LAHDEN AMMATTIKORKEAKOULU
Lahti University of Applied Sciences

SÄÄPALVELUN SUUNNITTELU JA TOTEUTUS

LAHDEN AMMATTIKORKEAKOULU
Tekniikan ala
Tietotekniikan koulutusohjelma
Ohjelmistotekniikka
Opinnäytetyö
Kevät 2013
Jarmo Erola

Lahden ammattikorkeakoulu
Tietotekniikan koulutusohjelma

EROLA, JARMO: Sääpalvelun suunnittelu ja toteutus

Ohjelmistotekniikan opinnäytetyö, 56 sivua

Kevät 2013

TIIVISTELMÄ

Tämä työ tehtiin Ohjelmistotalo Koodiavaimelle suunnitellen ja toteuttaen reaaliaikaista säätilatietoa tarjoava sääpalvelu. Sääpalvelun tarkoituksena on tarjota purjeveneilijöille ja muille merenkävijöille reaaliaikaista havaintotietoa sääennusteiden tueksi, jonka myötä se antaa tärkeää tietoa satamaan lähestymisestä sekä veneen kiinnityksestä. Palvelun tarkoituksena on myös kerätä kattavasti säätilatietoa tutkimista varten, jonka perusteella voidaan toteuttaa paikallisia suunnitelmia eri kohteiden ympärille.

Havaintoasema on toteutettu käyttäen Vaisalan laitteita, joita ovat Vaisala HydroMet™ -järjestelmä MAWS110 sekä WXT520-säälähetin. MAWS110-järjestelmä toimii tiedonkeruu- sekä tietoliikennealustana, joka yhdistää havaintoaseman laitteet erilaisin tiedonsiirtoyhteyksin ja suorittaa havaintotiedoille erilaisia laskelmia. WXT520-säälähetin mittaa tuulen nopeutta ja suuntaa, lämpötilaa, sademäärää, ilmanpainetta sekä suhteellista ilmankosteutta. Havaintoasema on rakennettu yksittäiselle saarelle Hangon kuntaan ja käyttää virtalähteenään akkua, jota ladataan aurinkokennoilla. Havaintoasema käyttää Ohjelmistotalo Koodiavaimen suljettua APN-mobiiliverkkoa turvalliseen tiedonsiirtoon GPRS/UMTS-verkon kautta. Havaintoasema on konfiguroitu lähettämään havaintotietoja tietyin väliajoin palvelinympäristöön, jossa Windows-palvelu vastaanottaa tulevia yhteyksiä ja tallentaa tiedot tietokantaan.

Www-käyttöliittymä on toteutettu Concrete5-sisällönhallintajärjestelmällä, jonka ominaisuuksia on hyödynnetty havaintoasemien ja -tietojen hallintaan ja käsittelyyn. Www-käyttöliittymän toteutuksessa on hyödynnetty MVC-arkkitehtuuria sekä REST-arkkitehtuurityyliä rajapinnan toteutuksessa. Havaintotietoja esittävä kaavio on toteutettu Flot-kaaviokirjastolla, joka on jQuery-Javascript-kirjaston liitännäinen. Asynkroninen tiedonsiirto www-käyttöliittymän ja palvelinrajapinnan välillä on toteutettu jQueryn AJAX-metodilla.

Työn tuloksena valmistui vielä kehitysvaiheessa oleva sääpalvelu, jossa havaintoasema lähettää reaaliaikaista havaintotietoa käsiteltäväksi helppokäyttöisellä www-käyttöliittymällä.

Avainsanat: Concrete5, REST, sääpalvelu, Vaisala

Lahti University of Applied Sciences
Degree Programme in Information Technology

EROLA, JARMO: Designing and developing a weather service

Bachelor's Thesis in Software Engineering, 56 pages

Spring 2013

ABSTRACT

This Bachelor's thesis deals with designing and developing a service which offers realtime weather information. The work was done for Ohjelmistotalo Koodiavain. The purpose of the service is to offer realtime weather information for sailors, to be used along with weather forecasts, which gives important information for approaching the dock and for attaching the boat. Another purpose of the service is to collect weather information for researching and improving the environments of the docks.

An observatory was built, using equipment provided by Vaisala, which are the Vaisala HydroMet™ system MAWS110 and the WXT520 weather transmitter. The MAWS110 system acts as a data logger and communication platform, which connects the observatory equipment by different communication methods and performs different kinds of calculations based on the weather information. The WXT520 weather transmitter measures wind speed and direction, temperature, rainfall, air pressure and relative air humidity. The observatory has been built on an island near Hanko. As a power source it uses a battery charged by solar power. The observatory uses a closed APN mobile network provided by Ohjelmistotalo Koodiavain for secured data communications over the GPRS/UMTS network. The observatory is configured to send weather information at regular intervals to a server where a Windows service receives incoming communications and saves the data to a database.

The WWW user interface was built with the Concrete5 content management system, used for managing the observatory and processing the weather information. MVC architecture was utilized in the WWW user interface and the REST architectural style in programming the interface. The chart presenting the weather information was built with the Flot charting library, which is a plugin of the jQuery-JavaScript library. Asynchronous communication between the user interface and the server API was built using the AJAX method of the jQuery library.

The result of this work was a service where the observatory sends weather information in realtime, processed with a user-friendly user interface. The service is still under development.

Keywords: Concrete5, REST, weather service, Vaisala

LYHENNELUETTELO

ADOdb	ActiveX Data Objects for DataBase
AJAX	Asynchronous JavaScript And XML
AXMLS	ADOdb XML Schema
DOM	Document Object Model
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
MVC	Model-View-Controller
PTU	Pressure, Temperature, Humidity
REST	Representational State Transfer
SaaS	Software as a Service
XML	Extensible Markup Language

SISÄLLYS

1 JOHDANTO

Sää on kaikkialla maapallolla vallitseva luonnonilmiö, joka ohjailee ihmisten elämää ja päätöksiä. Ihmisiä kiinnostaa lähinnä asuinseutunsa säätilanne sekä ennusteet lähipäivien säästä. Sää on tärkeä myös teollisuudelle sekä teiden kuntoa ylläpitäville tahoille, sillä sääennusteiden perusteella voidaan ennakoida maataloudelle haitallisia sateita sekä liikenteen turvallisuusriskejä lisääviä sään vaihteluita.

OmaSää-niminen sääpalvelu on Ohjelmistotalo Koodiavaimen ja yksittäisten asiakkaiden projekti, jossa kehitettävä palvelu tarjoaa reaaliaikaisia säätietoja www-palvelun kautta. Yritys on perustettu vuonna 2006 nimellä MN-Ohjelmisto Oy, ja vuonna 2008 sille rekisteröitiin aputoiminimi Ohjelmistotalo Koodiavain. Yrityksen toimialana on laitteisto- ja ohjelmistokonsultointi, ja sen toimitilat sijaitsevat Nurmijärvellä sekä Lahdessa. Yrityksen palvelutarjonta koostuu ohjelmistokehityksestä, konesali- ja integraatiopalveluista sekä konsultoinnista.

Työn tavoitteena on toteuttaa näyttävä ja toimiva sääpalvelu, jossa automaattiset havaintoasemat lähettävät palvelimelle havaintodataa, jota puolestaan on helppo ja selkeä seurata reaaliaikaisesti palvelussa olevalla työkalulla. Työssä toteutettava havaintoasema on prototyyppi laitekokoonpanosta, jonka pohjalta sääpalvelun havaintoasemat rakennetaan.

Työssä tutustutaan havaintoaseman toimintaperiaatteeseen sekä sen rakenteeseen. Havaintoasema koostuu Vaisala HydroMet™ -järjestelmästä MAWS110 sekä siihen liitetyistä GSM/GPRS-modeemista ja Vaisala WXT520 -säälähettimestä. Kokoonpano on prototyyppi Vaisalan mahdollisesta tulevasta tuotekokoonpanosta. Vapautta havaintoaseman maantieteelliseen sijoitteluun ja tietoturvallisuuteen tuo Koodiavaimen oma APN-mobiiliverkko, jossa havaintodata lähetetään yksityisen GPRS/UMTS-verkon avulla. Työssä perehdytään erilaisiin www-teknologioihin sekä Concrete5-sisällönhallintajärjestelmään, jonka ominaisuuksia hyödynnetään havaintoasemien hallinnassa sekä historiatietoja esittävän käyttöliittymän

toteutuksessa.

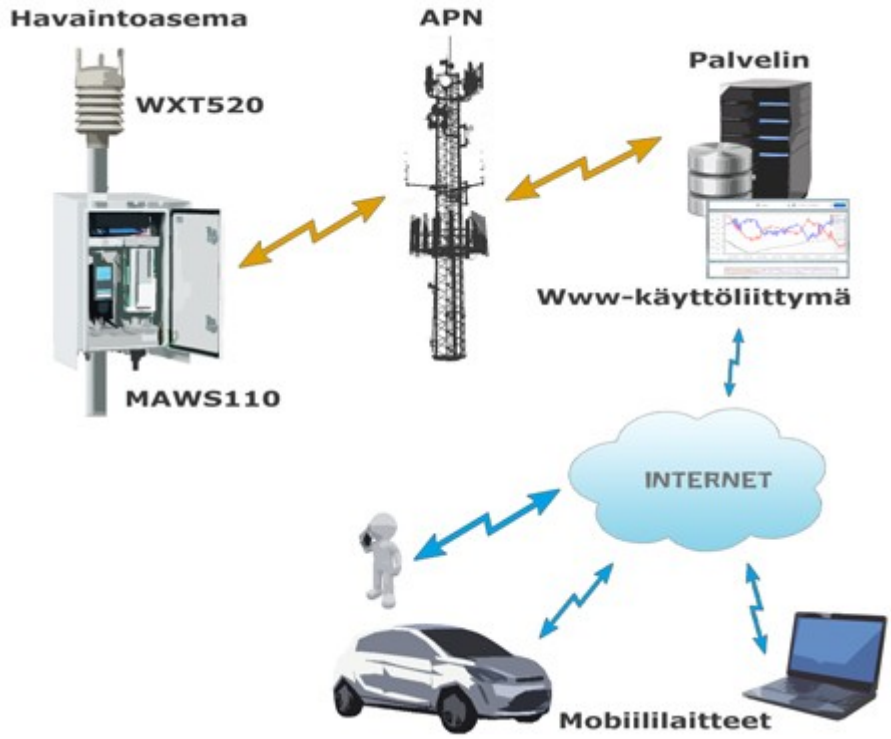
Tässä työssä luvussa 2 käsitellään havaintoaseman toimintaympäristöä sekä luvussa 3 sen toimintaperiaatetta ja rakennetta. Luku 4 käsittelee palvelinrajapintaa, joka koostuu tietokannasta sekä Windows-palvelusta. Luvussa 5 perehdytään erilaisiin sääpalvelussa sovellettaviin www-teknologioihin. Luku 6 käsittelee Concrete5-sisällönhallintajärjestelmää ja kertoo kattavasti sen ominaisuuksista ja rakenteesta. Luvussa 7 on kerrottu sääpalvelun käytännön osuuden toteutuksesta sekä eri tekniikoiden soveltamisesta.

2 SÄÄPALVELUN TOIMINTAYMPÄRISTÖ

Sääpalvelun kehittäminen lähti liikkeelle yksityisen asiakkaan tarpeesta ja pitkäaikaisesta haaveesta saada reaaliaikaista säätilatietoa kohteesta. Asiakkaan kiinnostus säätietoja sekä purjehdusharrastusta kohtaan ovat edistäneet sääpalvelun kehitystä. Kohteiksi on suunniteltu Merikarhut-yhdistyksen venesatamia ympäri Suomen rannikkoa, jolloin reaaliaikainen säätieto yhdistettynä ennusteeseen antaa purjeveneilijöille tärkeää tietoa veneen kiinnityksen ja satamaan lähestymisen suhteen. Sääpalvelun tavoitteena on kerätä pitkältä aikaväliltä kattavasti havaintotietoja, joita voidaan tutkia ja hyödyntää esimerkiksi satamapaikkojen uudelleensuunnittelun ja korjausten yhteydessä. Reaaliaikaista säätilatietoa on tarkoitus tarjota purjehtijoille sekä muille merenkävijöille mahdollisimman selkeän www-käyttöliittymän avulla.

Sääpalvelun kehittämisessä asiakkaan vastuulla on laitteiden hankinta, asennus sekä fyysinen seuranta. Ohjelmistotalo Koodiavaimen vastuulla on tietoliikenneyhteydet, palvelintila ja sovellusten sekä www-käyttöliittymän kehitys. Tietoliikenneyhteyksiä varten käytössä on maan kattava suljettu APN-mobiiliverkko, joten yksittäinen havaintoasema on mahdollista sijoittaa maantieteellisesti mihin tahansa mobiiliverkon kuuluvuusalueella. Järjestelmään kuuluu tietokannan sisältävä palvelin, johon tietyin väliajoin lähetetään havaintoasemien mittaustiedot. Ensimmäinen sääpalvelun kehityksen yhteydessä rakennettu havaintoasema sijaitsee yksittäisellä saarella Hangon kunnassa. Se on välittänyt havaintotietoa syyskuusta 2012 lähtien.

Kuvio 1 esittää järjestelmän toimintaperiaatetta, jossa on havainnollistettu havaintoaseman ja palvelimen välistä kommunikointia keltaisella ja käyttäjien kommunikointia palvelimen kanssa sinisellä värillä. Käyttäjät voivat hyödyntää palvelua Internetiin liitetyillä tietokoneilla ja mobiililaitteilla.



KUVIO 1. Sääpalvelun toimintaperiaate

3 AUTOMAATTINEN HAVAINTOASEMA

Automaattisiksi havaintoasemiksi tai sääasemiksi kutsutaan yleisesti järjestelmiä, jotka sisältävät säätilaa mittaavia laitteita, instrumentteja. Useimmiten havaintoasema sisältää järjestelmän käsittäen tiedonkeruun ja tiedonsiirtoyhteydet sekä useita instrumentteja ilmaisemaan tietoa säätilasta. Keskeisimpiä mittaustietoja ovat tuulen nopeus ja suunta, lämpötila, sademäärä, ilmanpaine sekä suhteellinen ilmankosteus.

Markkinoilla on useita erikokoisia komponentteja, jotka soveltuvat niin suuriin kuin pieniinkin kokoonpanoihin. Pienimpiä komponentteja hyödynnetään lähinnä edullisissa elektronisissa laitteissa, kuten esimerkiksi pienissä kodin sääasemissa. Suurimpia ja tarkempia komponentteja käyttävät mm. säätilan seurantaan erikoistuneet yritykset, jotka tarjoavat erilaisina palveluina säätilatietoja. Suurimmat havaintoasemat ovat tyypillisesti kiinteästi mastoon asennettuja järjestelmiä instrumentteineen, joita voi olla lukuisia määriä.

