jamk.fi

Organisaation paikkatietodatan visualisointi sekä ajoneuvokannan seuranta

Samuli Rasmus

Opinnäytetyö
Joulukuu 2018
Tekniikan ja liikenteen ala
Insinööri (AMK), Ohjelmistotekniikan koulutusohjelma

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

jamk.fi

Kuvailulehti

Tekijä(t)	Julkaisun laji	Päivämäärä
Rasmus, Samuli	Opinnäytetyö, AMK	Joulukuu 2018
	Sivumäärä 64	Julkaisun kieli Suomi
		Verkkojulkaisulupa myönnetty: x

Työn nimi
Organisaation paikkatietodata visualisointi sekä ajoneuvokannan seuranta

Tutkinto-ohjelma
Ohjelmistotekniikan koulutusohjelma

Työn ohjaaja(t)
Ari Rantala

Toimeksiantaja(t)
Admicom Finland Oy

Tiivistelmä

Opinnäytetyön tavoitteena oli toteuttaa Admicom Finland Oy:lle testisovellus, jonka avulla voitiin tarkastella kartalla eri laitteita, kohteita ja kuvia sekä seurata kalliimpia liikkuvia koneita reaaliajassa. Näille kartalla näytettäville objekteille määriteltiin myös erilaisia toiminnallisuuksia, kuten esimerkiksi reitinhaku valittuun kohteeseen, reittitiedon generointi ja liikkuvien koneiden kulkeman reitin tarkastelu. Työssä vaadittiin myös erilaisia rajauksia, kuten käyttäjäoikeuksien määrittämistä käyttäjän roolin perusteella. Toteutettavan sovelluksen lisäksi Admicomille pyrittiin löytämään paras karttaratkaisu, joka voitaisiin liittää osaksi olemassa olevaa Adminet-kokonaisuutta.

Sekä karttasovellus että seurantasovellus toteutettiin käytettäväksi selainympäristössä käyttäen PHP- sekä JavaScript-ohjelmointikieliä. Sovellukset toteutettiin alhaalta ylöspäin, jolloin kaikki seuraavassa osa-alueessa tarvittavat toiminnallisuudet sekä sovelluksen osat olivat valmiina ennen siihen siirtymistä.

Toimeksiantajalle sopivimmat karttarajapinnat seulottiin nykyisten sekä tulevien vaatimusten avulla, minkä lisäksi toimeksiantaja oli jo itse valinnut muutaman potentiaalisen vaihtoehdon. Kun suurin osa vaihtoehdoista oli saatu rajattua pois, aloitettiin jäljelle jääneitä tutkimaan tarkemmin sekä vertailemaan keskenään. Vertailun tuloksena valikoitui kaksi potentiaalisinta vaihtoehtoa, joista toinen valittiin opinnäytetyössä käytetyksi karttarajapinnaksi.

Opinnäytetyön lopputuloksena syntyi vaatimusmäärittelyn mukainen sovellus, joka todisti valitun karttarajapinnan toimivuuden osana Adminet-kokonaisuutta. Konkreettisen sovelluksen lisäksi työn tuloksena syntyi myös tietopohja, jonka perusteella tarkempia tutkimuksia aiheeseen liittyen voitaisiin toteuttaa.

Avainsanat (asiasanat)
PHP, JavaScript, Karttarajapinta, Verkkosovelluskehitys

Muut tiedot

jamk.fi

Description

Author(s)	Type of publication	Date
Rasmus, Samuli	Bachelor's thesis	December 2018
		Language of publication: Finnish
	Number of pages: 64	Permission for web publication: x

Title of publication

Visualization of organization's geographical data and tracking of vehicle base

Degree programme

Software Engineering

Supervisor(s)

Rantala Ari

Assigned by

Admicom Finland Oy

Abstract

The aim of the thesis was to implement a test application for Admicom Finland Oy with which it was possible to view devices, worksites, images and real time movement of vehicles on a map. Different functionalities were also defined to these objects showed on the map, such as route search to a specified worksite and examination of the path of the moving vehicles. Different kind of restrictions were also required in the project, such as defining user access depending on the role of the user. In addition to the application, the aim was to find the best map solution for Admicom that could be added to the existing Adminet system.

The applications for both map and tracking were implemented to be used in a browser environment utilizing PHP and JavaScript programming languages. The applications were executed using a bottom-up approach allowing for all functionalities and parts used in the following area of the application to be completed before moving to that area.

The most suitable web mapping APIs for the assignor were screened with the help of the current as well as future requirements, in addition to which the assignor had already chosen a few potential options. When the majority of the options had been filtered out, the remaining choices were then examined more closely as well as compared between each other. As a result of the comparisons, two potential options were selected, from which one was chosen as the web mapping API used in the thesis.

The thesis resulted in an application that was in accordance with the requirement-specifications, which demonstrated the functionality of the web mapping API as a part of the Adminet system. In addition to a concrete application, a knowledge base was produced as a result of the work, based on which further research on the topic could be conducted in the future.

Keywords/tags (subjects)

PHP, JavaScript, Web mapping API, Web application development

Miscellaneous

Sisältö

Kuviot

Sanasto

Ajax

Asynchronous JavaScript And XML. Joukko web-sovelluskehityksen tekniikoita, joilla mahdollistetaan sivun päivitys asynkronisesti.

API

Application Programming Interface. Suomenkieliseltä nimeltään ohjelmointirajapinta. Toimii kahden ohjelman välissä mahdollistaen viestinnän näiden ohjelmien kesken.

CSS

Cascading Style Sheets. Tyylimäärittelykieli, jota käytetään merkkauskielellä tehtyjen dokumenttien tyylin ehostamiseen sekä määrittelyyn.

Debuggeri

Debuggeri eli virheiden jäljittäjä on tietokoneohjelma, jota ohjelmoijat käyttävät virheiden etsimiseen. Monet debuggerit mahdollistavat koodin läpikäynnin jopa askel askeleelta.

Geokoodaus

Geokoodaus tarkoittaa prosessia, jossa haluttu osoite tai paikkanimi muutetaan karttakoordinaateiksi.

Github

Verkkopohjainen versionhallinta- sekä yhteistyöalusta sovelluskehittäjille.

GPS

Global Positioning System. Yksisuuntainen satelliittipaikannusjärjestelmä, joka mahdollistaa laitteen sijainnin paikantamisen reaaliajassa.

HTML

Hypertext Markup Language. Standardi merkkauskieli, jota käytetään verkkosovellusten luontiin. Uusin versio kyseisestä merkkauskielestä on HTML5.

JQuery

Avoimen lähdekoodin JavaScript-kirjasto.

JSON

JavaScript Object Notation. Avoimen standardin tiedostoformaatti, joka sisältää tekstiä, jota ihmisetkin ymmärtävät. Sitä käytetään mm. joidenkin rajapintojen kautta keskusteltaessa.

Karttalaatta

Tarkoittaa neliön muotoista palaa kartasta, joka muodostaa yhdessä muiden palojen kanssa kokonaisen kartan.

Karttalaattapalvelin

Tarkoittaa palvelinta, joka palauttaa halutut karttalaatat.

Karttaobjekti

Tarkoitetaan tässä kontekstissa kartalla näytettäviä objekteja. Näitä ovat esimerkiksi ikonit, jotka on sisällytetty Heren kirjaston merkkaaja-luokalla tehtyyn olioon.

Karttapalvelin

Tarkoitetaan tässä kontekstissa palvelimia, joita tarvitaan kartan näyttämiseen. Sisältävät esimerkiksi kartan eri palasia.

Karttarajapinta

Tarkoitetaan tässä kontekstissa karttarajapintatarjoajan tarjoamaa rajapintaa karttapalasten noutamiseen sekä muiden tarjottujen ominaisuuksien käyttöön.

Käänteinen geokoodaus

Tarkoittaa prosessia, jossa karttakoordinaatit muutetaan osoitteiksi tai paikkanimiksi.

MVC

Arkkitehtuurimalli, jossa sovellus jaetaan kolmeen loogiseen komponenttiin, joita ovat malli, näkymä ja ohjain.

SDK

Software Development Kit. Kokoelma koodia, jota käytetään sovellusten kehittämiseen. Sisältävät usein esimerkiksi valmiita kirjastoja, jotka helpottavat sovelluksen rakentamista.

Stack Overflow

Verkkosivusto, jolla ohjelmistokehittäjät esittävät ohjelmointityöhön perustuvia kysymyksiä ja vastaavat niihin.

SVG

Scalable Vector Graphics. Kuvauskieli, jolla määritellään kaksiulotteista vektorigrafiikkaa verkkosovelluksiin.

SWOT-analyysi

Suunnittelutekniikka, jolla pyritään tunnistamaan sekä ymmärtämään tutkittavan kohteen vahvuudet, heikkoudet, mahdollisuudet ja uhat.

UML

Unified Modeling Language. Mallinnuskieli, jota käytetään sovelluksen kuvaamiseen sille tehtyjen kaavioiden avulla.

1 Reaaliaikanen tieto sekä karttasovellukset nyky-yhteiskunnassa

Nykypäivänä ei voi olla kiinnittämättä huomiota siihen, kuinka tärkeää eri sijaintidatojen seuraamisesta on tullut sekä yksityishenkilölle että yrityksille. Enää ei riitä, että saadaan summittaista dataa jonkin tietyn asian, esineen tai henkilön olinpaikasta, vaan tiedon halutaan olevan mahdollisimman tarkkaa sekä reaaliaikaista. Tämä kehityssuunta on huomattavissa aina pienistä viihdesovelluksista suuriin yritysovelluksiin.

Miksi reaaliaikaisuudesta ja sijainnin tarkkuudesta on tullut nykyään niin tarpeellista? Yhtenä suurimmista syistä voidaan pitää tekniikan ja teollisuuden kehittymistä, jonka myötä tarve rikkaampaan ja tarkempaan sijaintitietodataan on kasvanut. Ajatellaan esimerkiksi kulkuneuvojen ajon automatisointia. Jotta kyseinen toiminto olisi mahdollista toteuttaa, ei pelkästään GPS-laitteilla hankittu data enää riitä vaadittuun tarkkuuteen, vaan ajoneuvon sijainnista sekä ympäristöstä on kerättävä tietoa esimerkiksi kameroilla ja sensoreilla. Kun kaikki tämä kerätty data integroidaan sekä käytetään tarkkuuden lisäämiseen luotuja algoritmeja, saadaan muodostettua tarkempi kokonaiskäsitys ympäristöstä, mikä mahdollistaa auton sijainnin tarkan määrityksen. (Silver 2017.)

Tarve sovellusten sijaintidatan tarkkuuden sekä reaaliaikaisuuden kehittämiseen ei tule pelkästään yritysmaailmasta, vaan myös kuluttajilta, jotka käyttävät päivittäin runsaasti aikaa eri sijaintien tarkasteluun. Siinä missä yksi tarkkailee saapuvia busseja kartalta, toinen voi seurata navigaattorinsa ajo-ohjeita aamuruuhkassa. Jos nämä bussien ja navigaattorin liikkeet kartalla ovat epätarkkoja, sovelluksen merkitys käyttäjälleen katoaa, mikä johtaa taas siihen, että kyseinen sovellus vaihdetaan markkinoilta löytyvään toimivampaan tuotteeseen. Tämän ovat huomanneet myös sovelluskehittäjät, jotka kilpaa valmistavat entistä tarkempia sekä reaaliaikaisempia sovelluksia. Koska tällaisten sovellusten paikkatietodatan visualisoimiseen tarvitaan kartta, on kehittäjille tärkeää löytää omia tarpeitaan vastaava karttapalvelutarjoaja.

Karttapalvelutarjoajien kesken on jo vuosien ajan ollut käynnissä kilpajuoksu siitä, kenestä tulee johtava karttapalveluiden tarjoaja. Siinä missä Google on voittanut kilpailun hakukonepalveluna ja Facebook sosiaalisen median alustana, ei samanlaista johtavaa firmaa ole kruunattu vielä karttapalveluiden joukosta. Tässä kilpajuoksussa on mukana niin suuria ja tunnettuja yrityksiä, kuin pieniä ja vasta perustettujakin. Niistä jokaisella on omat vahvuutensa, joilla ne pyrkivät erottumaan joukosta, kuten kattavat laajennusosat tai avoin lähdekoodi. Syitä karttapalveluiden laajenneelle tarjonnalle ovat edellä mainittujen asioiden lisäksi myös jatkuvasti laajenevat markkinat. Enää karttoja ei käytetä pelkästään tietyn pisteen näyttämiseen, vaan niiden päälle halutaan rakentaa erilaisia rakennelmia ja maailmoja. Yhtenä esimerkkinä voitaisiin käyttää suuren suosion saavuttanutta mobiilipeliä, Pokémon Gota.

Kehittäjän näkökulmasta reaaliaikaisten karttasovellusten luominen herättää useita kysymyksiä: Mitä kaikkea tarvittaisiin kyseisen sovelluksen luomiseen? Millaisella tietokannalla jatkuvasti muuttuvan datan ylläpitäminen onnistuisi ja mikä on tällaisen tietokannan toimintamalli? Mikä olisi paras karttapalveluntarjoaja yritykseni käyttötapausta ajatellen ja millaiset lisenssit heiltä olisi hankittava? Paljonko kyseisen sovelluksen ylläpitäminen maksaisi ja kuinka sovellus saataisiin sulautettua yrityksen jo valmiiseen sovelluskokonaisuuteen, jos tällainen on jo olemassa? Muun muassa näihin kysymyksiin pyritään vastaamaan tässä opinnäytetyössä.

2 Työn lähtökohdat

2.1 Taustat

Keväällä 2017 opinnäytetyöntekijä aloitti toimeksiantajan, Admicom Finland Oy:n, työntekijänä. Jo rekrytointivaiheessa toimeksiantaja ja opinnäytetyöntekijä keskustelivat opinnäytetyön tekemisen mahdollisuudesta yrityksessä. Vuotta myöhemmin puheet toteutuivat ja yhteistyösopimus osapuolien välille kirjoitettiin. Opinnäytetyön aihe valittiin lukuisista vaihtoehdoista, joista sekä koulun, työntekijän että toimeksiantajan näkökulmasta parhaaksi valikoitui reaaliaikaisen karttasovelluksen luominen yritykselle.

Karttasovelluksen luomista osaksi Adminet-kokonaisuutta oli harkittu jo aiemmin yrityksen sisällä, mutta tutkimuspöydälle aihetta ei vielä ollut ehditty nostaa. Tämä vuoksi sovelluksen luomiseen vaadittu tutkimus-, suunnittelu- ja toteuttamistyö soveltuivat erittäin hyvin opinnäytetyön aiheeksi. Työhön sisällytettiin tarvittavien työkalujen sekä palveluiden valinta ja niihin tarvittavien lisenssien sekä maksujen selvitys. Tämän lisäksi tekijän oli raportoitava yritykselle, miksi kyseiset tuotteet oli valittu käytettäviksi sekä tutkittava kuinka kyseinen sovellus saataisiin liitettyä osaksi jo olemassa olevaan Adminet-kokonaisuutta. Myös itse karttasovellukseen suunniteltiin eri toiminnallisuuksia, jotka hyväksytettiin sekä koululle että yritykselle ennen työn aloittamista.

2.2 Toimeksiantaja

Toimeksiantajana opinnäytetyössä toimi vuonna 2004 Jyväskylässä perustettu Admicom Finland Oy. Heidän palvelukokonaisuutensa ytimessä toimii verkkopohjainen toiminnanohjausjärjestelmä nimeltään Adminet, joka on suunnattu erityisesti talotekniikan, rakentamisen ja teollisuuden aloille. Tämän lisäksi Admicom tarjoaa tilitoimistopalveluita sekä apua liiketoiminnan kehittämiseen.

Admicomilla on useita toimipisteitä, jotka sijaitsevat esimerkiksi Jyväskylässä, Vantaalla sekä Tampereella. Työntekijöitä heillä on jo yli 90. Yhtiön liikevaihto on kasvanut vuosina 2012-2017 noin 37 %:n vuosivauhtia, ja lisäksi helmikuussa vuonna 2018 yhtiö listautui pörssiin lisätäkseen yhtiön tunnettuutta, tukeakseen strategian mukaisten yritysostojen toteuttamista sekä motivoidakseen ja sitouttaakseen työntekijöitä osakkuuden sekä osakepohjaisten kannustimien kautta. (Admicom Oyj listautui pörssiin 2018; Tietoa meistä n.d.)

2.3 Tavoitteet

2.3.1 Toimeksiantajalle

Toimeksiantajan päätavoitteena oli päästä testaamaan jo idean tasolla ollutta sovellusta. Tämä onnistui hyvin opinnäytetyöntekijän avulla, koska ideaa päästiin kokeilemaan turvallisessa ympäristössä ja vähäisin kustannuksin. Työn tavoitteena oli selvit-

tää yleisimmät sudenkuopat sekä ongelmat, joita kyseisen sovelluksen kehittämisessä ilmenisi, ja kuinka kalliiksi tällaisen sovelluksen ylläpitäminen tulisi. Lisäksi opinnäytetyönä syntyvä sovellus antaisi varmistuksen sille, toimisiko kyseinen karttasovellus toimeksiantajan käyttötapauksessa. Lopputuloksena valmistuvaa sovellusta voitaisiin myös käyttää pohjana sovellukselle, jota lähdettäisiin työstämään tuotantoon asti. Opinnäytetyön ansiosta opinnäytetyöntekijä kartuttaisi tarvittavan osaamisen käytetyistä teknologioista, jotta tarvittaessa hän voisi jatkokehittää tekemäänsä sovellusta.

Toimeksiantaja halusi myös nähdä, voisiko ammattikorkeakoulussa suoritettavalla opinnäytetyöllä olla jotain hyötyä yritykselle ja voisiko opinnäytetöitä hyödyntää myös jatkossa. Lisäksi rekrytointivaiheessa opiskelijalle myönnetyllä luvalla tehdä opinnäytetyö toimeksiantajalle haluttiin lisätä opiskelijan kiinnostusta yritystä kohtaan.

2.3.2 Tekijälle

Opinnäytetyöntekijän tavoitteena oli tehdä mahdollisimman hyvä sovellus toimeksiantajan kanssa sovitusta aiheesta, jotta kyseistä sovellusta voitaisiin tarvittaessa käyttää pohjana tuotantoon tehtävälle versiolle. Lisäksi tekijälle oli tärkeää saada selvitettyä mahdollisimman tarkkaan, paljonko hintaa sovellukselle tulisi, mitä lisenssejä tarvittaisiin sekä mitä rajoituksia palveluiden käytössä olisi, jotta yllätyksiltä vältyttäisiin myöhemmissä vaiheissa.

Tekijän tavoitteena oli myös saada mahdollisimman hyvä arvosana opinnäytetyöstään ja päästä käyttämään siinä koulussa opeteltuja taitoja ja käytänteitä. Lisäksi tavoitteena oli oppia hallitsemaan sekä tekemään isompia työelämän projekteja, joissa tekijä itse vastaa kaikesta aina suunnittelusta toteutukseen ja raportointiin. Myös tietotaidon kartuttaminen uusista tuntemattomista teknologioista ja aiheista sekä edistyneemmän englannin kielen parantaminen olivat tekijän päämäärinä.

2.3.3 Muille

Oppilaitoksen eli Jyväskylän ammattikorkeakoulun tavoitteena oli valvoa ja ohjata tekijää opinnäytetyössään, jotta lopputuloksesta tulisi JAMKin kriteerit täyttävä. JAMK

mm. valvoi, oliko työn aihe riittävän laaja sekä haastava, oliko siinä tarpeeksi tutkimusnäkökulmaa ja oliko työn raportti kirjoitettu oikeaan opinnäytetyöformaattiin. Lisäksi JAMK arvioi ja antoi lopullisen arvosanan tekijälle opinnäytetyöstään.

