Haaga-Helia
ammattikorkeakoulu Oy

Manuaalinen säähavaintojen laaduntarkistus – web-palvelun kehittäminen

Paula Juntti

Opinnäytetyö
Tietojenkäsittelyn koulutusohjelma 2015

Haaga-Helia ammattikorkeakoulu Oy

Tiivistelmä

19.5.2015

Tekijä(t)
Paula Juntti

Koulutusohjelma
Tietojenkäsittelyn koulutusohjelma

Opinnäytetyön otsikko
Manuaalinen säähavaintojen laaduntarkistus – web-palvelun kehittäminen

Sivu- ja liitesivumäärä
34 + 9

Opinnäytetyön otsikko englanniksi
Weather observation's Human Quality Control – web service's development.

Tässä raportissa kuvataan toiminnallisen opinnäytetyön suunnittelua ja toteutusta. Opinnäytetyössä kehitettiin uusia toimintoja säähavaintojen laaduntarkistusohjelmistoon. Ohjelmisto on alun perin kehitetty FNEP2-projektille. Opinnäytetyön tavoitteena oli tuottaa viisi uutta manuaalisen säähavainnon laaduntarkistuksen toimintoa sovellukseen. Näillä toiminnoilla voi hakea, muokata ja kuvantaa säähavaintodataa. Varsinkin kuvantamisen toiminnot helpottavat sovelluksen käyttäjien työtä säähavaintojen laadunvalvonnassa. Projektista rajattiin ulos sovelluksen käyttöönotto.

Opinnäytetyön toimeksiantaja oli Ilmatieteenlaitos. Sovelluksen kehityksessä käytettiin pääasiassa PHP 5- ja JavaScript-ohjelmointikieliä. Raportin tietoperustassa perehdytään säähavaintojen laaduntarkistusmenetelmien perusteisiin ja sovelluksen kehityksessä käytettyihin teknologioihin. Raportissa käydään myös läpi sovelluksen kehityksen eri vaiheita ja projektin valmistumisen myötä syntyneitä tuloksia.

Raportin viimeisessä luvussa käydään läpi ohjelmiston jatkokehitysehdotukset ja pohditaan koko projektin sujumista ja opinnäytetyön kirjoittajan kehittymistä projektin aikana.

Asiasanat
JavaScript, PHP, sääpalvelu, verkko-ohjelmointi

HAAGA-HELIA
University of Applied Sciences

Abstract

19.5.2015

Author(s)
Paula Juntti

Degree programme
Business Information Technology

Report/thesis title	**Number of pages and appendix pages**
Human Quality Control in Weather Observation Web Services Development	34 + 9

This report describes the design and implementation carried out in the thesis project. The object of the thesis project was to develop five new features to weather observation Quality Control (QC) software. The software is developed for the Finnish-Nepalese Project: Improved capability of the Government of Nepal to respond to the increased risks related to the weather-related natural disasters caused by climate change 2 (FNEP2). With these four features the user of the software can search, edit and visualize weather observation data. The visualization features will make the quality control of the weather observation data significantly easier. The software deployment was excluded from this thesis.

The commissioner for this thesis project is the Finnish Meteorological Institute (FMI). The main programming languages used were PHP 5 and JavaScript. The theoretical part of this report includes the basics of weather observation quality control and also the basics of the technologies used in the development. There is also a chapter describing the different development phases of this project. The results of the project are introduced in detail.

The final part of the study presents the further development proposals and the general experiences throughout this project. The last chapter also includes the author's reflections on professional and personal growth in the thesis project.

Keywords
meteorological service, JavaScript, PHP, quality control, web development, weather observation data

Sisällys

1 Johdanto

Opinnäytetyön tavoitteena oli toteuttaa lisää toiminnollisuuksia Ilmatieteen laitoksen kehittämään Web-sovellukseen. Ilmatieteen laitos on tutkimus- ja palvelulaitos, jonka tehtävä tuottaa elinkeinoelämän ja yleisen turvallisuuden kannalta tärkeitä sää-, meri- ja ilmastopalveluita (Ilmatieteen laitos 2015a). Ilmatieteen laitoksessa työskentelee noin 720 henkilöä, joista noin puolet työskentelee meteorologeina ja muut erilaisissa asiantuntijatehtävissä tutkijoina tai esimerkiksi sääennusteiden tekemiseen tarvittavien järjestelmien ja laitteiden asiantuntijoina (Ilmatieteen laitos 2015b).

Ilmatieteen laitoksella on meneillään yhteistyöprojekti Nepalin kansallisen sääpalvelun kanssa. Kyseinen yhteistyöprojekti ei ole ensimmäinen organisaatioiden yhteishanke (Ilmatieteen laitos 2013). Aikaisemmissa yhteistyöhankkeissa on toteutettu kQC1-niminen ohjelmisto, jolla voi hallita Nepalin sääasemien havaintojentarkistusjärjestelmää. kQC1 on toteutettu PHP 5- ja JavaScript-ohjelmointikielillä. Ohjelmiston seuraava kehitysvaihe on toteuttaa manuaalisesti tehtävät laaduntarkistustoiminnot, joilla voi hakea, selata, kuvantaa ja muokata säähavaintodataa. Varsinkin erilaiset datan kuvantamismenetelmät helpottavat säähavaintojen oikeellisuuden arvioimista ja tätä kautta parantavat sääennusteiden ja -tuotteiden luotettavuutta. Opinnäytetyön puitteissa suunniteltiin ja toteutettiin osa näistä manuaalisista laaduntarkistus toiminnoista kQC1-ohjelmistoon. Projektista rajattiin ulos ohjelmiston käyttöönotto.

Opinnäytetyössäni pystyin esittelemään Haaga-Helia ammattikorkeakoulun tietojenkäsittelyn koulutusohjelman koulutuksen aikana keräämääni tietoa ja ammattitaitoa sekä niiden soveltamista käytännössä. Projektissa pääsi käyttämään monia ohjelmistokehittäjäopintopolun taipaleelta opittuja taitoja.

1.1 Lyhenteet

FNEP2 = Finnish-Nepalese Project 2. Ilmatieteen laitoksen ja Nepalin kansallisen sääpalvelun yhteistyöprojekti.

FMI = Finnish Meteorological Institute, Ilmatieteen Laitos.

DHM = Nepal Department of Hydrology and Meteorologyn, Nepalin kansallinen sääpalvelu.

HQC = Human Quality Control. QC1 tai QC2 vaiheen jälkeen toteutettava manuaalinen tarkistus.

QC = Quality Control, havaintojentarkistusjärjestelmä.

QC0 = Quality control phase 0. Havaintojentarkistuksen ensimmäinen vaihe QC-moduulissa.

QC1 = Quality control phase 1. Havaintojentarkistuksen toinen vaihe QC-moduulissa.

QC2 = Quality control phase 2. Havaintojentarkistuksen kolmas vaihe QC-moduulissa.

1.2 Käsitteet

Aggregation = Kooste tiedoista. Tässä raportissa Aggregation-tuloksilla tarkoitetaan sää-havainnoista koostettua tietoa.

AJAX = Joukko ohjelmistokehityksen tekniikoita. AJAX-tekniikalla voi hakea selaimeen pieniä määriä dataa palvelimen kautta, eikä koko verkkosivua tarvitsee ladata uudelleen.

Attribuutti = Muuttuja, johon ohjelmaolio tallentaa tietoa.

Back-end = Ohjelmistokehityksen termi, jolla tarkoitetaan ohjelmiston datan käsittelyker-rosta.

Front-end = Ohjelmistokehityksen termi, jolla tarkoitetaan ohjelmiston käyttäjälle näkyvää kerrosta.

JavaScript = Web-ohjelmoinnissa käytettävä dynaaminen komentosarjakieli.

jQuery = Avoimen lähdekoodin JavaScript-kirjasto.

Muuttuja = Englanniksi variable. Yksikkö, johon ohjelma voi varastoida tietoa.

MySQL= Laajasti käytössä oleva relaatiotietokantaohjelmisto.

Ohjelmistoarkkitehtuuri = Ohjelmistorakenteen avainkohdat määrittelevä kuvaus ohjelmiston luokista ja olioista ja niiden keskinäisistä suhteista.

Parametri = Ohjelmoinnissa parametri tarkoittaa ohjelmalle tai funktiolle annettava alkuarvoa. Englanniksi käännettynä sana on parameter. Parametri tarkoittaa säähavaintodatassa havainnon tyyppiä, esimerkiksi: lämpötila (tempature, T), tuulennopeus (wind speed, WS) jne. Tässä raportissa sanaa käytetään molemmissa merkityksissä.

PostgreSQL = Avoimen lähdekoodin olio-relaatiotietokantaohjelmisto.

Rautalankamalli = Englanniksi tunnetaan nimillä wireframe, page schematic ja screen blueprint. Rautalankamalli on visuaalinen opas, johon suunnitellaan sovelluksen ulkoasun rakennetta.

Quality Flag = Laatulippu. Laatulippuun merkitään säähavainnon luotettavuus käyttäen ennalta sovittua numeroasteikkoa.

Mocup = Mocup on piirretty mallinnus toteutettavasta ohjelmasta. Tarkempi kuin rautalankamalli.

Istunto = Englanniksi session. Istunto luo pysyvän yhteyden selaimen ja palvelimen välille, kunnes se tulee elinkaarensa päähän.

Metodi = Itsenäinen ohjelman osa, jota voidaan kutsua pääohjelmasta.

2 Manuaalisen säähavaintotarkistuksen taustat

Säähavaintodataa tarkistetaan manuaalisesti ja automaattisesti. kQC1-ohjelmisto käyttää pohjoismaissa yleisesti käytössä olevaa Quality Control -järjestelmää. Ohjelmistoon on toteutettu automaattisen tarkistuksen toiminnot, joiden perusteiden ymmärtäminen on tärkeää manuaalisen tarkistuksen kehittäjille.

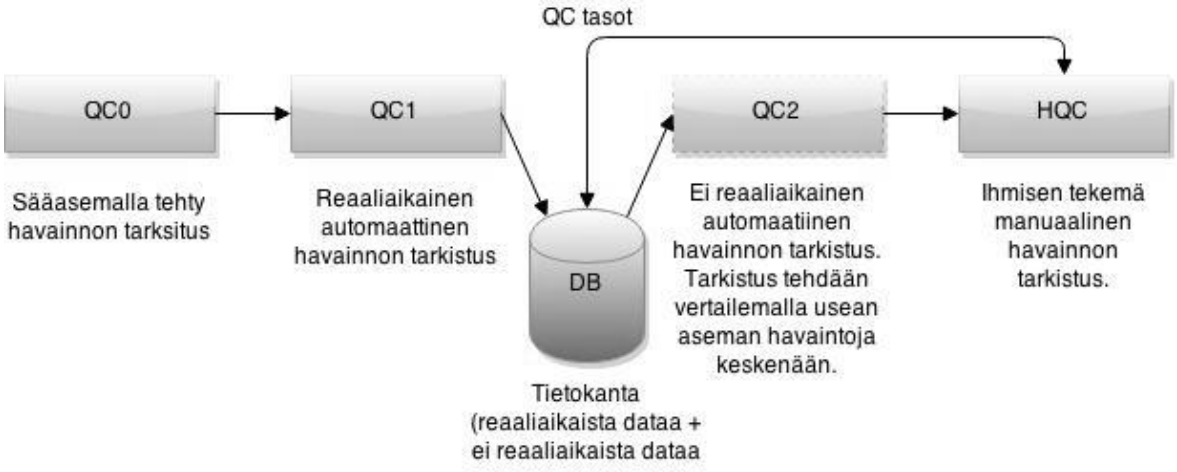
2.1 Quality Control

2.1.1 Esittely

Säähavaintoasemien automatisointi on yleistynyt viime vuosien aikana. Havaintoaikojen lyhentyminen ja datan siirtonopeuksien kasvaminen ovat asettaneet havaintodatan laadunvalvonnalle uusia haasteita. Pohjoismaiden sääpalveluiden laitokset ovat yhteistyössä kehittäneet reaaliaikaista säähavaintojen laaduntarkistusta vastamaan nykyisiin havaintojen keruun tarpeisiin. Heidän viimeaikaisten yhteistyöprojektiensa kehityskohteena on ollut Quality Control (QC) -järjestelmä. Esimerkiksi pohjoismaiden meteorologisien laitosten yhteistyöprojektissa NORDKLIM tehtiin paljon kehitystyötä QC -järjestelmän kehittämiseksi. Projektissa olivat mukana: SMHI (Sveriges meteorologiska och hydrologiska institut), DMI (Danmarks Meteorologiske Institut), FMI, VI (Iceland) ja met.no (Norway Meteorologisk Institutt) (Andresen 2002, 1–4.)

Järjestelmän tavoitteena on tuottaa mahdollisimman nopeaa ja tehokasta laadun valvontaa säähavaintodatalle. Virheellisten tai epäilyttävien havaintojen aikainen havaitseminen ja korjaaminen takaavat mahdollisimman luotettavan säähavaintodatan loppukäyttäjälle. QC-säähavaintotarkistuksen kehittämisen päätavoitteena on tehdä laaduntarkistuksesta tehokkaampaa ja mahdollisimman reaaliaikaista. Näihin tavoitteisiin pääseminen tarkoittaa kalibrointi-, mittaus- ja viestintävirheiden löytämisen mahdollisimman aikaisin datan elinkaaresta. Järjestelmän kehityksessä pyritään keskittymään automaattisen laaduntarkistuksen algoritmien kehittämiseen ja kehittämään virheellisten tai epäilyttävien datojen merkintäjärjestelmää. (Vejen 2002, 11.)