3.1 Toimintaperiaate

Havaintoasema kattaa koteloidun laitteen sekä siihen liitettyjä instrumentteja, jotka sisältävät useita säätilaa mittaavia antureita. Anturit koostuvat joko yksittäisistä elektroniikan komponenteista tai pienistä laitteista. Instrumentti asennetaan tyypillisesti ilmansuuntien mukaan, jotta laite itsessään pystyy prosessoimaan ja välittämään tiedon tuulen suunnasta. Laite voi vaatia konfigurointia tunnistaakseen asentonsa ilmansuuntiin nähden sekä myös tietoja sen osoitteesta ja muista asetuksista.

Havaintoasemaa sekä mahdollisesti myös yksittäisiä instrumentteja hallitaan ja käytetään erilaisin yksinkertaisin komennoin, joita voidaan välittää esimerkiksi kaapelia pitkin paikallisesti tai langattomasti mobiiliverkon avulla. Komentoja käytetään niin asetusten määrittämiseen kuin mittaustietojen noutamiseen. Laitteet sisältävät oman protokollansa komennon sekä palautettavan viestin muodosta. Säätilaa varten ei tarvitse

lähettää erillistä komentoa, jos laite on asetettu automaattisesti lähettämään mittaustietoja tietyin väliajoin. Tällöin vastaanottava järjestelmä odottaa laitteen lähettämää viestiä ja purkaa sen halutunlaiseen muotoon. Mittaustiedot voidaan tällöin myös tallentaa historiatietojen tarkastelua varten.

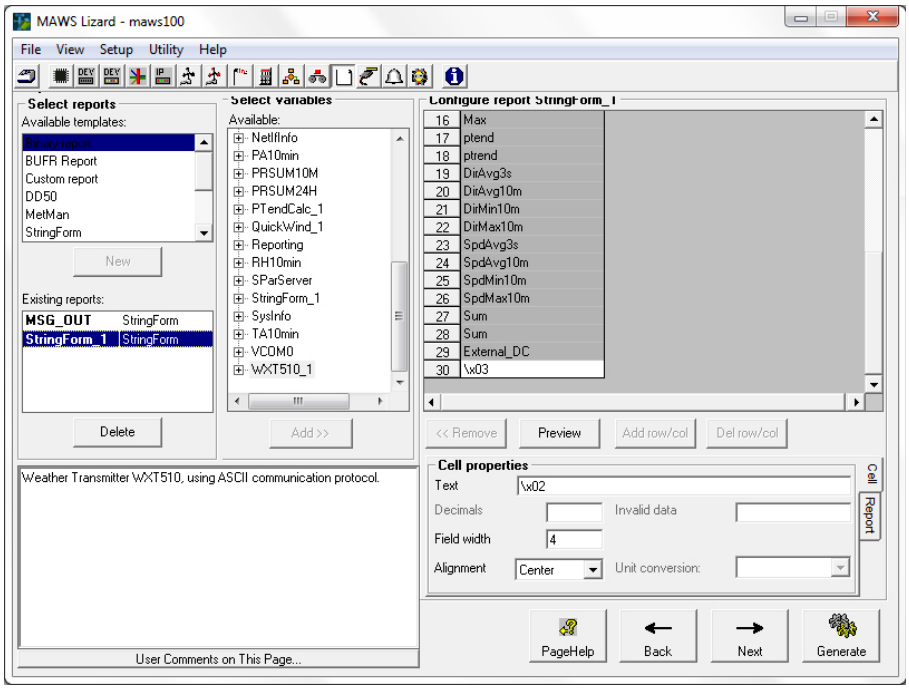
3.2 Vaisala HydroMet™ -järjestelmä MAWS110

Vaisala HydroMet™ -järjestelmä MAWS110 on kompakti, kestävä ja helppokäyttöinen järjestelmä, joka on suunniteltu sovelluksiin, joissa kaupallisen sähkön syöttö tai tietoverkkojen käyttö ei ole mahdollista tai jotka ovat liian kalliita toteuttaa. Joustavuutensa ansiosta MAWS110-järjestelmä sopii erinomaisesti meteorologisiin sovelluksiin, sillä siihen on mahdollista kytkeä monia erilaisia antureita ja telemetrialaitteita. MAWS110-järjestelmää on mahdollista käyttää useilla tiedonsiirtoyhteyksillä, kuten satelliitti-, radio- ja GSM/GPRS-yhteyksillä. Paikalliseen tiedonsiirtoon järjestelmä tarjoaa sarjaportteja sekä lähiverkon (Vaisala 2009).

MAWS110-järjestelmä sisältää QML201-datalogger-moduulin, jonka avulla havaintotietoja pystytään keräämään ilman tietoliikenneyhteyksiäkin. Moduulilla on myös mahdollista tehdä erilaisia laskelmia, kuten mitattujen tietojen keskiarvot, minimi- ja maksimiarvot sekä kumulatiiviset. Laskelmien perusteella voidaan muodostaa myös raportteja, joita voidaan lähettää säännöllisin väliajoin reaaliaikaisia säätietoja vastaanottavalle rajapinnalle. GSM/GPRS-modeemina järjestelmä käyttää Siemens GSM-MC35I-modeemia, joka soveltuu pienen koon, vähäisen virrankulutuksen ja keveyden ansiosta erilaisiin ympäristönvalvontakohteisiin. Modeemilla on mahdollista lähettää SMS-viestejä ja käyttää GPRS-verkkoa 14,4 kbps nopeudella. Järjestelmää varten on hankittu 12 V:n akku sekä aurinkopaneeli, joka mahdollistaa akun lataamisen aurinkoenergialla.

Vaisala HydroMet™-järjestelmä MAWS110 konfiguroidaan MAWS Lizard

-nimisellä ohjelmistolla, jolla määritellään kaikki MAWS-järjestelmään liitetyt anturit ja telemetrialaitteet sekä tehdään niiden asetukset. Ohjelmistolla voidaan määritellä myös erilaisia laskelmia havaintotiedoille. Kuvio 2 esittää MAWS Lizard -ohjelmiston raporttinäkymää, jossa määritetään raportoitavien historiatietojen muoto.



KUVIO 2. Vaisala MAWS Lizard -konfigurointiohjelmisto

3.3 Vaisala WXT520 -säälähetin

Vaisala WXT520 -säälähetin (kuvio 3) on pieni ja kevyt laite, joka tarjoaa kuuden keskeisen säätilaparametrin mittauksen yhdellä instrumentilla. Lähetin mittaa tuulen nopeutta ja suuntaa, lämpötilaa, sademäärää, ilmanpainetta sekä suhteellista ilmankosteutta (Vaisala 2010). WXT520-säälähetin on standardianturi MAWS110-järjestelmään.

KUVIO 3. Vaisala WXT520 -säälähetin (Vaisala 2010)

Laitteessa on neljä vaihtoehtoista sarjaliitäntää kommunikointia varten: RS-232, RS-485, RS-422 sekä SDI-12. Laitteessa on myös valinnaiset kommunikointiprotokollat SDI-12, ASCII sekä NMEA 0183 mittaustietojen välittämistä varten. Näistä ASCII-kommunikointiprotokollaa voidaan käyttää automaattisesti lähettämään mittaustietoja, mutta se on mahdollista asettaa myös kyselytilaan. Kyselytilassa laitteelta voidaan hakea viimeisimmät mittaustiedot haluttuna ajanhetkenä esimerkiksi silloin, kun halutaan nähdä vain nykyinen säätila ilman tarvetta tallentaa historiatietoja. Laitteen mukana on mahdollista saada Vaisalan PC-sovellus asetusten tekemiseen laitteeseen sekä lukuisan määrän erilaisia asennustarvikkeita ja kaapeleita. (Vaisala 2010.)

Laitteen päällä on pyöreä sadetunnistin sekä kolme ultraäänianturia, jotka muuntavat tuulen vireen sähköisiksi signaaleiksi. Laitteen sisäinen logiikka tunnistaa tuulen suunnan anturien välittämästä signaalista. Herkät paine-, kosteus- sekä lämpötila-anturit sijaitsevat laitteen sisässä. Laitteen pohjassa on tarvittavat liitännät virransaantiin, kommunikointikaapeleille sekä nuoli ilmaisemaan laitteen oikean sijoittamisen ilmansuuntien mukaan. Valinnaisena ominaisuutena laitteeseen saa lämmitystoiminnon sulattamaan jään ja lumen. (Vaisala 2010.)

3.3.1 Konfigurointi

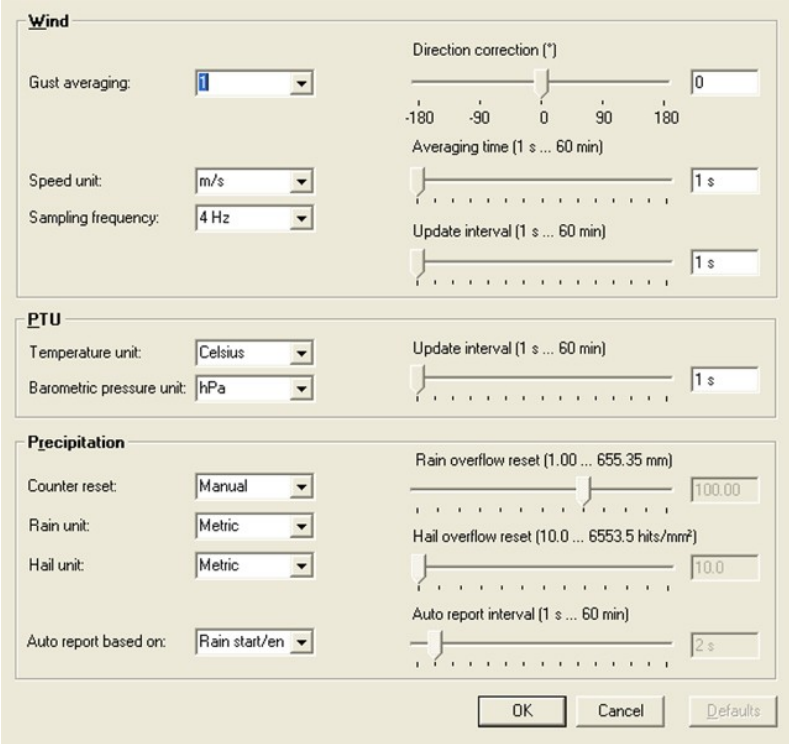
Vaisala WXT520 -säälähetin voidaan asettaa automaattiseen tilaan, jolloin siltä ei tarvitse erikseen noutaa senhetkistä säätilatietoa. Laitteen asetuksiin voidaan määrittää käytettävä kommunikointiprotokolla, joka puolestaan määrittää lähetettävän viestin muodon. ASCII-protokollaa käytettäessä laite voidaan asettaa lähettämään automaattisesti mittaustietoja. Tällöin laitteen asetuksiin tallennetaan yksi tai useampi komento, joita suoritetaan käyttäjän määrittämin väliajoin. Vaisala WXT520-säälähettimessä käytettäviä ASCII-protokollan mukaisia komentoja on esitetty taulukossa 1.

TAULUKKO 1. ASCII-protokollan mukaisia komentoja WXT520-säälähettimelle (Vaisala 2010)

Komento	**Kuvaus**
R0	Yhdistelmäkomento
R1	Tuulikomento
R2	Paine-, lämpötila ja ilmankosteuskomento
R3	Sademääräkomento
R5	Valvontakomento laitteen lämmitykseen sekä virransyöttöön
R	Yhdistelmäkomento komentojen R1, R2, R3 ja R5 yhdistämiseen

WXT520-säälähettimen asetusten tekemiseen käytetään Vaisala Configuration Tool -nimistä ohjelmistoa, jossa määritellään mitattavat suureet. Asetusten tekemiseen voidaan käyttää myös jotain tekstipohjaista pääteohjelmaa. Vaisalan konfigurointityökalulla otetaan yhteys tietokoneeseen liitettyyn säälähettimeen määrittelemällä käytettävä portti, kuten esimerkiksi COM1-sarjaliikenneportti. Lisäksi määritellään sarjaliikenteelle ominaiset asetukset sekä aikamääre, kuinka usein ohjelmisto lähettää säälähettimelle kyselyitä. Viestiasetuksissa määritetään suureet, jotka halutaan vastaanottaa laitteelle lähetettävillä kyselyillä. Kuviossa 4 on esitettynä konfigurointityökalun sensoriasetusikkunasta, jossa on asetukset tuulen suunnalle ja yksikölle, lämpötilan ja ilmanpaineen yksiköille, sademäärän yksiköille sekä edellä mainittujen päivitysaikavälit. Tuulisensorin asetuksissa voi ohjelmallisesti korjata säälähettimen asentoa

ilmansuuntiin nähden, jotta laitteen havainnoima tuulensuunta olisi mahdollisimman tarkka. Asetuksissa on tärkeää huomioida myös metrijärjestelmän mukaiset yksiköt.



KUVIO 4. Sensoriasetukset Vaisala Configuration Tool -ohjelmistossa

3.3.2 Mittaustietojen tulkinta

Yksinkertainen laitteelle lähetettävä tekstipohjainen komento tuulitietojen vastaanottamiseen voi olla esimerkiksi taulukossa 2 esitetyn komennon mukainen. Taulukossa on eritelty yksittäiset parametrit, joista komento koostuu.

TAULUKKO 2. WXT520-säälähettimelle lähetetty komento tuulitietojen noutamiseksi sekä selite

Komento	**0R1<cr><lf>**	
Selite	0	Laitteen osoite
	R1	Tuulitietojen kyselykomento
	<cr><lf>	Lopetusmerkit

Laitteen lähettämän paluuviestin rakenne on yksinkertainen, sillä se koostuu laitteen osoitteesta ja käytetystä komennosta, pilkuilla erotetuista avain-arvo-pareista sekä lopetusmerkeistä. Avain-arvo-parit sisältävät jonkin mitattua arvoa ilmaisevan lyhenteen, joita on esitetty taulukossa 3.