3 Työn kuvaus

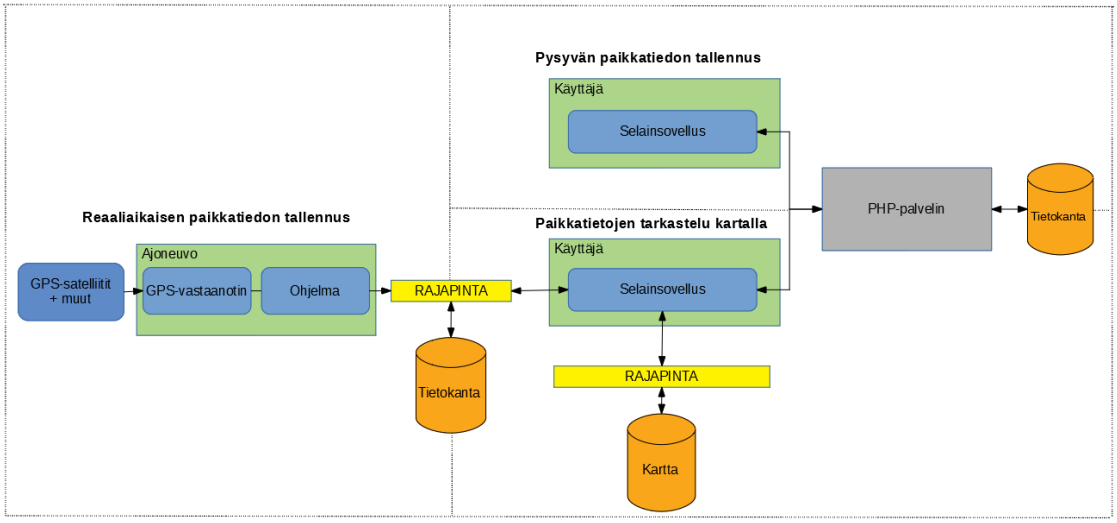
3.1 Yleistä

Adminet on monihaarainen sovelluskokonaisuus, minkä vuoksi asiakkaan tietokannan tauluihin kertyy vuosien varrella runsaasti eri tietorivejä. Jotta käyttäjät hallitsisivat tätä kertynyttä dataa paremmin, on avuksi luotu useita keinoja mm. suodattaa, lajitella ja tarkastella näitä rivejä annetuilla parametreillä. Taulusta riippuen riveihin voi olla tallennettu tietoa esimerkiksi fyysisistä asioista tai esineistä. Tällaiset rivit sisältävät yleensä myös sijaintidataa, joka kertoo, missä kyseinen esine tai asia sijaitsee. Sijaintien hahmottamisen parantamiseksi asiakkaille halutaan tarjota mahdollisuus visualisoida nämä kohteet kartalla. Tavoitteena on myös lisätä toiminnallisuutta kartalle, koska se toimii hyvänä alustana kohteiden hallinnoimiseen.

Kaikilla esineillä sekä asioilla sijainti ei ole kuitenkaan staattinen, vaan se voi muuttua. Tämä luo uusia haasteita edellä mainitun ominaisuuden luomiseen. Perinteisesti käyttäjä on itse syöttänyt osoitteen jollekin kohteelle, joka on muuttunut vain, jos käyttäjä on sen itse halunnut vaihtaa. Liikkuvien kohteiden tapauksessa sijainnin päivittäminen tällä tavoin ei kuitenkaan ole järkevää, joten uuden sijainnin vastaanottaminen sekä päivittäminen on automatisoitava. Jotta automatisointi olisi mahdollista, on hankittava komponentit viestintään eri kerrosten välillä sekä kehitettävä ohjelma, joka hoitaa tämän kaiken viestimisen. Lisäksi on mietittävä, soveltuvatko yrityksen jo käytössä olevat tietokantaratkaisut kyseisen datan tallentamiseen. Jos ne eivät sovellu, niin miksi ja minkälainen ratkaisu niiden tilalle olisi kehitettävä.

Kun kohteita halutaan tarkastella kartalla, on aluksi mietittävä, missä ympäristössä se halutaan suorittaa. Koska rakennettava sovellus liitetään osaksi Adminettiä, karttasovellus ladataan PHP-serveriltä, jonka jälkeen se näytetään verkkoselaimessa. Pohdittavaksi jää, hankitaanko kohteiden näyttämiseen tarvittava kartta joltain palveluntarjoajalta vai ylläpidetäänkö karttapalvelinta itse.

Kun kaikki edellä käydyt asiat niputetaan yhteen, saadaan karkea kuvaus, mistä osista kyseinen sovellus muodostuu (ks. kuvio 1). Tämän ansiosta sekä työn rajaus että määrittely helpottuvat ja niiden suorittamiseen tarvittava aika vähenee.



Kuvio 1. Karkea kuvaus mahdollisesta toteutuksesta

3.2 Työn rajaus

Tekijän alkuperäisenä ideana oli toteuttaa sovellus kahdella parhaaksi toteamallaan karttaratkaisulla. Tämä kuitenkin todettiin liian laajaksi opinnäytetyön aiheeksi, joten se rajattiin koskemaan vain yhtä. Myös toista karttaratkaisua päätettiin testata, mutta tämä tapahtuisi vasta opinnäytetyön valmistumisen jälkeen.

Karttasovellukseen tehtävät ominaisuudet rajattiin pakollisiin ja vapaehtoisiin ominaisuuksiin. Pakolliset ominaisuudet olivat niitä, jotka sovelluksesta pitäisi minimissään löytyä. Vapaehtoiset taas olivat niitä, jotka lisättäisiin ajan sekä muiden resurssien riittäessä. Koska osa ominaisuuksista sisälsi muuttuvaa paikkadataa, rajattiin nämä ominaisuudet päivittymään reaaliajassa. Muiden ominaisuuksien tuli päivittyä vain käyttäjän toiminnon kautta, koska näissä data oli lähes muuttumatonta.

Itse toteutustyön lisäksi opinnäytetyö rajattiin koskemaan erityisesti eri karttapalveluiden ja niissä käytettyjen ratkaisujen vertailua. Tähän sisällytettiin myös hintojen,

rajoitusten ja lisenssien selvitys, mutta koska osa näistä tiedoista on salassa pidettävää materiaalia, niitä voidaan käsitellä tässä raportissa vain sallituissa rajoissa. Tekijän tuli myös vertailla eri työkaluja sekä käytettävissä olevia teknologioita, joita hän tarvitsi työnsä toteutukseen, ja selvittää jatkuvasti muuttuvan tiedon tallennukseen mielestänsä parhaiten soveltuva ratkaisu.

Jotta sovellus saataisiin toimimaan saumattomasti Adminetissä, rajattiin opinnäytetyö koskemaan myös selvitystä, kuinka valittu karttatoteutus saataisiin liitettyä osaksi olemassa olevaa Adminettiä, mistä sovellukseen tarvittava data Adminetistä löytyy sekä kuinka karttasovelluksen ominaisuuksia saadaan rajoitettua esimerkiksi eri käyttäjäroolin perusteella. Koska karttasovellusta tullaan todennäköisesti jatkokehittämään, annettiin tekijälle myös rajaus suunnitella ja toteuttaa työ mahdollisimman järkevällä rakenteella sekä laadukkaalla ohjelmakoodilla.

3.3 Vaatimusmäärittely

3.3.1 Yleistä

Vaatimusmäärittely on ohjelmistojen kehityksessä käytetty perustehtävä. Se tarkoittaa järjestelmälle määrättäviä vaatimuksia, jotka kuvataan järjestelmän ominaisuuksina, mitkä sen pitäisi toteuttaa. Vaatimukset jaotellaan yleensä toiminnallisiin sekä ei-toiminnallisiin vaatimuksiin, ja ne tulevat eri sidosryhmiltä, jotka ovat tekemisissä järjestelmän kanssa. (Paakki 2010, 2-3.)

Siinä missä vaatimusmäärittely rajaa sekä määrittää ohjelmiston toiminnallisuutta ja selkeyttää tämän suunnittelua, se toimii myös eräänlaisena lupauksena asiakkaalle siitä, mitä järjestelmä sisältää. Samalla se myös auttaa tekijöitä hahmottamaan, mitkä ovat järjestelmältä vaaditut perustoiminnallisuudet. Seuraavaksi luvuissa 3.3.2-3.3.3 määritellään opinnäytetyötä koskevat ensisijaiset vaatimukset, jotka tarkentuvat myöhemmin.

3.3.2 Toiminnalliset vaatimukset

Toiminnallisilla vaatimuksilla ohjelmistolle määritellään, kuinka se vaikuttaa ympäristöönsä. Ne voivat myös viitata ohjelmiston toimintaympäristön tilaan vaikuttaen oh-

jelmiston toimintaan. (Paakki 2010, 27.) Usein toiminnallisten vaatimusten määrittelyssä käytetään apuna käyttäjätarinoita tai käyttötapauskaavioita, kuten myös tässä opinnäytetyössä (ks. kuvio 2).

Toteutettavan karttasovelluksen web-käyttöliittymässä käyttäjän tulee voida tarkastella valitsemiaan työmaita kartalla näille määrätyssä sijainnissa. Käyttäjän tulee myös pystyä hakemaan ajo-ohje sekä reitti haluamalleen työmaalle. Käyttäjän tulee pystyä tarkastelemaan halvempia laitteita kartalla sijainnista, joka niille on määritelty, kun laite on luettu käyttöön. Käyttäjän tulee pystyä tarkastelemaan kuvia kartalla sijainnista, joka niille on määritelty. Käyttäjän tulee voida tarkastella haluamiensa karttaobjektin tietoja klikkaamalla objektia kartalta. Käyttäjän tulee myös pystyä tyhjentämään kartta sekä muut karttatiedon käyttöön tarkoitetut elementit halutessaan. Jos käyttäjä on vastuuhenkilö, tulee hänen edellä olevien lisäksi pystyä tarkastelemaan kalliimpia laitteita reaaliajassa kartalta. Tällöin käyttäjän tulee myös pystyä tarkastelemaan kalliimpien laitteiden kulkemaa reittiä.

Kalliimmille laitteille toteutetussa seurantaan tarkoitetussa ympäristössä laitteen tulee pystyä jäljittämään oma sijaintinsa kartalla. Sijainnin muuttuessa laitteen tulee pystyä päivittämään sijaintinsa tietokantaan. Jotta laite pystyy tallentamaan sijaintinsa tietokantaan, tämän tulee pystyä tunnistautumaan käytettävään järjestelmään.

Kuvio 2. Toiminnalliset vaatimukset käyttötapauskaaviossa

3.3.3 Ei-toiminnalliset vaatimukset

Ei-toiminnallisia vaatimuksia voi olla vaikea havaita. Ne määrittelevät ehtoja, kuinka ohjelmiston tulee täyttää toiminnalliset vaatimukset. Tällaisia vaatimuksia ovat esimerkiksi laatuvaatimukset, mukautuvuusvaatimukset ja arkkitehtuurivaatimukset. (Paakki 2010, 28-30.)

Web-ympäristöön toteutetun karttasovelluksen tulee olla tietoturvallinen. Toteutuksen tulee olla skaalautuva sekä toimia ja olla muokattavissa myös Adminetin mobiiliversioon. Toteutuksen tulee olla myös Internet Explorer yhteensopiva. Toteutuksen rakenteen ja ohjelmakoodin tulee täyttää toimeksiantajan määrittelemät laatuvaatimukset. Jos karttapalvelinta ei ylläpidetä itse, tulee valitun karttapalveluntarjoajan olla tarpeeksi suuri ja luotettava yritys.

4 Työn suunnittelu ja teknologiavalinnat

4.1 Yleistä

Ohjelmistosuunnittelu on ohjelmistokehityksen vaihe, joka sijoittuu hallinnollisen päätöksenteon sekä itse ohjelmointityön väliin. Siihen sisältyy kaikki järjestelmän yleisestä suunnittelusta aina tekniseen päätöksentekoon. Ohjelmistosuunnittelulla pyritään saavuttamaan kyky tehdä päätöksiä, joilla päästään haluttuun lopputulokseen ymmärtäen sekä välttäen samalla mahdolliset tulevat ongelmatilanteet. (Kanat-Alexander 2008.)

Ohjelmistojen suunnitteluun käytettävää aikaa ja työn määrää on hankala määritellä, koska kehittäjien tietotaitotasot sekä ohjelmien luonteet vaihtelevat. Tässä työssä tekijälle tuntemattomia kokonaisuuksia oli muutamia, minkä vuoksi näihin asioihin tutustumiseen ja eri ratkaisujen vertailemiseen kului paljon aikaa. Ratkaisujen vertaileminen oli tekijälle itselleen erityisen tärkeää, jotta käyttöön valittaisiin kyseistä projektia parhaiten tukevat vaihtoehdot ja säästyttäisiin myöhemmässä vaiheessa ikäviltä ongelmatilanteilta. Luvuissa 4.2-4.7 on käyty läpi teknologioiden ja työkalujen vertailu sekä valinta.

4.2 Ohjelmointikielet

Ohjelmointikielet mahdollistavat viestimisen tietokoneille kielellä, jota ne ymmärtävät. Ohjelmointikieli esitetään ihmisten ymmärtämässä tekstimuodossa, jonka jälkeen se käännetään tietokoneen ymmärtämään binäärimuotoon. Ohjelmointikieliä on useita, joista jokaisella on ominaisuutensa. Tästä huolimatta useat ohjelmointikielet kuitenkin muistuttavat toisiaan. (Computer Programming Languages n.d.) Kaikille ohjelmointikielille on myös oma käyttötarkoituksensa. Joku ohjelmointikieli saattaa olla tarkoitettu esimerkiksi palvelinohjelmointiin soveltuvaksi, kun taas toinen käyttäjänäkymässä tapahtuvien toimintojen tueksi.

PHP

PHP (lyhenne sanoista PHP: Hypertext Preprocessor) on vapaaseen lähdekoodiin perustuva ohjelmointikieli, jota käytetään pääosin verkkopalvelimilla. PHP:ssä ohjelmakoodi suoritetaan palvelimella, minkä jälkeen vastaus lähetetään selaimelle HTML-muodossa. PHP on erittäin suosittu sekä aloittelevien ohjelmoijien että ammattilaisten keskuudessa, sillä se on helposti sisäistettävä kieli, mutta tarjoaa myös pitkälle kehitettyjä ominaisuuksia. (What is PHP? n.d.)

Java EE

Java EE (Java Platform, Enterprise Edition) on Oraclen omistama kokoelma Javan rajapintoja, joita kehittäjät käyttävät palvelinpuolen sovellusten kirjoittamiseen. Standardoitujen ja uudelleen käytettävien modulaaristen komponenttiensa sekä automaattisesti generoituvien ohjelmaosiensa ansiosta se vähentää kirjoitettavan ohjelmakoodin määrää sekä yksinkertaistaa koko kehitysprosessia. Sen arkkitehtuuri tarjoaa erilaisia palveluita, jotka helpottavat suositeltujen suunnittelumallien käyttöä sekä parhaimpien käytänteiden toteuttamista. (Rouse 2017b.)

JavaScript

JavaScript on alun perin LiveScript nimellä tunnettu kevyt ja dynaaminen ohjelmointikieli, jota käytetään usein osana verkkosivustoja, näiden interaktiivisuuden rakentamiseen. JavaScript sijoitetaan yleensä asiakasrajapintaan, josta se voi hallita HTML-dokumenttia, ilman että tätä tarvitsee päivittää verkkopalvelimen kautta. Tämä luo kuitenkin myös heikkouksia kielelle, sillä esimerkiksi tiedostojen lukeminen ja niihin kirjoittaminen on jouduttu estämään tästä johtuvista turvallisuussyistä. (What is JavaScript? n.d.)

Dart

Dart on Googlen kehittämä vapaaseen lähdekoodiin perustuva oliopohjainen ohjelmointikieli, joka on tarkoitettu sekä palvelimelle että selaimelle. Se toimii C-ohjelmointikielen kaltaisella syntaksilla ja se tukee useita ohjelmoinnissa käytettäviä apukeinoja, kuten luokkia sekä kokoelmia. Dart soveltuu esimerkiksi yksisivuisten sovellusten luomiseen. Nämä ovat verkkosovelluksia, jotka uuden sivun lataamisen sijaan

navigoivat sivulla oleviin eri näkymiin. (Dart Programming Overview n.d; Dart Programming Tutorial n.d.)

Yhteenveto

Koska työn tuloksena valmistuva karttasovellus liitetään osaksi Adminettiä, osa käytettävistä kielistä on määräytynyt jo etukäteen. Ei olisi järkevää etsiä esimerkiksi palvelinpuolella käytetylle PHP:lle kilpailevaa vaihtoehtoa, koska kaikki aikaisemmat toteutukset on rakennettu jo sen päälle. Kilpailevan vaihtoehdon liittäminen järjestelmään vaatisi lisäksi suuremmassa mittakaavassa strategian miettimistä, johon opinnäytetyön tekijällä tuskin olisi sanavaltaa.

Selainpuolen kieleksi valikoitui JavaScript, joka on myös jo käytössä oleva vaihtoehto. Päätöstä tukivat seikat, että valittu kartta- sekä epärelaatiotietokantaratkaisu tarjosivat valmiit JavaScript-kirjastot, jotka hoitivat keskustelun rajapintoihin ja tarjosivat hyödyllisiä funktioita rajapinnasta saadun datan näyttämiseen. JavaScriptin tueksi tekijä liitti työhön myös JQuery-kirjaston, joka helpotti toiminnallisuuksien luomista selainympäristössä. Selainympäristön rakentamiseen tekijä käytti HTML-kuvauskieltä, jota ehostettiin CSS-tyylimäärittelykielellä. Näiden lisäksi osa kartalla näytettävistä ikoneista haluttiin toteuttaa SVG-kuvauskieltä käyttäen animoinnin saavuttamiseksi.

4.3 Ohjelmistoympäristöt ja työkalut

Ohjelmistoympäristö tarkoittaa sovellusta, jota sovelluskehittäjät käyttävät eri tietokoneohjelmien luomiseen. Ohjelmistoympäristöihin on yleensä liitetty useita kehitystyökaluja, kuten editori lähdekoodin muokkaamiseen, debuggeri virheiden jäljittämiseen sekä kääntäjä, jolla lähdekoodi käännetään halutulle kohdekielelle. Ohjelmistoympäristön keskitetyn käyttöliittymän ansiosta sovelluskehitys tehostuu ja se mukavoittaa kehittäjän työskentelyä. (IDE 2015.)

PhpED

NuSpheren tuoteperheeseen kuuluva PhpED (ks. liite 1) on ohjelmistoympäristö, joka on suunnattu erityisesti PHP-, HTML-, CSS- ja XML-kielille. PhpED sisältää lukuisia ominaisuuksia, joiden avulla käyttäjät voivat mm. tarkastella ohjelmakoodia, julkaista

sitä salatun yhteyden yli etäpalvelimelle, eliminoida pullonkauloja koodista sekä integroida kolmansien osapuolien työkaluja osaksi ohjelmistoympäristöään. (NuSphere PhpED - The World Famous PHP IDE n.d.)

phpDesigner 8

Mpsoftwaren kehittämä phpDesigner 8 on PHP-ohjelmistoympäristö, joka on tehty erityisesti PHP-, JavaScript-, HTML5- ja CSS-kieliä ajatellen. Siihen on rakennettu lukuisia ominaisuuksia aina debuggerista PHP-ohjelmoinnin käyttöoppaisiin. PhpDesigner 8:sta on myös tarjolla versio, joka voidaan tallentaa USB-laitteille, mikä mahdollistaa koodin muokkaamisen millä koneella tahansa. (Welcome to phpDesigner 8 n.d.)

