HAAGA-HELIA
ammattikorkeakoulu

Paikkatietorajapintojen hyödyntäminen

Case Finavia

Tero Tuomala

Opinnäytetyö
Tietojenkäsittelyn koulutusohjelma
2014

HAAGA-HELIA
ammattikorkeakoulu

Tiivistelmä

19.11.2014

Tietojenkäsittelyn koulutusohjelma

Tekijä tai tekijät Tero Tuomala	**Ryhmä tai aloitusvuosi** HETIKIM12
Opinnäytetyön nimi Paikkatietorajapintojen hyödyntäminen Case Finavia	**Sivu- ja liitesivumäärä** 43 + 20
Ohjaaja tai ohjaajat Niina Kinnunen	

Opinnäytetyön tarkoituksena oli tutkia keskeisiä The Open Geospatial Consortiumin (OGC) rajapintastandardeja ja niiden hyödyntämistä sekä selvittää yleisellä tasolla mitä paikkatieto on ja miten sitä voidaan hyödyntää. Lisäksi opinnäytetyössä rakennettiin OGC:n paikkatietorajapintoja hyödyntävä visuaalinen lentoesterekisteri-web-sovellus Finavian sisäiseen käyttöön.

Opinnäytetyön aihe tuli opinnäytetyön tekijän kiinnostuksesta ohjelmistokehitystä ja moderneja web-tekniikoita kohtaan sekä Finavian tarpeesta kehittää lentoestetietojen julkaisupalvelua käyttäjäystävällisempään suuntaan. Nykyisessä julkaisupalvelussa tietokannassa olevia lentoesteiden tietoja julkaistaan tekstimuodossa. Tämä julkaisumuoto vaatii aina palvelun käyttäjältä lisätyötä, jotta lentoestetiedot saadaan näkyville visuaaliseen muotoon esimerkiksi paikkatieto-ohjelmistoon.

Opinnäytetyön tavoitteena oli lentoestetietojen tehokkaampi, monipuolisempi ja käytännöllisempi esittämistapa. Opinnäytetyön hyötyjä kohdeyritykselle ovat joustavampi ja käyttäjäystävällisempi tapa esittää lentoestetietoja. Lisäksi se mahdollistaa erilaisia jatkokehitysmahdollisuuksia julkaisupalvelun kehittämiseksi tulevaisuudessa.

Opinnäytetyössä toteutettavassa lentoestetietoja visualisoivassa web-sovelluksessa luotiin virtuaaliselle tietokoneelle paikkatietorajapintoja tukeva tietokanta, karttapalvelin sekä web-sivu, joka näyttää kaikki lentoestetiedot graafisena näkymänä verkkoselaimella. Toteutuksessa hyödynnettiin pääasiallisesti avoimen lähdekoodin ohjelmistoja käyttöjärjestelmän, tietokannan ja karttapalvelimen osalta. Lentoesterekisteri-web-sovellus toteutettiin kesän ja syksyn 2014 aikana. Toimeksiantaja ja opinnäytetyön tekijä olivat lopputulokseen tyytyväisiä.

Asiasanat
Rajapinnat, paikkatiedot, avoin lähdekoodi, JavaScript

HAAGA-HELIA
University of Applied Sciences

Abstract

19 November 2014

Degree Programme in Business Information Technology

Author(s) Tero Tuomala	**Group or year of entry** HETIKIM12
The title of thesis Advantages of implementing geospatial interfaces Case Finavia	**Number of report pages and attachment pages** 43 + 20
Advisor(s) Niina Kinnunen	

The purpose of this thesis was to study central interface standards of The Open Geospatial Consortium (OGC) and their implementation and to examine in general what geospatial information is and how it could be used. In addition, a flight obstacle registry Web-application using OGC's interface standards for Finavia's in-house use was developed as part of this thesis.

The subject for this thesis stems from the author's interest in software development and modern Web-technologies and also from Finavia's need to develop a more user-friendly flight obstacle publishing service. The current publishing service method is to publish the information from database in text form. In order to see the information in visual form, for example in geographical information systems, the current publishing method requires additional work from the end-user.

The goal of this thesis was to provide a more effective, versatile and practical way of presenting flight obstacle information and for Finavia, a more flexible and user friendly way of presenting this information. Furthermore, this will also permit future development of their publishing service.

For the flight obstacle registry Web-application that was developed as part of this thesis, a database supporting geospatial interfaces, a map server and a Web-page showing all flight obstacle information as graphical representation in a Web-browser were set up on a virtual computer. In the implementation of the operating system, the database and the map server, open source software was primarily used. The flight obstacle registry Web-application was developed during the summer and fall of 2014. Both the client and the author were satisfied with the end result.

Key words
Interfaces, geographic information systems, open source, JavaScript

Sisällys

1 Johdanto

Tämän opinnäytetyön tarkoituksena on tutkia keskeisiä The Open Geospatial Consortiumin (OGC) rajapintastandardeja ja niiden hyödyntämistä sekä selvittää yleisellä tasolla mitä paikkatieto on ja miten sitä voidaan hyödyntää. Lisäksi tulen opinnäytetyössäni rakentamaan OGC:n paikkatietorajapintoja hyödyntävän lentoesterekisteri-web-sovelluksen Finavian sisäiseen käyttöön.

Opinnäytetyöni aihe tuli kiinnostuksestani ohjelmistokehitystä ja moderneja web-tekniikoita kohtaan sekä Finavian tarpeesta kehittää lentoestetietojen julkaisupalvelua käyttäjäystävällisempään suuntaan. Nykyisessä julkaisupalvelussa tietokannassa olevia lentoesteiden tietoja julkaistaan tekstimuodossa. Tämä julkaisumuoto vaatii aina palvelun käyttäjältä lisätyötä, jotta lentoestetiedot saadaan näkyville visuaaliseen muotoon esimerkiksi paikkatieto-ohjelmistoon.

Opinnäytetyön teoriaosuudessa käydään läpi OGC:n rajapintastandardeja, jotka on rajattu lentoesterekisteri-web-sovelluksen toteutuksen kannalta olennaisimpiin standardeihin. Lisäksi siinä tutustutaan paikkatietoon yleisellä tasolla, joka auttaa ymmärtämään paikkatiedon perusteita ja sen hyödyntämisen mahdollistavia asioita.

Opinnäytetyössäni toteutettavassa lentoestetietoja visualisoivassa web-sovelluksessa luodaan virtuaaliselle tietokoneelle paikkatietorajapintoja tukeva tietokanta, karttapalvelin sekä web-sivu, joka näyttää kaikki lentoestetiedot graafisena näkymänä verkkoselaimella. Toteutuksessa tullaan pääasiallisesti hyödyntämään avoimen lähdekoodin ohjelmistoja käyttöjärjestelmän, tietokannan ja karttapalvelimen osalta.

Opinnäytetyön tavoitteena on lentoestetietojen tehokkaampi, monipuolisempi ja käytännöllisempi esittämistapa. Opinnäytetyön hyötyjä kohdeyritykselle ovat joustavampi ja käyttäjäystävällisempi tapa esittää lentoestetietoja. Lisäksi se mahdollistaa erilaisia jatkokehitysmahdollisuuksia julkaisupalvelun kehittämiseksi tulevaisuudessa. Itselleni se puolestaan mahdollistaa syvällisemmän tutustumisen paikkatietorajapintoja hyödyntävän web-sovelluksen kehitykseen.

Opinnäytetyön lopputuloksena syntyy paikkatietorajapintoja hyödyntävä web-sovellus, jolla voidaan näyttää lentoestetietoja visuaalisessa muodossa.

2 Paikkatieto

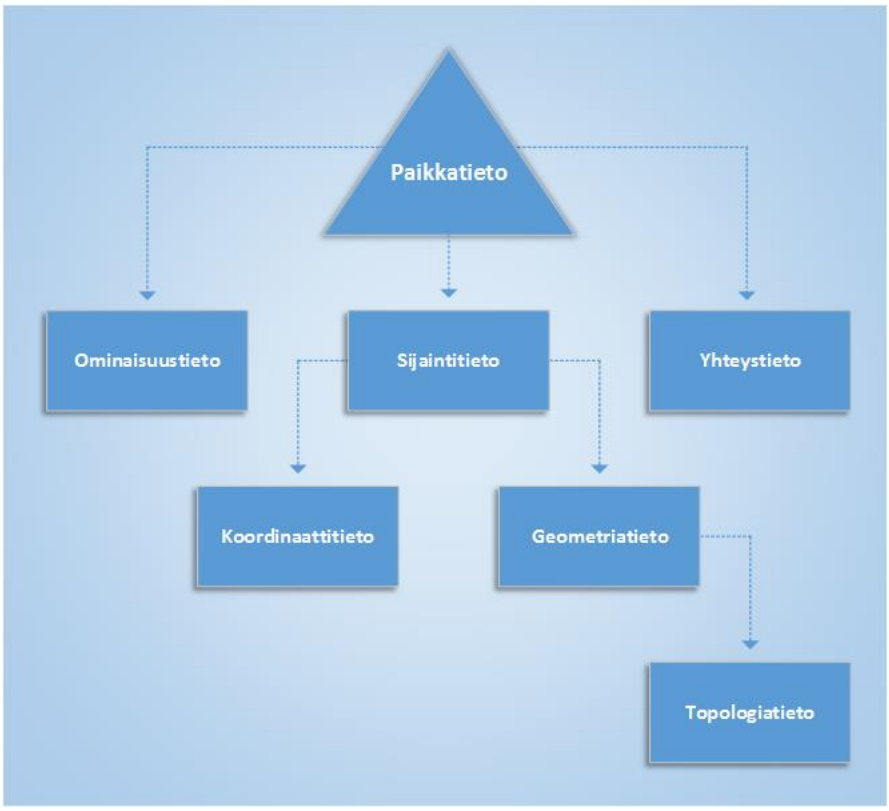
Paikkatiedolla ilmaistaan tietoja kohteista, joiden sijainti Maan suhteen tunnetaan. Paikkatietoon sisältyy aina viittaus johonkin tiettyyn paikkaan tai maantieteelliseen alueeseen. Paikkatieto voi myös ilmaista kohteen sijaintitiedon lisäksi muita ominaisuustietoja, kuten tietoja muodosta. Paikkatiedolla kuvataan usein luonnon tai rakennetun ympäristön kohteita, sillä voidaan myös kuvata ilmiöitä tai mitä tahansa toimintaa, jonka sijainti tiedetään. (Sanastokeskus 2014, 22.)

Paikkatiedolla pyritään vastaamaan jokapäiväisiin kysymyksiin liittyen sijaintiin, malleihin, trendeihin ja olosuhteisiin (Heywood, Cornelius & Carver 2011, 3) kuten esimerkiksi:

- *Sijainti*. Missä sijaitsee lähin ruokakauppa? Miltä metsäalueilta löytyy metsävaahteraa?
- *Mallit*. Millainen liikennevirta kulkee tällä moottoritiellä? Missä asuvat korkean keskittymiskyvyn omaavat opiskelijat tällä seudulla? Millainen rikostapauksien levinneisyys Helsingissä on?
- *Trendit*. Mihin harmaakarhujen kannan muutos on vaikuttanut? Minne jäätiköt vetäytyivät Pohjoisnavalta?
- *Olosuhteet*. Mistä löydän lomamajoituksen, joka on 2 kilometrin sisällä museosta, jonne pääsee julkisilla kulkuvälineillä? Missä ovat yli 50 000 potentiaalista asiakasta 10 kilometrin sisällä metroasemasta?
- *Vaikutukset*. Jos muutan tähän sijaintiin uuteen asuntoon, kuinka kaukana olen työpaikasta, kahvilasta tai kuntosalista? Millainen ajansäästö syntyisi jos kuljettaisimme tomaattimme tätä reittiä vanhan sijaan?

2.1 Paikkatiedon rakenne

Paikkatiedon erityisyys on sen sitominen tiettyyn ennalta määrättyyn koordinaatistoon. Itse sitominen tehdään joko antamalla suoraan tarkat koordinaatit tai epäsuorasti käyttämällä jonkinlaista tunnusta, minkä perusteella voidaan paikalle hakea koordinaatit. (Lepistö 2000.) Paikkatiedon tarkempi rakenne esitetään kuviossa 1.



Kuvio 1. Paikkatiedon rakenne (Lepistö 2000)

Paikkatiedon rakenteiden olennaisia käsitteitä ovat ominaisuustieto, sijaintitieto ja yhteystieto (Lepistö 2000).

Ominaisuustieto määrittelee paikkatiedon kohteita ja se sisältää tietoja kohteen erilaisista ominaisuuksista. Ominaisuustieto jaetaan neljään eri tyyppiin: yksilöivä tieto, ajoittava tieto, paikantava tieto ja kuvaileva tieto. (Lepistö 2000.)

Yksilöivä tieto on kaikista tarkinta ominaisuustietoa ja niitä voivat olla esimerkiksi nimet ja numerot. Ajoittavaa tietoa voivat olla esimerkiksi lämpötilat. Paikantavaa tietoa voi olla esimerkiksi asuntojen osoitteet sekä kuvailevaa tietoa voi olla esimerkiksi erilaiset värit. (Lepistö 2000.)

Sijaintitieto pitää sisällään tietoa kohteen sijainnista, geometriasta ja topologiasta. Sijaintitietoon sisältyy tavallisen koordinaattidatan lisäksi tietoa, kuinka tarkasti koordinaatit on ilmoitettu. (Lepistö 2000.)

Paikkatietoaineistot ovat geometrisessa mielessä yleensä pisteitä, viivoja, alueita, aluejakoja tai ruudustoja. Geometriatietoa voidaan ilmaista vektorigeometrisen tai rasterigeometrisen mallin mukaan. Geometriatieto määrittelee millaista yksilötyyppiä kohde on. Yksilötyyppejä voivat olla piste, viiva, alue ja hila-alkio. Näiden lisäksi sijaintitietoon voidaan myös sisällyttää topologiatietoa, joka kuvaa eri alueiden sijaintisuhteita toisiinsa nähden. (Lepistö 2000.)

Koordinaattitieto on tärkein ja ainoa tapa sijainnin ilmoittamiseen. Koordinaattitieto sisältää tarkan kohteen tarkan sijainnin maapallolla. Ilman koordinaattitietoa muulla kerätyllä tiedolla ei ole mitään merkitystä, koska kohteen tarkkaa sijaintia ei tiedetä. Tavallisesti kohteen sijainti ilmoitetaan paikan nimellä. Muita vaihtoehtoja sijaintitiedon ilmaisemiseen ovat esimerkiksi kiinteistö-, havaintopaikka- tai toimipaikkatunnus. (Lepistö 2000.)

Yhteystieto kuvaa kohteiden välisiä suhteita, joka perustuu todellisuuden kohteiden yksilöintiin. Suhteita kuvaava tieto on aina topologiatietoa eikä yhteystietoa useimmiten edes kerätä. (Lepistö 2000.)

Topologiatieto ilmaisee geometristen yksiköiden suhteita. Suhteet on esitetty suoraan eikä niitä tarvitse enää laskea koordinaattitiedosta. Tämänlaista asiaa kutsutaan myös eksplisiittiseksi spatiaaliseksi relaatioksi. (Lepistö 2000.)

Yksinkertainen esimerkki topologiasta on katuverkko. Katujen risteyksistä tehdään solmuja ja kerrotaan, mitkä kadunpätkät missäkin solmussa kohtaavat. Toinen yksinkertainen esimerkki on viiva, jossa pisteiden tallennusjärjestys on ilmaistu jollakin tavalla. (Lepistö 2000.)

2.2 Paikkatietojärjestelmä

Paikkatietojärjestelmällä (GIS = Geographic Information System) käytetään, kerätään ja ylläpidetään paikkaan sidottua tietoa. Paikkatietojärjestelmän avulla voidaan katsella ja yhdistellä erilaisia maantieteellisiä tietoja samanaikaisesti päällekkäisinä kerroksina.

Tämä mahdollistaa ympäröivän todellisuuden paremman hahmottamisen ja ymmärtämisen. (ESRI 2014a.)

Paikkatietojärjestelmä on teknologinen alue, jolla pyritään kartoittamaan, analysoimaan ja arvioimaan reaalimaailman ongelmia liittämällä yhteen maantieteelliset ominaisuudet taulukkomuodossa. Avainsana tähän teknologiaan on maantiede - tämä tarkoittaa sitä, että jokin osa tiedoista viittaa jollakin tavalla paikkoihin Maan päällä. (Dempsey Morais 2012.)

Fun ja Sunin (2011, 4) mukaan paikkatietojärjestelmä on kokoelma laitteistoa, ohjelmistoa ja menettelytapoja, joilla otetaan talteen, varastoidaan, käsitellään, muokataan, analysoidaan, hallitaan, jaetaan sekä näytetään georeferoituja tietoja.

Paikkatietojärjestelmää käyttämällä voidaan tietokoneella esittää karttoja sekä muuta maantieteellisiä dataa. Lisäksi sitä voidaan analysoida esimerkiksi mittaamalla etäisyyksiä tai etsimällä maantieteellisiä ominaisuuksia ja tuottamalla niistä materiaalia, kuten kuviossa 2 esitetään. (DeMers 2009, 10.)

Kuvio 2. Paikkatietojärjestelmän toiminnolliset komponentit (Huisman & de By 2009, 145)

Paikkatietojärjestelmä perustuu neljään peruskomponenttiin: Laitteisto, ohjelmisto, aineisto ja käyttäjät (Dempsey Morais 2012.) Kokonaisuus on esitelty kuviossa 3.

Kuvio 3. Paikkatietojärjestelmän kokonaisuus (DeMers 2009, 11)

Laitteisto käsittää välineet, joilla tuetaan eri toimintoja aina tiedon keräämisestä tiedon analysointiin. Paikkatietojärjestelmän keskipiste on tietokone, jolla käytetään paikkatieto-ohjelmistoa ja mahdollisia lisälaitteita. (Dempsey Morais 2012.)

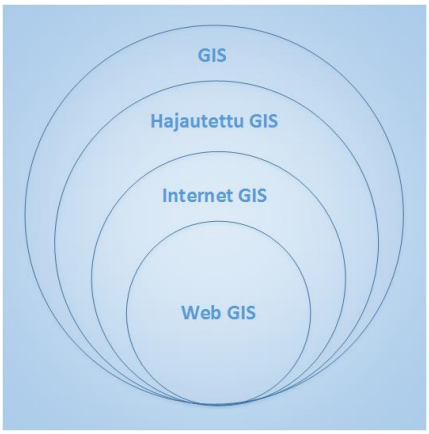
Ohjelmisto sisältää kaikki tarvittavat toiminnot ja työkalut sijaintitiedon esittämiseksi, analysoimiseksi ja varastoimiseksi (ESRI 2014a). Ohjelmistossa erilaisia kohteita esitetään ja käsitellään päällekkäisinä karttatasoina, kuten esimerkiksi rakennukset, tiestö, postinumeroalueet ja vesistö. Paikkatiedon ja analyysien tuloksien havainnollistaminen kartalla mahdollistaa erittäin hyvän tavan esittää asioita. (ESRI 2014c.)

Aineisto on koko paikkatietojärjestelmän ydin. Paikkatietojärjestelmän kaksi ensisijaista tietotyyppiä ovat vektori ja rasteri. Vektoridata on spatiaalista tietoa, joka esitetään pisteinä, viivoina ja monikulmioina kun taas rasteridata on solupohjaista tietoa, kuten esimerkiksi ilmakuvat ja digitaaliset korkeusmallit. (Dempsey Morais 2012.) Aineiston tulee olla laadultaan riittävän hyvää, jotta sen pohjalta tuotetut analyysit olisivat luotettavia. Paikkatietojärjestelmiin voidaan lisätä tietoja monista eri lähteistä, kuten: Taulukkomuotoinen data, kuvatiedostot, CAD-tiedostot ja relaatiotietokannat. (ESRI 2014a.)

Käyttäjät, kuten tietokannan luojat tai hallinnoijat, analyytikot, jotka työskentelevät ohjelmiston kanssa, sekä tuotteen loppukäyttäjät muodostavat myös olennaisen osan paikkatietojärjestelmää (Huisman & de By 2009, 43). Jotta organisaatio hyötyisi mahdollisimman tehokkaasti paikkatiedon käytöstä, vaatii se ammattitaitoiset käyttäjät (ESRI 2014a).

2.3 Terminologia

Neljä yleisimmin käytettyä muotoa paikkatietojärjestelmistä ovat: GIS, hajautettu GIS, Internet GIS ja Web GIS, kuten kuviossa 4 on esitetty (Fu & Sun 2011, 14.) GIS kattaa ylätasolla kaikki paikkatietojärjestelmään liittyvät asiat, mutta se ei määrittele sisältääkö paikkatietojärjestelmä ominaisuuksia hajautetusta GIS:stä, Internet GIS:stä tai Web GIS:stä (Peuralahti 2014, 15).



