.2013. http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

w3.org. 2012a. JavaScript - adding behavior to web pages. Viitattu 23.5.2013. http://www.w3.org/community/webed/wiki/The_web_standards_model_-_HTML_CSS_and_JavaScript

w3.org. 2012b. A Short History of JavaScript.Viitattu 23.5.2013. http://www.w3.org/community/webed/wiki/A_Short_History_of_JavaScript

Webvideobytes. 2009. How PHP Works. Viitattu 23.5.2013 http://www.youtube.com/watch?v=PemsuAfc7Jw

Wikibooks. 2013. Structured Query Language. Viitattu 13.6.2013. http://en.wikibooks.org/wiki/Structured_Query_Language/Introduction_to_SQL

Wikipedia. 2013a. Geographic Information System. Viitattu 19.6.2013. http://en.wikipedia.org/wiki/Geographic_information_system

Wikipedia. 2013b. Koordinaattijärjestelmä. Viitattu 23.6.2013. http://fi.wikipedia.org/wiki/Koordinaattij%C3%A4rjestelm%C3%A4

Wikipedia. 2013c. Shapefile. Viitattu 23.6.2013. http://en.wikipedia.org/wiki/Shapefile

Wikipedia. 2013d. PHP Viitattu 23.5.2013. http://fi.wikipedia.org/wiki/PHP

Wikipedia. 2013e. JavaScript. Viitattu 23.5.2013. http://en.wikipedia.org/wiki/JavaScript

Wikipedia. 2013f. HTML. Viitattu 30.5.2013. https://fi.wikipedia.org/wiki/HTML

Wikipedia. 2013g. SQL. Viitattu 13.6.2013. http://en.wikipedia.org/wiki/SQL

Wikipedia. 2013h. MySQL. Viitattu 30.5.2013. http://fi.wikipedia.org/wiki/MySQL

Wikipedia. 2013i. NetBeans. Viitattu 3.6.2013.

http://en.wikipedia.org/wiki/NetBeans

Wikipedia. 2013j. Google Maps. Viitattu 27.6.2013.
http://en.wikipedia.org/wiki/Google_Maps

Wikipedia. 2013k. Bing Maps. Viitattu 17.7.2013.
http://en.wikipedia.org/wiki/Bing_Maps

Wikipedia. 2013l. MapQuest. viitattu 22.7.2013.
http://en.wikipedia.org/wiki/MapQuest

Liite 1

MAANMITTAUSLAITOKSEN KUNTAJAKO - KÄSITTELYMAKSUT

Kuntajako

Aineisto ja käsittely on maksutonta, kun aineisto toimitetaan http://www.maanmittauslaitos.fi/ilmaisetaineistot -sivuston kautta.

Jos aineistoa toimitetaan muuten, peritään alla olevan taulukon mukainen käsittelymaksu.

	Arvonlisäveroton hinta, euroa (€)	**Arvonlisäverollinen hinta, euroa (€), alv. 24 %**
KÄSITTELYMAKSUT:		
VUOSIVERSIOT		
koko Suomi		
- Kuntajako 1:10 000	80,00	99,20
- Kuntajako 1:100 000	80,00	99,20
- Kuntajako 1:250 000	80,00	99,20
- Kuntajako 1:1 milj.	80,00	99,20
- Kuntajako 1:4,5 milj.	80,00	99,20
- Kuntajakorasteri 1:10 000	80,00	99,20
- Kuntajakorasteri 1:100 000	80,00	99,20
- Kuntajakorasteri 1:250 000	80,00	99,20
- Kuntajakorasteri 1:1 milj.	50,00	62,00
- Kuntajakorasteri 1:4,5 milj.	50,00	62,00