Chrome DevTools

Googlen tarjoama Chrome DevTools on Chrome-selaimeen rakennettu sarja selainsovellusten kehityksessä tarvittavia työkaluja (ks. liite 1). Sen avulla sovelluskehittäjä voi esimerkiksi tarkastella eri HTML-elementtejä ja muokata näitä editorissa, jäljittää virheitä debuggerilla sekä tarkastella konsoliin tulleita viestejä ja sovelluksen käytöstä tehtyjä raportteja. (Chrome DevTools 2018.)

Firefox Developer Tools

Firefox Developer Tools on Mozilla Firefox -selaimeen liitetty sarja työkaluja selainsovellusten kehitykseen. Firefoxin kehittäjätyökalut kattavat kaikkien perustyökalujen lisäksi muutaman erikoisemman toiminnallisuuden, kuten mahdollisuuden visualisoida työstettävä sivu 3D-muodossa. Jos käyttäjä jää kaipaamaan jotain toiminnallisuutta, on hänen mahdollista esittää tätä ideaa selaimen kehittäjille. Myös muiden käyttäjien ideoiden äänestäminen on mahdollista. (Firefox Developer Tools 2018.)

MySQL Workbench

MySQL Workbench (ks. liite 1) on avoimeen lähdekoodiin perustuva työkalu, jolla hoituu mm. palvelimen konfiguraatio, datan muotoilu, SQL-kehittäminen sekä varmuuskopiointi. Visualisoivien ominaisuuksiensa ansiosta se helpottaa esimerkiksi tietokantojen suunnittelu- sekä kehitystyötä, joka kaikki hoituu lähes samasta näkymästä. (MySQL Enterprise Edition n.d.)

Yhteenvetona

Opinnäytetyön ohjelmistoympäristöksi valittiin PhpED, koska se kattoi kaikki tarvittavat toiminnallisuudet valittujen ohjelmointikielten käsittelyyn ja oli tekijälle entuudestaan tuttu. Lisäksi toimeksiantaja oli etukäteen toimittanut tekijälle maksullisen version kyseisestä sovelluksesta, joten sen käyttö ja käyttöönotto opinnäytetyön ohjelmistoympäristönä luonnistui vaivattomasti.

Selainpuolella JavaScriptin virheenetsintään sekä HTML-elementtien tarkasteluun valittiin käyttöön Chrome DevTools. Tekijä oli jo aikaisemmin testannut molempien selainten tarjoamia kehittäjätyökaluja ja todennut Chromen työkalut tällä hetkellä toimivammiksi, sillä vastikään tulleiden päivityksien mukana osa Firefoxin tarjoamista työkaluista oli hidastunut huomattavasti.

Relaatiotietokantojen hallintaan työkaluksi valittiin MySQL Workbench, joka lukuisine ominaisuuksineen kattoi kaikki tarvittavat toiminnallisuudet. Kyseinen työkalu oli tekijälle jo entuudestaan tuttu ja hyväksi todettu, jolla hän oli toteuttanut myös jotain aikaisempia projektejaan. Lisäksi toimeksiantaja tarjosi tekijän käyttöön kyseistä työkalua, joten se valittiin osaksi opinnäytetyön toteutukseen tarvittavaa työkalusettiä.

4.4 Ohjelmistokehys

Ohjelmistokehykset ovat eräänlaisia runkoja, joita täydentämällä saadaan luotua toimivia sovellusosia sekä kokonaisuuksia. Kehysten ideana on lisätä tehokkuutta sovellusten luomiseen vähentäen kirjoitettavan ohjelmakoodin määrää sekä tarjoten laadukkaan, luotettavan sekä turvallisen alustan uudelle sovellukselle. Ohjelmistokehysten ansiosta kehittäjät voivat keskittyä sovelluksen rakenteen sijaan sen vaatimuksiin lisäten näin tuottavuutta. (Baker 2009.)

Koska opinnäytetyöaihe liitetään osaksi jo olemassa olevaa kokonaisuutta, uuden ohjelmistokehyksen valitseminen sen pohjaksi ei olisi järkevää. Opinnäytetyössä pohjana käytettiin siis Adminetin omaa ohjelmistokehystä, joka on rakennettu ja muovattu vuosien varrella käyttötarkoitukseen sopivaksi. Tästä huolimatta opinnäytetyötekijä on halunnut tarkastella myös vaihtoehtoisia ratkaisuja, jotka olisivat voineet toimia työn ohjelmistokehyksenä. Nämä vaihtoehtoiset kehykset käydään läpi seuraavaksi.

Laravel

Laravel on helposti lähestyttävä vapaanlähdekoodin PHP-ohjelmistokehys, joka seuraa MVC-arkkitehtuurin mukaista suunnittelumallia. Se hyödyntää olemassa olevien ohjelmistokehysten valmiita komponentteja ja tarjoaa suuren määrän eri ominaisuuksia sekä toiminnallisuuksia, jotka nopeuttavat selainsovellusten kehitystä. Laravelin ansiosta resurssienhallinta helpottuu, selainsovelluksen suunnitteluun käytettävä aika vähenee ja sovelluksista tulee skaalautuvampia. (Laravel – Overview n.d.)

Yii 2.0

Yii (Yes It Is!) on komponenttipohjainen PHP-ohjelmistokehys, joka mahdollistaa nopean sovelluskehityksen moderneille selainsovelluksille. Yii soveltuu erityisesti suurien sovellusten kehitykseen komponenttipohjaisen arkkitehtuurin sekä hienostuneen välimuistitukensa ansiosta. Yiin pyrkimyksenä on saavuttaa aina mahdollisimman suorituskykyisen lopputulos. Siitä on saatavilla nykyään kaksi versiota, joista versio 2.0 on täysin uudelleen kirjoitettu ja siihen on sisällytetty mm. viimeisimmät protokollat sekä teknologiat. (The Definitive Guide to Yii 2.0 n.d.)

4.5 Versionhallinta

Versionhallinnalla tarkoitetaan kykyä hallita sovellukseen tehtäviä muutoksia. Se on tärkeä osa sovelluskehitysprosessissa ja mahdollistaa usean ohjelmoijan työskentelyn saman järjestelemän parissa. Versionhallintaa varten on olemassa useita versionhallintaohjelmia. Nämä ohjelmat mahdollistavat sen, että kaikilla ohjelmoijilla on aina uusin version lähdekoodista, eivätkä he voi vahingossa ylikirjoittaa toistensa tekemiä muutoksia. Versionhallintajärjestelmä myös mahdollistaa lähdekoodin eri versioiden vertailun, muutosten liittämisen sekä vanhan version palautuksen. (Azarian 2013.)

Opinnäytetyössä tällaisena versionhallintaohjelmana käytettiin Microsoftin Visual SourceSafea (ks. liite 1). Se on tiedostotason versionhallintajärjestelmä, joka mahdollistaa työskentelyn eri projektiversioiden parissa järjestelmäriippumattomasti. Ohjelmakoodin lisäksi SourceSafeen on mahdollista taltioida myös muunlaisia tiedostoja, joten SourceSafe toimii hyvänä alustana tiedostojen hallintaan muidenkin kuin kehitystiimin sisällä. (Introducing Visual SourceSafe n.d.)

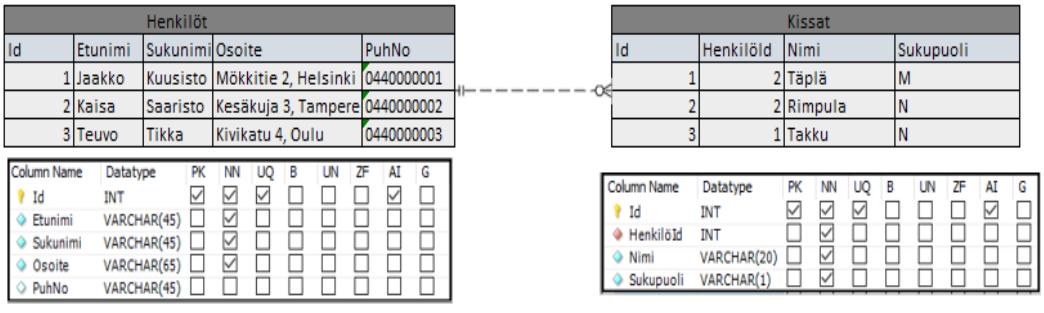
4.6 Tietokannat

4.6.1 Yleistä

Tietokannat ovat tietovarastoja, jotka sisältävät erilaisia kokoelmia tietoa. Perinteisesti nämä tiedot on jaoteltu tauluihin, ja niissä eri riveihin ja sarakkeisiin. Näistä tauluista käyttäjä on voinut lukea tai käydä kirjoittamassa dataa. Tietokannat ovat kehittyneet huimasti niiden alkuajoista, ja nykyään niistä onkin olemassa lukuisia eri versioita. (Rouse 2017a.) Jos tietokannat haluttaisiin jaotella kahteen ryhmään, ne voisivat olla relaatiotietokannat ja epärelaatiotietokannat. Tässä opinnäytetyössä tarvittiin molempia, koska tarkoituksena oli tallentaa sekä pysyvää että jatkuvasti muuttuvaa dataa.

Relaatiotietokanta

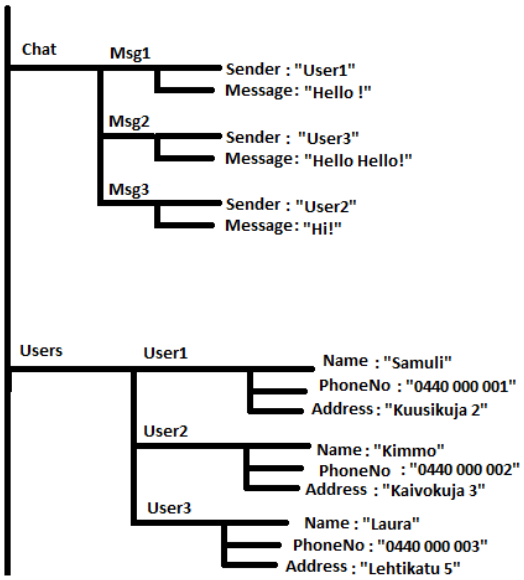
Relaatiotietokannoissa data esitetään riveinä sekä sarakkeina eri tauluissa. Näille tauluille luodaan oma mallinsa, joka määrää, millaista dataa rivin sarakkeisiin voidaan tallentaa (ks. kuvio 3). Taulun riveillä on jokaisella oma uniikki avaimensa, jonka ansiosta eri taulujen rivejä on mahdollista yhdistää toisiinsa. Yhdistäminen tapahtuu tallentamalla halutun taulurivin uniikkiavain toisen taulun riviin. Kyselyt relaatiotietokantoihin suoritetaan transaktioina käyttäen SQL-kieltä. Ne tukevat hyvin monimutkaisia kyselyitä, mistä johtuen helposta SELECT-kyselystäkin voi olla tuhansia mahdollisia toteutuspolkuja, joita evaluoidaan ajon aikana. (Serra 2015.)



Kuvio 3. Esimerkki relaatiotietokannan taulurakenteesta ja tyylimäärittelystä

Epärelaatiotietokanta

Epärelaatiotietokannat mahdollistavat tiedon tallentamisen monessa muodossa. Niiden rakenteen suunnittelu on helppoa, koska mitään relaatiotietokannoissa käytettäviä tarkkoja taulumalleja ei tarvitse luoda. Tämä mahdollistaa sen, että ne voivat vastaanottaa hyvinkin eri tavalla jäsenneltyä dataa ja eri dokumentit voivat sisältää eri määrän kenttiä. Tiedot tallennetaan epärelaatiotietokannoissa yleensä JSON-muodossa olevaan tiedostoon, jossa data kuvataan puukaaviossa (ks. kuvio 4). Tällaisesta tiedostosta tiedonhaku on yleensä nopeampaa, koska relaatiotietokannoissa tehtäviä taululiitoksia ei tarvita. Monet epärelaatiotietokannat eivät edes tue tällaisia liitoksia. (Serra 2015.)



Kuvio 4. Esimerkki epärelaatiotietokannan rakenteesta

4.6.2 Vaihtoehdot

MySQL

MySQL on suosituin vapaanlähdekoodin SQL-tietokantaohjelmisto, jonka lataaminen ja käyttäminen on täysin ilmaista. Sen tietokannat ovat relaatiotietokantoja, joihin kyselyt suoritetaan SQL-kielellä. MySQL:n tietokantapalvelimet ovat helppokäyttöisiä,

skaalautuvia sekä luotettavia ja tarjoavat runsaasti hyödyllisiä funktioita eri tilanteita varten. Ne kehitettiin käsittelemään suuria tietokantoja nopeammin kuin muut vastaavat tuotteet ja ovatkin onnistuneet tässä hyvin. Verkkoympäristöihin kyseiset palvelimet soveltuvat hyvin mm. yhdistettävyytensä, nopeutensa ja tietoturvallisuutensa vuoksi. (What is MySQL? n.d.)

Firebase

Googlen tarjoama maksullinen Firebase on pilvessä ylläpidetty epärelaatiotietokanta, joka mahdollistaa datassa tapahtuneiden muutosten synkronoinnin reaaliajassa eri käyttäjille. Firebasessa data tallennetaan yhteen isoon JSON-tiedostoon, joka muodostuu sisäkkäisistä objekteista ja listoista. Nämä objektit ja listat on kuvattu puukaaviossa, jossa haluttua dataa voidaan tarkastella navigoimalla hierarkiassa kyseisen lapsisolmun kohdalle. (Wingerath 2017.)

Koska Firebase palauttaa kaikki valitun solmun lapsisolmut sekä tarkastelee muutoksia näissä solmuissa, JSON-dokumentin rakenne kannattaa pitää mahdollisimman litteänä. Tämä hajauttaa datan samalla tasolla oleviin eri solmuihin, joista yhtäaikainen tiedon haku on kuitenkin hankalaa Firebasen yksinkertaisista kyselyistä johtuen. Tästä riippumatta Firebase soveltuu hyvin erilaisiin käyttötarkoituksiin mm. skaalautuvuutensa, suurten samanaikaisten käyttäjämäärien sekä suoritustehonsa ansiosta. (Wingerath 2017.)

RethinkDB

Samoin kuin useat muut epärelaatiotietokannat, myös RethinkDB on JSON-dokumenttipohjainen tietovarasto. Se on vapaaseen lähdekoodiin perustuva toteutus, jota kehitettiin yli viiden vuoden ajan tietokantaosaajista koostuvan tiimin sekä muiden avustajien kanssa. RethinkDB luotiin täysin tyhjälle pohjalle käyttäen C++-ohjelmointikieltä. Samalla sille myös kehitettiin oma kyselykielensä nimeltään ReQL, joka tarjoaa lähes kaikki toiminnallisuudet mitä perinteinen SQL-kielikin. Tämä mahdollistaa erittäin monimutkaisten kyselyjen suorittamisen RethinkDB:ssä, mikä useissa muissa epärelaatiotietokannoissa on mahdotonta. (What is RethinkDB? n.d.)

Datan reaaliaikainen näyttäminen RethinkDB:ssä onnistuu puskemiskomennolla, joka työntää päivityskyselystä saadun vastauksen sovelluksille päivityksen tapahtuessa.

Jos tarvetta jatkuvaan reaaliaikaiseen seurantaan ei ole, RethinkDB:llä on myös mahdollista suorittaa datan haku perinteisellä ajatusmallilla, jossa haluttu kysely lähetetään palvelimelle, johon tämä vastaa. (What is RethinkDB? n.d.)

4.6.3 Valinnat

Pysyvän tiedon tallennukseen

Pysyvällä tiedolla tarkoitetaan tässä opinnäytetyössä tietoa, joka muuttuu erittäin harvoin. Tällaisia ovat esimerkiksi kohteiden tiedot, edullisempien laitteiden tiedot ja muu vastaavanlainen data, jonka ei tarvitse päivittyä käyttäjälle reaaliajassa. Tietokannan valitseminen tällaista dataa varten on erittäin helppoa, koska mitään erikoisempia vaatimuksia datan tallentamiseen ei ollut.

Pysyvän tiedon tallennukseen tietokannan tyypiksi valittiin relaatiotietokannat, koska ne mahdollistavat tarkat tyyppimäärittelyt tiedolle ja ovat helposti hallittavissa. Lisäksi ne tukevat jo valmiita ratkaisuja paremmin kuin epärelaatiotietokannat. Tietokantaohjelmistoksi valittiin MySQL mm. sen helppokäyttöisyyden sekä skaalautuvuutensa vuoksi. Lisäksi se tarjosi runsaasti kehittyneitä toiminnallisuuksia sekä mahdollisti hankalampien kyselyiden luonnin tietokantaan.

Muuttuvan tiedon tallennukseen

Muuttuvalla tiedolla tarkoitetaan tässä opinnäytetyössä dataa, joka päivittyy usein ja jota halutaan näyttää käyttäjälle reaaliajassa. Tällaista dataa on esimerkiksi kalliimpien laitteiden sijaintitietodata.

Tietokannan valinta muuttuvan tiedon tallentamiseen oli hankalampaa kuin muuttumattoman, koska tekijä ei ollut täysin varma, millainen tietokanta tarkoitukseen sopisi parhaiten. Eri tietokantaratkaisuja tutkittaessa tekijälle kuitenkin selvisi, että relaatiotietokannat eivät tällaisen datan tallentamiseen sovellu, vaikka se niillä myös onnistuisi. Kyseisiä tietokantoja ei ollut suunniteltu jatkuvasti muuttuvan datan tallentamiseen sekä tarkasteluun, mikä kävi ilmi mm. kyselyiden määrästä, joka olisi tarvittu seurannan suorittamiseen. Jos kuitenkin esimerkiksi MySQL-relaatiotietokantaohjelmistoa olisi haluttu käyttää lopputuloksen saavuttamisen, olisi apuna voitu käyttää sen tarjoamia laukaisimia (laukaisevat määrätyn toiminnallisuuden tietyn tapahtuman toteutuessa) sekä muutosilmoituksia (lähettävät ilmoituksen, kun viimeksi

pyydetty data muuttuu). Niistäkin huolimatta turhien kyselyjen määrä olisi kuitenkin lisääntynyt ja monen laitteen yhtäaikainen seuranta olisi käynyt erittäin raskaaksi.

Edellä mainituista syistä tietokantatyypiksi valikoitui epärelaatiotietokanta. Kyseisiä tietokantoja oli markkinoilla useita, joista valittiin muutama tarkempaan tutkimukseen mm. eri arvostelujen perusteella. Tarkemman tutkimuksen jälkeen voittajaksi valikoitui Googlen tarjoama Firebase-tietokanta. Syitä Firebasen valintaan olivat esimerkiksi järkevä laskutusmalli, skaalautuvuus, vakaus, Googlen hoitama ylläpito ja se, ettei siitä löytynyt suurempia pullonkauloja. Hyvien puolien lisäksi myös muutamia huonoja puolia löytyi. Näitä olivat hinnannousu käyttäjämäärien noustessa sekä suppea kyselykieli tietokantaan. Huonoista puolista riippumatta tekijä totesi Firebasen parhaiten sopivaksi tarvittavaan käyttötarkoitukseen.