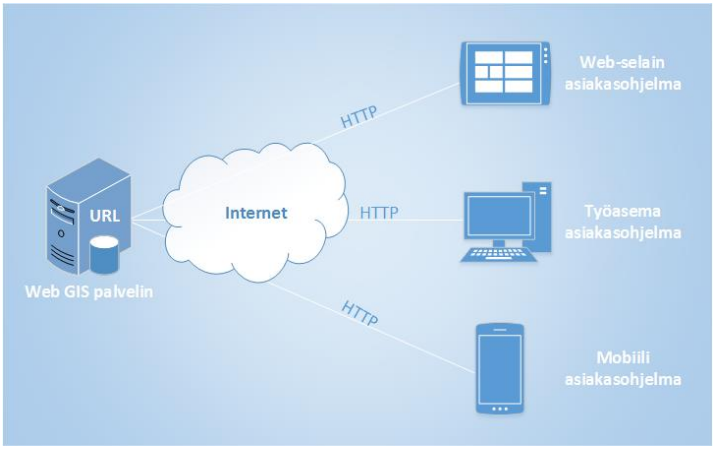
Kuvio 4. Paikkatietojärjestelmän muodot ja niiden suhteet toisiinsa (Fu & Sun 2011, 14)

Hajautettu GIS määrittelee GIS-komponentit, datan ja sovelluksen käytön, jotka voivat olla hajautettu esimerkiksi yrityksen sisäverkkoon (LAN) tai laajaverkkoon (WAN), kuitenkin siten, että ne eivät ole näkyvillä Internettiin (Peuralahti 2014, 15). Hajautettu GIS sisältää tyypillisesti peruskomponentteina asiakasohjelman, web-palvelimen, sovelluspalvelimen, tietokantapalvelimen ja yhden tai useamman GIS-palvelimen (Peng & Tsou 2003, 20.)

Internet GIS toimii viitekehyksenä verkkopohjaiselle paikkatietojärjestelmälle, joka käyttää Internetiä hyödyksi päästäkseen etäältä käsiksi maantieteellisin tietoihin ja geopro-

sessointi työkaluihin (Peng & Tsou 2003, 12). Internet GIS sisältää tavallisesti neljä pääkomponenttia, jotka ovat: asiakasohjelma, web-palvelin, joka sisältää sovelluspalvelimen, tietokantapalvelimen ja karttapalvelimen. (Peng & Tsou 2003, 20.)

Web GIS on tyypiltään hajautettu tietojärjestelmä, joka yksinkertaisimmassa muodossa sisältää asiakasohjelman ja palvelimen, jossa asiakasohjelma on verkkoselain, mobiilisovellus tai työasemasovellus ja palvelin on web-sovelluspalvelin. Palvelin ja asiakasohjelma kommunikoivat http-protokollan välityksellä, kuten kuviossa 5 esitetään. (Fu & Sun 2011, 13.)



Kuvio 5. Web GIS yksinkertaisimmassa muodossa sisältää web-sovelluspalvelimen ja asiakasohjelman (Fu & Sun 2011, 13)

Mikä tahansa paikkatietojärjestelmä joka käyttää web-teknologioita komponenttien väliseen kommunikointiin voidaan pitää Web GIS:nä (Fu & Sun 2011, 13).

Internet GIS:ä ja Web GIS:ä käytetään usein toistensa synonyymeinä, vaikka ne ovat hieman erilaisia keskenään. Internet tukee lukuisia erilaisia palveluita ja web on vain yksi niistä. Paikkatietojärjestelmä, joka käyttää mitä tahansa Internetin palvelua, eikä vain pelkkää web:iä, voidaan pitää Internet GIS:nä. (Fu & Sun 2011, 14.)

2.4 Hyödyntäminen

Perinteisesti paikkatietoja on hyödynnetty muun muassa liikenteen ja yhdyskuntahuollon verkostojen suunnittelussa, ympäristönsuojelussa sekä luonnonvarojen kartoituk-

sessa. Nykyään paikkatietoja käytetään kuitenkin hyväksi yhä enemmissä määrin lähestulkoon kaikilla yhteiskunnan osa-alueilla niin julkisella kuin yksityisellä sektorilla. (ESRI 2014b.)

Paikkatietoteknologioita käyttämällä voidaan auttaa erilaisia yhteiskunnan ja yritysten prosesseja: esimerkiksi kuluttajat voivat hyödyntää paikkatietopohjaisia laitteita ja palveluita kuten reittioppaat ja navigaattorit (Teknologiateollisuus 2013). Uusimpina paikkatietoteknologioiden hyödyntämiskohteina ovat markkinoiden kohdentaminen, kuljetusten suunnittelu ja optimointi, vakuutustoiminta sekä liikepaikka- ja palveluverkostojen suunnittelu (ESRI 2014b).

Analysoimalla ja tulkitsemalla erilaisia yhdisteltyjä tietoaineistoja voidaan tuottaa täysin uutta tietoa. Tällä tavalla voidaan esimerkiksi analysoida toiminnan ympäristövaikutuksia, selvittää paras sijainti uudelle kaupalle tai kartoittaa rikosten esiintymistä tietyllä kohdealueella niiden estämiseksi. (ESRI 2014a.)

Paikkatietoa voidaan hyödyntää myös seuraavasti:

Ominaisuuksien ja erityispiirteiden etsiminen. Paikkatietoja hyödynnetään tietynlaisten ominaisuuksien omaavien paikkojen etsimiseen tai tietyn paikan ominaisuuksien selvittämiseen (ESRI 2014b.)

Mallien luominen. Paikkatietoja hyödynnetään kartalla esiintyvän ilmiön spatiaalisen levinneisyyden tarkasteluun. Tutkimalla kaikkia esiintymäjoukon ominaisuuksia pystytään ilmiön ominaisuuksista luomaan malli. (ESRI 2014b.)

Määrätyt kriteerit täyttävien sijaintien löytäminen. Esimerkiksi uutta myymäläpaikkaa etsiessä lastentavaraliike voi määritellä alueet, joilla asuu paljon potentiaalisia lapsiperheitä. Vähittäiskaupassa sijainnin merkitys on erittäin tärkeää. (ESRI 2014b.)

Määritetyllä etäisyysvyöhykkeellä tapahtuvan toiminnan havaitseminen. Esimerkiksi ihmisten määrän havaitseminen 500 ja 100 metrin etäisyydellä uudesta moottoritiestä voidaan havaita kaupunkisuunnittelijan toimesta. (ESRI 2014b.)

Muutoksien mallintaminen kartalla. Paikkatietojärjestelmää voidaan hyödyntää muun muassa olosuhteiden ennakoimiseen tulevaisuudessa tai kohdealueella tietyn toiminnan tai politiikan vaikutusten arvioimiseen. (ESRI 2014b.)

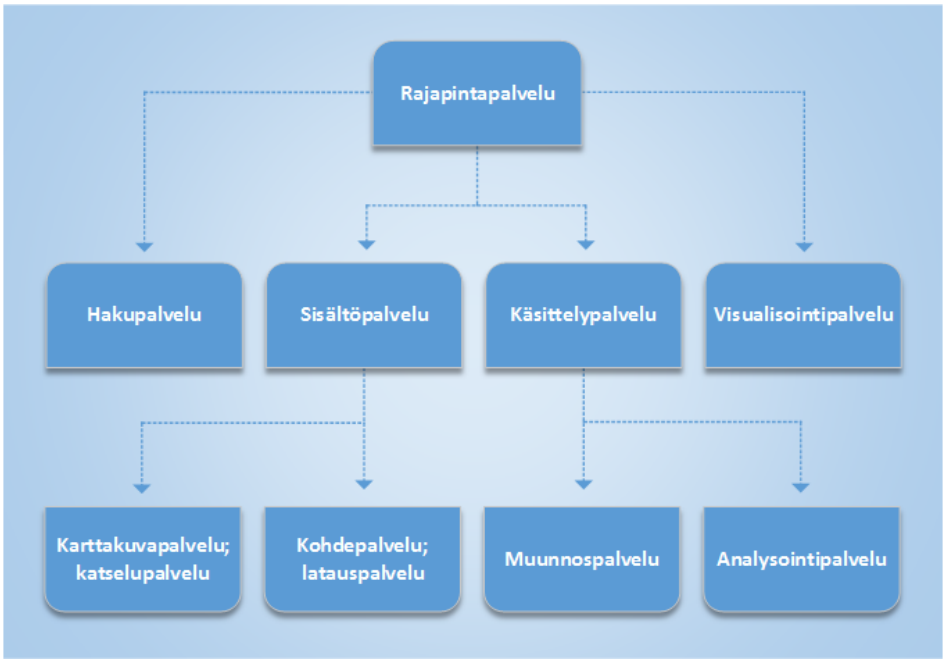
3 Rajapintapalvelut

Rajapintapalvelut mahdollistavat ajantasaisen tiedon saamisen suoraan tiedon tuottajalta, eikä sitä tarvitse hankkia itselle ja huolehtia sen ajantasaisuudesta (Maanmittauslaitos 2014). Rajapintapalveluita tarvitaan paikkatietopalveluiden toteuttamiseen, jotka on tarkoitettu ihmiskäyttäjille. Rajapintapalvelut mahdollistavat paikkatiedon esittämisen, käsittelemisen ja muuntamisen eri palvelimilla kuin missä tiedot ovat paikkatietopalveluissa. (Sanastokeskus 2014, 39.)

Rajapintapalvelu on tekninen käyttöyhteys, joka edellyttää että käytössä on ohjelmisto tai sovellus, jolla on mahdollista ottaa yhteys palveluntarjoajan palvelimelle. Yhteyden muodostumisen jälkeen voidaan määritellä mitä aineistoja tarvitaan ja tämän jälkeen ne ovat käytettävissä. (Maanmittauslaitos 2014.)

> Euroopan yhteisön paikkatietoinfrastruktuurin (INSPIRE) perustamisesta annetussa Euroopan parlamentin ja neuvoston direktiivissä 2007/2/EY tarkoitettuja rajapintapalveluita ovat hakupalvelu, katselupalvelu (eli karttakuvapalvelu), latauspalvelu (eli kohdepalvelu) ja muunnospalvelu (Sanastokeskus 2014, 39).

Rajapintapalvelut on esitelty kokonaisuudessaan kuviossa 6.



Kuvio 6. Rajapintapalvelut (Sanastokeskus 2014, 38)

Rajapintapalvelun keskeinen periaate on, että asiakassovelluksen ei tarvitse olla mitenkään tietoinen palvelun sisäisestä toteutustavasta. Ainoa asia mitä asiakassovellus näkee, on määritelty rajapinta, jonka mukaisilla käsitteillä se kommunikoi palvelun kanssa. Rajapintapalvelun tekninen toteutus on jopa mahdollista vaihtaa kauttaaltaan toiseksi rajapinnan takana, siten ettei asiakassovellus havaitse mitään muutosta. Jotta näin voidaan toimia, tulee asiakassovelluksen tukea määriteltyä rajapintaa. (Julkisen hallinnon tietohallinnon neuvottelukunta 2013b, 5.)

3.1 OGC:n rajapintastandardeja

The Open Geospatial Consortium (OGC) on voittoa tavoittelematon vuonna 1994 perustettu kansainvälinen konsortio. Siihen kuuluu yhteensä 474 yritystä, yliopistoa ja valtion virastoa, joiden pyrkimyksenä on kehittää julkisesti saatavilla olevia rajapintastandardeja. OGC:n standardit tukevat yhteensopivia ratkaisuja, jotka ”geomahdollistavat” webin, langattomat- ja paikkatietoon perustuvat palvelut ja valtavirta IT:n. (OGC 2014a.) OGC on julkaissut 35 hyväksyttyä standardia, jotka esitetään kuvissa 7.

Kuvio 7. OGC:n hyväksymät standardit

OGC-standardit ovat teknisiä dokumentteja, joilla määritetään käytettävä rajapinta tai koodaus, jolla maantieteellinen data ja palvelut voidaan sovittaa yhteen. Näitä OGC-standardointidokumentteja käytetään tukena rajapintojen ja koodausten käyttöönotossa, jotta voidaan kehittää alustariippumattoja tuotteita ja palveluita, jotka toimivat sijainnista riippumatta muiden tietolähteiden rinnalla ilman lisäprosessointia tai työtä. (Mimas 2013a.) Seuraavassa kerrotaan tarkemmin WMS-, WFS-, WCS-, WMTS- ja SLD-standardeista.

WMS (Web Map Service) on rajapinta, joka tarjoaa yksinkertaisen http-rajapinnan jolla voidaan pyytää karttakuvia yhdestä tai useammasta paikkatietoa sisältävästä tietokannasta. WMS-pyyntö määrittelee karttakerroksen tai karttakerrokset ja halutun alueen kartalta prosessointia varten. Vastauksena pyyntöön on yksi tai useampi karttakuva, joka voidaan esittää tietokoneen näytöllä ja sen formaattina on useimmiten PNG, JPEG tai GIF. (Karimi 2014, 282; Vehkaperä 2009.)

WMS-standardi (Vehkaperä 2009) määrittelee kolme operaatiota:

- *GetCapabilities* on pakollinen operaatio, joka palauttaa WMS-palvelun metatiedot. Tiedot sisältävät esimerkiksi palvelun tarjoaman tietosisällön, tuetut kuvaformaatit ja tuetut koordinaattijärjestelmät.
- *GetMap* on pakollinen operaatio, joka palauttaa karttakuvan.
- *GetFeatureInfo* on valinnainen operaatio, joka palauttaa lisätietoja karttakuvalla olevista kohteista. Standardissa määritellään eri parametrien käyttö kunkin operaation kutsussa sekä operaation vastauksen sisältö

WMS-palvelua tarjoavan tahon tulisi tarjota aineistostaan kyselyn mukainen karttakuva, jonka lisäksi sen tulisi toteuttaa GetCapabilities- ja GetMap-operaatiot. Yksinkertaisimmillaan WMS-palvelun käyttämiseen riittää verkkoselain, jolla voidaan katsoa vastauksena saatua kuvatiedostoa. (Vehkaperä 2009.)

WFS (Web Feature Service) on rajapintamäärittely, jonka avulla voidaan kysellä tarjolla olevien paikkatietokohteiden tietoja. Kyselyn vastauksena on siirtotiedosto, joka on

tavallisimmin GML-formaatissa (Geography Markup Language). Lisäksi myös muunlaiset formaatit kelpaavat, kuten esimerkiksi shape. (Vehkaperä 2009.)

WFS-standardi (Vehkaperä 2009) määrittelee pakollisina kolme operaatiota:

- *GetCapabilities* palauttaa muun muassa WFS-palvelun tarjoamat kohdetyypit ja sen tukemat operaatiot.
- *DescribeFeatureType* palauttaa WFS-palvelun tarjoamien kohdetyyppien rakenteen.
- *GetFeature* palauttaa pyynnössä esitetyt paikkatietokohteet, joita pyynnön esittäjän sovellus voi esimerkiksi piirtää halutulla kuvastekniikalla tai vaihtoehtoisesti tiedot voidaan tallentaa pyytäjän omaan tietovarastoon mahdollista myöhempää käyttöä varten.

Edellä mainittujen lisäksi WFS pitää sisällään operaatioita, jotka mahdollistavat palveluntarjoajan paikkatietokohteiden päivittämisen. WFS-standardi spesifioi mitä parametreja kussakin operaation kutsussa voidaan käyttää ja mitä operaation vastaus pitää sisällään. WFS-palvelun tarjoajan tulee laatia paikkatietokohteiden siirtotiedoston kuvaus (XML/skeema), joka ilmaisee mitä tietoa palvelusta on saatavilla ja mikä on siirrettävän tiedon rakenne. WFS-palveluun on lisäksi rakennettava välineet, joilla paikkatietokohteet kerätään tietovarastosta ja muunnetaan siirtotiedostoon. (Vehkaperä 2009.)

WCS (Web Coverage Service) määrittelee standardirajapinnan ja operaatiot, jotka mahdollistavat yhteen toimivan pääsyn maantieteellisille peittoalueille. Termillä *peittoalue* tyypillisesti tarkoitetaan digitaalisia ilmakuvia, digitaalisia korkeustietoja, satelliittikuvia ja muuta tietoa, joka voidaan ilmentää jollakin arvolla kullakin mittauspisteellä. Vastaus WCS-pyyntöön toimitetaan http-protokollan avulla, joka sisältää selostuksen metatiedoista ja tulosteista, joiden pikselit on koodattu johonkin tiettyyn kuvabinaarimuotoon kuten esimerkiksi NetCDF tai GeoTIFF. (Karimi 2014, 283.)

WCS-standardi (WCS 2012, 10) määrittelee kolme operaatiota:

- *GetCapabilities* palauttaa tietoa palvelun tarjoamista valmiuksista ja kattavuudesta.
- *DescribeCoverage* palauttaa kattavan ja yksityiskohtaisen metatiedon valitusta alueesta.

- *GetCoverage* palauttaa selostuksen, joko alkuperäisessä muodossa tai jossain soveltuvassa prosessoidussa muodossa

Edellä mainitut operaatiot spesifioivat yksinomaan palvelun teknisen rajapinnan. Mikäli palvelun tarjoajan tiedot ovat luvanvaraisia tai maksullisia, tulee niitä varten toteuttaa erilliset sovellukset. (Vehkaperä 2009.)

WMTS (Web Map Tile Service) on rajapinta, jonka avulla voidaan parantaa karttakuvapalvelun suorituskykyä käyttämällä etukäteen prosessoituja kiinteitä mittakaavatasoja ilmentäviä pieniä kuvapaloja, tiiliä (engl. ”tile”). Kuvapalat muodostavat koko kattavuusaluetta vastaavan yhtenäisen kuvan, joka mahdollistaa lähes reaaliaikaisen kartan vierityksen. (Julkisen hallinnon tietohallinnon neuvottelukunta 2013a, 12.)

WMTS-standardi (DigitalGlobe 2013, 13) määrittelee kolme operaatiota:

- *GetCapabilities* palauttaa tietoa saatavilla olevista karttatiili tyypeistä ja tuetuista operaatioista.
- *GetTile* palauttaa itse karttatiilen.
- *GetFeatureInfo* palauttaa metatiedot kaikista karttatiilistä, jotka on saatu GetTile pyynnön kautta.

WMTS-standardia sovelletaan yleensä ennemmin rasteri- kuin vektoridataan, jotta yhtäjaksoisen aineiston renderöinti olisi nopeampaa (Mimas 2013b). Tiili resurssi on yleensä suorakaiteen muotoinen kuva, joka sisältää kartografista tietoa. Vaihtoehtoisesti tämä resurssi voi olla ei-kuvallinen esitys tiilestä kuten esimerkiksi seloste tai linkki varsinaiseen kuvaan. (DigitalGlobe 2013, 5.)

SLD (Styled Layer Descriptor) on WMS-standardin rinnakkaisstandardi, joka mahdollistaa käyttäjän määrittämän maantieteellisten ominaisuuksien ja peittoalueen tiedon symboloinnin sekä värityksen. SLD ottaa huomioon käyttäjien ja sovellusten tarpeet, jotta ne voisivat hallita maantieteellisten tietojen visuaalista esitystapaa. Jotta näin voitaisiin toimi, vaatii se muotoilukielen jota niin asiakasohjelma kuin palvelin ymmärtävät. OGC:n SE (Symbology Enconding) standardi toimittaa tämän kielen, kun taas WMS:n

SLD-profiili mahdollistaa sovellusten symboliikan koodauksen WMS-karttatasoiksi käyttämällä WMS-operaatioiden laajennuksia. (OGC 2014b.)

SLD-spesifikaatiossa käytetään XSD (XML Schema Definition) kieltä, jolla määritellään mahdolliset elementit, jolla karttaa symbolisoidaan. Tärkein osa SLD:tä on *säännöt* kohta, jossa karttatason skaalan, täytevärin, viivan paksuuden tai läpinäkyvyyden tietoja voidaan määritellä. (Rumor, Coors, Fendel & Zlatanova 2007, 135.)

SLD-tiedosto sisältää seuraavanlaisen hierarkkisen rakenteen: ylätunniste, kohdetyypin tyyli, säännöt ja symboloinnit (Giuliani ym. 2014, 55).

Ylätunniste sisältää metatietoa XML-nimiavaruudesta ja on usein identtinen muiden eri SLD-tiedostojen kanssa (Giuliani ym. 2014, 55).

Kohdetyypin tyyli, (FeatureTypeStyle) on ryhmä tyylisääntöjä. Ryhmiteltäessä kohdetyypin tyylien mukaan, vaikuttaa se renderöinti järjestykseen; ensimmäinen kohdetyypin tyyli renderöidään ensin, jonka jälkeen renderöidään toinen ja niin edelleen. (Giuliani ym. 2014, 55.)

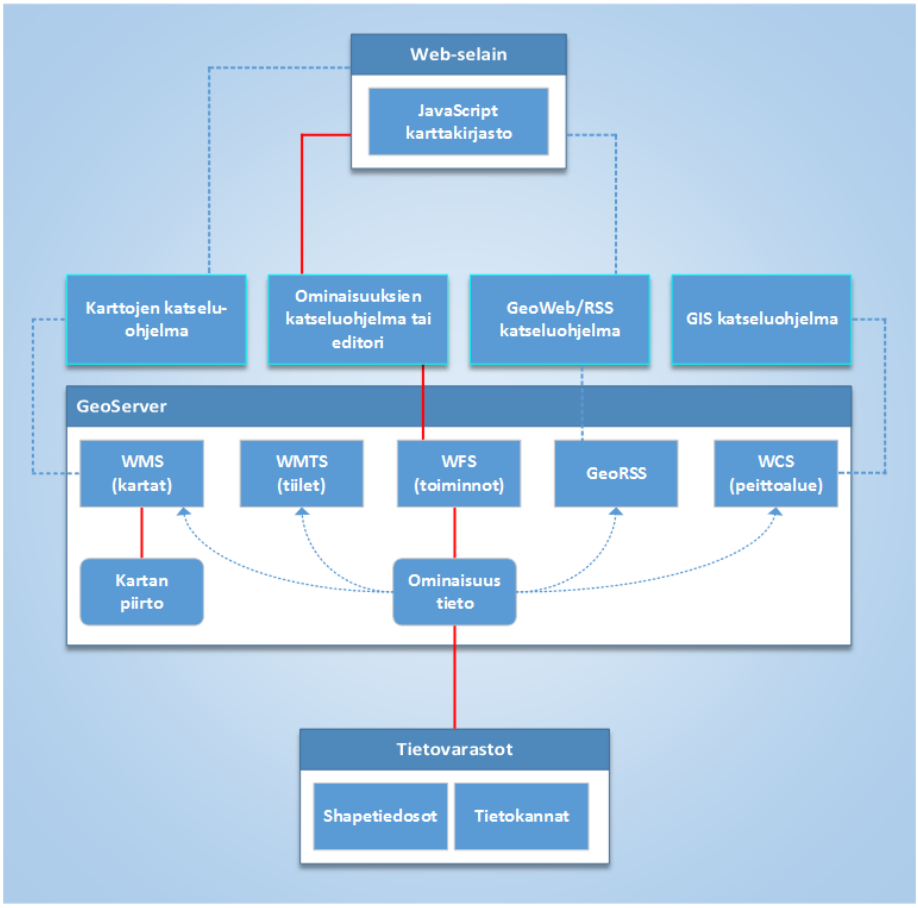
Sääntö (rule) on yksittäinen tyylisääntö, joka voi vaikuttaa globaalisti koko tasoon (layer) tai siihen voi liittyä suodatusta, jolloin sääntöä käytetään ehdollisesti. Nämä ehdot voivat perustua renderöintiin käytetyn tiedon attribuutteihin tai sen tarkkuuden tasoon. (Giuliani ym. 2014, 55.)

