OULUN AMMATTIKORKEAKOULU

Janne Kähkönen

Tietokannan suunnittelu ja toteutus Sääpäiväkirja -sivustolle

Avoimen datan hyödyntäminen ohjelmistorajapintaa käyttäen

Tietokannan suunnittelu ja toteutus Sääpäiväkirja -sivustolle

Avoimen datan hyödyntäminen ohjelmistorajapintaa käyttäen

Janne Kähkönen
Opinnäytetyö
Syksy 2018
Tietojenkäsittelyn tutkinto-ohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietojenkäsittelyn tutkinto-ohjelma, Digitaaliset palvelut

Tekijä: Janne Kähkönen
Opinnäytetyön nimi: Tietokannan suunnittelu ja toteutus Sääpäiväkirja -sivustolle
Työn ohjaaja: Esa Niiranen
Työn valmistumislukukausi ja -vuosi: Syksy 2018 Sivumäärä: 68 + 31

Opinnäytetyön tavoitteena oli tehdä Sääpäiväkirja -sivusto ja suunnitella tietokanta sivustolle. Sivustolla voidaan tallentaa päivittäisiä säähavaintoja ja tallennuksen yhteydessä haetaan Ilmatieteen laitoksen avoimesta datasta sen hetkiset säätiedot avointa ohjelmistorajapintaa hyväksi käyttäen. Suunniteltu sivusto mahdollistaa säätietojen / säähavaintojen tallennuksen mobiililaitteella ja tietokoneella.

Sääpäiväkirjan sivustoa ja tietokantaa kehitettiin virtualisoidussa Windows 10 kehitysympäristössä. Kehitysympäristössä oli asennettuna XAMPP ja NetBeans ohjelmistot. MariaDB toimii tietokantamoottorina sivustolla.

Lopputuloksena syntyi Sääpäiväkirja -sivusto ja tietokanta, johon voidaan tallentaa säähavaintoja ja johon myös tallennetaan Ilmatieteen laitoksen avoimesta datasta haetut tiedot. Sivustolta löytyy raportteja vanhojen säätietojen hakuun.

Kehitysehdotuksiin tuli tietokannan muutoksia, jotta Ilmatieteen laitoksen avoimesta datasta pystyttäisiin tallentamaan enemmän säätietoja. Jatkojalostus työhön onnistuu pienellä vaivalla.

Asiasanat:
PHP, Tietokanta, Tietokannan suunnittelu, Relaatiotietokanta, Avoin data, API, Bootstrap, SQL

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Business Information Systems, Digital Services

---

Author: Janne Kähkönen
Title of thesis: Database design and created for Weather Journal
Supervisor: Esa Niiranen
Term and year when the thesis was submitted: Autumn 2018 Number of pages: 68 + 31

The purpose of the work was to create Weather Journal website and design a database for the website. The website allows you to save your daily weather observations and when the weather observations are saved, current weather data is also fetched from the Finnish Meteorological Institute with the open data download by the API interface. The designed site allows weather and weather observations to be saved on your mobile device and computer.

The Weather Journal website and a database were developed in a virtualized Windows 10 development environment. The development environment included XAMPP and NetBeans software. The website's database engine is MariaDB.

As a result, website and a database were created to record your own weather observations and information extracted from the open data from the Finnish Meteorological Institute. Development suggestions for database were that more data could be stored from the open data. The further development of the work would be done with little effort.

Keywords:
PHP, Database, Design Database, Relation Database, Open Data, API, Bootstrap, SQL

SISÄLLYS

TERMISTÖ

Apache Apache HTTP Server on ilmainen avoimen lähdekoodin palvelinohjelmisto, jonka päällä toimii suurin osa internet sivustoista.

API API tulee sanoista Application Programming Interface eli ohjelmistorajapinta. Sitä käytetään eri ohjelmistojen ja sovellusten välisessä liikenteessä.

ASP.NET Microsoft on kehittänyt ASP.NET:in, jolla voidaan tehdä internet sivustoja. ASP.NET tulee Microsoftin julkaisemassa Visual Studio ohjelmointityökalupaketissa.

Bootstrap Bootstrapilla tehdään moderneja skaalautuvia internet sivustoja, ja siihen löytyy valmiina kirjastoja erilaisille toiminnoille.

HTML HTML tulee sanoista Hypertext Markup Language ja sitä käytetään internet sivustojen teossa.

JavaScript JavaScript on ohjelmointikieli, jota käytetään HTML:n kanssa internet sivujen teossa. JavaScriptillä saadaan tehtyä mm. pop up -ikkunoita sivustolle.

MariaDB MariaDB on tietokantamoottori, jota käytetään paljon internet sivustojen tietokantamoottorina ja MariaDB pohjautuu MySQL:ään.

MIT-lisenssi MIT-lisenssi on ohjelmalisenssi, jossa on hyvin vähän rajoittavia tekijöitä ohjelmiston käytössä. Käyttäjä voi muokata, kopioida ja käyttää ohjelmaa omassa työssään, kunhan lisenssi teksti säilyy ohjelmistossa.

PHP PHP on ohjelmointikieli ja se tulee sanoista Hypertext Preprocessor ja sitä käytetään web-palvelinympäristöissä.

phpMyAdmin	phpMyAdmin ohjelmistolla operoidaan MySQL:n ja MariaDB:n tietokantoja. phpMyAdmin asentuu samalla kun asennetaan XAMPP.
Project	Project on Microsoftin julkaisema ohjelmisto, jolla voidaan hallinnoida projektiaikataulua.
Relaatio	Relaatiota käytetään relaatiotietokannan peruskäsitteinä ja sillä voidaan määritellä taulut, sarakkeet ja rivit.
SQL	SQL tulee sanoista Structured Query Language, sitä käytetään relaatiotietokannan erilaisten kyselyjen, muutoksien, lisäyksien ja poistojen standardisoituna kielenä.
SQL komentoja	SQL komentoja ovat SELECT haku, UPDATE muutos, INSERT lisäys, DELETE poisto, CREATE DATABASE tietokannan luonti, CREATE TABLE taulun luonti, CREATE VIEW näkymän luonti tietokanta hausta.
Tietokanta	Tietokanta on tietosäilö, jota voidaan lukea, muokata ja poistaa tiedon muuttuessa. Tietokantaan voidaan tallentaa mm. havaintotietoja, joita voidaan myöhemmin hakea erilaisten hakuehtojen perusteella.
VirtualBox	VirtualBox on virtualisointiohjelmisto, jolla voidaan asentaa mm. Apple macOS koneelle Windows käyttöjärjestelmä.
Visio	Visio on Microsoftin julkaisema ohjelmisto, jolla voidaan piirtää kaavioita, kuvioita yms. Työssä Visiolla on piirretty tietokanta kaaviot.

1 JOHDANTO

Opinnäytetyön tavoitteena oli tehdä Sääpäiväkirja -sivusto ja suunnitella tietokanta sivustolle. Työssä hyödynnettiin Ilmatieteen laitoksen avoimen datan ohjelmistorajapintaa. Ohjelmistorajapinnan kautta haettiin säätiedot sivustolle ja tietokantaan.

Suunniteltu sivusto mahdollistaa säätietojen / säähavaintojen tallennuksen mobiililaitteella ja tietokoneella. Havainnot ja sen hetkinen säätieto tallentuu Sääpäiväkirja -sivuston tietokantaan, säähavaintojen tallennuksen yhteydessä haetaan säätieto Ilmatieteen laitokselta ohjelmistorajapintaa hyväksikäyttäen. Tietoja voidaan tarkistella myöhemmin tallennettujen raporttien kautta, esimerkiksi millainen sää oli viime juhannuksena. Ilmatieteen laitokselta voidaan hakea säätietoja, mutta ne ovat virallisten mittausasemien tietoja, silloin on voinut olla myös erilaisia havaintoja. Muita havaintoja voi olla vaikka, että puu kaatui pihaan tai juhannuksena tuli niin paljon rakeita, että maa oli valkoinen.

Sääpäiväkirjan sivustoa ja tietokantaa kehitettiin virtualisoidussa Windows 10 kehitysympäristössä, johon oli asennettuna XAMPP, Microsoft Visual Studio, Microsoft SQL Server, Microsoft SQL Server Management Studio. Ohjelmistojen kokeilun jälkeen valituksi tuli XAMPP ja NetBeans ympäristöt, koska XAMPP tukee suoraan HTML- ja PHP-kieliä, ja siinä tulee mukana MariaDB:n tietokantamoottori.

Kokonaisuuden yhdistäminen työssä on lajiteltu useampaan eri kategoriaan; internet sivustoon, PHP- ja tietokantaan. Internet sivusto -osiossa keskitytään HTML-puolelle. PHP -osiossa keskitytään PHP -koodiin ja siinä esiin tulleisiin ongelmiin. Tietokanta-osiossa keskitytään tietokannan tekemiseen ja mitä ongelmia tuli tietokannan tekemisessä.

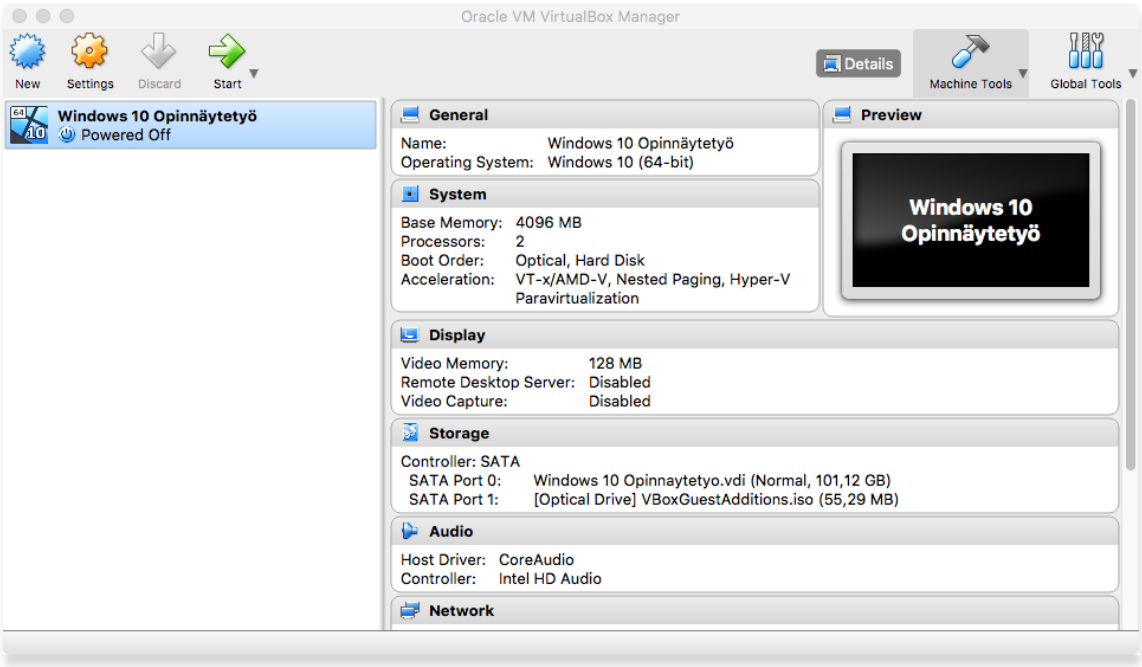
2 KEHITYSYMPÄRISTÖ

Työssäni kehitin Sääpäiväkirja -sivuston. Sivustolla voidaan syöttää omia säähavaintoja ja hakea vanhoja havaintoja. Sivusto on suunniteltu mobiililaitteita ajatellen, siten että käyttäminen on mobiililaitteilla sujuvaa. Työn pääsisältönä on tietokannan suunnittelu ja toteutus sivustolle, jossa haetaan avointa ohjelmistorajapintaa hyväksi käyttäen säätietoja Ilmatieteen laitoksen avoimesta datasta ja näiden tietojen tallennus Sääpäiväkirjan tietokantaan.

2.1 Kehitysympäristön virtualisointi

Kehitysympäristö on asennettu virtualisoituun Windows 10 -ympäristöön, jota pyöritetään VirtualBox virtualisointi työkalulla Applen koneella. Päädyin Windows 10 -kehitysympäristöön, koska siihen on saatavilla useita erilaisia kehitystyökaluja.

VirtualBox on tehokas x86- ja AMD64 / Intel 64-virtualisointituote yrityksille ja kotikäyttäjille. VirtualBox on vapaasti saatavilla Open Source -ohjelmistona GNU General Public License (GPL) -version 2 mukaisesti. VirtualBox toimii Windows-, Linux-, Machintosh- ja Solar- käyttöjärjestelmillä ja tukee useita virtualisoitavia käyttöjärjestelmiä Windows, DOS, Linux, Solar, OpenSolaris, OS / 2 ja OpenBSD. Ohjelmistoa kehitetään aktiivisesti useilla julkaisuilla ja uusia ominaisuuksia kehitetään tuetuista käyttöjärjestelmistä ja käyttöympäristöistä (Oracle VM VirtualBox 2018, viitattu 3.3.2018.) Kuvankaappaus Oracle VM VirtualBox Manager ikkunasta Apple macOS ympäristössä on kuviossa 1.



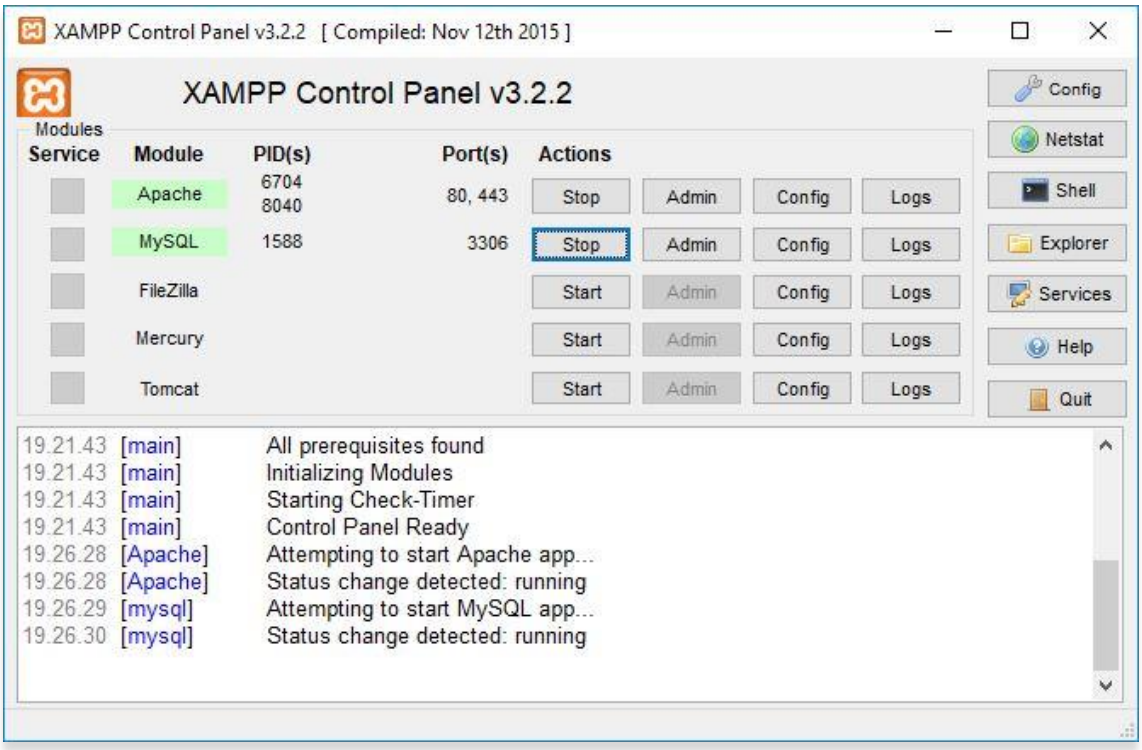
KUVIO 1. Kuvankaappaus Oracle VM VirtualBox Manager ikkunasta Apple macOS ympäristössä

2.2 Kehitysympäristötyökalujen testaus

Kehitysympäristöön on asennettu XAMPP, Microsoft Visual Studio Community 2017, Microsoft SQL Server, Microsoft SQL Server Management Studio, Programmer's Notepad ja NetBeans IDE. Asennetuista ohjelmistoista testaan, miten helposti näillä pystyy tekemään internet sivuja ja miten näiden siirto- ja julkaisu onnistuu. Tietokantamoottori tulee sen mukaan, kumpaan Visual Studion tai NetBeans IDE välillä päädyn.

2.2.1 XAMPP

XAMPP (katso kuvio 2) on suosittu PHP-kehitysympäristö, joka on täysin ilmainen ja helppo käyttöinen Apache-jakelu. XAMPP sisältää Apache, MariaDB:n, PHP:n ja Perl:n. XAMPP on saatavilla Windows-, Linux- ja macOS-käyttöjärjestelmille (Apache Friends 2018, viitattu 3.3.2018.)

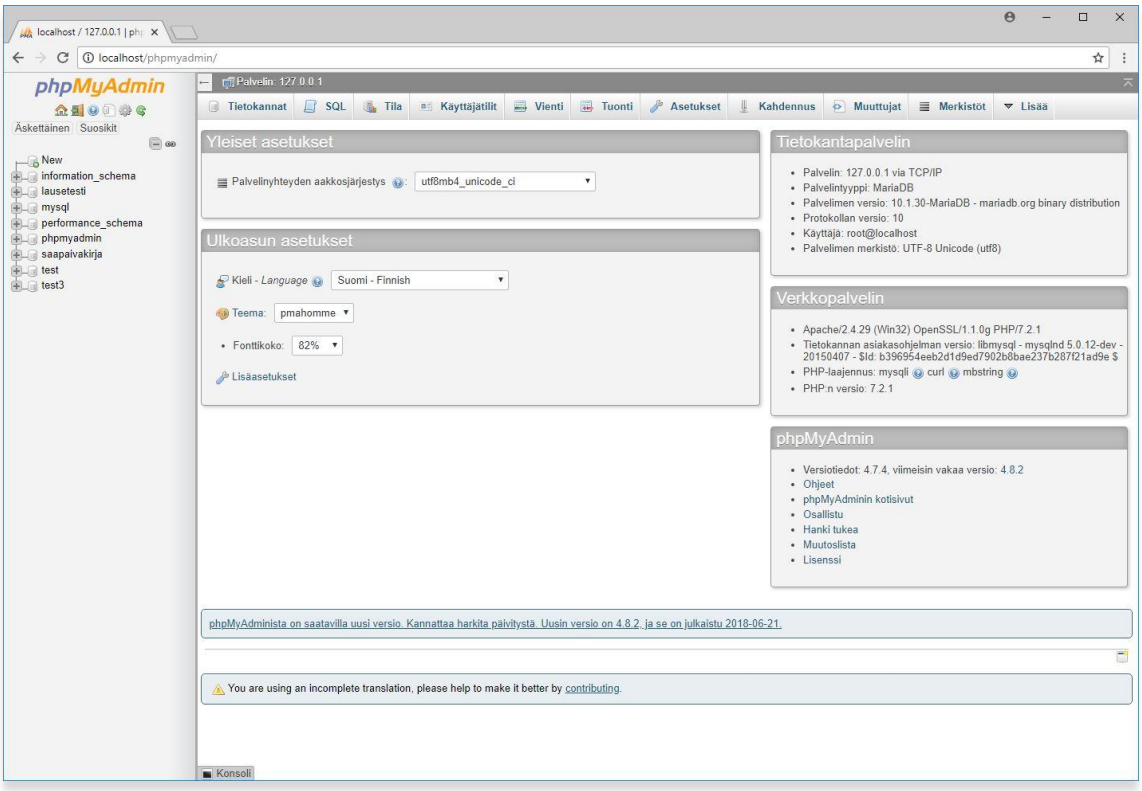


KUVIO 2. Kuvankaappaus XAMPP ohjelmasta

Apache on avoimen lähdekoodin HTTP-palvelinohjelmisto Unix ja Windows käyttöjärjestelmille. Apachea kehittää Apache HTTP Server Project, jonka tavoitteena on tarjota turvallinen, tehokas ja laajennettava palvelin, joka tarjoaa HTTP-palveluita synkronoituna nykyisten HTTP-standardien kanssa. Apache HTTP-palvelin lanseerattiin vuonna 1995 ja se on ollut huhtikuusta 1996 lähtien internetin suosituin web-palvelin (The Apache Software Foundation 2018, viitattu 3.3.2018.)

2.2.2 phpMyAdmin

phpMyAdmin (katso kuvio 3) on ilmainen ohjelmistotyökalu, joka on tehty PHP:llä ja sillä voidaan hallita MySQL:n ja MariaDB:n tietokanta moottoreita web-sivustoilla. phpMyAdmin tukee useita toimintoja MySQL:ssä ja MariaDB:ssä. Käyttöliittymän kautta voidaan hallita usein käytettyjä toimintoja kuten tietokantojen, taulukoiden, sarakkeiden, suhteiden, indeksien, käyttäjien, käyttöoikeuksien jne. phpMyAdmin sivustolla on myös mahdollisuus suorittaa SQL-käskyjä. phpMyAdmin on käännetty 72 eri kielelle. phpMyAdmin kuuluu voittoa tavoittelemattomaan organisaatioon ja on Software Freedom Conservancy- ohjelman jäsen, joka auttaa edistämään, kehittämään ja puolustamaan ilmaisia, vapaita ja avoimen lähdekoodin ohjelmistoja (phpMyAdmin 2018, viitattu 6.8.2018.)

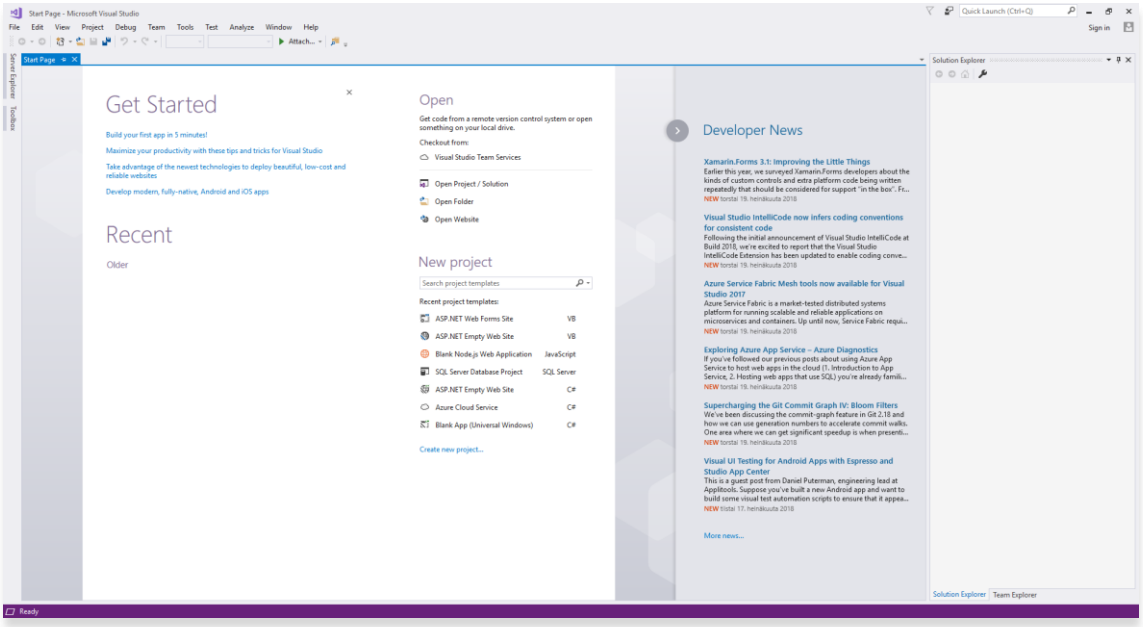


KUVIO 3. Kuvankaappaus phpMyAdmin hallinta ikkunasta

2.2.3 Microsoft Visual Studio 2017

Visual Studio on Microsoftin (katso kuvio 4) ohjelmistokehitysympäristö -työkalu, josta on saatavilla useita versioita erilaisiin käyttötarkoituksiin. Näistä Maksullisia ovat Visual Studio Professional, Visual Studio Enterprise ja Visual Studio Test Professional. Visual Studiolla voidaan kehittää internet sivuja ASP.NET ohjelmointikielellä (Microsoft 2018a, viitattu 20.7.2018.)

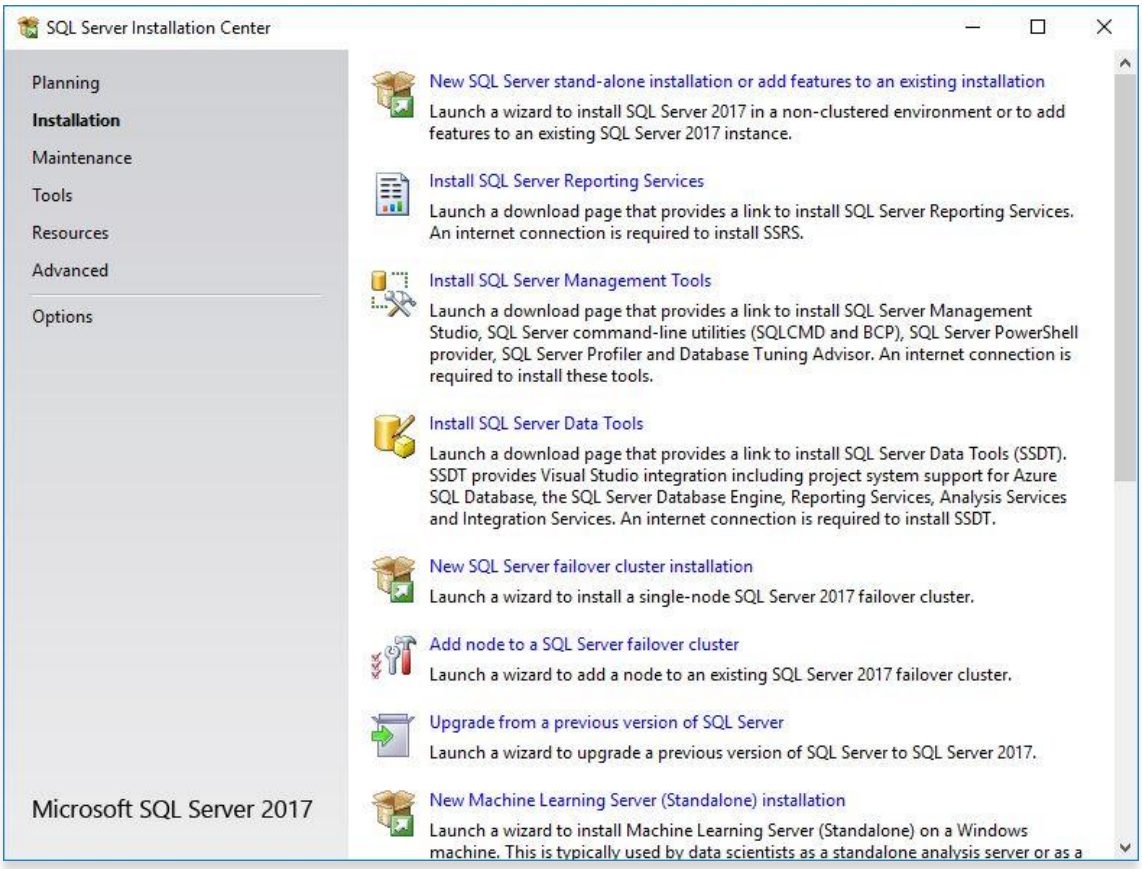
Visual Studiosta on myös ilmainen versio Visual Studio Community 2017, joka on laajennettavissa oleva ratkaisu yksittäisille kehittäjille ja sillä voidaan tehdä Android, iOS, Windows ja web-sovelluksia (Microsoft 2018b, viitattu 20.7.2018).