TAULUKKO 3. WXT520-säälähettimen ASCII-protokollan lyhenteet ja yksiköt (Vaisala 2010, 70)

Abbreviation	Name	Unit	Status[1]
Sn	Wind speed minimum	m/s, km/h, mph, knots	#,M, K, S, N
Sm	Wind speed average	m/s, km/h, mph, knots	#,M, K, S, N
Sx	Wind speed maximum	m/s, km/h, mph, knots	#,M, K, S, N
Dn	Wind direction minimum	deg	#, D
Dm	Wind direction average	deg	#, D
Dx	Wind direction maximum	deg	#, D
Pa	Air pressure	hPa, Pa, bar, mmHg, inHg	#, H, P, B, M, I
Ta	Air temperature	°C, °F	#, C, F
Tp	Internal temperature	°C, °F	#, C, F
Ua	Relative humidity	%RH	#, P
Rc	Rain accumulation	mm, in	#, M, I
Rd	Rain duration	s	#, S
Ri	Rain intensity	mm/h, in/h	#, M, I
Rp	Rain peak intensity	mm/h, in/h	#, M, I
Hc	Hail accumulation	hits/cm^2, hits/in^2, hits	#, M, I, H
Hd	Hail duration	s	#, S
Hi	Hail intensity	hits/cm^{2}h, hits/in^{2}h, hits/h	#, M, I, H
Hp	Hail peak intensity	hits/cm^{2}h, hits/in^{2}h, hits/h	#, M, I, H
Th	Heating temperature	°C, °F	#, C, F
Vh	Heating voltage	V	#, N, V, W, F[2]
Vs	Supply voltage	V	V
Vr	3.5 V ref. voltage	V	V
Id	Information field	alphanumeric	

Laitteelta tuleva vastausviesti komennolle ”0R1<cr><lf>” voi olla kuvion 5 mukainen. Vastausviestistä saadaan taulukon 3 lyhenteisiin viitaten tuulen suunnan sekä nopeuden minimi, maksimi ja keskiarvo. Oletusarvoilla tuulen suunnan yksikkö on aste ja nopeuden yksikkö m/s.

```
0R1,Dn=156D,Dm=169D,Dx=190D,Sn=0.0M,Sm=1.0M,Sx=2.1M<cr><lf>
```

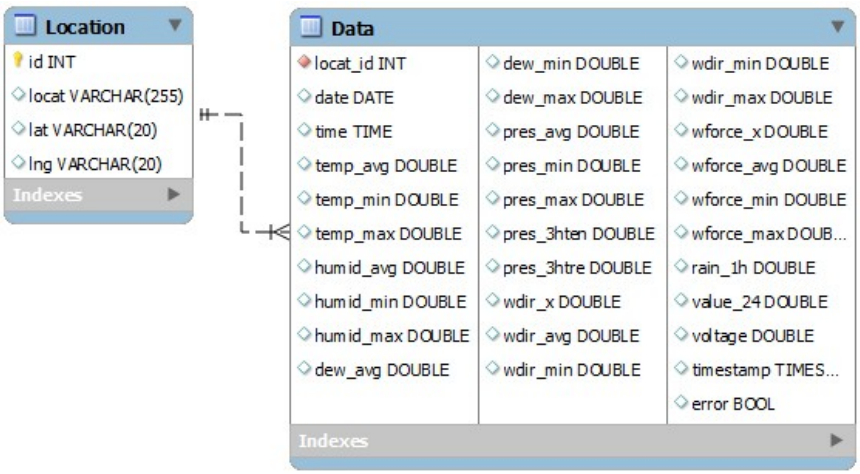
KUVIO 5. WXT520-säälähettimen lähettämä vastausviesti

4 PALVELINRAJAPINTA

Sääpalvelun palvelinrajapinta käsittää tietokantapalvelimen, jossa Windows -palveluna ajettava sovellus vastaanottaa havaintoasemalta tulevan datan ja tallentaa sen tietokantaan. Se käsittää myös www-palvelimen, joka tarjoaa käyttöliittymän havaintodatan tarkasteluun.

4.1 Tietokannan rakenne

Tietokantapalvelimena toimii Microsoft SQL Server -relaatiotietokanta-järjestelmä, johon varastoidaan tiedot havaintoasemista sekä -datasta. Tietokanta sisältää ”Location”-taulun, johon tallennetaan yksittäisen havaintoaseman sijainnin nimi sekä koordinaatit. ”Data”-taulu sisältää havaintoaseman lähettämiä tietoja DOUBLE-tyyppisinä numero-arvoina. Kuvio 6 esittää tietokannan rakennetta sekä taulujen välisen relaation.



KUVIO 6. Tietokannan rakenne

4.2 Windows-palvelu

Windows-palvelut ovat Microsoft Windows -käyttöjärjestelmässä suoritettavia sovelluksia, joiden käynnistäminen tai hallinnointi ei vaadi käyttäjän vuorovaikutusta. Nimensä mukaisesti ne ovat palveluita, joita muut sovellukset ja Windows-käyttöjärjestelmän komponentit käyttävät. Sääpalvelua varten toteutettava palvelu kuuntelee saapuvia yhteyksiä

havaintoasemalta ja tallentaa havaintotiedot tietokantaan.

Palveluna toimiva sovellus on toteutettu Microsoftin kehittämällä C#-ohjelmointikielellä, joka yhdistää C++-ohjelmointikielen tehokkuuden sekä Java-ohjelmointikielen helppokäyttöisyyden. Sovellus koostuu TcpListener-objektista, joka kuuntelee saapuvia TCP-tietoliikenneprotokollan yhteyksiä sekä säikeestä (Thread), joka pitää palvelun käynnissä. Kuvio 7 esittää palvelun Main-funktiota, jossa luodaan TcpListener-objekti sekä säie, jolle annetaan määritettyjä parametreja käynnistyksen yhteydessä.

```
1  static void Main(string[] args){
2      tcpListener = new TcpListener(IPAddress.Any, portNum);
3      listenThread = new Thread(new ParameterizedThreadStart(ListenForClients));
4      listenThread.Start(data);
5  }
```

KUVIO 7. Havaintotietoja kuuntelevan palvelun Main-funktio

Säikeen käynnistyksen yhteydessä käynnistetään TcpListener-objekti, joka aloittaa yhteyksien kuuntelemisen tietyssä portissa. Yhteyden saapuessa luodaan uusi säie käsittelemään havaintoasemalta tuleva data, jotta kuunteleva säie voi jatkaa uusien yhteyksien kuuntelemista. Yhteyttä käsittelevä säie purkaa havaintotiedot ja tallentaa ne tietokantaan.

5 WWW-TEKNOLOGIAT

5.1 MVC-arkkitehtuuri

Graafisten käyttöliittymien tapauksessa ohjelmistot voivat vaatia useita erilaisia muutoksia niin käyttöliittymän kuin ohjelmalogiikan osalta. Erilaiset käyttäjien vaatimukset laajentavat ohjelmistoa, ja usein myös käyttöliittymää täytyy päivittää uusien ominaisuuksien toteutumiseksi. Jos käyttöliittymä on upotettuna ohjelmalogiikkaan, voi erilaisten muutoksien toteuttaminen osoittautua hankalaksi, sillä tyypillisesti käyttöliittymä muuttuu ohjelmalogiikkaa useammin.

Ongelmaan on ratkaisuna Model-View-Controller-arkkitehtuuri (MVC), joka on tiettävästi käytetyimpiä arkkitehtuurimalleja graafisten sovellusten kehityksessä. MVC:n esitteli norjalainen tietotekniikan tutkija Trygve Reenskaug vuonna 1979 (Reenskaug 1979). MVC-arkkitehtuuri erottelee käyttöliittymän ohjelmalogiikasta, joka on jaoteltu vielä malleihin (Models) sekä ohjaimiin (Controllers). Arkkitehtuurissa käyttöliittymää ja sen osia kutsutaan näkymiksi (Views), jotka määrittävät tiedon esitystavan sovelluksessa. Mallit kuvaavat tiedon tallentamista ja käsittelyä järjestelmässä. Ohjaimet käsittelevät saapuvia kyselyitä ja ohjaavat malleja ja näkymiä muodostaakseen vastauksen kyselyyn.

Näkymät ja ohjaimet ovat riippuvaisia mallista, mutta yksittäinen malli ei ole riippuvainen kummastakaan. Näin ollen yksittäistä mallia on mahdollista testata ilman ympärillä olevaa sovellusta. Useimmissa tapauksissa www-sovelluksista näkymät voidaan yhdistää ohjaimeen, kun on tarkoitus lähettää esimerkiksi JSON-tyyppistä tietoa asiakkaalle. Varsinkin nykypäivänä kehittyneiden www-sovellusten puitteissa näkymät voidaan toteuttaa käyttäjän selaimessa JavaScript-näkyminä, jolloin käyttäjän selaimen ja palvelimen välillä siirretään tietoa asynkronisesti jonkin määritetyn datatyypin mukaisesti.

5.2 REST-arkkitehtuurityyli

Vuonna 2000 amerikkalainen tietotekniikan asiantuntija Roy Fielding käsitteli tohtorin väitöskirjassaan Architectural Styles and the Design of Network-based Software Architectures erilaisten arkkitehtuurityylien vahvuuksia ja heikkouksia sekä esitteli arkkitehtuurityylin ”Representational State Transfer” (REST). REST tarjoaa arkkitehtuurisia rajoitteita, joiden avulla voidaan korostaa komponenttien välistä skaalautuvuutta sekä pienentää vuorovaikutusviiveitä. Arkkitehtuuri, jonka katsotaan noudattavan REST:n rajoitteita, sanotaan olevan REST-arkkitehtuurityylin mukainen. (Fielding 2000b.)

REST-arkkitehtuurityylin ydin koostuu asiakas-palvelin-arkkitehtuurityylistä, tilattomuudesta, välimuistin käytön mahdollisuudesta, yhdenmukaisesta rajapinnasta sekä resurssikäsitteestä. Asiakas-palvelin-arkkitehtuurityylin mukaisesti käyttöliittymä erotetaan tietovarastosta, jolloin käyttöliittymä on siirrettävissä eri alustoille sekä parantaa skaalautuvuutta yksinkertaistamalla palvelimen komponentteja. Tilattomuudella tarkoitetaan palvelimen tilattomuutta, joka sisältää kolme ominaisuutta; näkyvyys, luotettavuus sekä skaalautuvuus. Näkyvyys paranee, kun palvelin ymmärtää asiakkaalta tulevan kyselyn. Luotettavuus paranee, koska se helpotta toipumista osittaisista virheistä. Skaalautuvuus paranee, koska kyselyiden välisiä tiloja ei tarvitse tallentaa, jolloin palvelin pystyy vapauttamaan resursseja nopeasti. Välimuistilla tarkoitetaan asiakkaalle lähetettävän vastauksen tallentamista välimuistiin (cacheable) tai lähettämällä se suoraan asiakkaalle (non-cacheable). Yhdenmukainen rajapinta korostaa komponenttien välistä rajapintaa. (Fielding 2000b.)

REST:ssä resurssilla tarkoitetaan abstraktiota, joka on keskeinen käsite sen toiminnasta. Resurssilla viitataan kaikkeen tietoon, joka voidaan nimetä. Resurssin esitys voi palauttaa muuttuneita arvoja erilaisten tapahtumien yhteydessä, esimerkiksi jonkin dokumentin tapauksessa resurssi ”dokumentin viimeisin muokkauspäivä” muuttuu ajan mittaan sitä tallennettaessa. (Fielding 2000b.)

5.2.1 REST ja HTTP-protokolla

Hypertext Transfer Protocol -protokollalla (HTTP) on ollut erityinen rooli web-arkkitehtuurin sovellusten välisessä viestinnässä protokollien avulla. REST:n ominaisuuksien tukemiseksi HTTP-protokollaan jouduttiin toteuttamaan useita muutoksia. HTTP-protokollan kehittäjät vaativat todisteita muutosten tuomista parannuksista, jotta ne voitiin hyväksyä. REST:ä käytettiin tunnistamaan ongelmia olemassa olevasta HTTP-toteutuksesta. Sitä käytettiin myös mallintamaan HTTP-protokollaan perustuvien web-sovellusten tehokkuutta. Loppujen lopuksi REST:ä on käytetty rajoittamaan standardisoituja HTTP-laajennuksia sen sijaan, että se olisi sallinut niiden vastaisia toimintoja. Yksi REST:n tärkeimmistä tavoitteista on tukea asteittaista ja hajautettua muutosten käyttöönottoa olemassa olevaan arkkitehtuuriin. HTTP-protokollaa muutettiin tukemaan tätä tavoitetta ottamalla käyttöön versioinnin vaatimukset ja sääntöjä laajentamaan protokollan jokaista syntaksin elementtiä. (Fielding 2000a.)

REST:n resurssien käsittelyyn täytyy käyttää standardisoituja metodeita, jotta arkkitehtuurityylin asettama rajoite yhdenmukaisesta rajapinnasta toteutuu. Nykyinen HTTP-protokolla määrittelee seuraavat metodit: OPTIONS, GET, HEAD, POST, PUT, DELETE, TRACE ja CONNECT. OPTIONS-metodilla voidaan selvittää saatavilla olevat yhteysominaisuudet annetusta osoitteesta. GET-metodilla voidaan noutaa tietoa annetusta osoitteesta. HEAD-metodi on samantyyppinen kuin GET-metodi, mutta se palauttaa ainoastaan otsikkotietoja. POST-metodilla luodaan uusia resursseja, PUT-metodilla luodaan uusia tai päivitetään olemassa olevia resursseja ja DELETE-metodilla poistetaan resursseja. TRACE-metodia käytetään palauttamaan kysely sellaisenaan takaisin asiakkaalle. CONNECT-metodia voidaan käyttää välityspalvelinten kanssa, jotka voivat dynaamisesti vaihtua tunneleiksi. (Fielding, Gettys, Mogul, Frystyk, Masinter, Leach, Barners-Lee 1999.)

HTTP-metodit on jaettu turvallisiin (safe) ja idempotentteihin (idempotent) metodeihin. Turvalliset metodit, kuten OPTIONS-, GET-, HEAD- ja

TRACE-metodit, eivät vaikuta resursseihin. Idempotentit metodit vaikuttavat resursseihin, mutta usean samanlaisen kyselyn toiminto on täysin samanlainen kuin yksittäisen kyselyn. Turvallisten metodien lisäksi PUT- ja DELETE-metodit sisältävät em. ominaisuuden, joten ne ovat idempotentteja metodeja. (Fielding, Gettys, Mogul, Frystyk, Masinter, Leach, Barners-Lee 1999.)

5.2.2 RESTful-palvelu

Www-palvelua, joka käyttää täsmällisesti HTTP-metodeja kuten ne on määritelty RFC 2616 -määritelmässä, kutsutaan myös RESTful-palveluksi. REST:n perusperiaate sitoo resurssien luomisen, lukemisen, päivittämisen ja poistamisen HTTP-metodeihin. (Rodriguez 2008.)