Symbolisoija (symbolizer) on varsinainen tyylimääritys. On olemassa viisi erilaista symbolisoijaa, jotka ovat: pistesymboloija, viivasymboloija, suorakulmasymboloija ja rasterisymboloija. (Giuliani ym. 2014, 55.)

3.2 Hyödyntäminen

OGC:n rajapintastandardeja voidaan hyödyntää sekä työasema- että web-sovelluksissa, kuvion 8 mukaisesti. Palvelut voidaan rakentaa esimerkiksi: WMS-, WMTS-, WFS- ja

WCS-rajapintojen avulla. Lisäksi palveluissa tarjottavia tietoja voidaan visualisoida, analysoida ja päivittää käyttöliittymien avulla. (Kylmäaho 2011.)



Kuvio 8. OGC:n rajapintastandardeja hyödyntävä web-sovellus (Wikipedia 2014)

Erilaisia avoimia OGC-standardien mukaisia palveluita tarjoavat esimerkiksi: kallioperä- ja maaperäaineistojen osalta geologian tutkimuskeskus, rataverkon osalta liikennevirasto ja vesistötietojen osalta Suomen ympäristökeskus (Kylmäaho 2011).

3.3 Käsittely JavaScript-kirjastoilla

OpenLayers on ilmainen, avoimen lähdekoodin JavaScript-kirjasto, joka on julkaistu BSD-lisenssin alaisuudessa, joka mahdollistaa niin kaupallisen, kuin ei kaupallisen käytön. OpenLayers-kirjastolla voidaan esittää karttadataa useimmissa moderneissa verkkoselaimissa ilman palvelinpuolen riippuvaisuuksia. (OSGeo 2014.)

OpenLayers toteuttaa vielä kehittyvällä JavaScript API:lla, rikkaita verkkopohjaisia paikkatietosovelluksia. OpenLayers-kirjasto on kirjoitettu olioperustaisella JavaScript-kielellä käyttämällä komponentteja Prototype.js- ja Rico-kirjastoista. Lisäksi OpenLayers toteuttaa standardimenetelmiä maantieteellisten tietojen saatavuuden osalta, kuten esimerkiksi OGC:n rajapintastandardeja Web Mapping Service (WMS) ja Web Feature Service (WFS) protokollia. (OSGeo 2014.) OpenLayersin versio 3.0 julkaisiin 29.8.2014 (OpenLayers 29.8.2014) ja se on tällä hetkellä uusin versio.

Keskeisin osa OpenLayersia on kartta (map), joka pitää huolen muun muassa tasoista (layers), ohjaimista (controls), peitteistä (overlays) ja tavasta miten se visualisoidaan. Visualisointi tehdään käyttämällä näkymää (view), joka on kuin ikkuna, josta kartta nähdään. Lisäksi näkymä käsite mahdollistaa esimerkiksi saman kartan renderöimisen eri näkymistä (keskitetty eri paikoista), saman kartan renderöinnin 2D- ja 3D-näkymässä tai eri karttojen renderöinti käyttämällä samaa näkymää. (Santiago 2014.)

Jotta kartta voidaan lisätä verkkosivulle (OpenLayers 2014), vaatii se kolme asiaa:

1. OpenLayers JavaScript-kirjaston sisällyttäminen koodiin.

```
<script src="http://openlayers.org/en/v3.0.0/build/ol.js"
```

2. Kartan sisällyttämisen HTML-elementtiin.

```
<div id="map" class="map"></div>
```

3. JavaScript-koodia, jolla karttaobjekti luodaan.

```
var map = new ol.Map({
        target: 'map',
        layers: [
          new ol.layer.Tile({
            source: new ol.source.MapQuest({layer: 'sat'})
          })
        ],
        view: new ol.View({
          center: ol.proj.transform([23.117294, 59.850082],
'EPSG:4326', 'EPSG:3857'),
          zoom: 5
        })
      });
```

Tällä JavaScript-koodilla luodaan karttaobjekti, joka käyttää tiilimuotoista MapQuestin Open Aerial tasoa ja näkymänä Suomen etelärannikon koordinaatteja (OpenLayers 2014).

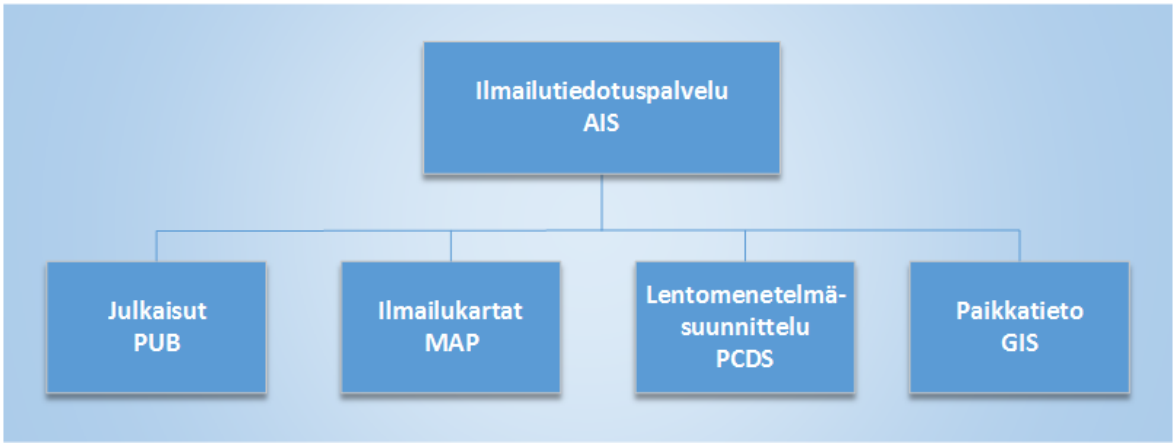
4 Paikkatietorajapintoja hyödyntävä visuaalinen lentoesterekisteri-web-sovellus

4.1 Toimeksiantajan esittely ja projektisuunnitelma

Opinnäytetyöni toimeksiantajana toimii Finavia Oyj:n Ilmailutiedotuspalvelu (Aeronautical Information Service - AIS) -yksikkö. Ilmailutiedotuspalvelun tarkoituksena on varmistaa kansainvälisen ja kansallisen ilmaliikenteen turvallisuuden, säännöllisyyden ja tehokkuuden kannalta tarpeellisen tiedon kulku sekä tukea uusia toimintaratkaisuja eurooppalaisessa ilmaliikenteen hallintaverkossa.

Ilmailutiedotuspalvelu on vastuussa ilmailutietojen kokoamisesta ja jakelusta koko Suomen valtakunnan alueella mukaan lukien sitä ympäröivien kansainvälisten vesialueiden yläpuolella oleva ilmatila, joka on Suomen lentotiedotusalueen rajojen sisällä.

AIS-yksikkö koostuu neljästä toimintasektorista kuvion 9 mukaisesti, jotka tekevät tiivistä yhteistyötä.



Kuvio 9. Ilmailutiedotuspalvelu organisaatiokaavio

Lentoesterekisteri web-sovelluksen projektisuunnitelmassa (liite 1) tavoitteeksi asetettiin paikkatietorajapintoja hyödyntävä web-sovellus, jolla olisi seuraavat toiminnollisuudet:

- Lentoesterekisteri-tietokannassa olevien tietojen näyttäminen pistemuotoisena karttapohjalla
- Lentoesteiden tietojen tarkastaminen kartalla olevista pisteistä
- Mahdollisuus käyttää taustakarttana Kapsi Internet-käyttäjät ry:n (kapsi) tarjoaman WMS-palvelun peruskarttarasteria, taustakarttaa ja ortoilmakuvakarttaa
- Mahdollisuus siirtyä kartalla mille tahansa Finavian hallinnoimalle lentokentälle pikavalinnasta
- Mahdollisuus suodattaa kartalle lentoesteitä niiden estetyypin mukaan
- Mahdollisuus etsiä lentoesteitä id-numeron perusteella.

Projektisuunnitelmassa määriteltiin alustavaksi aikatauluksi viikot 18-37, joiden välillä suunnittelu ja toteutus tehtiin. Lisäksi projektisuunnitelmassa määriteltiin toteutuksen vaiheet seuraavanlaisiksi: sovelluksen toiminnollisuuksien määrittely, teknisen ympäristön suunnittelu, tietokannan suunnittelu, teknisen ympäristön toteuttaminen, tietokannan toteuttaminen, web-sovelluksen suunnittelu ja web-sovelluksen toteuttaminen.

4.2 Toteutus

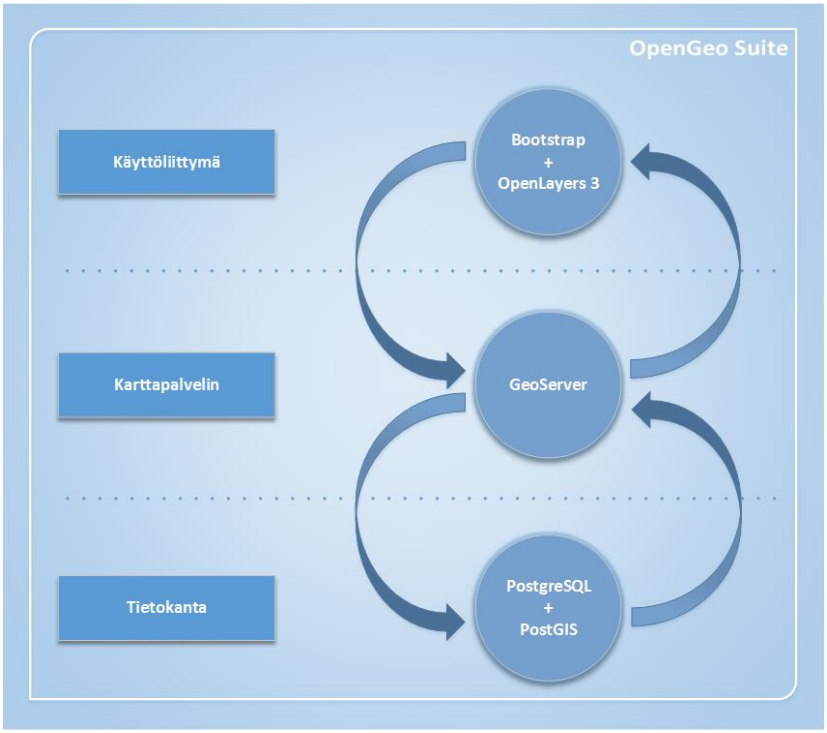
Lentoesterekisteri-web-sovelluksen lähtökohtana oli rakentaa web-sovellus, joka hyödyntää olemassa olevan lentoesterekisteri-tietokannan tietoja siten, että ne kopioidaan ja konvertoidaan säännöllisesti toiseen tietokantaan, josta web-sovellus voi niitä käyttää. Tietokannan tietojen käyttämistä varten tarkoituksena oli asentaa ja konfiguroida karttapalvelin, josta lentoestetietoja voitaisiin tarjota OGC:n paikkatietorajapintoja hyödyntäen web-sovellukselle.

4.2.1 Tekninen ympäristö

Teknisen ympäristön rakentamisessa pyrittiin suosimaan avoimen lähdekoodin ohjelmistoja kaikilla osa-alueilla. Sovellusta varten tarvittiin: tietokantapalvelin, karttapalvelin ja loppukäyttäjälle tarkoitettu web-sivu. Tämän kaltaista kokonaisuutta varten oli olemassa valmis ratkaisu nimeltään OpenGeo Suite. Kyseinen tuote oli ilmainen (basic versio) ja se sisälsi avoimen lähdekoodin ohjelmistoja, jonka lisäksi se tuki useita eri käyttöjärjestelmiä kuten Windows, Mac OS X, Red Hat/CentOS/Fedora ja Ubuntu.

OpenGeo Suite sisälsi seuraavat komponentit yhteen pakattuna ja helposti asennettavaan kokonaisuuteen: PostgreSQL ja PostGIS-laajennos, GeoServer, GeoWebCache, OpenLayers 3, SDK-työkalu, dokumentaatio ja esimerkkisovelluksia (Bootstrap + OpenLayers 3).

Tämä kokonaisuus sopi erinomaisesti lentoesterekisteri-web-sovelluksen vaatimia käyttötarpeita varten. OpenGeo Suiten teknisen ympäristön kuvaus löytyy kokonaisuudessaan kuviosta 10.



Kuvio 10. OpenGeo Suiten teknisen ympäristön kuvaus

Lähtökohtana oli hyödyntää avoimen lähdekoodin ohjelmistoja, joten käyttöjärjestelmäksi valittiin Linux Ubuntu 12.04.5 LTS Server, joka oli OpenGeo Suiten viimeisin tuotettu versio Ubuntusta. Perusteluina valinnalle oli käyttöjärjestelmän keveys (ei graafista käyttöliittymää), kustomoinnin mahdollisuus ja ilmaisuus.

Perusteluina PostgreSQL-tietokannan valinnalle olivat sen ilmaisuus, vähäinen resurssien kulutus, tuki OGC:n rajapintastandardeille ja hyvä dokumentaatio. Samat perustelut pätivät myös GeoServerin valinnalle karttapalvelimeksi.

Nykyinen lentoesterekisterin käytössä oleva tietokanta oli Oracle-pohjainen ja sitä ei voitu hyödyntää sellaisenaan, koska tiedot eivät olleet kannassa sellaisessa muodossa, että geometrisiä tietoja olisi voitu hyödyntää. Tämän lisäksi Oraclen ilmainen paikkatietoja tukeva laajennus (Oracle Locator) ei sisältänyt kaikkia toimintoja mitä tulevaisuudessa tultaisiin tarvitsemaan. Laajemman paikkatiedon tuen tarjoava lisäosta (Oracle Spatial) käyttäminen vaatisi lisenssin, joka taas olisi ollut kustannuksiltaan liian suuri.

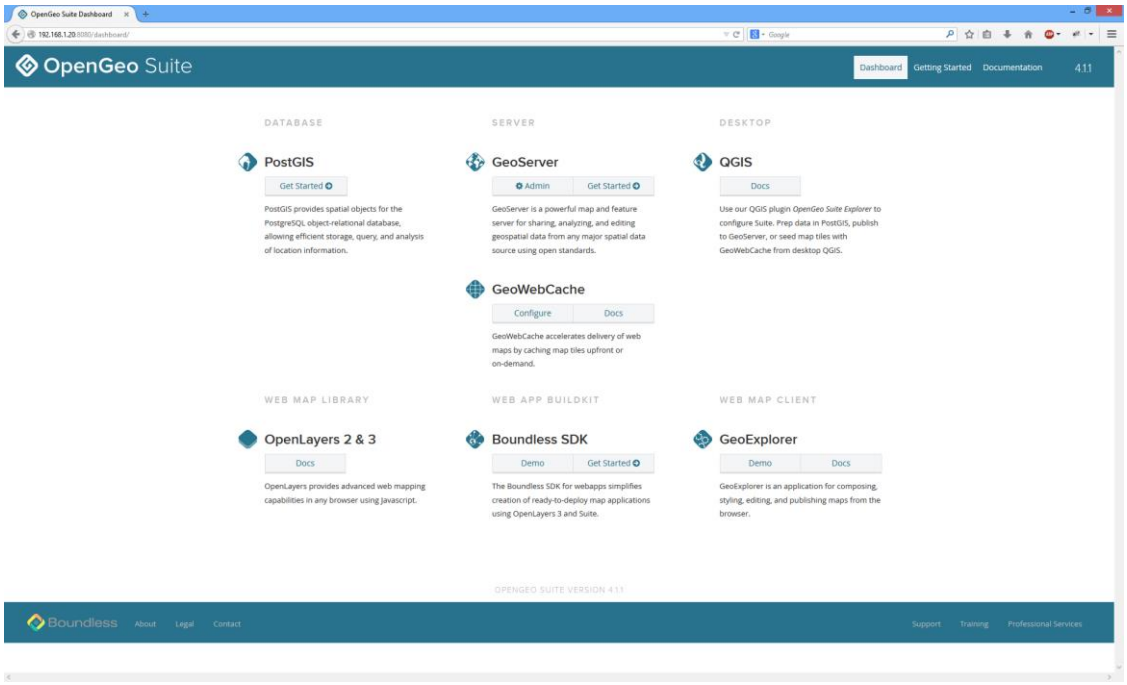
Linux Ubuntu 12.04.5 LTS Server käyttöjärjestelmän asentaminen tehtiin virtuaaliselle koneelle, jolle oli allokoitu resursseiksi 1 prosessori 4 ytimellä, 4 GT muistia ja 20 GT levytilaa. Asennuksessa valittiin kielisyydeksi englanti ja sijainniksi Suomi. Muiden asetusten osalta asennus tapahtui oletusasetuksilla.

Käyttöjärjestelmän asennuksen jälkeen koneelle määriteltiin kiinteä IP-osoite (192.168.1.20) ja asennettiin SSH-palvelin, jonka jälkeen konetta päästiin hallitsemaan Windows koneelta PuTTY-ohjelman kautta.

OpenGeo Suitesta asennettiin kaikki komponentit mitkä kuuluivat asennuspakettiin. Asennus tehtiin seuraavilla komennoilla Ubuntu palvelimen komentokehotteesta:

```
sudo su -
wget -qO- http://apt.opengeo.org/gpg.key | apt-key add -
echo "deb http://apt.opengeo.org/suite/v4/ubuntu/ precise main" >
/etc/apt/sources.list.d/opengeo.list
apt-get update
apt-get install opengeo -y
```

Asennuksen jälkeen testattiin sen toimivuus menemällä selaimella osoitteeseen: http://192.168.1.20:8080/dashboard/, josta päästiin OpenGeo Suiten ”kojelauta” näkymään.



Kuvio 11. OpenGeo Suite Dashboard-sivu

Sivuston kautta voitiin helposti navigoida eri komponenttien hallintasivuille ja dokumentaatioon.

4.2.2 Tietokanta

PostgreSQL ja PostGIS asentuivat OpenGeo Suiten asennuksen yhteydessä, joten seuraavaksi määriteltiin komentokehotteesta PostgreSQL:n asetukset siten, että PostgreSQL-tietokantaan sai otettua yhteyden myös toisilta koneilta. Tämä tapahtui muokkaamalla ensin tiedostoa:

```
/etc/postgresql/9.3/main/postgresql.conf
```

ja lisäämällä riville ”listen_address” palvelimen IP-osoite.

```
listen_address = '192.168.1.20'
```

Lisäksi määriteltiin pääsy tietokantaan 192.168.1.0/24 aliverkon osoitteista muokkaamalla tiedostoa:

```
/etc/postgresql/9.3/main/pg_hba.conf
```

ja lisäämällä riville ”host” aliverkon määritys.

```
host all all 192.168.1.0/24 md5
```

Tämän jälkeen postgresql palvelu käynnistettiin uudelleen komentokehotteesta komennolla:

```
service postgresql restart
```

Seuraavaksi otettiin yhteys tietokantaan komentokehotteesta psql-työkalua hyödyntäen ja vaihdettiin PostgreSQL-hallintatunnuksen salasana. Lisäksi luotiin uusi käyttäjä, uusi tietokanta sekä otettiin PostGIS-laajennus käyttöön. Nämä tapahtuivat komennoilla:

```
sudo -u postgres psql
alter user postgres password 'XXXXXXXXXXX';
create user XXX createdb createuser password 'XXXXXXXXXXX';
create database lentoesteet owner XXX;
\q
psql -U XXX lentoesteet
create extension postgis;
```

Tietokanta oli perusasetuksiltaan tämän jälkeen valmis ja seuraavaksi käytiin läpi mitä kaikkia tietoja nykyisestä Oracle-esterekisteritietokannasta tarvittaisiin PostgreSQL-tietokantaan, koska Oracle-tietokanta sisälsi myös paljon sellaisia tietoja, mitä lentoesterekisteri-web-sovelluksessa ei tultaisi käyttämään. Läpikäynnin jälkeen luotiin relaatiokaavio (liite 2), jossa lentoesteet-niminen taulu toimi päätauluna ja siihen liittyi relaatioita tyyppi-, ryhmä-, evlaji- ja evtoim-nimisistä tauluista. Seuraavaksi toteutettiin taulujen luominen PostgreSQL-tietokantaan. Taulujen luontilauseet koottiin SQL-tiedostoon Create_Tables.sql (liite 3).