Quality Control on nelitasoinen säähavaintojen tarkistusjärjestelmä. Tasot ovat QC0, QC1, QC2 ja HQC (Kuvio 1). Jokaisella tasolla säähavaintodatalle tehdään ennalta määritelty laaduntarkistus. Jokaisen tarkistuksen jälkeen datalle annetaan Quality Flag -merkintä. Kutsun tässä dokumentissa tätä merkintää laatulipuksi. Mitä isomman laatulippuarvon QC-tarkistustaso antaa, sitä epäilyttävämpi datan laatu on. (Niemelä 6.)



Kuvio 1. Havaintotarkistuksen tasot (Vejen 2002, 1).

2.1.2 QC-tasot

Reaaliaikaisia havainnon tarkistustasoja on kolme: QC0, QC1 ja QC2. QC0-tason tarkistus tehdään säähavaintoasemalla ennen kun se lähetetään sieltä eteenpäin. Manuaalisella säähavaintoasemalla tämän tarkistuksen tekee ihminen. Ihmisen tekemään tarkistukseen voi lipsahtaa huolimattomuusvirheitä, kuten esimerkiksi kirjoitusvirhe. Automaattisella havaintoasemalla virheellisiä havaintoja voi päästä järjestelmään viallisen laitteen tai mittarin vuoksi. Säänhavaintosensorit eivät myöskään aina pysty kattamaan kaikkia mahdollisia säätiloja. (Niemelä, 5.)

Niemelän (5) mukaan manuaalisilta ja automaattisilta havaintoasemilta on siis mahdollista lähettää virheellistä dataa vaikka sille suoritettaisiin QC0-tason tarkistus. QC1-tasolla datalle tehdään uusi tarkistus, kun se saapuu havaintoasemalta tietokantaan. Tässä vaiheessa datan elinkaarta säähavainto on pisteessä, josta se halutaan mahdollisimman nopeasti jatkotuotteille ja käyttäjille käytettäväksi. Tarkistuksen nopeus on siis avainasemassa. Koska nopeus ja tarkkuus eivät kaikissa tilanteissa kulje käsi kädessä, tämä tarkistus ei ole aina kaikkein perusteellisin. Tästä syntyy taas riski, että virheellistä dataa siirtyy eteenpäin järjestelmässä.

Q2-tason tarkistus on huomattavasti perusteellisempi kuin aikaisemmat tasot. Se yleensä valmistuu järjestelmiin kohtalaisen ajan kuluttua havaintoajasta (esim. 20 minuuttia myöhemmin). Q2-tasolla voidaan vertailla havaintoa toisiin samalta alueelta saapuviin havaintoihin tai numeraalisiin analyyseihin ja näin laskea havainnon laadukkuus. Tämä tarkistustaso on vielä kehitysvaiheessa, eikä sitä käytetä tässä projektissa. (Niemelä. 4)

Vaikka säähavaintodata on kulkenut useamman tarkistuksen läpi ennen kuin se tallennetaan tietokantaan, on virheellisen datan mahdollisuus silti olemassa. Näitä tilanteita varten QC-laaduntarkistukseen kuuluu vielä viimeinen vaihe, manuaalinen säähavaintotarkistus eli HQC (Human Quality Control). Tällä tasolla ihminen voi tarkistaa epäilyttävän datan oikeellisuuden ja korjata virheitä, jotka ovat päässeet vuotamaan automaattisesta tarkistuksesta. Jokaisella tarkistustasolla datalle annetaan myös laatulippu, joka kertoo siitä kuinka luotettavaksi kukin tarkistustaso on säähavainnon arvioinut.

2.2 Laatulippu – Quality flag

Pohjoismaissa käytettävissä järjestelmissä laatulipun arvo voi olla 0–9 (Niemelä, 6). Tässä projektissa käytetyssä asteikossa arvo voi olla 0,1, 2, 5, 6, 8 tai 9 (taulukko 1).

Taulukko 1. Laatulippujen arvot.

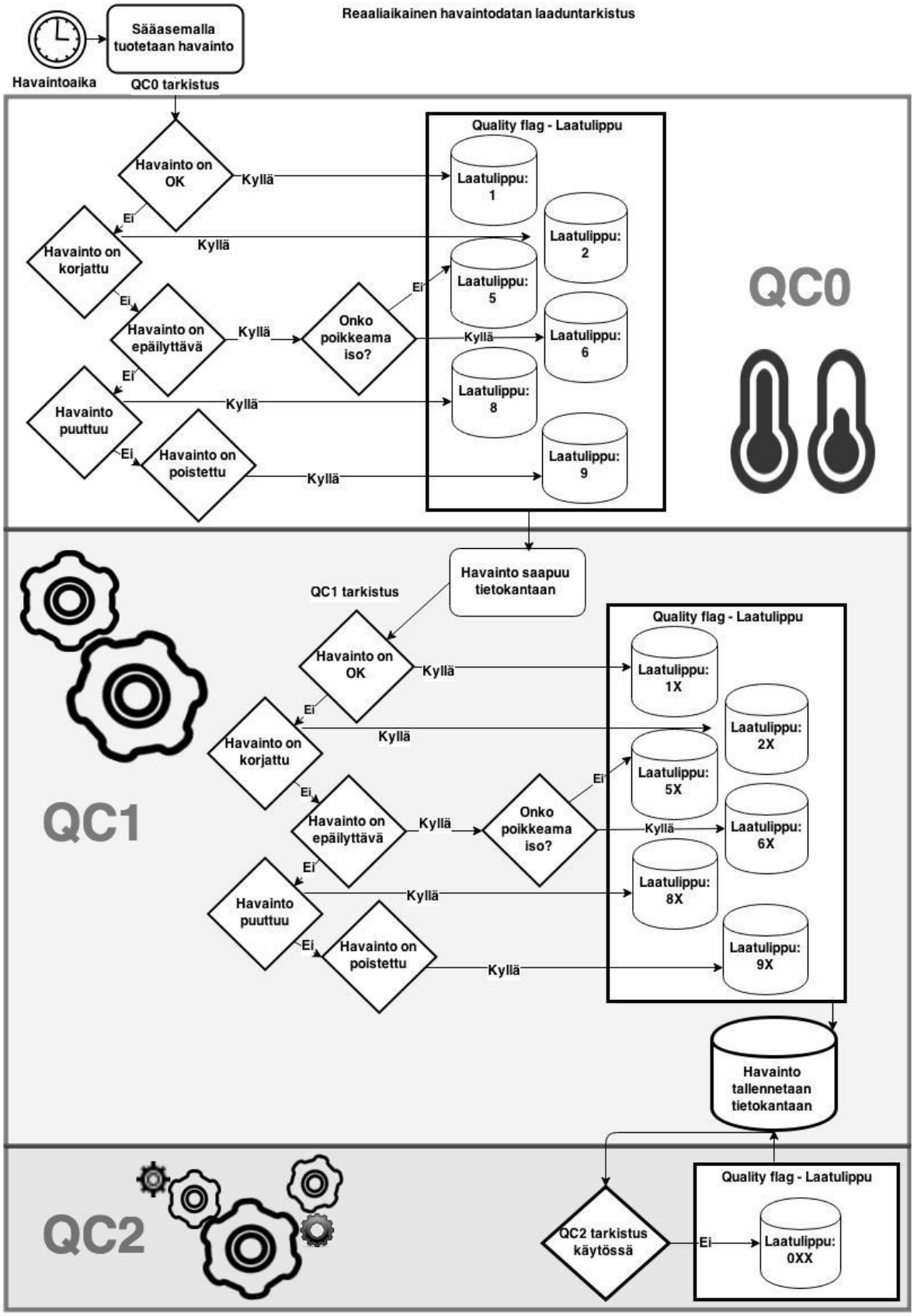
Tarkistuksen tulos tai tila	Numero
Tarkistusta ei ole tehty	0
OK	1
Korjattu	2
Numero ei ole käytössä	3
Numero ei ole käytössä	4
Poikkeavuus	5
Iso poikkeavuus	6
Numero ei ole käytössä	7
Puuttuva	8
Poistettu	9

Laatulippu saa uuden arvon jokaisen tarkistustason jälkeen. Laatulippu voi olla pituudeltaan maksimissaan nelinumeroinen luku. Luku kasvaa jokaisella tasolla. QC0-tason tarkistuksen tulos merkitään laatulippuun ykkösten kohdalle, QC1-tarkistuksen tulos kymmenysten kohdalle, QC2-tulos sadansien kohdalle ja HQC-tarkistus tuhansien kohdalle. (Niemelä, 6).

Esimerkiksi Nepalilainen säähavaintoasema Silgadhi Doti mittasi lämpötilahavainnon 8.10.2013 klo 06:00 UTC, jonka lopulliseksi laatulipuksi tuli 81. Asemalla tehtävä tarkistus eli QC0-tason tarkistus ei huomannut havainnossa mitään ongelmaa, joten laatulipun arvoksi tuli tältä tarkistustasolta 1.

Kun havainto saapui tietokantaan, sille tehtiin QC1-tason tarkistus. Lämpötilahavainnoille on QC-järjestelmässä tarkistus, johon on määritelty lämpötilahavaintotyypille minimi- ja maksimiarvot. Tässä vaiheessa järjestelmä huomasi, että aseman lähettämä lämpötilahavainto, jonka arvo oli 314, ei voi olla paikkaansa pitävä. Järjestelmä merkitsi laatulipun arvoksi 81 (QC1=8, QC0=1).

Laatulipun alemman tason tarkistuksen arvoja ei voi ikinä poistaa tai muokata jälkikäteen. Koska tarkistustaso QC2 ei ole käytössä, järjestelmä merkkaa laatulipun arvoksi 081 (kuvio 2). Jos tämän tason tarkistus olisi käytössä, se voisi tehdä vertailun havainnolle esimerkiksi tutkimalla lähiasemien lämpötilahavaintoja. Jos QC2-tason testit osoittaisivat, että 314 celsiusasteen lämpötila olisi epäilyttävä, muttei automaattisesti virheellinen voisi laatulipun arvoksi tulla 681 (QC2=6, QC1=8, QC0=1).
.



Kuvio 2. Reaaliaikainen havaintodatan laaduntarkistus.

Järjestelmässä ei tuolloin ollut HQC-toimintoja. Jos esimerkkihavainnolle olisi tässä vaiheessa tehty manuaalinen tarkistus eli HQC-tasontarkistus, olisi ihminen voinut todeta havainnon olleen virheellinen ja merkitä havainnon poistetuksi järjestelmästä. QC-järjestelmästä ei voi poistaa virheellisiä havaintoja kokonaan, vaan laatulipun arvoksi merkitään 9. Tämän jälkeen sitä ei enää käytetä ja sen näkyvyys eri käyttäjille rajataan. Esimerkki havainnon laatulipun arvosta olisi siis tullut tässä vaiheessa 9081 (taulukko 2).

Taulukko 2. Reaaliaikainen havaintodatan laaduntarkistus.

HQC tarkistuksen tulos tai tila	QC2 tarkistuksen tulos tai tila	QC1 tarkistuksen tulos tai tila	QC0 tarkistuksen tulos tai tila
9	0	8	1

2.3 Human Quality Control

HQC-tarkistustaso on siis säähavaintotarkistuksen viimeinen taso, jossa ihminen voi vertailemalla säädataa selvittää onko kantaan saapunut havainto virheellinen. Mikko Partion (2014) mukaan: ”HQC osuuden perimmäinen ajatus on: Mahdollisimman helppo ja nopea tapa visualisoida ja muokata havaintoja sekä qc-tuloksia."

Säähavainnon oikeellisuutta voi arvioida yhtä havaintoa tarkastelemalla, mutta useimmissa tapauksissa virhe on helpompi huomata, kun havaintoa voi vertailla toisiin havaintoihin. Datan kuvantaminen diagrammiin voi helpottaa oleellisen datan hahmottamista. Tyypillisimpiä graafisia diagrammeja ovat pylväs- ja viivadiagrammit. (KvantiMOTV 2004.)

Riippuen siitä millaista tilastoaineistoa halutaan selata, voi taulukkodiagrammi olla työläämpi tulkita kuin paikkatietoihin perustuva kuvantaminen. Suuriosa maapallolla kerätystä tilastoaineistosta pystytään kohdistamaan tiettyihin kohtiin maapallolla eli ne ovat paikkatietoa. (Blomqvist 2004, 6.)

Säähavaintodatan kuvantaminen erilaisiin diagrammeihin ja paikkatietoihin perustuvaan karttanäkymään helpottavat reaaliaikaisen tarkistuksen tulosten selaamista ja niiden laadun arvioimista. Kun virheellisiä havaintoja havaitaan, pitää niitä myös pystyä sovelluksen kautta korjaamaan. Muokkausnäkymä ja sen käytettävyys pitää siis huomioida sovellusta suunniteltaessa.

3 Toteutuksessa käytetyt teknologiat ja tekniikat

3.1 JavaScript ja kirjastot

HQC-toimintojen toteuttamiseen käytettiin JavaScript-ohjelmointikieltä. JavaScript on front-end-ohjelmointikieli. Tämä tarkoittaa sitä, että se suoritetaan selaimessa eikä palvelimella. Tämä asia on tärkeä huomioida ohjelmistoa suunniteltaessa. (Wright 2012, 25.)

Melkein aina koodin suorittaminen on nopeampaa palvelimella eli back-end-puolella. Jos jonkin osan ohjelmasta voi siis ajaa palvelimella, on se yleisesti ottaen parempi tehdä siellä. Ohjelmistokehittäjä ei voi myöskään olla täysin varma, miten ohjelman front-end-puoli käyttäytyy käyttäjän selaimessa. Jos back-end-koodi toimii halutulla tavalla kehittäjällä, se toimii oikein joka paikassa. (Wright 2012, 25.) Asiakkaan päässä selaimesta on voitu esimerkiksi valita asetus, joka estää JavaScriptin suorittamisen tai selain voi olla vanhentunut. Tällaisista syistä johtuen kaikki sovelluksen front-end-ominaisuudet eivät ole välttämättä tuettuja.