4.7 Karttarajapinnat

4.7.1 Yleistä

Jotta kehittäjien ei tarvitsisi aina kirjoittaa uudelleen koodia interaktiivisten karttojen sekä niiden kerrosten näyttämiseen, monet karttoihin erikoistuneet yritykset ovat alkaneet tarjoamaan omia karttarajapintojaan. Rajapinnat sisältävät yleensä luokkia mm. karttojen sekä niiden kerrosten näyttämiseen, joiden ansiosta kehittäjät voivat keskittyä itse kehitettävään sovellukseen kartan perustoiminnallisuuksien sijaan. (Quinn n.d.) Yksinkertaisuudessaan karttarajapintatarjoajana voidaan pitää tahoa, joka tarjoaa rajapinnan, josta ainakin kartan muodostavat karttalaatat eli neliönmuotoiset palaset, voidaan noutaa. Nykypäivänä kuitenkin monet yritykset tarjoavat jo hyvinkin laaja-alaisesti eri ominaisuuksia karttarajapintoihinsa. Tarkasteltaessa esimerkiksi Googlen tarjoamia ominaisuuksia huomataan, etteivät ne rajoitu pelkästään karttojen näyttämiseen ja niiden liikutteluun, vaan seasta löytyy mm. erilaisia reitti-haku-, paikkahaku- sekä geokoodausominaisuuksia. Ominaisuuksien lisäksi monista karttarajapinnoista on myös haettavissa karttoihin liittyvää karttadataa, kuten esimerkiksi päivittyviä liikennetietoja tai lisätietoja kartalla näytettävistä kohteista. Karttadata käsitteen alle voidaan myös lukea mukaan itse kartat, koska ne sisältävät esimerkiksi tiestöön sekä maanpinnanmuotoihin liittyvää dataa.

Koska vuosien varrella monet karttarajapintatarjoajat ovat laajentaneet rajapintansa kattamaan muun muassa edellä mainitut toiminnot sekä tiedot, itse karttarajapinta on usein jaoteltu selkeyden vuoksi eri osa-alueisiin. Näitä voivat olla esimerkiksi paikkatietoa käsittelevä rajapinta tai geokoodauksen hoitava rajapinta. Jotta kehittäjien olisi helpompi käyttää näitä eri rajapintoja keskenään, on niiden käsittelyyn luotu erilaisia kirjastoja, jotka hoitavat kyselyt eri rajapintoihin. Yleensä kehittäjät voivat kutsua näitä eri rajapintoja saman kirjaston kautta. Nämä kirjastot tarjoavat usein myös erilaisia toimintoja esimerkiksi kartalla näytettävien objektien luomiseen.

4.7.2 Vaihtoehdot ja valintaprosessi

Ensimmäinen kysymys karttarajapintaa valittaessa yleensä on, mitä kaikkea pitäisi ottaa huomioon rajapintaa valittaessa. Pohdittavia asioita on paljon, eikä niitä kannata sivuuttaa, sillä väärän karttarajapinnan valinta saattaa vaikuttaa isoissa yrityksissä moniin asioihin vielä vuosienkin päästä. Liikkeelle kannattaa lähteä pohtimalla mitkä ovat ympäristöt, joissa kartta halutaan näyttää. Kun ympäristö on selvillä, se määrittelee itsestään tarvittavat kielet, joita karttarajapinnan tulisi tukea. (Quinn n.d.) Kun ympäristö ja tarvittavat kielet on määritelty, siirrytään pohtimaan, mitä karttasovellukselta haluttaan nyt ja mitä tulevaisuudessa. Tämä on tärkeää vaihe, sillä jos määritelmä on huono, voidaan tulevaisuudessa törmätä tilanteeseen, jossa aikaisemmin valittu rajapinta ei välttämättä taivu myöhemmän vaiheen vaatimuksiin. Vaatimustenmäärittelyn jälkeen kannattaa pohtia, halutaanko karttapalvelin ylläpitää itse vai ostaa rajapinta joltain muulta. Jos karttapalvelimia ylläpidetään itse, se tuo toki vapautta ja vähentää kuluja, mutta kuinka paljon se lisää ohjelmointi- sekä ylläpitotyötä? Tärkeää on myös huomioida, kenelle karttasovellusta tehdään. Jos kyseessä on iso yritys, tulisi karttarajapinnan tarjoajan olla riittävän suuri sekä uskottava, koska pienen yrityksen valinta rajapinnan tarjoajaksi saattaisi vähentää ostajan uskottavuutta sekä heikentää itse sovelluksen toimivuutta. Huomioitavia asioita on vielä paljon lisää ja ne vaihtelevat sovelluskohtaisesti, joten niitä kaikkia ei tässä käydä läpi. Edellä mainitut antavat kuitenkin hyvän pohjan eri karttarajapintojen vertailuun ja niitä käytettiin myös tämän opinnäytetyön valintojen pohjana – muiden rajausten lisäksi.

Opinnäytetyössä toimeksiantaja antoi tekijälle heti alussa muutaman karttarajapinta-tarjoajan tutkittavakseen. Näiden lisäksi tekijän tuli myös itse etsiä muutama potentiaalinen vaihtoehto. Kun tekijä valikoi suuresta määrästä eri karttajarajapintojen tarjoajia omasta mielestään sopivimmat, alkoi hän vertailla näitä kaikkia vaihtoehtoja keskenään. Hyvänä työkaluna tässä toimi SWOT-analyysi, jolla saatiin karkea kuvaus karttarajapintojen vahvuuksista, heikkouksista, mahdollisuuksista ja uhista. Asioita, joihin kiinnitettiin huomiota, olivat mm. tarjolla oleva dokumentointi, lisenssityyppi, kustannukset, vakaus, laajennettavuus, rajoitukset ja erikoisuudet. Opinnäytetyössä tarkempaan tarkasteluun valikoituivat seuraavaksi esitetyt karttarajapinnat sekä ratkaisut.

OpenStreetMap

OpenStreetMap on vapaehtoisvoimin rakennettu ilmainen ja muokattavissa oleva kartta, joka koostuu karttalaatoista ja on noudettavissa OpenStreetMapin tarjoaman rajapinnan kautta. Koska karttaa rakennetaan vapaehtoisvoimin se ei ole vielä täysin valmis kokonaisuus, mutta jokainen halukas voi osallistua sen kehitykseen. Kartan data on koostettu mm. vapaehtoisten GPS-laitteilla keräämästä reittidatasta, Bingiltä saamien ilmakuvien perusteella sekä muilla kartoitustekniikoilla. (About OpenStreetMap 2017.)

Koska OpenStreetMapin karttalaattapalvelin toimii lahjoituksilla sekä sponsoreiden tuella, sen kova rasittaminen on kiellettyä. OpenStreetMapin datan käyttö on kuitenkin ilmaista, minkä johdosta monet yritykset tarjoavat omia rajapintojaan näiden karttapalasten tiheämpään lataamiseen. (Tile Usage Policy n.d.) Myös oman karttalaattapalvelimen pystyttäminen on sallittua ja sitä jopa suositellaan, koska se mm. vähentää kuormaa heidän omilta palvelimiltaan sekä edistää itse OpenStreetMap-projektia (Creating your own tiles 2018). Ennen oman karttalaattapalvelimen pystyttämistä on kuitenkin hyvä tutustua lisenssiehtoihin, joissa kerrotaan esimerkiksi, että OpenStreetMap on lisensoitu ODbl-lisenssillä. Se tarkoittaa yksinkertaisuudessaan sitä, että OpenStreetMapin karttojen sekä tietojen levittäminen, välittäminen, kopiointi ja mukauttaminen on sallittua, kunhan mm. vaadittua attribuutiota noudatetaan sekä tehdyt karttamuutokset jaetaan. (ODC Open Database License (ODbL) Summary n.d.)

Koska karttalaattapalvelin palauttaa pelkästään neliönmuotoisia kuvia eri karttapaloista, tarvitaan tähän rinnalle kirjastoja karttapalojen yhdistelemiseen ja eri toiminnallisuuksien luomiseen. Näistä kirjastoista löytyy kattava lista OpenStreetMapin omilta sivuilta, joista jokainen voi etsiä omaa käyttötarkoitustaan varten parhaat vaihtoehdot. Dokumentaatioita näihin kirjastoihin löytyy vaihtelevasti, mutta ainakin käytetyimmistä kirjastoista tietoa sekä esimerkkejä löytyy hyvin.

Hinta karttalaattapalvelimen alustamiselle olisi noin 2000 euroa. Hinta perustuu GEOFABRIK nimisen firman arvioon, jossa määriteltiin karttapalapalvelimen alustamiseen kuluva hinta. (OpenStreetMap Server Setup n.d.) Alustamisen lisäksi arvioituun määrään tulisi lisätä myös itse palvelimen hinta sekä tämän ylläpitoon kuluva summa.

Mapbox

Mapbox on vuonna 2010 perustettu sijaintitietoa näyttävä alusta mobiili- sekä verkkosovelluksille. Tunnetuimpia alustan käyttäjiä ovat esimerkiksi Snapchat sekä Lonely Planet -palvelut. Mapboxilla on vapaaseen lähdekoodiin perustuvat juuret, minkä vuoksi he pyrkivät vielä nykyäänkin julkaisemaan niin paljon lähdekoodiaan kuin mahdollista. (About Mapbox n.d.) Kirjoitushetkellä heiltä löytyy jo yli 700 eri projektia säiliöitynä GitHub-palvelusta.

Karttadatansa Mapbox hankkii lukuisista eri lähteistä, kuten esimerkiksi OpenStreetMapilta, valtiollisilta sekä kaupallisilta toimijoilta ja kulkuneuvojen anonyymistä sensoridatasta. Tämän ansiosta he tarjoavat laaja-alaisesti erilaisia karttalaattoja, jotka visualisoivat mm. katuja, rakennuksia, pinnanmuotoja sekä liikennetietoja. (Our map data n.d.) Koska Mapbox käyttää hyväksi OpenStreetMapin karttadataa, he ovat velvollisia jakamaan karttamuutoksena julkisesti. Tämä ei kuitenkaan ole heille ongelma, koska yrityksen ajatusmaailma perustuu jo muutenkin vapaaseen ajattelumalliin.

Kehittäjille Mapbox tarjoaa hyvin dokumentaatiota ja esimerkkejä eri SDK:ita sekä kirjastoja varten. Ne kattavat mm. karttojen liitämisen sovellukseen, navigoinnin sekä geokoodauksen. Lisäksi Stack Overflowssa löytyy kohtuullinen määrä Mapboxiin liittyviä kysymyksiä, joista kehittäjille saattaa löytyä vastauksia kohtaamiinsa Mapboxiin liittyviin ongelmiin.

Hinnoittelu Mapboxissa toimii ”maksa mitä käytät” -periaatteella, ellei kyseessä ole kaupallinen tuote, jolloin hinta koostuu 499 dollarista kuukaudessa, mihin lisätään maksa mitä käytät -hinnoittelun osuus. Kaupalliseen lisenssiin sisältyy 250 istuinta (yksittäistä käyttäjää), 1000 seurattavaa kohdetta sekä 50 kappaletta navigaatiosovelluksia. Jotta kaupallinen lisenssi voidaan hankkia yritykselle, täytyy yrityksen olla yhteydessä suoraan Mapboxin myyjään – niin kuin opinnäytetyöntekijänkin teki. (Plans & Pricing n.d.)

Google Maps

Google Maps on Googlen ylläpitämä verkkopohjainen palvelu, joka tarjoa hyvin monenlaista maantieteellistä dataa eripuolilta maailmaa. Peruskarttakuvien lisäksi sillä voi tarkastella alueita esimerkiksi satelliittinäkymän- tai Googlen kartoitusautojen ottamien kuvien kautta. Alueiden tarkastelun lisäksi Google Maps tarjoaa useita muita ominaisuuksia, kuten reittisuunnittelun ja mobiililaitteiden paikannuksen. (Rouse 2013.)

Googlen karttapalvelu tarjoaa kehittäjille useita keinoja esimerkiksi kartan liittämiseen, tiedonhakuun ja tiedon näyttämiseen mm. eri kirjastojen sekä rajapintojen avulla. Ne on lajiteltu kolmeen kategoriaan, joista muodostuvat Googlen tarjoamat perustuotteet: kartat, reitit ja paikat. Nämä kolme perustuotetta on hinnoiteltu eri ominaisuuksiin, joiden hinta määräytyvät käytön mukaan, jos ilmaisen 200 dollarin raja ylitettään kuukauden aikana. (Google Map Platform FAQ 2018.)

Koska Googlen hinnoittelu perustuu täysin käyttömäärään, tämä mahdollistaa ajoittain erittäin tarkkojenkin hinta-arvioiden tekemisen. Näiden hintojen arviointia lähestyttiin opinnäytetyössä kahdesta eri näkökulmasta Excel-taulukkolaskentaohjelmaa apuna käyttäen. Ensimmäisessä näkökulmassa käytettävät transaktiot porrastettiin ominaisuuksille eri käyttötapauksien perusteella, jonka jälkeen hinta laskettiin transaktioista (ks. kuvio 5). Toisessa näkökulmassa tarkasteltiin, mitä kilpailevan palvelun tarjoamalla hinnalla saataisiin Googlen palvelusta transaktioina (ks. kuvio 6).

Kehittäjille Google Maps tarjoaa runsaasti laadukkaita esimerkkejä ja oppaita eri kirjastojen sekä rajapintojen käyttöä varten. Niissä käydään läpi kaikki aina tarvittavien sovellusten lataamisesta eri kartan ominaisuuksien lisäilyyn. Lisäksi Google Mapsin

suuren suosion ansiosta internetistä löytyy mm. useita ulkopuolisten tekemiä oppaita, videoita, blogeja ja kysymyksiä sen eri ominaisuuksiin liittyen.

Harmaa kenttä = Muokattavissa

Ominaisuudet	vuodessa transaktioita	Hinta ennen alea	Transaktiot miinus ilmaiset	Hinta alen kanssa
Dynamic maps	**600 000**	3 642,00 €	404000	2 452,28 €
Static Maps	80 000	138,72 €	80 000	138,72 €
Directions	100 000	433,50 €	100 000	433,50 €
Directions Advanced	50 000	433,55 €	50 000	433,55 €
Geocoding	300 000	1 300,71 €	300 000	1 300,71 €
Find place	50 000	737,05 €	50 000	737,05 €
autocomplete	600 000	1 472,40 €	110 000	269,94 €
yht.	1 780 000	8 157,93 €	1094000	5 765,75 €

Hinta kuukaudessa + ale	Hinta päivässä + ale (30 pv)
480,48 €	16,02 €

Kuvio 5. Laskentataulukko hintojen arviointiin transaktioiden perusteella

Harmaa kenttä = muokattavissa

	hinta välille 0-100 000 kutsua hinta/1000 kutsua euroina	lkm/kk	lkm/v	(vuodessa [365 pvä]) kutsua/päivä	(työpäivien mukaan [261 pvä]) kutsua/työpäivä
Dynamic maps	11,97 €	20000	240000	658	920
Static map	1,71 €	1000	12000	33	46
Directions	4,28 €	4000	48000	132	184
Roads - Route Traveled	8,56 €	15000	180000	493	690
Geocoding	4,28 €	3000	36000	99	138

Hinta/kk	399,47 €
Hinta/v	4 793,64 €
vertaushinta/kk	400,00 €
vertaushinta/v	4 800,00 €
Jäljellä/kk	0,53 €
Jäljellä/v	6,36 €

Kuvio 6. Laskentataulukko vertailuun muiden hintojen kanssa

HERE

Heren juuret ulottuvat jo vuoteen 1985, jolloin heidän tavoitteenaan oli digitalisoida kartat sekä saavuttaa asema autonavigaatiojärjestelmien edelläkävijänä. 30 vuoden aikana tätä tavoitetta rakennettiin sekä NAVTEQin että Nokian lipun alla luoden näin perintöä tuleville kartoitusteknologioille. Nykyään tämä tavoite voidaan katsoa saavutetuksi, sillä Here työskentelee yhteistyössä maailman johtavien autovalmistajien kanssa, minkä johdosta Euroopassa sekä Pohjois-Amerikassa neljä autoa viidestä käyttää navigointijärjestelmänään Heren ratkaisuja. Here ei kuitenkaan keskity pelkästään autojen navigointijärjestelmien luomiseen, vaan heidän tavoitteenaan on kehittää tulevaisuuden paikannusteknologioita yhdessä mm. NVIDIAn sekä Mobileyen kanssa. (About HERE n.d.)

Here tarjoaa laaja-alaisesti eri ominaisuuksia, jotka tarjoavat toimintoja sekä pienille että suurille yrityksille. Juuret kuljetusalan järjestelmien parissa on huomattavissa selvästi, sillä erilaisia kalustonhallintaan sekä teihin liittyviä ominaisuuksia löytyy runsaasti. Näihin Here tarjoaa kohtuullisen määrän dokumentteja sekä erilaisia esimerkkejä. Kysymykset Heren karttarajapintaan pyydetään esittämään Stack Overflowssa, jossa Heren karttatiimi pyrkii vastaamaan niihin mahdollisimman usein. Esitettyjen kysymysten määrä on kuitenkin valitettavan vähäinen, joten moniin kysymyksiin kehittäjän täytyy selvittää itse vastaukset.

Here tarjoa kahta peruslisenssiä, joista ensimmäinen on ilmainen ja se kattaa 250 000 transaktiota (pyyntöä palvelimelle), 5000 kuukausittaista SDK:n käyttäjää sekä 250 seurattavaa kohdetta. Jos transaktiomäärät ylittyvät, ylittyvästä osuudesta maksetaan tällä lisenssillä yksi euro 1000 transaktiota kohden. Toinen tarjottu lisenssi on hinnaltaan 449 euroa kuukaudessa. Se kattaa kaiken edellä mainitun, mutta suuremmalla transaktioiden määrällä, joka on tällä lisenssillä miljoona. Jos arvioidaan että käyttömäärät ylittyvät, on parasta ottaa yhteyttä suoraan rajapinnantarjoajaan. (Plans And Pricing n.d.) Koska edellä esitetyissä lisensseissä seurattavien kohteiden määrä oli maksimissaan 250, tekijä halusi selvittää tarkemmat hinnat myös ylimeneville osuuksille. Tästä syystä myös hän oli yhteydessä rajapinnantarjoajaan.

Yhteenveto karttarajapinnoista

Tarkempaan tarkasteluun valikoitujen karttarajapintojen syvemmän tutkimisen sekä vertailun päätteeksi päätettiin luoda esimerkki karttasovellukset kahdesta potentiaalisimmasta rajapinnasta, joita olivat Googlen sekä Heren tarjoamat ratkaisut. Näistä kahdesta Heren rajapintaa päätettiin testata ensin, joten se myös valikoitui opinnäytetyössä käytettäväksi karttarajapinnaksi.