KUVIO 4. Kuvankaappaus Microsoft Visual Studio ohjelmistosta

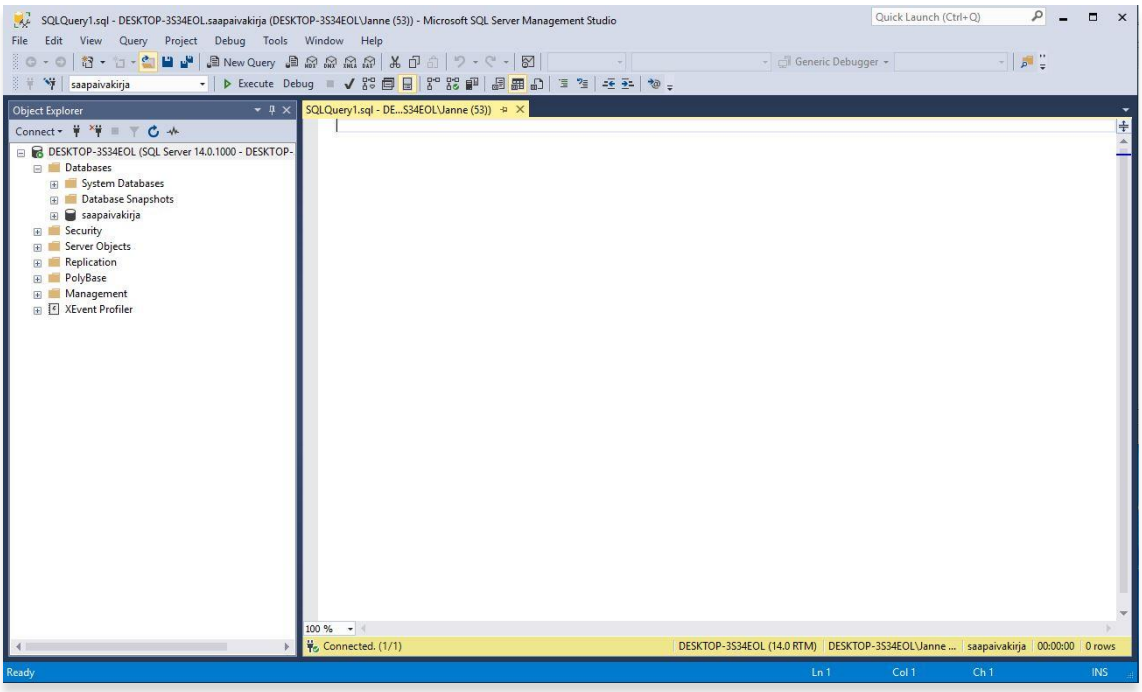
2.2.4 Microsoft SQL Server Express ja Microsoft SQL Server Management Studio

Microsoft SQL Server 2017 Express (katso kuvio 5) on maksuton tietokantamoottori, jolla voidaan tehdä enintään 10 Gt:n suuruisia tietokantoja (Microsoft 2018c, viitattu 20.7.2018).



KUVIO 5. Kuvankaappaus Microsoft SQL Server 2017 asennusvalikosta

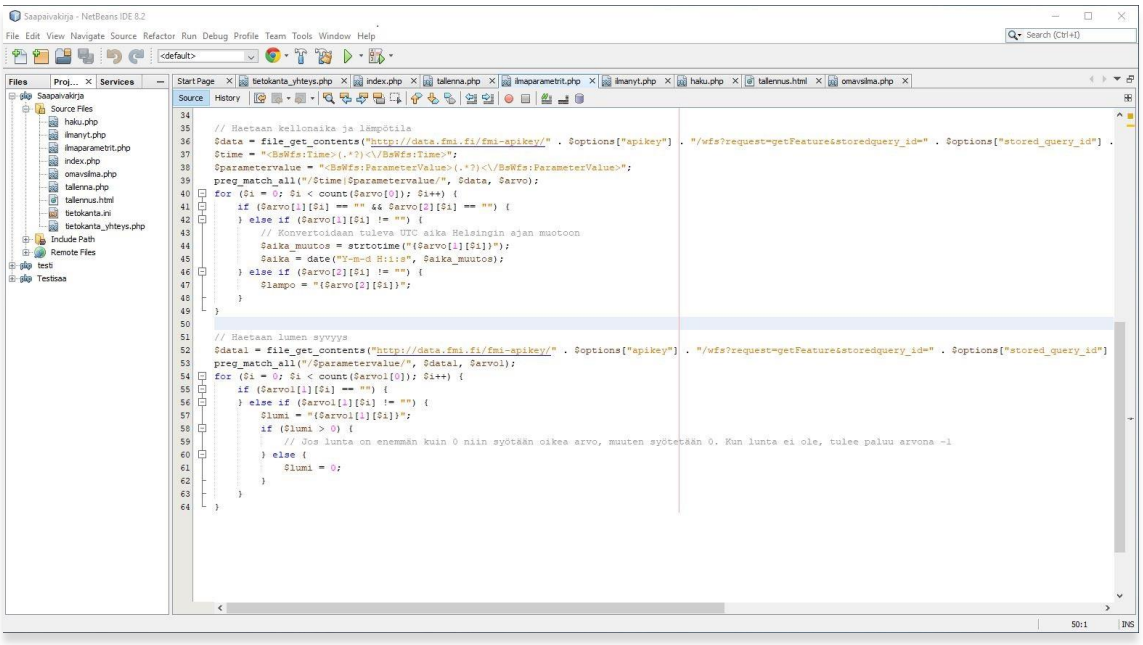
Microsoft SQL Server Management Studiolla (katso kuvio 6) voidaan määritellä, käsitellä, valvoa ja hallita paikallisia Windows-- ja Linux-ympäristössä pyöriviä tietokantoja ja myös Azure pilvipalveluissa olevia tietokantoja. Management Studio muistuttaa Visual Studio -ohjelmistokehitysympäristötyökalua (Microsoft 2018d, viitattu 20.7.2018.)



KUVIO 6. Kuvankaappaus Microsoft SQL Server Management Studio ohjelmistosta

2.2.5 NetBeans IDE

NetBeans IDE (katso kuvio 7) on ilmainen avoimen lähdekoodin ohjelmistokehitystyökalu, joka julkaistaan Apache Software Foundationin hallinnoimana projektina. NetBeans IDE:llä voidaan kehittää nopeasti ja helposti työpöytä-, mobiili- ja web-sovelluksia. Ohjelmistokielinä mm. Java, JavaScript, HTML5, PHP ja C / C++ (NetBeans 2018a, viitattu 20.7.2018 & NetBeans 2018b, viitattu 20.7.2018.)



KUVIO 7. Kuvankaappaus NetBeans IDE 8.2 ohjelmistosta

2.3 Valittu kehitysympäristö

Aikaisemmissa kohdissa olevia eri vaihtoehtoja kokeiltuani päädyin XAMPP ja NetBeans ympäristöihin, koska XAMPP tukee suoraan HTML- ja PHP-kieliä, ja sen mukana tulee MariaDB:n tietokantamoottori. Testaukset ja kokeilut onnistuvat sillä nopeasti, koska NetBeansissa voi määritellä projekti kansioksi XAMPP:in kansion, jossa sivusto sijaitsee. NetBeans on työkaluna minulle tutumpi kuin Microsoft Visual Studio, koska olen käyttänyt sitä web-ohjelmointi kurssilla. Ohjelmoinnin perusteet kurssilla käytettiin Cloud9 pilviympäristö -työkalua, mutta siinä täytyy siirtää joka kerta sivusto Cloud9:een, jos tekee kehitystä omalla koneella. Visual Studiolla web-kehitys tapahtuisi ASP.NET ohjelmointikielellä, josta minulla ei ole kokemuksia ja tämä vaatisi tähän myös ASP.NET ohjelmointikielen opiskelun.

Näiden lisäksi olen käyttänyt opinnäytetyön tekemiseen:

- Microsoftin Visio -työkalua kaavioiden ja tietokanta mallien piirtämiseen
- Microsoft Project Professional -työkalua opinnäytetyön aikataulusuunnitelman tekemiseen ja aikataulussa pysymiseen
- wireframe.cc -työkalua internet sivustojen ensimmäisten mallien piirtämiseen
- Programmer's Notepad -työkalua nopeisiin pieniin testeihin sivuston kokeilussa.

3 TIETOKANTA

Kappaleessa käydään läpi tietokannan relaatiomallia, rakennetta, taulujen välisiä suhteita ja tietokannan suunnittelua.

3.1 Käytetty tietokantamoottori

MariaDB on avoimen lähdekoodin tietokanta. MariaDB perustuu MySQL tietokantaan ja sitä on aloitettu kehittämään vuonna 2009 samojen henkilöiden toimesta, jotka ovat aikaisemmin tehneet MySQL:llä. Perustajat halusivat kehittää vaihtoehdon MySQL:lle, jonka Oracle osti Sun Microsystemsiltä (MariaDB 2018a, viitattu 21.7.2018.)

3.2 Relaatiotietokanta tai Relaatiomalli

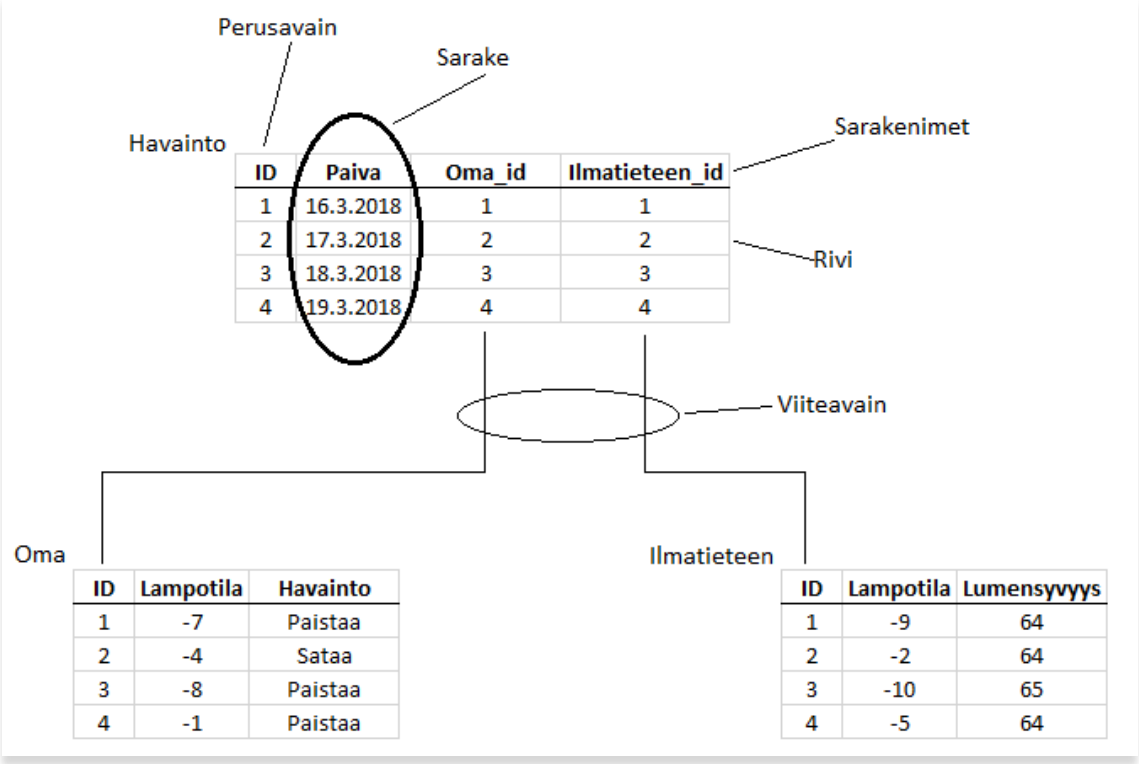
Edgar F Codd julkaisi vuonna 1970 relaatiotietokanta mallin. IBM:n tutkimuslaboratoriossa San Josessa tehtiin käytännön toteutus System R:stä vuonna 1973, jolla haluttiin näyttää relaatiomallin ominaisuudet ja hyödyt teollisessa toteutuksessa relaatiotietokannasta. System R oli menestys ja vuonna 1981 IBM ilmoitti ensimmäisen relaatiotietokanta tuotteestaan SQL / DS. Vuonna 1983 IBM julkaisi DB2 tietokannan suurtietokoneille (IBM Archives, viitattu 21.7.2018.)

Coddin kehittämä relaatiomalli määrittelee relaatiotietokannan teoreettisen pohjan. Relaatiomalli pohjautuu joukko-oppiin, matematiikkaan ja predikaattilogiikkaan. Relaatiotietokannan fyysiseen toteutukseen relaatiomalli ei ota kantaa. Hänen kehittämä relaatiomalli on vallankumouksellista tietokanta maailmassa. Myös SQL on standardoitunut melkein ainoaksi tietokantakieleksi. Relaatiomalli voidaan jakaa kolmeen osaan: rakenne, käsittely ja eheyssäännöt (Hovi, Huotari & Lahdenmäki 2005, 7-8.)

Relaatiotietokannoissa relaatiotiedot esitetään tauluina, jossa riviä kutsutaan tietueeksi ja taulun jokaisella rivillä on yhtä monta kenttää eli tietoa. Riveillä on yksilöivä perusavain, joka vastaa kohdetta. Näihin kohteisiin tulee liittää vain tiedot, jotka liittyvät tähän ominaisuuteen (Lahtonen, 2002, 4.)

3.2.1 Rakenne

Tietokannan peruselementtejä ovat taulut ja tauluissa on sekä sarakkeita että rivejä (katso kuvio 8). Taulussa sarakkeilla on toisistaan poikkeavat nimet ja sarakkeen tietojen arvot kuuluvat samaan arvo joukkoon. Näillä on yhteinen tietotyyppi joko numeerinen tai merkkimuotoinen ja niille on varattu enimmäispituus (Hovi ym. 2005, 8-9.)



ID	Paiva	Oma_id	Ilmatieteen_id
1	16.3.2018	1	1
2	17.3.2018	2	2
3	18.3.2018	3	3
4	19.3.2018	4	4

ID	Lampotila	Havainto
1	-7	Paistaa
2	-4	Sataa
3	-8	Paistaa
4	-1	Paistaa

ID	Lampotila	Lumensyvyys
1	-9	64
2	-2	64
3	-10	65
4	-5	64

KUVIO 8. Esimerkki kolmesta taulusta (Hovi ym. 2005, 8)

Tauluissa on tunnisteena perusavain ja tämän täytyy olla yksilöivä eli uniikki, joka tarkoittaa, ettei sarakkeessa saa olla kahdella tai useammalla rivillä samaa arvoa. Perusavain pystyy myös koostumaan useammasta sarakkeesta. Perusavaimen suunnittelu on tärkeää relaatiotietokannan suunnitteluprosessissa.

Taulujen välisiä suhteita kutsutaan yleensä äitilapsi -termillä. Äidillä voi olla useita lapsia, mutta lapsella voi olla vain yksi äiti. Lapsitaulun viiteavaimella viitaan äititaulun perusavaimeen, jolla taulut yhdistyvät keskenään. Eli viiteavaimia tarvitaan, jos halutaan yhdistää tauluja eli tehdä liitoksia (Hovi ym. 2005, 9.)

Kuviossa 8 on kuvattu kolme taulua, miten lapsitaulut Oma ja Ilmatieteen yhdistyvät äititauluun Havainto.

Relaatiomallisääntöjen mukaisesti taulussa ei saa olla tuplarivejä ja myöskään toistuvia ryhmiä. Nämä on estetty perusavaimella. Jos taulun jollakin sarakkeella ei ole arvoa, täytyy tähän merkitä ns. NULL -arvo eli tyhjä arvo. NULL ei tarkoita nollaa eikä välilyöntiä, vaan tuntematonta arvoa. NULL -arvo voidaan kieltää taulun määrittelyssä, ettei sarakkeessa voi olla NULL -arvoa, vaan jokaiseen sarakkeeseen täytyy syöttää arvo (Hovi ym. 2005, 10.)

3.2.2 Käsittely

Relaatiomallissa tietoja käsitellään joukko-oppimallisesti, joka oli Coddin nerokas ajatus relaatiomallissa. Tämä tarkoittaa, että taulu muodostuu riveistä ja tauluun voidaan kohdistaa joukko-operaatioita, esimerkiksi hae kaikki pilviset päivät vuodelta 2018. Joukko-operaatiossa voidaan käsitellä useita tauluja kerralla tai myös pelkästään yhtä taulua. Kuitenkin tiedon haku onnistuu helposti. Samoin tietojen päivitys onnistuu joukko-operaatiolla helposti, koska voidaan päivittää useita rivejä kerralla yhdellä päivitysoperaatiolla. Tästä johtuen tietojenkäsittelystä tulee hyvin tuottavaa (Hovi ym. 2005, 10.)

3.2.3 Eheyssäännöt

Relaatiomallissa otetaan kantaa myös tietokannan eheyteen. Tietokanta on silloin ehyt, kun tiedot ovat oikein ja ne ovat ristiriidattomat ja vastaavat reaalimaailmaa. Tietokannan eheys vaarantuu silloin, jos tallennetaan kaksi kertaa sama havaintotieto, jolla on sama päiväys ja muut tiedot. Codd määritteli eheysrajoitteita, joista ensimmäinen eheyssääntö avaineheys. Tämän säännön pohjalta taulun perusavain ei saa olla tyhjä eli NULL arvo, tästä johtuen perusavain arvo on pakollinen (Hovi ym. 2005, 11.)

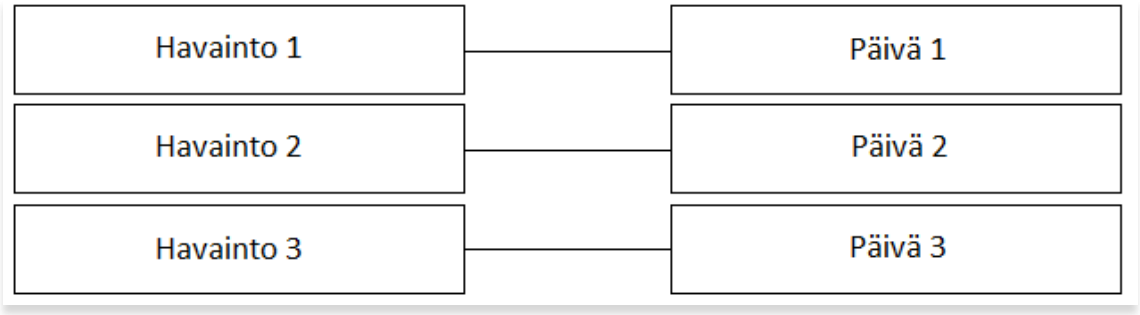
Toinen eheyssääntö on viite-eheys. Tietoja ei saa pystyä poistamaan äititaulusta, jos lapsitauluun jää tietoja. Muuten lapsitaulun tiedot jäävät ”orvoiksi”. Esimerkiksi (katso kuvio 8) jos havainto taulusta poistetaan ID 1 rivi, niin silloin lapsitauluihin oma ja ilmatieteen tauluihin jäävät tiedot ovat ”orpoja”. Tätä kutsutaan viite-eheyden särkymiseksi, jota ei saisi tapahtua (Hovi ym. 2005, 11-12.)

3.3 Taulujen välisiä suhteita

Meloni (2003, 29-32) mukaan erilaisia taulujen välisiä suhteita ovat yksi yhteen -suhteet, yksi moneen -suhteet ja moni moneen – suhteet.

Yksi yhteen -suhteet

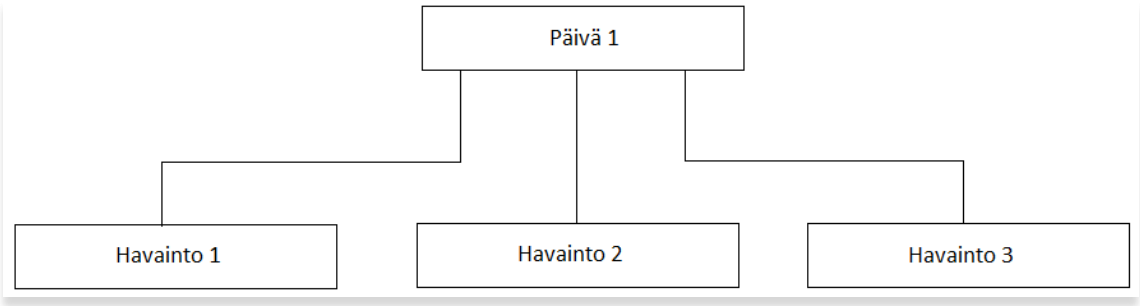
Yksi yhteen -suhteessa (katso kuvio 9) tieto esiintyy ainoastaan kerran taulussa, johon viitataan.



KUVIO 9. Jokaisessa päivässä on ainoastaan yksi havainto (Meloni 2003, 29)

Yksi moneen -suhteet

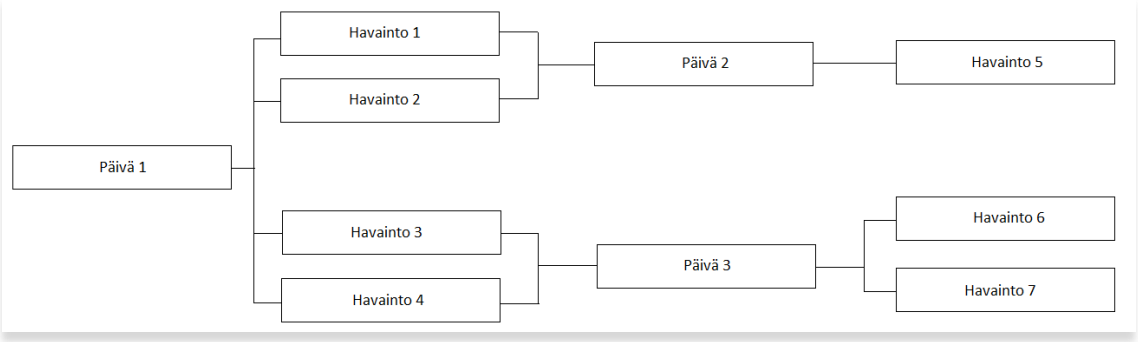
Yksi moneen -suhteessa (katso kuvio 10) viitattavien taulujen viiteavaimet löytyvät yhdestä taulusta. Tämä on kaikista yleisintä tietokannoissa. Esimerkiksi yhdessä päivässä voi olla useita eri havaintoja.



KUVIO 10. Yksi päivä sisältää useita havaintoja (Meloni 2003, 30)

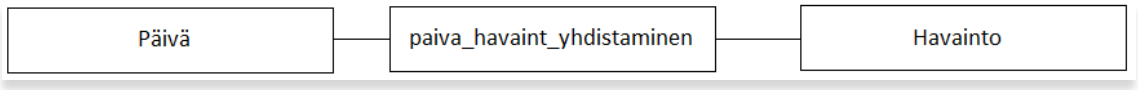
Moni moneen -suhteet

Moni moneen -suhteissa (katso kuvio 11) yhden taulun viiteavain voi esiintyä monessa taulussa, johon viitataan ja viitattavan taulun viiteavain voi esiintyä monta kertaa taulussa, josta viitattiin. Samoja havaintoja on useina päivinä ja päivässä on enemmän kuin yksi havainto.



KUVIO 11. Useita havaintoja päivässä, päivät sisältävät samoja havaintoja (Meloni 2003, 31)

Tällaista ei pystyisi helposti suoraan tekemään vaan tähän väliin jouduttaisiin tekemään uusi taulu (katso kuvio 12), jolla yhdistetään päivät ja havainnot yhteen. Usealla päivällä on samoja havaintoja ja samalla päivällä on useita eri havaintoja.



KUVIO 12. Päivä ja Havainto taulun yhdistämistaulu

Paiva_havainto_yhdistaminen (katso kuvio 12) taulu yhdistää taulut Päivä ja Havainto (katso kuvio 11) toisiinsa. Paiva_havainto_yhdistaminen taulu sisältäisi tiedot (katso kuvio 13).

PäiväID	HavaintoID
1	1
1	2
1	3
1	4
2	1
2	2
2	5
3	3
3	4
3	6
3	7

KUVIO 13. Paiva_havainto_yhdistaminen taulun sisältö

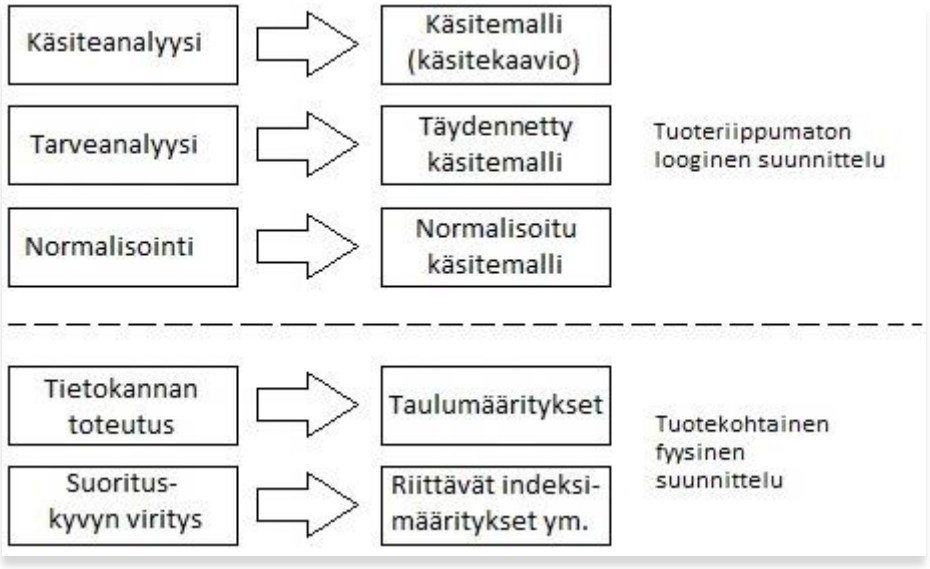
3.4 Tietokannan suunnittelu

Tietokannan hyvä suunnittelu on tärkeää, jotta tietokantaa käyttävät sovellukset toimisivat hyvin. Hyvä tietokanta, joka toimii tehokkaasti, on normalisoitu ja siinä on optimoidut suhteet (Meloni 2003, 28.)

Suorituskyvyn lisäksi hyvin suunniteltua tietokantaa on helppo ylläpitää. Tätä varten toistuvaa dataa on rajoitettu määrä. Jos on paljon toistuvaa dataa ja jonkin tämän datan ilmentymä muuttuu, esimerkiksi nimi, tällöin nimen muutos täytyy tehdä kaikkiin sarakkeisiin ja tauluihin, joissa tieto esiintyy. Jotta saman datan toistamisesta päästään eroon täytyy tehdä uusia tauluja, jotka sisältävät mahdolliset tiedot. Tämän jälkeen käytetään avainta viitattaessa arvoon, jos nyt arvo vaihtuu, niin tieto täytyy tehdä ainoastaan yhden kerran päätauluun. Viittaukset muihin tauluihin ja arvoihin pysyvät samana (Meloni 2003, 28.)

Tietokannan, joka on hyvin suunniteltu ja toteutettu, edut ovat jatkossa lukemattomat esimerkiksi päivitykset ja taulujen lisäyksen ohjelmiston laajentuessa. Mitä enemmän työskentelee tietokannan suunnittelussa etukäteen, sitä vähemmän joudutaan jälkikäteen tekemään korjauksia. Tietokannan suunnittelu sovelluksen julkaisemisen jälkeen on hukkaan heitettyä aikaa ja tämä käy kalliiksi. Tietokannan suunnitteluun kannattaa käyttää reilusti aikaa, ennen kuin sovellusta aloitetaan ohjelmoimaan (Meloni 2003, 28.)

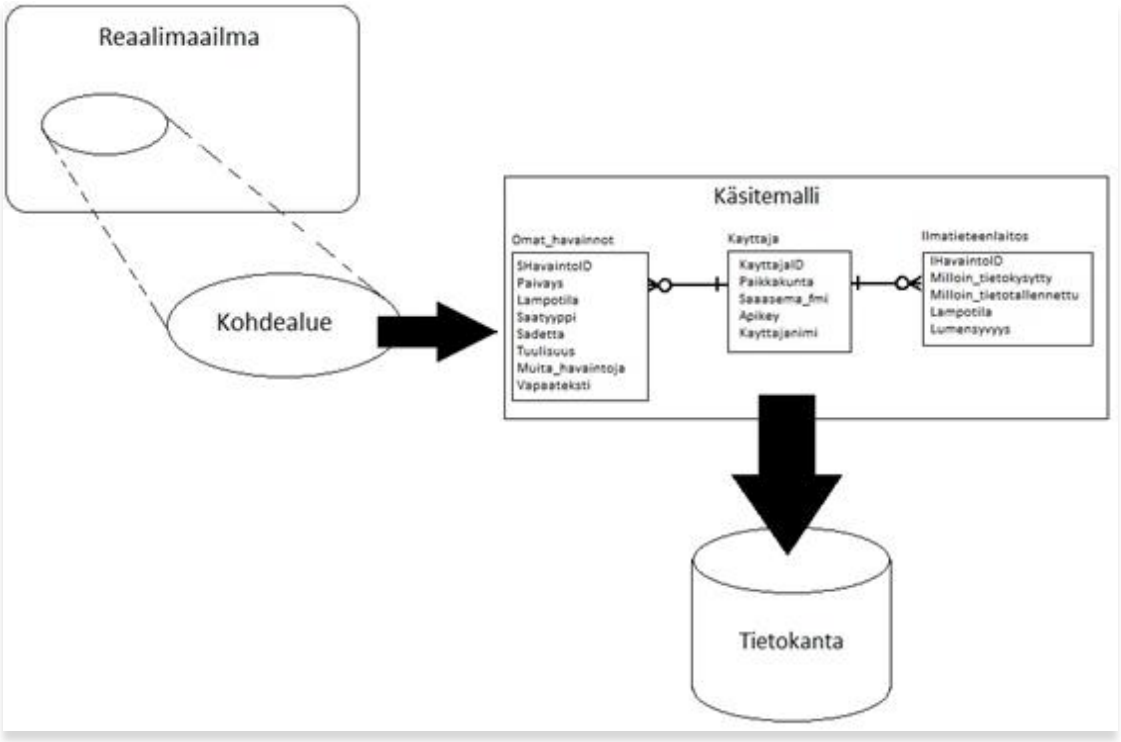
Tietokannan suunnittelussa käydään tietyt vaiheet läpi, jotta saadaan toimiva tietokanta. Tätä suunnitelmakokonaisuutta sanotaan suunnitteluputkeksi (katso kuvio 14) (Hovi ym. 2005, 24.)