Asiasanat: Hintatieto

Liite 2

SOVELLUKSEN LÄHDEKOODI – OSA 1

```
<!DOCTYPE html>
<html>
    <head>
        <link rel="stylesheet" type="text/css" href="style.css">
        <link rel="stylesheet" type="text/css" href="theme/default/style.css">
        <script src="OpenLayers.js"></script>
    </head>
    <body onload="init();">
        <?php
        header('Content-type: text/html; charset=UTF-8');
        // Luo tietokantayhteys
        $mysqli = new mysqli("localhost", "root", "", "oppari");
        // Vaihdetaan käytettävä merkistö utf8:aan
        if (!$mysqli->set_charset("utf8")) {

        } else {
        }

        // Yhteystarkistus
        if (mysqli_connect_errno($mysqli))
          {
          echo "Failed to connect to MySQL: " . mysqli_connect_error();
          }


          // Tietokantakysely väkilukutietojen hakemiseen
          $result = mysqli_query($mysqli,"SELECT k.Nimi AS Nimi, d.asukasluku AS Asukasluku
            FROM kunnat AS k
            INNER JOIN data AS d
            ON k.id = d.id_kunnat");

        while($row = mysqli_fetch_array($result))
          {
            $id = $row['Nimi'];
            $vakilukuArray[$id] = $row['Asukasluku'];
          }

          // Tietokantakysely pinta-ala tietojen hakemiseen
          $result2 = mysqli_query($mysqli, "SELECT k.Nimi AS Nimi, d.pintaAla AS PintaAla
            FROM kunnat AS k
            INNER JOIN data AS d
            ON k.id = d.id_kunnat");

         while($row = mysqli_fetch_array($result2))
          {
            $id = $row['Nimi'];
            $paArray[$id] = $row['PintaAla'];
          }
```

Liite 3

SOVELLUKSEN LÄHDEKOODI – OSA 2

```
  // Tietokantakysely asumistiheyden tietojen hakemiseen
  $result3 = mysqli_query($mysqli, "SELECT k.Nimi AS Nimi, d.asTiheys AS AsTiheys
    FROM kunnat AS k
    INNER JOIN data AS d
    ON k.id = d.id_kunnat");

 while($row = mysqli_fetch_array($result3))
  {
    $id = $row['Nimi'];
    $asTiheysArray[$id] = $row['AsTiheys'];
  }

  // Haetaan tietokannasta kuntien tiedot. Tiedot näytetään kuntien valinnan yhteydessä.
  $result4 = mysqli_query($mysqli, "SELECT k.Nimi AS Nimi, d.asTiheys AS AsTiheys, d.asukasluku AS Asukasluku, d.pintaAla AS PintaAla
    FROM kunnat AS k
    INNER JOIN data AS d
    ON k.id = d.id_kunnat");

 while($row = mysqli_fetch_array($result4))
  {
    $id = $row['Nimi'];
    $tiedotArray[$id."Asukasluku"] = $row['Asukasluku'];
    $tiedotArray[$id."PintaAla"] = $row['PintaAla'];
    $tiedotArray[$id."AsTiheys"] = $row['AsTiheys'];
  }
?>


<div id="wrapper">

<header id="header">
    <h1>Datan Visualisointi Kartalla</h1>
</header>

<section id="middle">

        <div id="container">
                <div id="content">
                    <div id="map"></div>

                        <script>

                                var map;
                                var kmllayer;

                                new OpenLayers.StyleMap(styleMap);
                                var refresh = new OpenLayers.Strategy.Refresh({force: true, active: true});
```

Liite 4

SOVELLUKSEN LÄHDEKOODI – OSA 3

```
// Tyylikartan alustus, tyylit katoaa kuntien värityksen yhteydessä.
var styleMap = new OpenLayers.StyleMap({
    "default": new OpenLayers.Style({
        fillOpacity: 0.1,
        strokeWidth: 0.5

    }),
    "select": new OpenLayers.Style({
        fillOpacity: 1,
        strokeWidth: 3
    }),
    "temporary": new OpenLayers.Style({
        fillOpacity: 0.3,
        strokeWidth: 1
    })
});
```

```
// Funktio, jolla kunnat haetaan KML-tiedostosta ja piirretään kartalle.
function showKunnat(){

    var tiedotArray = <?php echo json_encode($tiedotArray); ?>;
    filterStrategy = new OpenLayers.Strategy.Filter({});

    // KML-taso luodaan, määritetään asetukset.
    kmllayer = new OpenLayers.Layer.Vector("KML", {
        styleMap: styleMap,
        strategies: [new OpenLayers.Strategy.Fixed(), refresh],
        protocol: new OpenLayers.Protocol.HTTP({
            url: "data/kunnat_data.kml",
            format: new OpenLayers.Format.KML({
                extractStyles: false,
                extractAttributes: true
            })
        })
    });

    map.addLayer(kmllayer);

    // report muuttuja, jota käytetään hightlightCtrl -kontrollerissa.
    var report = function(e) {
    OpenLayers.Console.log(e.type, e.feature.id);
    };
```