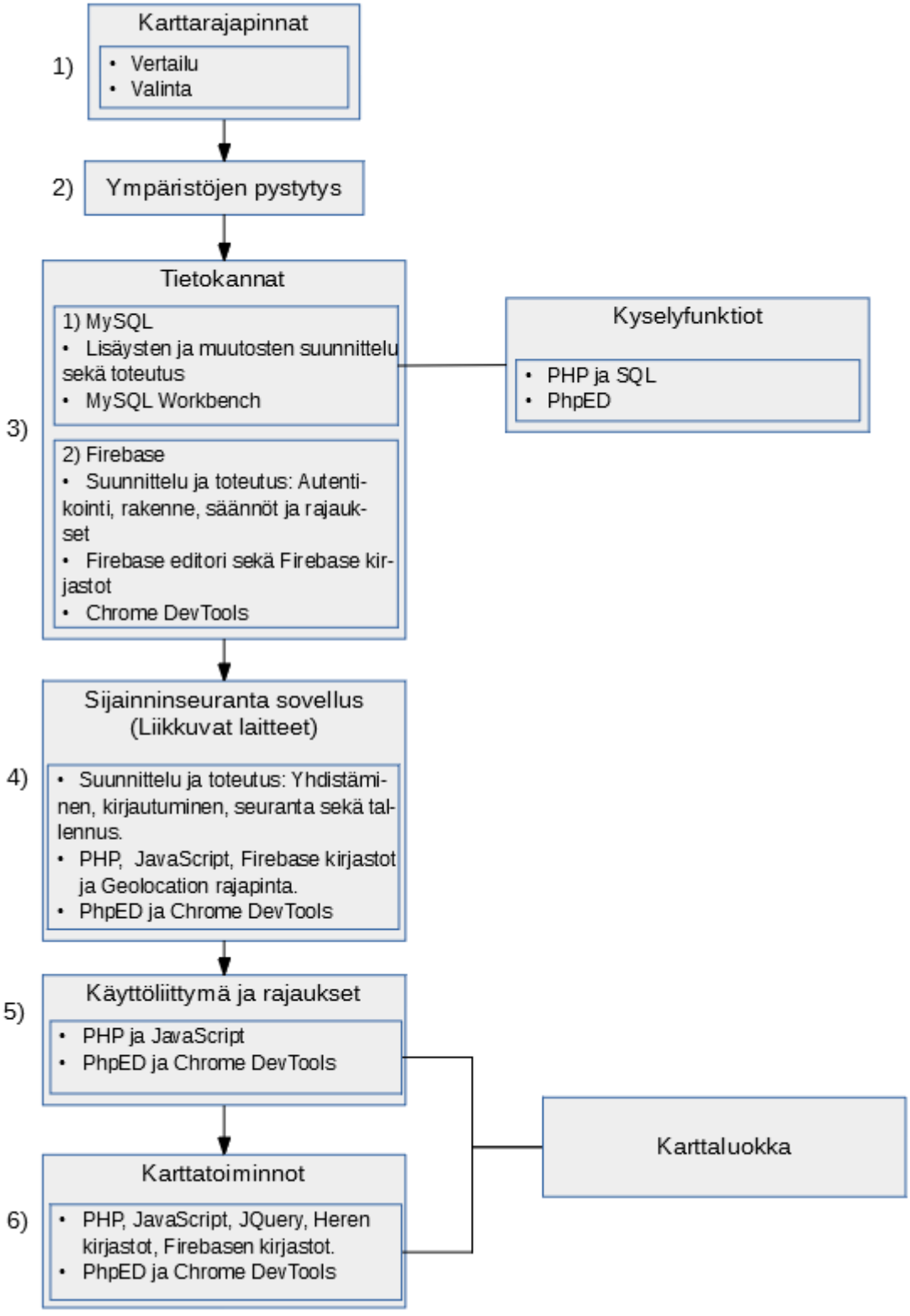
OpenStreetMap, MapBox sekä Bingin tarjoama Bing Maps, jota tässä opinnäytetyössä ei käyty tarkemmin läpi, karsiutuivat pois erilaisista syistä. OpenStreetMap olisi esimerkiksi vaatinut oman karttalaattapalvelimen pystyttämisen ja ylläpitämisen, joka olisi lisännyt valtavasti työnmäärää sekä työstövaiheessa että tulevaisuudessa. MapBox taas olisi ollut hyvinkin potentiaalinen vaihtoehto mm. avoimuutensa, kasvunsa sekä ominaisuuksiensa puolesta, mutta viestintä heidän yhteyshenkilöidensä kanssa oli erittäin hankalaa, sillä he vastailivat vaihtelevasti opinnäytetyön

tekijän yhteydenottoyrityksiin. Koska yhteydenpito oli jo heti aluksi ongelmallista, se olisi luultavasti aiheuttaa myöhemmässä vaiheessa suurempia ongelmia, jonka vuoksi opinnäytetyöntekijä ei pitänyt MapBoxia tarpeeksi luotettavana karttarajapintatarjoajana. Bing Maps taas hävisi hintavertailussa muihin rajapintoihin nähden, joten tekijä ei nähnyt syytä tarkastella kyseistä rajapintaa enempää.

Koska opinnäytetyön toteutusvaihetta aloitettaessa Googlen rajapinnan lisensointimalli oli muuttumassa, eikä tarkkoja ehtoja sekä hintoja ollut silloin vielä saatavissa edes Googlen yhteistyökumppanina toimivalta konsulttitalolta, tekijä päätyi testaamaan ensimmäisenä Heren ratkaisua. Tämän lisäksi kyseinen rajapinta tarjosi lukuisia mielenkiintoisia ominaisuuksia, joista tekijä halusi selvittää, voisiko niistä olla hyötyä nyt tai tulevaisuudessa. Here tarjosi myös selkeän hinnoittelun, jonka ansioista hinta-arvion tekeminen perustuen arvioituun käyttöön oli helppoa. Tällaisen hinta-arvion tekeminen ei Googlen tapauksessa edellä määriteltyjen syiden vuoksi vielä toteutusvaihetta aloittaessa onnistunut, joten jo opinnäytetyön selkeydenkin vuoksi tekijä päätyi testaamaan ensimmäisenä Heren rajapintaa.

4.8 Toteutusvaiheet

Ennen toteutuksen aloittamista tekijä pilkkoi opinnäytetyönä syntyvän sovelluksen eri toteutusvaiheisiin selkeyttääkseen itsellensä mm. ajallisia tavoitteita ja suunnitellakseen missä vaiheessa mikäkin palanen kannattaisi toteuttaa. Apunaan tekijä käytti mm. työn rajauksessa syntynyttä kuviota 1, vaatimusmäärittelyä sekä tähän mennessä tekemiään teknologiavalintoja. Tämän tuloksena syntyi kuvio 7, josta on nähtävissä esimerkiksi toteutusvaiheet, osa niiden tehtävistä sekä niissä käytettävät teknologia- ja työkaluvalinnat.



Kuvio 7. Alustava suunnitelma työn toteutusvaiheista

5 Toteutus

5.1 Yleistä

Työn toteutus on erittäin moniosainen vaihe, johon voidaan ajatella kuuluvaksi mm. suunnittelua, ympäristöjen pystytystä sekä ohjelmointia. Toteutusta lähdetään luomaan yleensä tavoitteiden, työnkuvauksen, vaatimusmäärittelyn, teknologiavalintojen sekä suunnittelujen pohjalta. Nämä yleensä tarkentuvat toteutusvaiheen elinkaaren aikana.

Tätä opinnäytetyötä lähdettiin työstämään kuvion 7 mukaisesti. Koska kuitenkin jo luvussa 4.7 käsiteltiin karttapalveluiden vertailu ja valinta ja ympäristöjen pystytys oli valittujen teknologiavalintojen ansiosta vaivatonta, siirrytään tässä opinnäytetyöraportissa suoraan kuviossa esitettyyn kohtaa kolme. Syyt ympäristöjen vaivattomaan pystytykseen johtuivat mm. siitä, että toimeksiantajalta löytyivät valmiit PHP-palvelimet sekä MySQL-tietokannat, joten niitä ei tarvinnut erikseen pystyttää. Tämän lisäksi esimerkiksi Firebase toimii suoraan Googlen tarjoamasta pilvestä, eli erillisiä alustuksia ei sitä varten tarvinnut tehdä.

5.2 MySQL-muutokset sekä kyselyfunktiot karttatoiminnoille

5.2.1 Yleistä

MySQL-tietokantataulujen suunnittelu, lisäys sekä muokkaus toteutettiin aikaisessa vaiheessa, koska opinnäytetyön tekijä halusi aloittaa sovelluksen toteutuksen jostain itselleen tutusta ympäristöstä. Lisäksi kyseisenlaiset yhden henkilön ohjelmointityöt olisi hyvä aloittaa lähes aina sovelluksen pohjalta, jotta lähemmäs käyttäjärajapintaa siirryttäessä tarvittavat komponentit olisivat jo saatavissa.

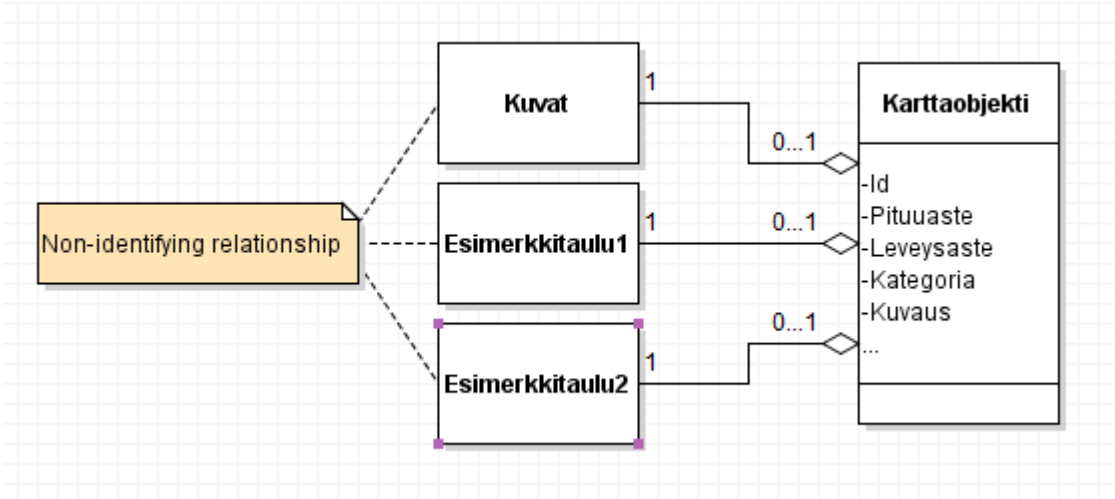
MySQL-muutosten lisäksi tekijä päätti toteuttaa karttatoimintoihin (kuvien, kohteiden, laitteiden ja koneiden näyttäminen kartalla) tarvittavat tietojenhakufunktiot jo tässä vaiheessa, jottei kyseisiin MySQL-tauluihin tarvitsisi perehtyä enää myöhemmin uudestaan. Tämä oli mahdollista jo nyt, sillä tekijä oli suunnitellut, että tarvittavat

tiedot haettaisiin käyttäjän halutessa Ajax-kutsulla, minkä seurauksena tietojenhaku-funktiot voitiin toteuttaa omana irrallisena kokonaisuutenaan itse ladattavasta sivusta.

5.2.2 MySQL-muutokset

MySQL-taulumuutosten suunnittelussa käytettiin apuna UML-editoria, jonka avulla tekijä visualisoi itselleen tarkemmin taulujen sen hetkiset sidokset sekä rakenteet. Editorin avulla hän myös suunnitteli uudet taulut, taulujen väliset sidokset sekä muutokset olemassa oleviin tauluihin. Osa tauluista oli jo olemassa, joten muutokset näihin onnistuivat helposti. Sen sijaan esimerkiksi sijainti- sekä karttaobjektitauluja ei ollut tarvitussa muodossa vielä olemassa, joten tekijän oli toteutettava ne itse.

Opinnäytetyöntekijä pyrki toteuttamaan tietokantamuutokset mahdollisimman uudelleen käytettäviksi sekä kompakteiksi. Esimerkiksi karttaobjektidataa sisältävän taulun tuli taipua kuvien lisäksi myös tulevaisuudessa lisättäviin eri tyyppisiin karttaobjekteihin. Tämän vuoksi karttaobjektitaulu tuli sisältämään tietoa mm. objektin tyypistä, sijainnista sekä kuvauksesta ja itse objektintaulu, kuten opinnäytetyön tapauksessa kuvat-taulu, tuli olla liitettävissä siihen (ks. kuvio 8).



Kuvio 8. Pelkistetty esimerkkikuva karttaobjektitaulun ideasta

5.2.3 Kyselyfunktiot

Kyselyfunktiot kantaan toteutettiin palvelimen kautta PHP-kielellä. Toteutus suunniteltiin toimivaksi niin, että käyttäjä pystyi hakemaan yhdellä Ajax-kyselyllä haluamansa kuva-, laite-, kone- sekä kohdetiedot. Ajaxilla kutsuttu PHP funktio tutki mitä kaikkea käyttäjä halusi näyttää kartalla ja toteutti tietokantakyselyt perustuen näihin ehtoihin. Tietokantavastauksien jälkeen PHP palautti vastaukset JavaScriptin objektimuodossa, jolloin JavaScript-ohjelman oli helppo tarkastella vastauksena tulleita arvoja.

5.3 Firebase-tietokannan suunnittelu ja toteutus

5.3.1 Yleistä

Firebase-tietokanta sekä sen malli olivat tekijälle entuudestaan tuntemattomia, minkä vuoksi ennen toteutuksen aloittamista tekijä joutui opettelemaan kyseisen tietokannan toimintaperiaatteet, siihen liittyvät säännöt sekä ajattelumallin. Netistä löytyneiden ohjesivujen, blogien sekä videoiden ansiosta tekijä sisäisti tarvittavat tiedot hyvinkin nopeasti. Silti relaatiotietokantamainen ajattelutapa oli juurtunut tekijän mieleen niin vahvasti, että tietokannan suunnittelu vaati huomattavasti aikaa. Suunnittelusta huolimatta opinnäytetyöntekijä löysi toteutuksen jälkeen entistä suorituskykyisemmän rakenteen kyseiselle tietokannalle, mitä voitaisiin käyttää apuna projektin myöhemmässä vaiheessa.

5.3.2 Suunnittelu

Firebase-tietokannan suunnittelu sisälsi kolme vaihetta: autentikoinnin-, rakenteen- sekä sääntöjen laatimisen. Tietokannan suunnittelu aloitettiin autentikoinnista, koska kyseisen tietokannan käyttörajapinta on kaikkien saatavilla ja tekijä halusi ensimmäiseksi varmistaa, ettei kukaan ulkopuolinen pääsisi tietokantaan käsiksi. Firebasen konfigurointitiedot ovat sovelluksessa nimittäin kaikkien nähtävissä ja ne ovat ikään kuin sijaintidataa, joka kertoo missä kyseinen tietokanta sijaitsee. Tämän vuoksi kuka tahansa voin tarkastella sekä muokkailla tietokannan tietoja, jos autentikoinnista sekä sallittujen osoitteiden rajaamisesta ei ole huolehdittu oikein. Opinnäytetyössä Firebasen lukuisista autentikointitavoista käyttöön valittiin sähköposti- sekä salasana

pohjainen kirjautuminen, koska se soveltui parhaiten opinnäytetyönä syntyvän esimerkkisovelluksen tarkoitukseen. Kirjautumistavan ansiosta jokaiselle käyttäjälle voitiin antaa oma käyttäjätunnuksensa, jolla hän pääsi sääntöjen sekä rajausten puitteissa lukemaan sekä kirjoittamaan tietojaan.

Autentikoinnintavan laatimisen jälkeen opinnäytetyöntekijä siirtyi suunnittelemaan JSON-tiedoston tietokantarakennetta. Tekijä hahmotteli paperille erilaisia puukaavioita, joista hän pyrki löytämään suorituskykyisimmän vaihtoehdon. Tällaisia suorituskykyisiä vaihtoehtoja olivat kaaviot, joissa rakenne oli mahdollisimman litteä, sillä Firebasessa tiettyä solmua tarkasteltaessa tarkastellaan samalla myös kyseisen solmun lapsisolmuissa tapahtuvia muutoksia. Tämän vuoksi muutosten tarkastelu vaatii enemmän resursseja monisolmuisissa toteutuksissa verrattuna matalarakenteisiin ja harvasolmuisiin toteutuksiin. Hankaluutta rakenteen suunnittelussa lisäsi myös seikka, että Firebasen tietokantana toimii vain yksi JSON-tiedosto, eikä useiden tietokantojen ylläpitäminen ollut mahdollista opinnäytetyöntekijän käyttämässä ilmaisversiossa. Tämän johdosta yhteen JSON-tiedostoon oli tallennettava usean eri testiyrityksen laitetietoja. Näihin tietoihin kuitenkin suunniteltiin säännöt sekä rajaukset, jotta eri yritysten henkilöt eivät pääsisi tarkastelemaan muuta kuin oman yrityksensä laitteiden tietoja.

Kun rakenne oli suunniteltu, siirryttiin sääntöjen sekä rajausten suunnitteluun. Firebasessa niillä määrätään, mitä käyttäjä saa lukea sekä kirjoittaa JSON-tiedostosta, ja minkä tyylistä kirjoitettavan datan tulee olla. Näiden sääntöjen määrittely on erittäin tärkeää, sillä ilman niitä kuka vain sallittu käyttäjä voi lähettää esimerkiksi selaimensa konsolin kautta kyselyitä tietokantaan ja lukea, lisätä, poistaa tai muokata dataa ilman rajoituksia. Koska rakenne suunniteltiin jo aikaisemmin, oli sen päälle helppo laatia halutunlaiset rajaukset ja säännöt.

5.3.3 Toteutus

Toteutusvaihe suoritettiin samassa järjestyksessä kuin suunnitteluvaihe. Liikkeelle lähdettiin autentikoinnin luonnista. Kyseisen toiminnon lisääminen osaksi Firebase-tietokantaa oli erittäin helppoa, sillä siihen löytyi Googlen tarjoamia opetusvideoita, joissa autentikoinnin rakentaminen käytiin läpi pala palalta. Yksinkertaistetusti autentikoinnin lisäämiseen tarvitsi vain valita sisäänkirjautumiseen käytettävä metodi,

luoda käyttäjätunnukset sekä tehdä käyttörajapinta, josta kirjautuminen voitiin suorittaa. Kyseisestä käyttörajapinnasta kutsuttiin valmiin JavaScript-kirjaston autentikointifunktiota, joka ilmoitti paluuviestissä, oliko kirjautuminen onnistunut vai ei. Jotta tietojen lukeminen sekä kirjoittaminen tietokannasta onnistuisi vain kirjautuneilta käyttäjiltä, kehotettiin ohjeissa vielä muokkaamaan tietokannan säännöt koskemaan näitä vaatimuksia.

Autentikoinnin toteuttamisen jälkeen tietokantaan luotiin esimerkki dataa aikaisemmin suunnitellun rakenteen muodossa. Datan lisääminen onnistui helposti sekä Firebase-tietokannan hallintasivulta että verkkoselaimen konsoliin syötettävillä Firebasen kirjaston tarjoamilla kyselyillä. Kun esimerkki data oli lisätty, siirryttiin määrittämään halutulle rakenteelle rajaukset sekä säännöt. Tämä onnistui tietokannan hallintasivulla tarjotun työkalun avulla. Kyseinen työkalu koostui editorista sekä simulointiominaisuudesta, jolla tehtyjen rajauksien ja sääntöjen toimintaa pystyttiin testaamaan (ks. kuvio 9).

Kun tietokantaan kirjautuminen, rakenne sekä säännöt oli saatu toteutettua, syntyi lopputuloksena toimiva ja turvallinen testidataa sisältävä kokonaisuus, johon toteutuksen myöhemmässä vaiheessa esitettävät kyselyt olisivat lähetettävissä.

Kuvio 9. Pelkistetty esimerkki tietokannan säännöistä sekä testaussimulaattorista

5.4 Sijainnin seuranta ja päivitys

5.4.1 Yleistä

Koska reaaliaikaisesti seurattaviin laitteisiin liitettäviä GPS-laitteita ei vielä ollut hankittu, täytyi muuttuvan sijaintidatan keräämiseen suunnitella väliaikainen ratkaisu. Opinnäytetyöntekijä yritti aluksi löytää jonkun valmiin sovelluksen, jolla dataa oltaisiin voitu kerätä tai generoida, mutta sopivaa sovellusta kyseiseen tarpeeseen ei löytynyt. Tämän vuoksi hän päätyi kehittämään mobiililaitteille tarkoitetun sovelluksen, jolla seuranta sekä siitä syntyneen datan tallentaminen voitaisiin suorittaa. Koska Android-sovelluksen luominen olisi vaatinut uusien ohjelmistoympäristöjen pystyttämistä, tekijä päätyi luomaan verkkosovelluksen, jota kyseiset laitteet voisivat käyttää verkkoselaimensa avulla. Tärkeänä osana kyseistä ratkaisua toimi HTML:n tarjoama rajapinta nimeltään ”Geolocation”, joka hoiti käyttäjien senhetkisestä sijainnista raportoinnin.