Front-end-puolen laitteiston ja ohjelmiston erilaisista skenaarioista johtuen olisi hyvä pyrkiä käyttämään websovelluksia suunniteltaessa ”Progressive Enhancement” tai ”Graceful Degradation” -strategioita. Näistä strategioista kerrotaan tarkemmin myöhemmin tässä raportissa.

JavaScriptiä käytetään siis esimerkiksi visualisten efektien luomiseen websivulle tai HTML-koodin manipuloimiseen. JavaScriptin kirjoittamisesta voi tehdä helpompaa käyttämällä JavaScript-kirjastoja (Wright 2012, 212.)

HQC-toteutuksen suunnittelussa päädyttiin toteuttamaan karttanäkymä, joka toteutettiin JavaScript-kirjastoa käyttäen. Kahdeksi parhaaksi ehdokkaaksi valikoituivat Open Layers 3 ja Google Maps JavaScript API 3.

Google Maps JavaScript API 3 valittiin parhaaksi vaihtoehdoksi lopputuloksen toteuttamiseen. Se sisälsi kaikki tarvittavat ominaisuudet suunnitellun sovelluksen toteuttamiseen ja se oli hyvin dokumentoitu. Suunnitelmissa on myös joskus toteuttaa sovellukseen diagrammeja piirtävä toiminto. Google developers tarjoaa tähänkin ohjelmistorajapinnan nimeltään Google Visualization API Reference (Google developers 2015d). Tuntui siis järkevältä lähteä käyttämään Google developers -tuotteita, koska saman perheen tuotteilla pystyy toteuttamaan kaikki tähän sovellukseen suunnitellut toiminnot.

JavaScriptillä toteutettiin myös jotain ulkoasun muokkauksia, jotka parantavat ohjelman käytettävyyttä ja validointitoimintoja.

3.1.1 Google Maps Javascript API v3

Google Maps JavaScript API v3 ei ole itsessään JavaScript-kirjasto, vaan ohjelmistorajapinta, jonka avulla voidaan kutsua ohjelmaan toimintoja ja funktioita Googlen palvelusta. Se on ilmainen käyttää, jos kartta ladataan palvelusta alle 25 000 kertaa päivässä 90 päivän aikana (Google developers. 2015a). Jos latauskertoja tulee enemmän, täytyy rajapinnan käyttöön hankkia maksullinen lisenssi. Käyttörajojen uskotaan riittävän hyvin ilmaiseen lisenssiin tämän sovelluksen osalta.

Google Maps JavaScript API v3 mahdollistaa karttojen piirtämisen sivulle ja niiden ulkoasun muokkaamisen. Se sisältää monenlaisia piirto-ominaisuuksia, joista tässä työssä eniten käytettiin marker -elementtien muokkaamista. API:n tarjoamien ominaisuuksien lisäksi käytettiin avoimen lähdekoodin lisäkirjastoa markerwithlabel.js. Tämä mahdollisti kerrosten lisäämisen marker -elementteihin.

Näillä työkaluilla marker-elementtejä pystyi muotoilemaan tarpeeksi monipuolisesti ja niiden dokumentaatio oli selkeä ja hyvin jäsennelty. Google developers -sivusto tarjoaa paljon vinkkejä ja esimerkkejä Google Maps JavaScript API v3:n käyttöön, joten sen tuominen ohjelmaan ja hyödyntäminen kävi vaivattomasti.

3.1.2 jQuery

jQuery on yksi maailman suosituimmista JavaScript-kirjastoista koko maailmassa. Se on avoimen lähdekoodin kirjasto, jonka alkuperäinen kehittäjä John Resig julkaisi vuonna 2006. (Wright 2012, 218.) kQC1-ohjelmistokin käyttää kyseisestä kirjastosta versiota jQuery JavaScript Library v1.6.2. Muita ohjelmistossa käytössä olevia jQuery-laajennuskirjastoja ovat: UI 1.8.17, jquery-ui-timepicker-addon.js, jquery.ui.core.js, jquery.ui.datepicker.js ja jquery.ui.widget.js (jQuery UI 1.8.16) -kirjastot.

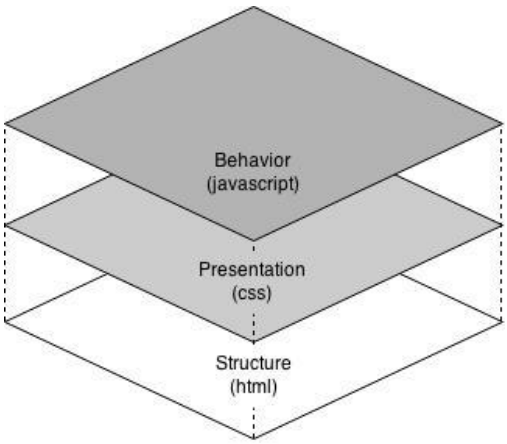
jQuery on hyvin monipuolinen kirjasto. Kaikista sen ominaisuuksista ehkä paras ja tässä projektissa käytetyin ominaisuus, oli AJAX request -pyyntöjen kirjoittaminen.

3.1.3 Progressive enhancement -strategia

Progressive enhancement -strategian perusidea on Tim Wrightin (2012, 3) mukaan taata sivuston sisällön saatavuus. Erilaisilla selaimilla ja asetuksilla pystytään vaikuttamaan sivustojen ulkoasun latautumiseen. Sisällön saatavuuden turvaaminen onnistuu pitämällä sivuston front-end-ratkaisut kolmessa eri tasossa: Structure (rakenne), presentation (esittely) ja behavior (käyttäytyminen).

Structure-kerros pitää sisällään HyperText Markup Language (HTML) -koodin sivustolla. Oikein ja standardin mukaisesti kirjoitettu HTML-koodi käyttäytyy hyvin luotettavasti monenlaisissa selaimissa. Jos sivuston sisältö esitetään tässä kerroksessa, pystytään takaamaan siihen pääsy melkein kaikissa olosuhteissa. (Wright 2012, 3–4.)

Presentation-tasolla toimivat Cascading Style Sheets (CSS) -tiedostot ja ylimmällä Behavior- kerroksella JavaScript (kuvio 3). Pitämällä nämä kolme omilla tasoillaan pystytään aina kuorimaan päällimmäinen kerros pois ilman, että sivuston sisällön saatavuus vaarantuisi. Front-end-puolta suunniteltaessa ja toteuttaessa olisi siis hyvä pyrkiä pitämään sivusto toimivana vaikka käytössä olisi vain HTML, tehdä sivustosta paremman näköinen kun käytössä on CSS ja parantaa sivuston käytettävyyttä, kun käytettävissä on myös JavaScript. Näitä periaatteita toteuttaen sivuston saavutettavuus paranee ja koodin uudelleen käytettävyys helpottuu (Wright 2012, 4–5.)



Kuvio 3. Progressive enhancement (Wright 2012, 4).

3.2 AJAX

AJAX ei ole lyhenne mistään termeistä, toisin kuin monissa lähteissä väitettään. AJAX-teknologian pääkonsepti on mahdollistaa HTML-osien päivittäminen selaimessa lataamatta koko sivua uudelleen. (Wright 2012, 150.)

AJAX voi kommunikoida palvelimen kanssa kahdella tavalla: synkronoiden tai asynkroiden. Paljon yleisempi tapa käyttää AJAX-kutsuja on asynkronoitu tapa, mikä käytännössä tarkoittaa, että ohjelmassa suoritettaan yksi AJAX request -pyyntö kerrallaan. AJAX voi käyttää request-pyyntöjen datalähteenä XML, JSON, HTML tai teksti -formaateissa olevaa dataa. (Wright 2012, 150–151.)

3.3 JSON

JSON-lyhenne tulee sanoista JavaScript Object Notation. Se on syntaksi datan säilyttämiseen ja kuljettamiseen eri rajapintojen välillä. Toinen samankaltainen ja samaan tarkoitukseen käytettävä teknologia on XML. AJAX-teknologiaa käyttävissä ohjelmissa JSON on paljon helppokäyttöisempi ja nopeampi vaihtoehto kuin XML. (w3school. 1–2.)

3.4 PHP

PHP on suosittu ohjelmointikieli, joka sopii hyvin web-ohjelmointiin. PHP-lyhenne tuli alun perin sanoista Personal Home Pages, mutta on myöhemmin vaihdettu tarkoittamaan Hypertext Processor termiä (Beighley 2009, 15). PHP on back-end ohjelmointikieli, eli se suoritetaan aina palvelimella.

3.4.1 SQL Injection

SQL Injection on tietoturvahyökkäys, jossa yritetään kalastaa tai tuhota tietoa tietokannasta SQL-kyselyiden avulla (PHP 2015). Yksi hyvä keino estää SQL Injection -hyökkäyksiä on kirjoittaa parametroituja SQL-kyselyitä (w3school. SQL Injection).

3.5 PostgreSQL

PostgreSQL on oliorelaatiotietokantahallintasysteemi (Object-Relational Database Management System, ORDBMS). PostgreSQL on avoimen lähdekoodin ohjelmisto, ja se tukee suurta osaa SQL-standardista. (The PostgreSQL Global Development Group 2010, 52.)

Transaktion hallinta on perustavanlaatuinen konsepti tietokannoissa. Transaktio tarkoittaa käytännössä sitä, että tietokantakyselyt rakennettaan asteittain, jotta niillä voidaan joko tehdä suoritus kokonaan tai ei ollenkaan. Tällä menetelmällä voidaan turvata datan oikeellisuutta ja estetään päällekkäiset muutokset dataan. (The PostgreSQL Global Development Group 2010, 76.)

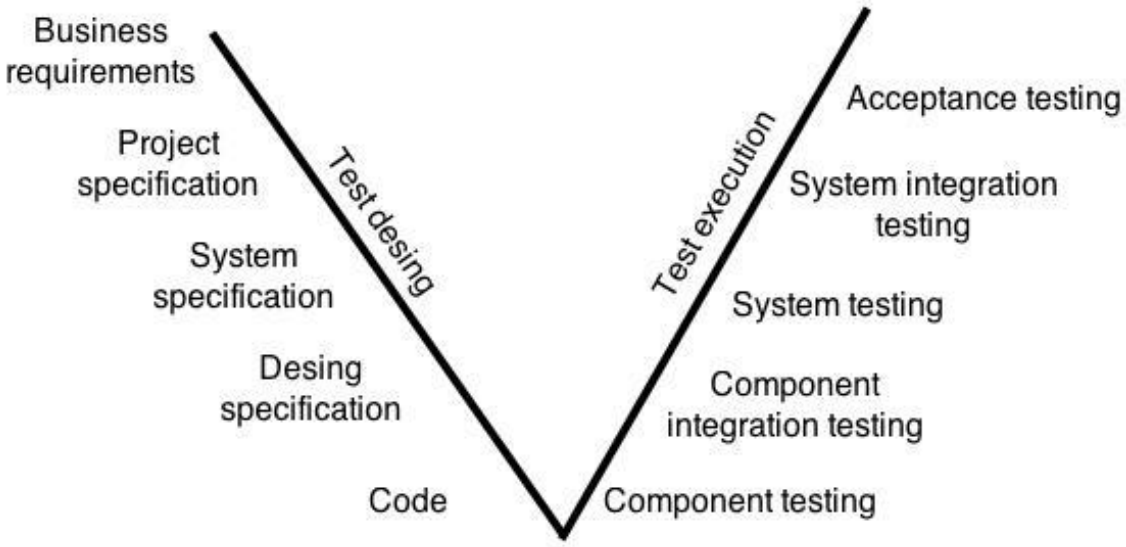
Pelkkä tietokannasta haku ei vaadi suurempaa transaktion suunnittelua. Kun tietoa halutaan muokata, poistaa tai tallentaa tietokantaan, on tärkeä miettiä miten datan oikeellisuus turvataan transaktiossa.

3.6 Testaus

Ohjelmistokehityksessä testaamisella varmistetaan, että tuotetaan ohjelmisto joka toimii oikein ja on sellainen kun tilaaja on halunnut. Testausta voi olla staattista tai dynaamista testausta. (Haaga-Helia 2011.)

Raine Kauppinen (Haaga-Helia 2011) on listannut luentomateriaalissaan ”Testaus ja Softalaprojekti 1” syitä miksi testausta tehdään ohjelmistokehityksessä. Testaamisella voidaan osoittaa, että ohjelmisto toimii oikein eri käyttötilanteissa ja että ohjelmisto täyttää sille asetetut vaatimukset ja määritykset. Testaamisella voi löytää ohjelmiston virhetilanteet ja korjata niiden taustalla olevat viat. Testaaminen voi myös alentaa ohjelmiston kehittämisen kustannuksia ja se on helppo tapa arvioida ohjelmiston laatua.

Testaamisessa käytetään usein V-mallia (kuvio 4). Mallissa esiintyvien vaiheiden lisäksi testauksen laadunvarmistuksessa voidaan käyttää viittä eri testaustyyppiä: toiminnallinen testaus (mustalaatikkotestaus), ei-toiminnallinen testaus, rakenteellinen testaus, uudelleen testaus ja regressiotestaus. Testauksen suunnitteluun kuuluu suunnitella millaista testausta tehdään, missä vaiheessa ohjelmistokehitystä testausta suoritetaan ja testitapausten suunnittelu. (Haaga-Helia 2011.)



Kuvio 4. Testauksen V-malli (Haaga-Helia 2011).