Liite 5

SOVELLUKSEN LÄHDEKOODI – OSA 4

```
    var highlightCtrl = new OpenLayers.Control.SelectFeature(kmllayer, {
        hover: true,
        highlightOnly: true,
        renderIntent: "temporary",
        eventListeners: {
            beforefeaturehighlighted: report,
            featurehighlighted: report,
            featureunhighlighted: report
        }
    });

    var selectCtrl = new OpenLayers.Control.SelectFeature(kmllayer,
        {
            clickout: true
        }
    );


    map.addControl(highlightCtrl);
    map.addControl(selectCtrl);
       selectCtrl.handlers.feature.stopDown = false;
       highlightCtrl.handlers.feature.stopDown = false;
    highlightCtrl.activate();
    selectCtrl.activate();

    // Valitun kunnan tietojen tulostus tietolaatikkoon.
    kmllayer.events.on({
        featureselected: function(event) {
            var feature = event.feature;
            var nimi = feature.data.text.value;
            nimi = nimi.substring(3, nimi.indexOf(','));

            document.getElementById("valinta").innerHTML = "Valittu kunta: " + nimi + "<br>";
            document.getElementById("valinta").innerHTML += "Asukasluku: " + tiedotArray[nimi+"Asukasluku"] + "<br>";
            document.getElementById("valinta").innerHTML += "Pinta-ala: " + tiedotArray[nimi+"PintaAla"] + " km²<br>";
            document.getElementById("valinta").innerHTML += "Väentiheys: " + tiedotArray[nimi+"AsTiheys"] + " asukasta/km²";
        }
    });



}
```

Liite 6

SOVELLUKSEN LÄHDEKOODI – OSA 5

```
// Kuntien väkiluvun mukaan väritettävän kartan tekoon käytettävä funktio.
function vakiColor()
{

 var vakiArray = <?php echo json_encode($vakilukuArray); ?>;

 // Kuntien väkilukujen erotteluun käytettävät numerot ja
 // kartalla käytettävät värit
 var num1 = 1000;
 var num2 = 5000;
 var num3 = 20000;
 var num4 = 50000;
 var num5 = 100000;
 var num6 = 200000;

 var color1 = "#FFFF00";
 var color2 = "#FFD400";
 var color3 = "#FFAA00";
 var color4 = "#FF7F00";
 var color5 = "#FF5500";
 var color6 = "#FF2A00";
 var color7 = "#FF0000";


    // Kunnat käydään läpi yksitellen ja väritetään niille kuuluvalla värillä
    for(var i = 0 ; i<320 ; i++)
        {
            var feature=kmllayer.features[i];

            try{
                var nimi = kmllayer.features[i].data.text.value;
                nimi = nimi.substring(3, nimi.indexOf(','));
                var vakiLuku = vakiArray[nimi];
                if(vakiLuku < num1)
                {
                    feature.style = {fillColor: color1, fillOpacity: 0.8, 'strokeWidth': 0.5};
                    kmllayer.drawFeature(feature);

                }

                if(vakiLuku >= num1 && vakiLuku < num2)
                {
                    feature.style = {fillColor: color2, fillOpacity: 0.8, 'strokeWidth': 0.5};
                    kmllayer.drawFeature(feature);

                }
```