5.4.2 Suunnittelu ja toteutus

Seurantasovelluksen toteutusta suunniteltaessa apuna käytettiin tehtyjä vaatimusmäärittelyjä. Vaatimusmäärittelyjen lisäksi sovellukseen lisättiin muutama uusi vaatimus, kuten mahdollisuus valita, minkä laitteen dataa tallennetaan. Kyseisen kokonaisuuden suunnittelu oli erittäin helppoa, koska sovellus oli yksinkertainen sekä pieni ja valittu rajapinta tarjosi juuri tarvittavat toiminnallisuudet sovelluksen toteuttamiseen.

Kyseisen sovelluksen toteuttaminen purettiin kolmeen eri rakennusvaiheeseen, jotka olivat yhdistäminen ja kirjautuminen, seuranta sekä tallennus. Yhdistäminen onnistui helposti, sillä Firebasessa oli mahdollista generoida tarvittavat konfigurointitiedot, jotka voitiin syöttää yhdistämisen suorittavalle funktiolle. Yhdistämisen jälkeen laite oli kirjattava tietokantaan sisälle, jotta se pystyisi muokkaamaan sijaintidataansa. Kyseiseen kirjautumiseen käytettiin Firebasen tarjoamaa autentikointifunktiota, jonka takaisinkutsufunktiossa pystyttiin tarkastelemaan, onko laite kirjautunut sisään vai ei. Jotta kirjautumistiedot voitiin syöttää funktiolle, luotiin kyseistä toimintoa varten tar-

vittavat elementit. Jos laite kirjattiin sisään onnistuneesti, käyttäjälle näytettiin elementit laitteen seuraamisen hallinnointia varten sekä päivittyvä tieto laitteen seurannan senhetkisistä tiedosta (ks. liite 2).

Seuraavaksi toteutettiin laitteen sijainnin seuraaminen, joka oli erittäin helppoa valitun rajapinnan ansiosta. JavaScriptissä kutsuttiin navigointiobjektin kautta oikeaa metodia, joka hoiti seurannan aloittamisen. Metodin ensimmäiseen parametriin voitiin syöttää takaisinkutsufunktio, jota kutsuttiin aina sijainnin muuttuessa. Jos seuranta haluttiin lopettaa, voitiin kutsua navigointiobjektin toista metodia, jolle syötettiin seurannan aloittamisessa vastauksena saatu seuranta id. Seurannan aloitus sekä lopetus käydään yksinkertaistetusti läpi kuviossa 10.

```
function startTracking()
{
  if(navigator.geolocation)
    nid = navigator.geolocation.watchPosition(updPosition, trackingError, {enableHighAccuracy: true});
  else
    alert("Geolocation is not supported by this browser.");
}
function stopTracking()
{
  navigator.geolocation.clearWatch(nid);
}
```

Kuvio 10. Yksinkertaisin versio seurannan aloittamisesta ja lopettamisesta

Viimeisenä vaiheena sovellusta rakennettaessa oli muuttuneen paikkatiedon tallentaminen tietokantaan. Tätä varten luotiin oma funktio, jota kutsuttiin aina sijaintidatan päivittyessä. Funktiossa purettiin parametrina saatu objekti haluttuun muotoon, joka sitten asetettiin Firebasen päivitysfunktion parametriksi. Päivitysfunktiota kutsuttaessa objektin data korvasi senhetkisen tiedon tietokannan määrätyssä sijainnissa. Samaisessa päivityskutsussa päivitettiin myös kuljettuihin reitteihin uusi reitti, sekä senhetkinen reittien lukumäärä. Kuviossa 11 on nähtävissä yksinkertaistettu esimerkki tietokannan päivitysfunktiosta.

```
var obj = {};
obj['Machine'] = data;
obj['Traces/'+tracecount] = data;
obj['Traces/TraceCount'] = traceCount;

firebase.database().ref(loc).update(obj, updError);
```

Kuvio 11. Yksinkertaistettu versio Firebase-tietokannan päivittämisestä

5.5 Karttaluokka

5.5.1 Yleistä

Karttaluokka oli PHP-kielellä toteutettu luokka, joka rakentui toteutuksen edetessä lopulliseen muotoonsa. Luokka ei tietenkään oikealta nimeltään ollut karttaluokka, mutta siitä käytetään tätä nimitystä tässä opinnäytetyössä. Alun perin opinnäytetyöntekijä suunnitteli tekevänsä kaksi erillistä karttaluokkaa, joista toinen olisi sisältänyt kaikki perusominaisuudet. Tästä perusominaisuudet sisältävästä luokasta olisi peritty toinen karttaluokka, joka olisi sisältänyt mm. reaaliaikaisten laitteiden näyttämiseen tarvittavia metodeja. Opinnäytetyöntekijä kuitenkin totesi, että kyseinen toteutus ei olisi tuonut paljoa lisäarvoa ja se olisi vain monimutkaistanut asioita turhaan, joten tekijä päätyi luomaan vain yhden karttaluokan, johon hän lisäsi ehdon, jolla kartalla pystyttiin tarkkailemaan myös reaaliaikaisia kohteita.

5.5.2 Luokan sisältö

Aluksi karttaluokkaan tehtiin metodit eri karttaelementtien hakuun. Näitä olivat esimerkiksi eri rajauslaatikot sekä valintapainikkeet, joita tarvittiin kartalla näytettävien objektien kuvaamiseen. Kyseiset metodit päätettiin luoda karttaluokkaan, jottei ohjelmoijien tarvitsisi aina tehdä elementtejä uudelleen ja että elementit saisivat yhtenäisen muodon sekä nimeämiskäytänteet.

Kun vuoroon tuli kartan lisääminen osaksi sovellusta, liitettiin kartan asetusten määrittely sekä kartan hakeminen osaksi luokkaa. Erilaisten karttaan sekä karttaobjekteihin liittyvien JavaScript funktioiden muodostuttua myös näiden noutaminen sisällytettiin samaan metodiin, kuin edellä mainitut.

Seuraavaksi karttaluokkaan lisättiin metodi Firebase-tietokannan määrittelyyn sekä sisäänkirjautumiseen. Kun reaaliaikaisten kohteiden tarkasteluun vaaditut JavaScript -funktiot saatiin luotua, lisättiin kyseisten JavaScript-funktioiden hakeminen metodiin. Metodiin tehtiin myös ehto, jolla voitiin määritellä, oliko sen hetkisellä käyttäjällä oikeudet hakea reaaliaikaisten kohteiden näyttämiseen vaaditut tiedot.

5.6 Käyttöliittymä sekä rajaukset

Käyttöliittymä koostui kolmesta pää elementistä: karttaobjektien valintoihin ja toimintoihin liittyvästä, itse kartan sisältävästä, sekä reittitiedon näyttämiseen tarkoitetusta elementistä (ks. liite 2). Elementtien luontiin käytettiin apuna ohjelmistokehyksen valmiita funktioita ja aikaisemmin luotuja karttaluokan metodeja. Näillä haettiin myös mm. käyttäjän rooli sekä asetettiin parametrejä ja rajauksia liittyen karttaluokkaan sekä sen sisällä haettaviin JavaScript tiedostoihin, kuten kuinka kauaksi kartalla pystyttiin tarkentamaan ja millä kielellä kartta näytettiin. Suurimpana rajauksena toimi kuitenkin jo edelläkin mainittu käyttäjän rooli, sillä jos käyttäjä oli asentaja, niin mitään reaaliaikaisten koneiden näyttämiseen liittyviä funktioita, elementtejä tai tietoja ei näytetty.

Karttaobjektien ikonit tehtiin skaalautuvalla vektorigrafiikalla (SVG). Se soveltui kyseiseen tarkoitukseen hyvin, koska sillä tehdyt ikonit säilyttivät kokonsa suhteessa karttanäkymään kartan tarkentamisesta riippumatta. Skaalautuvan vektorigrafiikan avulla oli myös mahdollista tehdä animoituja ikoneja, joten sitä käytettiin apuna reaaliajassa seurattavien kohteiden ikonien hahmottamisen parantamisessa.

5.7 Karttatoiminnot

5.7.1 Yleistä

Karttatoimintojen luominen oli viimeinen vaihe opinnäytetyönä syntyvän sovelluksen toteuttamisessa. Tekijä koki kyseisen vaiheen suorittamisen mielekkääksi, koska kaikki muut sovelluksen osa-alueet oli jo toteutettu. Tämän seurauksena hän pystyi keskittymään täysin karttatoimintojen rakenteen- sekä koodin suunnitteluun ja toteuttamiseen saadakseen lopputuloksesta mahdollisimman selkeän, kompaktin sekä suorituskykyisen paketin.

Ennen toteutusta sekä toteutuksen suunnittelua opinnäytetyöntekijä kertasi toiminnalliset sekä ei-toiminnalliset vaatimukset. Näiden vaatimusten pohjalta hän mm. priorisoi toteutettavat toiminnallisuudet sekä rajasi tekotavat, minkä jälkeen karkea kuva toteutettavasta suunnitelmasta alkoi muodostua.

5.7.2 Toteutus sekä sen suunnittelu

Karttatoimintojen toteutuksen suunnitelma koostui kolmesta osa-alueesta. Ensimmäiseen ja samalla ensimmäisenä toteutettavaan osa-alueeseen kuuluivat lähes kaikki asentajan ja vastuuhenkilön pakolliset vaatimukset sekä niiden toiminnallisuudet. Näitä olivat työmaiden, kuvien sekä halvempien laitteiden näyttäminen kartalla ja niitä koskevat toiminnallisuudet. Toiseen osa-alueeseen kuuluivat pelkästään vastuuhenkilöä koskevat vaatimukset. Näitä olivat kalliimpien laitteiden reaaliaikainen näyttäminen kartalla ja laitteiden tietojen sekä kuljetun reitin tarkastelu. Kolmanteen osa-alueeseen kuului karttaelementtien tyhjentäminen, sekä kaikki vapaaehtoiset vaatimukset mukaan lukien ne, mitä tekijä keksi opinnäytetyönteon aikana.

Ensimmäinen osa-alue

Karttatoimintojen luonti suunniteltiin toteutettavaksi ylhäältä alaspäin, jolloin aloitettiin lähimpänä käyttäjärajapintaa olevasta ja kaikille karttaobjektityypeille yhteisestä funktiosta, joka tarkasteli, mitä käyttäjä halusi nähdä kartalla ja haki näihin vaatimuksiin perustuen karttaobjekteille vaaditut tiedot. Tiedot haettiin Ajax-kutsulla luvussa 5.2.3 läpikäydyillä funktioilla, jotka palauttivat halutut datat JavaScript objektin muodossa. Kun Ajax-kutsu palautti tiedot onnistuneesti, kutsuttiin jokaisen käyttäjän

valitseman karttaobjektityypin kohdalla funktiota, jossa ikonin ja karttaobjektin luonti, määrittely ja sijoittaminen suoritettiin.

Jotta karttaobjektit voitiin näyttää kartalla, niille oli määriteltävä ikonit sekä sijainnit. Tämän jälkeen karttaobjektit luotiin rajapintatarjoajan kirjastosta löytyneen merkkaaja-luokan avulla. Jotta luotuja karttaobjekteja olisi helpompi hallita, niille rakennettiin samaisesta kirjastosta löytyneellä ryhmät-luokalla säiliöt, jotka jaoteltiin karttaobjektityypin mukaan. Näille säiliöille oli madollista antaa esimerkiksi kuuntelijoita, joita kaikki säiliön sisällä olevat karttaobjektit kuuntelisivat (ks. kuvio 12).

```
if(mapIconGroups[gno])
  mapIconGroups[gno].removeAll();
else
{
  mapIconGroups[gno] = new H.map.Group();
  mapIconGroups[gno].addEventListener('tap', function(o){map_ShowInfoBubble(o,gno)});
  hmap.addObject(mapIconGroups[gno]);
}

if(!mapIcons[gno])
  mapIcons[gno] = map_GetCustomizedIcon(gno);
for(i=0; i<rows.length; i++)
{
  row = rows[i];
  marker = new H.map.Marker({lat:row.Latitude, lng:row.Longitude}, {icon:mapIcons[gno]});
  marker.setData(map_AddDataToBubble(row, gno));
  mapIconGroups[gno].addObject(marker);
}
```

Kuvio 12. Ryhmien ja karttaobjektien luonti sekä sijoittelu

Kun ensimmäisen osa-alueen vaatimukset onnistuttiin näyttämään kartalla, niille aloitettiin tekemään toiminnallisuuksia. Ensimmäisenä toteutettiin infokupla, jossa karttaobjekteista kertovaa dataa voitaisiin esittää. Kuplaan oli mahdollista luoda myös painikkeita sekä muita elementtejä, joiden kautta eri toiminnallisuuksia toteutettiin karttaobjekteille. Kohteille eli työmaille kuplaan sijoitettiin reittihaku-painike, jolla reitinhaku sekä muodostus onnistui. Kyseinen toiminnallisuus oli moniosainen vaihe, joka muodostui oman sijainnin määrittämisestä, halutun reitin pyytämisestä, sen muotoilemisesta sekä piirtämisestä ja ajo-ohjeiden generoimisesta. Laitteille sekä kuville toiminnallisuuksien toteuttaminen oli helpompaa, sillä laitteiden kuplaan ei edes haluttu toiminnallisuutta ja kuvat-kuplassa täytyi näyttää vain tietokannasta haettava kuva, joka avautuisi isompaan ikkunaan tätä painettaessa.

Toinen osa-alue

Suunnitelman toista osa-aluetta lähdettiin toteuttamaan samalla lailla kuin ensimmäistä. Koska ensimmäisessä osa-alueessa luotu JavaScript-tiedosto ladattiin aina karttoja näytettäessä, sen funktioita pystyttiin käyttämään hyväksi myös reaaliaikaisten kohteiden näyttämisessä. Näistä esimerkkinä mainittakoon funktio, joka tarkisti mitä käyttäjä halusi tarkastella kartalla ja haki tarkastelun jälkeen näihin liittyvät tiedot. Suurin osa funktioista kuitenkin toteutettiin omaan JavaScript-tiedostoonsa, joka ladattiin vain, mikäli käyttäjällä oli oikeus tarkastella reaaliaikaisesti seurattavia laitteita.

Reaaliaikaisesti seurattavien laitteiden näyttäminen kartalla toteutettiin lähes samoin kuin muiden karttaobjektien näyttäminen. Ainoina eroina olivat seikat, että käytettävä ikoni oli rakennettava eri tavalla, ikoneita oli liikuteltava päivittyvän datan mukaan ja sijainnin hakuun oli avattava yhteys suoraan Firebase-tietokantaan, sen sijaan että tiedot olisi haettu vain kertaalleen. Yhteyden avaamiseen sekä sen sulkemiseen Firebase tarjosi kuitenkin valmiit funktiot, joilla tiedoissa tapahtuvien muutosten seuraaminen sekä seuraamisen lopettaminen oli erittäin helppoa (ks. kuvio 13).

```
for(key in mapAssets)
{
  if(!rows[key])
  {
    hmap.removeObject(mapAssets[key]['Marker']);
    fb.ref(loc).off();
    delete mapAssets[key];
  }
}
for(key in rows)
{
  if(!mapAssets[key])
  {
    mapAssets[key] = rows[key];
    fb.ref(loc).on('value', rtm_CreateOrMoveAsset);
  }
}
```

Kuvio 13. Seurattavan laitteen muutosten tarkastelun aloitus ja lopetus

Kun reaaliaikaisesti seurattavien laitteiden näyttäminen sekä liikuttelu oli saatu toteutettua kartalle, tekijä siirtyi työstämään vaadittuja toiminnallisuuksia laitteille. Laitetta näpäytettäessä avautuvaan infokuplaan lisättiin painike, jota painamalla voitiin tarkastella laitteen kulkemaa reittiä. Koska aikaisemmissa vaiheissa oli jo toteutettu laitteen reittidatan tallentaminen Firebase-tietokantaa, tekijän ei tarvinnut muuta kuin hakea kyseiset datat ja muotoilla tämän datan muodostama reitti kartalle.

Kolmas osa-alue

Kolmannessa osa-alueessa toteutettava toiminnallisuus karttaelementtien tyhjentämiseen onnistui helposti. Tämä johtui mm. siitä, että karttaluokasta metodeilla haettavat elementit sisälsivät aina samat yksilöivät tunnisteet, jonka johdosta esimerkiksi reittitieto elementti voitiin tyhjentää asettamalla tunnistetiedoilla löytyvän elementin arvo tyhjäksi. Lisäksi kartalla olevat karttaobjektit oli sijoiteltu omiin objektimuotoisiin säiliöihinsä, jotka sisälsivät metodin näiden karttaobjektien poistoon. Karttaobjektien lisäksi kartalta oli muistettava poistaa myös esimerkiksi kaikki turhaksi jääneet kuuntelijat.

Karttaelementtien tyhjentämistoiminnon lisäksi kolmanteen osa-alueeseen sisältyi vapaehtoisten toiminnallisuuksien toteuttaminen käytettävien resurssien puitteissa. Kyseisiä kartalle toteutettavia toiminnallisuuksia tekijä oli keksinyt sekä opinnäytetyötä aloittaessaan että itse prosessin aikana. Yksi tällainen toiminnallisuus, jota opinnäytetyöntekijä aloitti kehittämään, oli henkilöiden päivittäisten työsijaintien suunnittelu ja tarkastelu kartalla. Kyseisellä toiminnolla työntekijät olisivat voineet esimerkiksi tarkastella työtehtäviensä sijainteja ja suunnitella työtehtävilleen loogisimman toteutusjärjestyksen. Lisäksi he olisivat voineet merkitä töiden vaiheet, jolloin valmiit oltaisiin näytetty kartalla vihreällä, keskeneräiset punaisella ja työnalla olevat keltaisella. Opinnäytetyöntekijä ehti suunnitella kyseiselle toiminnolle jo mm. tietokantamuutokset, mutta hän kuitenkin päätti jättää kyseisen toiminnon toteuttamatta, koska opinnäytetyöhön käytettävät resurssit olisivat ylittyneet.

6 Tulokset

Tulokset ovat opinnäytetyön tärkein osa-alue, koska siinä työssä saadut tulokset sekä havainnot niputetaan yhteen ja niiden perusteella tehdään johtopäätökset, joita

myöhemmässä vaiheessa käytetään valintojen tukena. Tässä opinnäytetyöstä haluttiin erityisesti selvittää mitkä karttarajapinnat sopisivat parhaiten toimeksiantajan vaatimuksiin, kuinka ne soveltuisivat osaksi jo valmista kokonaisuutta ja mitä asioita tällaista toteutusta tehtäessä olisi huomioitava. Lisäksi haluttiin selvittää olemassa olevien teknologioiden soveltuvuus karttasovelluksen pohjaksi sekä mitä teknologioita olisi hankittava näiden tueksi.

Opinnäytetyön tuloksena syntyi toimeksiantajan, oppilaitoksen sekä opinnäytetyöntekijän yhdessä laatimilla toiminnallisuuksilla varustettu karttasovellus, joka toimi esimerkkinä sovelluksen toimivuudesta Adminet-ympäristössä (ks. liite 2).

6.1 Karttarajapinta

Opinnäytetyössä lopulliseen toteutukseen valittu Heren karttarajapinta todettiin toimivaksi vaihtoehdoksi osaksi Adminet kokonaisuutta. Tätä tukivat muun muassa seikat, että se pärjäsi hyvin hintavertailussa muiden rajapintojen kanssa, se tarjosi toimeksiantajan vaatimaan tarpeeseen sopivan lisenssin, se toimi jouhevasti osana Adminettiä sekä se sisälsi runsaasti eri lisäominaisuuksia, jotka tarjosivat laajat mahdollisuudet jatkokehittämiseen. Näiden lisäksi rajapinnan vakaiden, jatkuvien sekä positiivisten tulevaisuuden näkymien vuoksi sitä uskaltaisi käyttää osana kaupallisia sovelluksia vielä vuosien sekä vuosikymmenienkin päästä.