Testitapausten suunnittelussa määritellään millä perusteella testit katsotaan hyväksytyksi tai hylätyksi ja milloin testaus suoritetaan. Testitapauksia on kahdentasoisia. Korkean tason testitapaukset ovat otsikkotason kuvauksia siitä mitä testataan. Konkreettinen testitapaus on tarkempi kuvaus, jossa on askelittain kuvattu testitapauksen suunniteltu eteneminen. Jokaisella askeleella on kerrottu askeleen osalta konkreettiset syötteet ja odotetut tulokset. Testitapausten pitää edustaa järjestelmän käyttötilanteita, joissa sovellusta käytetään niin kuin sitä normaalitilanteissa käytettäisiin ja käyttötilanteita, jossa sovellusta käytetään poikkeuksellisesti. Poikkeuksellinen käyttö voi olla esimerkiksi tilanne, jossa käyttäjä syöttää ohjelmaan vahingossa virheellistä tietoa. (Haaga-Helia 2011.)

Ei-toiminnallinen testaus keskittyy testaamaan sovelluksen ominaisuuksia, jotka eivät liity suoranaisesti siihen mitä sovelluksella voi tehdä. Ei-toiminnallinen testaus voi liittyä suorituskyvyn tai tietoturvan testaamiseen. Toiminnallinen testaus tarkoittaa testausta, jolla testataan sovelluksen toimintoja ja käyttäjälle näkyvää osaa. Rakenteellinen testaus perustuu ohjelmiston toteutuksen rakenteen, kuten esimerkiksi lähdekoodin testaamiseen. (Haaga-Helia 2011.)

Selenium on avoimen lähdekoodin ohjelma, jolla voi tehdä automaattisia toiminnallisia testejä. Selenium toimii useimmissa selaimissa ja käyttöjärjestelmissä. Siitä on tarjolla kaksi versiota, Selenium WebDriver ja Selenium IDE, joista IDE-ohjelmistolla voi nauhoittaa testejä. (Selenium.)

4 Sovelluksen toteutus

Työ toteutettiin jo valmiina olevaan kQC1-ohjelmistoon mahdollistamaan manuaalisesti tehtävät säähavaintojen laaduntarkistustoiminnot.

4.1 kQC1

kQC1-ohjelmiston kehitystyö alkoi alun perin vuonna 2006. Silloinen kehitystyö oli EU:n rahoittamassa projektissa Liettuan Meteorologisen Instituutin ja Ilmatieteen laitoksen kanssa. Sittemmin ohjelmiston kehitystä on jatkettu FNEP- ja FNEP2-projekteissa. (Partio, 1–2.)

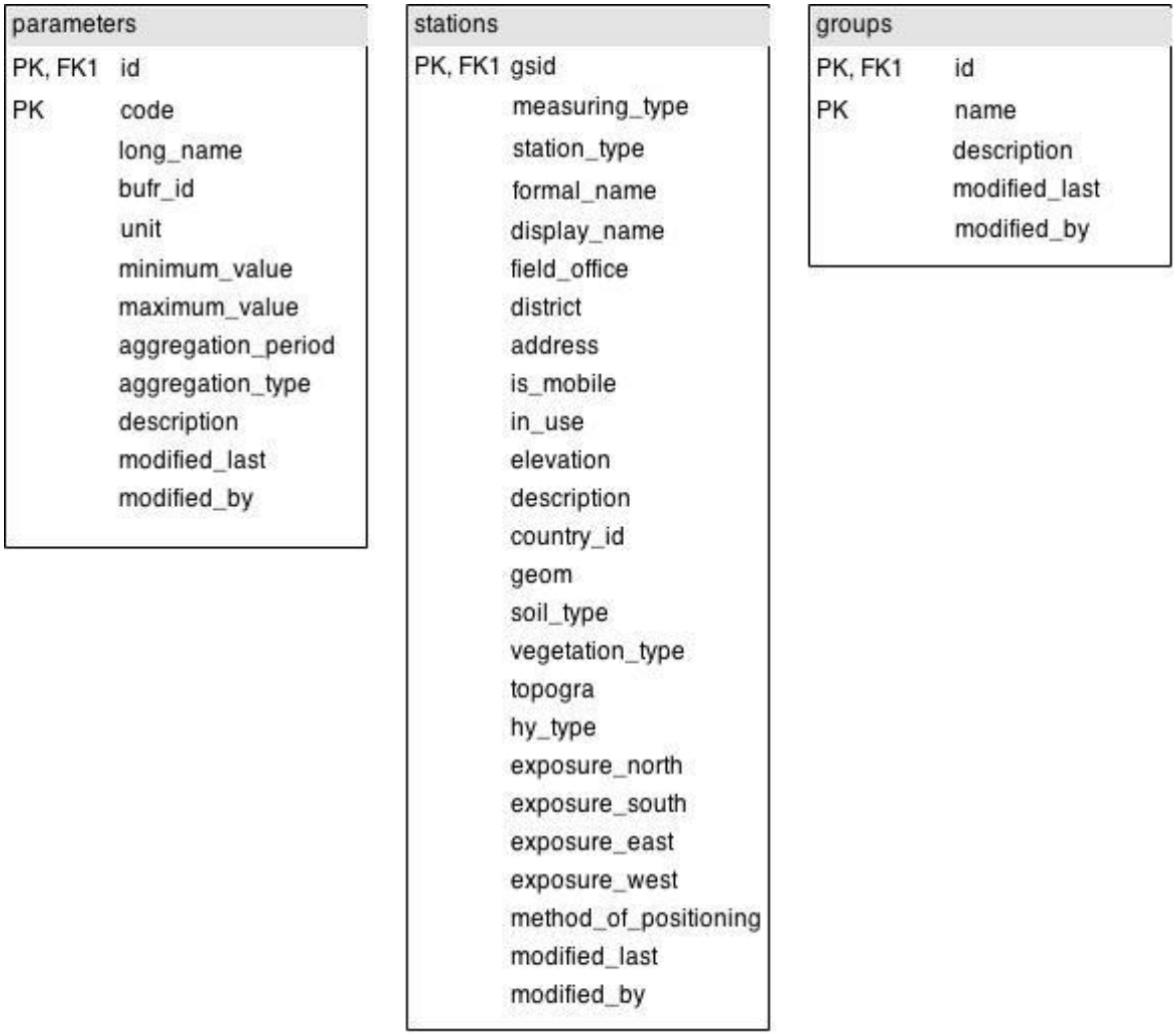
4.1.1 Tekniikat

kQC1 on kirjoitettu PHP 5.3-ohjelmointikielellä ja se käyttää PostgreSQL-tietokantaa. Ohjelmistolle suositeltu PostgreSQL-versio on 9.0, mutta myös jotkin vanhemmat versiot ovat yhteensopivia ja PostGIS pitäisi olla versio 1.5 tai uudempi. Apache web-palvelin on paras ratkaisu QC-ohjelmalle. kQC1 on kehitetty ja testattu Linux-ympäristössä, mutta sen pitäisi toimia muissakin käyttöjärjestelmissä. (Partio, 3–4.)

4.1.2 Tietokannan rakenne

kQC1-ohjelmistolla on kohtalaisen monimutkainen tietokanta, koska osa QC-tarkistuksista suoritetaan tietokannan ominaisuuksia hyödyntäen. Kuvaan tässä raportissa tietokantaa niiltä osilta, mitä HQC-toiminnot käyttävät.

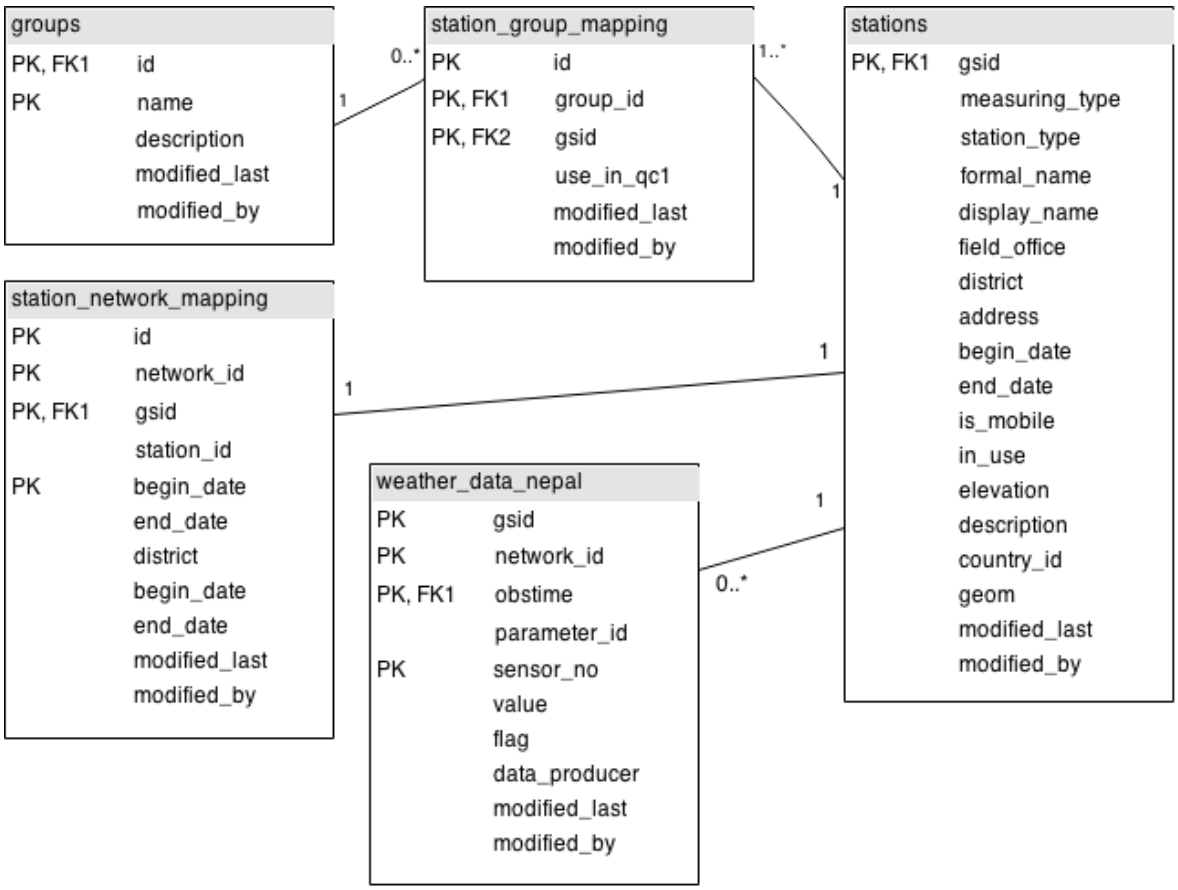
Sovellukseen suunniteltiin hakulomake, joka hakee AJAX-kutsuilla tietokannasta lomakkeelle hakuehtoja. Lomakkeelle haetaan tietoa kolmesta eri taulusta (kuvio 5), kun sivu ladataan.



Kuvio 5. Hakulomakkeen käyttämät taulut.

Hakulomakkeen käyttämät SQL-kyselyt ovat melko yksinkertaisia. Hieman monimutkaisempiin kyselyihin päästään, kun hakuehdot on määritelty ja tietokannasta tulisi löytää halutut havainnot. Nämä kyselyt hakevat tietoa viidestä eri tietokantataulusta (kuvio 6). Kyselyissä käytettiin JOIN komentoa ja kyselyt kirjoitettiin parametroituina SQL-kyselyitä.

Opin hakuja kirjoittaessani, että joissakin tapauksissa oli nopeampaa tehdä tietokantahausta yksinkertaisempi ja parsia tietoa ohjelman sisällä, kuin suorittaa monimutkainen ja tarkempi haku kantaan ja saada sieltä suoraan haluttu tieto.



Kuvio 6. Tietokannan kuvaus HQC-taulujen osalta.

4.1.3 Sovellusarkkitehtuuri

kQC1-ohjelmisto ei noudata orjallisesti mitään tunnettua ohjelmistoarkkitehtuurityyliä, mikä teki ohjelmaan tutustumisen melko vaivalloiseksi. HQC-osion toteutuksessa pyrittiin noudattamaan samanlaisia käytäntöjä millä muukin ohjelmisto on toteutettu.

Ohjelmisto noudattaa hyvin pitkälle Progressive Enhancement -strategiaa, mikä helpotti front-end-osion työstämistä. Jokaisen tason tiedostoille on sovelluksessa oma hakemisto ja ne ovat nimetty kuvaavasti.

Nimeämiskäytäntöinä esimerkiksi kirjattiin ylös:

- Välilyönti kirjoitetaan tiedoston nimiin, luokkiin, attribuutteihin ja objekteihin alaviivamerkillä (”_”).
- Lyhenteitä ei käytetä missään. Tämä koskee myös attribuutteja, elementtejä ja parametreja.

4.1.4 Loppukäyttäjä

Sovelluksen loppukäyttäjä on meteorologi tai joku säähavaintojen kanssa työskentelevä henkilö, esimerkiksi teknikko (Partio 2015). Vaikka sovelluksella tulee olemaan paljon käyttäjiä, on käyttäjätyyppi ja heidän käyttämänsä työympäristöt hyvin tiedossa, eikä kaikkiin eri skenaarioihin tarvitse valmistautua samalla tavalla, kuin julkiseen käyttöön menevässä sovelluksessa.

Toimintoja suunniteltaessa ja toteutettaessa voidaan esimerkiksi olettaa, että käyttäjillä on päivitetty selain käytössä ja JavaScriptin käyttö on sallittu sen asetuksissa. Vaikka ohjelmaa kirjoittaessa on pyritty toteuttamaan Progressive Enhancement -strategiaa, niin se ei täysin toteudu. Esimerkiksi osa toiminnoista ei toimi ilman JavaScriptiä.