KUVIO 14. Tietokantojen suunnitteluputki (Hovi ym. 2005, 24)

3.4.1 Käsiteanalyysi

Tietokannan suunnittelu alkaa käsiteanalyysin teolla. Sen tarkoituksena on määritellä ja kuvata tietokantaan tuleva tieto havainnollisilla kaavioilla, joiden pohjalta lopuksi voidaan rakentaa tietokanta. Käsiteanalyysissä kuvataan reaalimaailmasta rajattua osaa eli kohdealuetta (katso kuvio 15), josta on tarkoitus tehdä tietokanta. Tämän lopuksi syntyy käsitemalli, joka kuvaa kohdealuetta ja määrittelee tietokannan fyysisen pohjan (Hovi ym. 2005, 32.)



KUVIO 15. Kohdealueen valinta tulevaan tietokantaan (Hovi ym. 2005, 32)

Käsiteanalyysin käsitemallia pyöritetään ja tarkennetaan jokaisella kierroksella, kunnes syntyvät piirustukset, joiden pohjalta toteutetaan fyysinen tietokanta. Käsiteanalyysiä tehtäessä ei ajatella mitä tietokanta olisi loppujen lopuksi, vaan tietokanta kuvataan tarpeeksi riittävällä ja karkealla tasolla, jotta tiedetään mitä tietoja tietokantaa halutaan saada. Tässä vaiheessa ei vielä kannata miettiä tietokannan suoritus kykyä ja tauluja tarkalla tasolla (Hovi ym. 2005, 32-33.)

Käsitemalli voidaan ottaa käyttöön eri tietokannoissa, esimerkiksi MariaDB, MS SQL Server tai Oracle. Tästä johtuen käsitemalli on tietokanta- ja tuoteriippumaton. Käsitemallissa pyritään, että tiedot tallennetaan ainoastaan kerran. Esimerkiksi käyttäjän kaikkia tietoja ei tallenneta havaintotauluun useita kertoja, vaan siihen viitataan käyttäjätaulusta (Hovi ym. 2005, 33.)

Käsitemallia tehdessä aloitetaan karkealta tasolta, jossa ei ole vielä kaikkea nippelitietoa kerrottu ja tällä tasolla saadaan selville, millainen tietokanta täytyy olla. Tarkempaa tietoa tietokannasta saadaan, kun on ensiksi tehty käsitemalli ja se siirretään tarveanalyysiin (Hovi ym. 2005, 33.)

3.4.2 Tarveanalyysi

Tarveanalyysin tarkoituksena on saada tarkennettua tarkalle tasolle aikaisemmin tehty käsiteanalyysi. Tässä yhteydessä lisätään tietoja käsitteisiin tietotarpeiden perusteella. Tarveanalyysissä myös tarkistetaan ja testataan aikaisemmin tehtyä käsitemallia tiedossa olevilla tietotarpeilla. Aikaisemmin tehtyä käsitemallia täydennetään uusilla tiedoilla, joita huomataan tarvittavan ja mahdollisesti myös lisätään uusia käsitteitä ja yhteyksiä (Hovi ym. 2005, 80.)

Sovellusten suunnittelussa määritellään tietotarpeita ja tarveanalyysi perustuu näiden tietotarpeisiin. Tällä tarkoitetaan sovelluksen suunnittelussa mm. raportteja, näyttötietoja ja kaikkia muita ohjelman tarpeita, jotka käsittelevät tietokantaa. Tietotarpeissa täytyy olla kaikki tiedot saatavilla tarkasti ja tiedettävä mitä yksittäisiä tietoja ne tarkoittavat. Tarveanalyysiin sisältyy mm. lajittelukriteeri ja tieto-, käyttäjä- ja tapahtumamäärien selvitykset, joita tietoja käytetään testausta varten ja ne myös vaikuttavat mitä tietokannan hallintajärjestelmää käytetään (Hovi ym. 2005, 80.)

Tarveanalyysin tekeminen aloitetaan ottamalla esiin nykyinen käsiteanalyysi ja ottamalla käsittelyyn internet sivuston joku raportti tai tietojen tallennus sivulta. Tämän jälkeen tarkistetaan kaikki kyseinen sivuston tietojen tallennus yksitellen käsitemallista, jotta nämä tiedot löytyvät. Monesti tässä vaiheessa vielä käsitemallista puuttuu tietoja, koska käsitemalli tehtiin aikaisemmin karkealla tasolla. Jos tietoja puuttuu, niin tässä vaiheessa lisätään tiedot käsitteiden yhteyteen. Joskus saattaa myös käsite kokonaisuudessaan puuttua tai yhteys, niin tässä vaiheessa ne myös lisätään käsitemalliin. Kun nämä on tehty ensimmäiselle sivustolle tai raportille, niin samalla tavalla käydään kaikki muutkin läpi, kunnes kaikki tietotarpeet on käyty läpi. Kun kaikki tunnistettavat tietotarpeet on käyty läpi, käsitemalli on tarkennettu tarkalle- ja yksityiskohtaiselle tasolle. Käsitemalli on myös testattu ja on todettu toimivaksi näille tietotarpeille, joita on käsitelty (Hovi ym. 2005, 81.)

3.4.3 Normalisointi

Normalisoinnin on kehittänyt E. F. Codd vuonna 1972. Hän on myös kehittänyt mm. relaatiomallin. Normalisointia voidaan soveltaa kaikenlaisiin tietokanta rakenteisiin. Normalisointi on menetelmä,

jolla tietokanta jalostetaan parempaan tallennukseen. Kun tietojen toistaminen on saatu minimoitua, tietokantaa voidaan päivittää tehokkaasti, helposti pitää yhdenmukaisena (jossa tieto päivitetään ainoastaan yhteen paikkaan) ja tietokanta on muutosjoustava.

Normalisoinnissa käytetään eniten kolmea ensimmäistä vaihetta, ensimmäinen, toinen ja kolmas normaalimuoto. Näiden lisäksi myös on ns. Boyce-Code-normaalimuoto, neljäs normaalimuoto ja viides normaalimuoto, mutta näitä ei käytetä usein, koska nämä ovat hieman teoreettisia ja näiden merkitys on vähentynyt (Hovi ym. 2005, 86.)

Ensimmäinen normaalimuoto

Tietokanta on ensimmäisessä normaalimuodossa, kun jokaisen attribuutin arvo joukko koostuu ainoastaan yhdestä arvosta, eikä esimerkiksi listoista. Jokainen kenttä sisältää ainoastaan yhtä tietotyyppiä, jotka voivat olla tekstiä, numeroita tai muita perustietotyyppejä, mutta näitä ei saa laittaa sekaisin. Ensimmäinen normaalimuoto kieltää toistuvat ryhmät (Lahtonen 2002, 31.)

Toinen normaalimuoto

Tietokanta on toisessa normaalimuodossa, kun tietokanta on ensimmäisessä normaalimuodossa ja jokainen attribuutti, joka ei ole avainehdokas, on funktionaalisesti täysin riippuva tästä avainehdokkaasta. Kaikki kentät taulussa, jotka eivät ole avainehdokkaita, liittyvät suoraan taulun avaimeen, joka voi muodostua yhdestä tai useammasta kentästä. Tauluun täytyy tallentaa sellaista tietoa, joka liittyy yhteen asiaan tai kohteeseen. Taulun perusavain määrittelee tämän kohteen (Lahtonen 2002, 33.)

Kolmas normaalimuoto

Tietokanta on kolmannessa normaalimuodossa, kun tietokanta on toisessa normaalimuodossa ja muuttuja-attribuutit eivät ole transitiivisesti eli matemaattisesti riippuvaisia mistään relaation avainehdokkaasta. Kenttien kaikki tiedot, jotka eivät kuulu avaimeen täytyy olla itsenäisiä ja riippumattomia muista kentistä (Lahtonen 2002, 34.)

Muut normaalimuodot

Lahtonen (2002, 35) mukaan muut normaalimuodot lueteltuna alla

- Boyce-Codd normaalimuoto kieltää määräävän ominaisuuden esiintymisen taulussa, jollei kyseessä ei ole avain.

- Neljäs normaalimuoto koskee pelkästään tunnistetietoja sisältäviä tauluja, jotka esittävät erilaisia ominaisuuksien välisiä riippuvuuksia. Tauluissa ei saa esiintyä kahden tai useamman ominaisuuden moniarvoisia riippuvuuksia, jos nämä ovat keskenään riippumattomia ominaisuuksia.

- Viides normaalimuoto sanoo, että jos sama informaatio voidaan esittää tauluilla, joiden asteluku on alempi, on käytettävä näitä tauluja. Taulujen asteluku tarkoittaa sarakkeiden lukumäärää.

4 ILMATIETEEN LAITOS

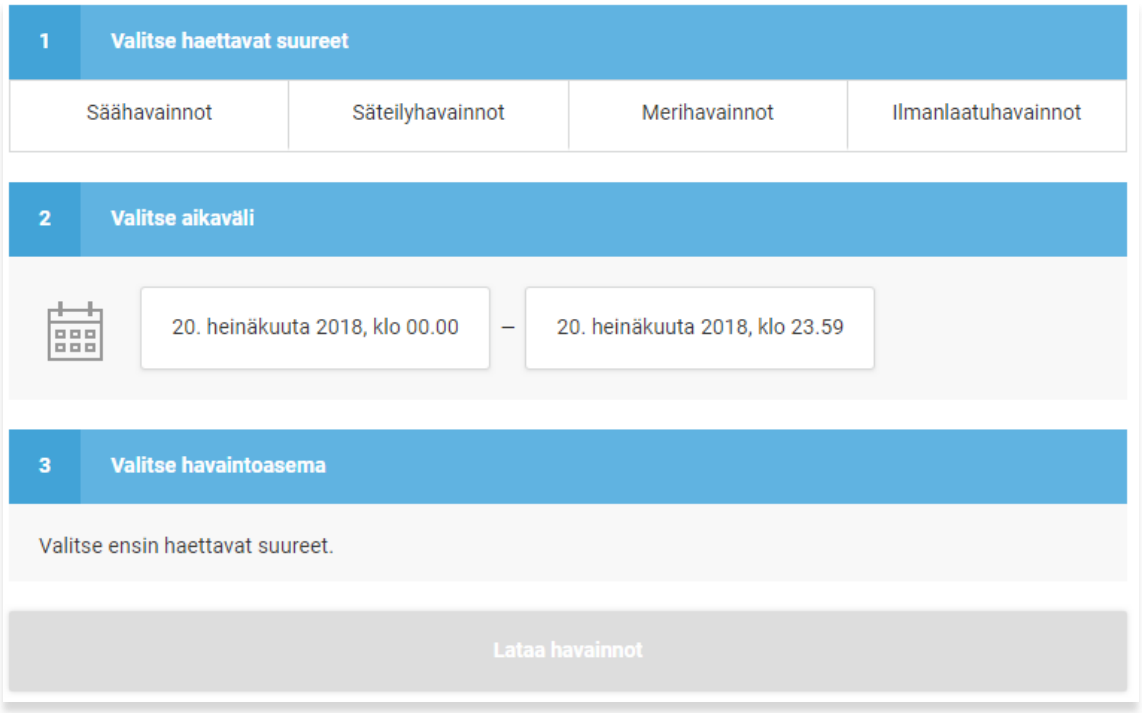
Ilmatieteen laitos aloitti julkaisemaan ohjelmistojaan avoimena lähdekoodina vuonna 2016. Ohjelmistoja voi käyttää, hyödyntää, muokata ja myös jaella vapaasti edelleen. Nämä ohjelmistot ovat julkaistu MIT-lisenssillä. MIT-lisenssi asettaa hyvin vähän rajoitteita ohjelmiston käytölle. MIT-lisenssi sallii ohjelmiston käytön osana suljettua ohjelmistoa, mutta tässä tapauksessa täytyy alkuperäinen lisenssi toimittaa ohjelmiston mukana. Ilmatieteen laitos julkaisee avatut ohjelmistot GitHub-verkkopalvelussa osoitteessa http://github.com/fmidev (Ilmatieteen laitos 2018a, viitattu 25.2.2018.)

4.1 Säätietojen haku

Säätietojen koneellista hakua varten täytyy rekisteröityä ja hakea API-avainta (fmi-apikey), johon liittyy tiettyjä rajoituksia. Yhdellä API-avaimella saa tehdä enintään 20 000 kyselyä vuorokaudessa latauspalveluun, katselupalveluun saa tehdä enintään 10 000 kyselyä vuorokaudessa ja yhdessä saa tehdä molempiin palveluihin viiden minuutin aikana yhteensä 600 kyselyä (Ilmatieteen laitos 2018b, viitattu 25.2.2018.)

Säätietoja pystytään hakemaan koneellisesti XML -formaatissa tai MetOLib JavaScript-kirjaston komennoilla. Näihin löytyy jo valmiiksi tallennettuja kyselyitä, joihin annetaan haluttuja parametrejä. Kyselyiden avulla saadaan haettua tietoja ja voidaan määritellä ajallisesti mitä tietoja haetaan (Ilmatieteen laitos 2018b, viitattu 25.2.2018, Ilmatieteen laitos 2018c, viitattu 21.7.2018.)

Hetkelliset säähavainnot (katso kuvio 16) on pystynyt lataamaan myös Exceliin, CSV- ja CSV (metatiedot) -formaatissa vuodesta 2010 alkaen (Ilmatieteen laitos 2018d, viitattu 21.7.2018).



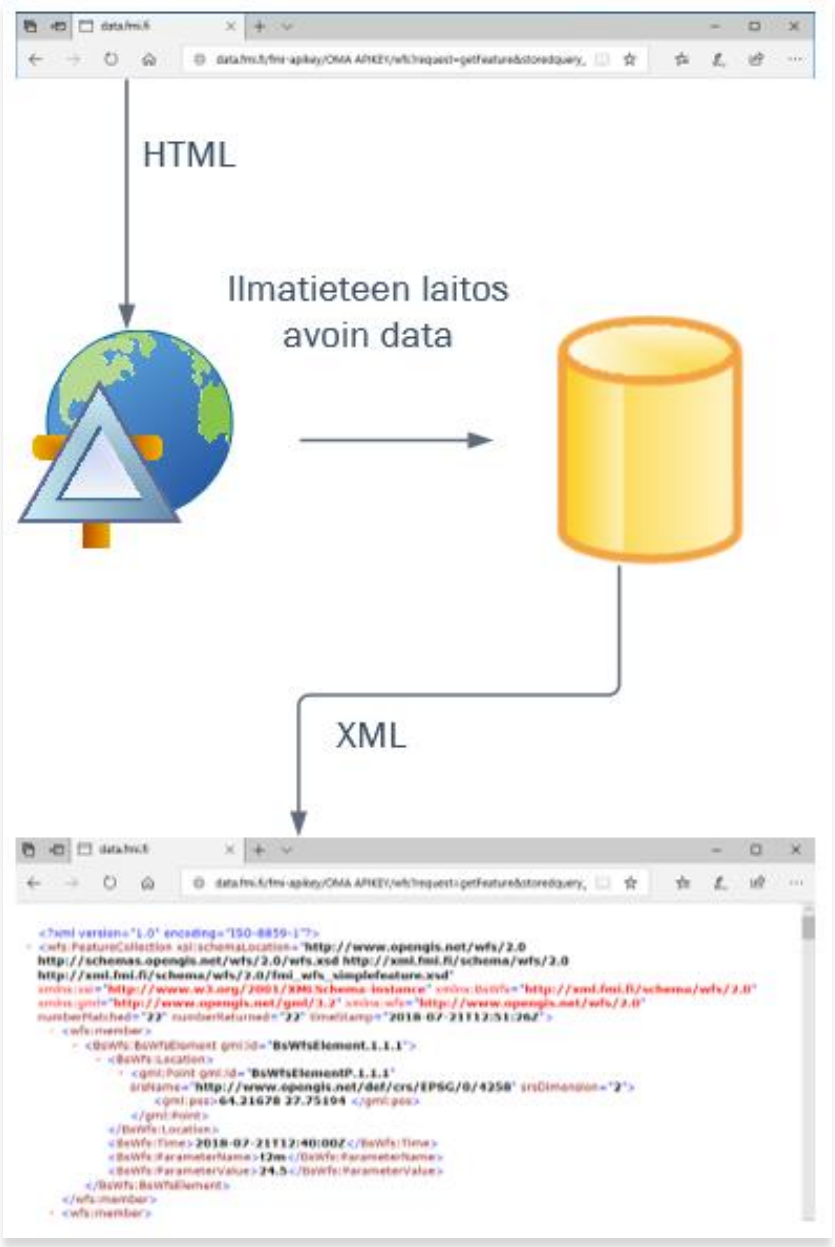
KUVIO 16. Säähavaintojen lataus (Ilmatieteen laitos 2018d, viitattu 21.7.2018)

4.1.1 XML

XML -muotoinen haku toteutetaan (katso kuvio 17) internet -selaimeen tehtävällä ”linkillä” (Ilmatieteen laitos 2018b, viitattu 21.7.2018). Esimerkki oheinen linkki:

http://data.fmi.fi/fmi-apikey/OMA APIKEY/wfs?request=getFeature&storedquery_id=fmi::observations::weather::simple&starttime=2018-05-13T14:00&place=kajaani

palauttaa Kajaanin Petäisenniskan säätiedot kello 17.00 eteenpäin. Haun aikavyöhyke on 0, jolloin hakuun pitää laittaa haluttu kellonaika -2 tuntia, jos aikana on talviaika, mutta kesällä ajaksi pitää laittaa -3 tuntia, jolloin tulee oikea kellonaika. Haku palauttaa mm. lämpötilan, lumensyvyyden, tuulen nopeuden ja tuulen suunnan (katso liite 1). Hakutuloksen perusteella Kajaanissa oli äitienpäivänä 13.5.2018 kello 17.00 oli 24,9 astetta lämmintä.



KUVIO 17. XML -muotoisen datan haku Ilmatieteen laitokselta internet -selaimella

Valmiiksi tallennetuista kyselyistä hetkelliset säähavaintoarvot saadaan kyselyllä *fmi::observations::weather::simple*. Hakuun voidaan laittaa parametrejä, joilla rajataan mitä tietoja halutaan saada. Mahdollisia parametrejä, joilla kysely tehdään ovat starttime, endtime, timestep, parameters, crs, bbox, place, fmisid, maxlocations, geoid ja wmo (Ilmatieteen laitos 2018e, viitattu 13.5.2018.)

Kun hakuehtoon laitetaan parametreiksi starttime, parameters ja place, joka palauttaa liitteen 2 mukaisen XML-sanoman. Alla missä muodossa oheisia parametrejä annetaan hakuehtoon.

Starttime **Alkuaika**

Aikajakson alkuaika ISO-muodossa (esim. 2018-05-13T17:00:00Z)

Parameters **Meteorologiset parametrit**

Meteorologiset parametrit annetaan pilkulla erotettuna mm. lämpötila t2m ja lumi snow_aws

Place **Paikkanimi**

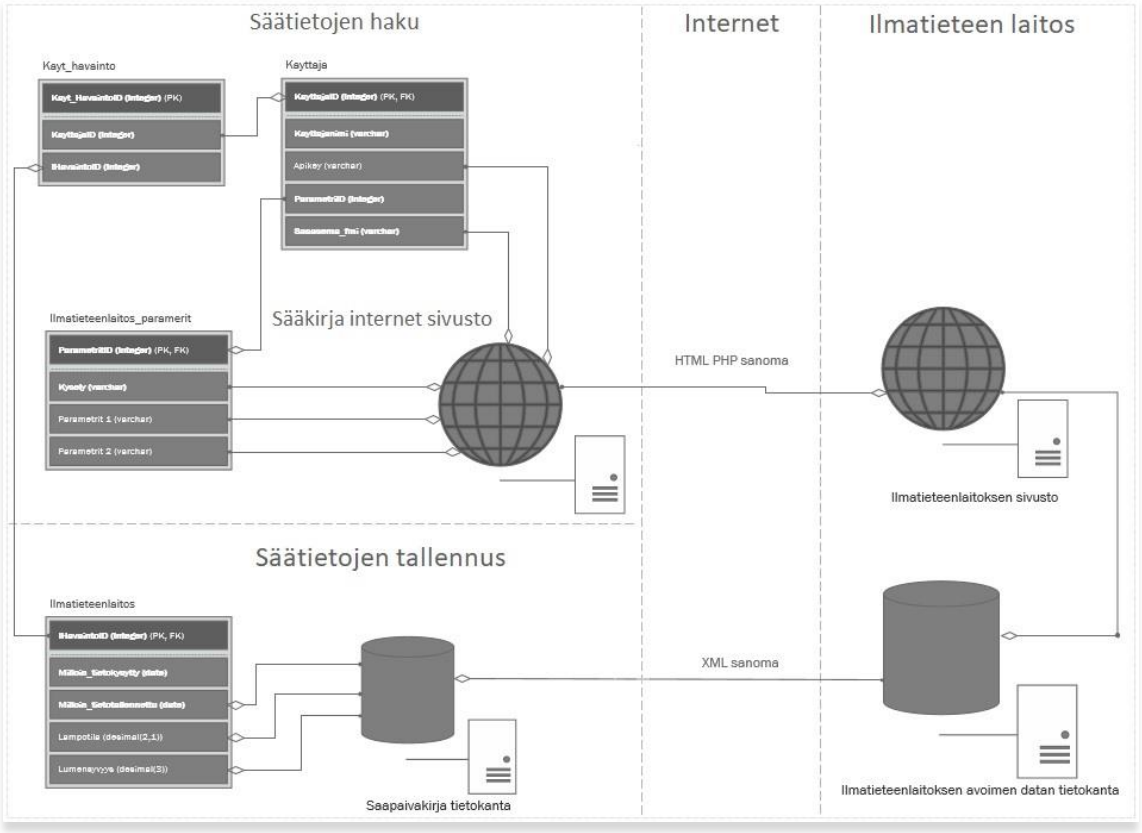
Paikkanimi esim. Kajaani

4.1.2 Säätietojen haku Sääpäiväkirja -tietokantaan

Kuviossa 18 esitetään Sääpäiväkirja -sivustolle tapahtuvan säätietojen haun Ilmatieteen laitoksen tietokannasta. Säätietojen hakuun noudetaan parametritietoja kahdesta eri tietokanta taulusta. Näillä tiedoilla muodostetaan internet sivuston osoite:

*http://data.fmi.fi/fmi-apikey/**APIKEY_PARAMETRI**/wfs?request=getFeature&storedquery_id=**KYSELY_PARAMETRI**&starttime=2018-05-13T14:00&place=**PAIKKAKUNTA_PARAMETRI**&parameters=**PARAMETRI1**,**PARAMETRI2***

Tieto tulee takaisin Ilmatieteen laitoksen tietokannasta XML-formaatissa ja tästä XML -sanomasta parseroidaan kellonaika, lämpötila ja lumensyvyys tiedot. Tietojen parseroinnin jälkeen tiedot tallennetaan Sääpäiväkirjan tietokanta tauluihin.



KUVIO 18. Säätietojen haku Ilmatieteen laitoksen sivustolta

5 KOKONAISUUDEN YHDISTÄMINEN

Kokonaisuuden yhdistäminen on lajiteltu useampaan eri kategoriaan; internet sivustoon, PHP- ja tietokantaan. Internet sivusto -osiossa keskitytään HTML-puolelle. PHP -osiossa keskitytään PHP -koodiin ja siinä esiin tulleisiin ongelmiin. Tietokanta-osiossa keskitytään tietokannan tekemiseen ja mitä ongelmia tuli tietokannan tekemisessä.

Kappaleen 5.1 – 5.3.3 osioita on tehty ja parannettu yhtä aikaa työn edetessä, josta johtuen aikajärjestys ei ole yhtenäinen. Tiedot on kerätty omiin kategorioihin. Kun tekstissä puhutaan tietokanta taulusta, niin tietokannan taulunnimi on alleviivattu selvyyden vuoksi.

5.1 Internet sivusto

Fyysisen internet sivuston ulkonäön tekemisen aloitin wireframe.cc -työkalulla, jolla voidaan piirtää sivujen ”rautalanka” malleja. Liitteessä 3 on wireframe.cc -työkalulla tehty hahmotelma sivustosta, jossa syötetään omat säähavainnot ja millainen seuraava sivu olisi säähavaintojen syöttämisen jälkeen.

Omien säähavaintojen syöttöön on otettu mallia Sääpäiväkirja 2016 / Otavan Kirjapaino Oy Keuruu 2015 kirjasta.

Sivustolla voidaan tallentaa:

- Päiväys ja kellonaika
- Lämpötila yhden desimaalin tarkkuudella esimerkiksi 21.2 astetta (Pakotettu)
- Säätyyppi -pudotusvalikko, josta voidaan valita säätyyppi (Pakotettu)
    - Aurinkoista
    - Pilvistä
    - Vaihtelevaa pilvisyyttä
- Sade -pudotusvalikko, josta voidaan valita sadetyyppi
    - Vesi
    - Lumi
    - Räntä

- Tuulisuus -ruksi
- Muita havaintoja -pudotusvalikko
    - Ukkosta
    - Sumua
    - Rakeita
- Omat havainnot tekstikenttä, joka on 100 merkkiä pitkä

Ensimmäisissä sivuissa ei käytetty muotoilua ollenkaan, vaan sivusto oli tässä vaiheessa hyvin pelkistetty. Liitteessä 4 on IE -selaimella ja mobiililaitteella omien havaintojen tallennus sivu. Kun omat havainnot on syötetty ja tiedot tallennetaan, tallennuksen yhteydessä haetaan säätiedot Ilmatieteen laitoksen palvelusta.

Tässä vaiheessa sivusto ei ollut kovin modernin näköinen, mutta se toimi hyvin, koska sitä ei tarvinnut lainkaan skaalata vaan kaikki tieto mahtui yhdelle sivulle. Tietokanta ja PHP osuus alkoi olla tässä vaiheessa valmiina, joten päätin uudistaa ja parantaa myös sivuston ulkonäön paremman näköiseksi.

W3Schools 2018a (Viitattu 14.7.2018) sivuilta sain malleja ja ideoita, miten sivustosta tehdään skaalautuva ja moderni. Tässä yhteydessä myös sivujen määrä kasvoi ja sivuja tulikin yhteensä 6 kpl (katso liite 5).

Sivut:

- Aloitussivu
- Omien säähavaintojen tallennus
    - Säätiedot tallennettu (tallennuksen jälkeen tuleva sivu)
- Hakutoimintoja
    - **Omat tallennukset** haku palauttaa omat tallennukset ja samalla hetkellä haetut Ilmatieteen laitoksen datan säätiedot. Oletuksena haetaan 5 viimeisintä tallennusta, mutta hakua voidaan pudotusvalikosta lisätä jopa 50:een tallennukseen saakka.

    - **Ilmatieteen laitos nyt** haku palauttaa haetut säätiedot ja samalla haetaan sen hetkinen viimeisin tallennus Ilmatieteen laitoksen datasta. Oletuksena haetaan 5

viimeisintä, mutta hakua voidaan pudotusvalikosta jopa 50:een tallennukseen saakka.

- **Omat vs. Ilmatieteen laitos** haku palauttaa omat tallennukset ja samalla Ilmatieteen laitoksen datasta tallennettu säätiedot. Tästä hausta näkee, kuinka paljon esimerkiksi oma lämpömittari eroaa Ilmatieteen laitoksen viralliseen mittauspisteen lukemasta. Oletuksena haetaan 5 viimeisintä, mutta hakua voidaan pudotusvalikosta lisätä jopa 50:een tallennukseen saakka.

W3Schools 2018b, W3Schools 2018c, W3Schools 2018d & W3Schools 2018e (Viitattu 14.7.2018) sivuilta sain ohjeita ja malleja Bootstrapin ominaisuuksista valikoissa, tietojen syöttämisessä, painikkeissa, pudotusvalikoissa ja taulukoissa.

Kun tein sivustolle uusia sivuja, joissa voidaan hakea vanhempia säätietoja, eteen tuli tilanne, että kuinka tuulisuus haetaan tietokannasta, kun tieto on 1 tuulee ja 0 ei tuule. Päädyin siihen, että tuulisuudesta voidaan kertoa enemmän, joten tuulelle tuli oma pudotusvalikko, mistä voidaan valita, onko tuuli tyyntä, heikkoa, kohtalaista, kovaa vai myrsky.

Omien säähavaintojen tallennuksen yhteydessä huomasin, ettei aina saatu haettua Ilmatieteen laitoksen tiedoista viimeisintä havaintotietoa, vaan haku palautti edellisen tallennuksen.

W3Schools 2018g & W3Schools 2018f (Viitattu 14.7.2018) sivuilta sain mallia tähän kohtaan. Omien tallennusten jälkeen tulee kohta wait -komentoparametri, jossa odotetaan 5 sekuntia, ennen kun haetaan tietokannasta Ilmatieteen laitoksen säähavainto. Tämä aluksi oli toteutettu PHP:n sleep5 parametrillä, mutta tästä ei kerrota käyttäjälle mitään, vaan hän joutuu odottamaan. Löysin sen jälkeen CSS:llä tehdyn lataus ikkunan, jossa pyörii ympyrä 5 sekuntia, ennen kuin siirrytään seuraavalle sivulle.