Liite 7

SOVELLUKSEN LÄHDEKOODI – OSA 6

```
        if(vakiLuku >= num2 && vakiLuku < num3)
        {
            feature.style = {fillColor: color3, fillOpacity: 0.8, 'strokeWidth': 0.5};
            kmllayer.drawFeature(feature);

        }

        if(vakiLuku >= num3 && vakiLuku < num4)
        {
            feature.style = {fillColor: color4, fillOpacity: 0.8, 'strokeWidth': 0.5};
            kmllayer.drawFeature(feature);


        }
        if(vakiLuku >= num4 && vakiLuku < num5)
        {
            feature.style = {fillColor: color5, fillOpacity: 0.8, 'strokeWidth': 0.5};
            kmllayer.drawFeature(feature);


        }
        if(vakiLuku >= num5 && vakiLuku < num6)
        {
            feature.style = {fillColor: color6, fillOpacity: 0.8, 'strokeWidth': 0.5};
            kmllayer.drawFeature(feature);


        }
        if(vakiLuku >= num6)
        {
            feature.style = {fillColor: color7, fillOpacity: 0.8, 'strokeWidth': 0.5};
            kmllayer.drawFeature(feature);


        }
    }
    catch(e)
    {
        console.log(e);
    }


}
```

Liite 8

SOVELLUKSEN LÄHDEKOODI – OSA 7

```
    // Täytetään kuntien värejä vastaava seloste oikeilla luvuilla
    document.getElementById("legend").innerHTML =  "Väkiluku kunnittain <br>";
    document.getElementById("legend").innerHTML +=  "<ul>\n\
                                                        <li>\n\
                                                            <div class='input-color'>\n\
                                                                <input type='text' value=' < "+num1+"' />\n\
                                                                <div class='color-box' style='background-color: "+color1+";'></div>\n\
                                                            </div>\n\
                                                        </li>\n\
                                                        <li>\n\
                                                        <li>\n\
                                                            <div class='input-color'>\n\
                                                                <input type='text' value='"+num1+" < "+num2+"' />\n\
                                                                <div class='color-box' style='background-color: "+color2+";'></div>\n\
                                                            </div>\n\
                                                        </li>\n\
                                                        <li>\n\
                                                            <div class='input-color'>\n\
                                                                <input type='text' value='"+num2+" < "+num3+"' />\n\
                                                                <div class='color-box' style='background-color: "+color3+";'></div>\n\
                                                            </div>\n\
                                                        </li>\n\
                                                        <li>\n\
                                                            <div class='input-color'>\n\
                                                                <input type='text' value='"+num3+" < "+num4+"' />\n\
                                                                <div class='color-box' style='background-color: "+color4+";'></div>\n\
                                                            </div>\n\
                                                        </li>\n\
                                                        <li>\n\
                                                            <div class='input-color'>\n\
                                                                <input type='text' value='"+num4+" < "+num5+"' />\n\
                                                                <div class='color-box' style='background-color: "+color5+";'></div>\n\
                                                            </div>\n\
                                                        </li>\n\
                                                        <li>\n\
                                                            <div class='input-color'>\n\
                                                                <input type='text' value='"+num5+" < "+num6+"' />\n\
                                                                <div class='color-box' style='background-color: "+color6+";'></div>\n\
                                                            </div>\n\
                                                        </li>\n\
                                                        <li>\n\
                                                            <div class='input-color'>\n\
                                                                <input type='text' value=' > "+num6+"' />\n\
                                                                <div class='color-box' style='background-color: "+color7+";'></div>\n\
                                                            </div>\n\
                                                        </li>\n\
                                                    </ul>\n\
                                                    ";
}
```