4.2 Kehitysympäristön pystyttäminen

Kehitysympäristö pystytettiin Dell Latitude 7240 -tietokoneelle, jossa on Windows 7 64 bit käyttöjärjestelmä. Ohjelmistoympäristöksi asennettiin Eclipse Luna (Eclipse for PHP Developers) ja siihen ilmaisen lisäosa Subversive – SVN Team Provider 2.0.1, jonka pystyi lataamaan Eclipse Marketplace -sovelluskaupasta.

Koska kQC1-ohjelmiston pyörittämiseen tarvittiin Apache web-palvelin, asennettiin kehitysympäristöön avoimen lähdekoodin XAMPP-ohjelma. Valitsin XAMPP-ohjelman WAMP-ohjelman sijaan, koska se oli minulle entuudestaan tutumpi.

kQC1-projektin tietokannan asetukset piti käydä vaihtamassa vastaamaan testitietokannan tietoja ja osa ohjelmiston asetuspoluista piti kirjoittaa uusiksi vastaamaan kehitysympäristöä.

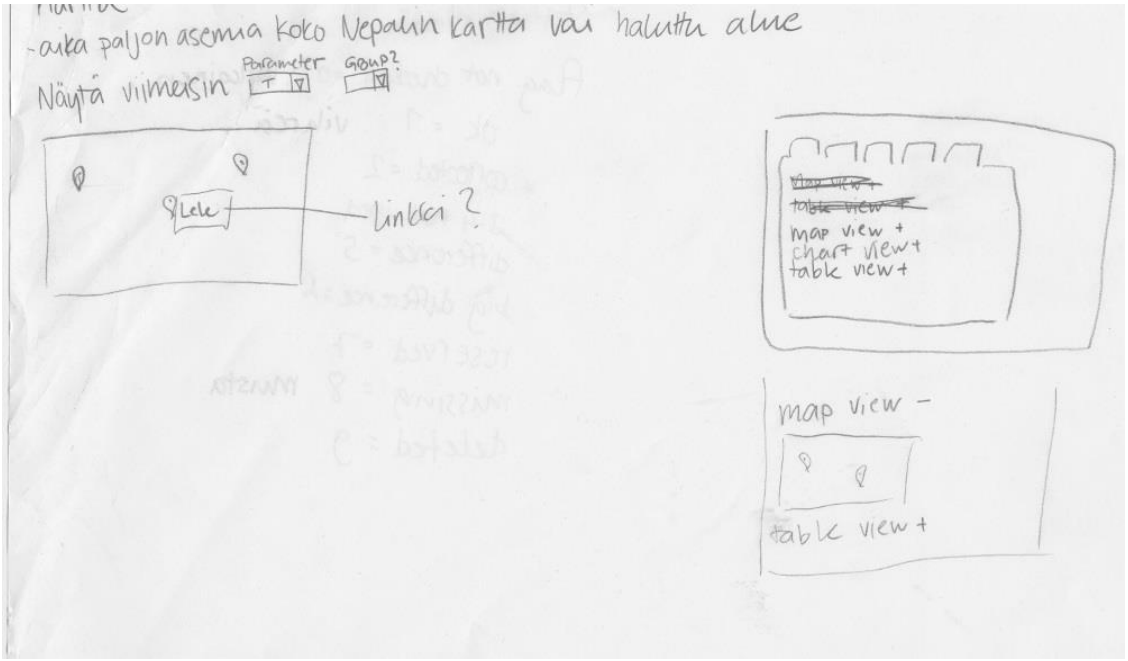
Kehitysympäristön pystyttäminen vei enemmän aikaa kuin sen toteuttamiseen oli suunniteltu. Pystyttämisestä kirjoitettiin ohjeet, jotta tulevaisuudessa ympäristö saataisiin nopeammin toimintakuntoon.

4.3 Suunnittelu ja määrittely

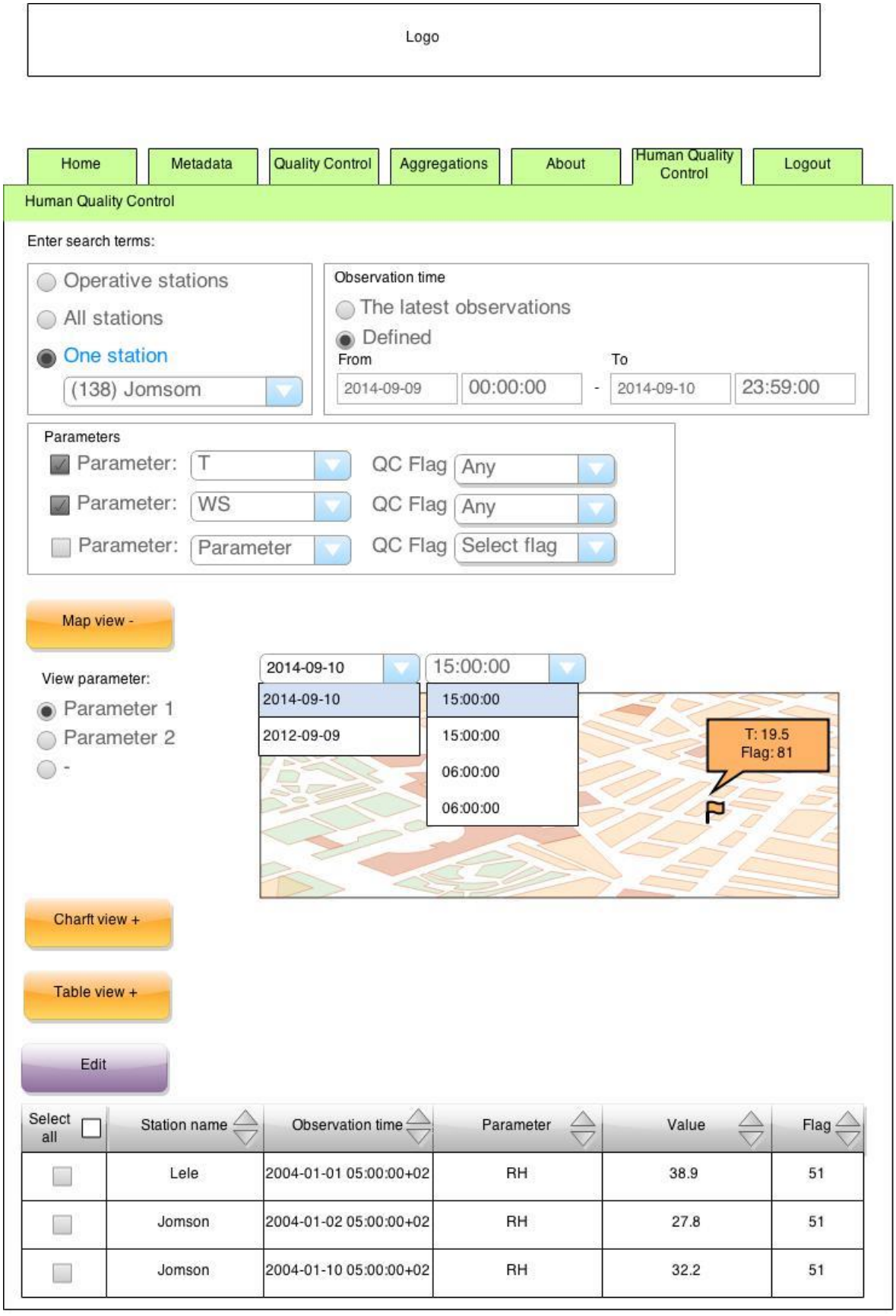
Kun sain ensimmäisen viestin Ilmatieteen laitokselta opinnäytetyöaihe-ehdotukseksi, en heti ymmärtänyt mitä viestissä minulle ehdotettiin. Aihe piti sisällään paljon minulle uutta säähavaintoihin liittyvää sanastoa ja erilaisia lyhenteitä. Työn alkaessa minun pitikin alkuun käyttää paljon aikaa säähavaintojen laaduntarkistukseen liittyvään dokumentaatioon

tutustumiseen ja loppukäyttäjien tarpeiden määrittelemiseen. Onneksi tähän sain tukea kQC1-ohjelmiston asiantuntijoilta.

Kun QC tai HQC alkoivat tulla tutuiksi termeiksi ja loppukäyttäjän tarpeet oli kartoitettu, aloitettiin HQC-sivun suunnittelu. Työ lähti käyntiin piirtämällä käyttöliittymästä rautalankamalleja. Nämä mallit selvensivät kehitettävän sovelluksen kokonaiskuvaa ohjelmiston kehittäjälle (kuva 1) ja tilaajalle. Kun kehittäjän mielestä malleihin oli saatu suunniteltua toimivimmat ratkaisut, pyydettiin niihin vielä kommentteja ohjelmiston asiantuntijalta. Viimeisin versio rautalankamallista vastasi hyvin pitkälle tulevaa toteutusta, joten sitä voisi jo melkein kutsua mocup-kuvaksi (kuva 2). Kuva piirrettiin Draw.io -ilmaisohjelmalla, jolla voi piirtää kuvioita ja kaavioita (Benson 2014).



Kuva 1. Rautalankamallin ensimmäinen versio.



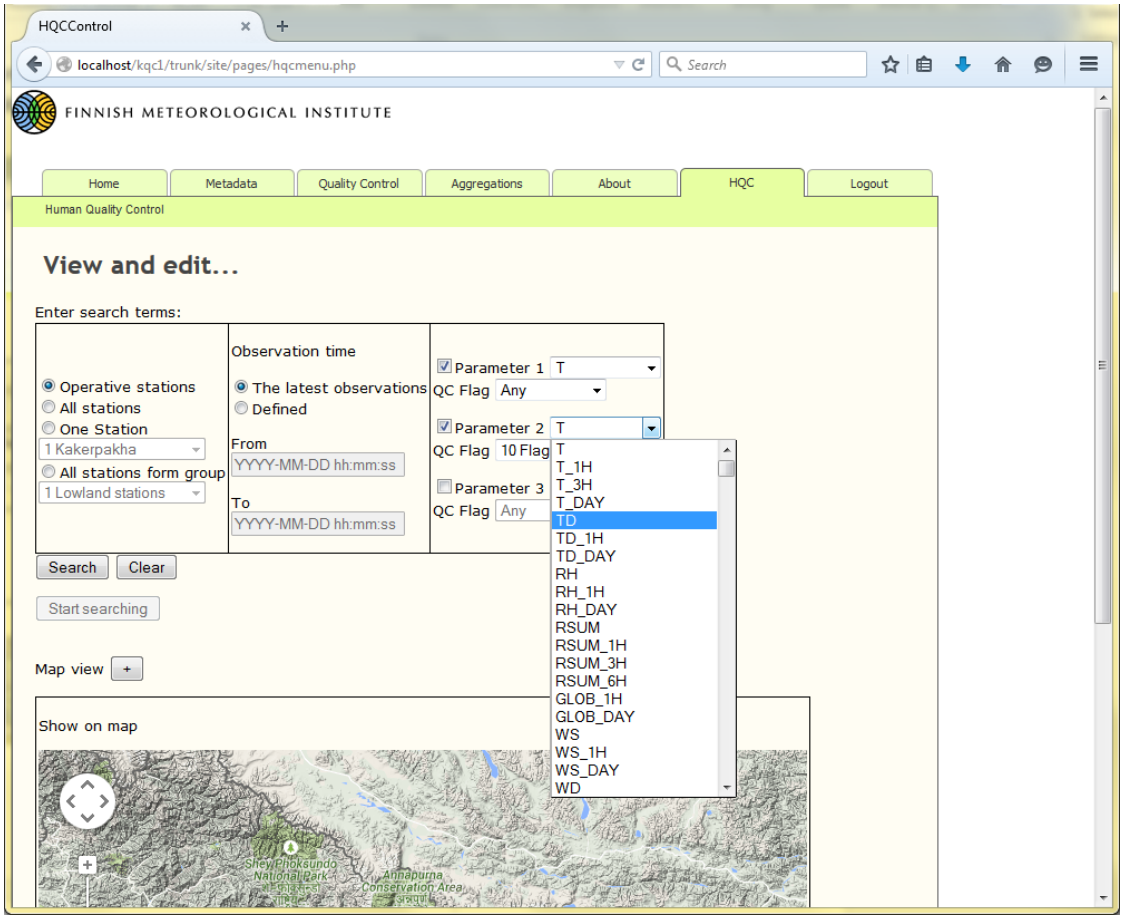
Kuva 2. Rautalankamalli käyttöliittymästä.

Jo projektin alkaessa oli tiedossa, että kQC1-ohjelmiston käyttäjiltä oli saatu palautetta, että ohjelmaa joutuu ”klikkailemaan liikaa”. Tulevalta kehitykseltä toivottiin siis tehokkaammin käytettävää käyttöliittymää. Tämän vuoksi HQC-sivulle sijoitettiin eri näkymät yhdelle sivulle, eikä useamman valikon taakse. Näkymiä voisi piilottaa +/- -painikkeilla käyttäjän tarpeiden mukaan.

Sivulle suunniteltiin kolme erilaista näkymää: taulukko-, kartta- ja diagramminäkymä. Lisäksi sivulla on hakulomake, johon käyttäjä voi syöttää haluamiensa asemien ja sääparametrien hakuehdot.