Täydellinen Heren tarjoama karttarajapinta ei kuitenkaan ole. Se sisältää muutamia puutteita ja sen dokumentaatio sekä kehittäjätuki ei ole yhtä kattava, kuin esimerkiksi Googlen tarjoaman ratkaisun. Lisäksi sen lähdekoodi ei ole avointa ja muokattavissa olevaa, mikä mahdollistaa pahimmillaan toimittajan loukkuun joutumisen.

Tärkeimmäksi huomioon otettaviksi seikaksi karttarajapintaa valittaessa todettiin tarkan vaatimusmäärittelyn tekeminen, joka kattaisi sekä nykyiset- että tulevat vaatimukset. Opinnäytetyössä hyvällä vaatimusmäärittelyllä onnistuttiin seulomaan suuresta määrästä karttarajapinnantarjoajia epäsopivimmat pois, jonka jälkeen vaatimusmäärittelyä tarkentamalla pystyttiin jäljelle jääneistä valitsemaan potentiaalisimmat vaihtoehdot tarkempaan tutkimiseen. Tarkka vaatimusmäärittely on lisäksi enemmänkin sääntö kuin kehotus karttarajapintoja valittaessa, sillä ilman sitä yritys voi joutua tilanteeseen, jossa väärän karttarajapinnan valinta aiheuttaa vuosien

päästä mittavat tappiot mm. puuttuvien ominaisuuksien tai ylittyvien rajoitusten vuoksi.

6.2 Tietokannat

Tutkittaessa valmiina olevien tietokantaratkaisujen sopivuutta reaaliaikaisesti päivittyvän datan tallentamiseen päädyttiin tulokseen, että nykyiset tietokantaratkaisut eivät sovellu kyseisen datan tallentamiseen mm. määritetyn sijaintinsa sekä relaatiomaisuutensa vuoksi. Myös niillä toteutus olisi onnistunut, mutta siitä olisi tullut erittäin raskas, varsinkin käyttäjämäärien kasvaessa. Tämän vuoksi tekijä päätyi tutkimaan muita tietokantaratkaisuja, joista voittajaksi valikoitu Firebasen tarjoama epärelaatiotietokanta.

Firebase tarjosi yksinkertaisen käyttöliittymän tietokannan muokkaamiseen sekä ylläpitämiseen ja sisälsi kattavat kirjastot sen käyttöön. Opinnäytetyön aikana Firebase toimi jouhevasti ja ilman suurempia ongelmia, mutta huonoja puoliakin siitä löytyi. Monimutkaisten kyselyjen suorittaminen tietokantaan oli mahdotonta ja käyttäjämäärien kasvaessa hinta saattaisi huomaamatta tulla kalliiksi. Jos reaaliaikainen tietojenpäivitys haluttaisiin toteuttaa tuotantoon asti, kannattaisi tietokantaratkaisuja siis vielä vertailla tarkemmin. Firebasea kannattaa opinnäytetyöntekijän mielestä pitää kuitenkin potentiaalisena vaihtoehtona, sillä se tarjoaa suorituskykyisen sekä vakaan ja skaalautuvan vaihtoehdon.

Opinnäytetyönä syntyneen karttasovelluksen tietovarastoina toimineiden relaatio- sekä epärelaatiotietokantojen muutokset tehtiin vaatimuksia vastaaviksi. Relaatiotietokantamuutokset olisivat tekijän mielestä käytettävissä lopullisessa sovelluksessa sellaisinaan, kun taas epärelaatiotietokanta muutoksiin voitaisiin tehdä pieniä muutoksia mm. suorituskyvyn parantamiseksi.

6.3 Laitteiden reaaliaikainen seuraaminen

Laitteiden reaaliaikaiseen seuraamiseen ei alun perin vaadittu seurantasovelluksen toteuttamista, vaan pelkkä esimerkkidata olisi riittänyt. Tekijä kuitenkin halusi toteuttaa sovelluksen itse, jotta hän saisi samalla tietoa siitä, mitä kannattaisi ottaa huomioon, kun tällaista sovellusta lähdettäisiin kehittämään.

Sovellus muodostui nopeasti ilman suurempia ongelmia, jonka seurauksena seuranta dataa pystyttiin keräämään kattavasti erilaisina päivinä. Karttasovellusta toteuttaessaan tekijä huomasi, että päivästä riippuen data oli erilaista. Kirkkaana päivänä data oli määritelty muutaman metrin tarkkuudella, kun taas sateisina päivinä data saattoi olla määritetty jopa 5-10 metrin tarkkuudella (ks. kuvio 14). Tästä johtuen, jos myöhemmässä vaiheessa laite haluttaisiin sijoittaa tielle, olisi siihen käytettävä avuksi esimerkiksi Heren tapauksessa käänteistä geokoodausominaisuutta. Tämä ominaisuus muuttaisi sijainti koordinaatit osoitteeksi, jonka perusteella laite voitaisiin sijoittaa kartalla siihen todennäköisimpään pisteeseen, jossa se oli tiedonantohetkellä. Käänteistä geokoodausominaisuutta käytettäessä kannattaa kuitenkin huomioida sille määritellyt rajat, jotta ne eivät ylittyisi.

Kuvio 14. Sijaintidata erilaisina päivinä

6.4 Karttasovellus

Konkreettisena tuloksena opinnäytetyössä syntynyt karttasovellus täytti kaikki sille laaditut toiminnallisuudet ja määritti helposti lähestyttävän pohjan uusien toiminnallisuuksien luontiin. Kyseinen karttasovellus antoi varmuuden karttarajapinnan toimivuudesta osana järjestelmää ja se toimi hyvänä esimerkkinä mahdollisuuksista, mitä karttasovellus voisi olla lopullisessa ympäristössä.

6.5 Jatkokehitys

Opinnäytetyön valmistuttua tekijä aloittaa Googlen tarjoaman karttarajapinnan testaamisen. Testauksen päätteeksi molempien rajapintojen tuloksia verrataan keskenään ja pohditaan voisiko toinen soveltua osaksi Adminettiä. Jos päätös on myönteinen ja karttarajapinnaksi valikoituu Heren tarjoama ratkaisu, voitaisiin opinnäytetyönä syntynyttä sovellusta käyttää pienten muutosten jälkeen jatkokehityksen pohjana.

Jos lopullisessa karttasovelluksessa haluttaisiin näyttää tietoa liikkuvista laitteista, olisi laitteisiin toteutettava seuranta tehtävä täysin uudelleen. Syynä tähän on se, että opinnäytetyössä seurantaan tarkoitettu sovellus toteutettiin selainympäristöön, joka tuskin liikkuvien laitteiden tapauksessa olisi se paras vaihtoehto. Opinnäytetyön seurantasovelluksen tuloksia voitaisiin kuitenkin käyttää hyväksi kyseistä toteutusta suunniteltaessa sekä toteuttaessa.

Mikäli edellä mainittu toiminnallisuus haluttaisiin lisätä lopulliseen karttasovellukseen, voitaisiin reaaliaikaisen tiedon tallentamiseen soveltuvien tietokantaratkaisujen tarkemman vertailun pohjana käyttää opinnäytetyössä hankittua tietoa. Jos tarkastelun jälkeen käyttöön valittaisiin opinnäytetyöhönkin valikoitunut ratkaisu, voitaisiin opinnäytetyön epärelaatiotietokannan rakennetta sekä sääntöjä käyttää hyväksi tietyissä määrin. Lisäksi opinnäytetyöntekijän kartuttama tietämys vähentäisi tietokannan rakenteen suunnitteluun ja toteutukseen vaadittavaa työtä.

Opinnäytetyön aikana rakentunut karttaluokka kävisi jatkokehitykseen sellaisenaan, jos käyttöön valikoituisi opinnäytetyössä käytetty karttarajapinta. Jos rajapinta kuitenkin olisi eri tarjoajalta, karttaluokkaan pitäisi tehdä muutoksia. Mitä luultavimmin

nämä muutokset olisivat kuitenkin niin pieniä että koko luokkaa ei tarvitsisi rakentaa uudestaan.

6.6 Johtopäätökset

Opinnäytetyössä onnistuttiin vastamaan toimeksiantajan kysymyksiin ja esittämään mahdollisia toteutusvaihtoehtoja, käytettäviä teknologioita sekä tietokantaratkaisuja. Työssä onnistuttiin valikoimaan potentiaalisimmat karttarajapintatarjoajat, vertailemaan näitä keskenään ja lopulta valikoimaan niistä kaksi parasta testivaiheeseen. Testivaiheessa onnistuttiin selvittämään karttasovelluksen toteutukseen liittyvät säännöt, rajaukset, ohjeet sekä yleisimmät sudenkuopat.

Testiin valikoitunut karttarajapinta toimii hyvin osana toimeksiantajan järjestelmään ja on potentiaalinen vaihtoehto lopulliseen toteutukseen. Se sisältää mm. useita laajennuksia, on kohtuuhintainen, tarjoaa sopivan lisenssin sekä on vakaa. Huonoja puolia siinä mm. ovat rajalliset dokumentit, suljettu lähdekoodi sekä muutamat pienet puutteet.

Valitut tietokantaratkaisut tukivat hyvin toteutettua kokonaisuutta ja osoittautuivat sopiviksi määriteltyihin tarkoituksiinsa. Tulevaisuudessa näitä ratkaisuja voidaan vielä vertailla tarkemmin muihin mahdollisiin vaihtoehtoihin parhaimman kokonaisuuden saavuttamiseksi.

Opinnäytetyönä valmistunut sovellus täytti kaikki vaaditut toiminnallisuudet ja siitä tuli toivotunlainen. Tämän vuoksi sitä voitaisiin käyttää osittain lopullisen tuotteen pohjana. Sovellus loi myös lisävarmuutta sille, että karttarajapinta tarjoaa runsaasti eri mahdollisuuksia Adminetin kehitykseen.

7 Pohdinta

7.1 Työn onnistuminen

Opinnäytetyön tuloksiin voidaan olla tyytyväisiä, sillä ne vastaavat täysin odotettua lopputulosta. Sekä tutkimis- että kehitystyöt sujuivat odotetusti eikä suurempia ongelmia kohdattu. Koko työn laajuus oli hyvä, sillä se mahdollisti tarkan tutustumisen

eri osa-alueisiin sekä teknologioihin jättäen silti aikaa toteutukselle ja raportoinnille. Myös sen luonne oli tekijälleen sopiva, koska se tarjosi paljon uutta opittavaa, mutta sisälsi silti myös tuttuja teknologioita.

Haastavinta opinnäytetyössä oli karttarajapintojen lisenssien, maksujen sekä muiden tietojen selvitys, sillä näistä kertova tieto oli tietyillä rajapintatarjoajilla ripoteltu ympäri heidän sivustoaan. Tämän johdosta kokonaisen sekä selkeän kuvan saaminen kyseisistä tiedoista oli hankalaa ja opinnäytetyön tekijä joutuikin varmistelemaan näiden tietojen paikkansapitävyyttä asiaan erikoistuneelta konsulttitalolta.

Lisätutkittavaa opinnäytetyön jälkeen jäi runsaasti, sillä karttarajapinnat tarjoavat paljon lisäominaisuuksia, joiden hyödyllisyyttä järjestelmälle voitaisiin tutkia. Lisäksi tarkempi selvitys epärelaatiotietokannoista sekä niiden laajemmista hyödyistä Adminetissä voisivat avata mielenkiintoisia näkökulmia.

7.2 Työn hyödyt

Toimeksiantajalle opinnäytetyö toi varmuutta karttasovelluksen toimivuudesta osana olemassa olevaa järjestelmää sekä sen mahdollisuuksista eri toiminnallisuuksien tukena. Lisäksi se oli osoitus siitä, että ammattikorkeakoulussa suoritettavasta opinnäytetyöstä voi olla konkreettista hyötyä yritykselle, mahdollistaen näin myös myöhemmässä vaiheessa opinnäytetöiden hyödyntämisen.

Tekijälleen opinnäytetyö antoi ensimmäisen kosketuksen työelämässä suoritettavan isomman projektin suunnitteluun, toteutukseen, hallintaan sekä koko prosessin läpivientiin. Se lisäsi paljon tekijän itsevarmuutta ja oli osoitus siitä, että hän hallitsee tarvittavat taidot sekä tietotason, jotta kyseisenlaisten projektien läpivienti onnistuisi. Lisäksi se todisti, että tekijä on kykenevä huomioimaa mahdollisuudet sekä vaaranpaikat, jotka voisivat vaikuttaa sovellukseen toimintaan, joko nyt tai tulevaisuudessa.

Suuremmalle yleisölle opinnäytetyö tarjoaa vinkkejä ja ohjeita itselleen sopivan karttarajapinnan valintaan. Se myös esittelee monia tämän hetken suosituimpia vaihtoehtoja ja herättelee lukijaansa pohtimaan todellisia vaatimuksiaan rajapintaa kohtaan, nyt sekä tulevaisuudessa. Tämän lisäksi se tarjoaa katsauksen epärelaatiotietokantojen maailmaan ja esittelee tarkemmin Firebase-tietokannan toimintaa.

7.3 Yhteenveto

Kokonaisuudessaan tämä opinnäytetyö oli kaikkia osapuolia hyödyttävä projekti, joka edisti toimeksiantajan sekä ammattikorkeakoulun yhteistyötä, mahdollisti toimeksiantajan idean testaamisen turvallisessa ympäristössä ja edisti tekijän opinto- sekä työuraa. Lopputuloksena syntyi karttasovellus, joka oli eheä kokonaisuus ja tarjosi hyvät jatkokehityspuitteet. Kaikki opinnäytetyön vaatimukset saatiin toteutettua onnistuneesti ja niiden toimivuus osana järjestelmää todettiin hyväksi. Muun muassa näiden edellä mainittujen seikkojen vuoksi tätä opinnäytetyötä voidaan pitää onnistuneena.

Lähteet

About HERE. N.d. Määritelmä Herestä heidän omalla verkkosivullaan. Viitattu 10.8.2018. https://www.here.com/en/company/about-here-technologies/about-us

About Mapbox. N.d. Mapboxin kotisivut. Viitattu 9.7.2018. https://www.mapbox.com/about/

About OpenStreetMap. 2017. OpenStreetMapin esittely sen omalla verkkosivulla. Viitattu 8.7.2018. https://wiki.openstreetmap.org/wiki/About_OpenStreetMap

Admicom Oyj listautui pörssiin. 2018. Teollisuuden Maailma 2.5.2018. Viitattu 27.7.2018. https://www.admicom.fi/asiakaslehti/teollisuuden-maailma-12018/admicom-listautui-porssiin/

Azarian, I. 2013. A Review of Software Version Control: System, Benefits, and Why it Matters. Blogijulkaisu versionhallinnasta seguetechin verkkosivustolla. Viitattu 2.8.2018. https://www.seguetech.com/a-review-of-software-version-control-systems-benefits-and-why-it-matters/

Baker, M. 2009. What is a Software Framework? And why should you like 'em?. Blogijulkaisu cimetrix-verkkosivustolla. Viitattu 2.8.2018. https://www.cimetrix.com/blog/bid/22339/what-is-a-software-framework-and-why-should-you-like-em

Chrome DevTools. 2018. Chrome DevToolsin verkkosivusto. Viitattu 1.8.2018. https://developers.google.com/web/tools/chrome-devtools/

Computer Programming Languages. N.d. Määritelmä Computer Sciencen verkkosivustolla. Viitattu 1.8.2018. https://www.computerscience.org/resources/computer-programming-languages/

Creating your own tiles. 2018. Itse ylläpidetyistä karttalaatoista kertova sivu OpenStreetMapin verkkosivuilla. Viitattu 8.7.2018. https://wiki.openstreetmap.org/wiki/Creating_your_own_tiles

Dart Programming Overview. N.d. Dat ohjelmoinnin verkko-opas. Viitattu 1.12.2018. https://www.tutorialspoint.com/dart_programming/dart_programming_quick_guide.htm

Dart Programming Tutorial. N.d. Dart ohjelmoinnin verkko-opas. Viitattu 1.12.2018. https://www.tutorialspoint.com/dart_programming/index.htm

Firefox Developer Tools. 2014. Firefox Developer Toolsin verkkosivusto. Viitattu 1.8.2018. https://developer.mozilla.org/fi/docs/Tools

Google Maps Platform FAQ. 2018. Yleisimmät kysymykset Googlen kartta-alustasta. Viitattu 9.8.2018. https://developers.google.com/maps/faq

IDE. 2015. Määritelmä TechTermsin verkkosivustolla. Viitattu 31.7.2018. https://techterms.com/definition/ide

Introducing Visual SourceSafe. N.d. Visual SourceSafen esittely Microsoftin verkkosivustolla. Viitattu 2.8.2018. https://msdn.microsoft.com/en-us/library/3h0544kx(v=vs.80).aspx

Kanat-Alexander, M. 2008. What Is Software Design?. Blogijulkaisu Code Simplicity -verkkosivustolla. Viitattu 31.7.2018. https://www.codesimplicity.com/post/what-is-software-design/

Laravel - Overview. N.d. Laravelin määrittely tutorialspointin verkkosivustolla. Viitattu 2.8.2018. https://www.tutorialspoint.com/laravel/laravel_overview.htm

MySQL Enterprise Edition. N.d. MySQL Workbenchin määrittely mysql.com-verkkosivustolla. Viitattu 7.7.2018. https://www.mysql.com/products/workbench/

NuSphere PhpED - The World Famous PHP IDE. N.d. PhpED-ohjelmistoympäristön esittelysivu NuSpheren kotisivuilla. Viitattu 31.7.2018. http://www.nusphere.com/products/phped.htm

ODC Open Database License (ODbL) Summary. N.d. Helposti ymmärrettävä versio ODbL 1.0 lisenssistä opendatacommons-verkkosivulla. Viitattu 08.07.2018. https://opendatacommons.org/licenses/odbl/summary/

OpenStreetMap Server Setup. N.d. Geofarbrikin tarjoamat palvelut ja hinnat OpenStreetMap serverin alustukseen. Viitattu 8.7.2018. http://www.geo-fabrik.de/services/server.html

Our map data. N.d. Mapboxin karttadatan määrittely Mapboxin verkkosivustolla. Viitattu 9.7.2018. https://www.mapbox.com/help/how-mapbox-data-works/#data-sources

Paakki, J. 2010. Ohjelmistojen vaatimusmäärittely. Viitattu 30.7.2018. https://www.cs.helsinki.fi/u/paakki/Vaatimus-11-Luentokalvot-1.pdf

Plans & Pricing. N.d. Mapboxin karttarajapinnan hinnoittelu. Viitattu 9.8.2018. https://www.mapbox.com/pricing/

Plans And Pricing. N.d. Heren karttarajapinnan hinnoittelu. Viitattu 10.8.2018. https://developer.here.com/plans

Quinn, S. N.d. What is web mapping API?. Artikkeli karttarajapinnoista PennStaten yliopiston verkkosivuilla. Viitattu 7.7.2018. https://www.e-education.psu.edu/geog585/node/763

Rouse, M. 2013. Google Maps. Määritelmä TechTargetin verkkosivustolla. Viitattu 9.8.2018. https://whatis.techtarget.com/definition/Google-Maps

Rouse, M. 2017a. database (DB). Määritelmä TecTargetin verkkosivustolla. Viitattu 3.8.2018. https://searchsqlserver.techtarget.com/definition/database

Rouse, M. 2017b. Java Platform, Enterprise Edition (Java EE). Määritelmä TheServerSide verkkosivustolla. Viitattu 1.12.2018. https://www.theserverside.com/definition/J2EE-Java-2-Platform-Enterprise-Edition

Serra, J. 2015. Relational databases vs Non-relational databases. Blogijulkaisu relaatio- sekä epärelaatiotietokannoista sekä niiden vertailusta. Viitattu 3.8.2018.

https://www.jamesserra.com/archive/2015/08/relational-databases-vs-non-relational-databases/