Tämän jälkeen huomasin, että tallennettaessa lämpötilaa omiin säähavaintoihin, lämpötilan desimaalit voidaan erotella joko pilkulla tai pisteellä. Näillä onkin eri vaikutus tietokantaan tallennuksen yhteydessä. Kun käytetään pistettä erottimena, tieto tallentuu tietokantaan oikein, mutta taas pilkkua käytettäessä tallentuu pelkkä alkuluku ja pilkun jälkeistä tietoa ei tallenneta.

W3Schools 2018h (Viitattu 14.7.2018) sivulta löysin ratkaisun, että muutetaan inputissa oleva type arvo number -muotoon, tämä arvo oli aikaisemmin text- muodossa. Muutoksen jälkeen pystyi kirjoittamaan pisteen ja pilkun niin, että tieto tallentui oikein tietokantaan. Kun mobiililaitteella syötetään lämpötilaa, tulee pelkästään numeronäppäimistö käyttöön.

5.2 PHP

Banas 2018 (Viitattu 7.4.2018) sivustolta aloitin tutustumaan PHP:hen. PHP MySQL Tutoriaalissa tehdään perus sivusto, jossa tallennetaan ja haetaan tietoja tietokannasta taulukkoon.

Matias.biz 2018 (Viitattu 6.5.2018) sivustolla kerrotaan Ilmatieteen laitoksen avoimesta datasta ja sen hyödyntämisestä. Tästä kerrottiin todella hyvin ja selkeästi, kuten esimerkiksi miten muodostetaan PHP:lla sivulinkki ja miten sivulinkkiin lisätään parametrejä, joiden muuttujat saadaan PHP -koodista.

ekurssit 2018 (Viitattu 7.6.2018) sivustolta sain neuvoja, miten tehdään lomakkeelta tietojen tallentaminen tietokantaan. Tietojen tallennukseen käytetään (katso kuvio 19) $_POST arvoa INSERT lausekkeessa. Sivustolla oleva INSERT esimerkki ei toiminut suoraan, koska se ei tallentanut kaikkia tietoja tietokantaan, mm. kellonajan tallentamista en saanut onnistumaan. Lisäksi NetBeans -sovellus antoi koko ajan virheilmoituksia näistä INSERT riveistä.

The Programming Geek 2018 (Viitattu 23.6.2018) sivustolta sainkin tähän hyvän mallin ja idean. Aikaisemmin sivustolle tallennetuissa tiedoissa ei käytetty muuttujia, vaan tieto oli suoraan INSERT -lausekkeessa (katso kuvio 19). Kuviossa 20 on myöhemmin tehty malli, jossa tallennus tehdäänkin suoraan muuttujaan. Tämän jälkeen muuttujan tiedot lisätään INSERT -lausekkeeseen. Näin sain kellonajan onnistuneesti tallentumaan ja tässä ei tarvitse myöskään miettiä UPDATE -lausekkeen tekemistä, koska vapaateksti sarakkeen arvo haetaan SELECT:llä viimeisimmästä Vapaateksti tallennuksesta.

```
$sql = 'INSERT INTO sov_havainto VALUES ('.
"\"{$_POST['lampotila']}\", " .
"\"{$_POST['saatyyppi']}\", " .
"\"{$_POST['sadetta']}\");";
```

KUVIO 19. Tietojen tallennus lomakkeelta ensimmäinen malli

```
$lampotila = ($_POST['lampotila']);
$saatyyppi = ($_POST['saatyyppi']);
$sadetta = ($_POST['sadetta']);

$sql .= "INSERT INTO sov_havainto () VALUES (NULL, LOCALTIMESTAMP, '$lampotila',
    '$saatyyppi', '$sadetta', '$tuulisuus', '$muita_havaintoja',
    (SELECT max(VapaaID) AS VapaaID FROM vapaateksti));";
```

KUVIO 20. Tietojen tallennus lomakkeelta muuttujalla

Tietojen tallennus lomakkeelta aiheutti useamman eri INSERT -lausekkeen, koska tietoja tallennetaan neljään eri tauluun Ilmatieteenlaitos, Vapaateksti, Sov_havainto ja Kayt_havainto. Tämä aiheutti aluksi ongelmaa, että kuinka saadaan useampi INSERT rivi tehtyä.

W3Schools 2018j (Viitattu 23.6.2018) sivulta tähän löytyi hyvä malliesimerkki. Aikaisemmin INSERT -lausekkeet olivat peräkkäisillä riveillä, mutta tallennus onnistui ainoastaan ensimmäisestä INSERT:stä. PHP:ssä on käytössä MySQL INSERT MULTIPLE, kun halutaan tehdä useampia tietokanta lausekkeita peräkkäin. Kuviossa 21 on tehty useita INSERT -lausekkeita peräkkäin käyttäen MySQL INSERT MULTIPLE parametriä, että ensimmäisestä toiseksi viimeiseen lausekkeen lopussa on puolipiste lainausmerkki puolipiste **;";**. Viimeisessä on pelkästään puolipiste. Samoin toisesta lausekkeesta eteenpäin lauseen alussa on **.=** eli **$sql .= "**, kun ensimmäisessä on pelkästään **$sql = "**.

```php
// Muodotetaan tietokanta yhteys
$conn = new mysqli($config['servername'], $config['username'], $config['password'], $config['dbname']);

// Tarkistetaan tietokanta yhteys
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// Tehdään tietokantaan tallennukset
$sql = "INSERT INTO ilmatieteenlaitos () VALUES (NULL, LOCALTIMESTAMP, '$aika', '$lampo', '$lumi');";
$sql .= "INSERT INTO vapaateksti () VALUES (NULL, '$omat_havainnot');";
$sql .= "INSERT INTO sov_havainto () VALUES (NULL, LOCALTIMESTAMP, '$lampotila',
'$saatyyppi', '$sadetta', '$tuulisuus', '$muita_havaintoja',
(SELECT max(VapaaID) AS VapaaID FROM vapaateksti));";
$sql .= "INSERT INTO kayt_havainto () VALUES (NULL, 1,
(SELECT max(SHavaintoID) AS SHavaintoID FROM sov_havainto),
(SELECT max(IHavaintoID) AS IHavaintoID FROM ilmatieteenlaitos))";

if ($conn->multi_query($sql) === TRUE) {
    } else {
        echo "Error: " . $sql . "<br>" . $conn->error;
    }

// Suljetaan tietokanta yhteys
$conn->close();
```

KUVIO 21. Useampi tietokanta lauseke peräkkäin

Tähän saakka sivustolle on tallennettu tietoja, eikä Ilmatieteen laitoksen avoimesta datasta ole haettu mitään tietoja, vaan testissä on muuttujassa ollut valmiiksi tallennetut arvot.

Laaksonen 2014 (Viitattu 29.6.2018) sivulta sain tähän hyvän mallin ja idean, jossa käytetään PHP:n PREG_MATCH_ALL ominaisuutta. Siinä tallennetaan FILE_GET_CONTENTS ominaisuudella HTML-sivun arvo muuttujaan. Tämän jälkeen otetaan muuttujiin arvot, joita XML datasta halutaan ja PREG_MATCH_ALL ominaisuudella ajetaan muuttujien tiedot FOR silmukalla, kuviossa 22 käytetään näitä ominaisuuksia lumensyvyyden hakuun Ilmatieteen laitoksen avoimen datan XML-sanomasta.

```php
// Haetaan lumen syvyys
$data1 = file_get_contents("http://data.fmi.fi/fmi-apikey/" . $options["apikey"] .
"/wfs?request=getFeature&storedquery_id=" . $options["stored_query_id"] . "&starttime=" . $options["starttime"] .
"&place=" . $options["place"] . "&parameters=" . $options["parameters2"]);
$parametervalue = "<BsWfs:ParameterValue>(.*?)<\/BsWfs:ParameterValue>";
preg_match_all("/$parametervalue/", $data1, $arvo1);
for ($i = 0; $i < count($arvo1[0]); $i++) {
    if ($arvo1[1][$i] == "") {
    } else if ($arvo1[1][$i] != "") {
        $lumi = "{$arvo1[1][$i]}";
        if ($lumi > 0) {
            // Jos lunta on enemmän kuin 0 niin syötään oikea arvo, muuten syötetään 0. Kun lunta ei ole, tulee paluu arvona -1
        } else {
            $lumi = 0;
        }
    }
}
```

KUVIO 22. Ilmatieteen laitoksen datan haku

PREG_MATCH_ALL ominaisuudella tietojen haku onnistui todella hyvin. Kyseinen haku tehdään kaksi kertaa eri parametreillä. Ensimmäisessä haussa haetaan kellonaika ja lämpötila viimeisestä tallennuksesta, toisella haulla haetaan pelkästään lumensyvyys. Kesäaikaan lumensyvyys tulee arvolla -1 Ilmatieteen laitoksen avoimesta datasta, joten tämän johdosta IF lauseke on tehty tallennukseen, jos arvo on enemmän kuin 0 niin tallennetaan tuleva arvo, mutta jos arvo on pienempi kuin 0 niin tallennetaan 0.

Tämän jälkeen tuli haaste, miten muuttujien arvoja laitetaan sivulinkkiin.
Matias.biz 2018 (Viitattu 6.5.2018) sivulta löytyi tähän hyvä ratkaisu. Kuviossa 22 muuttujat asetetaan $OPTIONS[”Arvo URL lausekkeessa”] parametreillä sivulinkkiin. Tässä muodostetaan muuttujan $DATA1 internet sivun sivuosoite. Muuttujista haetut parametrit ovat apikey (apikey), kysely (stored_query_id), aloitusaika (starttime), sääasema (place) ja lumensyvyys (parameters2).

Kun Ilmatieteen laitoksen datasta alkoi tietojen haku toimimaan, uudeksi haasteeksi tuli, että miten aika kerrotaan hakuun, ettei ole kiinteä aika parametri. Jos kiinteää aikaparametriä käyttäisi, niin joka päivä hakuun lisääntyy päivän verran uusia hakutuloksia. Tämä ratkesi siten, että oletus aikavyöhykkeeksi muutin Atlantic Azores, missä kello on -4 tuntia. Ilmatieteen laitoksen datassa kello on UTC ajassa ja Suomessa on kesä- ja talviaika käytössä. Tällöin haku ei tee pitkää hakua taaksepäin vaan enintään 4 tuntia. Kuviossa 23 on aika asetuksien muutoksia ja lopuksi palautetaan takaisin Helsingin aikavyöhyke.

```
// Muutetaan aika asetukseksi -4 tuntia
date_default_timezone_set('Atlantic/Azores');

// Otetaan päiväys ja tunti aika
$paivanyt = date("Y-m-d");
$kellonyt = date("H");
$kirjain = "T";

// Tehdään yhteen pötköön päiväys, lisätään T kirjain, tunti ja lisään minuuteksi 00.
$options["starttime"] = "$paivanyt$kirjain$kellonyt:00";

// Palautetaan oikea aikavyöhyke
date_default_timezone_set('Europe/Helsinki');
```

KUVIO 23. Kellonaika asetus

Kuviossa 23 on käytetty muuttujaa $PAIVANYT, jossa päiväys tulee muotoon vuosi-kuukausi-päivä ja sen jälkeen kellonaika tulee muodossa tunnit. Päiväyksen ja kellonajan erittelee kirjain T sivulin-

kissä. Tälle tuli oma muuttuja, koska muutoin olisi ollut välilyönti ennen ja jälkeen T-kirjaimen. Muuttujat yhdistetään tämän jälkeen ja loppuun laitetaan :00 niin saadaan tasatunnilta aloitus. Tämän jälkeen päiväys ja kellonaika esitetään muodossa 2018-08-05T14:00.

Seuraavaksi tuli ongelma ajan kanssa. Miten saadaan Ilmatieteen laitoksen datasta palautuva aika muutettuna takaisin Suomen aikaan, kun kesäaika on -3 tuntia.

Stack Overflow 2018a (Viitattu 9.7.2018) sivustolta löytyi tähän lopulta ratkaisu, kuinka aika muotoillaan (katso kuvio 24), tässä saadaan XML-datasta kellonaika, johon käytetään PHP:n STRTOTIME ominaisuutta muuttamaan aika oikeaan muotoon.

```
// Konvertoidaan tuleva UTC aika Helsingin ajan muotoon
$aika_muutos = strtotime("{$arvo[1][$i]}");
$aika = date("Y-m-d H:i:s", $aika_muutos);
```

KUVIO 24. Aika muutos

Ilmatieteen laitoksen haussa tarvittavia muuttujia ei ole kovakoodattu, koska tarvittaessa muuttujien tietoja voitaisiin muuttaa, kun nämä haetaan tietokannasta. Muuttujan tietojen hakuun tietokannasta käytetään PHP:n MYSQL_QUERY ja MYSQL_FETCH_ARRAY ominaisuuksia. MYSQL_FETCH_ARRAY ominaisuutta käytetään myös, kun haetaan SELECT:llä tietoja tauluk-koon.

Lintulaakso 2004 & Banas D 2018 (Viitattu 7.4.2018) löytyi hyvä ratkaisu, miten tietokannasta muuttujien tiedot haetaan PHP -muuttujiin. Kuviossa 25 haetaan Kayttaja ja Ilmatieteenlaitos_parametrit tauluista muuttujat.

```
// Tietokannasta tarvittavat parametrit
$query = "select apikey, saaasema_fmi, parametri1, parametri2, kysely from kayttaja
inner join ilmatieteenlaitos_parametrit on kayttaja.ParametriID = ilmatieteenlaitos_parametrit.ParametriID";
$response = mysqli_query($conn, $query);
if ($response) {
    while ($rivi = mysqli_fetch_array($response)) {
        $options["apikey"] = $rivi["apikey"];
        $options["place"] = $rivi["saaasema_fmi"];
        $options["stored_query_id"] = $rivi["kysely"];
        $options["parameters1"] = $rivi["parametri1"];
        $options["parameters2"] = $rivi["parametri2"];
    }
```

KUVIO 25. Muuttujat tietokannasta

Laaksonen 2011 & W3Schools 2018i (Viitattu 16.7.2018) sivustoilta löytyi hyvin tietoa tietoturvaan lomakkeen käsittelyyn ja näiden pohjalta muutin $_POST kohdassa lomakkeelta tietojen tallennusta. Lisäsin alkuun HTMLSPECIALCHARS arvon, jolla saadaan estettyä HTML-muotoilu eikä JavaScript koodia pystytä ajamaan ja loppuun laitoin ENT_QUOTES. Tällä saadaan muutettua esimerkiksi &QUOT arvo muotoon ” lainausmerkki (katso kuvio 26).

```
$lampotila = htmlspecialchars($_POST['lampotila'], ENT_QUOTES);
$saatyyppi = htmlspecialchars($_POST['saatyyppi'], ENT_QUOTES);
$sadetta = htmlspecialchars($_POST['sadetta'], ENT_QUOTES);
```

KUVIO 26. Tietoturvaa lomakkeelta luvussa

Tietoturvaa on lisätty tietokantayhteyden muodostukseen, koska aikaisemmin tietokannan luonti lauseessa oli käyttäjätunnukset ja salasanat selväkielisinä.

maxisme 2017 (Viitattu 16.7.2018) löytyi tähän hyvä ratkaisu. Kuviossa 27 muodostetaan tietokanta yhteys kuvion 28 muuttujilla. Tässä käytetään PHP:n PARSE_INI_FILE ominaisuutta, jossa kerrotaan tiedoston nimi, josta halutaan etsiä halutut arvot. PHP -lauseessa $CONFIG[’arvo’] kerrotaan mikä arvo tiedostosta laitetaan tähän kohtaan.

```
// Muodotetaan tietokanta yhteys
$config = parse_ini_file('tietokanta.ini');
$conn = mysqli_connect($config['servername'], $config['username'], $config['password'], $config['dbname'])
    OR die('Could not connect to MySQL: ' .
        mysqli_connect_error());
```

KUVIO 27. Tietokanta yhteys

```
servername=localhost
username=saakirja
password=salasana
dbname=saapaivakirja
```

KUVIO 28. Tietokanta.ini tiedoston sisältö

Lopuksi tein sivustolle vielä kolme erilaista haku toimintoa. Näistä oman säätietojen tallennuksessa tallennetun lämpötilan ja Ilmatieteen laitoksen tiedoista samalla haetun lämpötila arvon vertailu tuotti ongelmia. Miten saadaan toteutettu haku, koska Sov_havainto ja Ilmatieteenlaitos tauluissa

lämpötilalle on sama sarakkeen arvo lampotila. Tämä toimi normaalisti phpMyAdminissa, kun tehdään SELECT useampaan tauluun (katso kuvio 29). Mutta kun sama SELECT tehdään PHP:lla nii tulee virhe ilmoitus tallennettaessa MYSQL_FETCH_ARRAY toiminnolla tietoja taulukkoon.

```
SELECT sov_havainto.paivays, sov_havainto.lampotila as SovLampo, ilmatieteenlaitos.Lampotila as IlmaLampo
from kayt_havainto
inner join sov_havainto on kayt_havainto.SHavaintoID=sov_havainto.shavaintoid
inner join ilmatieteenlaitos on kayt_havainto.IHavaintoID=ilmatieteenlaitos.IHavaintoID
```

KUVIO 29. Lämpötilan haku kahdesta eri taulusta

MySQL Reference Manual 2018a & MySQL Reference Manual 2018b (Viitattu 16.7.2018) sivustoilta sain tähän hyvän ohjeen. Tehdään ensin näkymä SELECT:llä ja ennen näkymän tekoa tehdään näkymän poisto, ettei tule ongelmia sen suhteen, jos ennestään on olemassa vanha näkymä. Näkymän luonnissa päädyin antaa Omat vs Ilmatieteen laitoksen näkymän nimeksi omavsilma. PHP -koodissa ensiksi tehdään DROP VIEW omavsilma ja tämän jälkeen tehdään näkymä CREATE VIEW komennolla (katso kuvio 30).

```
// Tehdään tietokantaan näkymän poisto ja sitten tehdään näkymä
$sql = "drop view omavsilma;";
$sql .= "create VIEW omavsilma as SELECT sov_havainto.paivays, sov_havainto.lampotila as SovLampo,
    ilmatieteenlaitos.Lampotila as IlmaLampo from kayt_havainto
    inner join sov_havainto on kayt_havainto.SHavaintoID=sov_havainto.shavaintoid
    inner join ilmatieteenlaitos on kayt_havainto.IHavaintoID=ilmatieteenlaitos.IHavaintoID ";
```

KUVIO 30. Näkymän tekeminen

Näkymän teon jälkeen pystyi tekemään SELECT:n näkymään omavsilma ja nämä tiedot saatiin haettua taulukkoon samalla tavalla kuin muissakin haku toiminnoissa. Näkymässä Sov_havainto ja Ilmatieteenlaitos taulujen lämpötila arvot ovat eri nimisissä sarakkeissa Sov_havainto on SovLampo ja Ilmatieteenlaitos IlmaLampo.

Oletuksena näissä hauissa sivuille haettiin enintään 5 viimeisintä tallennusta. Tämän kuitenkin muutin siten, että voidaan nähdä enemmänkin vanhoja hakuja. Hakuun tuli valittavaksi 5, 10, 15, 25 tai 50 viimeisintä riviä. Oletuksena näytettävä määrä on 5 kpl, josta tein muuttujan ja valikon tälle samalla tavalla kuin omat tallennukset sivulla, kun valitaan esimerkiksi tuulisuus. Valikko toimii mutta, määrä ei muuttunut, kun valittiin joku muu arvo.

Smith F 2008 (Viitattu 17.7.2018) sivulta löytyi tähän hyvä vinkki eli tehdään muuttuja, joka on SELECT ja muuttujan arvo laitetaan tähän. Tässä tapauksessa muuttuja on $MAARA, ja tämän jälkeen ajetaan QUERY komennolla tämä muuttuja (katso kuvio 31). Tämän jälkeen haku muutos onnistui.

```
$maara1 = "select paivays, SovLampo, IlmaLampo from omavsilma order by paivays desc limit $maara";

$query = "$maara1";
```

KUVIO 31. Muuttuja SELECT:ssä

Hakutoimintoja tehdessäni huomasin ongelman siinä, kun omat tallennuksessa tehdään valintoja esimerkiksi tuulisuus tyyntä (katso liite 5). Haussa omat tallennukset ja samalla hetkellä haetut Ilmatieteen laitoksen säätiedot haussa tuulisuuden tyyntä sanassa ä kirjain on ? merkkisenä. Mutta kuitenkaan omat havainnot palauttaa normaalisti ä kirjaimet. Kaikilla sivuilla oli määriteltynä HTML-koodissa <META CHARSET=”UTF-8”> ja tietokanta on asennettu oletusarvoilla.

Rytkönen I 2018 (Viitattu 19.7.2018) keskustelufoorumilta löytyi tähän hyvä vinkki, että tietokanta yhteyden muodostuksessa ajetaan myös tietokanta yhteysmuuttujalle SET_CHARSET(”UTF8”) (katso kuvio 32). Tämän jälkeen ääkköset tulivat normaalisti.

```
$conn = new mysqli($config['servername'], $config['username'], $config['password'], $config['dbname']);
$conn->set_charset("utf8");
```

KUVIO 32. Tietokanta merkkiarvo laitetaan utf-8 muotoon

Tietokannan kanssa oli myös ongelmana, kun PHP:n alussa luodaan tietokanta yhteys ja tehdään tarvittavat tietokanta toiminnot ja yhteys suljetaan. Kun samassa koodissa avataan uudestaan tietokantayhteys, niin yhteyden muodostaminen ei enää toimi $CONN muuttujalla, vaan täytyy tehdä yhteys uuteen muuttujaan, jotta se toimii. Jos sivulla kerran on käytetty $CONN muuttujaa, niin seuraava tietokanta yhteys täytyi avata $CONN1 muuttujalla.

Päädyin vielä muokkaamaan pääsivua (katso kuvio 33) ja lisäämään alkuun Ilmatieteen laitokselta haun, jossa haetaan lämpötila, kosteus, kastepiste, edeltävän tunnin sademäärä, tuulen nopeus, tuulen suunta, tuulen puuska, paine, näkyvyys ja lumensyvyys. Näiden tiedot tulevat suoraan XML-sanomassa, jos ei määritellä ollenkaan parametrejä. Sivustolle kuitenkin määrittelin säätietojen haun yksi kerrallaan Ilmatieteen laitoksen tiedoista, ettei tietoja parsita kerralla XML-sanomasta,

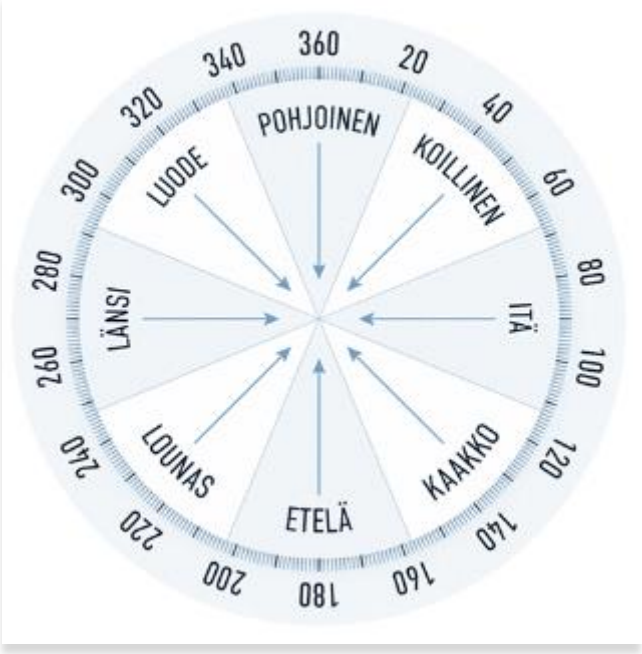
tällöin pystyin käyttämään osittain hyväksi vanhoja hakutoimintoja ja samalla pystyi muokkaamaan haun tulosta.

Sää Kajaanin Petäisenniskassa 19.08.2018 15:50

Lämpötila	19.4 °C	**Kosteus**	45.0 %
Kastepiste	7.1 °C	**Edeltävän tunnin sademäärä**	0.0 mm (15:00)
Tuulen nopeus	kohtalaista 4.4 m/s	**Tuulen suunta**	länsituulta (257.0°)
Tuulen puuska	navakkaa 8.9 m/s	**Paine**	1001.8 hPa
Näkyvyys	50 km	**Lumensyvyys**	- cm

KUVIO 33. Säätietojen haku (Ilmatieteen laitos 2018f, viitattu 19.8.2018)

Ilmatieteen laitos 2018g (Viitattu 19.8.2018) sivuilta löytyi hyvää tietoa tuulista ja tuulen suunnista. Kuviossa 34 on tuulen suunnat asteluvuittain. Suunnan lisäksi halusin vielä laittaa aloitussivulle tuulen nopeuden ja puuska tietoon millainen tuuli on kyseessä (katso kuvio 35).



KUVIO 34. Tuulen suunnat ja asteluvut (Ilmatieteen laitos 2018g, viitattu 19.8.2018.)

Nopeus	Nimitys
0 m/s	tyyntä
1 - 3 m/s	heikkoa tuulta
4 - 7 m/s	kohtalaista tuulta
8 - 13 m/s	navakkaa tuulta
14 - 20 m/s	kovaa tuulta
21 - 32 m/s	myrskyä
yli 32 m/s	hirmumyrskyä

KUVIO 35. Tuuliasteikko 10 minuutin keskituulen nopeuksille (Ilmatieteen laitos 2018g, viitattu 19.8.2018.)

Tuulensuunta tuotti aluksi jonkin verran ongelmia, miten määritellään mistä tuuli puhaltaa. Eniten tuotti haasteita pohjoistuuli, koska pohjoistuuli on asteluvulla 338 – 22. Asteet loppuvat 360 ja alkaa tämän jälkeen 0 ja loppuu 22 asteeseen. Pohjoistuulen laitoin aluksia kokonaan muodossa $SUUNTA >= 338 && $SUUNTA <= 360 && $SUUNTA <= 22, mutta tämä ei toiminut oikein. Kun pohjoistuuli jaetaan kahteen eri ELSEIF -lausekkeeseen, niin silloin saadaan oikea tulos, tämän sain ratkaistua (katso kuvio 36).

```
if ($suunta >= 23 && $suunta <= 68) {
    $suuntanimi = "koillistuulta";
} elseif ($suunta >= 69 && $suunta <= 112) {
    $suuntanimi = "itätuulta";
} elseif ($suunta >= 113 && $suunta <= 158) {
    $suuntanimi = "kaakkoistuulta";
} elseif ($suunta >= 159 && $suunta <= 202) {
    $suuntanimi = "etelätuulta";
} elseif ($suunta >= 203 && $suunta <= 248) {
    $suuntanimi = "lounaistuulta";
} elseif ($suunta >= 249 && $suunta <= 292) {
    $suuntanimi = "länsituulta";
} elseif ($suunta >= 293 && $suunta <= 337) {
    $suuntanimi = "luoteistuulta";
} elseif ($suunta >=338 && $suunta <=360) {
    $suuntanimi = "pohjoistuulta";
} elseif ($suunta >=0 && $suunta <=22 ) {
    $suuntanimi = "pohjoistuulta";
}
```

KUVIO 36. Tuulen suunnan nimen määrittely

Kun sain ratkaistua tuulensuunnan, niin tuulen nopeus oli tämän jälkeen helppo toteuttaa samalla IF ELSEIF rakenteella (katso kuvio 37). Tämän tein myös puuskaiselle tuulelle, jossa arvot ovat samat, mutta muuttujat ovat $PUUSKA ja $PUUSKANIMI.

```
if ($nopeus < 1) {
    $nopeusnimi = "tyyntä";
    } elseif ($nopeus >= 1 && $nopeus < 4) {
        $nopeusnimi = "heikkoa";
    } elseif ($nopeus >= 4 && $nopeus < 8) {
        $nopeusnimi = "kohtalaista";
    } elseif ($nopeus >= 8 && $nopeus < 14) {
        $nopeusnimi = "navakkaa";
    } elseif ($nopeus >= 14 && $nopeus < 21) {
        $nopeusnimi = "kovaa";
    } elseif ($nopeus >= 21 && $nopeus <= 32) {
        $nopeusnimi = "myrskyä";
    } elseif ($nopeus > 32) {
        $nopeusnimi = "hirmumyrsky";
    }
```

KUVIO 37. Tuulen nopeuden nimen määrittely

Tietoja tallentaessa huomasin, että näkyvyyteen tulee desimaalit esimerkiksi 49.424 km. Tämä ei ole kauhean selkeä ilmaisu näkyvyydelle, joten tämän pyöristin lähimpään kokonaislukuun. Tässä käytin hyödyksi PHP:n ROUND() funktiota. Mutta tuulessa ajattelin, että desimaalit voi olla käytössä, enkä käytä PHP:n ROUND() funktiota.

Siirsin sivuston web-hotelliin, jotta sitä on helpompi käyttää. Siirron kanssa tuli muutamia haasteita vastaan. Kellonaika haut eivät toimi samalla tavalla kuin XAMPP:ssa, myöskään tietokanta lauseet eivät menneet läpi, mihin tietokannan salasana tiedosto tallennetaan web-hotellissa ja näkymän luonnissa/poistossa tuli ongelmia.