Lentoesteet-tauluun oli relaatiokaavion mukaisesti valittu vain ne sarakkeet, joita tarvittiin. Tärkeimpänä uutena sarakkeena verrattuna vanhaan oli ”geom”, jossa määriteltiin PostGIS-laajennuksen käyttämä geometriatieto. Geometriaksi määriteltiin piste (point) muotoiseksi ja EPSG-koodiksi 4258. Nämä määritykset valittiin, koska tietoa haluttiin käyttää pistemuotoisena ja Oracle-tietokannassa oli käytetty ETRS89-koordinaattijärjestelmää, jonka EPSG-koodia 4258 vastasi. Lisäksi sarakkeet

”wgs84_n” ja ”wgs84_e” olivat Oraclessa ”number” tyyppisiä ja konversion yhteydessä PostgreSQL muunsi ne ”bigint”-muotoon, jolloin desimaalit jäivät pois. Ratkaisuna tähän oli asettaa ne ”double precision”-muotoon, jolloin desimaalit säilyivät.

```
SET client_encoding TO 'LATIN1';
\set ON_ERROR_STOP ON
CREATE TABLE lentoesteet (
            nimi varchar(40),
            tyyppi varchar(8),
            id bigint,
            wgs84_n double precision,
            wgs84_e double precision,
            agl_m_m double precision,
            msl_m_m double precision,
            diaari varchar(40),
            huom varchar(4000),
            vaikutus varchar(40),
            pallot varchar(1),
            maalaus varchar(1),
            ryhma varchar(8),
            luonti_pvm timestamp,
            muutos_pvm timestamp,
            valmis varchar(1),
            valmis_pvm timestamp,
            voimassa_pvm timestamp,
            poisto_pvm timestamp,
            evlaji1 varchar(8),
            evtoim1 varchar(8),
            geom geometry(Point,4258)
```

Kaikki tarvittavat taulut ja niiden väliset relaatiot luotiin kerralla ajamalla komentokehotteessa komennot:

```
psql -U XXX lentoesteet
lentoesteet < Create_Tables.sql
```

Tämän jälkeen PostgreSQL-tietokanta oli valmis tietojen lisäämistä varten Oracle-tietokannasta. Tätä toimenpidettä varten käytettiin ilmaista avoimen lähdekoodin ohjelmaa nimeltä Ora2pg. Ohjelman asentamiseen liittyi useita eri vaiheita, jonka lisäksi

asennettiin myös muita pieniä ohjelmia, jotta yhteys molempiin tietokantoihin saatiin muodostettua. Asennukset on kuvattu kokonaisuudessaan liitteessä 4.

Asentamisen jälkeen muokattiin Ora2Pg-työkalun konfiguraatioita, joka löytyi hakemistosta:

```
/etc/ora2pg/ora2pg.conf
```

Konfiguraatioissa määriteltiin miten ja mitä tietoja tietokantojen välillä siirrettiin. Lisäksi konfiguraatioissa voitiin määritellä todella paljon erilaisia asetuksia, joista oleellisimmat olivat:

```
ORACLE_HOME /usr/lib/oracle/11.2/client64
ORACLE_DSN dbi:Oracle:XXXXX
ORACLE_USER XXXX
ORACLE_PWD XXXX
TYPE INSERT
ALLOW LENTOESTEET ESTETYYPPI RYHMA EVLAJI EVTOIM
MODIFY_STRUCT LENTOESTEET(nimi,tyyppi,id,wgs84_n,wgs84_e,
agl_m_m,msl_m_m,diaari,huom,vaikutus,pallot,maalaus,ryhma,luonti_pvm,m
uutos_pvm,valmis,valmis_pvm,voimassa_pvm,poisto_pvm,evlaji1,evtoim1)
PG_DSN dbi:Pg:dbname=lentoesteet;host=192.168.1.20;port=5432
PG_USER XXXXX
PG_PWD XXXXX
TRUNCATE TABLE 1
```

- ORACLE, joissa määriteltiin Oracle Clientin asennushakemisto, lähde tietokannatapalvelimen tiedot, käyttäjätunnus ja salasana.
- TYPE, jossa määriteltiin minkä tyyppinen toiminto suoritetaan (INSERT), esimerkiksi UPDATE komentoa ei ollut käytettävissä.
- ALLOW, jossa määriteltiin vain ne taulut, joita käytetään.
- MODIFY_STRUCT, jossa määriteltiin mitkä kaikki sarakkeet lähde tietokannan taulusta kopioidaan kohde tietokannan vastaavaan tauluun.
- PG, joissa määriteltiin kohde tietokantapalvelimen tiedot, käyttäjätunnus ja salasana.

- TRUNCATE_TABLE, jolla määriteltiin kohde tietokannan taulujen poistaminen ja uudelleen luominen ilman tietueita ennen kuin uudet tiedot kopioidaan kantaan.

Konfiguraatio muutosten jälkeen tietojen siirto ajettiin komentokehotteesta komennolla:

```
sudo ora2pg -d -c ora2pg.conf
```

Ensimmäisten tietojen siirtojen aikana ilmeni ongelmia ”truncate_table” toiminnon kanssa, joka ilmeni virheilmoituksena:

```
DBD::Pg::db do failed: ERROR:  cannot truncate a table referenced in a
foreign key constraint DETAIL:  Table "lentoesteet" references "es-
tetyyppi".
HINT:  Truncate table "lentoesteet" at the same time, or use TRUNCATE
... CASCADE. at /usr/local/share/perl/5.18.2/Ora2Pg.pm line 3554.
```

Tämä johtui siitä, että ”lentoesteet” taulusta viitattiin viiteavaimella ”estetyyppi”-tauluun, joka taas esti ”estetyyppi”-taulun poistamisen ja uudelleen luomisen. Ongelma korjattiin ohjelman antaman vinkin mukaan muokkaamalla sen tiedostoa:

```
/usr/local/share/perl/5.18.2/Ora2Pg.pm
```

ja lisäämällä tiedoston riville 3554 loppuun ”CASCADE”-parametri:

```
my $s = $self->{dbhdest}->do("TRUNCATE TABLE $tmptb CASCADE;")
```

Tämän lisäyksen jälkeen muiden kuin ”lentoesteet”-taulun poistamisen ja uudelleenluonnin yhteydessä poistui myös niihin viitekehyksellä viittaava taulu ”lentoesteet”. Ora2Pg-ohjelma toimi siten, että taulujen poistaminen ja uudelleenluonti tapahtui taulujen nimien mukaan aakkosjärjestyksessä. Käytännössä tämä tarkoitti, että kun ”ryhma”-taulu poistettiin viimeisenä, poistettiin ja uudelleenluotiin samalla myös ”lentoesteet”-taulu, jolloin se jäi aina tyhjäksi.

Ongelma ratkaistiin muokkaamalla jälleen Ora2Pg-ohjelman lähdekoodia siten, että kun tauluja alettiin poistamaan ja uudelleenluomaan, tehtiin se siten, että ”lentoesteet”-

taululle operaatio tehtiin aina viimeisenä, jolloin tiedot säilyivät operaation jälkeen myös ”lentoesteet”-taulussa. Tämä tapahtui muokkaamalla tiedostoa:

```
/usr/local/share/perl/5.18.2/Ora2Pg.pm
```

lisäämällä tiedoston riveille 3318 ja 3370 samat ehtolausekkeet:

```
my @ordered_tables = sort {
   if ($a eq 'LENTOESTEET') { return 1; }
   elsif ($b eq 'LENTOESTEET') { return -1; }
   else { return $a cmp $b; }
   } keys %{$self->{tables}};
```

Viimeinen muutos tietokantaan oli geometria sarakkeen (geom) päivittäminen, koska itse tietojen siirrossa kyseiseen kenttään ei tuotu dataa. Tämä tehtiin siten, että ”geom”-sarakkeen sisältö muodostettiin ”wgs84_e” ja ”wgs84_n”-sarakkeiden arvojen perusteella ja asetettiin pistemäiseen muotoon sekä EPSG-koodiksi 4258. Tämä tapahtui komentokehotteesta psql-työkalua käyttäen antamalla komento:

```
UPDATE xxx.lentoesteet SET geom = ST_SetSRID(ST_MakePoint(WGS84_E,
WGS84 N), 4258);
```

Koska tietojen siirto haluttiin käytännön syistä johtuen automatisoida, luotiin geometria-sarakkeen (geom) automaattista päivittämistä varten tietokantaan funktio (function) ja laukaisin (trigger), joiden avulla päivitys tehtiin automaattisesti, kun ”lentoesteet”-tauluun oli lisätty tietoa. Funktion luominen tapahtui antamalla komentokehotteessa psql-työkalua käyttäen komento:

```
CREATE OR REPLACE FUNCTION update_geom_function() RETURNS TRIGGER AS
$update_geom$
            DECLARE
            row_count bigint;
            BEGIN
                        IF pg_trigger_depth() <> 1 THEN
                                    RETURN NEW;
                        END IF;
                        SELECT COUNT(*) INTO row_count FROM xxx.
                        lentoesteet;

                        IF row_count > 22400 THEN
                        UPDATE xxx.lentoesteet SET geom =
                        ST_SetSRID(ST_MakePoint(WGS84_E, WGS84_N), 4258)
                        WHERE geom IS NULL;
                        END IF;
                        RETURN NULL;
            END;
$update_geom$ LANGUAGE plpgsql;
```

Funktiossa ensimmäiseksi varmistettiin, että laukaisin ei jää silmukkaan, kun sarakkeen tietoja päivitetään. Lisäksi laskettiin kuinka monta riviä ”lentoesteet”-taulussa oli ja päivitys aloitettiin vasta, kun rivien määrä oli suurempi kuin tietty lukumäärä. Tämä asetettiin, koska päivitys olisi ollut todella hidas, mikäli se olisi ajettu jokaisen rivin päivittämisen yhteydessä.

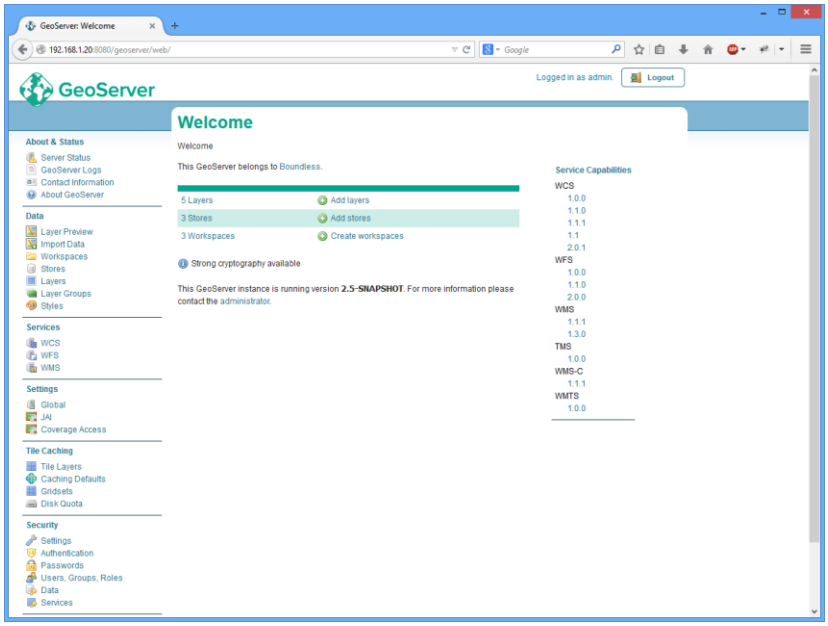
Laukaisimen luominen tapahtui antamalla komentokehotteessa psql-työkalua käyttäen komento:

```
CREATE TRIGGER update_geom AFTER INSERT ON xxx.lentoesteet
FOR EACH STATEMENT EXECUTE PROCEDURE update_geom_function();
```

Laukaisimessa määriteltiin, että se laukaisee aikaisemmin luodun funktion jokaisen lausekkeen jälkeen, joka oli tehty ”lentoesteet”-tauluun.

4.2.3 Karttapalvelin

GeoServerin avulla voitiin julkaista lentoesterekisteri-web-sovellukselle PostgreSQL-tietokannassa olevia lentoestetietoja OGC:n paikkatietorajapintoja käyttäen. Kaikki GeoServeriin liittyvät toimet tehtiin hallintasivun kautta, jonne pääsi selaimella osoitteesta http://192.168.1.20:8080/geoserver/web/. Kaikki julkaisuun liittyvät vaiheet on kuvattu kokonaisuudessaan liitteessä 5.



Kuvio 12. GeoServerin hallintasivu

Julkaisun tekeminen eteni neljässä vaiheessa seuraavasti: ensimmäiseksi luotiin uusi työtila nimeltään ”Lentoesteet”.

Toiseksi luotiin uusi PostGIS-tietolähde nimeltään ”Lentoesteet_PostGIS”, jossa viitattiin lentoesteiden PostgreSQL-tietokantaan. Lisäksi tietolähde liitettiin ”Lentoesteet”-työtilaan.

Kolmanneksi luotiin kolme uutta tasoa nimiltään: ”Lentoesteet_kaikki”, ”Lentoesteet_tyypin_mukaan” ja ”Lentoesteet_idn_mukaan”. Jokaiseen tasoon luotiin oma SQL-näkymä, jonka avulla haettiin ”Lentoesteet_PostGIS”-tietolähteestä kuhunkin tasoon liittyviä tietoja. Esimerkiksi ”Lentoesteet_kaikki”-SQL-näkymään määriteltiin, että se hakee kaikki lentoesteet tietokannasta, kun taas ”Lentoesteet_tyypin_mukaan”

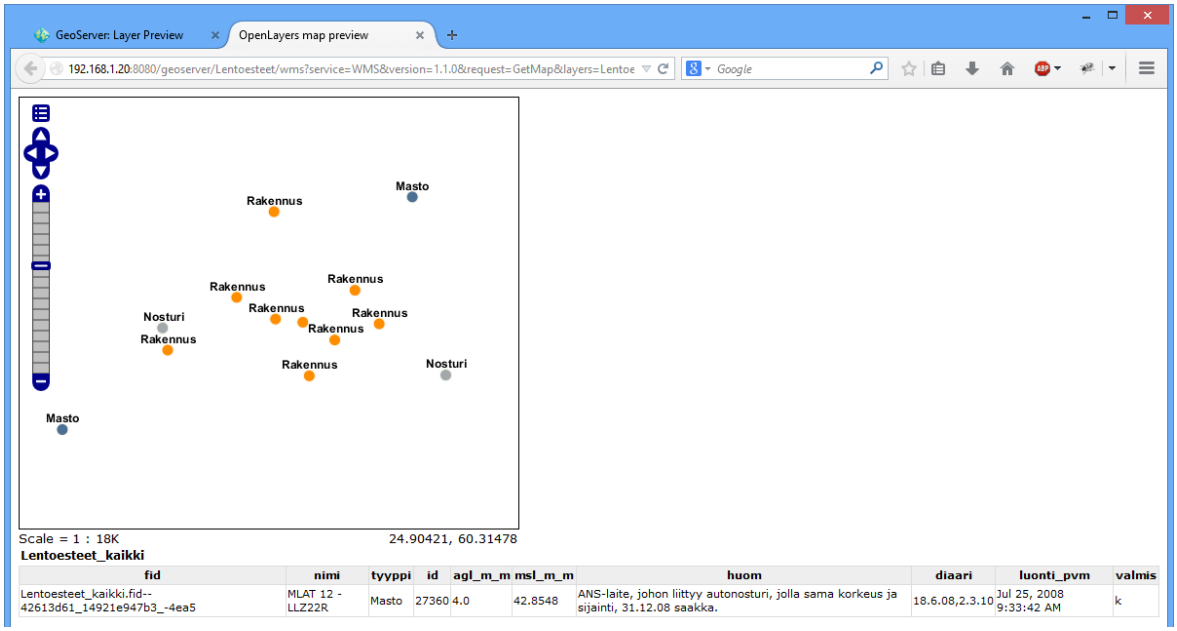
otti vastaan haettavan lentoesteen tyypin parametrina ja haki sen perusteella tietoja tietokannasta. Näiden lisäksi kaikki tasot liitettiin ”Lentoesteet”-työtilaan. Tämän jälkeen luotuja tasoja voitiin hyödyntää OGC:n WMS- ja WFS standardien mukaisesti; standardeja käsiteltiin luvussa 3.1.

Neljänneksi luotiin kaksi uutta tyylitiedostoa, jotka perustuivat OGC:n SLD-standardiin, joka käsiteltiin luvussa 3.1.

Ensimmäiseen tiedostoon nimeltään ”Tyylit_kaikki” määriteltiin tyylit tasoille ”Lentoesteet_kaikki” ja ”Lentoesteet_tyypin_mukaan”. Määrityksissä lentoesteet asetettiin näkymään erivärisinä palloina siten, että lentoesteen tyyppi luki niiden yläpuolella. Lisäksi määriteltiin asetus, jonka avulla lentoesteitä ei näytetty ennen kuin tarkennuksen taso oli tarpeeksi lähelle maata. Tämä tehtiin siksi, että lentoesteiden kokonaismäärä oli niin suuri, että ne olisivat tarkennuksen suuntautuessa kauemmaksi maasta peittäneet koko kartan ja tämä taas olisi hidastanut tasojen toimintaa merkittävästi.

Toiseen tiedostoon nimeltään ”Tyylit_id” määriteltiin tyylit tasolle ”Lentoesteet_idn_mukaan”. Määritykset olivat samanlaiset kuin ensimmäisessäkin lukuun ottamatta asetusta, jossa määriteltiin lentoesteiden näkyvyys. Tässä tapauksessa asetus määriteltiin siten, että lentoesteet näytettiin kaikilta tarkennustasoilta, koska kyseisellä tasolla näytettiin vain yksi lentoeste kerrallaan. Näiden lisäksi molemmat luodut tyylitiedostot liitettiin ”Lentoesteet”-työtilaan.

Kun tasot ja tyylit olivat valmiit, testattiin ”Lentoesteet_kaikki”-tasoa GeoServerin ”Layer Preview”-työkalulla, jolla se saatiin näkyviin selaimella OpenLayers-formaattia käyttäen, kuten kuviossa 13 on esitetty.



Kuvio 13. GeoServer Layer Preview näkymä

Lentoesteiden ominaisuustietoja pystyttiin testaamaan klikkaamalla lentoesteen päällä, jonka jälkeen sen tiedot tulivat näkyville kartan alapuolelle.

4.2.4 Web-sovellus

Web-sovelluksen toteutukseen käytettiin OpenGeo Suiten mukana asentunutta Boundless SDK-työkalua, jolla oli mahdollista luoda valmis mallipohja JavaScript-pohjaiselle web-karttasovellukselle. Käytettäväksi pohjaksi valittiin ”ol3view”, joka pohjautui OpenLayers 3:een, jota käytiin läpi luvussa 3.3, ja Bootstrapiin, joka on HTML-, CSS- ja JavaScript-kehys responsiivisten web-sivujen rakentamista varten. Kyseinen pohja valittiin, koska se sisälsi valmiiksi muokatun pohjan, johon sisältyi kaikki tarvittavat ominaisuudet, joita lentoesterekisteri-web-sovelluksessa tultiin tarvitsemaan.

Uuden lentoesterekisteri-web-sovelluksen pohjan luominen tapahtui komentokehotteesta komennolla:

```
suite-sdk create /home/xxx/lentoesteet/ ol3view
```

Tämän jälkeen määritettyyn kansioon muodostui hakemistorakenne, jonka sai kokonaisuudessaan listattua komennolla:

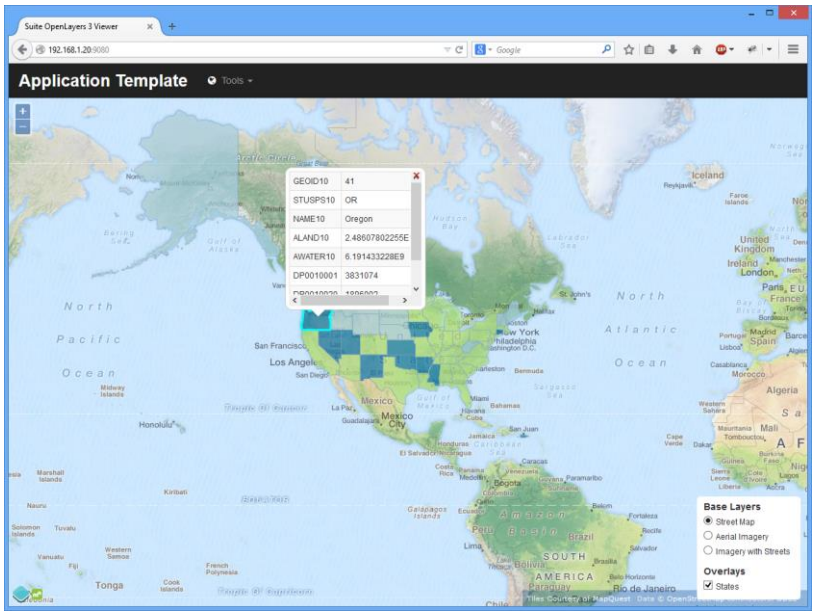
```
tree -d -f -A /home/xxx/lentoesteet/
```

```
/home/xxx/lentoesteet
└── /home/xxx/lentoesteet/src
    ├── /home/xxx/lentoesteet/src/app
    ├── /home/xxx/lentoesteet/src/bootbox
    ├── /home/xxx/lentoesteet/src/bootstrap
    │   ├── /home/xxx/lentoesteet/src/bootstrap/css
    │   ├── /home/xxx/lentoesteet/src/bootstrap/fonts
    │   └── /home/xxx/lentoesteet/src/bootstrap/js
    ├── /home/xxx/lentoesteet/src/css
    ├── /home/xxx/lentoesteet/src/font-awesome
    │   ├── /home/xxx/lentoesteet/src/font-awesome/css
    │   ├── /home/xxx/lentoesteet/src/font-awesome/fonts
    │   ├── /home/xxx/lentoesteet/src/font-awesome/less
    │   └── /home/xxx/lentoesteet/src/font-awesome/scss
    ├── /home/xxx/lentoesteet/src/jquery
    └── /home/xxx/lentoesteet/src/ol3
```

Luotua pohjaa pääsi kokeilemaan testaustilassa antamalla komentokehotteessa komennon:

```
suite-sdk debug /home/xxx/lentoesteet/

Starting debug server for application (use CTRL+C to stop)
Buildfile: /usr/share/opengeo/webapp-sdk/build.xml

debug:
[main] INFO  ringo.httpserver  - Server on http://localhost:9080
started.
```

Komennon suorittamisen jälkeen pohja käynnistyi ja sitä pääsi katsomaan selaimella osoitteesta: http://localhost:9080. Luodusta pohjasta löytyi valmiiksi valikkorivi, johon sisältyi ”Tools”-alasvetovalikko, kartan tarkennus- ja loitonnus-painikkeet, taustakartta sekä taustakartan vaihtaja (kuvio 14). Lisäksi siitä löytyi toiminnollisuudet, jotka mahdollistivat kartalla olevien kohteiden tietojen tarkastelemisen ponnahdusikkunasta klikkaamalla jotakin kohdetta kartalla.