Tavanomaisessa www-palvelussa tietoa tallennetaan palvelimelle lähettämällä kysely, joka sisältää lisättäviä tietoja. Palvelimelle lähetettävä kysely asiakkaan lisäämisestä voi olla esimerkiksi kuvion 8 mukainen.

```
GET /addcustomer?name=John HTTP/1.1
```

KUVIO 8. Kysely asiakkaan lisäämiseksi

Yllä olevasta kyselystä käy ilmi, että käytettävä HTTP-metodi on GET ja kutsuttava resurssi ”addcustomer”. Palvelimelle välitetään asiakkaan nimi ”name”-muuttujana. Tyypillisesti edellä oleva kysely suoritetaan selaimen osoiteriville kirjoitetulla osoitteella, jolloin kaikki kyselyt ovat GET-tyyppisiä. Asiakkaan tietoja muutettaessa kysely voi olla esimerkiksi kuvion 9 mukainen.

```
GET /updatecustomer?id=4&name=Jake HTTP/1.1
```

KUVIO 9. Kysely asiakastietojen päivittämiseksi

Yllä olevassa kyselyssä kutsutaan ”updatecustomer”-resurssia

asiakastietojen päivittämiseksi. Kyselyssä välitetään ”id”-muuttuja, joka on asiakkaan yksilöllinen tunniste sekä ”name”-muuttuja, joka on määritetyn asiakkaan uusi nimi. Asiakkaan poistamiseksi kutsuttaisiin resurssia ”deletecustomer” ja välitettäisiin muuttujana poistettavan asiakkaan yksilöllinen tunniste.

HTTP-metodien täsmällisellä hyödyntämisellä edellä oleva esimerkki voidaan toteuttaa yhdenmukaiseksi saavuttaen RESTful-palvelu. Käytännössä yksi resurssi, esimerkiksi ”customers”, toimii asiakkaiden noutamisen, lisäämisen, päivittämisen sekä poistamisen resurssina. Rajapinta toteuttaa halutun toiminnon käytettävän HTTP-metodin perusteella. Kuviossa 10 on listattu REST-rajanpinnan mukaiset kyselyt, jotka toteuttavat asiakkaan tietojen noutamisen (rivi 1), lisäämisen (rivi 2), päivittämisen (rivi 3) sekä poistamisen (rivi 4).

```
1  GET /customers/4          HTTP/1.1
2  POST /customers/John      HTTP/1.1
3  PUT /customers/4/Jake     HTTP/1.1
4  DELETE /customers/4       HTTP/1.1
```

KUVIO 10. REST-rajapinnan mukaisia kyselyitä

Taulukossa 4 on eritelty HTTP-metodit sekä niiden käyttötarkoitukset resurssin perusteella. Taulukossa esitetään resurssit kokoelmina, jotka kuvaavat kaikkia asiakkaita sekä elementteinä, jotka kuvaavat yksittäisiä asiakkaita.

TAULUKKO 4. HTTP-metodit sekä niiden käyttötarkoitukset

Resurssi	**GET**	**PUT**	**POST**	**DELETE**
Kokoelma /customers	Asiakaslistan noutaminen	Kokonaisen kokoelman korvaaminen toisella kokoelmalla	Uuden elementin luonti kokoelmaan	Kokonaisen kokoelman poistaminen
Elementti /customers/4	Asiakkaan tietojen noutaminen	Asiakkaan tietojen päivittäminen tai lisääminen, jos asiakasta ei löydy	Uuden asiakkaan lisääminen	Asiakkaan poistaminen

5.3 jQuery ja AJAX

jQuery on avoimen lähdekoodin JavaScript-kirjasto sisältäen monia ominaisuuksia, kuten mm. XHTML-kuvauskielisten www-sivujen sisällön muokkaamisen, tapahtumien käsittelyn, animaatiot sekä helppokäyttöisen Asynchronous JavaScript And XML (AJAX) -tekniikalla toteutetun metodin asynkronisten kyselyiden suorittamiseksi. Kirjaston monipuolisuuden ja laajennettavuuden ansiosta miljoonat ihmiset käyttävät sitä JavaScript-kielisissä sovelluksissaan. jQuery-kirjastoa käytetään niin tavallisten www-sivujen kuin laajojenkin ohjelmistokokonaisuuksien toteutuksessa. Tyypillisesti kirjaston avulla toteutetaan interaktiivisia ja asynkronisia toimintoja sulavan toimivuuden saavuttamiseksi. (The jQuery Foundation 2013.)

AJAX koostuu monista www-sovelluksissa käytettävistä tekniikoista, joiden avulla sovelluksesta saadaan vuorovaikutteisempi. AJAX perustuu moniin olemassa oleviin www-standardeihin ja tekniikoihin, joihin lukeutuvat mm. JavaScript, XML, XHTML, DOM, CSS sekä XMLHttpRequest-objekti. AJAXin tarkoitus on hyödyntää edellä mainittuja tekniikoita keskenään mahdollistaakseen nopeita ja dynaamisia www-sivustoja. AJAX mahdollistaa sivuston sisällön asynkronisen päivittämisen ilman tarvetta päivittää koko sivua. (W3Schools 2013.)

jQuery sisältää AJAX-metodin, jonka avulla JavaScript-kielinen sovellus voi suorittaa asynkronisia AJAX-kyselyitä. Metodin syntaksi on helposti ymmärrettävä, josta esimerkki on havainnollistettu kuviossa 11. Kyselyä varten määritettävät asetukset (rivit 2-10) koostuvat käytetystä HTTP-metodista (rivi 2), kutsuttavan tiedoston osoitteesta (rivi 3), datatyypistä (rivi 4), datan eri verkkotunnukseen lähettämisen salliminen (rivi 5) sekä lähetettävästä datasta (rivit 6-10). Onnistuneen kyselyn jälkeen suoritetaan ”done”-metodi, joka saa parametrikseen palvelimelta palautuvan datan, joka esimerkin tapauksessa sijoitetaan ”alert”-funktiolla selainikkunaan ilmestyvään ikkunaan.

```
$.ajax({
    type: "POST",
    url: "http://site.com/api.php",
    dataType: 'jsonp',
    crossDomain: true,
    data: {
        id: 2,
        action: "save",
        name: "John"
    }
}).done(function( reply ) {
    alert( "Data Saved: " + reply );
});
```

KUVIO 11. Tietojen lähetys palvelimelle käyttäen jQuery-kirjaston AJAX-metodia

6 CONCRETE5-SISÄLLÖNHALLINTAJÄRJESTELMÄ

Concrete5 on helppokäyttöinen ja monipuolinen avoimeen lähdekoodiin perustuva sisällönhallintajärjestelmä, jonka kehitys alkoi vuonna 2003 (Concrete5 2013c). Sitä käytetään monien nettisivustojen sekä www-sovellusten alustana, sillä avoimen lähdekoodin sekä MIT-ohjelmisto-lisenssin ansiosta sen käyttöä ei ole rajoitettu. Concrete5 sisältää monia hyödyllisiä ominaisuuksia, kuten esimerkiksi käyttäjien ja tiedostojen hallinnan, helpon laajennettavuuden lisäosilla sekä helposti omaksuttavan rakenteen. Concrete5 on suunniteltu helppokäyttöiseksi niin, että kuka tahansa voi aloittaa omien nettisivujen luomisen ilman sen suurempaa kokemusta asiasta. (Concrete5 2013a.)

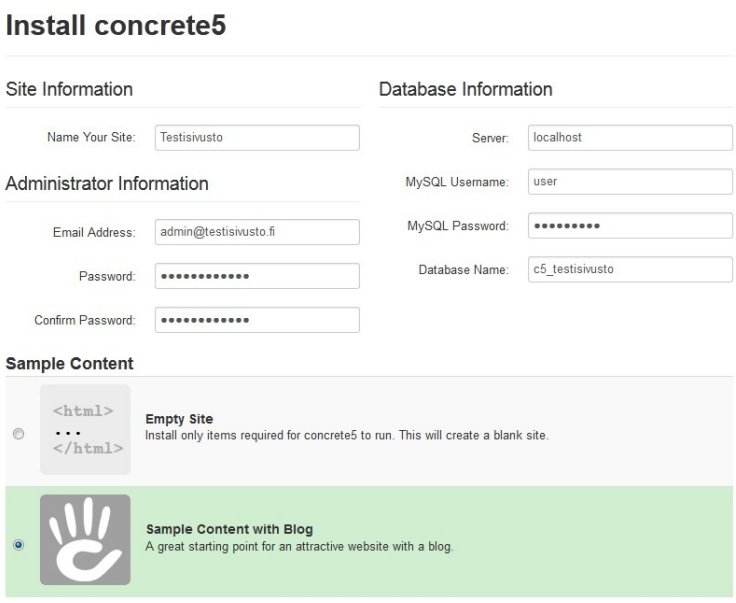
Concrete5-sisällönhallintajärjestelmä käyttää MVC-arkkitehtuuria ja sisältää ylikirjoitusominaisuuden. Lähes kaikki Concrete5:n toiminnallisuus voidaan ylikirjoittaa, joten sen ohjelmaydin ei aseta rajoitteita kehitystyölle. Concrete5 käyttää jQuery-JavaScript-kirjastoa lähes kaikissa JavaScript-tarpeissa sekä Zend-kirjastoja mm. sivuston välimuistin hallintaan. Concrete5 käyttää ADOdb-tietokanta-abstraktiokirjastoa tietokantojen käsittelyyn. Myös tietokannan taulujen määritys esimerkiksi lohkojen tapauksessa tapahtuu ADOdb-kirjaston avulla käyttäen sen omaa AXMLS-formaattia. Concrete5 perustuu in-context-editing-periaatteeseen, jossa käyttäjä voi sivustolle kirjauduttuaan navigoida sivustolla normaalisti ja tehdä haluamansa muutokset suoraan sisältöön. Modulaarisen ja ylikirjoitettavan rakenteensa ansiosta Concrete5 on erittäin tehokas ja helppo ymmärtää. (Concrete5 2013b.)

6.1 Käyttöönotto

Concrete5-sisällönhallintajärjestelmä on mahdollista ladata pakattuna tiedostona joko sen kotisivuilta tai GitHub-versionhallinnasta. Järjestelmän tiedostot puretaan pakatusta tiedostosta ja siirretään www-palvelimelle haluttuun hakemistoon. Concrete5:n juurihakemistoille ”config”, ”packages” sekä ”files” tulee asettaa luku-, kirjoitus- ja suoritusoikeudet kaikille

käyttäjille. Concrete5 käyttää yhtä tietokantaa, jonka käyttäjällä täytyy olla vähintään INSERT-, SELECT-, UPDATE-, DELETE-, CREATE-, DROP- sekä ALTER-oikeudet.

Alkutoimenpiteiden jälkeen navigoidaan hakemistoon, johon järjestelmän tiedostot kopioitiin. Mikäli jotkin järjestelmän asentamiseen vaadittavat asiat, kuten esimerkiksi tietokantatuki tai hakemistojen käyttöoikeudet, ovat puutteellisia, ilmoittaa asennusohjelma niistä ja pyytää korjaamaan ne. Asennusnäkymässä käyttäjä määrittelee kuvion 12 mukaisesti sivustolleen nimen, pääkäyttäjän sähköpostiosoitteen ja salasanan sekä tietokanta -yhteyden tiedot. Tässä vaiheessa käyttäjä voi päättää, asennetaanko sivustolle esimerkkisisältöä, joka on kokemattomalle käyttäjälle suositeltavaa. Asennusohjelma generoi asennushakemistoon mm. konfiguraatiotiedostot ja asentaa tietokantaan kaikki järjestelmän ydintoiminnot. Kun asennus on valmis, voi käyttäjä heti aloittaa sivustonsa rakentamisen.

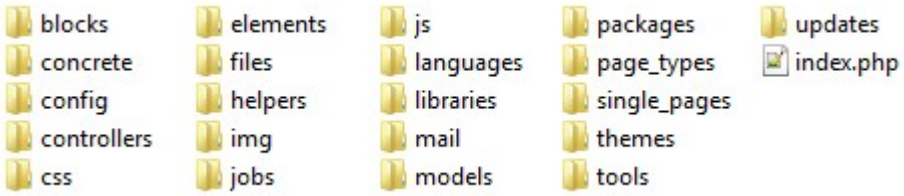


KUVIO 12. Concrete5-sisällönhallintajärjestelmän asennusohjelma

6.2 Rakenne

Concrete5-sisällönhallintajärjestelmä koostuu useista hakemistoista sen

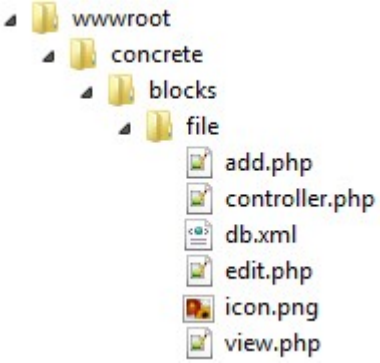
modulaarisen rakenteen sekä MVC-arkkitehtuurin takia. Esimerkiksi Concrete5:n ohjelmaytimen hakemisto ”concrete” sisältää täysin samanlaisen rakenteen kuin juurihakemisto. Kuvio 13 esittää Concrete5:n hakemistorakennetta.



KUVIO 13. Concrete5-sisällönhallintajärjestelmän hakemistorakenne

6.2.1 Lohkotyypit

Lohkotyypit ovat sivustolla käytettäviä yksittäisiä komponentteja, joita voidaan lisätä useita erilaisia sivustolla oleviin lohkoalueisiin. Lohkotyypit sijaitsevat hakemistossa ”blocks”, ja ne asennetaan järjestelmään hallintapaneelin kautta. Lohkotyypin rakenne on esitetty kuviossa 14, joka esittää Concrete5:n oletuslohkotyyppiä ”file”. Lohkotyyppi koostuu neljästä pakollisesta tiedostosta, jotka ovat ohjain ”controller.php”, näkymä ”view.php” sekä lisäys- ja muokkauslomakkeet ”add.php” ja ”edit.php”. Lohkotyypille voidaan määritellä myös tietokantakuvaus ”db.xml” sekä kuva ”icon.png”, joka näkyy hallintapaneelissa lohkotyyppien kohdalla. Lohkotyypit löytyvät hallintapaneelin osiosta ”Stacks & Blocks → Block Types”.