4.4 Hakulomake

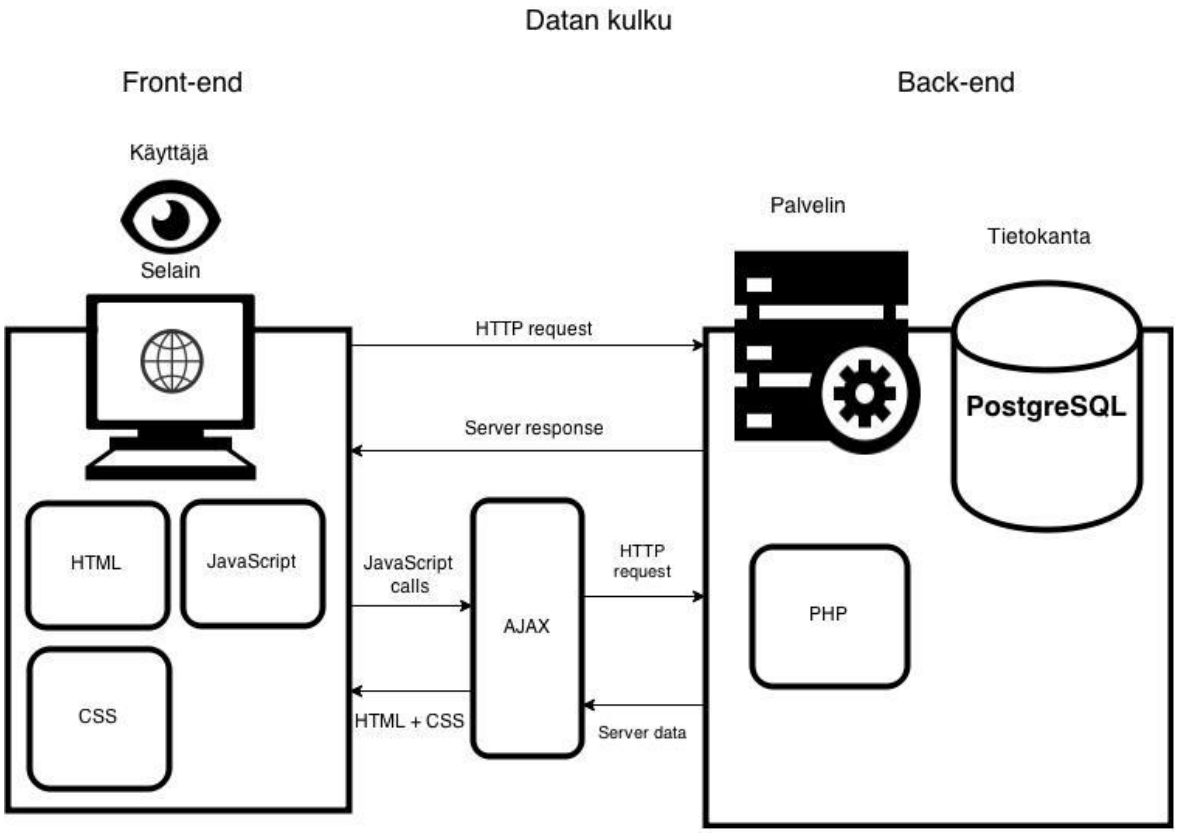
Sivulle toteutettiin HTML-lomake, johon voi syöttää erilaisia hakuehtoja. Pudotusvalikkoihin haetaan tiedot AJAX-kutsuilla. Haku käynnistyy, kun sivu ladataan. Lomakkeelle on myös toteutettu JavaScript-ohjelmointikieltä käyttäen käytettävyyttä parantavia ominaisuuksia. Esimerkiksi viimeisin havaintoaika -valinta estää tietojen lisäämisen määritellyn ajan -kenttiin (kuva 3). Lomakkeen voi tyhjentää Clear-painikkeella ja haun aloittaa Search-painikkeella. Search-painike tallentaa validit hakuehdot selaimen istuntoon.



Kuva 3. Kuvakaappaus hakulomakkeesta. Tietojen syöttäminen.

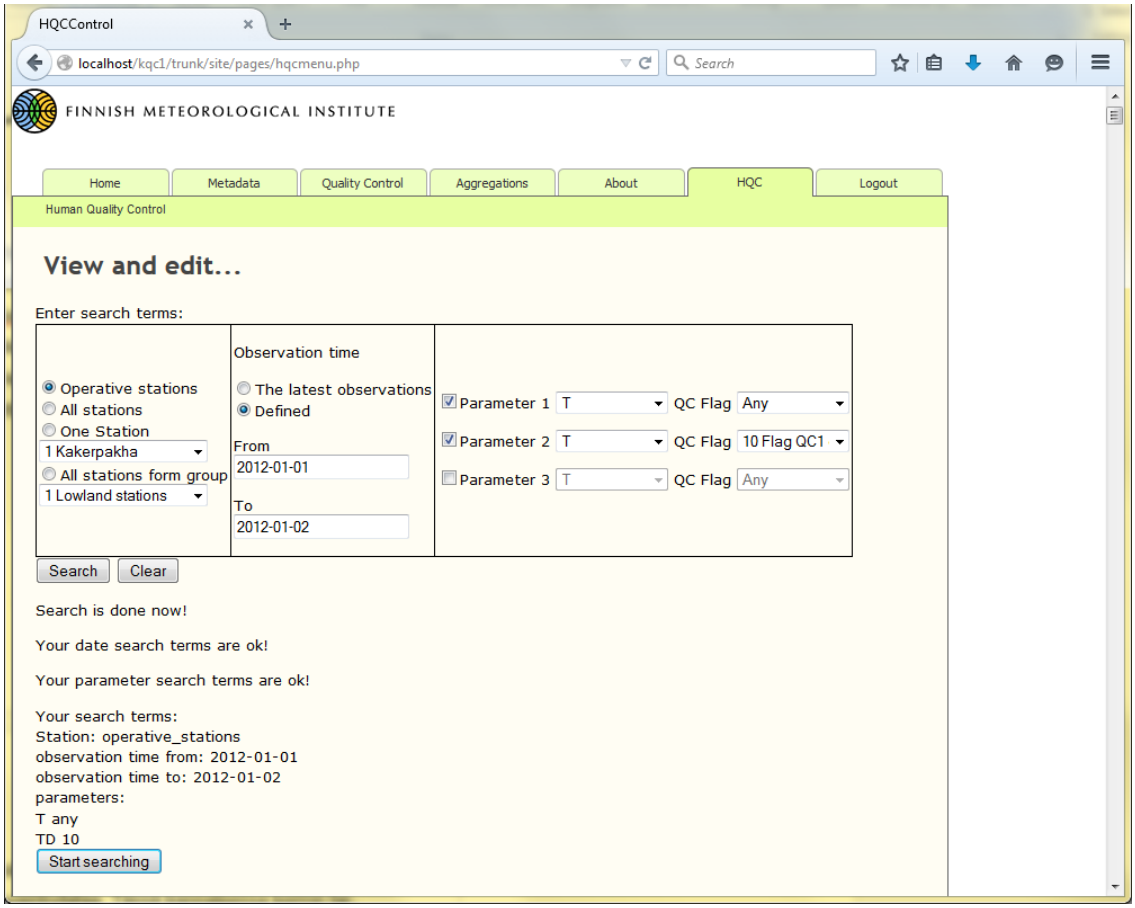
Lomakkeella on mahdollista hakea kolmea eri sääparametria. Ohjelma on kuitenkin kirjoitettu niin, että vain yhtä toistorakennetta kasvattamalla tai pienentämällä voi kehittäjä lisätä tai poistaa parametrien lukumäärää hakulomakkeesta.

Tämä on ehkä koko toteutuksesta lempikohtani. Kirjoittamani ratkaisu ei vain lisää hakukenttään HTML form -elementtejä, vaan muutokset kulkevat koko datavirran läpi (kuvio 7).



Kuvio 7. Datan kulku AJAX (Thivent 2009).

Lomakkeen Search-painike lähettää syötetyt tiedot PHP-luokkaan, jossa ne validoidaan ja tallennettaan selaimen istuntoon. Jos syötetyt tiedot ovat oikeassa muodossa ja hakuehtoja on annettu tarpeeksi, vapautuu Start Searching painike disable-tilasta ja käyttäjä voi siitä klikkaamalla aloittaa haun. Lomakkeelle haetaan istunnosta käyttäjän syöttämät hakuehdot, joten jos esimerkiksi päivämäärä kenttään on syötetty virheellinen päivämäärä, ei käyttäjän tarvitse aloittaa lomakkeen täyttämistä alusta asti (kuva 4).



Kuva 4. Kuvakaappaus hakulomakkeesta.

4.5 Taulukkonäkymän toteutus

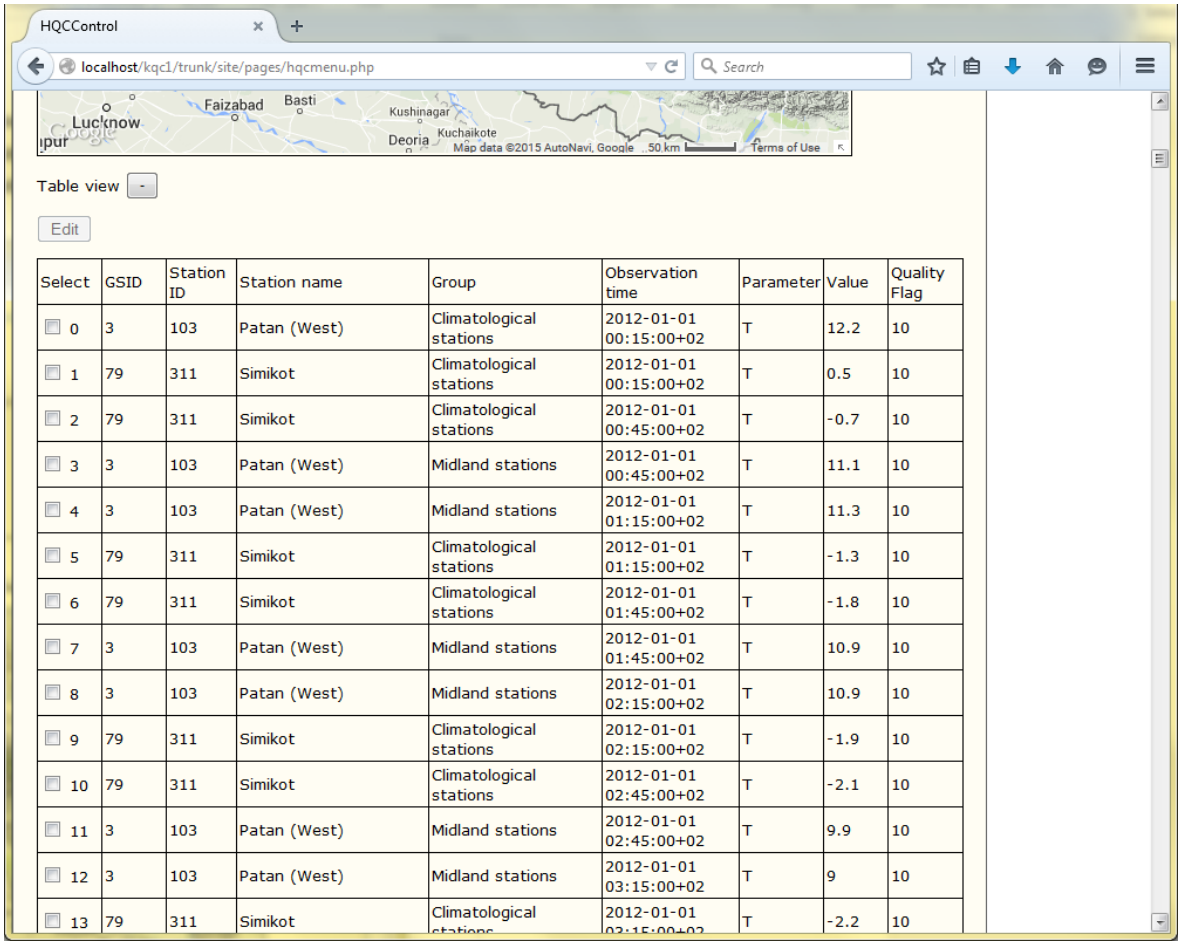
Taulukkonäkymä suunniteltiin pitämään sisällään kahdenlaisia HQC-toimintoja. Sillä pitäisi pystyä selaamaan ja muokkaamaan säähavaintodataa. Tässä kappaleessa kerrotaan tarkemmin selaamistoimintojen toteutuksesta.

Kun käyttäjä on syöttänyt validit hakuehdot ja klikannut hakulomakkeen Start searching -painiketta, kutsutaan JavaScript-funktiota. JavaScript-funktio hakee hakulomakkeelta sääparametrien hakuehdot ja tallentaa ne JSON-muodossa. Funktio lähettää JSON-datan AJAX-kutsulla. Kutsu saapuu ensin PHP-luokkaan, jolla kutsutaan toisia PHP-luokkia, joiden tehtävä on hoitaa tietokantayhteyden avaus, SQL-kyselyiden lähettäminen ja kyselyiden vastausten vastaanottaminen.

Kun PHP luokka saa JSON-datan, se parsii sen monitasoiseksi PHP-taulukoksi ja hakee istunnosta loput tarvittavat hakutiedot. PHP-luokassa vertaillaan hakuehtoja sinne kirjoitettuihin SQL-kyselyihin ja sopivan vaihtoehdon löydyttyä lähetetään kysely hakuehtoineen tietokannalle.

Tietokanta lähettää vastauksen takaisin PHP-luokkaan, jossa vastauksena saadut relaatiorivit käydään läpi ja data tallennetaan JSON-muotoon. JSON-data palautetaan AJAX-kutsun lähettäneelle funktiolle.

Tässä vaiheessa on siis palattu takaisin JavaScript-funktioon, jota prosessin alussa kutsuttiin Start searching -painikkeen klikkaamisella. Funktio kirjoittaa HTML-sivulle taulukkoon vastaanottamansa tiedot jQuery-kirjaston append()-metodia käyttäen (kuva 5).



