Tomi Aalto

DEWIL-POHJAINEN ANTUREIDEN TESTAUSYMPÄRISTÖ

Tietotekniikan koulutusohjelma

2015

samk
Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

DEWIL-POHJAINEN ANTUREIDEN TESTAUSYMPÄRISTÖ

Aalto, Tomi
Satakunnan ammattikorkeakoulu
Tietotekniikan koulutusohjelma
Marraskuu 2015
Ohjaaja: Trast, Ismo
Sivumäärä: 42

Asiasanat: DEWIL, XML, PHP, web-ohjelmointi, MySQL

Opinnäytetyön tarkoituksena oli luoda käyttöliittymä DEWIL-nimiseen antureiden mittausympäristöön. DEWIL-ympäristö on Satakunnan Ammatikorkeakoulussa toimivan RETEE-projektiryhmän kehityksessä ja heillä oli tarve saada ympäristöön graafinen käyttöliittymä, jolla mittausympäristöön kuuluvia tiedostoja pystytään hallitsemaan. Tavoitteena oli pystyä muokkaamaan ympäristöön kuuluvia konfiguraatiotiedostoja ja tietokantoja siten, että loppukäyttäjän ei tarvitse itse tietää ympäristön taustalla toimivasta koodista mitään tai joutua käsin muokkaamaan kyseisiä tiedostoja tai tietokantoja. Käyttöliittymään tehtiin myös ominaisuus, jolla pystytään piirtämään kuvaajia DEWIL-ympäristön keräämästä datasta.

Käyttöliittymä toteutettiin internet-selain pohjaiseksi, joten sen toteuttamiseen käytettiin erilaisia web-ohjelmointitekniikoita. Keskeisimmät menetelmät olivat HTML, CSS, PHP, JavaScript ja MySQL.

DEWIL based sensor testing environment

Aalto, Tomi
Satakunnan ammattikorkeakoulu, Satakunta University of Applied Sciences
Degree Programme in Information Technology
November 2015
Supervisor: Trast, Ismo
Number of pages: 42

Keywords: DEWIL, XML, PHP, web-programming, MySQL

The purpose of this thesis was to create an interface for DEWIL sensor measurement environment. DEWIL is developed by a RETEE project group which operates in the Satakunta University of Applied Sciences and they had a need for an interface to modify the files of the DEWIL environment. The created interface was supposed to be able to modify the databases and the files of the DEWIL enviroment so that the end-user does not have to understand the code behind the DEWIL environment itself or need to modify its files by hand. The interface also needed to include a feature that it is possible to draw graphs from the data that DEWIL has gathered.

The Interface was made to work in the internet browser so different kind of web programming techniques were used. The most crucial techniques were HTML, CSS, PHP, JavaScript and MySQL.

SISÄLLYS

LYHENTEET JA TERMIT

DEWIL	Device Ecosystem with Infinite Limits. Opinnäytetyössä käytetty mittausympäristö
HTML	Hypertext Markup Language. Merkintäkieli, johon web-sivujen luominen pohjautuu.
PHP	Hypertext Preprocessor. Ohjelmointikieli, jota käytetään web-sivujen dynaamisuuden luomiseen.
CSS	Cascading Style Sheets. Tämän avulla pystytään tyylittelemään web-sivuja.
MySQL	Relaatiotietokantaohjelmisto, jota käytetään tiedon tallentamiseen.
XML	Extensible Markup Language. Merkintäkieli, joka on suunniteltu tallentamaan ja siirtämään dataa.
JavaScript	Ohjelmointikieli, jota käytetään web-sivujen dynaamisuuden luomiseen.
AJAX	Asynchronous JavaScript and XML. Käytetään tietojen välittämiseen web-sivun ja palvelimen välillä ilman että sivua tarvitsee ladata uudelleen.
1-Wire	Protokolla sarjamuotoisen datan siirtämiseen.
jQuery	jQuery on JavaScript-kirjasto, joka on tarkoitettu JavaScript-ohjelmoinnin helpottamiseen.
Apache2	Opinnäytetyössä käytetty web-palvelin.

1 JOHDANTO

Opinnäytetyössä luotiin käyttöliittymä Satakunnan Ammattikorkeakoulussa kehityksessä olevalle DEWIL-pohjaiselle antureiden mittausympäristölle. Satakunnan ammattikorkeakoulussa toimii RETEE-niminen projektiryhmä, jonka toimialaa ovat lähinnä erilaiset aurinkoenergiaprojektit. Heillä oli tarve saada antureiden mittausympäristöön käyttöliittymä, jonka avulla loppukäyttäjä pystyy tekemään haluttuja operaatioita mittausympäristössä ilman, että hänen tarvitsee koskea ollenkaan itse ympäristön koodiin. Näitä operaatioita ovat mm. mittausten käynnistäminen, ympäristön konfiguraatiotiedostojen muokkaaminen, tietokantakyselyjen tekeminen ja mittaustulosten esittäminen graafisessa muodossa.

Toteutusvaiheessa käytössä oli 1-wire protokollaan perustuva mittauslaitteisto, joka sisältää kolme anturia. Kahdella antureista pystyi mittaataan lämpötilaa ja yhdellä lämpötilan lisäksi pariston varausta. Näiden antureiden avulla testausta oli mahdollista suorittaa koko projektin ajan pienissä osissa.