Ongelman kellonajassa sain korjattua, kun käytin eri muuttujaa ajan määrittelyssä. Aikaisemmin käytin DATE_DEAULT_TIMEZONE_SET muuttujaa, mutta tämä täytyi muuttaa muotoon DATE_CREATE, jonka jälkeen aika alkoi toimimaan oikein.

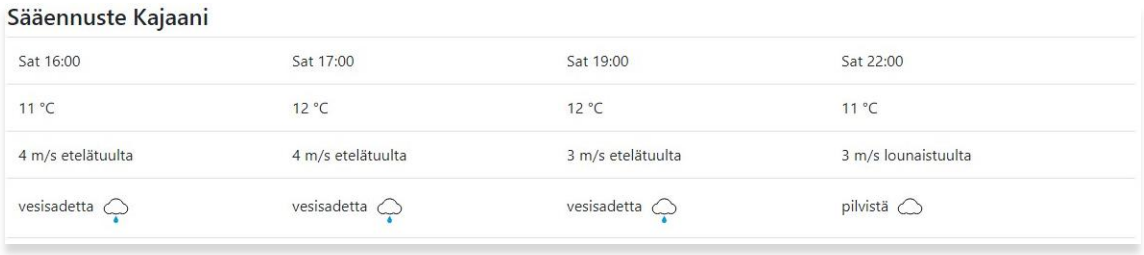
Tietokannan teossa kävi ilmi, ettei, joka kohtaan tullut sarakkeen nimeä kirjoitettu samalla tavalla, vaan osassa kohtaa olin kirjoittanut isolla alkukirjaimella ja toisessa pienellä alkukirjaimella saman kohdan. Nämä piti muuttaa samalla tavalla joka kohtaan.

Tietokanta lauseissa oli myös sama ongelma, että nämä piti muuttaa samalla tavalla, joka kohtaan ja tämän jälkeen haut toimivat normaalisti.

Tietokannan salasanan ja käyttäjätunnuksen tallensin tekstitiedostoon ja siirsin web-hotellissa sellaiseen paikkaan, ettei se ole jaetussa kansiossa.

Alussa näkymän kanssa ongelmana oli, että näkymä poistetaan ensimmäisenä, ennen kuin luodaan uusi näkymä. Tämän kanssa tuli sellainen ongelma, ettei näkymää voitu poistaa, koska sellaista ei vielä ollut ja haku pysähtyi tähän. Tämän korjaus onnistui helposti siten, että tehdään ihan aluksi näkymä käsin phpMyAdmin sivustolla ja tämän jälkeen Oma vs Ilmatieteen laitos haku alkoi toimimaan.

Sivun alku muokkautui vielä vähän lisää, koska halusin lisätä vielä sääennustuksen. Tämä saadaan myös Ilmatieteen laitoksen avoimesta datasta, kun haetaan kyselyllä *fmi::forecast::hirlam::surface::point::simple*. Haussa tiedot ovat alkavan tunnin mukaan ja hausta halusin ottaa mukaan lämpötilan, tuulen, tuulensuunnan ja säätilan. Sääennustus haetaan nykyinen tunti, tunnin päästä, kolmen tunnin päästä ja kuuden tunnin päästä (katso kuvio 38).



KUVIO 38. Sääennuste Kajaani

Ilmatieteen laitos, 2018b (Viitattu 12.9.2018) sivulta löytyi käytettyjen sääsymbolien selitykset ja vapaasti käytettävät sääsymbolit GitHub sivustolta. Tiedot haetaan samalla tavalla kuin kuviossa 22, mutta kysely on *fmi::forecast::hirlam::surface::point::simple* ja haku parametreinä käytetty weathersymbol3, winddirection, windspeedms ja temperature. Alussa ongelmaa tuotti, että millä tavalla sääsymbolin kuva tuodaan PHP:ssa. Tämä ratkesi, että muuttujaan laitetaan HTML polku, josta kuva löytyy (katso kuvio 39). XML-sanomassa tulee säätyypin numero, joka kertoo, millainen sää on ennustuksessa. Kuviossa 39 on säätyyppien tietoa, esimerkiksi numero 3 pilvistä ja sen sääsymboli on 3.svg kuva. Tuulen suunta on tehty samalla tavalla, kuin aikaisemmin kuviossa 36 tuulensuunta.

```
if ($symbol == 1) {
 $symbolnimi = "selkeää";
 $kuva = '<img src="saasymbol/1.svg"/>';
} elseif ($symbol == 2) {
 $symbolnimi = "puolipilvistä";
 $kuva = '<img src="saasymbol/2.svg"/>';
} elseif ($symbol == 3) {
 $symbolnimi = "pilvistä";
 $kuva = '<img src="saasymbol/3.svg"/>';
} elseif ($symbol == 21) {
 $symbolnimi = "heikkoja sadekuuroja";
 $kuva = '<img src="saasymbol/21.svg"/>';
```

KUVIO 39. Sääsymbolin, vallitsevan sään nimi ja sään numerotieto

Kun sain oikean sääsymbolin haettua, niin tämän jälkeen piti saada aloitusaika muokattua oikeanlaiseksi. Tämä onnistui DATE muuttujalla (katso kuvio 40). Siinä saadaan kellonaikaa siirrettyä eteen- ja taaksepäin, jotta haku tuottaa halutun tuloksen.

```
$date=date_create("UTC");

$paivanyt = date_format($date,"Y-m-d");
$kellonyt = date_format($date,"H");
$kirjain = "T";

$paivayhdistelma = "$paivanyt$kirjain$kellonyt:00";
$alku = date("H:i",strtotime(date($paivayhdistelma). " -10 minutes"));
$loppu = date("H:i",strtotime(date($paivayhdistelma). " +10 minutes"));

$alku1 = "$paivanyt$kirjain$alku";
$loppu1 = "$paivanyt$kirjain$loppu";
```

KUVIO 40. Kellonajan siirto

5.3 Tietokannan tekeminen

5.3.1 Mitä tietoja tietokannassa olisi

Kuviossa 41 on ensimmäinen hahmotelma mitä tietoja tietokannassa olisi, jos tietokanta toteutettaisiin näillä tiedoilla, niin siihen tulisi yksi taulu.

Kentännimi	Mitä tietoja kenttä sisältää
Juokseva id	Yksilöivä juokseva numerosarja
Kellonaika	Milloin sovelluksella tiedot tallennetaan
Räntää	Tieto onko kyseisenä päivänä satanut räntää
API-Key	Ilmatieteen laitoksen datan hakuun tarvittavat API-Key tunnus
Päiväys_fmi	Ilmatieteen laitoksen datasta haetun tiedon päiväys
Edellisen_päivä_keskilämpötila	Sovellukseen annettujen tietojen edelisen päivänkeskilämpötila
Käyttäjä	Käyttäjän nimi
kellonaika_fmi	Ilmatieteen laitoksen datasta haetun tiedon kellonaika
Edellisen_päivä_keskilämpötila_fmi	Ilmatieteen laitoksen datasta laskettu keskilämpötila
Paikkakunta_käyttäjä	Missä paikkakunnalla käyttäjä on
Sataako	Tieto sataako kyseisenä päivänä
Paikkakunta_fmi	Ilmatieteen laitoksen datasta haetun paikkakunnan tieto
Paistaako	Tieto paistaako kyseisenä päivänä
Lämpötila_käyttäjä	Sovelluksessa syötetty lämpötila
Tuuleeko	Sovelluksessa tallennetu tieto, että onko kyseisenä päivänä tullut
Lämpötila_fmi	Ilmatieteen laitoksen datasta haettu kyseisen paikkakunnan lämpötilatieto, kun sovelluksessa tallennetaan tietoja
Vettä	Tieto sataako kyseisenä päivänä vettä
Päiväys	Sovelluksessa tallennettujen tietojen päiväys
Lunta	Tieto sataako kyseisenä päivänä lunta
Ukkonen	Tieto onko ollut kyseisenä päivänä ukkosta
Vapaakenttä	Vapaa tietokenttä, esim. Kevät alkaa tulemaan, linnut lauloivat kauniisti. Tai mahtavat revontulet oli yöllä
Pilvisyys	Tieto onko kyseisenä päivänä ollut pilvistä

KUVIO 41. Ensimmäinen hahmotelma mitä tietoja tietokannassa olisi

5.3.2 Tietokannan suunnittelukuvaukset

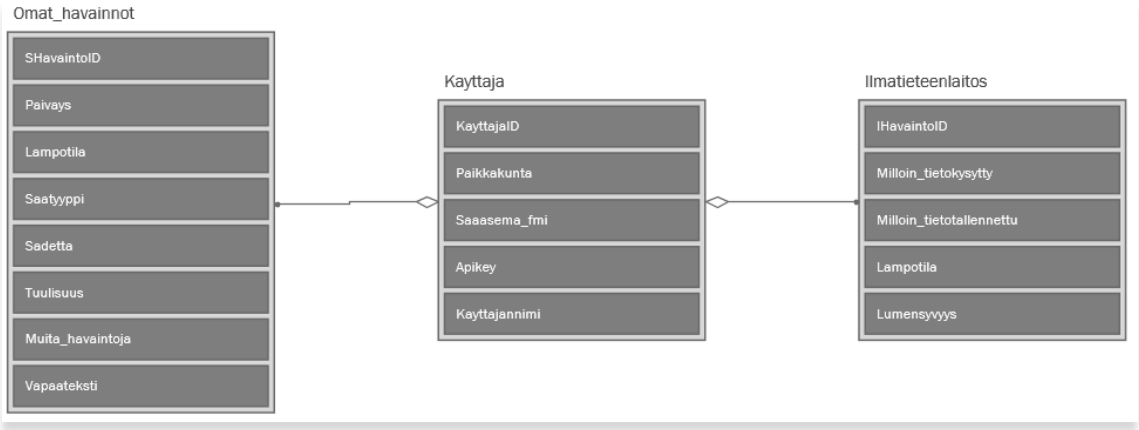
5.3.2.1 Käsitemalli

Käsitemallin ensimmäisessä vaiheessa (katso kuvio 42) jaoin asiat 3. eri osaan; Internet sovellus, Ilmatieteenlaitos ja Käyttäjään, joista tulisi omat taulut. Internet -sivulla kysytään oheisia tietoja tietojen tallennuksessa. Ilmatieteen laitoksen avoimesta datasta haetaan lämpötila ja lumensyvyys tiedot ja sen lisäksi halutaan tietää, milloin Ilmatieteen laitoksen tietokantaan nämä tiedot on tallennettu ja milloin nämä tiedot on haettu. Ilmatieteen laitos tallentaa tiedot 10 minuutin välein. Käyttäjän tiedoissa ovat käyttäjännimi, paikkakunta, miltä Ilmatieteen laitoksen havaintoasemalta tiedot haetaan ja apikey avain, jolla saadaan avoimesta datasta tieto haettua.

Internet sovellus	Ilmatieteenlaitos	Käyttäjä
päiväys ja kellonaika	päiväys ja kellonaika tietokysytty	Paikkakunta käyttäjä
lämpötila	päiväys ja kellonaika tallennettu	Sääasema ilmatieteen laitos
säätyyppi	lämpötila	Apikey
sadetta	lumensyvyys	Käyttäjännimi
tuulisuus		
muita havaintoja		
omat havainnot		

KUVIO 42. Käsiteanalyysin ensimmäinen vaihe

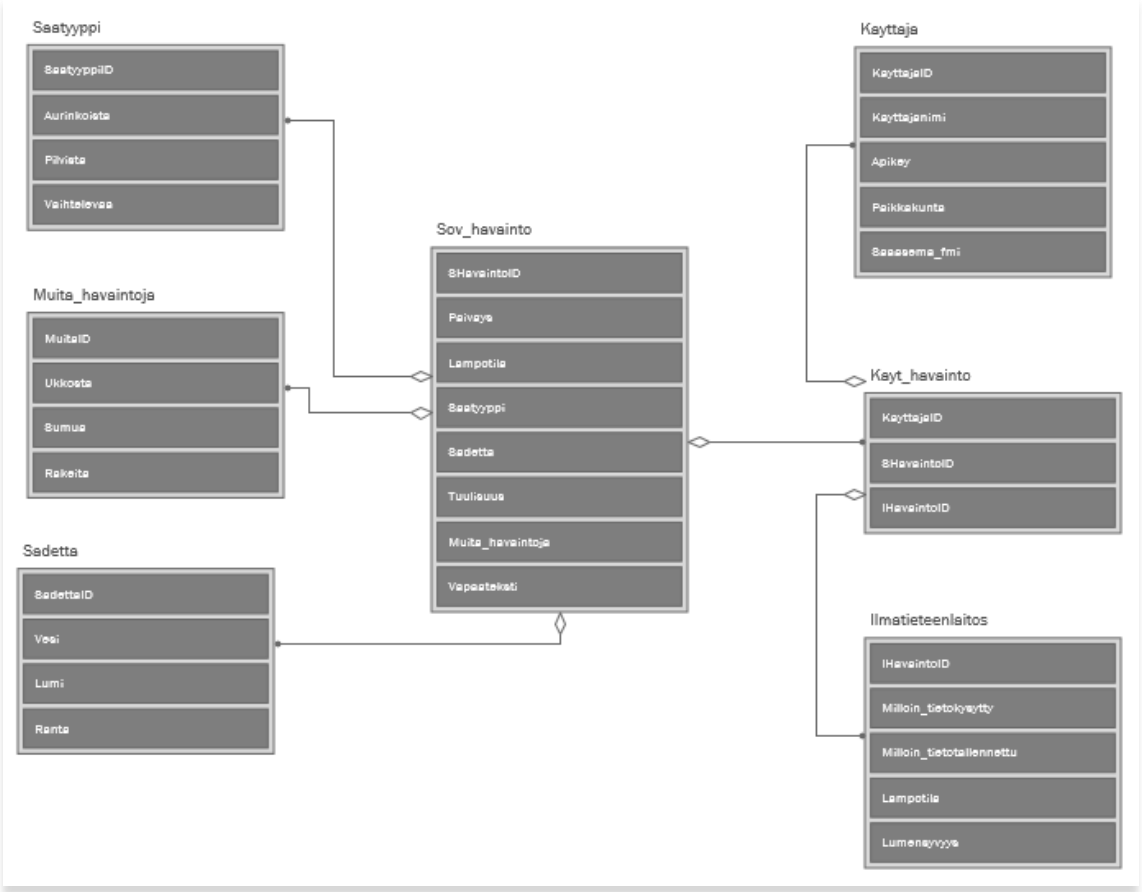
Tietokannan ensimmäinen karkea kuva (katso kuvio 43), joka on piirretty (katso kuvio 42) tietokanta taulujen tiedoista.



KUVIO 43. Karkea malli käsiteanalyysistä

Omat_havainnot taulu muuttui Sov_havainto tauluksi. Tämän jälkeen tein uusia tauluja, joissa tieto kerrotaan, ettei tämä kaikki tieto olisi Sov_havainto taulussa. Uusia tauluja ovat Muita_havaintoja, Sadetta ja Saatyyppi, jotka linkittyvät Sov_havainto taulun saman nimisiin sarakkeisiin. Jos tämän jälkeen, halutaan muuttaa Aurinkoista muotoon Aurinko, niin muutos tarvitsee tehdä ainoastaan Saatyyppi tauluun. Ennen taulun luontia muutos olisi täytynyt tehdä Sov_havainto taulun jokaiselle riville Saatyyppi sarakkeessa, jossa on Aurinkoista sana.

Kayttaja taulua en ensin ajatellut, että tätä tarvitsee muuttaa ollenkaan vaan se käy suoraan. Mutta tarkemmin ajateltuna Kayttaja taulussa on paljon toistuvaa tietoa esimerkiksi Saaasema_fmi, Api-key, Kayttajanimi ja Paikkakunta. Tästä johtuen Kayttaja taulun merkitys muuttui ja taulun tilalle tuli Kayt_havainto taulu, jolla yhdistetään Kayttaja, Sov_havainto ja Ilmatieteenlaitos taulut keskenään.



KUVIO 44. Käsiteanalyysi viimeinen

Käsiteanalyysin lopuksi kolme taulua muuttui seitsemäksi tauluksi (katso kuvio 44). Käsiteanalyy-sissä tuli seuraavat taulut ja taulujen sarakkeet (katso kuvio 45).

Sov_havainto	Ilmatieteenlaitos	Kayttajanimi
SHavaintoID	IHavaintoID	KayttajaID
Paivays	Milloin_tietokysytty	Kayttajanimi
Lampotila	Milloin_tietotallennettu	Apikey
Saatyyppi	Lampotila	Paikkakunta
Sadetta	Lumensyvyys	Saaasema_fmi
tuulisuus		
Muita_havaintoja		
Vapaateksti		

Muita_havaintoja	Sadetta	Saatyyppi
MuitaID	SadettaID	SaatyyppiID
Ukkosta	Vesi	Aurinkoista
Sumua	Lumi	Pilvista
Rakeita	Ranta	Vaihtelevaa

Kayt_havainto
KayttajaID
SHavaintoID
IHavaintoID

KUVIO 45. Käsiteanalyysin lopputulos

5.3.2.1 Tarveanalyysi

Aloitin tarveanalyysin käymällä läpi internet -sivustolta tallennettavia tietoja. Kävin myös muutkin taulut läpi samalla tavalla. Mihin kaikkiin tauluihin tallentuu tieto, kun internet -sivustolla tallennetaan säähavainto ja haetaan tietoja Ilmatieteen laitoksen avoimesta datasta?

Kävin taulujen sarakkeet läpi, mistä tieto tulee sarakkeeseen, mitä tietoa tallennetaan ja minkä tyyppinen sarakkeen täytyy olla. Kävin kaikkien taulujen sarakkeet oheisella tavalla läpi, ohessa Sov_havainto taulun Paivays sarake.

- Päiväys ja kellonaika
    - Tieto tallennetaan Sov_havainto tauluun Paivays sarakkeeseen.
    - Tieto on päivämuotoinen.

Ilmatieteen laitoksen tiedon tallennus

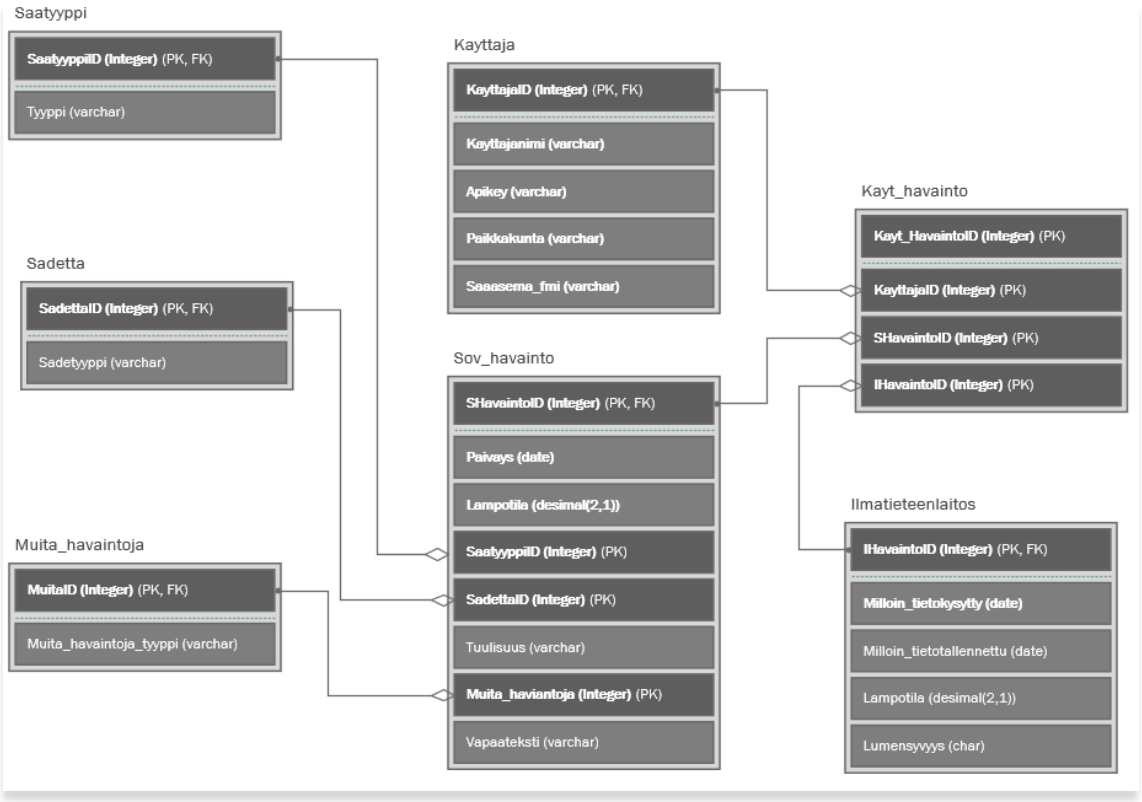
- Päiväys
    - Tieto tallennetaan Ilmatieteenlaitos taulun, Milloin_tietokysytty sarakkeeseen.

- Samassa yhteydessä tallennetaan myös tieto, milloin Ilmatieteen laitos on tallentanut viimeisimmän säätiedon ja tämä tallennetaan Milloin_tietotallennettu sarakkeeseen.
- Tieto on kummassakin päivämuotoinen.

Tarveanalyysin teossa huomasin, että olin laittanut jatkuvasti Saatyyppi, Sadetta ja Muita_havaintoja tauluihin myös arvot, mitä taulun riveillä on aikaisemmin tehdyissä kuvissa.

Käsiteanalyysissa (katso kuvio 45) esimerkiksi Saatyyppi on taulu, johon olen laittanut SaatyyppiID, Aurinkoista, Pilvista ja Vaihtelevaa. Näistä kolme viimeistä on säätyyppejä, jotka täytyy olla omassa sarakkeessa. Tämän muutin uudessa kuvassa siten, että Saatyyppi taulussa ovat 2 saraketta SaatyyppiID ja Tyyppi. SaatyyppiID on numeerinen arvo ja siitä viitataan Sov_havainto taulun SaatyyppiID riville. Saatyyppi taulun Tyyppi -sarakkeeseen tulee rivit Aurinkoista, Pilvista ja Vaihtelevaa. Tämän johdosta näistä tauluista saadaan järkevämmät ja luettavammat. Myös Kayt_havainto taulu sai myös uuden sarakkeen Kayt_HavaintoID arvon, joka on pääavain ja sillä saadaan yksilöivä tieto tauluun.

Tarveanalyysin jälkeen tietokanta on testattuna ja kuviossa 46 on saatu aikaan tietokannan malli siitä, minkälainen tietokanta on tarveanalyysin jälkeen.



KUVIO 46. Tietokanta tarveanalyysin jälkeen

5.3.2.2 Tietokannan normalisointi

Normalisointia on tehty koko tietokannan suunnittelun ajan. Tarveanalyysissä huomasin, että Saatyyppi, Sadetta ja Muita_havaintoja tauluissa olin laittanut tietoihin jo valmiit arvot. Poistin arvot ja laitoin Tyyppi tiedot jokaiseen tauluun, joihin tallennetaan kyseinen tieto. Tämä olisi ollut normalisointia, jos tässä vaiheessa olisin poistanut Saatyyppi, Sadetta ja Muita_havaintoja tauluista valmiit arvot, ensimmäinen normaalimuoto kieltää toistuvat ryhmät.

Mietin normalisoinnin tiedon toistamista tarkemmin ja tiedon tallennusta rakenteellisesti. Tämän johdosta lisäsin Sadetta tauluun kohdan Ei_sadetta ja Muita_havaintoja tauluun Ei_muitahavaintoja rivit. Nämä tallennetaan tietokantaan omalla numerolla. Muuten jos ei sataisi niin Sov_havainto tauluun tallennettaisiin NULL SadettaID riville. Tämän muutoksen jälkeen Sov_havainto taulun jokaiselle riville tulee arvo.

Sov_havainto taulussa on Vapaateksti kenttä ja, jos siihen syötetään vapaata tekstiä, niin tämä koko teksti tallennetaan kyseiseen tauluun. Ensimmäinen normaali muoto kieltää sekoittamasta

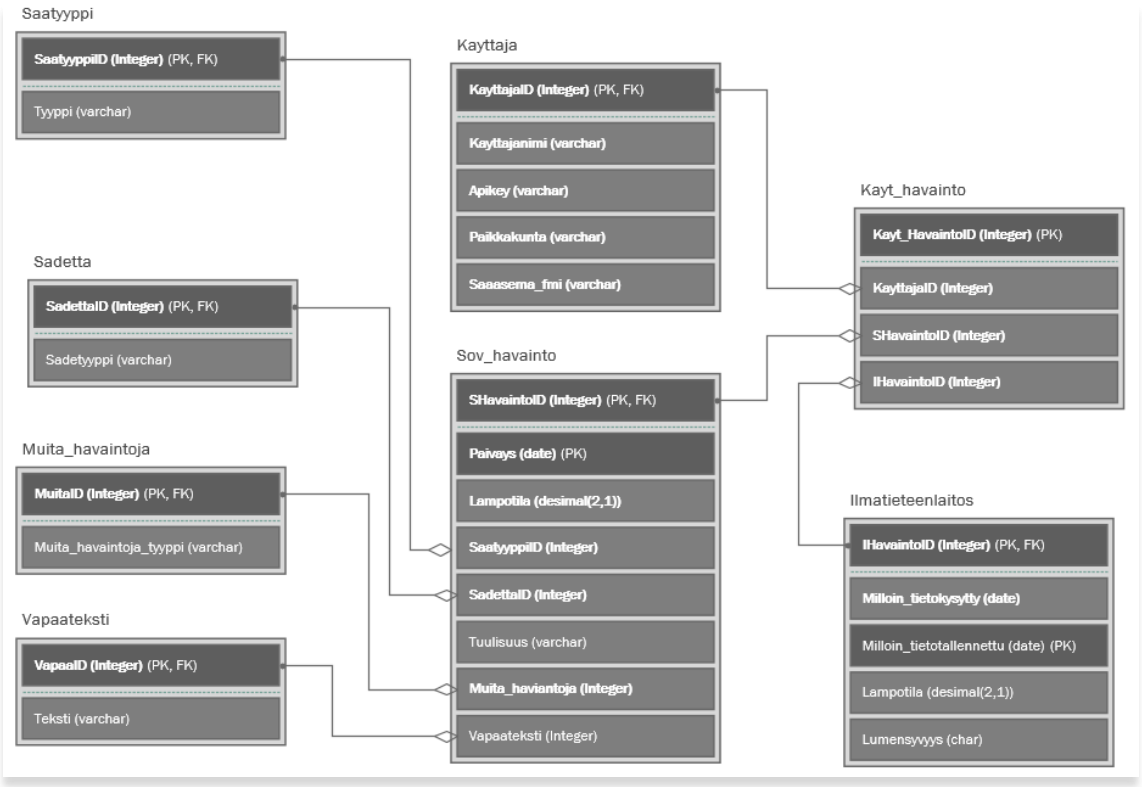
arvo joukkoja, tällöin olisi tekstiä ja numeroita sekaisin Sov_havainto taulussa. Tämän johdosta lisäsin uuden taulun Vapaateksti, jolle tulee kaksi saraketta VapaaID ja Teksti.

Samoin Sov_havainto taulun Paivays sarakkeesta tuli perusavain, koska SHavaintoID ja Paivays muodostavat kyseisen havainnon.

Kayt_havainto taulusta myös poistin perusavaimet muilta paitsi Kayt_havaintoID sarakkeelta. Muut sarakkeet ovat kuitenkin pakollisia.

Ilmatieteenlaitos tauluun laitoin Milloin_tietotallennettu sarakkeen perusavaimeksi, koska kyseinen tieto kertoo, milloin Ilmatieteen laitos on tallentanut ja saanut kyseisen tiedon.

Tähän asti tietokantaa on tehty ilman, että tietokannan tauluja olisi fyysisesti yritetty luoda tietokantaan. Normalisoinnin lopputulos (katso kuvio 47).



KUVIO 47. Normalisoinnin lopputulos

5.3.3 Tietokannan rakentaminen ja kokeileminen

Tietokannan normalisoinnin jälkeen tein tietokannan luonti lauseet normalisoinnin lopputuloksesta (katso kuvio 47).

Kaikissa tauluissa ensimmäisenä sarakkeessa on perusavain ja se on numerokenttä. Tietojen viennissä tauluihin Saatyyppi, Sadetta, Muita_havaintoja, Vapaateksti ja Kayttaja laitetaan INSERT -lausekkeeseen numero, että monesko rivi tulee. Mutta tätä ei voida käyttää tauluissa Sov_havainto, Ilmatieteenlaitos ja Kayt_havainto, koska tauluihin tulee koko ajan uusia rivejä ja INSERT -lausekkeesta ei tule järkevä, jos tietojen viennissä laitetaan seuraava vapaa numero. Tämä ratkesi sillä, että INTEGER -lauseeseen lisätään AUTO_INCREMENT, joka lisää seuraavan vapaan numeron, kun tauluun tallennetaan tietoja (MariaDB 2018b, viitattu 31.5.2018.)