Select	GSID	Station ID	Station name	Group	Observation time	Parameter	Value	Quality Flag
0	3	103	Patan (West)	Climatological stations	2012-01-01 00:15:00+02	T	12.2	10
1	79	311	Simikot	Climatological stations	2012-01-01 00:15:00+02	T	0.5	10
2	79	311	Simikot	Climatological stations	2012-01-01 00:45:00+02	T	-0.7	10
3	3	103	Patan (West)	Midland stations	2012-01-01 00:45:00+02	T	11.1	10
4	3	103	Patan (West)	Midland stations	2012-01-01 01:15:00+02	T	11.3	10
5	79	311	Simikot	Climatological stations	2012-01-01 01:15:00+02	T	-1.3	10
6	79	311	Simikot	Climatological stations	2012-01-01 01:45:00+02	T	-1.8	10
7	3	103	Patan (West)	Midland stations	2012-01-01 01:45:00+02	T	10.9	10
8	3	103	Patan (West)	Midland stations	2012-01-01 02:15:00+02	T	10.9	10
9	79	311	Simikot	Climatological stations	2012-01-01 02:15:00+02	T	-1.9	10
10	79	311	Simikot	Climatological stations	2012-01-01 02:45:00+02	T	-2.1	10
11	3	103	Patan (West)	Midland stations	2012-01-01 02:45:00+02	T	9.9	10
12	3	103	Patan (West)	Midland stations	2012-01-01 03:15:00+02	T	9	10
13	79	311	Simikot	Climatological	2012-01-01	T	-2.2	10

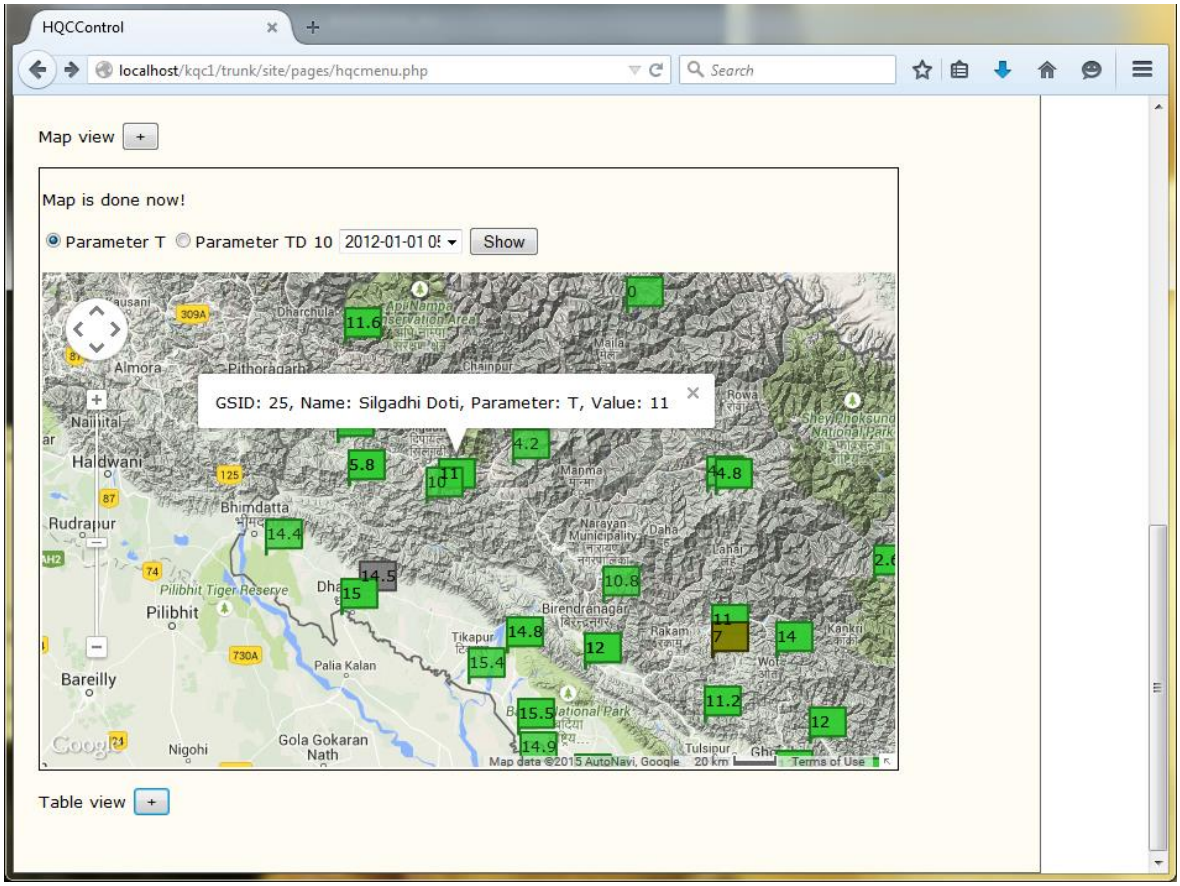
Kuva 5. Kuvakaappaus taulukkonäkymästä.

Taulukkonäkymän voi piilottaa klikkaamalla miinuspainiketta (kuva 5). Klikkaamalla jonkun säähavaintodatan rivin valintaruutua Edit-painike vapautuu disable-tilasta. Muokkaus toimintoa ei toteutettu tässä projektissa, joten Edit-painikkeella ei ole vielä mitään toiminnollisuutta.

4.6 Karttanäkymän toteutus

Karttanäkymän tehtävä on kuvantaa haettua säähavaintodataa ja tehdä näin havaintojen manuaalisesta tarkistamisesta helpompaa sovelluksen loppukäyttäjälle. Karttanäkymässä on esimerkiksi helppo vertailla lähiasemien arvoja keskenään tai arvioida säähavainnon oikeellisuutta aseman sijainnin perusteella.

Kuten rautalankamallissa näkyy (kuva 2) kartalla suunniteltiin näkymään hakutulosten arvoja lipun muotoisessa marker-elementissä. Kun elementtiä klikkaa, se antaa lisäinformaatiota asemasta. Kartalle piirrettävät liput ovat erivärisiä säähavainnon laatulipun arvosta riippuen (kuva 6).



Kuva 6. Kuvakaappaus sovelluksen karttanäkymästä.

Sovelluksessa kartalle piirtyvällä lipulla voi olla kuusi erilaista väriä: valkoinen, vihreä, sininen, ruskea, harmaa ja musta. Värit eivät kulje vihreästä punaiseen vaan vihreästä mustaan, koska nepalilaisessa kulttuurissa värit koetaan tarkoittavan eri asioita kuin länsimaalaisessa kulttuurissa (Bortoli 2001, 25). Lippujen värejä saatetaan vielä muuttaa myöhemmin käyttäjien palautteen perusteella.

Google maps API 3 marker -elementeille voi antaa minkä tahansa ikonin. Siihen voi käyttää kuvatiedostoa tai piirtää se SVG-tekniikalla. (Google developers 2015c.) Kartan flag iconit on toteutettu SVG-tekniikalla ja niiden piirtämiseen käytettiin Janvas-websovellusta. Janvas on monipuolinen ilmainen sovellus, jolla voi tehdä muutakin kuin vektorigrafiikkakuvia (Janvas 2015).

JavaScript-luokassa luodaan olio flag, jolle määritellään SVG-koordinaatit, väri (joka on tallennettu color nimiseen muuttujaan) ja viivan paksuus. Tässä on esimerkki sovelluksen käyttämästä JavaScript-oliosta:

```
var flag = {
path: transform='M-7.105427357601002e-15,-3.552713678800501e-15
L28.837893023051997,-3.552713678800501e-15
L28.94923623935704,23.063906921724293
L0.668059297830851,23.200920230169935
L0.6680592978311282,28.270412642668067 L-
0.11134321630542843,28.270412642668052 L-7.105427357601002e-
15,17.03532135010545 L-7.105427357601002e-15,-3.552713678800501e-15 Z ',
fillColor: color,
fillOpacity: 0.8,
scale: 1,
strokeColor: strokecolor,
strokeWeight: 2,
};
```

Tämän jälkeen ohjelma voi luoda marker-elementin, jonka ikonin arvoksi annetaan juuritehty flag-olio. Olioon annetaan myös location-, map- ja value-muuttujat, joihin on tallennettuna tieto paikkatiedosta, kartasta ja marker-elementiin kirjoitettavasta säähavainnon arvosta. Sovelluksessa marker-olio luodaan näin:

```
var marker = new MarkerWithLabel({
        position: location,
        icon: flag,
        map: map,
        labelContent: value,
        labelAnchor: location,
        labelStyle: {opacity: 0.75}
        });
```

Marker-olio tallennetaan taulukkoon samaan aikaan, kun toistorakenne käy hakemassa jokaisen valittuihin ehtoihin sopivan säähavainnon. Toistorakenteen loputta sovellus piirtää marker-elementit kartalle.

4.7 Jatkokehitysehdotukset

Opinnäytetyöprojektin aikana projektin rajausta jouduttiin tiukentamaan alkuperäisestä suunnitelmasta työn laajuuden ja ajanpuutteen vuoksi. Alkuperäisestä viidestä toiminnosta (haku, muokkaus, diagrammi-, kartta, ja taulukkonäkymä) toteutettiin kolme.

Minulla on tämän projektin aikana syntynyt paljon jatkokehitysehdotuksia. Ilmeisimmät ovat tästä projektista pois rajatut muokkaustoimintojen ja diagramminäkymän toteutus. Niiden suunnittelu käynnistettiin jo tämän projektin aikana.

HQC-haku toiminnot hakevat tällä hetkellä vain säähavainto dataa, mutta tulevaisuudessa se voisi myös hakea aggregaatti-tilastoja.

HQC-toiminnoille on kirjoitettu testaussuunnitelma, mutta itse testauksia ei ajanpuutteen vuoksi tässä projektissa suoritettu. kQC1-ohjelmistoa ei ole testattu mitenkään aikaisemmin, joten testaussuunnitelman voisi laajentaa koskemaan koko ohjelmistoa. Tietysti jos ja kun testauksissa löytyy bugeja sovelluksesta, niin virhetilanteet tulisi raportoida ja korjata niiden taustalla olevat syyt.

Ohjelmiston voisi beta-julkaista ja kerätä käyttäjiltä palautetta ja kehittämisehdotuksia sovelluksesta. Sovelluksen ulkoasua voisi hioa ja miettiä mitkä hakulomakkeen oletusarvot olisivat kaikkein hyödyllisimmät käyttäjälle. Painikkeiden toimintoja voisi tehostaa, jotta ”liika klikkaillu” vähenisi ja validointi ilmoitukset voisi tehdä näkyvimmiksi.

HQC-toimintoja kehittäessä pyrittiin noudattamaan Progressive enhancement -strategiaa, mutta koodin kerrostamista voisi viedä vielä pidemmälle.

5 Pohdinta

Olen ollut Ilmatieteen laitoksella töissä vuodesta 2010. Kun opiskeluni olivat siinä vaiheessa, että tarvitsin opinnäytetyön aiheen, kysyin sitä työnantajaltani. Sain Ilmatieteen laitokselta parikin aihe-ehdotusta ja niistä lähdin toteuttamaan kQC1-ohjelmiston kehittämistä. Projektin aloituskokous oli 16.12.2014. Opinnäytetyön aihe oli todella mielenkiintoinen, mutta se vaati aluksi paljon perehtymistä säähavaintojen laaduntarkistusjärjestelmään ja kQC1-ohjelmistoon. Ohjelmisto käytti myös jonkin verran minulle uusia tekniikoita (esimerkiksi AJAX ja JSON), joita en ollut käytännössä päässyt käyttämään aikaisemmin töissäni tai opiskeluiden aikana. Projektin alku sujui siis pitkälti uuden oppimisessa.

Työhön käytettävän ajan arviointi ja työnsuunnittelu tuottivat vaikeuksia heti projektin alussa. Projektisuunnitelmassakin oli mietitty ajankäytön olevan riski projektin onnistumiselle. Opinnäytetyön ohella minun piti suorittaa tradenomitutkinnosta jäljellä olevat kurssit ja päivätyöni, joten ajankäytön suunnittelussa oli haastetta.

Opinnäytetyön edetessä rajausta jouduttiin tiukentamaan alkuperäisestä suunnitelmasta. Lopulliseksi rajaukseksi päätettiin toteuttaa säähavaintojen hakutoiminto, kartta- ja taulukkonäkymät. Suunnittelu- ja määrittelyvaiheessa mietittiin jo diagrammi- ja muokkausnäkymän käytettävyyttä ja ulkoasua, mutta toteutus jätettiin pois projektista. Testauksen suoritus rajattiin myös opinnäytetyön ulkopuolelle. Opinnäytetyöprojektin aikana opin paljon ohjelmistokehitysprojektin suunnittelusta ja ajankäytöstä. Jos nyt pääsisin aloittamaan projektin alusta, osaisin suunnitella paljon tarkemman ja realistisemman aikataulun.

Vaikka ajankäytön kanssa oli hankaluuksia, työ sujui koko projektin ajan kohtalaisen hyvin ja etenemistä estäviä ongelmia ei esiintynyt. Alussa käytetty aika projektiin perehtymiseen kantoi hedelmää koko projektin ajan. Jo suunnitteluvaiheen tuloksista sain hyvää palautetta ohjelmiston periaatteiden ja käyttäjien tarpeiden ymmärtämisestä, vaikka säähavaintojen laaduntarkistus oli minulle uusi asia. Ohjelmiston suunnittelu ja dokumentointi luonnistuivat hyvin, koska ne olivat tuttuja asioita Softala-kursseilta.

Haaga-Helian tarjoama työnohjaus ohjauskokouksissa ja opinnäytetyöklinikalla olivat merkittävä tuki koko projektin ajan ja edistivät varsinkin tämän raportin kirjoittamista. Ilmatieteen laitokselta sain hyvän perehdytyksen kQC1-ohjelmiston rakenteisiin ja tarvittaessa tukea sovelluksen kehityksessä. Loppujen lopuksi sain toteuttaa projektin melko itsenäisesti ja olen erittäin tyytyväinen opinnäytetyöprojektin lopputuloksiin, vaikka alkuperäiseen suunnitelmaan jouduttiin tekemään rajauksia.