Liite 9

SOVELLUKSEN LÄHDEKOODI – OSA 8

```
// Kuntien pinta-alan mukaan väritettävän kartan tekoon käytettävä funktio.
function paColor()
{
 var paArray = <?php echo json_encode($paArray); ?>;

 // Kuntien pinta-alan erotteluun käytettävät numerot ja
 // kartalla käytettävät värit
 var num1 = 100;
 var num2 = 300;
 var num3 = 500;
 var num4 = 1000;
 var num5 = 2000;
 var num6 = 15000;

 var color1 = "#FFFF00";
 var color2 = "#FFD400";
 var color3 = "#FFAA00";
 var color4 = "#FF7F00";
 var color5 = "#FF5500";
 var color6 = "#FF2A00";
 var color7 = "#FF0000";

    // Kunnat käydään läpi yksitellen ja väritetään niille kuuluvalla värillä
    for(var i = 0 ; i<320 ; i++)
        {
            var feature=kmllayer.features[i];

            try{
                var nimi = kmllayer.features[i].data.text.value;
                nimi = nimi.substring(3, nimi.indexOf(','));
                var pAla = paArray[nimi];
                if(pAla < num1)
                {
                    feature.style = {fillColor: color1, fillOpacity: 0.8, 'strokeWidth': 0.5};
                    kmllayer.drawFeature(feature);

                }

                if(pAla >= num1 && pAla < num2)
                {
                    feature.style = {fillColor: color2, fillOpacity: 0.8, 'strokeWidth': 0.5};
                    kmllayer.drawFeature(feature);

                }
```

Liite 10

SOVELLUKSEN LÄHDEKOODI – OSA 9

```
        if(pAla >= num2 && pAla < num3)
        {
            feature.style = {fillColor: color3, fillOpacity: 0.8, 'strokeWidth': 0.5};
            kmllayer.drawFeature(feature);

        }

        if(pAla >= num3 && pAla < num4)
        {
            feature.style = {fillColor: color4, fillOpacity: 0.8, 'strokeWidth': 0.5};
            kmllayer.drawFeature(feature);


        }
        if(pAla >= num4 && pAla < num5)
        {
            feature.style = {fillColor: color5, fillOpacity: 0.8, 'strokeWidth': 0.5};
            kmllayer.drawFeature(feature);


        }
        if(pAla >= num5 && pAla < num6)
        {
            feature.style = {fillColor: color6, fillOpacity: 0.8, 'strokeWidth': 0.5};
            kmllayer.drawFeature(feature);


        }
        if(pAla >= num6)
        {
            feature.style = {fillColor: color7, fillOpacity: 0.8, 'strokeWidth': 0.5};
            kmllayer.drawFeature(feature);


        }
    }
    catch(e)
    {
        console.log(e);
    }


}
```

Liite 11

SOVELLUKSEN LÄHDEKOODI – OSA 10

```
    // Täytetään kuntien värejä vastaava seloste oikeilla luvuilla
    document.getElementById("legend").innerHTML =  "Pinta-ala kunnittain km²";
    document.getElementById("legend").innerHTML +=  "<ul>\n\
                                                        <li>\n\
                                                            <div class='input-color'>\n\
                                                                <input type='text' value=' < "+num1+"' />\n\
                                                                <div class='color-box' style='background-color: "+color1+";'></div>\n\
                                                            </div>\n\
                                                        </li>\n\
                                                        <li>\n\
                                                        <li>\n\
                                                            <div class='input-color'>\n\
                                                                <input type='text' value='"+num1+" < "+num2+"' />\n\
                                                                <div class='color-box' style='background-color: "+color2+";'></div>\n\
                                                            </div>\n\
                                                        </li>\n\
                                                        <li>\n\
                                                            <div class='input-color'>\n\
                                                                <input type='text' value='"+num2+" < "+num3+"' />\n\
                                                                <div class='color-box' style='background-color: "+color3+";'></div>\n\
                                                            </div>\n\
                                                        </li>\n\
                                                        <li>\n\
                                                            <div class='input-color'>\n\
                                                                <input type='text' value='"+num3+" < "+num4+"' />\n\
                                                                <div class='color-box' style='background-color: "+color4+";'></div>\n\
                                                            </div>\n\
                                                        </li>\n\
                                                        <li>\n\
                                                            <div class='input-color'>\n\
                                                                <input type='text' value='"+num4+" < "+num5+"' />\n\
                                                                <div class='color-box' style='background-color: "+color5+";'></div>\n\
                                                            </div>\n\
                                                        </li>\n\
                                                        <li>\n\
                                                            <div class='input-color'>\n\
                                                                <input type='text' value='"+num5+" < "+num6+"' />\n\
                                                                <div class='color-box' style='background-color: "+color6+";'></div>\n\
                                                            </div>\n\
                                                        </li>\n\
                                                        <li>\n\
                                                            <div class='input-color'>\n\
                                                                <input type='text' value=' > "+num6+"' />\n\
                                                                <div class='color-box' style='background-color: "+color7+";'></div>\n\
                                                            </div>\n\
                                                        </li>\n\
                                                    </ul>\n\
                                                    ";
}
```