KUVIO 14. Lohkotyypin rakenne

Lähes jokainen lohkotyyppi käyttää tietokantaa tallentaakseen lohkotyyppikohtaisia tietoja, kuten esimerkiksi sen yksilöllisen tunnisteen sekä nimen. Tietokannan rakenne määritellään tiedostossa ”db.xml”, jossa tiedot esitetään AXMLS-formaatin mukaisesti. Kuvio 15 esittää ”file”-lohkotyypin tietokannan rakenteen. Tiedosto määrittelee tietokannan taulun ”btConctentFile”, joka koostuu kokonaislukutyyppisistä ”bID”- ja ”fID”-kentistä sekä tekstityyppisistä ”fileLinkText”- ja ”filePassword”-kentistä. Kenttä ”bID” on määritetty avainkentäksi.

```
1  <?xml version="1.0"?>
2  <schema version="0.3">
3      <table name="btContentFile">
4          <field name="bID" type="I">
5              <key />
6              <unsigned />
7          </field>
8          <field name="fID" type="I">
9              <unsigned />
10         </field>
11         <field name="fileLinkText" type="C" size="255"></field>
12         <field name="filePassword" type="C" size="255"></field>
13     </table>
14 </schema>
```

KUVIO 15. Lohkotyypin tietokantarakenne AXMLS-formaatissa

6.2.2 Ohjaimet

Ohjaimet ovat MVC-arkkitehtuurin mukaisia ohjaimia, joita käytetään tyypillisesti yksittäisillä sivuilla, erilaisilla näkymillä sekä hallintapaneelin moduuleilla. Ohjain käsittelee käyttäjän kyselyt, tekee tarvittaessa tietokantakyselyitä ja välittää tietoa käytettävään näkymään. Kuviossa 16 on esitetty esimerkki ohjaintiedostosta, joka sisältää luokan ”TestController”. Concrete5-järjestelmän ”Controller”-luokasta periytetty luokka sisältää monia metodeja sen käsittelyyn, kuten esimerkiksi ”on_start”-metodin (rivi 8), joka suoritetaan aina kun luokasta luodaan objekti. Lähes kaikki Concrete5-järjestelmän komponentit on ladattavissa ohjaimessa, kuten esimerkiksi mallit (rivi 19) ja apufunktiot (rivi 9). Ohjain voi välittää muuttujia näkymälle määrittämällä ne käyttäen ”set”-metodia (rivi 23), jossa ensimmäisessä parametrissa nimetään muuttujan nimi ja toisessa sen arvo.

```php
<?php defined('C5_EXECUTE') or die(_("Access Denied."));

class TestController extends Controller {

    private $parameter1 = '';
    private $error = '';

    public function on_start() {
        $this->error = Loader::helper('validation/error');
    }

    public function on_before_render() {
        $this->set('error', $this->error);
    }

    public function view() {
        $atID = $this->request('atID');

        Loader::model('database', 'testpackage');
        $dbModel = new DatabaseModel();
        $data = $dbModel->getByAtID($atID);

        $this->set('type', $data);
    }

}
```

KUVIO 16. Concrete5-järjestelmän ohjaintiedosto

6.2.3 Mallit

Mallit ovat MVC-arkkitehtuurin mukaisia malleja, jotka toimivat esimerkiksi rajapintana tietokannoille. Concrete5 sisältää ADOdb Active Record -kirjaston tietokantakyselyitä varten. ADOdb Active Record -kirjasto on olio-relaatiokuvaus (ORM), jossa yksittäinen luokka edustaa kuvion 17 mukaisesti tietokannassa sijaitsevaa taulua. Mallit sijaitsevat hakemistossa ”models”.

```php
<?php

class Customer extends Model {

    private $id;
    private $name;
    static $primary_table = 'customers';

    public function __construct($id = 0) {
        $this->init($id);
    }

    private function init($id) {
        if ($id !== 0) {
            $data = $this->getByID($id);
            $this->name = $data['name'];
        }
    }

    public function getName() {
        return $this->name;
    }

    public static function getByID($id) {
        $db = Loader::db();
        $sql = 'SELECT * from '.self::$primary_table.' WHERE id='.$id;
        return $db->getRow($sql);
    }

}
```

KUVIO 17. ORM-tyyppinen tietokantaluokka

6.2.4 Näkymät

Näkymät ovat MVC-arkkitehtuurin mukaisia näkymiä, jotka määräävät ohjaimelta tulevan datan esitystavan. Näkymiä kutsutaan myös yksittäisiksi sivuiksi, jotka nimensä mukaisesti toimivat yksittäisinä sivuina sivustolla. Tällainen sivu voi olla esimerkiksi ”login”-sivu, jota ei voi olla useampia samalla sivustolla. Näkymät sijaitsevat hakemistossa ”single_pages”, ja ne vaativat asentamisen järjestelmään hallintapaneelin osiosta ”Pages & Themes → Single Pages”. Näkymät asennetaan järjestelmään määrittämällä sivun polku sivuston osoitteen jälkeen, esimerkiksi ”testi”, jolloin näkymää pääsee tarkastelemaan osoitteella ”www.sivusto.net/testi”. Tässä tapauksessa hakemistossa ”single_pages” tulee olla tiedosto ”testi.php”, joka on määritetty sivustolla yksittäiseksi sivuksi.

6.2.5 Tehtävät

Tehtävät ovat sivustolla suoritettavia tehtäviä, joiden avulla sivuston ylläpitoon liittyviä rutiineja pystytään automatisoimaan. Tehtävät voidaan suorittaa hallintapaneelista käsin tai automatisoida suoritettavaksi tiettyinä ajankohtina. Concrete5 sisältää valmiiksi mm. sivukartan generoinnin sekä sivuston indeksoinnin hakukoneita varten. Tehtävät ovat luokkia ja sijaitsevat hakemistossa ”jobs”.

6.2.6 Teemat

Teemat ovat sivustolla käytettäviä ulkoasumalleja, joiden avulla sivuston ulkoista ilmettä voidaan ylläpitää. Jokaiselle Concrete5-järjestelmän sisältösivulle voidaan määritellä oma teema. Teemat sisältävät ”header.php”- ja ”footer.php”-tiedostot, jotka määrittävät sivuston ylä- ja alaosan. Sisältöä varten teema sisältää erilaisia sivutyyppejä. Lisäksi teemat sisältävät CSS-tyylitiedostoja sekä kuvia. Teemat sijaitsevat omissa hakemistoissaan juurihakemistossa ”themes”.

6.2.7 Sivutyypit

Sivutyypit ovat teemoissa käytettäviä malleja, jotka määrittelevät sivuston sisällön asettelun. Esimerkiksi ”full”-sivutyyppi ilmaisee sisältöalueen olevan koko sivun levyinen ja ”left_sidebar” sisältöalueen sisältävän sivupalkin vasemmalla reunalla. Jokaiselle Concrete5-järjestelmän sivulle voi määritellä oman sivutyypin. Sivutyypeille määritetään yleensä lohkoalueita, jotka näkyvät sivun muokkaustilassa punaisella katkoviivalla rajattuina alueina, joihin voi sijoittaa erilaisia lohkotyyppejä. Lohkoalueet ovat kuvion 18 mukaisia PHP-kielisiä ohjelman osia HTML-tiedostoon upotettuna. Teemakohtaiset sivutyypit sijaitsevat teemojen hakemistoissa ja kaikilla teemoilla käytettävät sivutyypit hakemistossa ”page_types”.

```
1  <?php
2      $a = new Area('Main');
3      $a->display($c);
4  ?>
```

KUVIO 18. Lohkoalueen määritys Concrete5-teemassa

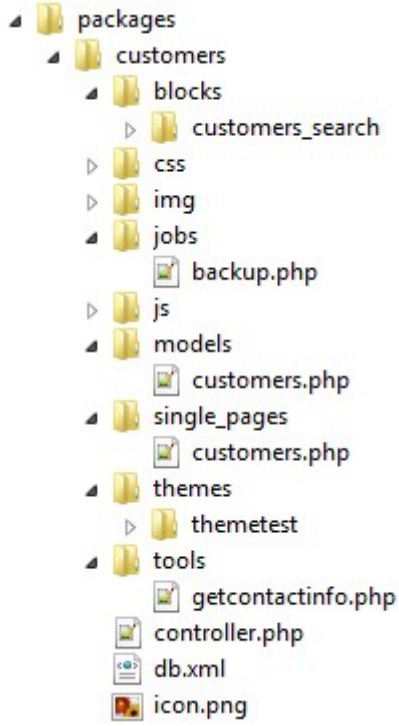
6.2.8 Työkalut

Työkalut ovat yksittäisiä tiedostoja, joiden avulla voidaan toteuttaa pieniä aputoimintoja. Työkalujen avulla on helppoa toteuttaa esimerkiksi datan noutaminen ja tallentaminen. Työkaluilla on käytössä kaikki Concrete5-järjestelmän osat, kuten esimerkiksi mallit. Työkalut sijaitsevat hakemistossa ”tools”.

6.2.9 Paketit

Paketit ovat ohjelmakokonaisuuksia, jotka voivat sisältää erilaisia juurihakemiston mukaisia hakemistoja lisäämään tai ylikirjoittamaan toimintoja. Paketti voi koostua esimerkiksi lohkotyypeistä, teemoista, työkaluista, ohjaimista, malleista sekä näkymistä. Jokaisessa paketissa täytyy olla ohjaintiedosto ”controller.php”, joka sisältää oman versionumeronsa sekä määritykset eri osien asentamiseen. Paketti voi sisältää tietokantamäärityksen, jonka mukainen rakenne toteutetaan tietokantaan asennuksen ja päivityksen yhteydessä. Pakettien avulla on helppo siirtää ja asentaa edellä mainittuja järjestelmän osia toiseen Concrete5-asennukseen.

Paketit sijaitsevat hakemistossa ”packages”, ja yhden paketin rakenne voi olla esimerkiksi kuvion 19 mukainen. Esimerkkipaketin nimi on ”customers”, joka sisältää ohjaimen, tietokantamäärityksen, ”customers_search”-lohkotyypin, ”backup”-tehtävän, ”customers”-mallin, ”customers”-näkymän, ”getcontactinfo”-työkalun sekä ”themetest”-teeman. Lisäksi paketti sisältää CSS-tyylitiedostoja sekä JavaScript- ja kuvatiedostoja.



KUVIO 19. Concrete5-sisällönhallintajärjestelmän paketin rakenne

Kuviossa 20 on esitetty esimerkkipaketin ohjaintiedoston lähdekoodi. Paketin ohjaintiedosto koostuu ”CustomersPackage”-luokasta, joka on periytetty Concrete5:n sisältämästä ”Package”-luokasta. Rivillä 8 on paketin luojan määrittämä Concrete5:n pienin ohjelmistoversionumero, jonka paketti vaatii asentuakseen järjestelmään. Rivillä 9 on määritetty paketin versionumero, jota muuttamalla paketin pystyy päivittämään hallintapaneelin kautta. Pakettiluokan ”install”-metodi (rivit 19-32) suoritetaan paketin asennuksen yhteydessä ja ”upgrade”-metodi (rivit 34-36) päivityksen yhteydessä. Paketin sisältämä näkymä halutaan asentaa automaattisesti paketin asennuksen yhteydessä, joten sen vaatima asennuslogiikka täytyy sisällyttää ”install”-metodiin. Rivillä 22 ladataan Concrete5:n sisäinen malli ”single_page”, rivillä 25 määritetään uuden näkymän osoite ja riveillä 26-27 lisätään näkymälle nimi sekä kuvaus. Tehtävän asennusta varten ladataan ”job”-malli rivillä 23 ja asennetaan tehtävä rivillä 30. Rivillä 29 asennetaan lohkotyyppi ”customers_search” ja rivillä 31 teema.

```php
<?php

defined('C5_EXECUTE') or die(_('Access Denied.'));

class CustomersPackage extends Package {

    protected $pkgHandle = 'customers';
    protected $appVersionRequired = '5.6.0.2';
    protected $pkgVersion = '1.0.0';

    public function getPackageDescription() {
        return t('Show customers');
    }

    public function getPackageName() {
        return t('Customers');
    }

    public function install() {
        $pkg = parent::install();

        Loader::model('single_page');
        Loader::model('job');

        $pCustomers = SinglePage::add('/customers', $pkg);
        $pCustomers->update(array('cName' => t('Customers'),
                                  'cDescription' => t('Show customers')));

        BlockType::installBlockTypeFromPackage('customers_search', $pkg);
        Job::installByPackage('backup', $pkg);
        PageTheme::add('themetest', $pkg);
    }

    public function upgrade() {
        parent::upgrade();
    }

}
```

KUVIO 20. Concrete5-sisällönhallintajärjestelmän paketin ohjaintiedosto

Paketissa oleva näkymä noutaa ”customers”-mallilta asiakkaita, joiden tietokannan taulu on määritetty ”db.xml”-tiedostossa. Näkymässä listattuja asiakkaita klikkaamalla kutsutaan työkalua ”getcontactinfo” ja välitetään kyseisen asiakkaan tunniste. Työkalu käyttää ”customers”-mallia noutaakseen asiakkaan yhteystiedot tunnisteen perusteella ja palauttaa hakutulokset. Lohkotyypin ”customers_search” avulla sivustolla vieraileva käyttäjä voi etsiä asiakkaita. Lohkotyyppi on lisätty sivustolla olevaan lohkoalueeseen ja se käyttää ”customers”-mallia asiakkaiden hakemiseen. Tehtävä ”backup” suorittaa asiakastietoja sisältävän tietokannan taulun varmuuskopioinnin päivittäin ottamalla yhteyden ulkopuoliseen palvelimeen ja lähettämällä asiakastiedot sinne.

7 WWW-PALVELUN TOTEUTUS

7.1 Vaatimusten kartoitus

Sääpalvelu on www-pohjainen palvelu, joka toimii julkisena käyttöliittymänä havaintoasemilla vallitsevan sään sekä historiatietojen tarkastelua varten. Palvelun alustaksi valittiin Concrete5-sisällönhallintajärjestelmä, joka tarjoaa MVC-arkkitehtuurin mukaisen rakenteen sekä erilaisten komponenttien helpon toteuttamisen. Palvelussa käyttäjät voivat valita havaintoaseman sekä aikavälin, jolta havaintotietoja haetaan. Havaintotiedot esitetään kaaviona, jossa viivadiagrammina esitetään lämpötila, ilmanpaine ja ilmankosteus sekä palkkidiagrammina sademäärä. Viivadiagrammin tulee osata lämpötilan tapauksessa näyttää myös lämpötilan vaihtelut mittausjaksojen ajalta ja esittää se täyttökaaviona. Kaavion toteutukseen valittiin Flot-JavaScript-kaaviokirjasto, joka tarjoaa monipuolisia ominaisuuksia kaavioiden esittämiseen. Kirjautuneiden käyttäjien tulee pystyä lisäämään, poistamaan ja muokkaamaan havaintoasemia hallintapaneelissa. Palvelun rajapinnaksi implementoidaan REST-arkkitehtuurityylin rajoitteiden mukainen rajapinta.