Kuvio 14. Lentoesterekisteri-web-sovelluksen pohjasivu

Seuraavassa vaiheessa alettiin muokkaamaan lentoesterekisteri-web-sovelluksen pohjaa vaatimusten mukaiseksi. Muokkauksen kannalta oleellisimmat tiedostot olivat

```
/home/xxx/lentoesteet/src/index.html
```

```
/home/xxx/lentoesteet/src/app/app.js
```

Karkeasti jaoteltuna sovelluksen ulkoasuun tehdyt muutokset määriteltiin index.html-tiedostoon ja toiminnollisuudet app.js-tiedostoon. Muokatut index.html- ja app.js-tiedostot löytyvät kokonaisuudessaan liitteistä 6 ja 7.

Lentoeste-web-sovelluksen muokkaaminen eteni pääpiirteittäin seuraavasti: Ensimmäiseksi muokattiin app.js-tiedostoa siten, että sen konfiguraatio-osioon lisättiin lähteeksi GeoServer (toteutus kuvattu luvussa 4.2.3) ja sen tarjoamat rajapintapalvelut (rivit 9-13). Lisäksi määriteltiin tasolle otsikko (rivi 14), tietoformaatti (rivi 15), koordinaatit aloitusnäkymää varten (rivi 16), tarkennuksen taso aloitusnäkymää varten (rivi 17), tarkennuksen taso pikavalintoina olevien lentokenttien osalta (rivi 18) ja muuttujat tyylitiedostoja varten (rivit 19-20).

```
var url = 'http://192.168.1.20:8080/geoserver/ows?';
var featurePrefix = 'Lentoesteet';
var featureType = 'Lentoesteet_kaikki';
var featureTypeSort = 'Lentoesteet_tyypin_mukaan';
var featureTypeId = 'Lentoesteet_idn_mukaan';
var layerTitle = 'Lentoesteet';
var infoFormat = 'application/json';
var center = [2812316,9583000,2831425,9594466];
var zoom = 5;
var zoom_bookmark = 12;
var styleId = 'Tyyppi_teema_id';
var styleAll = 'Tyyppi_teema_testi';
```

Seuraavaksi app.js-tiedostoon luotiin kirjanmerkki-osio ja siihen muuttujiksi kaikkien Finavian hallinnoimien lentokenttien ICAO-koodit, sekä niiden koordinaatit alla olevan esimerkin mukaisesti:

```
var EFHK = [2776905.555, 8468705.106, 2780905.555, 8472705.106];
var EFHF = [2785903.881, 8454481.675, 2789903.881, 8458481.675];
var EFTU = [2476157.398, 8513237.62, 2480157.398, 8517237.62];
var EFLP = [3131489.055, 8634354.816, 3135489.055, 8638354.816];
```

Tämän jälkeen luotiin uusi OpenLayers-karttatiili WMS-lähde käyttämällä aikaisemmin tehtyjä muuttujia hyödyksi siten, että oletustasona käytettiin luvussa 4.2.3 luotua ”Lentoesteet_kaikki”-tasoa.

```
var wmsSource = new ol.source.TileWMS({
 url: url,
 params: {'LAYERS': featurePrefix + ':' + featureType,
'FORMAT':'image/png8', 'TILED': true},
 serverType: 'geoserver'
 });
```

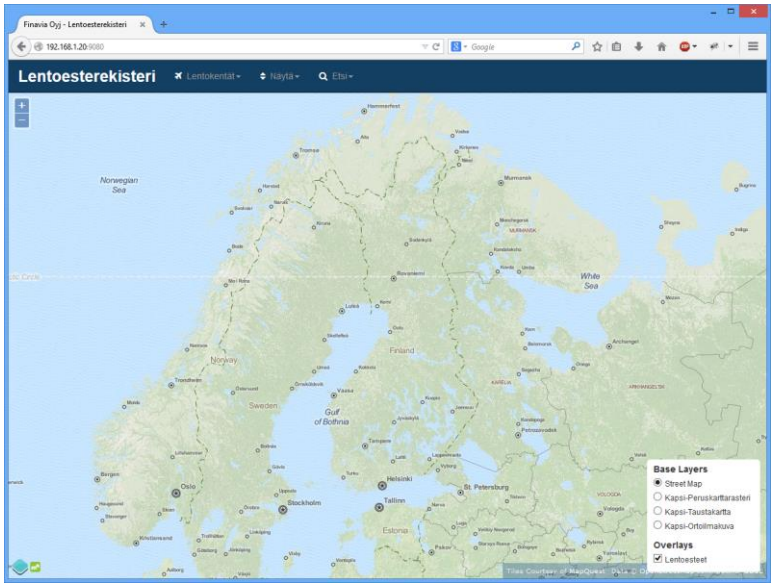
Tämän jälkeen app.js-tiedostoon luotiin sovelluksen toiminnollisuuksia varten funktioita, joiden avulla voitiin siirtyä kartalla pikalinkkeinä olevien lentoasemien sijaintiin, siirtyä kartalla aloitusnäkymään, valita minkä tyyppisiä lentoesteitä kartalla näytetään, etsiä lentoesteitä id-numeron ja nimen mukaan sekä mahdollista vaihtaa pohjataso kapsin

tarjoamaan kartta-aineistojen WMS-palveluihin. Nämä toiminnollisuudet löytyvät liitteestä 7 riveiltä 95-229 ja 238-297.

Esimerkiksi kartalla näkyvien estetyyppien mahdollistava funktio toteutettiin siten, että kun index.html-sivulla valittiin ”Näytä”-valikosta mikä tahansa estetyyppi, mikä haluttiin kartalla näkyvän, asetettiin valitun estetyypin id-arvo parametriksi ”SortType”-muuttujaan. Tämän jälkeen rivillä 71 luodun WMS-lähteen parametreja muutettiin siten, että tasoksi vaihdettiinkin luvussa 4.2.3 luotu ”Lentoesteet_tyypin_mukaan” ja sille annettiin parametriksi ”SortType”-muuttujan arvo, jonka jälkeen se palautti tietokannasta vain valitun estetyypin mukaiset esteet.

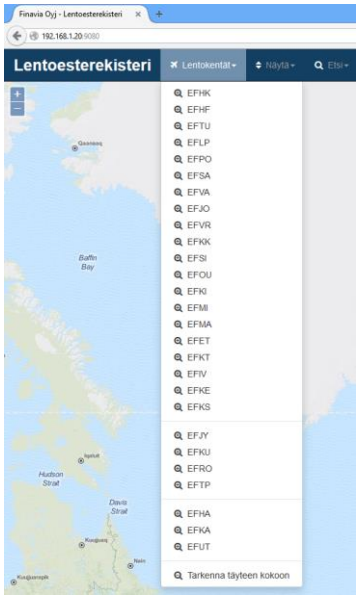
```
206 function showType(type) {
207  return function() {
208    var sortType = type;
209    wmsSource.updateParams({'LAYERS': featurePrefix + ':' + fea-
tureTypeSort, viewparams:"tyyppi:" + sortType, 'STYLES':styleAll,
'FORMAT':'image/png8', 'TILED': true});
210    }
211  }
```

Seuraavaksi index.html-tiedostoa muokattiin siten, että valikkorivin väri vaihdettiin tummansiniseksi, jonka lisäksi siihen lisättiin uudet alas vetovalikot (lentokentät, näytä ja etsi) ja niille ikonit. Tämän lisäksi määriteltiin taustakarttavaihtoehdoiksi kapsi WMS-palvelun peruskarttarasteri, taustakartta ja ortoilmakuvakartta. Nämä muutokset löytyvät liitteestä 6 riveiltä 57, 93, 106 ja liitteestä 7 alkaen riviltä 265. Muutosten jälkeen sivu näytti selaimella kuvion 15 mukaiselta.



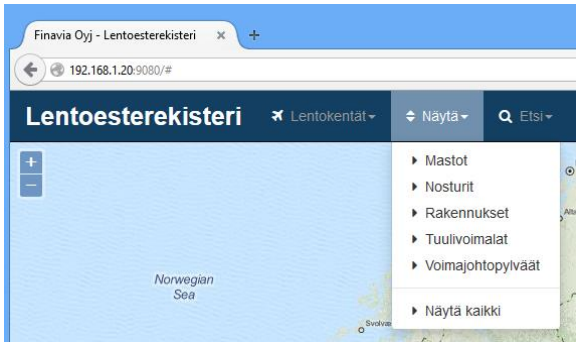
Kuvio 15. Lentoeste-web-sovelluksen etusivu

Lentokentät-alasvetovalikkoon lisättiin kaikki Finavian hallinnoimat lentokentät kuvion 16 mukaisesti. Kaikille lentoasemille määriteltiin oma id-tunniste, joita hyödynnettiin app.js-tiedostoon ohjelmoiduissa funktioissa. Lopputuloksena, kun jotakin lentokentistä klikattiin, siirryttiin kartalla kyseisen lentokentän kohdalle. Lisäksi samaan valikkoon lisättiin kohta ”Tarkenna täyteen kokoon”, jota klikkaamalla siirryttiin kartalla aloitusnäkymään, joka on näkyvillä kuviossa 15. Nämä muutokset löytyvät liitteestä 6 riveiltä 59-89.



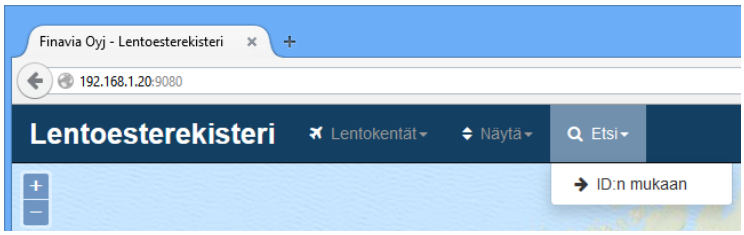
Kuvio 16. Lentokentät valikon sisältö

Näytä-alasvetovalikkoon lisättiin viisi eniten käytössä olevaa lentoestetyyppiä, jotka ovat esitetty kuviossa 17. Estetyyppiä klikkaamalla tuli kartalle näkyviin ainoastaan kyseisen tyyppiset lentoesteet. Lisäksi samaan valikkoon lisättiin kohta ”Näytä kaikki”, jota klikkaamalla kaikki estetyypit näkyivät kartalla. Nämä muutokset löytyvät liitteestä 6 riveiltä 95-101.



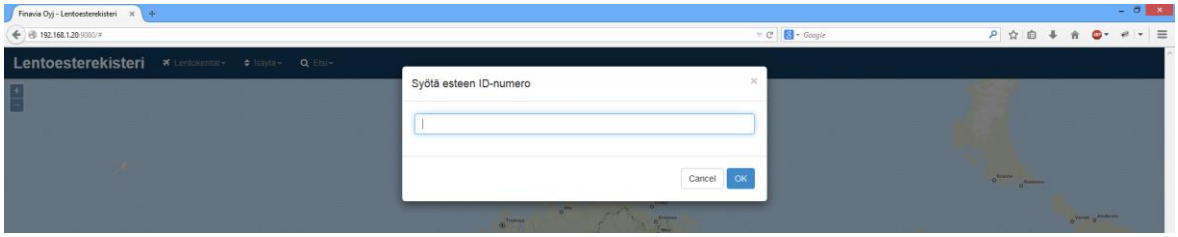
Kuvio 17. Näytä valikon sisältö

Etsi-alasvetovalikkoon lisättiin hakutoiminto lentoesteen id-numeron perusteella kuvion 18 mukaisesti. Tämä muutos löytyy liitteestä 6 riviltä 108.



Kuvio 18. Etsi valikon sisältö

Lentoesteen etsimistä id-numerolla toiminnon painiketta klikatessa aukesi erillinen ikkuna kuvion 19 mukaisesti.



Kuvio 19. Lentoesteen haku id-numeron perusteella

Ikkunassa olevaan tekstikenttään voitiin syöttää lentoesteen id-numero, jonka jälkeen kartalla näkyi vain kyseinen lentoeste.

4.3 Tulokset

Lopputuloksena syntynyt lentoesterekisteri-web-sovellus saatiin kokonaisuutena toimimaan luvussa 4.1 esitettyjen vaatimusten mukaisella tavalla. Lentoesteet näytettiin pistemuotoisena vaihdettavalla karttapohjalla ja niiden ominaisuustietoja voitiin tarkastella. Lisäksi lentoesteitä voitiin suodattaa estetyypin mukaan ja etsiä id-numeron perusteella.

Teknisen ympäristön toteutuksessa hyödynnettiin avoimen lähdekoodin ohjelmistoja käyttöjärjestelmän, tietokannan ja karttapalvelimen osalta. Avoimen lähdekoodin vahvuus tuli esille Ora2Pg-ohjelman käytön yhteydessä, kun lähdekoodia muokkaamalla saatiin ohjelma toimimaan halutulla tavalla.

Tietokannan toteutuksessa tehty tietojen siirto ja konversio Oracle-tietokannasta PostgreSQL-tietokantaan onnistui halutulla tavalla. Lisäksi geometriasarakkeen automaattinen päivittäminen tietojen lisäämisen jälkeen saatiin toteutettua käyttämällä funktiota ja laukaisinta.

Karttapalvelimen toteutuksessa hyödynnettiin teoriaosuuden luvussa 3.1 käsiteltyä WMS-standardia, jonka avulla GeoServerillä voitiin julkaista lentoesteistä erilaisia tasoja, jotka näytettiin lentoesterekisteri-web-sovelluksessa tiilimuotoisina karttakuvina PNG-formaatissa. Lisäksi WMS-standardia hyödynnettiin kapsi-karttojen lisäämiseksi vaihtoehtoisiksi taustakartoiksi. Karttapalvelimella hyödynnettiin myös luvussa 3.1 käsiteltyä WFS-standardia, jonka avulla voitiin kysellä lentoesteiden ominaisuustietoja ja esittää ne lentoesterekisteri-web-sovelluksessa.

Lentoesteiden symboloinnin ja värityksen toteutuksessa hyödynnettiin teoriaosuuden luvussa 3.1 käsitellyn SLD-standardin sääntöjä ja niihin liittyviä suodattimia sekä pistesymboloijaa.

5 Yhteenveto ja pohdinta

Opinnäytetyöni tavoitteena oli tutkia keskeisiä The Open Geospatial Consortiumin (OGC)-rajapintastandardeja ja niiden hyödyntämistä sekä selvittää yleisellä tasolla mitä paikkatieto on ja miten sitä voidaan hyödyntää. Lisäksi tavoitteenani oli tehostaa ja monipuolistaa lentoestetietojen esittämistapaa rakentamalla OGC:n paikkatietorajapintoja hyödyntävä lentoesterekisteri-web-sovellus Finavian sisäiseen käyttöön.

Opinnäytetyön tavoitteessa onnistuttiin hyvin jokaisella osa-alueella. Tietoperustassa tutkittuja OGC:n rajapintastandardeja hyödynnettiin laajasti empiriaosuudessa toteutetussa lentoesterekisteri-web-sovelluksessa. Kerätty tietoperusta toimi hyvänä pohjana toteutuksessa, lukuun ottamatta teknistä ympäristöä ja siihen liittyviä komponentteja, jotka vaativat lisätutkimusta erilaisista lähteistä. Lopputuloksena syntynyt lentoesterekisteri-web-sovellus täytti kaikki sille asetetut vaatimukset ja ominaisuudet. Lentoesteiden esittämistapa tehostui ja monipuolistui merkittävästi aikaisempaan esittämistapaan verrattuna. Lisäksi toimeksiantaja ja minä itse olimme lopputulokseen tyytyväisiä.

Opinnäytetyön empiirisen osuuden toteuttaminen sisälsi monia erilaisia vaiheita ja se vaati paljon selvitystyötä. Suurimpina haasteina olivat tietokantakonversion toteuttaminen ja karttapalvelimen konfigurointi, jotka veivät suurimman osan ajasta. Toteutuksessa teknisen ympäristön osalta tehdyt valinnat tietokannan, karttapalvelimen ja web-sovelluksen osalta osoittautuivat onnistuneiksi ja tulen hyödyntämään niitä tulevaisuudessa muissakin projekteissa.

Kaiken kaikkiaan opinnäytetyö kokonaisuudessaan tarjosi todella paljon uusia asioita paikkatiedosta, moderneista web-tekniikoista, rajapintastandardeista ja niitä hyödyntävän web-sovelluksen kehittämisestä. En ollut aikaisemmin tehnyt työssäni näin laajaa toteutusta ja lähestulkoon kaikki projektissa käytetyt tekniikat lukuun ottamatta Oracle-tietokantaa, olivat minulle uusia asioita. Tästä johtuen niihin perehtyminen oli todella mielenkiintoista ja opettavaista. Opin toteutuksen aikana paljon hyödyllisiä asioita itsenäisestä projektityöskentelystä, vaatimusmäärittelyistä ja paikkatietorajapintoja hyödyntävän web-sovelluksen toteutusarkkitehtuurista. Ammattitaitoni kehittymisen kannalta

tärkeimpinä oppeina olivat syvällisempi tutustuminen JavaScript-kieleen ja teknisen ympäristön toteutuksessa käytettyihin avoimen lähdekoodin ohjelmistoihin.

Lentoesterekisteri-web-sovelluksen toteutuksessa käytetyt OGC:n rajapintastandardit mahdollistavat lentoestetietojen hyödyntämisen myös muillakin rajapinnoilla kuin web-sovelluksella. Tämä mahdollistaa erilaisia jatkokehitysmahdollisuuksia toimeksiantajan prosessien kehittämisen kannalta, kuten esimerkiksi tiedon helpompaa ja tehokkaampaa jakamista erilaisten käyttöliittymien avulla organisaation eri yksiöiden kesken. Oleellisia jatkokehitystehtäviä ilmeni projektin aikana tietokannan, karttapalvelimen ja web-sovelluksen eriyttämisestä omille alustoilleen, kun sovellusta aletaan käyttämään tuotantoympäristössä. Tällä mahdollistetaan parempi suorituskyky ja selkeytetään ylläpitoa. Tämän lisäksi ilmeni tarve saada tulevaisuudessa lentoesteet-taulun ”diaari”-sarakkeesta linkitys dokumentinhallintajärjestelmään, josta löytyy lentoesteiden lausunnot ja luvat.

Opinnäytetyön tulosten perusteella jatkotutkimuksena voitaisiin tehdä OGC:n rajapintapastandardien kokonaisvaltaisemmasta hyödyntämisestä osana yrityksen paikkatietoprosesseja.

Lähteet

DeMers, N. M. 2009. GIS For Dummies. Wiley Publishing, Inc. Hoboken, NJ.

Dempsey Morais, C. 2012. What is GIS?. Luettavissa: http://www.gislounge.com/what-is-gis/. Luettu: 19.8.2014.

DigitalGlobe 2013. Web Map Tile Service Developer Guide. Luettavissa: http://www.digitalglobe.com/sites/default/files/dgcs/DGCS_DeveloperGuide_WMTS.pdf. Luettu: 30.8.2014.

ESRI 2014a. Paikkatiedon (GIS) perusteet. Luettavissa: http://www.esri.fi/referenssit/mita_paikkatieto_on/paikkatiedon_perusteet/. Luettu: 23.7.2014.

ESRI 2014b. Paikkatiedon käyttö. Luettavissa: http://www.esri.fi/referenssit/mita_paikkatieto_on/paikkatiedon_kaytto/. Luettu: 23.7.2014.

ESRI 2014c. Mitä ovat paikkatieto ja GIS?. Luettavissa: http://www.esri.fi/referenssit/mita_paikkatieto_on/. Luettu: 23.7.2014.

Fu, P & Sun, J. 2011. Web GIS: Principles and Applications. First edition. ESRI Press. Redlands, California.

Giuliani, G., Lacroix, P., Guigoz, Y., Bigagli, L., Ray, N & Lehmann, A. 2014. Bringing GEOSS services into practice. GIS Open Source Workshop Material. University of Geneva, United Nations Environment Programme, National Research Council of Italy.

Heywood, I. Cornelius, S. & Carver, S. 2011. An introduction to geographical information systems. Fourth edition. Pearson Education Limited. Edinburg Gate, England.

Huisman, O. & de By, R. A. 2009. Principles of Geographic Information Systems. Fourth edition. The International Institute for Geo-Information Science and Earth Observation. Enschede, The Netherlands. Luettavissa: http://www.itc.nl/library/papers_2009/general/PrinciplesGIS.pdf. Luettu: 22.7.2014.