Liite 12

SOVELLUKSEN LÄHDEKOODI – OSA 11

```
// Kuntien asukastiheyden mukaan väritettävän kartan tekoon käytettävä funktio.
function asTiheysColor()
{
 var asTiheysArray = <?php echo json_encode($asTiheysArray); ?>;

 // Kuntien asukastiheyden erotteluun käytettävät numerot ja
 // kartalla käytettävät värit
 var num1 = 5;
 var num2 = 15;
 var num3 = 50;
 var num4 = 100;
 var num5 = 500;
 var num6 = 1000;

 var color1 = "#FFFF00";
 var color2 = "#FFD400";
 var color3 = "#FFAA00";
 var color4 = "#FF7F00";
 var color5 = "#FF5500";
 var color6 = "#FF2A00";
 var color7 = "#FF0000";

    // Kunnat käydään läpi yksitellen ja väritetään niille kuuluvalla värillä
    for(var i = 0 ; i<320 ; i++)
        {
            var feature=kmllayer.features[i];

            try{
                var nimi = kmllayer.features[i].data.text.value;
                nimi = nimi.substring(3, nimi.indexOf(','));
                var asTiheys = asTiheysArray[nimi];
                if(asTiheys < num1)
                {
                    feature.style = {fillColor: color1, fillOpacity: 0.8, 'strokeWidth': 0.5};
                    kmllayer.drawFeature(feature);

                }

                if(asTiheys >= num1 && asTiheys < num2)
                {
                    feature.style = {fillColor: color2, fillOpacity: 0.8, 'strokeWidth': 0.5};
                    kmllayer.drawFeature(feature);

                }
```

Liite 13

SOVELLUKSEN LÄHDEKOODI – OSA 12

```
        if(asTiheys >= num2 && asTiheys < num3)
        {
            feature.style = {fillColor: color3, fillOpacity: 0.8, 'strokeWidth': 0.5};
            kmllayer.drawFeature(feature);

        }

        if(asTiheys >= num3 && asTiheys < num4)
        {
            feature.style = {fillColor: color4, fillOpacity: 0.8, 'strokeWidth': 0.5};
            kmllayer.drawFeature(feature);


        }
        if(asTiheys >= num4 && asTiheys < num5)
        {
            feature.style = {fillColor: color5, fillOpacity: 0.8, 'strokeWidth': 0.5};
            kmllayer.drawFeature(feature);


        }
        if(asTiheys >= num5 && asTiheys < num6)
        {
            feature.style = {fillColor: color6, fillOpacity: 0.8, 'strokeWidth': 0.5};
            kmllayer.drawFeature(feature);


        }
        if(asTiheys >= num6)
        {
            feature.style = {fillColor: color7, fillOpacity: 0.8, 'strokeWidth': 0.5};
            kmllayer.drawFeature(feature);


        }
    }
    catch(e)
    {
        console.log(e);
    }



}
```