Tietokannan luonti lause ei onnistunut tauluihin Sov_havainto ja Ilmatieteenlaitos, koska näissä tauluissa on kaksi PRIMARY KEY arvoa. Kummastakin taulusta poistin Paivays kentän PRIMARY KEY arvon, jonka jälkeen myös näiden taulujen luonti onnistui normaalisti. Tauluissa Paivays kentän PRIMARY KEY arvolla ei ole merkitystä, koska taulun ID arvo yksilöi kuitenkin aina rivin.

Tauluihin Saatyyppi, Sadetta, Muita_havaintoja ja Kayttaja tietojen tallennus onnistui helposti, koska näihin tulee perus INSERT -lauseke ja muutama rivi, eikä näiden taulujen tieto muutu jatkuvasti. Mutta koska tauluihin Sov_havainto, Vapaateksti, Kayt_havainto ja Ilmatieteenlaitos viedään tietoa jatkuvasti, aloin miettimään, että missä vaiheessa mihinkin tauluun pitäisi tehdä tietojen vienti INSERT -lauseke, jotta saadaan oikea arvo lapsi- ja äititaulun viiteavaimiin. Taulujen vientijärjestykseksi tuli Ilmatieteenlaitos, Vapaateksti, Sov_havainto ja viimeiseksi viedään tieto Kayt_havainto tauluun. Tällöin saadaan ID arvojen viimeisin arvo selville.

Vapaateksti tauluun tällä hetkellä ei ole pakko tallentaa tietoa ja, jos käyttäjä ei kirjaa omia havaintoja, niin silloin INSERT -lauseketta ei suoriteta. Sov_havainto taulun kanssa Vapaateksti taulun linkitys ei oikein onnistunut ja arvot eivät tulleet oikeille kohdilleen, koska tässä aluksi yritin tehdä UPDATE -lauseketta tauluun ja muuttaa VapaaID kenttään oikean arvon. Joten muutin tämän siten, että Vapaateksti tauluun on aina pakko tallentaa ja VapaaID kentän muutin AUTO_INCREMENT arvon, jotta tauluun vietäessä tulee aina seuraava vapaa numero VapaaID kenttään. Tämän muutoksen jälkeen aina viimeisin tallennus tauluun oli viimeisin omien havaintojen syöttö.

Ongelmaksi muodostui Sov_havainto taulun Vapaateksti kenttä, Kayt_havainto taulun SHavaintoID ja IHavaintoID taulujen kentät, joihin pitää saada toisen taulun ID arvon viimeisin arvo syötettyä, kun tehdään INSERT -lauseketta näihin tauluihin. Ensimmäiseksi ajattelin, että laitan näihin kenttiin arvon esimerkiksi 9999 INSERT:ssä ja tämän jälkeen teen UPDATE -lausekkeen jokaiseen näihin tauluihin ja arvoihin ja muutan ne oikeiksi. Tästä tuli todella sekava, kun millä tavalla UPDATE -lauseke täytyy muodostaa, että saadaan viimeisin arvo tallennettua taulujen ID kenttiin.

TechOnTheNet.com 2018 & W3Schools 2018k (Viitattu 7.6.2018) sivustojen vinkkien mukaan suunnittelin ensimmäistä UPDATE -lauseketta, jossa on SELECT, missä viimeisin arvo saadaan selville, mutta tästä alkoi tulemaan aika sekava, joten mietin, että eikö tämän voisi INSERT:ssä hoitaa, ettei erillistä UPDATE -lauseketta tarvitsisi tehdä.

Stack Overflow 2018b (viitattu 7.6.2018) löytyi tähän hyvä vinkki, INSERT -lausekkeen sisälle sulkujen sisään tehdään SELECT, jossa on MAX(SARAKKEENNIMI) AS SARAKKEENNIMI, jolloin INSERT onnistui oikein ja sain ilman UPDATE lauseketta oikean arvon sarakkeeseen. Tätä yritin aluksi myös (katso kuvio 48), mutta tämä ei onnistunut, kun yritin ilman sulkuja ja oli ’’ merkit SELECT:n ympärillä.

```
INSERT INTO Kayt_havainto () VALUES (NULL, 1, (SELECT max(SHavaintoID) AS SHavaintoID FROM Sov_havainto),
(SELECT max(IHavaintoID) AS IHavaintoID FROM Ilmatieteenlaitos))

INSERT INTO Kayt_havainto () VALUES (NULL, 1, 'SELECT max(SHavaintoID) AS SHavaintoID FROM Sov_havainto',
'SELECT max(IHavaintoID) AS IHavaintoID FROM Ilmatieteenlaitos')
```

KUVIO 48. SELECT INSERT:in sisässä () ja ’’ merkeillä

Tässä vaiheessa huomasin, että Ilmatieteenlaitos taulussa lumensyvyys on vääränlainen arvo, kun arvona on CHAR, joten muutin sen DESIMAL(3) arvoksi, koska kenttään syötetään pelkästään numeraalisia arvoja eikä kirjaimia.

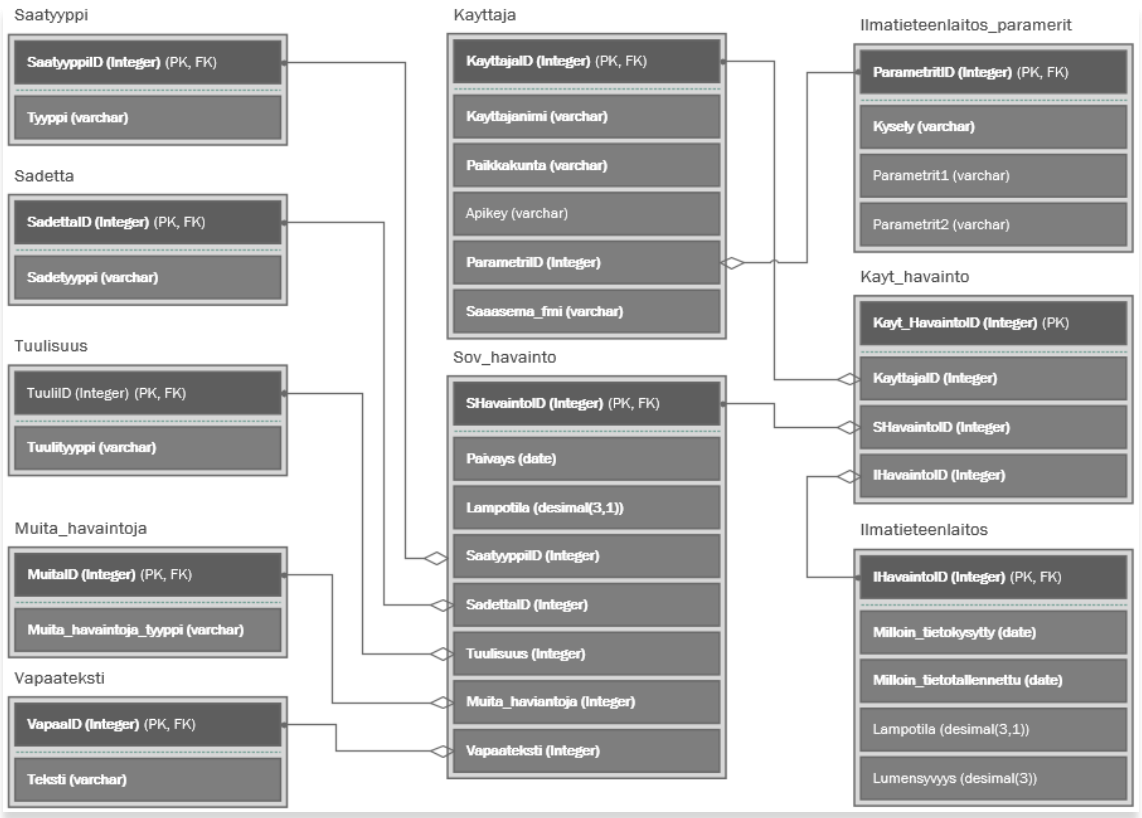
Sivusto ja tallennukset toimivat nyt oikein. Tuli mieleen, että PHP -koodia voisi siivota jonkin verran ja parametrejä muuttaa, että nämä haettaisiin tietokannasta, ettei ne olisi PHP -koodissa. Tämän johdosta tein uuden taulun Ilmatieteenlaitos_parametrit, tauluun tuli Ilmatieteen laitoksen kysely parametri, lämpötila ja lumensyvyyden parametrit, jotka tulevat XML-sanomassa. Samalla lisäsin Kayttaja tauluun uuden sarakkeen ParametriID, joka linkittyy Ilmatieteenlaitos_parametrit taulun ParametriID kenttään. Samalla mietin, että pitäisikö Kayttaja taulusta siirtää Apikey sarake myös

Ilmatieteenlaitos_parametrit tauluun, mutta päädyin siihen, että Kayttaja taulu on käyttäjäkohtainen ja Ilmatieteenlaitos_parametrit taulun samoja parametrejä voivat muutkin käyttäjät käyttää.

Sivustolle tehty haku ominaisuuksia, jotta voidaan hakea omat havainnot tallennus ja samalla hetkellä tallennetut Ilmatieteen laitoksen tiedot. Tässä vaiheessa tuli ongelmaksi tuulisuus, kun tähän asti on ollut ruksi kohdassa kyllä 1 ja ei 0. Miettiä, että miten tästä tehdään SELECT -hakuun kohta, jos kentän arvo on 0 niin kerrotaan ei ja jos arvo on 1 niin haku palauttaa kyllä. Päädyin siihen, että tuulesta voidaan kertoa enemmänkin. Kappaleen kohdassa 5.1 on kohta, tuulisuudelle tuli useampi eri arvo, päädyin tekemään uuden taulun Tuulisuus. Tuulisuus tauluun viedään erilaisia tuuli arvoja ja tämän taulun TuuliID linkitetään Sov_havainto taulun Tuulisuus sarakkeeseen. Tämän jälkeen haku SELECT:n teko onnistui helposti ja tuulesta voidaan nyt kertoa enemmän tietoa.

5.3.4 Tietokannan taulut

Tietokantaan tuli yhteensä 10 taulua. Kuviossa 49 on kuvattu käytössä olevat tietokantataulut ja miten ne linkittyvät toisiinsa. Taulujen määrä lisääntyi työn edetessä. Joihinkin tauluihin tuli uusia arvoja ja sarakkeiden arvojen määrittelyt muuttuivat.



KUVIO 49. Tietokannan taulut ja yhteydet

Tietokanta taulujen kuvaukset (katso liite 6), tietokannan luontilauseet (katso liite 7) ja tietokantaan tarvittavien esitietojen vientilauseet (katso liite 8).

5.3.4.1 Ominaisuudet

Tietokanta taulujen ominaisuudet löytyvät kuviosta 50.

Saatyyppi	Sadetta	Kayt_havaintoja	Muita_havaintoja
SaatyyppiID	SadettaID	Kayt_HavaintoID	MuitaID
Tyyppi	Saatyyppi	KayttajaID	Muita_havaintoja_tyyppi
		SHavaintoID	
		IHavaintoID	

Sov_havainto	Ilmatieteenlaitos	Kayttaja
SHavaintoID	IHavaintoID	KayttajaID
Paivays	Milloin_tietokysytty	Kayttajanimi
Lampotila	Milloin_tietotallennettu	Apikey
SaatyyppiID	Lampotila	Paikkakunta
SadettaID	Lumensyvyys	Saaasema_fmi
Tuulisuus		
Muita_havaintoja		
Vapaateksti		

Tuulisuus	Ilmatieteenlaitoksen_parametrit	Vapaateksti
TuuliID	ParametriID	VapaaID
Tuulityyppi	Kysely	Teksti
	Parametrit1	
	Parametrit2	

Kuvio 50. Tietokanta taulujen ominaisuudet

5.3.4.2 Relaatiokaavat

Alla olevassa tiedoissa alleviivattuna on taulun perusavain ja *kursivoituna* äititaulun sarake, johon lapsitaulun perusavain viittaa.

- Saatyyppi (SaatyyppiID, tyyppi)
- Sadetta (SadettaID, Saatyyppi)
- Tuulisuus (TuuliID), Tuulityyppi
- Muita_havaintoja (MuitaID, Muita_havaintoja_tyyppi)
- Vapaateksti (VapaaID, Teksti)
- Kayttaja (KayttajaID, Kayttajanimi, Apikey, *ParametriID*, Paikkakunta, Saaasema_fmi)
- Sov_havainto (SHavaintoID, Paivays, Lampotila, *SaatyyppiID, SadettaID*, Tuulisuus, *Muita_havaintoja, Vapaateksti*)
- Kayt_havainto (Kayt_HavaintoID, *KayttajaID, SHavaintoID, IHavaintoID*)
- Ilmatieteenlaitos (IHavaintoID, Milloin_tietokysytty, Milloin_tietotallennettu, Lampotila, Lumensyvyys)
- Ilmatieteenlaitos_parametrit (ParametritID), Kysely, Parametrit1, Parametrit2

6 POHDINTA JA JATKOKEHITYS

Työssäni tein Sääpäiväkirja -sivuston ja tietokannan. Sivustolle ja tietokantaan haetaan Ilmatieteen laitoksen avoimesta datasta säätietoja ohjelmistorajapintaa käyttäen. Työ alkoi kasvamaan työn edetessä, sivustolle tuli uusia ominaisuuksia, joita en aikaisemmin ajatellut ottavani sivustolle. Sivustolle tuli alkuperäisen suunnitelman mukaan lämpötilan ja lumensyvyyden, lisäksi sääennustus ja XML-sanomasta tulevat hetkelliset säätiedot. Aluksi en myöskään ajatellut tässä vaiheessa siirtää sivustoa web-hotelliin.

Web-hotellin siirron yhteydessä sivustolle täytyi luoda PHP:llä evästeet ja sisäänkirjautuminen. Nämä on rajattu työstä pois, keskityn pelkästään säätietoihin ja niiden tallennukseen tietokantaan. Siirrossa muodostui paljon ongelmia vastaan, joita en aluksi osannut ajatella. Esimerkiksi kellonajan ja isojen ja pienien kirjaimen kanssa sai olla tarkkana, että toimivat oikein. Sivuston lopullinen näkymä web-hotellissa kaikkien lisäyksien jälkeen Chrome selaimessa löytyy liitteestä 10.

Työssä pääsin asettamiini tavoitteisiin ja niiden ylikin, koska sääennustusta ja etusivun säätietojen hakua en ajatellut aloittaessa ollenkaan. Nämä tulivat lisänä työn edetessä, kun tutkin lisää Ilmatieteen laitoksen avointa dataa ja sen tuomia mahdollisuuksia.

6.1 Jatkokehitys sivustolle

XML-sanomassa tulee paljon enemmän tietoa, kuin mitä nyt tallennetaan tietokantaan. Ilmatieteenlaitos taulua täytyisi muokata, että näiden kaikkien XML-sanomassa tulleiden arvojen tallennus onnistuu tietokantaan. XML-sanomassa tulevat tiedot näkyvät kuviossa 33. Samoin omia hakutoimintoja pitää muokata, että niihin saadaan uudet tiedot tietokannasta.

Tietokanta muutoksien yhteydessä täytyy tehdä muutoksia etusivun säätietojen hakuun ja päivitykseen. Koska XML-sanomassa säätiedot päivittyvät 10 minuutin välein, joten säätietoja ei kannata tallentaa joka kerta, kun etusivua päivitetään. Tämä aiheuttaa tuplarivejä tietokantaan. Tähän täytyy rakentaa IF -lauseke, jossa tarkistetaan milloin säätiedot ovat viimeksi tallennettu ja onko tulevassa XML-sanomassa uudempi säätieto vai ei.

Käyttäjätietoihin voisi tehdä jatkokehitystä siten, että kirjautumisvaiheessa, kuka käyttäjä on kirjautunut ja tämän tietoihin tallennetaan mitä säätietoja on tallennettu, mutta raportointi olisi kuitenkin yhteinen.

Samoin havaintoasemaa pitäisi pystyä muuttamaan, kun tällä hetkellä havaintoasema on Kajaani. Tähän voisi lisätä sivustolle kohdan, jossa voidaan muuttaa havaintoasemaa ja minkä havaintoaseman tiedot tietokantaan tallennetaan.

Säätietojen haussa pitäisi XML-sanoma parseroida, kun tällä hetkellä jokainen kohta haetaan erikseen XML-sanomasta ja se lisää koodin- ja kyselyiden määrää Ilmatieteen laitokselle.

Tietokantaan voisi lisätä haun parametri tietoja varten, koska tällä hetkellä löytyy ainoastaan lämpötila ja lumensyvyys.

Web-hotellin jatkokehityksenä olisi, että tietokannasta ja sivustosta pystyisi ottamaan varmuuskopion tietyin väliajoin. Tällä hetkellä tietoja ei varmuuskopioida lainkaan.

LÄHTEET

Apache Friends 2018. Viitattu 3.3.2018, https://www.apachefriends.org/index.html

The Apache Software Foundation 2018. Viitattu 3.3.2018, http://httpd.apache.org/

Banas, D, 2014. PHP MySQL Tutorial. New Think Tank. Viitattu 7.4.2018, http://www.newthink-tank.com/2014/09/php-mysql-tutorial/

ekurssit 2018. SQL -tietojen syöttäminen tietokantaan lomakkeella. Viitattu 7.6.2018, http://www.ekurssit.net/kurssit/int_ohj2/sqlsyotto.php

Hovi, A., Huotari, J. & Lahdenmäki, T. 2005. Tietokantojen suunnittelu & indeksointi. Jyväskylä: Docendo Finland Oy.

IBM Archives: Edgar F. Codd 2003. Viitattu 21.7.2018, https://www-03.ibm.com/ibm/history/exhibits/builders/builders_codd.html

Ilmatieteen laitos, 2018a. Avoin lähdekoodi – Ilmatieteen laitos. Viitattu 25.2.2018, http://ilmatieteenlaitos.fi/avoin-lahdekoodi

Ilmatieteen laitos, 2018b. Latauspalvelun pikaohje – Ilmatieteen laitos. Viitattu 25.2.2018, http://ilmatieteenlaitos.fi/latauspalvelun-pikaohje

Ilmatieteen laitos, 2018c. Open data manual – Finnish Meteorological Institute. Viitattu 21.7.2018, http://en.ilmatieteenlaitos.fi/open-data-manual

Ilmatieteen laitos, 2018d. Havaintojen lataus – Ilmatieteen laitos. Viitattu 21.7.2018, http://ilmatieteenlaitos.fi/havaintojen-lataus#!/

Ilmatieteen laitos, 2018e. Tallennetut kyselyt – Ilmatieteen laitos. Viitattu 13.5.2018, http://ilmatieteenlaitos.fi/tallennetut-kyselyt

Ilmatieteen laitos, 2018f. Sää Kajaani – Ilmatieteen laitos. Viitattu 19.8.2018, http://ilmatieteenlaitos.fi/saa/kajaani

Ilmatieteen laitos, 2018g. Tuulet – Ilmatieteen laitos. Viitattu 19.8.2018, http://ilmatieteenlaitos.fi/tuulet

Laaksonen, A, 2014. Keskustelu: Muut kielet: PHP ja Ilmatieteen laitoksen avoin data? Ohjelmointiputka. Viitattu 29.6.2018, https://www.ohjelmointiputka.net/keskustelu/27950-php-ja-ilmatieteen-laitoksen-avoin-data/sivu-1

Laaksonen, A, 2011. Oppaat: PHP-ohjelmointi: Osa 13 – Tietoturva. Ohjelmistoputka. Viitattu 16.7.2018, https://www.ohjelmointiputka.net/oppaat/opas.php?tunnus=php_13

Lahtonen, T. 2002. SQL ToolKit. Jyväskylä: Docendo Finland Oy.

Lintulaakso T. 2004. Vapaan koodin voimakaksikko: PHP ja MySQL. MicroPC 8 / 2004, 55. http://mikropc.net/nettilehti/pdf/1706200452.pdf

MariaDB, 2018a. Viitattu 21.7.2018, https://mariadb.com/about-us

MariaDB, 2018b. Auto_increment – MariaDB Knowledge Base. Viitattu 31.5.2018, https://mariadb.com/kb/en/library/auto_increment/

Matias.biz 2018. Ilmatieteen laitoksen avoimen datan hyödyntäminen Matias.biz. Viitattu 6.5.2018, http://matias.biz/ilmatieteen-laitoksen-avoimen-datan-hyodyntaminen/

maxisme 2017. How to securely connect to a database with PHP? – Information Security Stack Exchange. Viitattu 16.7.2018, https://security.stackexchange.com/questions/152590/how-to-securely-connect-to-a-database-with-php

Meloni, J. 2003. MySQL Trainer Kit. Suom. R. Santala-Köykkä. Helsinki: IT Press

Microsoft, 2018a. Microsoft Visual Studio. Viitattu 20.7.2018, https://www.microsoft.com/fi-fi/store/b/visualstudio

Microsoft, 2018b. Microsoft Visual Studio. Viitattu 20.7.2018, https://visualstudio.microsoft.com/downloads/

Microsoft, 2018c. Microsoft SQL Server 2017 Express Edition. Viitattu 20.7.2018, https://www.microsoft.com/fi-fi/sql-server/sql-server-editions-express

Microsoft, 2018d. Microsoft SQL-kehittäjien työkalut. Viitattu 20.7.2018, https://www.microsoft.com/fi-fi/sql-server/developer-tools

MySQL Reference Manual, 2018a. MySQL :: MySQL 8.0 Reference Manual 13.1.21. CREATE VIEW Syntax. Viitattu 16.7.2018, https://dev.mysql.com/doc/refman/8.0/en/create-view.html

MySQL Reference Manual, 2018b. MySQL 8.0 Reference Manual 13.1.32 DROP View Syntax. Viitattu 16.7.2018, https://dev.mysql.com/doc/refman/8.0/en/drop-view.html

NetBeans, 2018a. NetBeans. Viitattu 20.7.2018, https://netbeans.org/

NetBeans, 2018b. NetBeans, Apache Software Foundation. Viitattu 20.7.2018, https://cwiki.apache.org/confluence/display/NETBEANS

Oracle VM VirtualBox 2018. Viitattu 3.3.2018, https://www.virtualbox.org/

phpMyAdmin. Viitattu 6.8.2018, https://www.phpmyadmin.net/

The Programming Geek 2015. PHP – Insert Form Data Into MySQL Database Using PHP. Video. Viitattu 23.6.2018, https://www.youtube.com/watch?v=0BoZc5oUioA

Rytkönen I 2018. Savonia-ammattikorkeakoulu 2018. PHP ohjelmointi, ääkkösongelma tietokannassa. Sisäinen lähde. Viitattu 19.7.2018, https://moodle.savonia.fi/mod/forum/discuss.php?d=35855#p61865

Stack Overflow, 2018a. datetime – Convert UTC dates to local time in PHP – Stack Overflow. Viitattu 9.7.2018, https://stackoverflow.com/questions/3792066/convert-utc-dates-to-local-time-in-php

Stack Overflow, 2018b. sql – Insert Into … values (Select … from ..) – Stack Overflow. Viitattu 7.6.2018, https://stackoverflow.com/questions/25969/insert-into-values-select-from

Smith F 2008. MySQL :: Re: how to use PHP variable in MYSQL query?. Viitattu 17.7.2018, https://forums.mysql.com/read.php?52,144546,211501#msg-211501

Sääpäiväkirja 2016 / Otavan Kirjapaino Oy Keuruu 2015

TechOnTheNet.com 2018. MariaDB: UPDATE Statement. Viitattu 7.6.2018, https://www.techonthenet.com/mariadb/update.php

W3Schools, 2018a. Bootstrap 4 Get Started. Viitattu 14.7.2018, https://www.w3schools.com/bootstrap4/bootstrap_get_started.asp

W3Schools, 2018b. Bootstrap 4 Navigation Bar. Viitattu 14.7.2018, https://www.w3schools.com/bootstrap4/bootstrap_navbar.asp

W3Schools, 2018c. Bootstrap 4 Custom Forms. Viitattu 14.7.2018, https://www.w3schools.com/bootstrap4/bootstrap_forms_custom.asp

W3Schools, 2018d. Bootstrap 4 Form Inputs. Viitattu 14.7.2018, https://www.w3schools.com/bootstrap4/bootstrap_forms_inputs.asp

W3Schools, 2018e. Bootstrap 4 Tables. Viitattu 14.7.2018, https://www.w3schools.com/bootstrap4/bootstrap_tables.asp

W3Schools, 2018f. PHP sleep() Function. Viitattu 14.7.2018, https://www.w3schools.com/php/func_misc_sleep.asp

W3Schools, 2018g. How to Make a CCS Loader. Viitattu 14.7.2018, https://www.w3schools.com/howto/howto_css_loader.asp

W3Schools, 2018h. HTML Input Attributs. Viitattu 14.7.2018, https://www.w3schools.com/html/html_form_attributes.asp

W3Schools, 2018i. PHP htmlspecialchars() Function. Viitattu 16.7.2018, https://www.w3schools.com/php/func_string_htmlspecialchars.asp

W3Schools, 2018j. PHP Insert Multiple Records Into MySQL. Viitattu 23.6.2018, https://www.w3schools.com/php/php_mysql_insert_multiple.asp

W3Schools, 2018k. SQL min() and Max() Functions. Viitattu 7.6.2018, https://www.w3schools.com/sql/sql_min_max.asp

This XML file does not appear to have any style information associated with it. The document tree is shown below.


```
<wfs:FeatureCollection xmlns:wfs="http://www.open-
gis.net/wfs/2.0" xmlns:gml="http://www.open-
gis.net/gml/3.2"xmlns:BsWfs="http://xml.fmi.fi/schema/wfs/2.0" xmlns:x
si="http://www.w3.org/2001/XMLSchema-instance" timeStamp="2018-05-
13T14:02:38Z"numberReturned="11" numberMatched="11" xsi:schemaLoca-
tion="http://www.opengis.net/wfs/2.0 http://schemas.open-
gis.net/wfs/2.0/wfs.xsd http://xml.fmi.fi/schema/wfs/2.0
http://xml.fmi.fi/schema/wfs/2.0/fmi_wfs_simplefeature.xsd">
<wfs:member>
<BsWfs:BsWfsElement gml:id="BsWfsElement.1.1.1">
<BsWfs:Location>
<gml:Point gml:id="BsWfsElementP.1.1.1" srsDimen-
sion="2" srsName="http://www.opengis.net/def/crs/EPSG/0/4258">
<gml:pos>64.21678 27.75194</gml:pos>
</gml:Point>
</BsWfs:Location>
<BsWfs:Time>2018-05-13T14:00:00Z</BsWfs:Time>
<BsWfs:ParameterName>t2m</BsWfs:ParameterName>
<BsWfs:ParameterValue>24.9</BsWfs:ParameterValue>
</BsWfs:BsWfsElement>
</wfs:member>
<wfs:member>
<BsWfs:BsWfsElement gml:id="BsWfsElement.1.1.2">
<BsWfs:Location>
<gml:Point gml:id="BsWfsElementP.1.1.2" srsDimen-
sion="2" srsName="http://www.opengis.net/def/crs/EPSG/0/4258">
<gml:pos>64.21678 27.75194</gml:pos>
</gml:Point>
</BsWfs:Location>
<BsWfs:Time>2018-05-13T14:00:00Z</BsWfs:Time>
<BsWfs:ParameterName>ws_10min</BsWfs:ParameterName>
<BsWfs:ParameterValue>2.2</BsWfs:ParameterValue>
</BsWfs:BsWfsElement>
</wfs:member>
<wfs:member>
<BsWfs:BsWfsElement gml:id="BsWfsElement.1.1.3">
<BsWfs:Location>
<gml:Point gml:id="BsWfsElementP.1.1.3" srsDimen-
sion="2" srsName="http://www.opengis.net/def/crs/EPSG/0/4258">
<gml:pos>64.21678 27.75194</gml:pos>
</gml:Point>
</BsWfs:Location>
<BsWfs:Time>2018-05-13T14:00:00Z</BsWfs:Time>
<BsWfs:ParameterName>wg_10min</BsWfs:ParameterName>
<BsWfs:ParameterValue>4.2</BsWfs:ParameterValue>
</BsWfs:BsWfsElement>
</wfs:member>
<wfs:member>
<BsWfs:BsWfsElement gml:id="BsWfsElement.1.1.4">
<BsWfs:Location>
<gml:Point gml:id="BsWfsElementP.1.1.4" srsDimen-
sion="2" srsName="http://www.opengis.net/def/crs/EPSG/0/4258">
<gml:pos>64.21678 27.75194</gml:pos>
</gml:Point>
```

```
</BsWfs:Location>
<BsWfs:Time>2018-05-13T14:00:00Z</BsWfs:Time>
<BsWfs:ParameterName>wd_10min</BsWfs:ParameterName>
<BsWfs:ParameterValue>316.0</BsWfs:ParameterValue>
</BsWfs:BsWfsElement>
</wfs:member>
<wfs:member>
<BsWfs:BsWfsElement gml:id="BsWfsElement.1.1.5">
<BsWfs:Location>
<gml:Point gml:id="BsWfsElementP.1.1.5" srsDimen-
sion="2" srsName="http://www.opengis.net/def/crs/EPSG/0/4258">
<gml:pos>64.21678 27.75194</gml:pos>
</gml:Point>
</BsWfs:Location>
<BsWfs:Time>2018-05-13T14:00:00Z</BsWfs:Time>
<BsWfs:ParameterName>rh</BsWfs:ParameterName>
<BsWfs:ParameterValue>20.0</BsWfs:ParameterValue>
</BsWfs:BsWfsElement>
</wfs:member>
<wfs:member>
<BsWfs:BsWfsElement gml:id="BsWfsElement.1.1.6">
<BsWfs:Location>
<gml:Point gml:id="BsWfsElementP.1.1.6" srsDimen-
sion="2" srsName="http://www.opengis.net/def/crs/EPSG/0/4258">
<gml:pos>64.21678 27.75194</gml:pos>
</gml:Point>
</BsWfs:Location>
<BsWfs:Time>2018-05-13T14:00:00Z</BsWfs:Time>
<BsWfs:ParameterName>td</BsWfs:ParameterName>
<BsWfs:ParameterValue>0.6</BsWfs:ParameterValue>
</BsWfs:BsWfsElement>
</wfs:member>
<wfs:member>
<BsWfs:BsWfsElement gml:id="BsWfsElement.1.1.7">
<BsWfs:Location>
<gml:Point gml:id="BsWfsElementP.1.1.7" srsDimen-
sion="2" srsName="http://www.opengis.net/def/crs/EPSG/0/4258">
<gml:pos>64.21678 27.75194</gml:pos>
</gml:Point>
</BsWfs:Location>
<BsWfs:Time>2018-05-13T14:00:00Z</BsWfs:Time>
<BsWfs:ParameterName>r_1h</BsWfs:ParameterName>
<BsWfs:ParameterValue>0.0</BsWfs:ParameterValue>
</BsWfs:BsWfsElement>
</wfs:member>
<wfs:member>
<BsWfs:BsWfsElement gml:id="BsWfsElement.1.1.8">
<BsWfs:Location>
<gml:Point gml:id="BsWfsElementP.1.1.8" srsDimen-
sion="2" srsName="http://www.opengis.net/def/crs/EPSG/0/4258">
<gml:pos>64.21678 27.75194</gml:pos>
</gml:Point>
</BsWfs:Location>
<BsWfs:Time>2018-05-13T14:00:00Z</BsWfs:Time>
<BsWfs:ParameterName>ri_10min</BsWfs:ParameterName>
<BsWfs:ParameterValue>0.0</BsWfs:ParameterValue>
</BsWfs:BsWfsElement>
</wfs:member>
<wfs:member>
<BsWfs:BsWfsElement gml:id="BsWfsElement.1.1.9">
<BsWfs:Location>
```