7.2 Esivalmistelut

Concrete5-sisällönhallintajärjestelmää käytettäessä on suositeltavaa kehittää omat lohkotyypit ym. järjestelmän komponentit pakettina. Paketti auttaa kehittäjiä huomioimaan asennettavat komponentit, kuten esimerkiksi yksittäiset sivut, kun ne on listattu paketin hakemistossa sijaitsevassa ohjaintiedostossa. Paketti mahdollistaa myös kaikkien komponenttien siirtämisen helposti uuteen järjestelmään.

Sääpalvelun pakettia varten luodaan järjestelmän ”packages”-hakemistoon ”omasaa”-hakemisto, johon luodaan alihakemistot lohkotyypeille, ohjaimille, malleille sekä yksittäisille sivuille. Pakettiin luodaan myös kuvake ja ohjaintiedosto, joka sisältää paketille kuuluvat määritykset.

Havaintotiedot sisältävä tietokanta käsittää taulut havaintoasemille sekä -tiedoille, joten erillistä tietokantarakennetta paketille ei tarvita.

7.3 Mallien toteutus

Tietokanta käsittää taulut havaintoasemille sekä -tiedoille, joille molemmille toteutetaan omat mallit. Mallitiedostoille tyypillinen rakenne on PHP-luokka, joka sisältää vapaavalintaisesti määritettäviä metodeja tietokantakyselyiden toteuttamiseksi. Mallit luodaan paketin hakemistoon ”models”.

Kuviossa 21 on havainnollistettu havaintoasemien mallitiedoston sisältö. Havaintoasemien mallilla tulee pystyä noutamaan, lisäämään, muokkaamaan sekä poistamaan asemia, joten tarvittavia metodeja mallitiedostoon on vähintään kolme kappaletta. Havaintoaseman lisääminen ja muokkaaminen voidaan toteuttaa samalla metodilla (rivi 13), sillä varsinaisessa tietokantakyselyssä aseman tiedot lisätään uutena asemana, ellei tallennettavaa asemaa ole entuudestaan olemassa. Havaintoaseman muokkaamisen tapauksessa aseman tiedot täytyy noutaa muokkauslomaketta varten, joten tarvitaan vielä yksi metodi yksittäisen aseman tietojen noutamiseen sen tunnisteen perusteella (rivi 11). Mallin ”getAll”-metodi sisältää esimerkinomaisesti tietokantayhteyksien lataamisen (rivi 7), tietokantakyselyn kaikkien asemien noutamiseen (rivi 8) sekä tulosten palauttamisen metodia kutsuneelle komponentille (rivi 9).

```php
<?php

class WxStationsModel {

    public function getAll() {
        $db = Loader::db($host, $username, $password, $tablename, true);
        $data = $db->getAll("SELECT * FROM Location");
        return $data;
    }

    public function getByID($id) {}

    public function save($id, $location, $lat, $lng) {}

    public function delete($id) {}

}
```

KUVIO 21. Havaintoasemien käsittelyyn toteutettu malli

Havaintotietojen malli puolestaan käsittää ainoastaan havaintotietojen noutamisen tietyltä asemalta tietyllä aikavälillä, sillä havaintotietojen muunlainen käsittely tapahtuu niitä ylläpitävän Windows-palvelun toimesta.

7.4 REST-rajapinnan implementointi

REST-rajapinta voidaan toteuttaa Concrete5-järjestelmään monella eri tavalla, esimerkiksi käyttämällä valmista ohjelmistokehystä REST-toiminnallisuuksien käyttämiseen, mutta MVC-arkkitehtuuria hyödyntävä rajapinta voidaan implementoida myös suoraan Concrete5-järjestelmään.

7.4.1 Yksittäinen sivu

Yksinkertaisin tapa rajapinnan toteuttamiseen on käyttää yksittäistä sivua, joka voidaan nimetä rajapinnan tapauksessa nimellä ”api”. Yksittäisen sivun käyttöönottoa varten täytyy luoda ”single_pages”-hakemistoon tiedosto ”api.php”. Samalla lisätään paketin ohjaintiedoston ”install”-metodiin yksittäisen sivun asennusproseduuri, joka on esitetty kuviossa 22. Asennusproseduuri käsittää tarkistuksen, onko yksittäinen sivu jo olemassa järjestelmässä.

```
22          Loader::model('single_page');
23          $pAPI = Page::getByPath('/api');
24          if (!is_object($pAPI) || !intval($pAPI->getCollectionID())) {
25              $pAPI = SinglePage::add('/api', $pkg);
26          }
27          if (is_object($pAPI) && intval($pAPI->getCollectionID())) {
28              $pAPI->update(array('cName' => t('API'),
29                                  'cDescription' => t("RESTful API for OmaSää")));
30          }else {
31              throw new Exception(t('Error: /api page not created'));
32          }
```

KUVIO 22. Yksittäisen sivun asennusproseduuri paketin ohjaintiedostossa

7.4.2 Ohjaintiedostot

Yksittäisellä sivulla olevat toiminnot suoritetaan siinä ohjaimessa, joka luodaan paketin hakemistoon ”controllers/api” nimellä ”controller.php”. Kaikki käyttäjien tekemät rajapintakyselyt ohjautuvat tähän tiedostoon. REST-arkkitehtuurityylin mukaisesti rajapinnan tulee hyödyntää täsmällisesti HTTP-protokollan mukaisia metodeja tiedon noutamiseen käyttäen GET-metodia, tiedon tallentamiseen käyttäen PUT-metodia sekä tiedon poistamiseen käyttäen DELETE-metodia. POST-metodi tiedon lisäämistä varten ei ole välttämätön, sillä PUT-metodilla voidaan sekä lisätä että muuttaa tietoja. PHP-ohjelmointikielen avulla on helppo saada selville kyselyssä käytetty HTTP-metodi, sillä kyselyyn liittyviä tietoja on saatavilla ”$_SERVER”-taulukosta. Taulukosta selviää tietoja mm. käyttäjän selaimesta sekä palvelimen asetuksista. Käytettävä HTTP-metodi löytyy elementistä ”REQUEST_METHOD”, jonka arvo on talletettu isoin kirjaimin. Kuvio 23 esittää rajapinnan ohjaintiedostoa, joka sisältää metodit havaintotietojen (wx) sekä -asemien (stations) käsittelyyn.

```php
<?php defined('C5_EXECUTE') or die(_("Access Denied."));

class ApiController extends Controller {

    private $type = '';

    public function __construct() {
        $this->type = $_SERVER['REQUEST_METHOD'];
    }

    public function __call($method, $args) {
        if(method_exists($this, $method)){
            return call_user_func_array(array($this, $method), $args);
        }else{
            die('Action not allowed');
        }
    }

    public function wx() {
        $this->{'wx' . $this->type}(func_get_args());
    }
    public function wxGET() {}

    public function stations() {
        $this->{'stations' . $this->type}(func_get_args());
    }
    public function stationsGET() {}
    public function stationsPOST() {}
    public function stationsPUT() {}
    public function stationsDELETE() {}

}
```

KUVIO 23. REST-rajapinnan ohjaintiedosto

Ohjaintiedoston alustuksen yhteydessä noudetaan kyselyssä käytettävä HTTP-metodi (rivi 8) ja tallennetaan se ohjaimen attribuutiksi. Kyselystä riippuen kutsutaan joko ”wx”- tai ”stations”-metodia, joiden sisällä kutsutaan erillisiä HTTP-metodin mukaisia metodeja. Mahdollinen GET-tyyppinen kysely rajapinnalle voi olla esimerkiksi kuvion 24 mukainen.

```
GET /api/wx/abc/def HTTP/1.1
```

KUVIO 24. GET-tyyppinen kysely

Kysely on GET-tyyppinen, ja se koostuu yksittäisestä sivusta ”api”, jolloin kysely ohjautuu REST-rajapinnan ohjaintiedostolle. Kyselyssä seuraavaksi on kutsuttavan metodin nimi ”wx”, jonka jälkeen kaksi parametria: ”abc” ja ”def”. Kutsuttu metodi kutsuu toista metodia ”wxGET”, joka muodostuu alun perin kutsutun metodin nimestä sekä käytettävästä HTTP-metodista.

Metodille välitetään myös kaikki kyselyssä annetut parametrit käyttäen PHP:n funktiota ”func_get_args” (rivi 20). Kutsu ohjautuu PHP:n sisäiseen nk. ”maagiseen metodiin” (Magic method) ”__call” (rivi 11), joka suoritetaan automaattisesti jokaisen PHP-luokan metodin kutsumisen yhteydessä. Kyseinen metodi tarkistaa, onko kutsuttu metodi olemassa PHP-luokassa, ja palauttaa virheilmoituksen, mikäli metodia ei ole olemassa (rivi 15). Olemassa olevaa metodia kutsutaan käyttämällä PHP:n funktiota ”call_user_func_array”, johon määritellään kutsuttavan metodin lisäksi sen luokka ja parametrit (rivi 13). Metodi ”wxGET” suorittaa REST-arkkitehtuurityylin mukaisesti historiatietojen noutamisen tietokannasta.

Kutsuttavat metodit ottaa vastaan osoiteriville kirjoitettuja arvoja ja suorittavat HTTP-metodia vastaavia toimintoja. Metodit käyttävät malleja suorittamaan tietokantakyselyitä, ja ne palauttavat tulokset JSON-tiedonsiirtomuodossa. Tyypillinen JSON-muotoinen vastaus voi olla esimerkiksi kuvion 25 mukainen.

```
1  {"status":1,
2   "rows":[{
3      "timestamp":"1354843802",
4      "temp_min":"1.2",
5      "temp_max":"1.2",
6      "temp_avg":"1.2",
7      "humid_avg":"79",
8      "pres_avg":"1010.7"
9   }]
10 }
```

KUVIO 25. JSON-muotoinen datakatkelma

7.4.3 Yhteenveto

REST-rajapinta käsittää keskitetyn rajapinnan havaintotietojen noutamiseen sekä havaintoasemien käsittelyyn. Tämän työn tapauksessa havaintotietojen käsittelyssä käytetään ainoastaan GET-metodia, sillä niiden lisääminen, muokkaaminen ja poistaminen tapahtuu havaintotietoja vastaanottavan Windows palvelun toimesta. Havaintoasemien käsittely tapahtuu

kokonaisuudessaan REST-rajapinnan kautta. Tietojen muokkaamisen ja poistamisen tapauksessa selvitetään, että käyttäjä on kirjautuneena järjestelmään ennen toimenpiteiden suorittamista. Taulukko 5 esittää REST-rajapinnan sisältämät resurssit sekä niissä käytettävät HTTP-metodit.

TAULUKKO 5. REST-rajapinnan resurssit

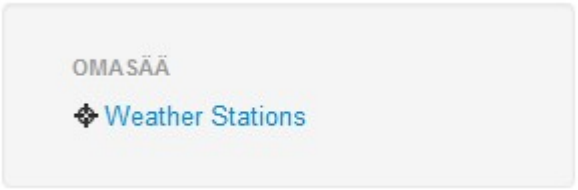
Resurssi	**Metodi**	**Kuvaus**
/api/wx/{stationID}/{timespan}	GET	Havaintotietojen haku tietyltä asemalta tietyllä aikavälillä
/api/stations	GET	Havaintoasemien noutaminen
/api/stations/{stationID}	GET	Havaintoaseman tietojen noutaminen
/api/stations/{location}/{lat}/{lng}	POST	Havaintoaseman lisääminen määrittäen sen sijainnin sekä koordinaatit
/api/stations/{location}/{lat}/{lng}	PUT	Havaintoaseman sijainnin sekä koordinaattien päivittäminen
/api/stations/{stationID}	DELETE	Havaintoaseman poistaminen

7.5 Havaintoasemien hallinta hallintapaneelissa

Concrete5-sisällönhallintajärjestelmä mahdollistaa ominaisuuksien laajentamisen myös hallintapaneeliin, johon voidaan rakentaa järjestelmän toimintaan liittyviä hallintakomponentteja. Concrete5 sisältää valmiiksi hallintakomponentteja mm. tiedostojen, käyttäjien sekä lohkotyyppien hallintaan. Hallintakomponenteilla on mahdollista käyttää lähes kaikkia Concrete5:n ominaisuuksia, sillä myös niiden kohdalla kaikki järjestelmän komponentit ovat ladattavissa ja käytettävissä. Tyypillisesti hallintakomponentit sisältävät tietokantoja käyttäviä asetuksia eri järjestelmän osista. Sääpalvelun ylläpitäjän tulee pystyä hallitsemaan havaintoasemia järjestelmässä, joten sen toteutus hallintakomponenttina tukee järjestelmän arkkitehtuuria.

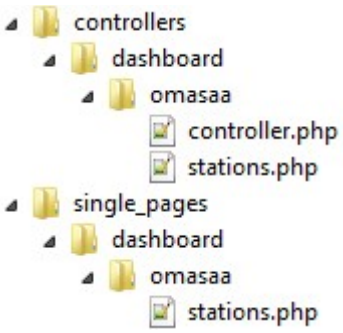
Hallintapaneelin laajentaminen tapahtuu käyttäen yksittäisiä sivuja, jotka sijaitsevat hakemistossa ”single_pages/dashboard/”. Oman hallintakomponentin luomiseen täytyy em. hakemistoon luoda uusi hakemisto ”omasaa”, joka sisältää kaikki komponentin yksittäiset sivut. Paketin ohjaintiedostoon täytyy lisätä hallintakomponentin yksittäisten

sivujen asennusproseduurit, jotka määrittelevät myös nimen, jolla ne näkyvät hallintapaneelissa. Ohjaintiedoston ”install”-metodiin lisätään yksittäinen sivu, jonka osoite on ”/dashboard/omasaa” ja nimi ”OmaSää”. Tämä toimii hallintakomponentit pääsivuna ja otsikkona, joka käytännössä ohjaa jollekin alisivulle. Lisätään myös havaintoasemien alisivuna toimiva yksittäinen sivu ohjaintiedostoon osoitteella ”/dashboard/stations” ja nimellä ”Weather Stations”. Kuvio 26 havainnollistaa näkymää hallintapaneelissa, kun hallintakomponentin sivut on asennettuna järjestelmään. Kuviossa näkyvä ikoni on asetettu yksittäisen sivun asennusproseduurissa.