Lähteet

Andresen, L., Björnsson, H., Fredriksson, U., Iden, K., Jacobsson, C., Pálsdóttir, P., Rissanen, P., Samuli, A., Vejen, F. 2003. Manual Quality Control of Meterological Observations. Raportti. Luettavissa: http://www.smhi.se/hfa_coord/nordklim/task1/Report_HQC_redigert.pdf. Luettu: 3.3.2015.

Andrew, A. 2006. Chart Suggestions—A Thought Provoker. Kaavio. Luettavissa: http://img.labnol.org/di/choosing_a_good_chart2.pdf. Luettu: 15.5.2015.

Beighley, L & Morrison, M. 2009. Head First PHP & MySQL. O'Relly Media.

Benson, D. 2014. Käyttöopas. User Manual. Luettavissa: https://support.draw.io/display/DOD/Draw.io+Online+User+Manual. Luettu: 5.5.2015.

Blomqvist, I. 2004. Paikkatiedon tukimateriaali lukion maantieteen opettajille. Luettavissa: http://www03.edu.fi/oppimateriaalit/paikkatieto.pdf. Luettu: 15.5.2015.

Bortoli, M & Maroto, J. 2001. Colours Across Cultures: Translating Colours in Interactive Marketing Communications. Luettavissa: http://globalpropaganda.com/articles/TranslatingColours.pdf. Luettu: 15.5.2015.

Google developers. 2015a. Usage Limits and Billing. Luettavissa: https://developers.google.com/maps/documentation/javascript/usage#usage_limitshttps://developers.google.com/maps/. Luettu: 4.3.2015.

Google developers. 2015b. Symbols. Luettavissa: https://developers.google.com/maps/documentation/javascript/symbols#add_to_marker. Luettu: 4.3.2015.

Google developers 2015c. Custom Marker Symbols. Luettavissa: https://developers.google.com/maps/documentation/javascript/examples/marker-symbol-custom

Google developers. 2015d. Google Charts. Luettavissa: https://developers.google.com/chart/. Luettu: 4.3.2015

Haaga-Helia ammattikorkeakoulu. 2011. Tietojenkäsittelyn koulutusohjelma. Intranet. Testaus ja Softalaprojekti I. Luettu 13.5.2015.

Ilmatieteen laitos 2013. Tiedote 1204666. Luettavissa: http://ilmatieteenlaitos.fi/tiedote/1204666. Luettu: 9.1.2015.

Ilmatieteen laitos. 2015a. Ilmatieteen laitos. Luettavissa: http://ilmatieteenlaitos.fi/ilmatieteen-laitos. Luettu 29.1.2015.

Ilmatieteen laitos. 2015b. Henkilöstö ja työpaikat. Luettavissa: http://ilmatieteenlaitos.fi/henkilosto-ja-tyopaikat. Luettu 29.1.2015.

Janvas 2015. Kotisivut. Luettavissa: http://www.janvas.com/site/home_en.php. Luettu 17.5.2015.

KvantiMOTV. 2004. Graafinen esitys. Menetelmäopas. Luettavissa: http://www.fsd.uta.fi/menetelmaopetus/kuviot/kuviot.html. Luettu: 15.5.2015

Niemelä, K., Partio M. Quality Control, phase one. Käyttöopas. Ilmatieteen laitos.

Partio, M. kQC1 software user guide. Käyttöopas Ilmatieteen laitos.

Partio, M. 21.11.2014. Ryhmäpäälikkö. Ilmatieteen laitos. Sähköposti.

Partio, M. 29.1.2015. Ryhmäpäälikkö. Ilmatieteen laitos. Sähköposti.

PHP. 2015. SQL Injection. PHP Manual. Luettavissa: http://php.net/manual/en/security.database.sql-injection.php. Luettu: 20.3.2015.

Selenium. What is selenium?. Luettavissa: http://docs.seleniumhq.org/. Luettu: 15.5.2015.

The PostgreSQL Global Development Group. 2010. PostgreSQL 9.0.18 Documentation. Käsikirja. Luettavissa: http://www.postgresql.org/files/documentation/pdf/9.0/postgresql-9.0-A4.pdf. Luettu: 7.5.2015.

Thivent, P. 2009. How does AJAX work?. Foorumi keskustelu. Luettavissa: http://stackoverflow.com/questions/1510011/how-does-ajax-work. Luettu: 7.5.2015.

Vejen, F., Jacobsson, C., Fredriksson, U., Moe, M., Andresen, L., Hellsten, E., Rissanen, P., Palsdóttir, T., Arason, T. 2002. Quality Control of Meterological Observations Automatic Methods Used in the Nordic Countries. Raportti. Luettavissa: http://www.smhi.se/hfa_coord/nordklim/task1/quality_control.pdf. Luettu: 14.1.2015.

Wright, T. 2012. Learning JavaScript : a hands-on guide to the fundamentals of modern JavaScript. Addison-Wesley Professional.

w3school. JSON Tutorial. Luettavissa: http://www.w3schools.com/json/default.asp. Luettu: 25.4.2015.

w3school. SQL Injection. Luettavissa: http://www.w3schools.com/sql/sql_injection.asp. Luettu: 15.5.2015.

Liitteet

Liite 1: Luettelo työn kuvioista, kuvista ja taulukoista

Liite 2: kQC1-Testaussuunnitelma yhdellä testitapauksella

ILMATIETEEN LAITOS

kQC1 testaussuunnitelma

Sisällys

Muutoshistoria

Henkilö	Päiväys	Versio	Komentti
Paula Juntti	10.5.2014	0.01	Dokumentti luotu

Dokumentin tarkoitus

Tähän dokumenttiin on tarkoitus kuvata kQC1-ohjelmiston testaussuunnitelma. Testauksella todistetaan ohjelmiston toimintojen toimivan sovitulla tavalla.
Testitapausten pitää edustaa järjestelmän käyttötilanteita, joissa sovellusta käytetään niin kuin sitä normaalitilanteissa käytettäisiin ja käyttötilanteita, jossa sovellusta käytetään poikkeuksellisesti. Poikkeuksellinen käyttö voi olla esimerkiksi tilanne, jossa käyttäjä syöttää ohjelmaan vahingossa virheellistä tietoa.
Ei-toiminnallinen testaus keskittyy testaamaan sovelluksen ominaisuuksia, jotka eivät liity suoranaisesti siihen mitä sovelluksella voi tehdä. Ei-toiminnallinen testaus voi liittyä suorituskyvyn tai tietoturvan testaamiseen. Toiminnallinen testaus tarkoittaa testausta, jolla testataan sovelluksen toimintoja ja käyttäjälle näkyvää osaa. Rakenteellinen testaus perustuu ohjelmiston toteutuksen rakenteen, kuten esimerkiksi lähdekoodin testaamiseen.

Dokumentin sisältö

Testaussuunnitelma sisältää testien taustat, tavoitteet ja tulokset käyttötapauksittain. Dokumentissa käsitellään myös erikoistapaukset, rajaukset ja testausten dokumentointi.

Määritelmät ja termien selitykset

Termi	Kuvaus
FNEP2	Finnish-Nepalese Project 2. Ilmatieteen laitoksen ja Nepalin kansallisen sääpalvelun yhteistyöprojekti.
HQC	Human Quality Control. QC1 tai QC2 vaiheen jälkeen toteutettava manuaalinen tarkistus.
QC	Quality Control, havaintojentarkistusjärjestelmä.
QC0	Quality control phase 0. Havaintojentarkistuksen ensimmäinen vaihe QC –moduulissa.
QC1	Quality control phase 1. Havaintojentarkistuksen toinen vaihe QC –moduulissa.
QC2	QC2 = Quality control phase 2. Havaintojentarkistuksen kolmas vaihe QC –moduulissa.

Testauksen kohde ja tavoitteet

Testauskohde

Testauskohde on kQC1 ohjelmiston HQC toiminnot.

Tavoitteet

Tavoitteena on saada testata kQC1-ohjelmiston HQC toiminnot mahdollisten virhetilanteiden löytämiseksi. Testauksesta syntyvät raportteja voidaan käyttää kQC1 ohjelmiston laadun arvioinnissa.

Tavoite on saada sovellus toimimaan halutulla tavatta.

Vaikka kehittäjä voi olettaa JavaScript-asetuksen olevan käyttäjän selaimessa aina päällä, tulee tietoturvallisuutta testattaessa ottaa huomioon myös tämä asetus.

Testattavat toiminnot

- Operatiivisten asemien säähavaintojen hakeminen viimeiseltä havaintohetkeltä hakulomakkeella
- Operatiivisten asemien säähavaintojen hakeminen määrittelyltä ajan jaksolta.
- Kaikkien asemien säähavaintojen hakeminen viimeiseltä havaintohetkeltä hakulomakkeella
- Kaikkien asemien säähavaintojen hakeminen määrittelyltä ajan jaksolta.
- Säähavaintojen hakeminen yhdeltä asemalta viimeiseltä havaintohetkeltä hakulomakkeella
- Säähavaintojen hakeminen yhdeltä asemalta määrittelyltä ajan jaksolta.
- Säähavaintojen hakeminen määritellyn ryhmän asemilta viimeiseltä havaintohetkeltä hakulomakkeella
- Säähavaintojen hakeminen määritellyn ryhmän asemilta määrittelyltä ajan jaksolta.
- Karttan piirtyminen annetuilla hakuehdoilla.

Käyttötapaukset testataan kerran manuaalisesti ja niistä kirjoitetaan automaattiset selenium testit.

Toimintojen testitapaukset ja raportointi

Käyttötapauksen kuvaus			
Käyttötapauksen nimi	Testitapauspohja	Testaaja	
Esiehto		Testin suoritus pvm	
Lopputulos		Hyväksytty / Hylätty	Tulos:
1.			
2.			
3.			
4.			
Huomioita			
P:Poikkeama, V:Variaatio			

ILMATIETEEN LAITOS

Käyttötapauksen kuvaus			
Käyttötapauksen nimi	Operatiivisten asemien säähavaintojen hakeminen viimeiseltä havaintohetkeltä hakulomakkeella	Testaaja	
Esiehto	Käyttäjä on kirjautunut QC-sovellukseen ja navigoinut HQC-välilehdelle	Testin suoritus pvm	
Lopputulos	Ohjelma tulostaa taulukkonäkymään hakuehtoihin sopivat havainnot.	Hyväksytty / Hylätty	Tulos:
1. Käyttäjä valitsee ”Operative stations”-valintanapin			Hakulomakkeen asema ja ryhmä pudotusvalikot menevät disable-tilaan.
2. V: Sovellus on valinnut ”Operative stations”-valintanapin oletusarvona			
3. Käyttäjä valitsee ajankohdaksi ”Latest observations”.			”Defined observation time” –tekstikentät menevät disable-tilaan.
4. Käyttäjä valitsee valintaruudulla yhden parametrin hakuehdoksi. Arvoiksi valitaan Parameter: T ja QC Flag: any.			Parametri ja QC flag -pudotusvalikot poistuvat disabled-tilasta.
5. V1: Sovellus on valinnut oletusarvoiksi Parameter: T ja QC Flag: any valitulle hakuehdolle.			Parametri ja QC flag -pudotusvalikot poistuvat disabled-tilasta.
6. V2: Käyttäjä ei valitse yhtään haku parametria.			
7. Käyttäjä klikaa Search-painiketta.			Ohjelma tekee hakuehdoille validoinnin ja tulostaa ruudulle: Your date search terms are not ok! Your parameter search terms are ok! Your search terms: Station: operative_stations observation time: latest_observation_time

		parameters: T any Start searching painike poistuu disabled-tilasta. Täytetyt hakuehdot säilyvät lomakkeella.
8. V: Käyttäjä klikkaa Search-painiketta ja 6.V2 variaatio on tapahtunut		Ohjelma tekee hakuehdoille validoinnin ja tulostaa ruudulle: Your date search terms are not ok! Your parameter search terms are not ok! Your search terms: Station: operative_stations observation time: latest_observation_time parameters: no parameters selected! Start searching painike ei vapaudu disabled-tilasta. Täytetyt hakuehdot säilyvät lomakkeella.
9. Käyttäjä klikkaa Start searching -painiketta		Ohjelma tulostaa ruudulle: Searching... This might take few seconds Kun haku on onnistuneesti valmis teksti vaihtuu: Search is done now! Ohjelma listaa taulukkonäkymään havainnot. Jos taulukkonäkymä on ollut piilotettuna, se avataan automaattisesti. Havainnot: 243 44434 Pokhara Airport Lowland stations 2014-11-21 09:30:00+02 T 21.5 10
10. P:käyttäjä klikkaa Start Searching -painiketta		Ohjelma tulostaa ruudulle: Searching... This might take few seconds Haku ei jostain syystä löydä yhtään hakutulosta ja vaihtaa tekstin: (Virhe viesti) something went wrong!
Huomioita		
P:Poikkeama, V:Variaatio		

Ominaisuudet, joita ei testata

Ohjelman metadata-, Quality Control-, Aggregatrions-, About.-. ja kirjautumis-toimintoja ei testata.

Käyttettävyyden testaaminen

Järjestetään testaustilaisuus jossa kolme koekäyttäjää pääsee testaamaan sovellusta. Testaamiset raportoidaan lomakkeelle johon testauksen valvoja kirjoittaa ylös koehenkilön sovelluksen käyttöä päiväkirjamaisesti. Koe käyttäjää kannustetaan ”ajattelemaan ääneen” sovellusta käyttäessään.

Raportointi

Testauksen tulokset raportoidaan ja arkistoidaan.