Julkisen hallinnon tietohallinnon neuvottelukunta 2013a. JHS 180 Paikkatiedon sisältöpalvelut. Liite 1 Karttakuvapalvelu. Luettavissa: http://docs.jhs-suositukset.fi/jhs-suositukset/JHS180_liite1/JHS180_liite1.pdf. Luettu: 17.8.2014.

Julkisen hallinnon tietohallinnon neuvottelukunta 2013b. JHS 180 Paikkatiedon sisältöpalvelut. Rajapintapalvelut. Luettavissa: http://docs.jhs-suositukset.fi/jhs-suositukset/JHS180/JHS180.pdf. Luettu: 30.8.2014.

Karimi, H. A. 2014. Big Data: Techniques and Technologies in Geoinformatics. CRC Press Taylor & Francis Group. Boca Raton, FL 33487-2742.

Kylmäaho, J. 2011. Katselupalvelut ja latauspalvelut - Paikkatietoa karttakuvina ja GML-muodossa. Luettavissa: http://www.paikkatietoikkuna.fi/c/document_library/get_file?uuid=962cebfc-06ec-4706-923f-e055a740385b&groupId=108478. Luettu: 25.8.2014.

Lepistö, J. 2000. Paikkatieto. Tietotekniikan LuK-tutkielma. Jyväskylän yliopisto. Luettavissa: http://www.mit.jyu.fi/opiskelu/opinnayte/LuK/Paikkatieto/. Luettu: 7.7.2014.

Maanmittauslaitos 2014. Rajapintapalvelut ABC. Luettavissa: http://www.maanmittauslaitos.fi/aineistot-palvelut/rajapintapalvelut/rajapintapalvelut-abc. Luettu: 5.8.2014.

Mimas 2013a. Unit 2 - OGC Standards and Geoportals. Luettavissa: http://find.jorum.ac.uk/resources/bitstream/533370. Luettu: 10.11.2014.

Mimas 2013b. Unit 3 - Geographic server OGC standards. Luettavissa: http://find.jorum.ac.uk/resources/bitstream/533370. Luettu: 10.11.2014.

OGC 2014a. About OGC. Luettavissa: http://www.opengeospatial.org/ogc. Luettu: 22.8.2014.

OGC 2014b. Styled Layer Descriptor. Luettavissa: http://www.opengeospatial.org/standards/sld. Luettu: 5.9.2014.

OpenLayers. 29.8.2014. #The OpenLayers community is proud to announce the release of OpenLayers 3.0! Check out http://openlayers.org. Twitter-viesti @openlayers. Luettavissa: https://twitter.com/openlayers/status/505372137340014592. Luettu: 6.9.2014.

OpenLayers 2014. OpenLayers 3 Quick Start. Luettavissa: http://openlayers.org/en/v3.0.0/doc/quickstart.html. Luettu: 6.9.2014.

OSGeo 2014. OpenLayers Info Sheet. Luettavissa: http://www.osgeo.org/openlayers. Luettu: 6.9.2014.

Peng, Z. & Tsou, M. 2003. Internet GIS: Distributed geographic information service for the Internet and wireless networks. John Wiley & Sons. Hoboken, New Jersey.

Peuralahti, J. 2014. Geographic Information System : A Case Study for Developers. Yamk-opinnäytetyö. Metropolia ammattikorkeakoulu. Luettavissa: http://urn.fi/URN:NBN:fi:amk-201402082142. Luettu: 20.7.2014.

Rumor, M., Coors, V., Fendel, E.M. & Zlatanova, S. 2007. Urban and Regional Data Management: UDMS 2007 Annual. Taylor & Francis/Balkema. AK Leiden, The Netherlands.

Sanastokeskus 2014. Sanastokeskus TSK ry. 2014. Geoinformatiikan sanasto. 3. laitos. Sanastokeskus TSK ry. Helsinki.

Santiago, A. 2014. The book of OpenLayers 3. Luettavissa: https://leanpub.com/thebookofopenlayers3/read. Luettu: 30.8.2014.

Teknologiateollisuus 2013. Paikkatieto-osaaminen - suomalainen kasvuala. Luettavissa: http://teknologiateollisuus.fi/fi/uutishuone/tiedotteet/2013-11/paikkatieto-osaaminen-suomalainen-kasvuala. Luettu: 10.7.2014.

Vehkaperä, H. 2009. Mitä ovat WMS, WFS, WCS – ja mihin niitä tarvitaan?. Positio 2/2009, s. 24-25.

WCS 2012. OGC WCS 2.0 Interface Standard- Core: Corrigendum. Luettavissa: https://portal.opengeospatial.org/files/09-110r4. Luettu: 16.8.2014.

Wikipedia 2014. Open Geospatial Consortium. Luettavissa: http://en.wikipedia.org/wiki/Open_Geospatial_Consortium. Luettu: 30.8.2014.

Liitteet

Liite 1. Projektisuunnitelma

FINAVIA Projektisuunnitelma 1 (1)

5.5.2014

1 Paikkatieto rajapintoja hyödyntävä lentoesteri-web-sovellus

Projektin aloituskokous pidettiin 16.4.2014.

2 Projektin tavoite

Projektin tavoitteena on rakentaa paikkatieto rajapintoja hyödyntäen web-sovellus, jolla näytetään lentoesterekisterin tiedot kartalla. Web-sovelluksella tulisi olla seuraavia ominaisuuksia ja toimintoja:

- Lentoesterekisteri tietokannassa olevien tietojen näyttäminen pistemuotoisena karttapohjalla
- Lentoesteiden tietojen tarkastaminen kartalla olevista pisteistä
- Mahdollisuus käyttää taustakarttana Kapsi Internet-käyttäjät ry:n (kapsi) tarjoaman WMS-palvelun peruskarttarasteria, taustakarttaa ja ortoilmakuvaa
- Mahdollisuus siirtyä kartalla mille tahansa Finavian hallinnoimalle lentokentälle pikavalinnasta
- Mahdollisuus suodattaa kartalle lentoesteitä niiden estetyypin mukaan
- Mahdollisuus etsiä lentoestettä id-numeron perusteella

3 Projektin aikataulu

Projektilla ei ole lukkoon lyötyä aikataulua, mutta sitä tarkennetaan seurantapalavereissa etenemisen mukaan. Alustavasti aikataulu olisi seuraavan lainen:

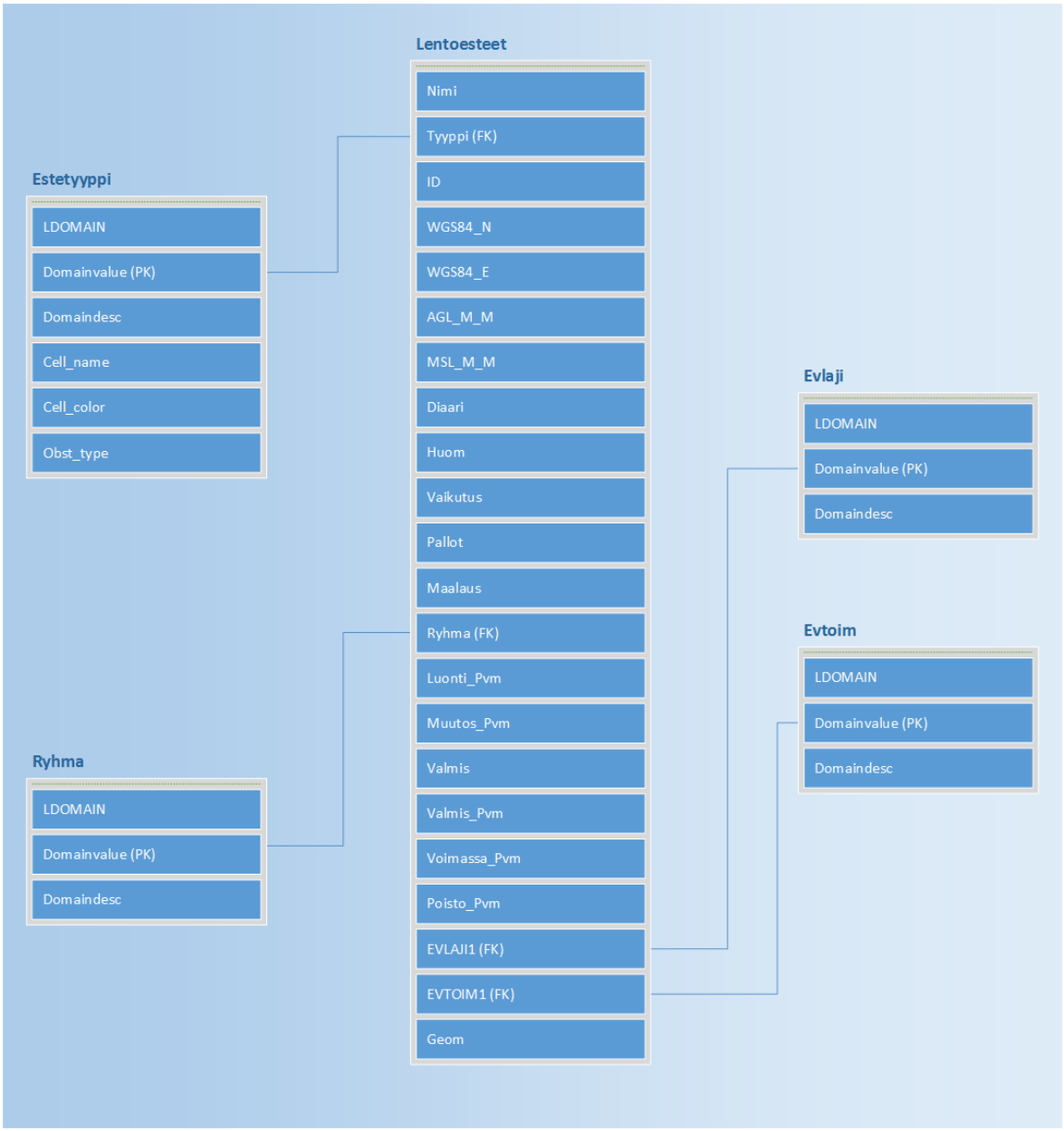
- Viikot 18 - 20 Sovelluksen toiminnollisuuksien määrittelyjä
- Viikot 20 - 21 Teknisen ympäristön suunnittelua
- Viikot 22 - 24 Tietokannan suunnittelua
- Viikot 24 - 27 Teknisen ympäristö toteuttamista
- Viikot 27 - 29 Tietokannan toteuttamista
- Viikot 30 - 32 Web-sovelluksen suunnittelua
- Viikot 33 - 37 Web-sovelluksen toteuttamista

4 Projektin toteutuksen vaiheet

Projektin toteutus aloitetaan suunnittelun- ja selvitystöiden osalta toukokuussa. Demoversio sovelluksesta olisi tarkoitus saada valmiiksi elokuun loppuun mennessä.

- Sovelluksen toiminnollisuuksien määrittely
- Sovelluksen teknisen ympäristön suunnittelu
- Tietokannan suunnittelu
- Tietokannan toteuttaminen
- Web-sovelluksen suunnittelu
- Web-sovelluksen toteuttaminen

Liite 2. Relaatiokaavio



Liite 3. PostgreSQL-tietokannan taulujen luontilauseet

```
SET client_encoding TO 'LATIN1';

\set ON_ERROR_STOP ON

CREATE TABLE lentoesteet (
            nimi varchar(40),
            tyyppi varchar(8),
            id bigint,
            wgs84_n double precision,
            wgs84_e double precision,
            agl_m_m double precision,
            msl_m_m double precision,
            diaari varchar(40),
            huom varchar(4000),
            vaikutus varchar(40),
            pallot varchar(1),
            maalaus varchar(1),
            ryhma varchar(8),
            luonti_pvm timestamp,
            muutos_pvm timestamp,
            valmis varchar(1),
            valmis_pvm timestamp,
            voimassa_pvm timestamp,
            poisto_pvm timestamp,
            evlaji1 varchar(8),
            evtoim1 varchar(8),
            geom geometry(Point,4258)
);
CREATE UNIQUE INDEX lentoesteet_id ON lentoesteet (id);

CREATE TABLE estetyyppi (
            ldomain bigint NOT NULL,
            domainvalue varchar(8) PRIMARY KEY,
            domaindesc varchar(32),
            cell_name varchar(15),
            cell_color bigint,
            obst_type varchar(32)
);
CREATE INDEX estetyyppi_i ON estetyyppi (domaindesc);
```

```
CREATE TABLE ryhma (
           ldomain bigint NOT NULL,
           domainvalue varchar(8) PRIMARY KEY,
           domaindesc varchar(32)
);
ALTER TABLE ryhma ADD UNIQUE (domainvalue);
CREATE INDEX ryhma_i ON ryhma (domaindesc);

CREATE TABLE evlaji (
           ldomain bigint NOT NULL,
           domainvalue varchar(8) PRIMARY KEY,
           domaindesc varchar(32)
);
CREATE INDEX evlaji_i ON evlaji (domainvalue);

CREATE TABLE evtoim (
           ldomain bigint NOT NULL,
           domainvalue varchar(8) PRIMARY KEY,
           domaindesc varchar(32)
);
CREATE INDEX evtoim_i ON evtoim (domainvalue);

ALTER TABLE xxx.lentoesteet
           ADD CONSTRAINT foreign_key01 FOREIGN KEY (tyyppi)
           REFERENCES estetyyppi (domainvalue);

ALTER TABLE xxx.lentoesteet
           ADD CONSTRAINT foreign_key02 FOREIGN KEY (evlaji1)
           REFERENCES evlaji (domainvalue);

ALTER TABLE xxx.lentoesteet
           ADD CONSTRAINT foreign_key03 FOREIGN KEY (evtoim1)
           REFERENCES evtoim (domainvalue);

ALTER TABLE xxx.lentoesteet
           ADD CONSTRAINT foreign_key04 FOREIGN KEY (ryhma)
           REFERENCES ryhma (domainvalue);
```

Liite 4. Ora2Pg-ohjelman asennus Linux Ubuntu 12.04-käyttöjärjestelmälle

Ensimmäiseksi asennettiin paketteja, joita tarvittiin Ora2Pg:n asentamista varten.

```
sudo apt-get install libyaml-perl -y
sudo apt-get install libaio1 -y
sudo perl -MCPAN -e 'install DBI'
sudo apt-get install postgresql-client -y
sudo apt-get install libpq-dev -y
sudo apt-get install alien -y
```

Tämän jälkeen ladattiin osoitteesta:

http://www.oracle.com/technetwork/topics/linuxx86-64soft-092277.html

Oracle Instant Client version 11.2.0.2.0 paketit:

- oracle-instantclient11.2-basic-11.2.0.3.0-1.x86_64.rpm
- oracle-instantclient11.2-devel-11.2.0.3.0-1.x86_64.rpm
- oracle-instantclient11.2-sqlplus-11.2.0.3.0-1.x86_64.rpm

Seuraavaksi ne asennettiin:

```
cd /hakemisto_johon_Oracle_Instant_Client_ladattiin/
sudo alien -i oracle-instantclient11.2-basic-11.2.0.2.0.i386.rpm
sudo alien -i  oracle-instantclient11.2-devel-11.2.0.2.0.i386.rpm
sudo alien -i  oracle-instantclient11.2-sqlplus-11.2.0.2.0.i386.rpm
```

Tämän jälkeen muokattiin Oracle Instant Clientin asetuksia:

```
sudoedit /etc/ld.so.conf.d/oracle.conf
                /usr/lib/oracle/11.2/client64/lib
                export ORACLE_HOME=/usr/lib/oracle/11.2/client64
                export TNS_ADMIN=/usr/lib/oracle/11.2/client64/network/admin
sudo ldconfig
```

```
sudo ln -s /usr/include/oracle/11.2/client64
/usr/lib/oracle/11.2/client64/include
```

Seuraavaksi luotiin hakemistot network/admin ja kopioitiin valmiiksi määritellyt tns-names.ora ja sqlnet.ora tiedostot sinne. Tämän lisäksi hakemistossa oleville tiedostoille annettiin kaikille luku oikeudet.

```
cd /usr/lib/oracle/11.2/client64
sudo mkdir network
cd network
sudo mkdir admin
cd admin
sudo cp /tmp/ORA2PG/sqlnet.ora
/usr/lib/oracle/11.2/client64/network/admin
sudo cp /tmp/ORA2PG/tnsnames.ora
/usr/lib/oracle/11.2/client64/network/admin
sudo chmod a+r *.ora
```

Seuraavaksi määriteltiin ympäristömuuttujat.

```
sudoedit ~/.bashrc
                export LD_LIBRARY_PATH=/usr/lib/oracle/11.2/client64/lib
                export ORACLE_HOME=/usr/lib/oracle/11.2/client64
                export TNS_ADMIN=/usr/lib/oracle/11.2/client64/network/admin
Log out -> Log in
```

Tämän jälkeen ladattiin osoitteesta:
http://search.cpan.org/~pythian/DBD-Oracle-1.74/lib/DBD/Oracle.pm
tiedosto DBD-Oracle-1.74.tar.gz

Seuraavaksi ladattiin osoitteesta: http://search.cpan.org/dist/DBD-Pg/Pg.pm
tiedosto DBD-Pg-3.3.0.tar.gz

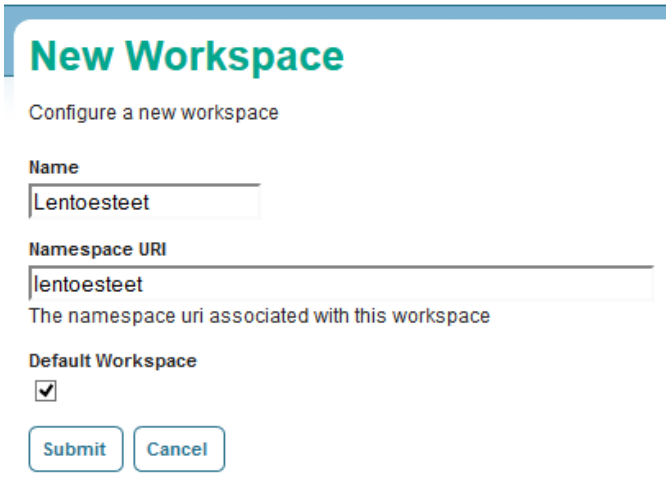
Ja viimeiseksi ladattiin osoitteesta: http://sourceforge.net/projects/ora2pg/
tiedosto ora2pg-13.0.tar.bz2

Tämän jälkeen ne purettiin ja asennettiin.