Liite 14

SOVELLUKSEN LÄHDEKOODI – OSA 13

```
    // Täytetään kuntien värejä vastaava seloste oikeilla luvuilla
    document.getElementById("legend").innerHTML =  "Väestön tiheys asukasta / km²";
    document.getElementById("legend").innerHTML +=  "<ul>\n\
                                                        <li>\n\
                                                            <div class='input-color'>\n\
                                                                <input type='text' value=' < "+num1+"' />\n\
                                                                <div class='color-box' style='background-color: "+color1+";'></div>\n\
                                                            </div>\n\
                                                        </li>\n\
                                                        <li>\n\
                                                        <li>\n\
                                                            <div class='input-color'>\n\
                                                                <input type='text' value='"+num1+" < "+num2+"' />\n\
                                                                <div class='color-box' style='background-color: "+color2+";'></div>\n\
                                                            </div>\n\
                                                        </li>\n\
                                                        <li>\n\
                                                            <div class='input-color'>\n\
                                                                <input type='text' value='"+num2+" < "+num3+"' />\n\
                                                                <div class='color-box' style='background-color: "+color3+";'></div>\n\
                                                            </div>\n\
                                                        </li>\n\
                                                        <li>\n\
                                                            <div class='input-color'>\n\
                                                                <input type='text' value='"+num3+" < "+num4+"' />\n\
                                                                <div class='color-box' style='background-color: "+color4+";'></div>\n\
                                                            </div>\n\
                                                        </li>\n\
                                                        <li>\n\
                                                            <div class='input-color'>\n\
                                                                <input type='text' value='"+num4+" < "+num5+"' />\n\
                                                                <div class='color-box' style='background-color: "+color5+";'></div>\n\
                                                            </div>\n\
                                                        </li>\n\
                                                        <li>\n\
                                                            <div class='input-color'>\n\
                                                                <input type='text' value='"+num5+" < "+num6+"' />\n\
                                                                <div class='color-box' style='background-color: "+color6+";'></div>\n\
                                                            </div>\n\
                                                        </li>\n\
                                                        <li>\n\
                                                            <div class='input-color'>\n\
                                                                <input type='text' value=' > "+num6+"' />\n\
                                                                <div class='color-box' style='background-color: "+color7+";'></div>\n\
                                                            </div>\n\
                                                        </li>\n\
                                                    </ul>\n\
                                                    ";
}
```

Liite 15

SOVELLUKSEN LÄHDEKOODI – OSA 14

```
                                function refresh() {
                                    kmllayer.redraw(true);
                                    kmllayer.refresh({force: true});
                                }
                                // Funktio jolla kartta alustetaan ja lisätään sivulle. Funktio ajetaan sivun bodyn latautuessa.
                                function init() {
                                    map = new OpenLayers.Map('map');
                                    var osm = new OpenLayers.Layer.OSM();
                                    map.addLayer(osm);
                                    map.addControl(new OpenLayers.Control.LayerSwitcher({}));

                                    map.setCenter(
                                    new OpenLayers.LonLat(26.987915, 65.138033).transform(
                                        new OpenLayers.Projection("EPSG:4326"),
                                        map.getProjectionObject()
                                    ), 5 // Zoom taso, koko suomi näkyy numerolla 5
                                    );
                                    showKunnat();
                                }
                        </script>
                        </div><!-- #content-->
                </div><!-- #container-->
                <aside id="sideLeft">
                    <div id="napit">
                    <input type="button" value="Väkiluku" class="button" onclick="vakiColor();">
                    <input type="button" value="Pinta-Ala" class="button" onclick="paColor();">
                    <input type="button" value="Väentiheys" class="button" onclick="asTiheysColor();">
                    </div>
                </aside><!-- #sideLeft -->
                <aside id="sideRight">
                    <div id="valinta">
                        Valittu kunta: <br>
                        <br>
                        <label id="nimiLabel" value=""></label>
                        <br>
                        <label id="asukaslukuLabel" value=""></label>
                        <br>
                        <label id="pintaalaLabel" value=""></label>
                        <br>
                        <label id="astiheysLabel"></label>
                    </div>
                    <div id="legend" class="input-color"></div>
                </aside><!-- #sideRight -->
        </section><!-- #middle-->
        <footer id="footer">
        </footer><!-- #footer -->
    </div><!-- #wrapper -->
    </body>
</html>
```