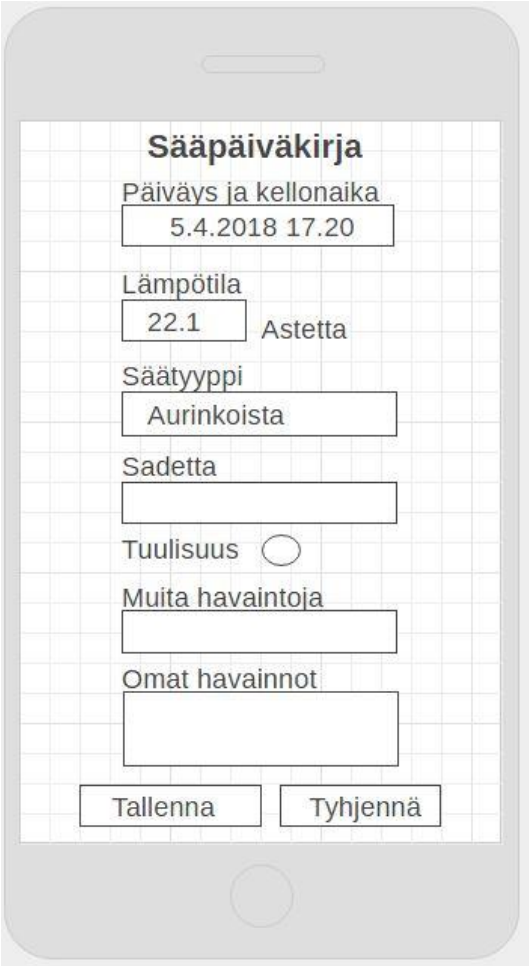
<gml:Point gml:id="BsWfsElementP.1.1.9" srsDimen-
sion="2" srsName="http://www.opengis.net/def/crs/EPSG/0/4258">
<gml:pos>64.21678 27.75194</gml:pos>
</gml:Point>
</BsWfs:Location>
<BsWfs:Time>2018-05-13T14:00:00Z</BsWfs:Time>
<BsWfs:ParameterName>snow_aws</BsWfs:ParameterName>
<BsWfs:ParameterValue>0.0</BsWfs:ParameterValue>
</BsWfs:BsWfsElement>
</wfs:member>
<wfs:member>
<BsWfs:BsWfsElement gml:id="BsWfsElement.1.1.10">
<BsWfs:Location>
<gml:Point gml:id="BsWfsElementP.1.1.10" srsDimen-
sion="2" srsName="http://www.opengis.net/def/crs/EPSG/0/4258">
<gml:pos>64.21678 27.75194</gml:pos>
</gml:Point>
</BsWfs:Location>
<BsWfs:Time>2018-05-13T14:00:00Z</BsWfs:Time>
<BsWfs:ParameterName>p_sea</BsWfs:ParameterName>
<BsWfs:ParameterValue>1023.6</BsWfs:ParameterValue>
</BsWfs:BsWfsElement>
</wfs:member>
<wfs:member>
<BsWfs:BsWfsElement gml:id="BsWfsElement.1.1.11">
<BsWfs:Location>
<gml:Point gml:id="BsWfsElementP.1.1.11" srsDimen-
sion="2" srsName="http://www.opengis.net/def/crs/EPSG/0/4258">
<gml:pos>64.21678 27.75194</gml:pos>
</gml:Point>
</BsWfs:Location>
<BsWfs:Time>2018-05-13T14:00:00Z</BsWfs:Time>
<BsWfs:ParameterName>vis</BsWfs:ParameterName>
<BsWfs:ParameterValue>43002.0</BsWfs:ParameterValue>
</BsWfs:BsWfsElement>
</wfs:member>
</wfs:FeatureCollection>

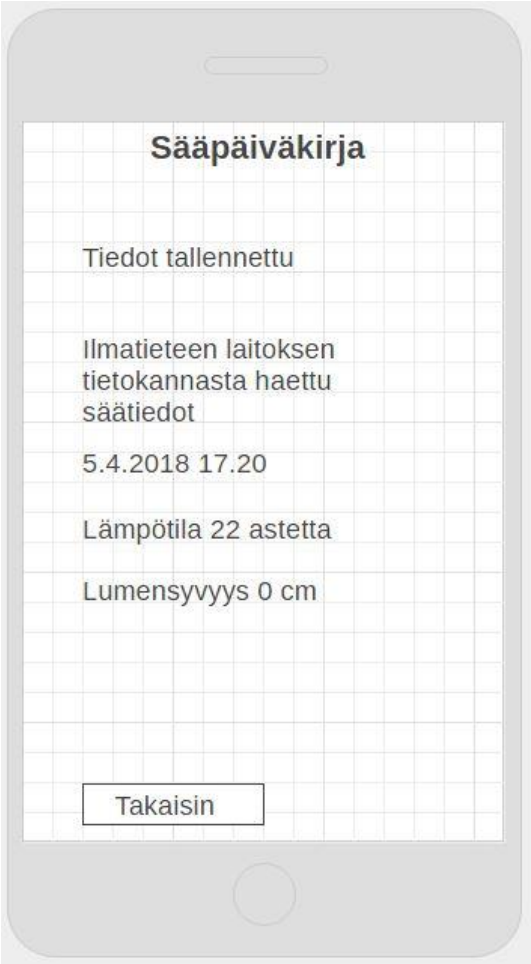
This XML file does not appear to have any style information associated with it. The document tree is shown below.

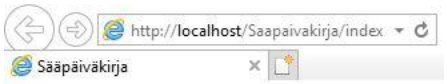
```
<wfs:FeatureCollection xmlns:wfs="http://www.open-
gis.net/wfs/2.0" xmlns:gml="http://www.open-
gis.net/gml/3.2" xmlns:BsWfs="http://xml.fmi.fi/schema/wfs/2.0"xmlns:x
si="http://www.w3.org/2001/XMLSchema-instance" timeStamp="2018-07-
21T12:46:12Z" numberReturned="11" numberMatched="11" xsi:schemaLoca-
tion="http://www.opengis.net/wfs/2.0              http://schemas.open-
gis.net/wfs/2.0/wfs.xsd               http://xml.fmi.fi/schema/wfs/2.0
http://xml.fmi.fi/schema/wfs/2.0/fmi_wfs_simplefeature.xsd">
<wfs:member>
<BsWfs:BsWfsElement gml:id="BsWfsElement.1.1.1">
<BsWfs:Location>
<gml:Point gml:id="BsWfsElementP.1.1.1" srsDimen-
sion="2" srsName="http://www.opengis.net/def/crs/EPSG/0/4258">
<gml:pos>64.21678 27.75194 </gml:pos>
</gml:Point>
</BsWfs:Location>
<BsWfs:Time>2018-05-13T17:00:00Z</BsWfs:Time>
<BsWfs:ParameterName>t2m</BsWfs:ParameterName>
<BsWfs:ParameterValue>23.2</BsWfs:ParameterValue>
</BsWfs:BsWfsElement>
</wfs:member>
<wfs:member>
<BsWfs:BsWfsElement gml:id="BsWfsElement.1.1.2">
<BsWfs:Location>
<gml:Point gml:id="BsWfsElementP.1.1.2" srsDimen-
sion="2" srsName="http://www.opengis.net/def/crs/EPSG/0/4258">
<gml:pos>64.21678 27.75194 </gml:pos>
</gml:Point>
</BsWfs:Location>
<BsWfs:Time>2018-05-13T17:00:00Z</BsWfs:Time>
<BsWfs:ParameterName>snow_aws</BsWfs:ParameterName>
<BsWfs:ParameterValue>0.0</BsWfs:ParameterValue>
</BsWfs:BsWfsElement>
</wfs:member>
```

SIVUSTON HAHMOTELMA KUVA LIITE 3

Ensimmäinen hahmotelma sivustosta




Sääpäiväkirja
Tiedot tallennettu
Ilmatieteen laitoksen tietokannasta haettu säätiedot
5.4.2018 17.20
Lämpötila 22 astetta
Lumensyvyys 0 cm
Takaisin



Sääpäiväkirja

Päiväys ja kellonaika:
6.4.2018 19:49

Lämpötila:
Astetta

Säätyyppi:
Aurinkoista

Sadetta:

Tuulisuus

Muita havaintoja:

Omat havainnot:

Tallenna Tyhjennä

Sääpäiväkirja

Päiväys ja kellonaika:
6.4.2018 19:50

Lämpötila:
Astetta

Säätyyppi:
Aurinkoista

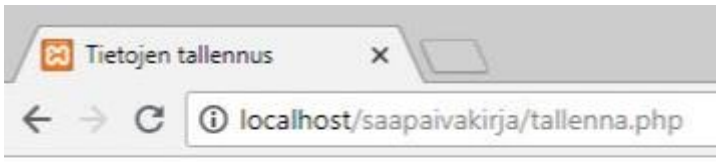
Sadetta:

Tuulisuus

Muita havaintoja:

Omat havainnot:

Tallenna Tyhjennä

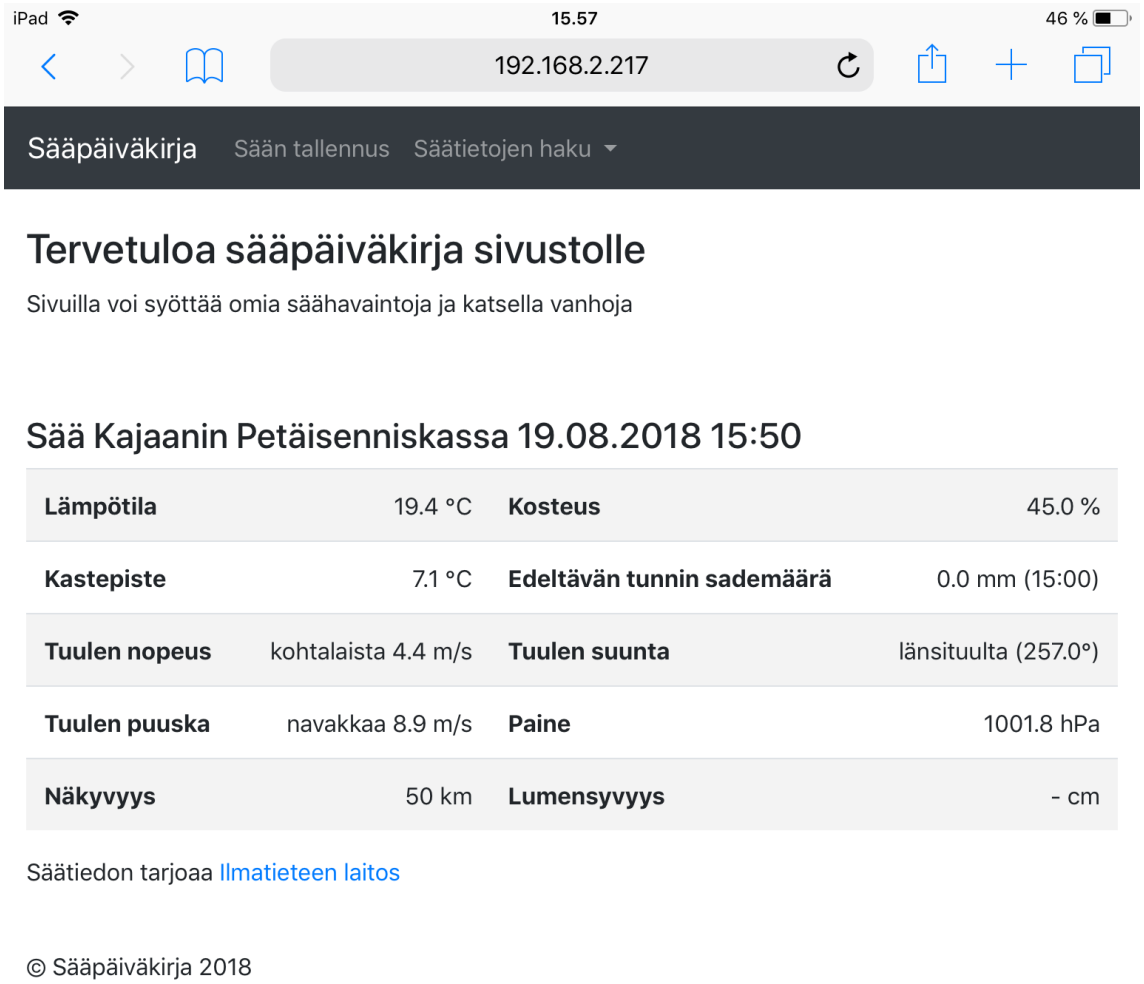


Tiedot tallennettu

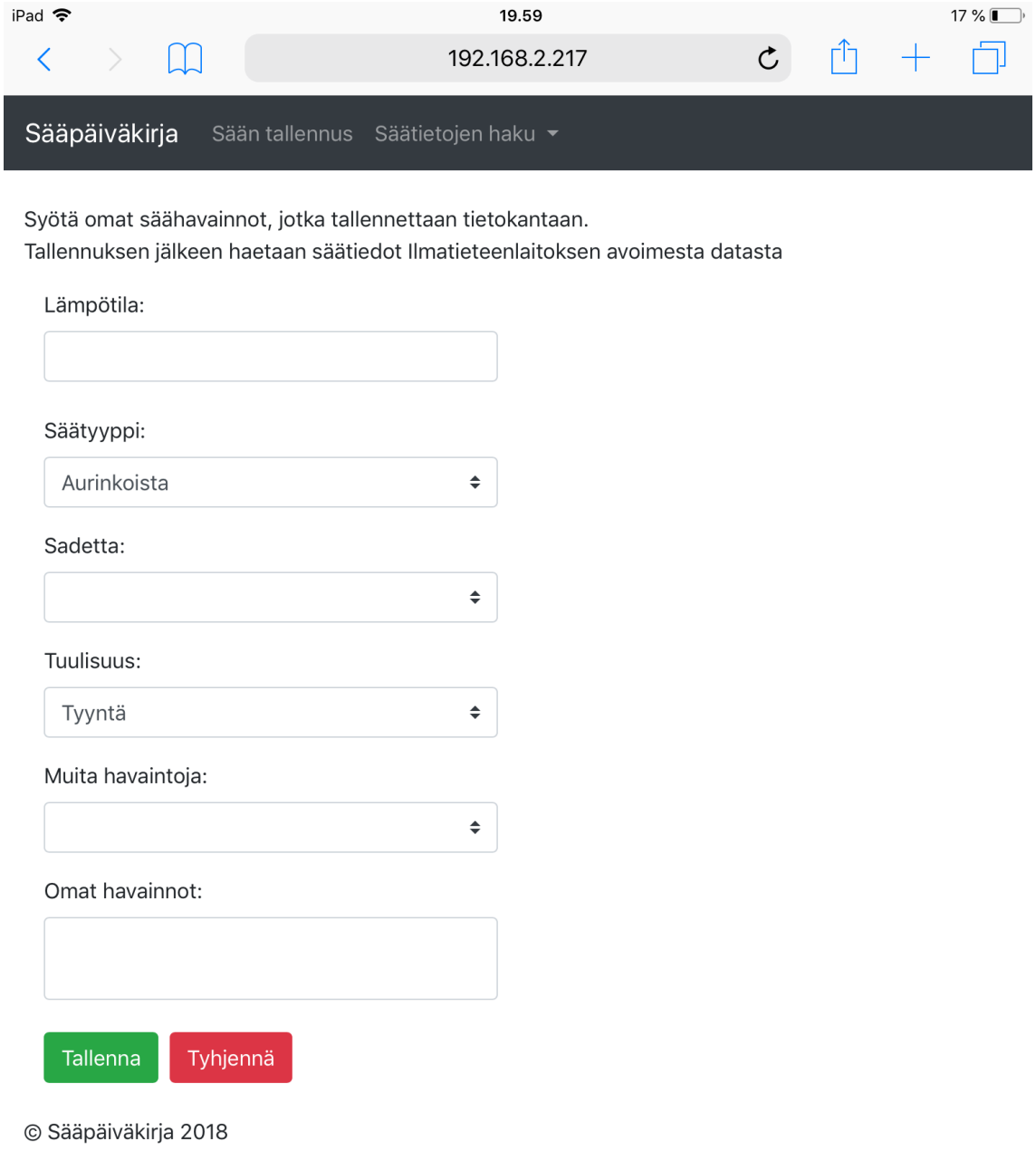
Ilmatieteenlaitoksen avoimesta datasta haetut tiedot
Viimeisin tallennus **2018-07-14 13:30:00**
Lämpötila **22.6** astetta
Lumensyvyys **0** cm

Palaa tietojen tallennukseen

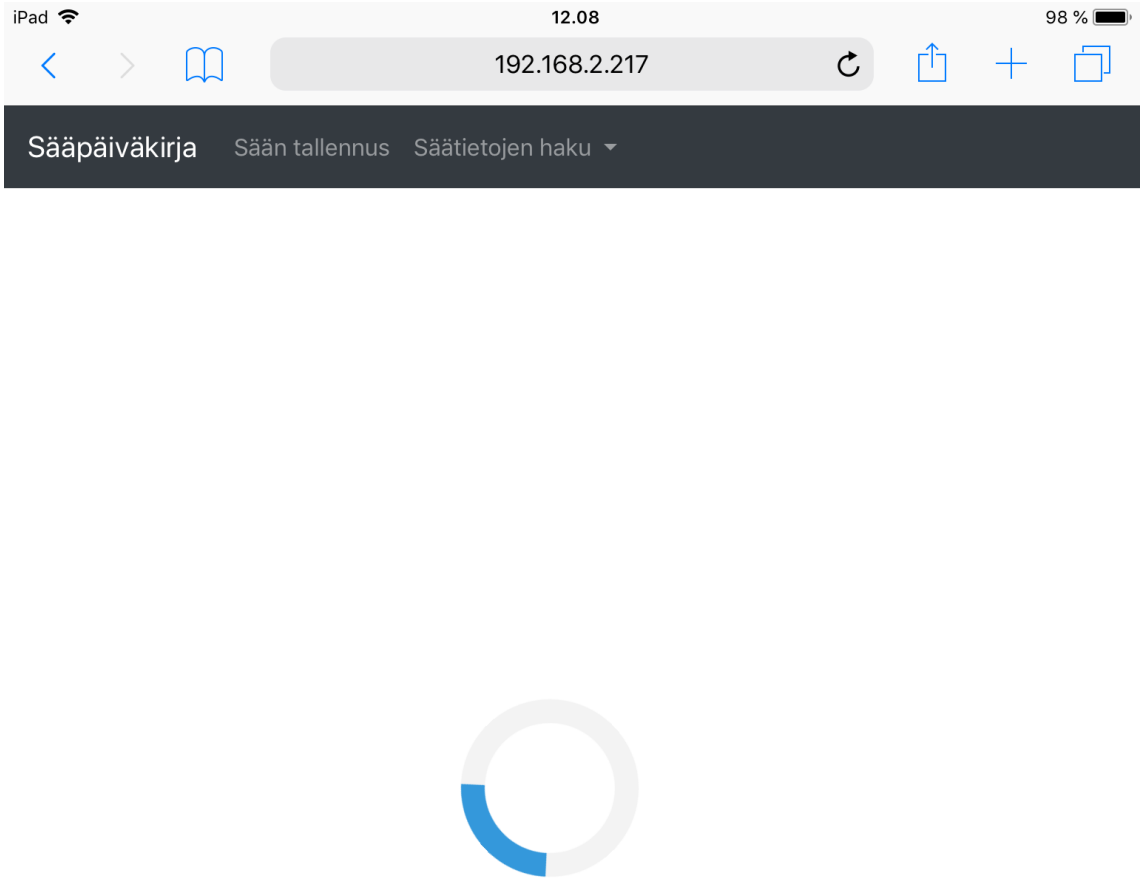
Aloitus sivu, ilman sääennustetta



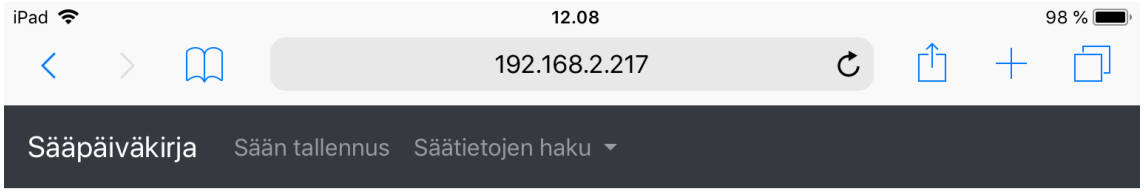
Omien säähavaintojen tallennus



Omien säähavaintojen tallennuksen jälkeen tuleva 5 sekunnin tauko ja siihen tuleva intro



Kun omat säätiedot on tallennettu



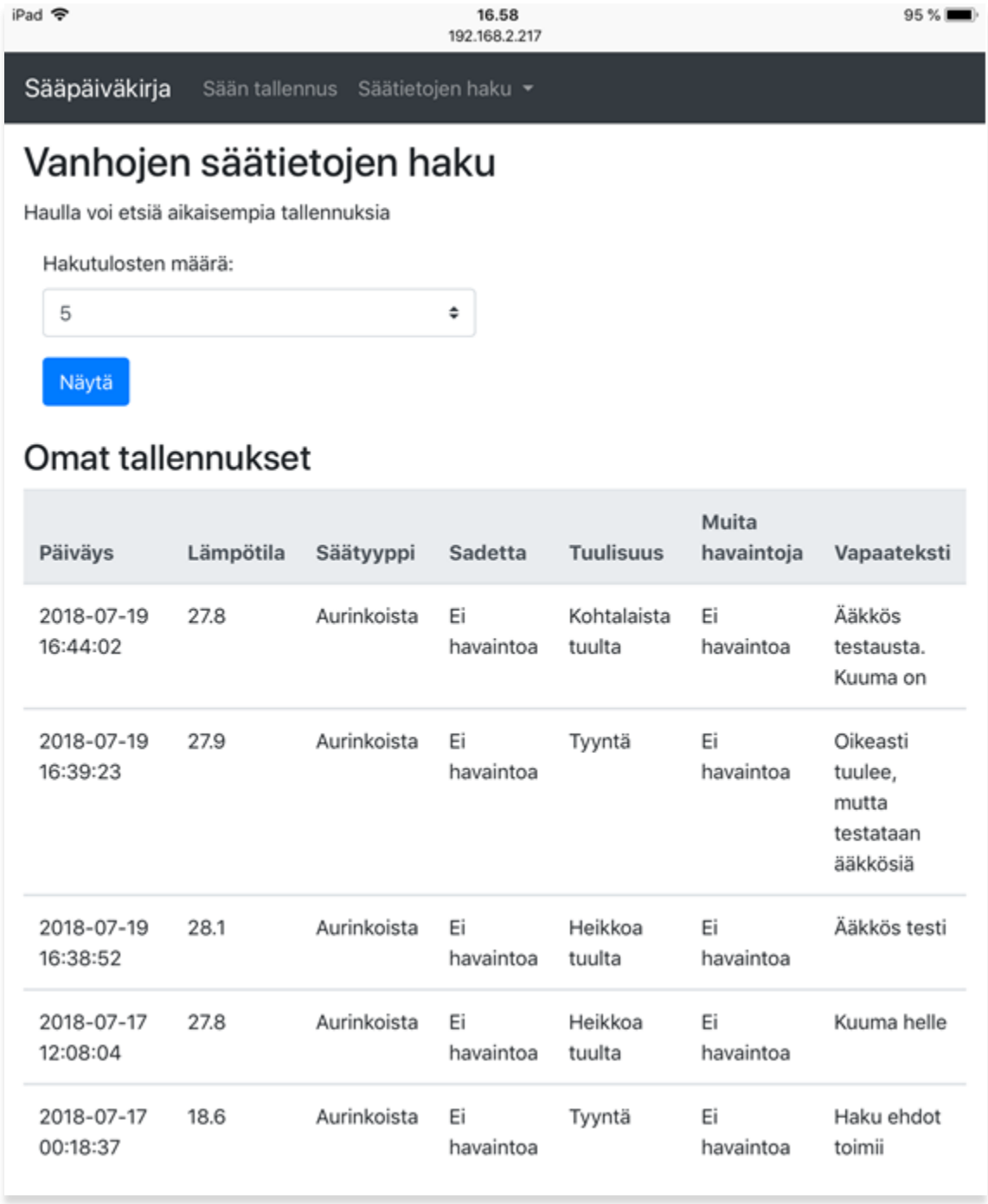
Säätiedot tallennettu

Tiedot tallennettu 2018-07-17 12:08:04

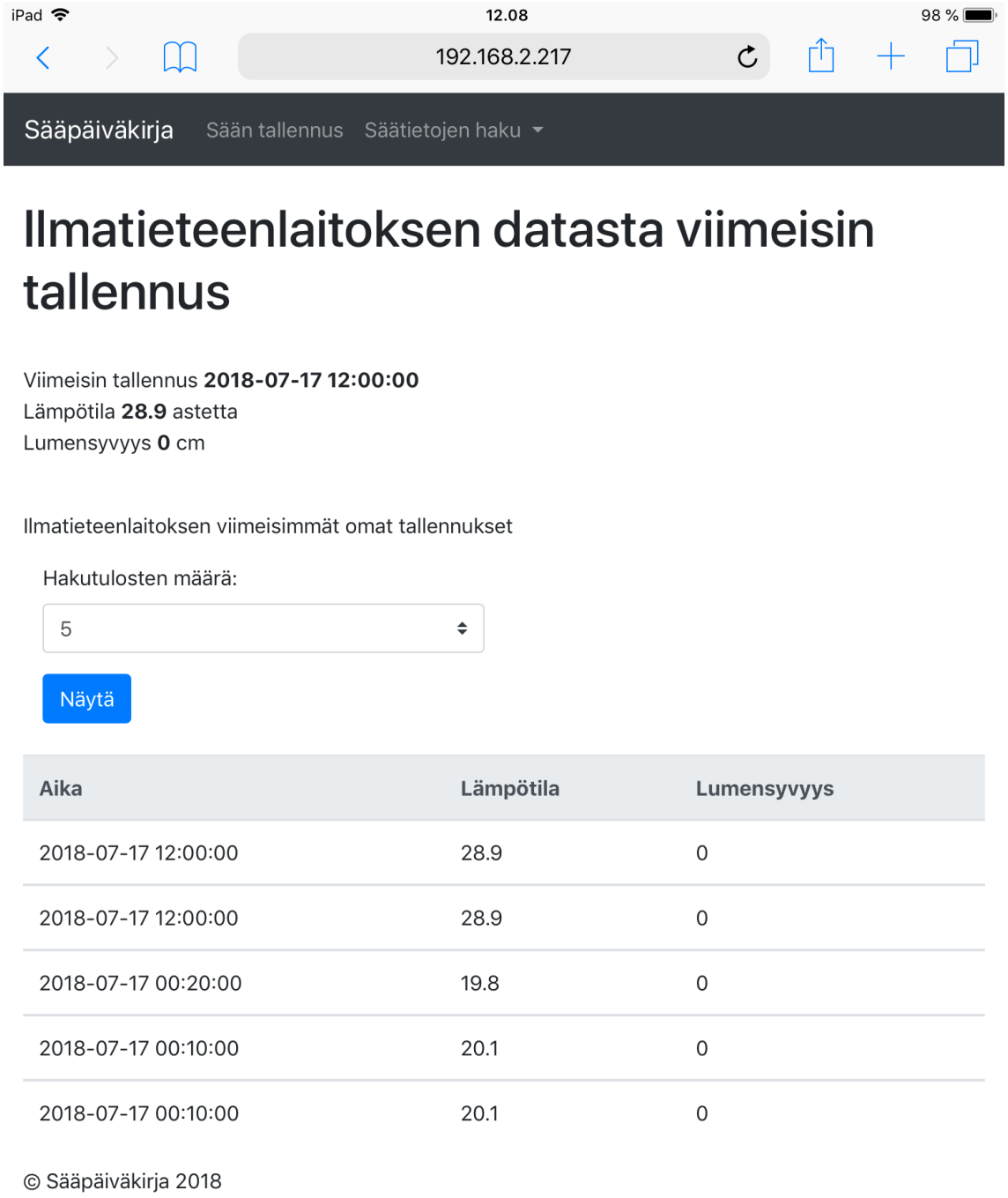
Ilmatieteenlaitoksen avoimesta datasta haetut tiedot:
Viimeisin tallennus **2018-07-17 12:00:00**
Lämpötila **28.9** astetta
Lumensyvyys **0** cm