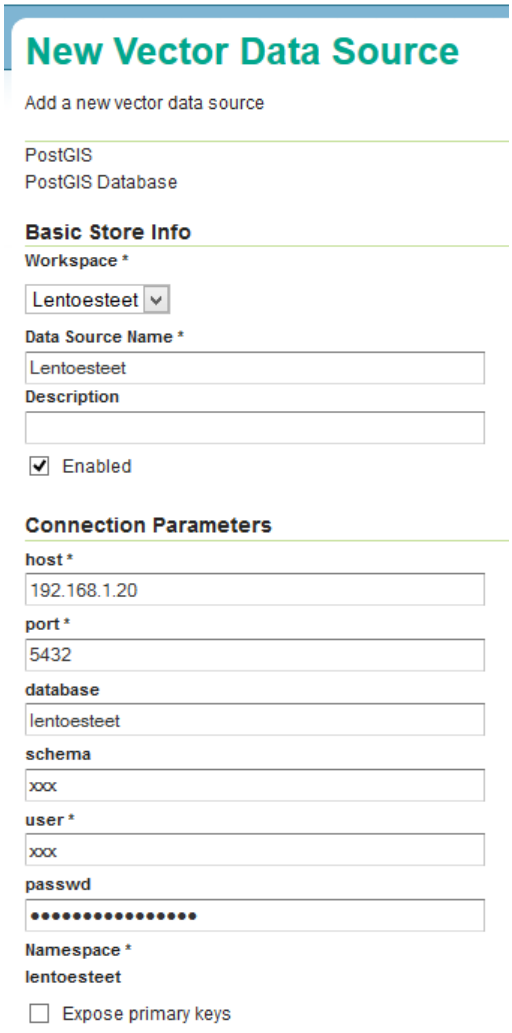
```
sudo tar xvf /mihin_tiedosto_ladattiin/ORA2PG/DBD-Oracle-1.74.tar.gz
cd /tmp/ORA2PG/DBD-Oracle-1.74
perl Makefile.pl
sudo make test
sudo make install

sudo tar xvf /mihin_tiedosto_ladattiin/ORA2PG/DBD-Pg-3.3.0.tar.gz
cd /tmp/ORA2PG/DBD-Pg-3.3.0
perl Makefile.pl
sudo make test
sudo make install

sudo tar xvf /mihin_tiedosto_ladattiin/ORA2PG/ora2pg-13.0.tar.bz2
cd /tmp/ORA2PG/ora2pg-13.0/
perl Makefile.pl
sudo make && make install
```

Liite 5. GeoServerin tasojen luominen ja julkaisu

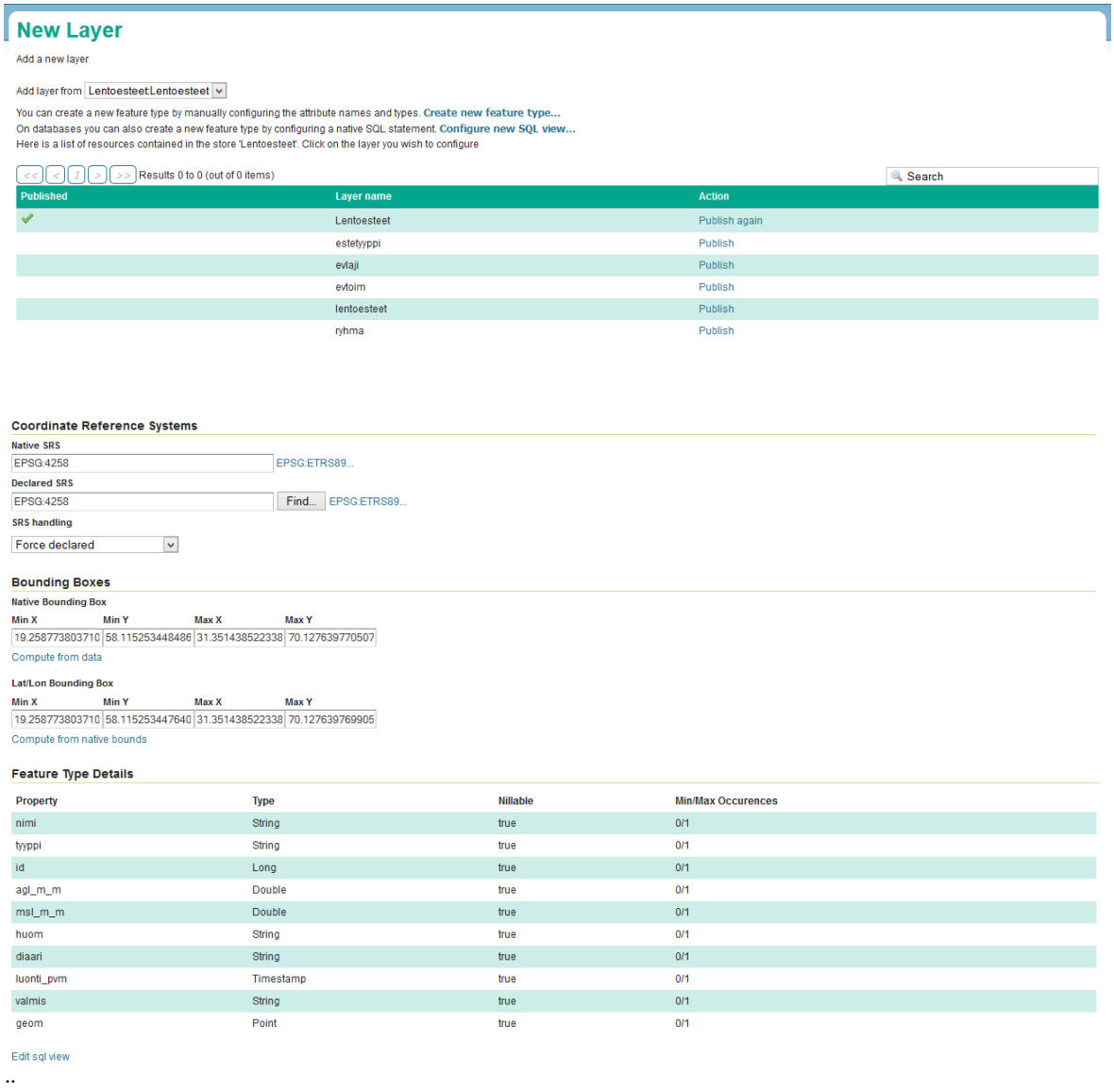
Ensimmäiseksi luotiin uusi työtila (workspace).



Toiseksi luotiin uusi PostGIS tietolähde.



Kolmanneksi luotiin 3 uutta tasoa (layer) ”Lentoesteet_kaikki”, ”Lentoesteet_tyypin_mukaan” ja ”Lentoesteet_idn_mukaan” ja määriteltiin niiden asetukset.



New Layer

Add a new layer

Add layer from Lentoesteet:Lentoesteet

You can create a new feature type by manually configuring the attribute names and types. Create new feature type...
On databases you can also create a new feature type by configuring a native SQL statement. Configure new SQL view...
Here is a list of resources contained in the store 'Lentoesteet'. Click on the layer you wish to configure

<< < 1 > >> Results 0 to 0 (out of 0 items) Search

Published	Layer name	Action
✓	Lentoesteet	Publish again
	estetyyppi	Publish
	evlaji	Publish
	evtoim	Publish
	lentoesteet	Publish
	ryhma	Publish

Coordinate Reference Systems

Native SRS
EPSG:4258 EPSG:ETRS89...

Declared SRS
EPSG:4258 Find... EPSG:ETRS89...

SRS handling
Force declared

Bounding Boxes

Native Bounding Box

Min X	Min Y	Max X	Max Y
19.258773803710	58.115253448486	31.351438522338	70.127639770507

Compute from data

Lat/Lon Bounding Box

Min X	Min Y	Max X	Max Y
19.258773803710	58.115253447640	31.351438522338	70.127639769905

Compute from native bounds

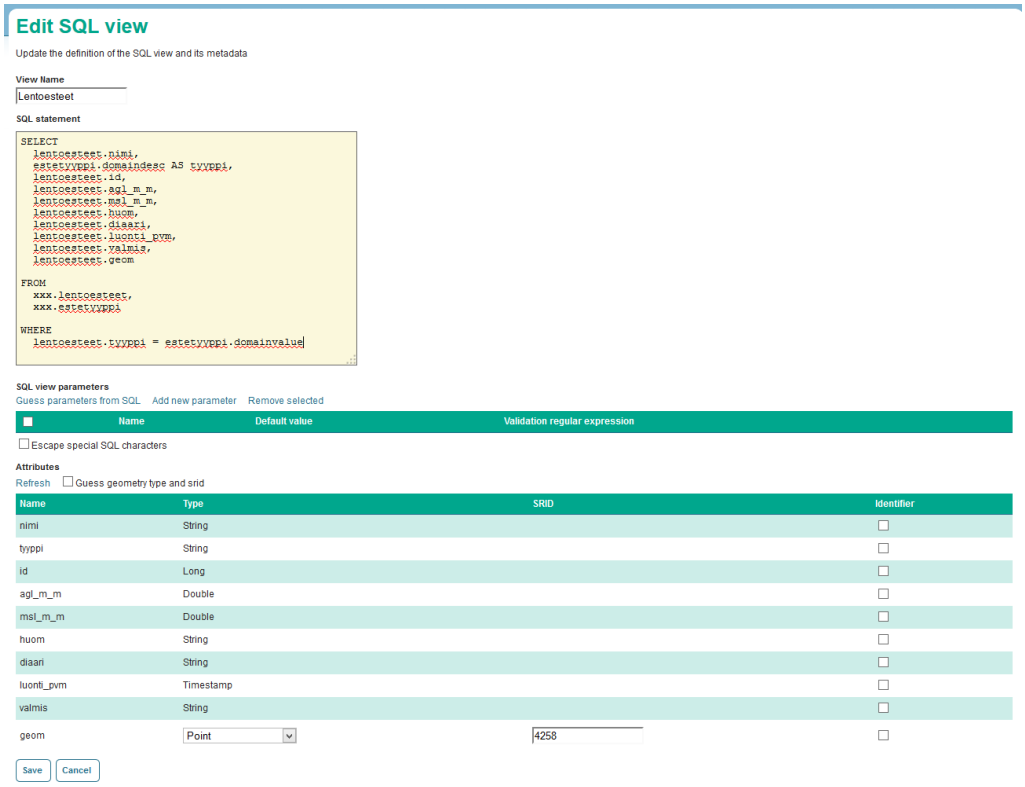
Feature Type Details

Property	Type	Nillable	Min/Max Occurences
nimi	String	true	0/1
tyyppi	String	true	0/1
id	Long	true	0/1
agl_m_m	Double	true	0/1
msl_m_m	Double	true	0/1
huom	String	true	0/1
diaari	String	true	0/1
luonti_pvm	Timestamp	true	0/1
valmis	String	true	0/1
geom	Point	true	0/1

Edit sql view

..

”Lentoesteet_kaikki” tason SQL-näkymän asetukset.



Edit SQL view

Update the definition of the SQL view and its metadata

View Name

Lentoesteet

SQL statement

```
SELECT
  lentoesteet.nimi,
  estetyyppi.domaindesc AS tyyppi,
  lentoesteet.id,
  lentoesteet.agl_m_m,
  lentoesteet.msl_m_m,
  lentoesteet.huom,
  lentoesteet.diaari,
  lentoesteet.luonti_pvm,
  lentoesteet.valmis,
  lentoesteet.geom

FROM
  xxx.lentoesteet,
  xxx.estetyyppi

WHERE
  lentoesteet.tyyppi = estetyyppi.domainvalue
```

SQL view parameters

Guess parameters from SQL Add new parameter Remove selected

	Name	Default value	Validation regular expression

Escape special SQL characters

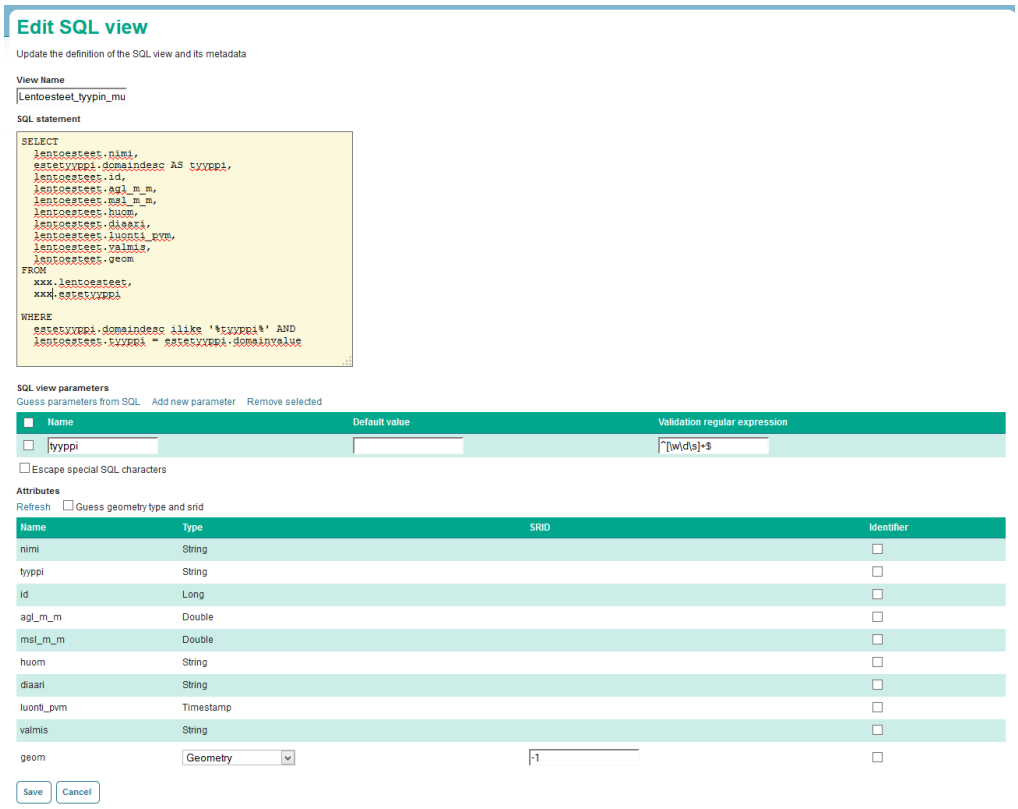
Attributes

Refresh Guess geometry type and srid

Name	Type	SRID	Identifier
nimi	String		
tyyppi	String		
id	Long		
agl_m_m	Double		
msl_m_m	Double		
huom	String		
diaari	String		
luonti_pvm	Timestamp		
valmis	String		
geom	Point	4258	

Save Cancel

”Lentoesteet_tyypin_mukaan” tason SQL-näkymän asetukset.



Edit SQL view

Update the definition of the SQL view and its metadata

View Name

Lentoesteet_tyypin_mu

SQL statement

```
SELECT
  lentoesteet.nimi,
  estetyyppi.domaindesc AS tyyppi,
  lentoesteet.id,
  lentoesteet.agl_m_m,
  lentoesteet.msl_m_m,
  lentoesteet.huom,
  lentoesteet.diaari,
  lentoesteet.luonti_pvm,
  lentoesteet.valmis,
  lentoesteet.geom
FROM
  xxx.lentoesteet,
  xxx.estetyyppi

WHERE
  estetyyppi.domaindesc ilike '%tyyppi%' AND
  lentoesteet.tyyppi = estetyyppi.domainvalue
```

SQL view parameters

Guess parameters from SQL Add new parameter Remove selected

	Name	Default value	Validation regular expression
	tyyppi		^[\w\d\s]+$

Escape special SQL characters

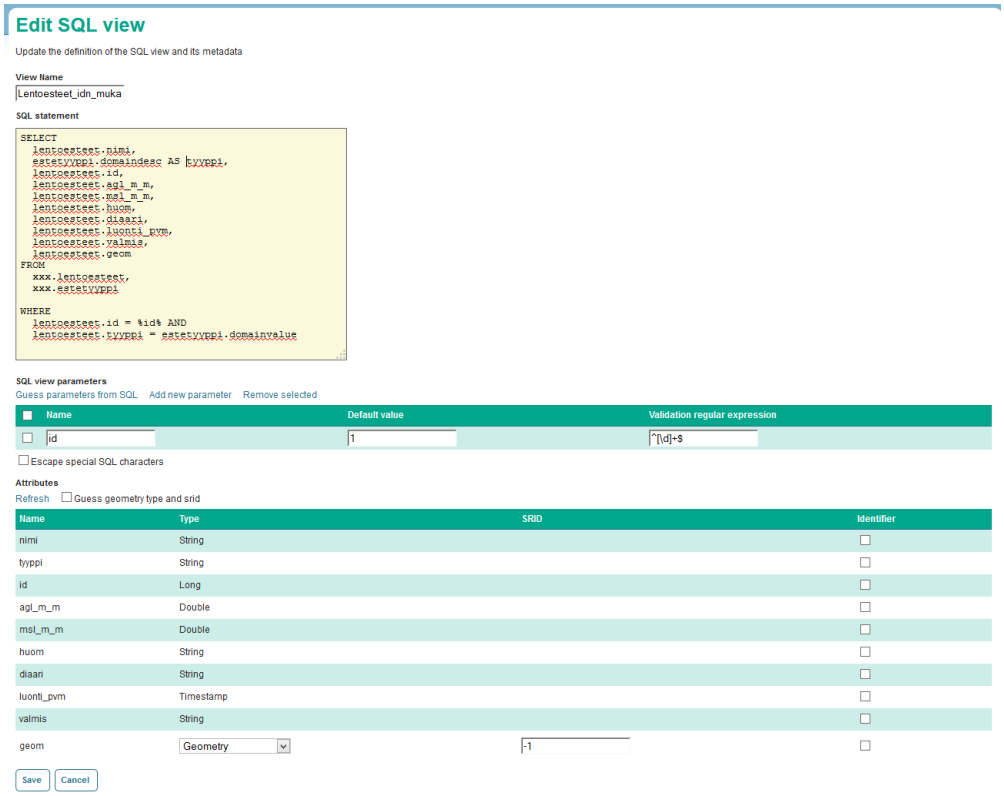
Attributes

Refresh Guess geometry type and srid

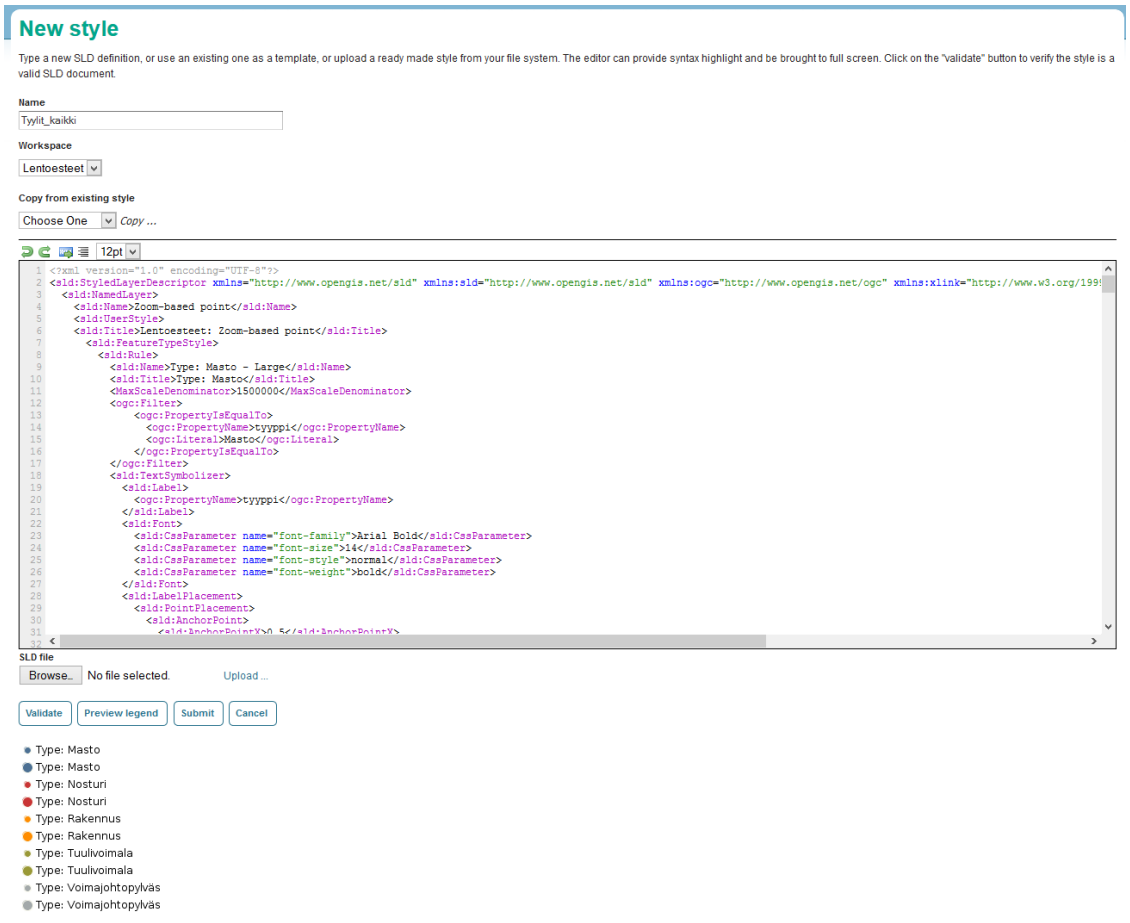
Name	Type	SRID	Identifier
nimi	String		
tyyppi	String		
id	Long		
agl_m_m	Double		
msl_m_m	Double		
huom	String		
diaari	String		
luonti_pvm	Timestamp		
valmis	String		
geom	Geometry	-1	

Save Cancel

”Lentoesteet_idn_mukaan” tason SQL-näkymän asetukset.



Neljänneksi luotiin 2 uutta tyylitiedostoa (style) ”Tyylit_kaikki” ja ”Tyylit_id”.



New style

Type a new SLD definition, or use an existing one as a template, or upload a ready made style from your file system. The editor can provide syntax highlight and be brought to full screen. Click on the "validate" button to verify the style is a valid SLD document.

Name

Tyylit_id

Workspace

Lentoesteet

Copy from existing style

Choose One *Copy ...*

12pt

```
<?xml version="1.0" encoding="UTF-8"?>
<sld:StyledLayerDescriptor xmlns="http://www.opengis.net/sld" xmlns:sld="http://www.opengis.net/sld" xmlns:ogc="http://www.opengis.net/ogc" xmlns:xlink="http://www.w3.org/199
  <sld:NamedLayer>
    <sld:Name>Zoom-based point</sld:Name>
    <sld:UserStyle>
    <sld:Title>Lentoesteet: Zoom-based point</sld:Title>
      <sld:FeatureTypeStyle>
        <sld:Rule>
          <sld:Name>Type: Masto - Large</sld:Name>
          <sld:Title>Type: Masto</sld:Title>
          <MaxScaleDenominator>15000</MaxScaleDenominator>
          <ogc:Filter>
              <ogc:PropertyIsEqualTo>
                <ogc:PropertyName>tyyppi</ogc:PropertyName>
                <ogc:Literal>Masto</ogc:Literal>
              </ogc:PropertyIsEqualTo>
          </ogc:Filter>
          <sld:TextSymbolizer>
            <sld:Label>
              <ogc:PropertyName>tyyppi</ogc:PropertyName>
            </sld:Label>
            <sld:Font>
              <sld:CssParameter name="font-family">Arial Bold</sld:CssParameter>
              <sld:CssParameter name="font-size">14</sld:CssParameter>
              <sld:CssParameter name="font-style">normal</sld:CssParameter>
              <sld:CssParameter name="font-weight">bold</sld:CssParameter>
            </sld:Font>
            <sld:LabelPlacement>
              <sld:PointPlacement>
                <sld:AnchorPoint>
                  <sld:AnchorPointX>0.5</sld:AnchorPointX>
```

SLD file

Browse... No file selected. Upload ...