Silver, D. 2017. How Self-Driving Cars Work. Blogijulkaisu medium-verkkosivustolla. Viitattu 26.7.2018. https://medium.com/udacity/how-self-driving-cars-work-f77c49dca47e

The Definitive Guide to Yii 2.0. N.d. Yii 2.0-ohjelmistokehyksen esittely yiiframeworkin verkkosivustolla. Viitattu 2.8.2018. https://www.yiiframework.com/doc/guide/2.0/en/intro-yii

Tietoa meistä. N.d. Admicom Finland Oy:n kotisivut. Viitattu 27.7.2018. https://www.admicom.fi/yritys/#tietoa-meista

Tile Usage Policy. N.d. OpenStreetMapin karttalaatoille määritellyt käyttöehdot OpenStreetMap Foundationin verkkosivuilla. Viitattu 8.7.2018. https://operations.osmfoundation.org/policies/tiles/

Welcome to phpDesigner 8. N.d. phpDesigner 8-ohjelmistoympäristön esittelysivu mpsoftwaren kotisivuilla. Viitattu 1.8.2018. http://www.mpsoftware.dk/phpdesigner.php

What is JavaScript?. N.d. JavaScriptin määrittely tutorialspointin verkkosivustolla. Viitattu 1.8.2018. https://www.tutorialspoint.com/javascript/javascript_overview.htm

What is MySQL?. N.d. MySQL-tietokantaohjelmiston määrittely sen omalla verkkosivulla. Viitattu 6.8.2018. https://dev.mysql.com/doc/refman/5.7/en/what-is-mysql.html

What is PHP?. N.d. PHP:n määritelmä PHP:n verkkosivustolla. Viitattu 1.8.2018. http://php.net/manual/en/intro-whatis.php

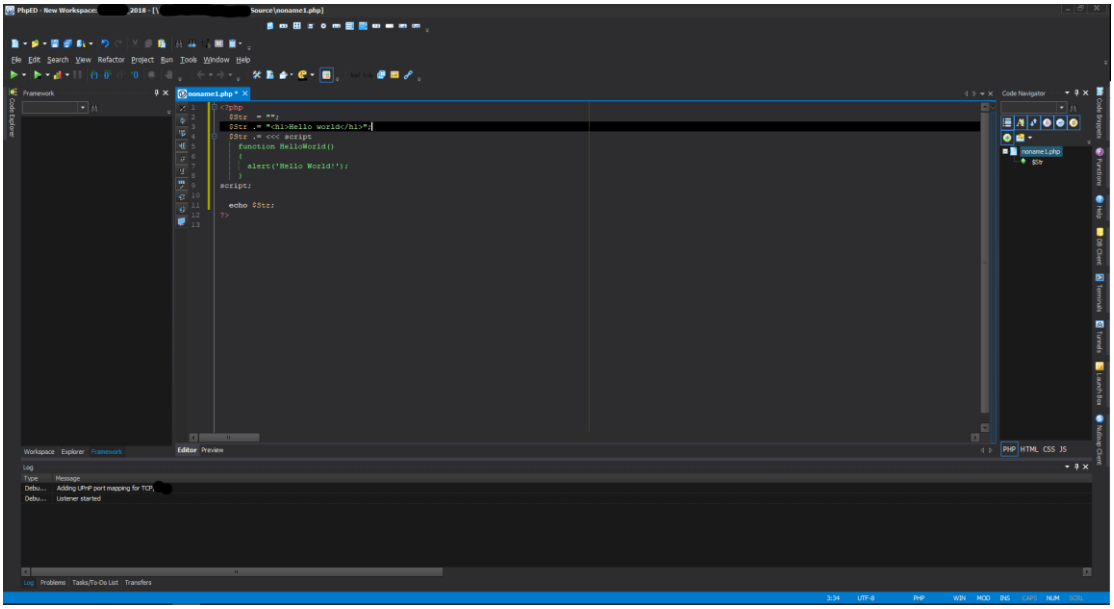
What is RethinkDB?. N.d. RethinkDB-tietokannan määrittely sen kotisivulla. Viitattu 6.8.2018. https://rethinkdb.com/faq/

Wingerath, W. 2017. A Real-Time Database Survey: The Architecture of Meteor, RethinkDB, Parse & Firebase. Artikkeli reaaliaikaiseen tallennukseen soveltuvien tietokantojen tutkimuksesta medium-verkkosivustolla. Viitattu 6.8.2018. https://medium.baqend.com/real-time-databases-explained-why-meteor-rethinkdb-parse-and-firebase-dont-scale-822ff87d2f87

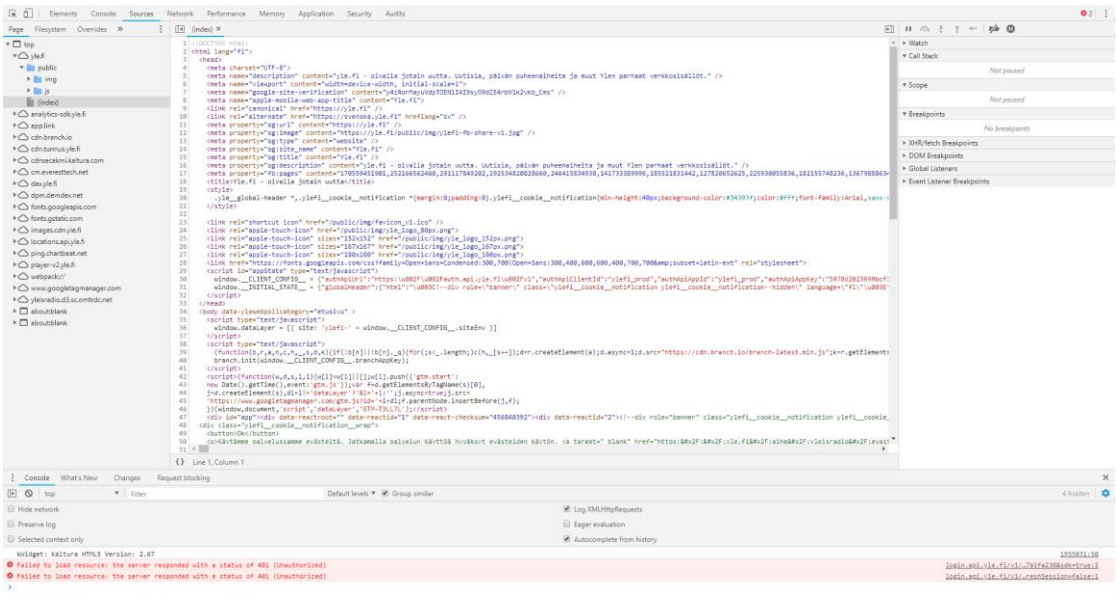
Liitteet

Liite 1. Kuvakaappauksia käytetyistä sovelluksista

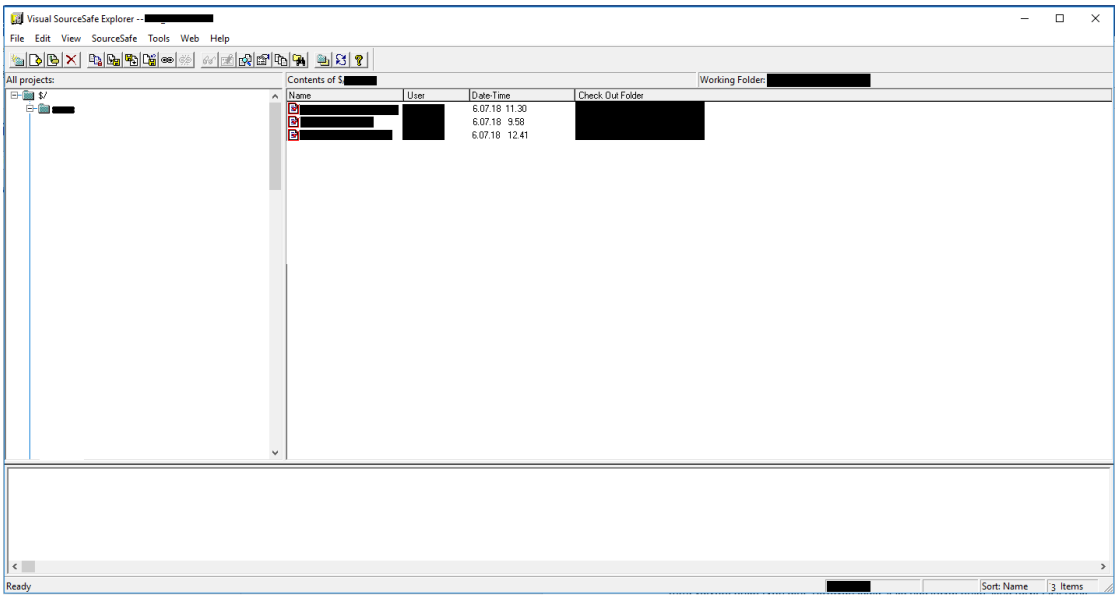
PhpED



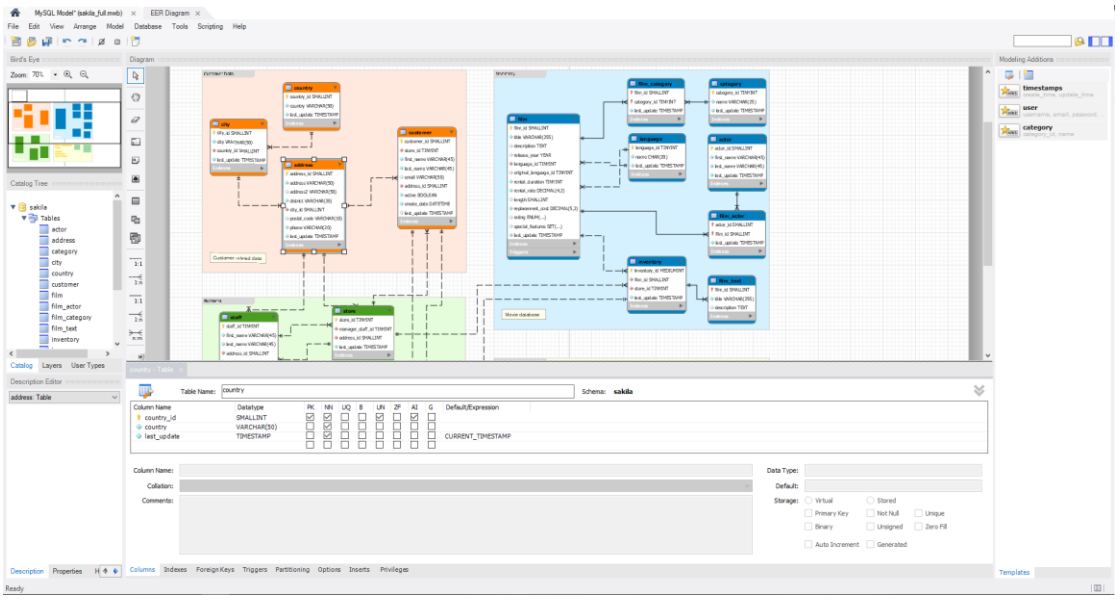
Google DevTools



Visual SourceSafe

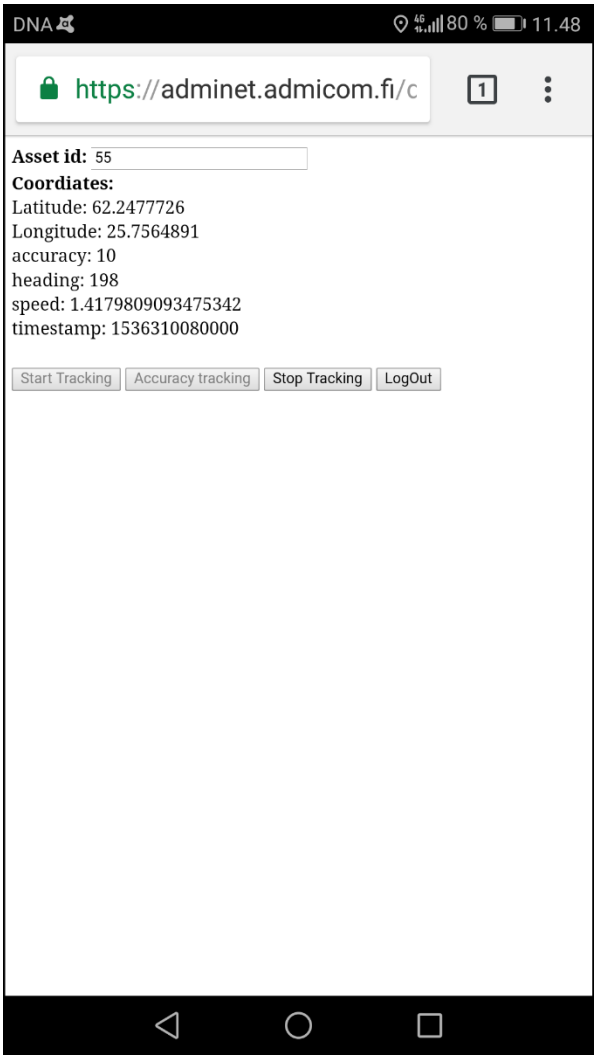


MySQL Workbench



Liite 2. Kuvakaappaksia käyttöliittymistä sekä toiminnoista

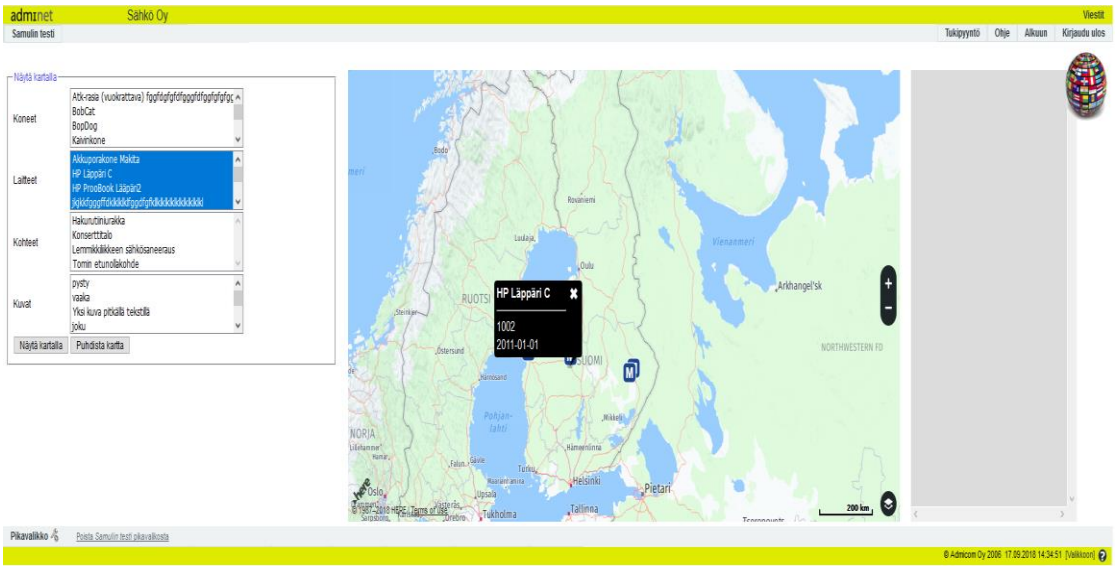
Reaaliajassa seurattavien laitteiden seurantasovellus



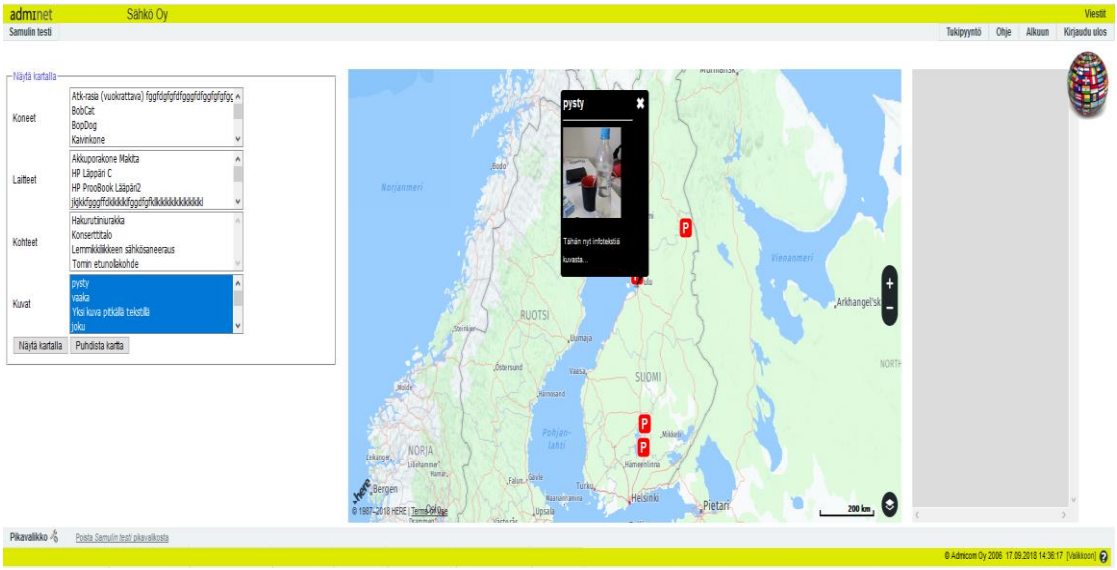
Sovelluksen käyttöliittymä sekä valittujen objektien näyttäminen kartalla

Kohteiden tarkastelu, reitin haku kohteeseen ja reittitiedon näyttäminen

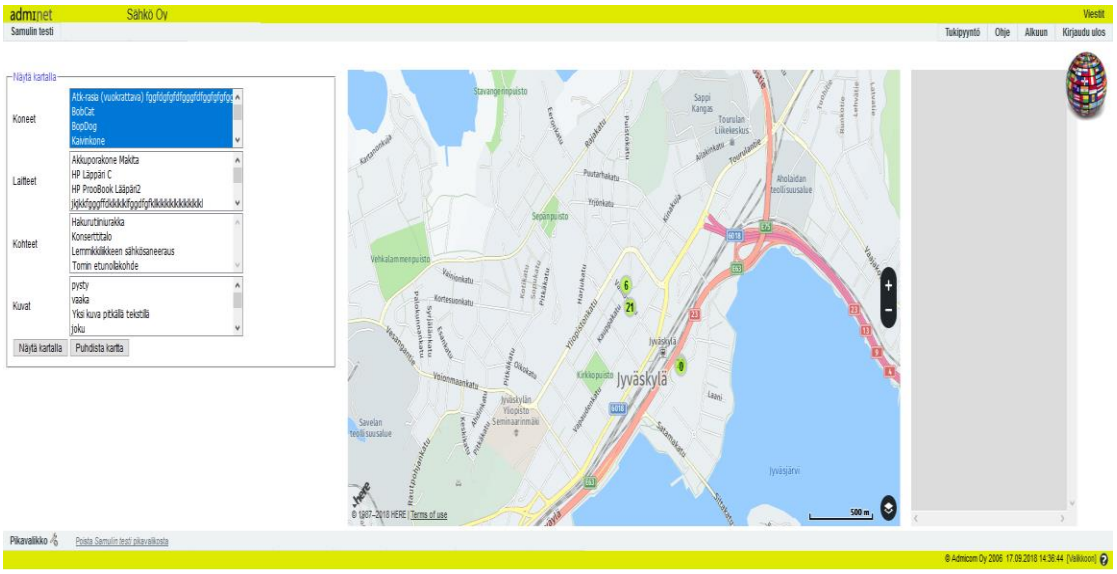
Halvempien laitteiden- sekä niiden tietojen tarkastelu



Kuvien tarkastelu



Reaaliajassa seurattavien laitteiden tarkastelu



Reaaliajassa seurattavien laitteiden kulkeman reitin tarkastelu