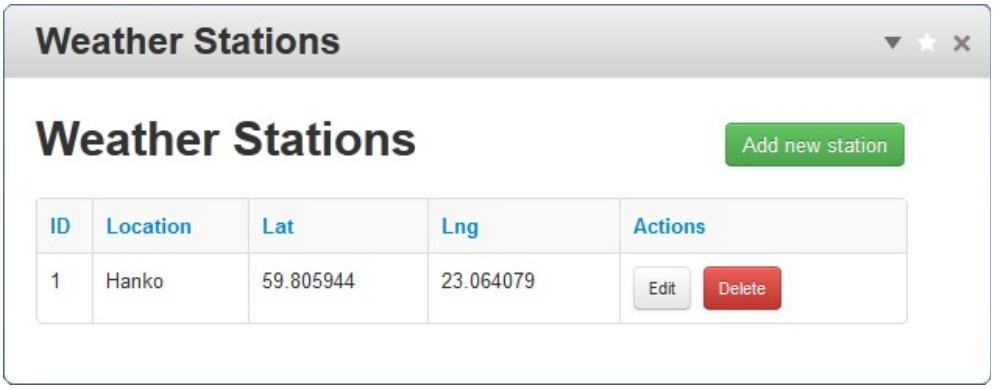
KUVIO 26. Sääpalvelun hallintaosio Concrete5-järjestelmän hallintapaneelissa

Hallintakomponentin toiminnallisuuden luomiseksi täytyy luoda myös ohjaintiedostot yksittäisten sivujen toimintaan liittyen. Ne luodaan paketin hakemistoon ”controllers/dashboard/omasaa/”. Kuvio 27 esittää paketin hakemistorakennetta ohjaimien ja yksittäisten sivujen osalta. Ohjaintiedosto ”controller.php” ohjaa osoitteeseen ”/dashboard/omasaa/” tulevan käyttäjän osoitteeseen ”/dashboard/omasaa/stations/”, jossa ohjaintiedosto ”stations.php” vastaa kyselyihin.



KUVIO 27. Hallintakomponentin rakenne

Havaintoasemien hallintasivulle saavuttaessa ohjaintiedostossa ”stations.php” ladataan ”stations”-malli ja kutsutaan sen metodia ”getStations”. Kyselyn tulokset ohjataan muuttujana yksittäiselle sivulle ”stations.php”, jossa tulostetaan kuvion 28 mukainen näkymä. Yksittäisellä sivulla tulostetaan taulukko havaintoasemista sekä painikkeet niiden lisäämiseen, muokkaamiseen ja poistamiseen.



KUVIO 28. Havaintoasemien hallintaosio

Tietojen lisäämiseen ja muokkaamiseen käytetään tavallista HTML-kuvauskielen mukaista lomaketta, jonka tallennus toteutetaan AJAX-kutsuilla REST-rajapintaan. jQuery-JavaScript-kirjastoon kuuluvan AJAX-metodin käyttöä on havainnollistettu kuviossa 29, jossa muuttujiin sijoitetaan lomakkeen kenttien arvot (rivit 1-3) ja määritetään AJAX-kutsun parametrit (rivit 6 ja 7). Parametreina annettava ”type” määrittää kyselyssä käytettävän HTTP-metodin, joka havaintoaseman lisäämisen tapauksessa on ”POST”. Toisena parametrina määritetään osoite ”url”, johon kysely tehdään. Osoitteessa määritellään rajapinnan mukaiset arvot, joita tässä tapauksessa on havaintoaseman sijainti arvona ”location” sekä sen koordinaatit arvoina ”lat” ja ”lng”. Esimerkin omaisesti osoitteesta on jätetty sen alkuosa pois. AJAX-kyselyn toteutuessa suoritetaan ”done”-metodi (rivi 8), joka saa parametrikseen rajapinnalta tulevan vastauksen. Mikäli rajapinta on toteuttanut kyselyn toiminnot onnistuneesti, palautuu siitä statusviesti, jonka mukaisesti näytetään viesti tallentamisesta (rivi 10). Virhetilanteissa näytetään virheilmoitus, joka sisältää myös palvelimella tapahtuneen virheen (rivi 12).

```
1  var location = $('input[name="location"]').val();
2  var lat = $('input[name="lat"]').val();
3  var lng = $('input[name="lng"]').val();
4
5  $.ajax({
6      type: "POST",
7      url: "/api/stations/"+location+"/"+lat+"/"+lng
8  }).done(function( reply ) {
9      if(reply.status == 1){
10         msg('Saved');
11     }else{
12         msg('Something went wrong: '+reply.errmsg);
13     }
14 });
```

KUVIO 29. Havaintoaseman lisääminen jQueryn AJAX-metodilla

7.6 Havaintotietoja esittävän komponentin toteutus

Havaintotietoja tarkastellaan www-palvelussa kaaviona, jonka havaintotiedot voidaan noutaa määrittelemällä haluttu havaintoasema sekä tietty aikaväli. Havaintotietojen tarkasteluun toteutettavan työkalun tulee sääpalvelun tavoitteiden mukaisesti olla näyttävä ja toimiva ollen samalla myös helppokäyttöinen. Työkalu toteutetaan Concrete5-järjestelmään lohkotyyppinä, joka hyödyntää avoimen lähdekoodin Flot-JavaScript-kirjastoa kaavion toteuttamiseksi.

7.6.1 Flot-JavaScript-kaaviokirjasto

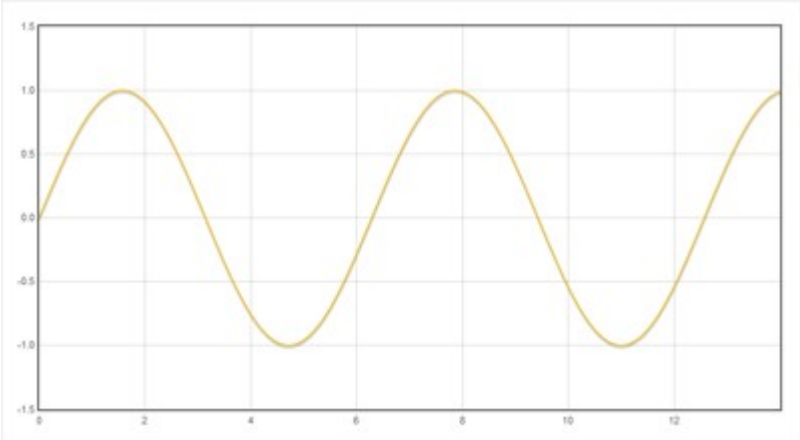
Flot-kaaviokirjasto on tanskalaisen Ole Laursenin kehittämä liitännäinen jQuery-JavaScript-kirjastoon, jonka avulla interaktiivisten ja monipuolisten kaavioiden tekeminen www-sivustolle on helppoa (IOLA & Ole Laursen 2013). Kirjaston avulla on mahdollista esittää mm. viiva-, palkki- ja piirakkadiagrammeja sekä ladata niiden tietoja asynkronisesti AJAX-metodin avulla. Flot sisältää myös monia lisäominaisuuksia sen perustoiminnallisuuden laajentamiseksi.

Kuvio 30 esittää yksinkertaista JavaScript-ohjelmakoodia, jossa datataulukkoon generoidaan sinikäyrän pisteitä suhteessa x-akseliin (rivit 4-

6). Datataulukkoon lisättävät tiedot koostuvat x- ja y-arvoista. Kaavion luontia varten HTML-dokumentissa on oltava DIV-elementti, johon Flot-kaaviokirjasto generoi asetusten mukaisen kaavion. Elementin tunniste sekä datataulukko määritellään ”plot”-objektin parametreina rivillä 8. Kolmantena parametrina ”plot”-objektille voidaan määritellä erilaisia asetuksia esimerkiksi väreistä sekä interaktiivisista toiminnoista. Kuvio 31 havainnollistaa kuvion 30 ohjelmakoodin tuottaman kaavion.

```
1   $(function() {
2
3       var d1 = [];
4       for (var i = 0; i < 14; i += 0.1) {
5           d1.push([i, Math.sin(i)]);
6       }
7
8       $.plot("#placeholder", [ d1 ]);
9
10  });
```

KUVIO 30. Yksinkertaisen kaavion luominen Flot-kaaviokirjastolla



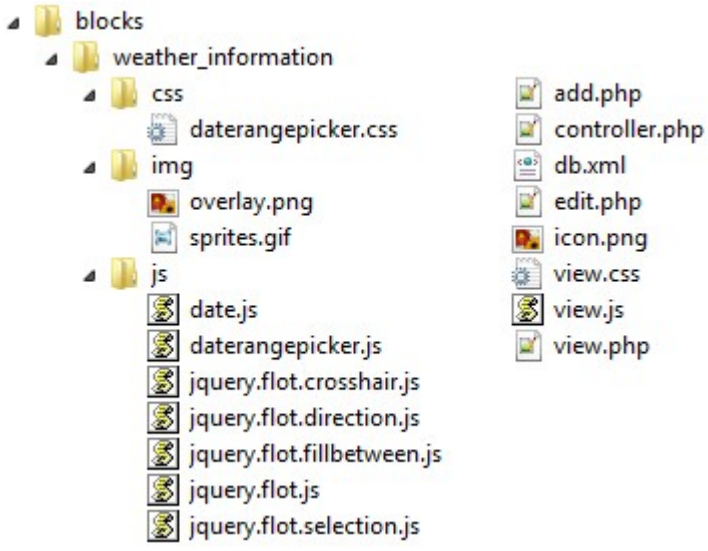
KUVIO 31. Flot-kaaviokirjastolla luotu yksinkertainen viivadiagrammi

Havaintotietoja esittävään kaavioon on suunniteltu mm. lämpötilatietojen minimi- ja maksimiarvojen esittämistä täyttökaaviona, joka käytännössä tarkoittaa sekä minimin että maksimin viivadiagrammia, jonka välinen alue on täytetty jollakin määritettävällä värillä. Flot sisältää ”fillbetween”-liitännäisen, jolla em. toiminto voidaan toteuttaa. Kaavioon halutaan toteuttaa myös tarkennustoiminto, jonka avulla käyttäjä voi valita alueen, jota tarkastella. Käytännössä se sisältää kaksi kaaviota, joista alempi esittää

koko aikavälin havaintoja, josta valitsemalla haluttujen rajojen väliset tiedot päivittyvät ylempään kaavioon. Flot-kirjastossa tällainen toiminnallisuus saavutetaan ”selection”-liitännäisellä. Muita ominaisuuksia, kuten hiiren osoittimella tarkasteltavat arvot x-akselin perusteella toteutetaan ”crosshair”-liitännäisellä.

7.6.2 Lohkotyypin toteutus

Concrete5-järjestelmän sivuilla esitettävä kaaviotyökalu on toiminnallisena komponenttina toteutettava lohkotyyppinä, joka koostuu ohjaimesta, näkymästä, tietokantarakenteesta ja erinäisistä CSS-tyyleistä sekä JavaScript-koodeista. Lohkotyyppi sisältää kaikki sen tarvitsemat tiedostot, kuten Flot-kaaviokirjaston liitännäisineen sekä tarvittavat tyylit. Kuvio 32 esittää lohkotyypin hakemistorakennetta, jossa on esitetty kaikki kaaviotyökalun tarvitsemat tiedostot. Flot:n JavaScript-tiedostot ja aikavälin valinnassa käytettävä JavaScript-kirjasto sijaitsevat hakemistossa ”js”. Tarvittavat tyylit sijaitsevat hakemistossa ”css” ja kuvat mm. latausanimaatiota varten hakemistossa ”img”. Lohkotyypin tietokantarakenne käsittää ainoastaan kokonaislukutyyppisen tunnisteen ”btWeatherInformationBlock”-taulussa lisätyille lohkotyypeille.



KUVIO 32. Havaintotietoja esittävän lohkotyypin hakemistorakenne

Lohkotyypin ”view.php”-tiedosto sisältää yksinkertaisen HTML-kuvauskielisen rakenteen lomakkeelle, jossa valitaan haluttu havaintoasema ja aikaväli. Se sisältää myös tulostettavien kaavioiden sijainnit sekä JavaScript-koodin latausanimaatiota varten. ”view.js”-tiedosto sisällytetään lohkotyypin esittämälle www-sivulle automaattisesti. Se sisältää ohjelmakoodin kaavion piirtämiseen sivustolle, lomakkeen uusien havaintotietojen noutamiseen sekä monia sen apufunktioita. Eräs käytettävä apufunktio värittää viikonloppua ilmaisevat päivät harmaalla värillä.