Validate | Preview legend | Submit | Cancel

- Type: Masto
- Type: Masto
- Type: Nosturi
- Type: Nosturi
- Type: Rakennus
- Type: Rakennus
- Type: Tuulivoimala
- Type: Tuulivoimala
- Type: Voimajohtopylväs
- Type: Voimajohtopylväs

Liite 6. Lentoesterekisteri-web-sovelluksen index.html-tiedoston sisältö

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="initial-scale=1,user-scalable=no,maximum-scale=1,width=device-width">
    <meta name="mobile-web-app-capable" content="yes">
    <meta name="apple-mobile-web-app-capable" content="yes">
    <link rel="stylesheet" href="src/bootstrap/css/bootstrap.css">
    <link rel="stylesheet" href="src/font-awesome/css/font-awesome.css">
    <link rel="stylesheet" href="src/ol3/ol.css">
    <link rel="stylesheet" href="src/css/Popup.css">
    <link rel="stylesheet" href="src/css/LayersControl.css">
    <style>
      .layers-control {
        position: fixed;
        bottom: 10px;
        top: auto;
      }
      html, body, #map {
        height: 100%;
        width: 100%;
        overflow: hidden;
      }
      body {
        padding-top: 50px;
      }
      .navbar .navbar-brand {
        font-weight: bold;
        font-size: 25px;
        color: white;
      }
      #popup-content {
        max-height: 200px;
        overflow-y: auto;
      }
    </style>
    <script src="src/ol3/ol-whitespace.js"></script>
    <script src="src/jquery/jquery.js"></script>
    <script src="src/bootstrap/js/bootstrap.js"></script>
    <script src="src/bootbox/bootbox.min.js"></script>
    <title>Finavia Oyj - Lentoesterekisteri</title>
  </head>
  <body>
    <div class="navbar navbar-inverse navbar-fixed-top" role="navigation">
      <div class="navbar-header">
        <button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-collapse">
          <span class="icon-bar"></span>
          <span class="icon-bar"></span>
          <span class="icon-bar"></span>
        </button>
        <a class="navbar-brand" href="#">Lentoesterekisteri</a>
      </div>
      <div class="navbar-collapse collapse">
        <ul class="nav navbar-nav">
```

```
          <li class="dropdown">
            <a id="airportsDrop" href="#" role="button" class="dropdown-toggle" data-toggle="dropdown">
            <i class="fa fa-plane" style="color: white"></i>  Lentokentät<b class="caret"></b></a>
            <ul class="dropdown-menu" role="menu">
                <li><a href="#" id="pan-to-EFHK" data-toggle="collapse"><i class="fa fa-search-plus"></i>  EFHK</a></li>
                <li><a href="#" id="pan-to-EFHF" data-toggle="collapse"><i class="fa fa-search-plus"></i>  EFHF</a></li>
                <li><a href="#" id="pan-to-EFTU" data-toggle="collapse"><i class="fa fa-search-plus"></i>  EFTU</a></li>
                <li><a href="#" id="pan-to-EFLP" data-toggle="collapse"><i class="fa fa-search-plus"></i>  EFLP</a></li>
                <li><a href="#" id="pan-to-EFPO" data-toggle="collapse"><i class="fa fa-search-plus"></i>  EFPO</a></li>
                <li><a href="#" id="pan-to-EFSA" data-toggle="collapse"><i class="fa fa-search-plus"></i>  EFSA</a></li>
                <li><a href="#" id="pan-to-EFVA" data-toggle="collapse"><i class="fa fa-search-plus"></i>  EFVA</a></li>
                <li><a href="#" id="pan-to-EFJO" data-toggle="collapse"><i class="fa fa-search-plus"></i>  EFJO</a></li>
                <li><a href="#" id="pan-to-EFVR" data-toggle="collapse"><i class="fa fa-search-plus"></i>  EFVR</a></li>
                <li><a href="#" id="pan-to-EFKK" data-toggle="collapse"><i class="fa fa-search-plus"></i>  EFKK</a></li>
                <li><a href="#" id="pan-to-EFSI" data-toggle="collapse"><i class="fa fa-search-plus"></i>  EFSI</a></li>
                <li><a href="#" id="pan-to-EFOU" data-toggle="collapse"><i class="fa fa-search-plus"></i>  EFOU</a></li>
                <li><a href="#" id="pan-to-EFKI" data-toggle="collapse"><i class="fa fa-search-plus"></i>  EFKI</a></li>
                <li><a href="#" id="pan-to-EFMI" data-toggle="collapse"><i class="fa fa-search-plus"></i>  EFMI</a></li>
                <li><a href="#" id="pan-to-EFMA" data-toggle="collapse"><i class="fa fa-search-plus"></i>  EFMA</a></li>
                <li><a href="#" id="pan-to-EFET" data-toggle="collapse"><i class="fa fa-search-plus"></i>  EFET</a></li>
                <li><a href="#" id="pan-to-EFKT" data-toggle="collapse"><i class="fa fa-search-plus"></i>  EFKT</a></li>
                <li><a href="#" id="pan-to-EFIV" data-toggle="collapse"><i class="fa fa-search-plus"></i>  EFIV</a></li>
                <li><a href="#" id="pan-to-EFKE" data-toggle="collapse"><i class="fa fa-search-plus"></i>  EFKE</a></li>
                <li><a href="#" id="pan-to-EFKS" data-toggle="collapse"><i class="fa fa-search-plus"></i>  EFKS</a></li>
                <li role="presentation" class="divider"></li>
                <li><a href="#" id="pan-to-EFJY" data-toggle="collapse"><i class="fa fa-search-plus"></i>  EFJY</a></li>
                <li><a href="#" id="pan-to-EFKU" data-toggle="collapse"><i class="fa fa-search-plus"></i>  EFKU</a></li>
                <li><a href="#" id="pan-to-EFRO" data-toggle="collapse"><i class="fa fa-search-plus"></i>  EFRO</a></li>
                <li><a href="#" id="pan-to-EFTP" data-toggle="collapse"><i class="fa fa-search-plus"></i>  EFTP</a></li>
                <li role="presentation" class="divider"></li>
                <li><a href="#" id="pan-to-EFHA" data-toggle="collapse"><i class="fa fa-search-plus"></i>  EFHA</a></li>
                <li><a href="#" id="pan-to-EFKA" data-toggle="collapse"><i class="fa fa-search-plus"></i>  EFKA</a></li>
                <li><a href="#" id="pan-to-EFUT" data-toggle="collapse"><i class="fa fa-search-plus"></i>  EFUT</a></li>
                <li role="presentation" class="divider"></li>
                <li><a href="#" id="zoom-to-full" data-toggle="collapse"><i class="fa fa-search-minus">
                </i>  Tarkenna täyteen kokoon</a></li>
            </ul>
          </li>
          <li class="dropdown">
                <a id="filterDrop" href="#" role="button" class="dropdown-toggle" data-toggle="dropdown">
                <i class="fa fa-sort" style="color: white"></i>  Näytä<b class="caret"></b></a>
                <ul class="dropdown-menu" role="menu">
                <li><a href="#" id="Masto" data-toggle="collapse"><i class="fa fa-caret-right"></i>  Mastot</a></li>
                <li><a href="#" id="Nosturi" data-toggle="collapse"><i class="fa fa-caret-right"></i>  Nosturit</a></li>
                <li><a href="#" id="Rakennus" data-toggle="collapse"><i class="fa fa-caret-right"></i>  Rakennukset</a></li>
                <li><a href="#" id="Tuulivoimala" data-toggle="collapse"><i class="fa fa-caret-right"></i>  Tuulivoimalat</a></li>
                <li><a href="#" id="Voimajohtopylvas" data-toggle="collapse"><i class="fa fa-caret-right"></i>  Voimajohtopylväät</a></li>
                <li role="presentation" class="divider"></li>
                <li><a href="#" id="Kaikki" data-toggle="collapse"><i class="fa fa-caret-right"></i>  Näytä kaikki</a></li>
                </li>
        </ul>
        </li>
```

```
          <li class="dropdown">
                <a id="searchDrop" href="#" role="button" class="dropdown-toggle" data-toggle="dropdown">
                <i class="fa fa-search" style="color: white"></i>  Etsi<b class="caret"></b></a>
                <ul class="dropdown-menu" role="menu">
                <li><a href="#" id="search-by-id" data-toggle="collapse"><i class="fa fa-arrow-right"></i>  ID:n mukaan</a></li>
                </li>
        </ul>
        </ul>
      </div><!--/.navbar-collapse -->
    </div>
    <div id="map">
      <div id="popup" class="ol-popup">
      </div>
    </div>
    <script src="lib/app.js"></script>
  </body>
</html>
```

Liite 7. Lentoesterekisteri-web-sovelluksen app.js-tiedoston sisältö

```
/**
 * Add all your dependencies here.
 *
 * @require Popup.js
 * @require LayersControl.js
 */

// ========= config section ===============================================
var url = 'http://192.168.1.20:8080/geoserver/ows?';
var featurePrefix = 'Lentoesteet';
var featureType = 'Lentoesteet_kaikki';
var featureTypeSort = 'Lentoesteet_tyypin_mukaan';
var featureTypeId = 'Lentoesteet_idn_mukaan';
var layerTitle = 'Lentoesteet';
var infoFormat = 'application/json';
var center = [2812316,9583000,2831425,9594466];
var zoom = 5;
var zoom_bookmark = 12;
var styleId = 'Tyyppi_teema_id';
var styleAll = 'Tyyppi_teema_testi';
// ========================================================================

// ========= bookmarks section ============================================
var EFHK = [2776905.555, 8468705.106, 2780905.555, 8472705.106];
var EFHF = [2785903.881, 8454481.675, 2789903.881, 8458481.675];
var EFTU = [2476157.398, 8513237.62, 2480157.398, 8517237.62];
var EFLP = [3131489.055, 8634354.816, 3135489.055, 8638354.816];
var EFJY = [2855880.549, 8952818.619, 2859880.549, 8956818.619];
var EFOU = [2820567.533, 9587738.386, 2824567.533, 9591738.386];
var EFRO = [2873475.213, 10030288.18, 2877475.213, 10034288.18];
var EFKT = [2764072.892, 10356013.08, 2768072.892, 10360013.08];
var EFIV = [3049700.152, 10628956.21, 3053700.152, 10632956.21];
var EFET = [2605937.371, 10554159.29, 2609937.371, 10558159.29];
var EFHA = [2757517.411, 8823170.2, 2761517.411, 8827170.2];
var EFJO = [3295778.071, 9015094.085, 3299778.071, 9019094.085];
var EFKA = [2564068.873, 9128761.807, 2568068.873, 9132761.807];
var EFKE = [2734758.759, 9814666.123, 2738758.759, 9818666.123];
var EFKI = [3080158.401, 9420294.848, 3084158.401, 9424294.848];
var EFKK = [2573840.251, 9277084.692, 2577840.251, 9281084.692];
var EFKS = [3252085.17, 9872185.534, 3256085.17, 9876185.534];
var EFKU = [3092063.402, 9100362.682, 3096063.402, 9104362.682];
var EFMA = [2212855.88, 8424937.662, 2216855.88, 8428937.662];
var EFMI = [3025890.15, 8783160.186, 3029890.15, 8787160.186];
var EFPO = [2424517.523, 8730543.182, 2428517.523, 8734543.182];
var EFSA = [3220142.661, 8843587.187, 3224142.661, 8847587.187];
var EFSI = [2539640.429, 9023514.805, 2543640.429, 9027514.805];
var EFTP = [2623779.411, 8719806.892, 2627779.411, 8723806.892];
var EFUT = [2996730.627, 8600071.259, 3000730.627, 8604071.259];
var EFVA = [2420775.951, 9109361.739, 2424775.951, 9113361.739];
var EFVR = [3100319.598, 8897830.515, 3104319.598, 8901830.515];
// ========================================================================

```

```
52
53  // override the axis orientation for WMS GetFeatureInfo
54  ol.proj.addProjection(
55      new ol.proj.EPSG4326('http://www.opengis.net/gml/srs/epsg.xml#4326', 'enu')
56  );
57
58  // create a GeoJSON format to read WMS GetFeatureInfo response
59  var format = new ol.format.GeoJSON({featureType: featureType});
60
61  // create a new popup with a close box
62  // the popup will draw itself in the popup div container
63  // autoPan means the popup will pan the map if it's not visible (at the edges of the map).
64  var popup = new app.Popup({
65    element: document.getElementById('popup'),
66    closeBox: true,
67    autoPan: true
68  });
69
70  // the tiled WMS source for GeoServer layer
71  var wmsSource = new ol.source.TileWMS({
72    url: url,
73    params: {'LAYERS': featurePrefix + ':' + featureType, 'FORMAT':'image/png8', 'TILED': true},
74    serverType: 'geoserver'
75    });
76
77  // create a vector layer to contain the feature to be highlighted
78  var highlight = new ol.layer.Vector({
79    style: new ol.style.Style({
80      stroke: new ol.style.Stroke({
81        color: '#00FFFF',
82        width: 3
83      })
84    }),
85    source: new ol.source.Vector()
86  });
87
88  // when the popup is closed, clear the highlight
89  $(popup).on('close', function() {
90    highlight.getSource().clear();
91  });
```

```
92
93  // Pan to airport
94   moveToAirport('pan-to-EFHK', EFHK);
95   moveToAirport('pan-to-EFHF', EFHF);
96   moveToAirport('pan-to-EFTU', EFTU);
97   moveToAirport('pan-to-EFLP', EFLP);
98   moveToAirport('pan-to-EFJY', EFJY);
99   moveToAirport('pan-to-EFOU', EFOU);
100  moveToAirport('pan-to-EFRO', EFRO);
101  moveToAirport('pan-to-EFKU', EFKU);
102  moveToAirport('pan-to-EFTP', EFTP);
103  moveToAirport('pan-to-EFHA', EFHA);
104  moveToAirport('pan-to-EFJO', EFJO);
105  moveToAirport('pan-to-EFKA', EFKA);
106  moveToAirport('pan-to-EFKE', EFKE);
107  moveToAirport('pan-to-EFKI', EFKI);
108  moveToAirport('pan-to-EFKK', EFKK);
109  moveToAirport('pan-to-EFKS', EFKS);
110  moveToAirport('pan-to-EFMA', EFMA);
111  moveToAirport('pan-to-EFMI', EFMI);
112  moveToAirport('pan-to-EFPO', EFPO);
113  moveToAirport('pan-to-EFSA', EFSA);
114  moveToAirport('pan-to-EFSI', EFSI);
115  moveToAirport('pan-to-EFUT', EFUT);
116  moveToAirport('pan-to-EFVA', EFVA);
117  moveToAirport('pan-to-EFVR', EFVR);
118  moveToAirport('pan-to-EFET', EFET);
119  moveToAirport('pan-to-EFKT', EFKT);
120  moveToAirport('pan-to-EFIV', EFIV);
121
122  function moveToAirport(selector, airport) {
123     document.getElementById(selector).addEventListener('click', move(airport), false);
124  }
125
126 function move(coordinates) {
127  return function () {
128      if (map.getView().getView2D().getZoom() <= 8) {
129           var pan = ol.animation.pan({
130             duration: 2000,
131             source: /** @type {ol.Coordinate} */ (map.getView().getView2D().getCenter())
132           });
133           var zoom_map = ol.animation.zoom({
134           duration: 2000,
135           resolution: map.getView().getView2D().getResolution(),
136           source: /** @type {ol.Coordinate} */ (map.getView().getView2D().getZoom())
137           });
138           map.beforeRender(pan,zoom_map);
139           map.getView().getView2D().setCenter(coordinates);
140           map.getView().getView2D().setZoom(zoom_bookmark);
141
```

```
    } else if (map.getView().getView2D().getZoom() > 10 && map.getView().getView2D().getZoom() < 12) {
        var duration = 2000;
        var start = +new Date();
        var pan = ol.animation.pan({
        duration: duration,
        source: /** @type {ol.Coordinate} */ (map.getView().getView2D().getCenter()),
        start: start
        });
        var bounce = ol.animation.bounce({
        duration: duration,
        resolution: 15 * map.getView().getView2D().getResolution(),
        start: start
        });
        map.beforeRender(pan, bounce);
        map.getView().getView2D().setCenter(coordinates);
        map.getView().getView2D().setZoom(zoom_bookmark);

      } else {
          var duration = 2000;
          var start = +new Date();
          var pan = ol.animation.pan({
            duration: duration,
            source: /** @type {ol.Coordinate} */ (map.getView().getView2D().getCenter()),
            start: start
          });
          var bounce = ol.animation.bounce({
            duration: duration,
            resolution: 15 * map.getView().getView2D().getResolution(),
            start: start
          });
          map.beforeRender(pan, bounce);
          map.getView().getView2D().setCenter(coordinates);
          map.getView().getView2D().setZoom(zoom_bookmark);
      }
  }
}

var zoomTofull = document.getElementById('zoom-to-full');
zoomTofull.addEventListener('click', function() {
  var pan = ol.animation.pan({
    duration: 2000,
    source: /** @type {ol.Coordinate} */ (map.getView().getView2D().getCenter())
  });
  var zoom_map = ol.animation.zoom({
  duration: 2000,
  resolution: map.getView().getView2D().getResolution(),
  source: /** @type {ol.Coordinate} */ (map.getView().getView2D().getZoom())
  });
  map.beforeRender(pan,zoom_map);
  map.getView().getView2D().setCenter(center);
  map.getView().getView2D().setZoom(zoom);
}, false);

```

```
// Sort by type
sortByType('Masto');
sortByType('Nosturi');
sortByType('Rakennus');
sortByType('Tuulivoimala');
sortByType('Voimajohtopylväs');

function sortByType(selector) {
  document.getElementById(selector).addEventListener('click', showType(selector), false);
}

function showType(type) {
  return function() {
    var sortType = type;
    wmsSource.updateParams({'LAYERS': featurePrefix + ':' + featureTypeSort,
    viewparams:"tyyppi:" + sortType, 'STYLES':styleAll, 'FORMAT':'image/png8', 'TILED': true});
    }
  }

var sortByAll = document.getElementById('Kaikki');
 sortByAll.addEventListener('click', function() {
    wmsSource.updateParams({'LAYERS': featurePrefix + ':' + featureType,
    'STYLES':styleAll, 'FORMAT':'image/png8', 'TILED': true});
 }, false);

var searchById = document.getElementById('search-by-id');
 searchById.addEventListener('click', function() {
 bootbox.prompt("Syötä esteen ID-numero", function(typeId) {
    if (typeId === null || typeId === false || typeId.length === 0) {
      console.log('Peruutettu');
      } else {
        wmsSource.updateParams({'LAYERS': featurePrefix + ':' + featureTypeId,
        viewparams:"id:" + typeId, 'STYLES':styleId, 'FORMAT':'image/png8', 'TILED': true});

      }
    })
  }, false);

// create the OpenLayers Map object
// we add a layer switcher to the map with two groups:
// 1. background, which will use radio buttons
// 2. default (overlays), which will use checkboxes

var map = new ol.Map({
  controls: ol.control.defaults().extend([
    new app.LayersControl({
      groups: {
        background: {
          title: "Base Layers",
          exclusive: true
        },
        'default': {
          title: "Overlays"
        }
      }
    })
```

```
251     ]),
252     // add the popup as a map overlay
253     overlays: [popup],
254     // render the map in the 'map' div
255     target: document.getElementById('map'),
256     // use the Canvas renderer
257     renderer: 'canvas',
258     layers: [
259       // MapQuest streets
260       new ol.layer.Tile({
261         title: 'Street Map',
262         group: "background",
263         source: new ol.source.MapQuest({layer: 'osm'})
264       }),
265     // Kapsi kartat - Peruskarttarasteri
266      new ol.layer.Tile({
267      title: "Kapsi-Peruskarttarasteri",
268      group: "background",
269      visible: false,
270        source: new ol.source.TileWMS({
271                 url: 'http://tiles.kartat.kapsi.fi/peruskartta?',
272               })
273      }),
274       // Kapsi kartat - Taustakartta
275      new ol.layer.Tile({
276      title: "Kapsi-Taustakartta",
277      group: "background",
278      visible: false,
279        source: new ol.source.TileWMS({
280                 url: 'http://tiles.kartat.kapsi.fi/taustakartta?',
281               })
282      }),
283     // Kapsi kartat - Ortoilmakuva
284      new ol.layer.Tile({
285      title: "Kapsi-Ortoilmakuva",
286      group: "background",
287      visible: false,
288        source: new ol.source.TileWMS({
289                 url: 'http://tiles.kartat.kapsi.fi/ortokuva?',
290               })
291      }),
292       new ol.layer.Tile({
293         title: layerTitle,
294         source: wmsSource
295       }),
296       highlight
297     ],
298     // initial center and zoom of the map's view
299     view: new ol.View2D({
300       center: center,
301       zoom: zoom
302     })
303   });
```

```
304
305   // register a single click listener on the map and show a popup
306   // based on WMS GetFeatureInfo
307 map.on('singleclick', function(evt) {
308     var viewResolution = map.getView().getView2D().getResolution();
309     var url = wmsSource.getGetFeatureInfoUrl(
310         evt.coordinate, viewResolution, map.getView().getView2D().getProjection(),
311         {'INFO_FORMAT': infoFormat});
312     if (url) {
313       if (infoFormat == 'text/html') {
314         popup.setPosition(evt.coordinate);
315         popup.setContent('<iframe seamless frameborder="0" src="' + url + '"></iframe>');
316         popup.show();
317       } else {
318         $.ajax({
319           url: url,
320           success: function(data) {
321             var features = format.readFeatures(data);
322         console.log(features, data)
323             highlight.getSource().clear();
324             if (features && features.length >= 1 && features[0]) {
325               var feature = features[0];
326               var html = '<table class="table table-striped table-bordered table-condensed">';
327               var values = feature.getProperties();
328               var hasContent = false;
329               for (var key in values) {
330                 if (key !== 'geom' && key !== 'boundedBy') {
331                   html += '<tr><td>' + key + '</td><td>' + values[key] + '</td></tr>';
332                   hasContent = true;
333                 }
334               }
335               if (hasContent === true) {
336                 popup.setPosition(evt.coordinate);
337                 popup.setContent(html);
338                 popup.show();
339               }
340               feature.getGeometry().transform('EPSG:4326', 'EPSG:3857');
341               highlight.getSource().addFeature(feature);
342             } else {
343               popup.hide();
344             }
345           }
346         });
347       }
348     } else {
349       popup.hide();
350     }
351 });
```