© Sääpäiväkirja 2018

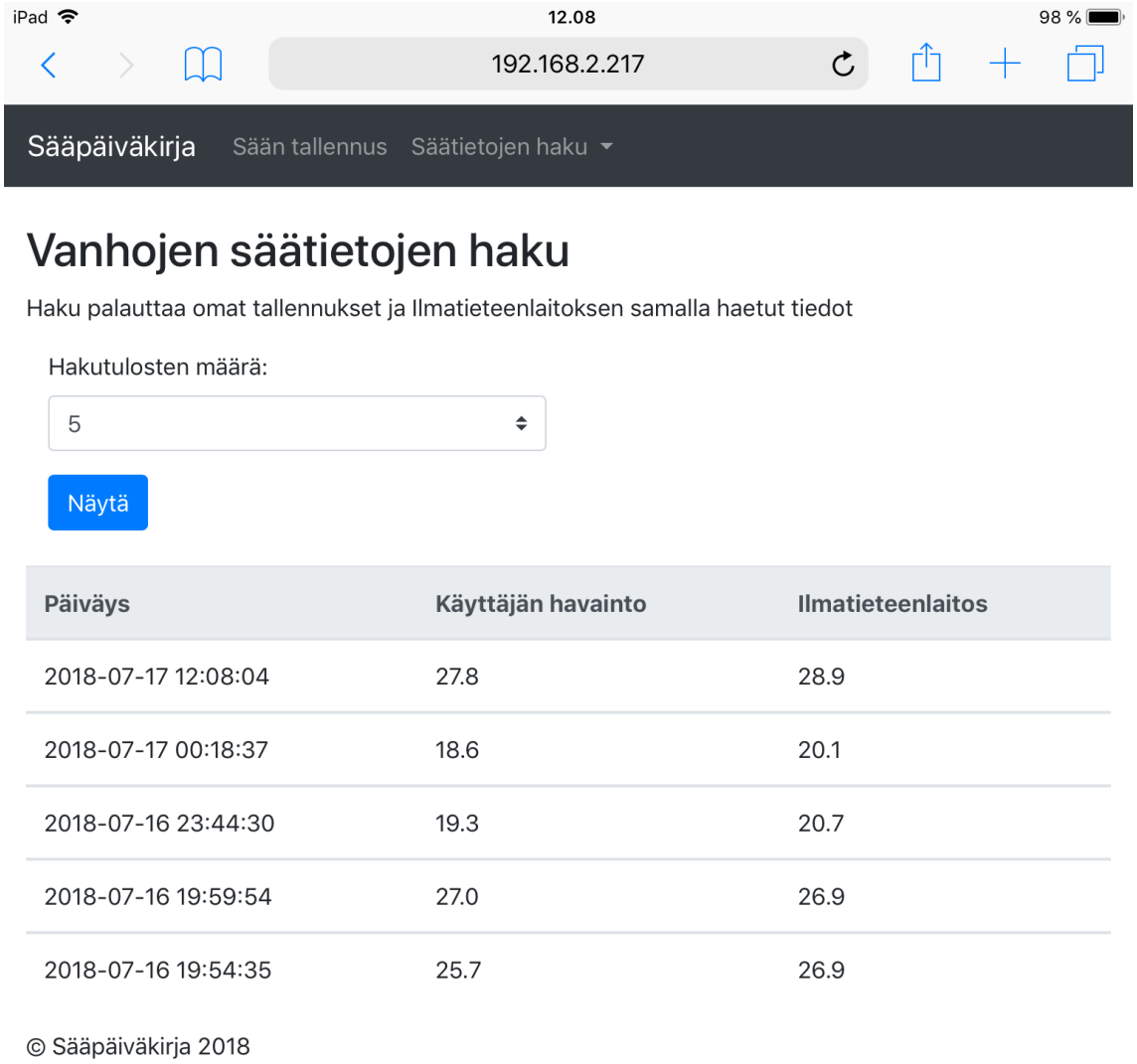
Hakutoiminto: Vanhojen säätietojen haku omat tallennukset ja samalla hetkellä haettu Ilmatieteen laitoksen säätiedot



Hakutoiminto: Ilmatieteen laitoksen datasta haettu sää nyt ja viimeisiä tallennuksia



Hakutoiminto: Omat tallennukset vs Ilmatieteen laitoksen säätieto



Ilmatieteenlaitos_parametrit

ParametriID:

Merkitys	Yksilöi jokaisen rivin		
Tietotyyppi *	integer	Arvoalue	
Pakollisuus*	On	Esitysmuoto	numero
Yksikäsitteisyys*	On	Oletusarvo	

Kysely:

Merkitys	Ilmatieteen laitoksen kysely parametritieto		
Tietotyyppi *	varchar	Arvoalue	1 – 40
Pakollisuus*	On	Esitysmuoto	vapaa
Yksikäsitteisyys*	On	Oletusarvo	

Parametri1:

Merkitys	Ilmatieteen laitoksen kysely tarkempi parametri		
Tietotyyppi *	varchar	Arvoalue	1 – 10
Pakollisuus*	Ei	Esitysmuoto	vapaa
Yksikäsitteisyys*	On	Oletusarvo	

Parametri2:

Merkitys	yksilöi päivän ja säätiedot		
Tietotyyppi *	varchar	Arvoalue	1 – 10
Pakollisuus*	Ei	Esitysmuoto	vapaa
Yksikäsitteisyys*	On	Oletusarvo	

Kayttaja

KayttajaID:

Merkitys	Yksilöi jokaisen rivin		
Tietotyyppi *	integer	Arvoalue	
Pakollisuus*	On	Esitysmuoto	numero
Yksikäsitteisyys*	On	Oletusarvo	

Kayttajanimi:

Merkitys	Käyttäjän nimi		
Tietotyyppi *	varchar	Arvoalue	1 – 50
Pakollisuus*	On	Esitysmuoto	vapaa
Yksikäsitteisyys*	On	Oletusarvo	

Apikey:

Merkitys	Ilmatieteen laitoksen haku avain		
Tietotyyppi *	varchar	Arvoalue	1 – 40
Pakollisuus*	On	Esitysmuoto	vapaa
Yksikäsitteisyys*	On	Oletusarvo	

Paikkakunta:

Merkitys	Oma paikkakunta		
Tietotyyppi *	varchar	Arvoalue	1 – 20
Pakollisuus*	On	Esitysmuoto	vapaa
Yksikäsitteisyys*	On	Oletusarvo	

Saaasema_fmi:

Merkitys	Ilmatieteen laitoksen sääasema, jota käytetään		
Tietotyyppi *	varchar	Arvoalue	1 – 20
Pakollisuus*	On	Esitysmuoto	vapaa
Yksikäsitteisyys*	On	Oletusarvo	

ParametriID:

Merkitys	Yksilöi halutut haku parametrit		
Tietotyyppi *	integer	Arvoalue	
Pakollisuus*	On	Esitysmuoto	numero
Yksikäsitteisyys*	On	Oletusarvo	

Ilmatieteenlaitos

IHavaintoID:

Merkitys	Yksilöi jokaisen rivin		
Tietotyyppi *	integer	Arvoalue	
Pakollisuus*	On	Esitysmuoto	numero
Yksikäsitteisyys*	On	Oletusarvo	seuraava vapaa

Milloin_tietokysytty:

Merkitys	Milloin tieto kysytään Ilmatieteen laitokselta		
Tietotyyppi *	timestamp	Arvoalue	
Pakollisuus*	On	Esitysmuoto	päiväys
Yksikäsitteisyys*	On	Oletusarvo	

Milloin_tietotallennettu:

Merkitys	Milloin Ilmatieteen laitos on tallentanut säätiedon		
Tietotyyppi *	timestamp	Arvoalue	
Pakollisuus*	On	Esitysmuoto	päiväys
Yksikäsitteisyys*	On	Oletusarvo	

Lampotila:

Merkitys	Lämpötila Ilmatieteen laitoksella		
Tietotyyppi *	desimal	Arvoalue	3,1
Pakollisuus*	On	Esitysmuoto	numero
Yksikäsitteisyys*	On	Oletusarvo	

Lumensyvyys:

Merkitys	Lumensyvyys Ilmatieteen laitoksella		
Tietotyyppi *	desimal	Arvoalue	3
Pakollisuus*	On	Esitysmuoto	numero
Yksikäsitteisyys*	On	Oletusarvo	

Saatyyppi

SaatyyppiID:

Merkitys	Yksilöi jokaisen rivin		
Tietotyyppi *	integer	Arvoalue	
Pakollisuus*	On	Esitysmuoto	numero
Yksikäsitteisyys*	On	Oletusarvo	

Tyyppi:

Merkitys	Eri säätyypit		
Tietotyyppi *	varchar	Arvoalue	1 – 25
Pakollisuus*	On	Esitysmuoto	vapaa
Yksikäsitteisyys*	On	Oletusarvo	

Sadetta

SadettaID:

Merkitys	Yksilöi jokaisen rivin		
Tietotyyppi *	integer	Arvoalue	
Pakollisuus*	On	Esitysmuoto	numero
Yksikäsitteisyys*	On	Oletusarvo	

Sadetyyppi:

Merkitys	Eri sade vaihtoehtoja		
Tietotyyppi *	varchar	Arvoalue	1 – 25
Pakollisuus*	On	Esitysmuoto	vapaa
Yksikäsitteisyys*	On	Oletusarvo	

Tuulisuus

TuuliID:

Merkitys	Yksilöi jokaisen rivin		
Tietotyyppi *	integer	Arvoalue	
Pakollisuus*	On	Esitysmuoto	numero
Yksikäsitteisyys*	On	Oletusarvo	

Tuulityyppi:

Merkitys	Eri tuulen kovuuksia		
Tietotyyppi *	varchar	Arvoalue	1 – 25
Pakollisuus*	On	Esitysmuoto	vapaa
Yksikäsitteisyys*	On	Oletusarvo	

Muita_havaintoja

MuitaID:

Merkitys	Yksilöi jokaisen rivin		
Tietotyyppi *	integer	Arvoalue	
Pakollisuus*	On	Esitysmuoto	numero
Yksikäsitteisyys*	On	Oletusarvo	

Muita_havaintoja_tyyppi:

Merkitys	Erilaisia muita havaintoja		
Tietotyyppi *	varchar	Arvoalue	1 – 25
Pakollisuus*	On	Esitysmuoto	vapaa
Yksikäsitteisyys*	On	Oletusarvo	

Vapaateksti

VapaaID:

Merkitys	Yksilöi jokaisen rivin		
Tietotyyppi *	integer	Arvoalue	
Pakollisuus*	On	Esitysmuoto	numero
Yksikäsitteisyys*	On	Oletusarvo	seuraava vapaa

Tyyppi:

Merkitys	Omat havainnot sivustolta		
Tietotyyppi *	varchar	Arvoalue	1 – 100
Pakollisuus*	On	Esitysmuoto	vapaa
Yksikäsitteisyys*	On	Oletusarvo	

Sov_havainto

SHavaintoID:

Merkitys	Yksilöi jokaisen rivin		
Tietotyyppi *	integer	Arvoalue	
Pakollisuus*	On	Esitysmuoto	numero
Yksikäsitteisyys*	On	Oletusarvo	seuraava vapaa

Paivays:

Merkitys	Milloin oma havainto tallennettu		
Tietotyyppi *	timestamp	Arvoalue	
Pakollisuus*	On	Esitysmuoto	päiväys
Yksikäsitteisyys*	On	Oletusarvo	

Lampotila:

Merkitys	Oma lämpötila havainto		
Tietotyyppi *	desimal	Arvoalue	3,1
Pakollisuus*	On	Esitysmuoto	numero
Yksikäsitteisyys*	On	Oletusarvo	

Saatyyppi:

Merkitys	Oma säätyyppi havainto		
Tietotyyppi *	integer	Arvoalue	
Pakollisuus*	On	Esitysmuoto	numero
Yksikäsitteisyys*	On	Oletusarvo	

SadettaID:

Merkitys	Oma sadetta havainto		
Tietotyyppi *	integer	Arvoalue	
Pakollisuus*	On	Esitysmuoto	numero
Yksikäsitteisyys*	On	Oletusarvo	

TuulisuusID:

Merkitys	Oma tuuli havainto

Tietotyyppi *	integer	Arvoalue	
Pakollisuus*	On	Esitysmuoto	numero
Yksikäsitteisyys*	On	Oletusarvo	

Muita_havaintojaID:

Merkitys	Omia muita havaintoja		
Tietotyyppi *	integer	Arvoalue	
Pakollisuus*	On	Esitysmuoto	numero
Yksikäsitteisyys*	On	Oletusarvo	

Vapaateksti:

Merkitys	Vapaateksti taulun ID numero viimeisin		
Tietotyyppi *	integer	Arvoalue	
Pakollisuus*	On	Esitysmuoto	numero
Yksikäsitteisyys*	On	Oletusarvo	

Kayt_havainto

Kayt_HavaintoID:

Merkitys	Yksilöi jokaisen rivin		
Tietotyyppi *	integer	Arvoalue	
Pakollisuus*	On	Esitysmuoto	numero
Yksikäsitteisyys*	On	Oletusarvo	seuraava vapaa

KayttajaID:

Merkitys	Kayttajan ID numero		
Tietotyyppi *	integer	Arvoalue	
Pakollisuus*	On	Esitysmuoto	numero
Yksikäsitteisyys*	On	Oletusarvo	

SHavaintoID:

Merkitys	Sov_havainto taulun ID numero viimeisin		
Tietotyyppi *	integer	Arvoalue	
Pakollisuus*	On	Esitysmuoto	numero
Yksikäsitteisyys*	On	Oletusarvo	

IHavaintoID:

Merkitys	Ilmatieteenlaitoksen taulun ID numero viimeisin		
Tietotyyppi *	integer	Arvoalue	
Pakollisuus*	On	Esitysmuoto	numero
Yksikäsitteisyys*	On	Oletusarvo	

TIETOKANNAN LUONTILAUSEET LIITE 7

```
CREATE DATABASE Saapaivakirja;

CREATE TABLE Ilmatieteenlaitos_parametrit (
   ParametriID INTEGER PRIMARY KEY,
   Kysely VARCHAR(40) NOT NULL,
   Parametri1 VARCHAR(10),
   Parametri2 VARCHAR(10));

CREATE TABLE Kayttaja (
   KayttajaID INTEGER PRIMARY KEY,
   Kayttajanimi VARCHAR(50) NOT NULL,
   Apikey VARCHAR(40),
   Paikakkunta VARCHAR(20) NOT NULL,
   Saaasema_fmi VARCHAR(20) NOT NULL,
   ParametriID INTEGER NOT NULL,
   FOREIGN KEY (ParametriID) REFERENCES Ilmatieteenlaitos_parametrit(ParametriID));

CREATE TABLE Ilmatieteenlaitos (
   IHavaintoID INTEGER AUTO_INCREMENT,
   Milloin_tietokysytty TIMESTAMP NOT NULL,
   Milloin_tietotallennettu TIMESTAMP NOT NULL,
   Lampotila DECIMAL(3,1),
   Lumensyvyys DECIMAL(3),
   PRIMARY KEY (IHavaintoID));

CREATE TABLE Saatyyppi (
   SaatyyppiID INTEGER PRIMARY KEY,
   Tyyppi VARCHAR(25) NOT NULL);

CREATE TABLE Sadetta (
   SadettaID INTEGER PRIMARY KEY,
```

```
    Sadetyyppi VARCHAR(25) NOT NULL);

CREATE TABLE Tuulisuus (
    TuuliID INTEGER PRIMARY KEY,
    Tuulityyppi VARCHAR(25) NOT NULL);

CREATE TABLE Muita_havaintoja (
    MuitaID INTEGER PRIMARY KEY,
    Muita_havaintoja_tyyppi VARCHAR(25) NOT NULL);

CREATE TABLE Vapaateksti (
    VapaaID INTEGER AUTO_INCREMENT,
    Teksti VARCHAR(100) NOT NULL,
    PRIMARY KEY (VapaaID));

CREATE TABLE Sov_havainto (
    SHavaintoID INTEGER AUTO_INCREMENT,
    Paivays TIMESTAMP NOT NULL,
    Lampotila DECIMAL(3,1) NOT NULL,
    SaatyyppiID INTEGER NOT NULL,
    SadettaID INTEGER NOT NULL,
    TuulisuusID INTEGER NOT NULL,
    Muita_havaintoja INTEGER NOT NULL,
    Vapaateksti INTEGER NOT NULL,
    FOREIGN KEY (SaatyyppiID) REFERENCES Saatyyppi(SaatyyppiID),
    FOREIGN KEY (SadettaID) REFERENCES Sadetta(SadettaID),
    FOREIGN KEY (Muita_havaintoja) REFERENCES Muita_havaintoja(MuitaID),
    FOREIGN KEY (TuulisuusID) REFERENCES Tuulisuus(TuuliID),
    FOREIGN KEY (Vapaateksti) REFERENCES Vapaateksti(VapaaID),
    PRIMARY KEY (SHavaintoID));

CREATE TABLE Kayt_havainto (
    Kayt_HavaintoID INTEGER AUTO_INCREMENT,
    KayttajaID INTEGER NOT NULL,
```

```
SHavaintoID INTEGER NOT NULL,
lHavaintoID INTEGER NOT NULL,
FOREIGN KEY (KayttajaID) REFERENCES Kayttaja(KayttajaID),
FOREIGN KEY (SHavaintoID) REFERENCES Sov_havainto(SHavaintoID),
FOREIGN KEY (lHavaintoID) REFERENCES Ilmatieteenlaitos(lHavaintoID),
PRIMARY KEY (Kayt_HavaintoID));
```

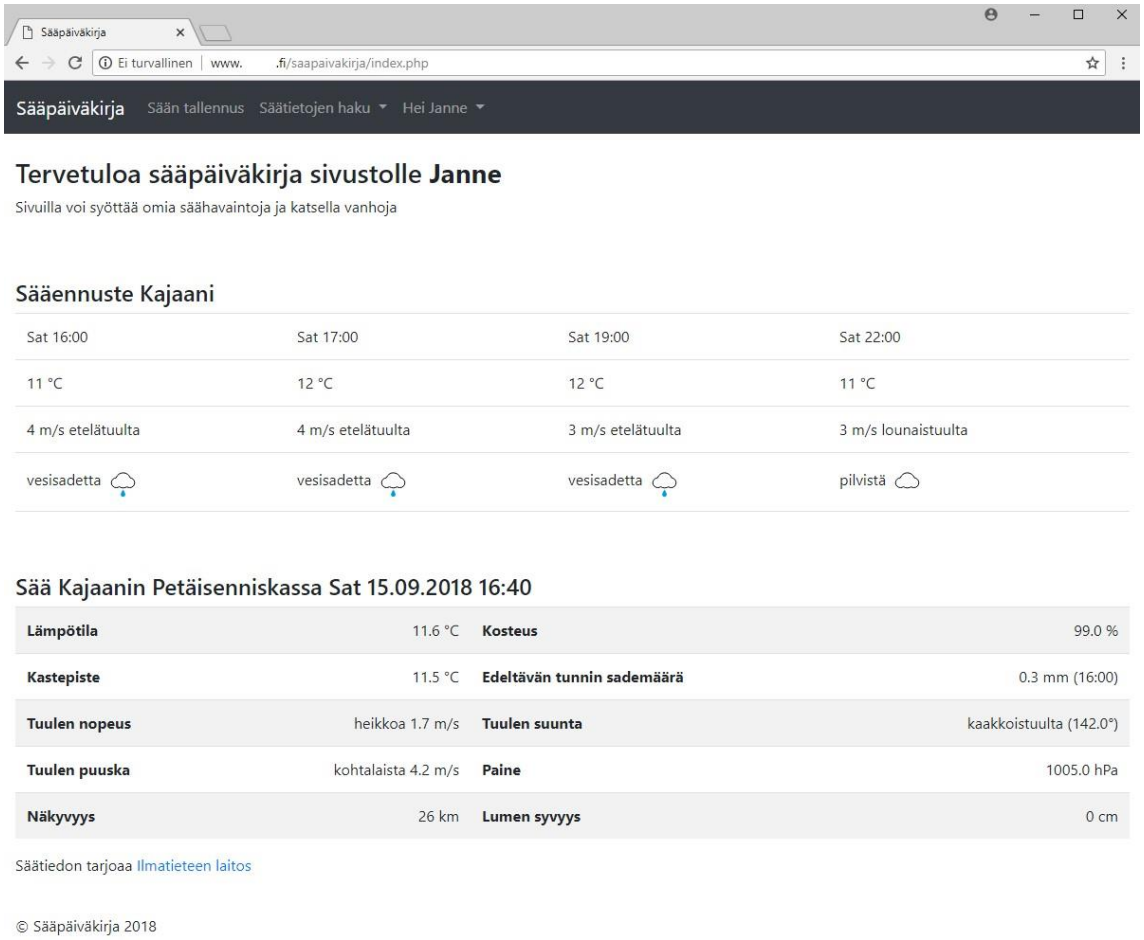
```
INSERT INTO Saatyyppi VALUES (1, 'Aurinkoista');
INSERT INTO Saatyyppi VALUES (2, 'Pilvistä');
INSERT INTO Saatyyppi VALUES (3, 'Vaihtelevaa pilvisyyttä');

INSERT INTO Sadetta VALUES (1, 'Ei havaintoa');
INSERT INTO Sadetta VALUES (2, 'Vesi');
INSERT INTO Sadetta VALUES (3, 'Lumi');
INSERT INTO Sadetta VALUES (4, 'Räntä');

INSERT INTO Muita_havaintoja VALUES (1, 'Ei havaintoa');
INSERT INTO Muita_havaintoja VALUES (2, 'Ukkosta');
INSERT INTO Muita_havaintoja VALUES (3, 'Sumua');
INSERT INTO Muita_havaintoja VALUES (4, 'Rakeita');

INSERT INTO Tuulisuus VALUES (1, 'Tyyntä');
INSERT INTO Tuulisuus VALUES (2, 'Heikkoa tuulta');
INSERT INTO Tuulisuus VALUES (3, 'Kohtalaista tuulta');
INSERT INTO Tuulisuus VALUES (4, 'Kovaa tuulta');
INSERT INTO Tuulisuus VALUES (5, 'Myrsky tuuli');

INSERT INTO Ilmatieteenlaitos_parametrit VALUES (1, 'fmi::observations::weather::simple', 't2m',
'snow_aws');

INSERT INTO Kayttaja VALUES (1, 'Janne', 'OMA_APIKEY_TÄHÄN', 'Kajaani', 'Kajaani', 1);
```

PHP SIVUN PARAMETRIT SÄÄTIETOJEN HAKUUN ILMATIETEEN LAITOKSEN TIETOKANNASTA LIITE 9

```
<?php

// Muodostetaan tietokanta yhteys
require_once('tietokanta_yhteys.php');

// Muutetaan aika asetukseksi -4 tuntia
date_default_timezone_set('Atlantic/Azores');

// Otetaan päiväys ja tunti aika
$paivanyt = date("Y-m-d");
$kellonyt = date("H");
$kirjain = "T";

// Tehdään yhteen pötköön päiväys, lisätään T kirjain, tunti ja lisään minuuteiksi 00.
$options["starttime"] = "$paivanyt$kirjain$kellonyt:00";

// Palautetaan oikea aikavyöhyke
date_default_timezone_set('Europe/Helsinki');

// Tietokannasta tarvittavat parametrit
$query = "select apikey, saaasema_fmi, parametri1, parametri2, kysely from Kayttaja
inner join Ilmatieteenlaitos_parametrit on Kayttaja.ParametriID = Ilmatieteenlaitos_parametrit.ParametriID";
$response = mysqli_query($conn, $query);
if ($response) {
    while ($rivi = mysqli_fetch_array($response)) {
        $options["apikey"] = $rivi["apikey"];
        $options["place"] = $rivi["saaasema_fmi"];
        $options["stored_query_id"] = $rivi["kysely"];
        $options["parameters1"] = $rivi["parametri1"];
```

```
        $options["parameters2"] = $rivi["parametri2"];
    }
    mysqli_close($conn);
}

// Haetaan kellonaika ja lämpötila
$data = file_get_contents("http://data.fmi.fi/fmi-apikey/" . $options["apikey"] . "/wfs?request=get-
Feature&storedquery_id=" . $options["stored_query_id"] . "&starttime=" . $options["starttime"] .
"&place=" . $options["place"] . "&parameters=" . $options["parameters1"]);
$time = "<BsWfs:Time>(.*?)<\/BsWfs:Time>";
$parametervalue = "<BsWfs:ParameterValue>(.*?)<\/BsWfs:ParameterValue>";
preg_match_all("/$time|$parametervalue/", $data, $arvo);
for ($i = 0; $i < count($arvo[0]); $i++) {
    if ($arvo[1][$i] == "" && $arvo[2][$i] == "") {
    } else if ($arvo[1][$i] != "") {
        // Konvertoidaan tuleva UTC aika Helsingin ajanmuotoon
        $aika_muutos = strtotime("{$arvo[1][$i]}");
        $aika = date("Y-m-d H:i:s", $aika_muutos);
    } else if ($arvo[2][$i] != "") {
        $lampo = "{$arvo[2][$i]}";
    }
}

// Haetaan lumen syvyys
$data1 = file_get_contents("http://data.fmi.fi/fmi-apikey/" . $options["apikey"] . "/wfs?request=get-
Feature&storedquery_id=" . $options["stored_query_id"] . "&starttime=" . $options["starttime"] .
"&place=" . $options["place"] . "&parameters=" . $options["parameters2"]);
preg_match_all("/$parametervalue/", $data1, $arvo1);
for ($i = 0; $i < count($arvo1[0]); $i++) {
    if ($arvo1[1][$i] == "") {
    } else if ($arvo1[1][$i] != "") {
        $lumi = "{$arvo1[1][$i]}";
        if ($lumi > 0) {
```

```
            // Jos lunta on enemmän kuin 0, niin syötetään oikea arvo, muuten syötetään 0. Kun lunta
ei ole, tulee paluu arvona -1
        } else {
            $lumi = 0;
        }
    }
}
```


Sääpäiväkirja
Ei turvallinen | www. .fi/saapaivakirja/index.php
Sääpäiväkirja
Sään tallennus
Säätietojen haku
Hei Janne
Tervetuloa sääpäiväkirja sivustolle Janne
Sivuilla voi syöttää omia säähavaintoja ja katsella vanhoja
Sääennuste Kajaani
Sat 16:00
Sat 17:00
Sat 19:00
Sat 22:00
11 °C
12 °C
12 °C
11 °C
4 m/s etelätuulta
4 m/s etelätuulta
3 m/s etelätuulta
3 m/s lounaistuulta
vesisadetta
vesisadetta
vesisadetta
pilvistä
Sää Kajaanin Petäisenniskassa Sat 15.09.2018 16:40
Lämpötila 11.6 °C
Kosteus 99.0 %
Kastepiste 11.5 °C
Edeltävän tunnin sademäärä 0.3 mm (16:00)
Tuulen nopeus heikkoa 1.7 m/s
Tuulen suunta kaakkoistuulta (142.0°)
Tuulen puuska kohtalaista 4.2 m/s
Paine 1005.0 hPa
Näkyvyys 26 km
Lumen syvyys 0 cm
Säätiedon tarjoaa Ilmatieteen laitos
© Sääpäiväkirja 2018