Kuviossa 33 on esitetty osa havaintotietoja esittävän lohkotyypin ohjaintiedostosta, jossa on määritelty tietokannan taulun nimi (rivi 7) sekä ”on_page_view”-metodi (rivi 11), joka suoritetaan aina kun lohkotyyppi esitetään sivulla. Metodi sisältää kaikkien lohkotyypin käyttämien CSS- sekä JavaScript-tiedostojen lisäämisen www-sivun ”head”-osioon.

```
1  <?php
2
3  class WeatherInformationBlockController extends BlockController {
4
5      protected $btDescription = "This module generates Flot charts.";
6      protected $btName = "WeatherInformationBlock";
7      protected $btTable = 'btWeatherInformationBlock';
8      protected $btInterfaceWidth = "350";
9      protected $btInterfaceHeight = "300";
10
11     public function on_page_view() {
12         $html = Loader::helper('html');
13         $this->addHeaderItem($html->css(
14             '../blocks/weather_information/css/daterangepicker.css',
15             'omasaa'));
16         $this->addHeaderItem($html->javascript(
17             '../blocks/weather_information/js/date.js',
18             'omasaa'));
19         ...
20     }
21 }
```

KUVIO 33. Havaintotietoja esittävän lohkotyypin ohjain

Lohkotyypin ”view.js”-tiedosto sisältää kaiken kaavion luomiseen liittyvät toiminnot, kuten myös havaintotietojen noutamisen rajapinnalta AJAX-kyselynä. Kuvio 34 esittää ”view.js”-tiedoston sisältämän ”getData”-funktion ohjelmakoodin, jossa on sen lyhentämisen vuoksi toteutettu

ainoastaan keskilämpötilan käsittely. Lohkotyypin lomakkeessa käyttäjä pystyy määrittelemään havaintoaseman sekä aikavälin, jolta havaintotietoja noudetaan. Nämä tiedot välitetään ”getData”-funktion parametreina rivillä 1. Vastaanotettavat tiedot tallennetaan taulukkoon, joka alustetaan rivillä 3. AJAX-metodi toteuttaa asetusten mukaisen kyselyn rajapintaan (rivit 5-7) ja sijoittaa palvelimelta tulevan datan ”data”-muuttujaan (rivi 8). Palvelimen palauttaman statusviestin ollessa kunnossa luetaan jokainen havaintotietorivi yksi kerrallaan (rivi 11). Taulukkoon ”tempAvg” tallennetaan havaintotiedon aikaleima x-akselin arvoksi (rivi 12). Aikaleima tulee kertoa tuhannella, sillä JavaScript-ohjelmakoodi käsittelee aikaleimoja sekuntien sijasta millisekunteina. Y-akselin arvoksi tallennetaan palvelimen palauttama arvo keskilämpötilasta. Kun kaikki rivit on luettu, määritetään ”dataset”-datataulukon sisällöksi sarja, joka koostuu sen nimestä (rivi 16), datasta (rivi 17), viivan asetuksista (rivit 18-22) sekä viivan väristä (rivi 23).

```
1  function getData(station, timespan){
2
3      var tempAvg = new Array();
4
5      $.ajax({
6          url: '/api/wx/'+station+'/'+timespan,
7          type: "GET",
8          success: function(data){
9              if(data.status != 0){
10
11                 $(data.rows).each(function(i,item){
12                     tempAvg.push([(item.timestamp * 1000), item.temp_avg]);
13
14                     if(!--count){
15                         dataset = [{
16                             label: 'Lämpötila &deg;C',
17                             data: tempAvg,
18                             lines: {
19                                 show:true,
20                                 lineWidth: 0.5,
21                                 shadowSize:0
22                             },
23                             color:"rgb(255, 90, 90)"
24                         }]
25
26                         drawFlot();
27                     }
28                 });
29             }else{
30                 // palvelin palautti virheen
31             }
32         }
33     });
34 }
```

KUVIO 34. Keskilämpötilan nouto AJAX-kyselynä sekä sen tallennus datataulukkoon

Kuvion 34 rivillä 26 kutsutaan ”drawFlot”-funktiota, kun kaikkien havaintotietojen suureet on talletettu datataulukkoon. Kuvio 35 esittää ”drawFlot”-funktiota, jossa luodaan pääkaavio datataulukon tiedoista (rivi 3). Tärkeimpinä asetuksina määritellään x-akselin tyypiksi ”time” (rivi 5), jotta talletetut aikaleimat näkyvät kaaviossa päivämäärinä sekä valintaliitännäisen tyypiksi ”x” (rivi 7), jotta kaaviossa olevia havaintotietoja voidaan skaalata x-akselin perusteella. Lisäksi määritellään kaavion taustaruudukon asetukset mm. kutsumalla funktiota ”weekendAreas” (rivi 8), joka värittää viikonloppua ilmaisevat päivät harmaalla värillä.

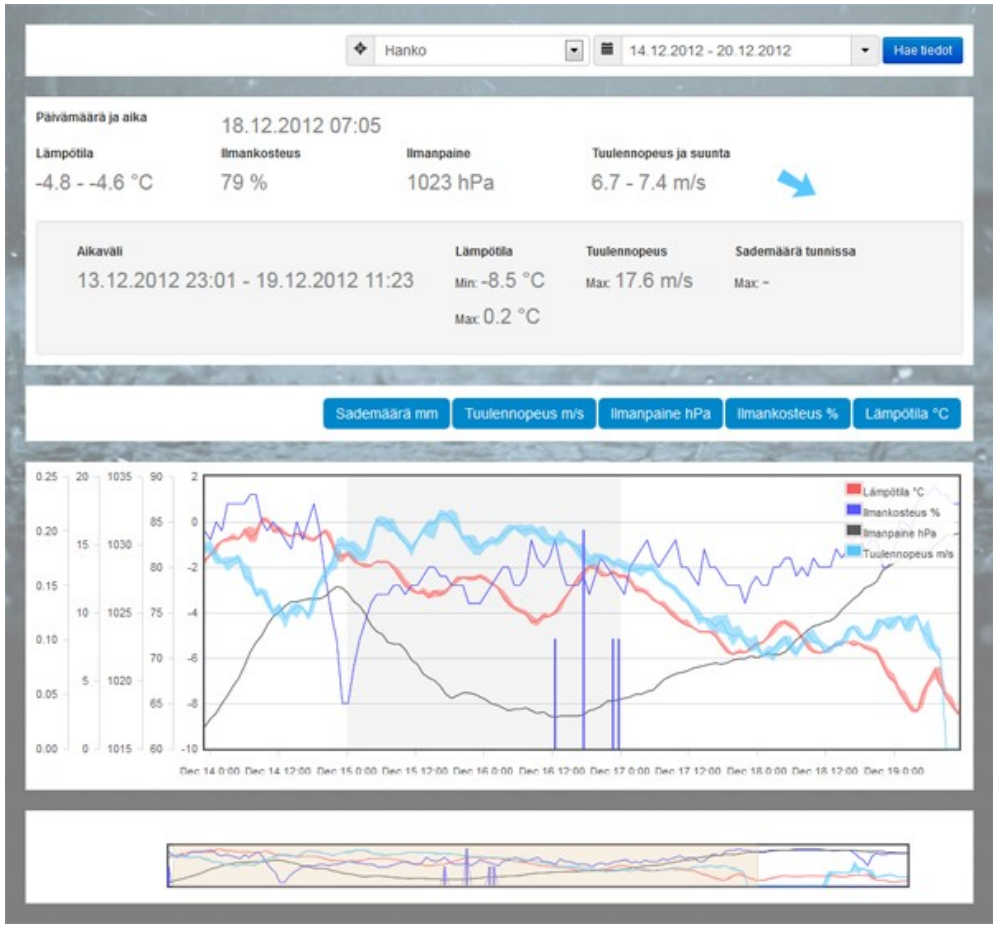
Sääpalvelua varten käytettävä Flot-kaaviokirjaston ”selection”-liitännäinen mahdollistaa koko kaavioalueen esittämisen pienemmässä kaaviossa. Pienempää kaaviota varten luodaan toinen kaavio saman datataulukon tietojen perusteella (rivi 11). Sen asetukset ovat pääosin samat kuin pääkaaviollakin sillä erolla, että viivan koko on asetettu hieman pienemmäksi (rivi 12). Lopuksi kutsutaan ”axesInteraction”-funktiota (rivi 19), joka lisää kaavion akseleille interaktiivisia toimintoja.

```
1  function drawFlot(){
2
3      plot = $.plot( $("#placeholder"), dataset, {
4          series: { shadowSize: 0 },
5          xaxis: { mode: "time", tickLength: 5 },
6          crosshair: { mode: "x", color:'#aaa' },
7          selection: { mode: "x"},
8          grid: { markings: weekendAreas, hoverable: true, autoHighlight: false }
9      });
10
11     overview = $.plot( $("#overview"), dataset, {
12         series: { lines: { show: true, lineWidth: 0.5 }, shadowSize: 0 },
13         xaxis: { ticks: [], mode: "time" },
14         yaxis: { ticks: [], autoscaleMargin: 0.1 },
15         selection: { mode: "x" },
16         legend: { show: false }
17     });
18
19     axesInteraction();
20
21 }
```

KUVIO 35. Flot-kaavioiden asetusten määritys

Sääpalvelun havaintotietoja esittävä lohkotyyppi on esitetty kuviossa 36. Lohkotyypin ylin osio koostuu lomakkeesta, jossa pudotusvalikosta voi valita halutun havaintoaseman sekä aikavälikenttä, josta aukeaa

kalenterinäkymä aikavälin valitsemiseksi. Lisäksi se sisältää painikkeen, jonka klikkaus suorittaa havaintotietojen noutamisen palvelimelta. Seuraavana alueena lohkotyypissä on havaintotietojen tarkat arvot, jotka päivittyvät keskellä olevaa pääkaaviota tarkastelemalla. Seuraavana alueena on painikkeet eri suureiden näyttämiseen ja piilottamiseen. Pääkaaviossa on esitettynä lämpötila punaisella, ilmankosteus sinisellä, ilmanpaine harmaalla viivalla sekä tuulen nopeus vaaleansinisellä viivalla. Lisäksi kaaviossa on sinisellä värillä palkkikaavio, joka havainnollistaa sademäärää edeltäneeltä tunnilta. Kaaviosta on nähtävissä myös harmaa taustaväri, joka osoittaa viikonlopulle sijoittuvia havaintoarvoja. Pääkaavion vasemmassa reunassa on havaintosuureiden y-akselit, joiden interaktiivisina ominaisuuksina on akselin nimen näyttäminen, kun hiiren osoitin viedään sen päälle. Lohkotyypin alimmainen osio sisältää pienemmän kaavion samoilla havaintotiedoilla kuin pääkaavio. Pienemmässä kaaviossa keltaisella värillä on valittu aikaväli, joka näytetään pääkaaviossa.



KUVIO 36. Havaintotietoja esittävä lohkotyyppi toiminnassa sivustolla

8 YHTEENVETO

Opinnäytetyön tavoitteena oli toteuttaa näyttävä ja toimiva sääpalvelu, jossa havaintoasema lähettää reaaliaikaista havaintotietoa www-palvelussa käytettäväksi. Työn lähtökohtana oli tarve reaaliaikaista säätilatietoa tarjoavasta palvelusta, joka antaa tärkeää tietoa purjeveneilijöille sekä muille merenkävijöille. Lisäksi havaintohistoriaa on tarkoitus tutkia ja sen perusteella toteuttaa paikallisia suunnitelmia eri kohteiden ympärille.

Työn teoriaosuudessa tutustuttiin havaintoasemien sisältämiin laitteisiin sekä niiden konfigurointiin. Työssä perehdyttiin myös nykyaikaiselle www-palvelulle ominaisiin teknologioihin sekä niiden soveltamiseen sääpalvelun toteuttamiseksi.

Opinnäytetyönä toteutettu sääpalvelu on vielä kehitysvaiheessa, mutta erilaisia käyttökokemuksia siitä kuitenkin on. Havaintoaseman ympärivuotista testausta koskevat käyttökokemukset ovat olleet positiivisia, mutta aseman virransaanti on tuottanut ongelmia; pelkän aurinkokennon käyttö akkujen lataamiseksi on osoittautunut epävarmaksi, sillä auringon esiintyminen talvisaikaan Suomessa on erittäin vähäistä. Tästä johtuen havaintoaseman akun varaustaso ei pysy riittävän korkealla tasolla eikä näin ollen pysty lähettämään havaintotietoja. Havaintoaseman ympärivuotisella testauksella on haluttu osoittaa kokoonpanon luotettavuus ja toimivuus eri vuoden-aikoina. Erityistä kiitosta järjestelmän toimivuuden osalta on tullut havaintotietojen reaaliaikaisuudesta sekä kerran vuorokaudessa tulevasta datasta, joka sisältää koko vuorokauden havaintotiedot.

Sääpalvelulle asetettu alustava tavoite vuodelle 2013 käsittää noin 20 uutta havaintoasemaa, jotka jäävät todennäköisesti toteuttamatta virransaanti-ongelmien johdosta. Tavoitteet uusista asemista siirtyvät näin ollen vuodelle 2014, kun olemassa olevan aseman ympärivuotinen testaus on saatu onnistuneesti päätökseen. Nykyinen asema tulee saamaan virransaanti-ongelmien myötä kesällä 2013 tuulivoimageneraattorin, jolloin asema hyödyntää sekä aurinko- että tuulienergiaa. Lisäksi sääpalvelussa on

tarkoitus soveltaa SaaS-mallia, jossa laitteen ostajat saavat oman APN-verkon SIM-kortin sekä käyttöoikeuden palveluun. Suunnitelmissa on myös mahdollisia käyttömaksuja osalle palvelusta muille kuin laitteen omistajille.

LÄHTEET

Rodriguez, A. 2008. RESTful Web services: The basics
[viitattu 11.3.2013]. Saatavissa:
http://www.ibm.com/developerworks/webservices/library/ws-restful/

Concrete5. 2013a. About [viitattu 28.2.2013]. Saatavissa:
http://www.concrete5.org/about/

Concrete5. 2013b. For Developers [viitattu 28.2.2013]. Saatavissa:
http://www.concrete5.org/about/developers-tour/

Concrete5. 2013c. History [viitattu 28.2.2013]. Saatavissa:
http://www.concrete5.org/about/our_philosophy/history/

IOLA & Ole Laursen. 2013. Flot: Attractive JavaScript plotting for jQuery
[viitattu 15.3.2013]. Saatavissa:
http://www.flotcharts.org/

Fielding, R. 2000a. Architectural Styles and the Design of Network-based
Software Architectures [viitattu 11.3.2013]. Saatavissa:
http://www.ics.uci.edu/~fielding/pubs/dissertation/evaluation.htm

Fielding, R. 2000b. Architectural Styles and the Design of Network-based
Software Architectures [viitattu 11.3.2013]. Saatavissa:
http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

Fielding, R. Gettys, J. Mogul, J. Frystyk, H. Masinter, L. Leach, P. Barners-
Lee, T. 1999. Hypertext Transfer Protocol – HTTP/1.1
[viitattu 11.3.2013]. Saatavissa:
http://www.w3.org/Protocols/rfc2616/rfc2616.html

The jQuery Foundation. 2013. jQuery [viitattu 15.3.2013]. Saatavissa:
http://jquery.com/

Reenskaug, T. 1979. MVC XEROX PARC 1978-79
[viitattu 12.3.2013]. Saatavissa:
http://heim.ifi.uio.no/~trygver/themes/mvc/mvc-index.html

Vaisala. 2009. Vaisala Hydromet™ System MAWS110/301 [viitattu 5.3.2012]. Saatavissa: http://www.vaisala.com/Vaisala%20Documents/Brochures%20and%20Datasheets/MAWS110_datasheet_B210844EN-B_LowRes.pdf

Vaisala. 2010. WXT520 User's Guide in English [viitattu 5.3.2012]. Saatavissa: http://www.vaisala.fi/Vaisala%20Documents/User%20Guides%20and%20Quick%20Ref%20Guides/WXT520_User_Guide_in_English.pdf

W3Schools. 2013. AJAX Introduction [viitattu 15.3.2013]. Saatavissa: http://www.w3schools.com/ajax/ajax_intro.asp