2 MENETELMÄT JA YMPÄRISTÖ

2.1 Fit-PC2

Käytössä oli pienikokoinen energiatehokas PC ilman tuuletinta, joka soveltuu opinnäytetyötä varten hyvin, koska se vie vain vähän tilaa mittauspaikalla.
Oleelliset tiedot laitteesta:
Koko: 101x115x27mm
Prosessori: Intel Atom Z5xx CPU 1.1-1.6Hz
Virrankulutus: lepotilassa 5W ja rasituksessa 8W
Muisti: 1GB
Kiintolevy: 2.5” SATA HDD /1/

2.2 Linux Mint

Linux Mint on ilmainen ja avoimeen lähdekoodiin pohjautuva käyttöjärjestelmä, joka perustuu Ubuntuun. Linux Mint-käyttöjärjestelmää päädyttiin käyttämään, koska se oli jo entuudestaan tuttu. Vertailemalla erilaisia Ubuntu-pohjaisia käyttöjärjestelmiä huomattiin, että Linux Mint toimi näistä käyttöjärjestelmistä parhaiten käytössä olevalla tietokoneella. Linux Mintistä asennettiin versio 17.1, jota kutsutaan myös nimellä Rebecca. Tämä versio oli uusin asennushetkellä 2015.

2.3 DEWIL

DEWIL eli Device Ecosystem with Infinite Limits on modulaarinen mittaus ja -ohjausympäristö, jota kehitetään Satakunnan Ammattikorkeakoulussa. DEWIL on toteutettu Java- ja XML-kielillä ja on toiminnaltaan client-server pohjainen. Client sisältää rajapinnan haluttujen päätelaitteiden osalta ja välittää näille dataa. Jokaiselle erilaista toiminnallisuutta toteuttavalle päätelaitteelle on syytä tehdä oma client-rajapintansa. Clientille oleelliset konfiguraatiotiedot on tallennettu client-kohtaiseen XML-tiedostoon. Server-puolella tehdään saaduille mittaustuloksille haluttuja operaatioita, joista oleellisin toiminto tämän opinnäytetyön kannalta on kerätyn datan tallennus MySQL-tietokantaan /2/.

2.4 HTML ja CSS

HTML ja CSS ovat pääasialliset menetelmät web-sivujen rakentamisessa. HTML-merkintäkielen avulla pystytään määrittelemään sivun rakenne, kun taas CSS-kielellä määritellään sivun tyyli ja visuaalinen ilme /3/. HTML-sivu koostuu erilaisista elementeistä, joista keskeisimmät ovat HTML, BODY, HEAD ja TITLE. Kaikilla elementeillä on usein aloitus- ja lopetusmerkintänsä, esimerkiksi HTML-elementti alkaa merkinnällä <HTML> ja loppuu merkintään </HTML>. HTML-elementti paketoi koko sivun yhteen ja kaikki HTML-dokumentin rakenteelliset seikat tulevatkin HTML-elementin sisälle. HEAD-elementti pitää sisällään muuta sivustoa koskevaa tietoa, esimerkiksi JavaScript-tiedostot sekä CSS-tyylitiedostot. BODY-

elementin sisälle tulee itse varsinainen web-sivun sisältö, esimerkiksi sivulla näkyvä teksti ja linkit. TITLE-elementti pitää sisällään sivun otsikon, joka yleensä näkyy selaimen yläpalkissa /4/.
CSS-tiedostossa taas pystytään määrittämällä kullekkin HTML-elementille omanlaisensa visuaalinen tyyli.

2.5 PHP

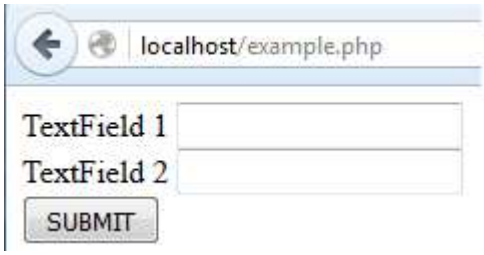
PHP on skriptikieli, joka on tarkoitettu web-sivujen dynaamisuuden eli toiminnallisuuksien ohjelmointiin. PHP perustuu avoimeen lähdekoodiin ja kaikki yleisimmät internet-selaimet tukevat sitä. Toimiakseen PHP vaatii palvelimen, jolla skriptejä suoritetaan. PHP-kieli poikkeaa esim. JavaScriptistä siten, että kaikki PHP-koodi suoritetaan palvelimella, eikä selaimessa. PHP-koodia voidaan kirjoittaa, joko HTML-koodin sisään <?php ja ?> merkintöjen väliin tai vaihtoehtoisesti voidaan tehdä oma PHP-skriptitiedosto, jota kutsutaan HTML-koodissa /9/. PHP tukee myös laajasti erilaisia tietokantoja, joista opinnäytetyössä käytin MySQL-tietokantaa. Lähes kaikki tietokantoja käsittelevät PHP-funktiot sijoitettiin erilliseen tiedostoon nimeltä dbFunctions.php. Tässä tiedostossa olevia PHP-skriptejä pystytään kutsumaan muista PHP-tiedostoista kirjoittamalla PHP-koodin alkuun rivi:

```
include_once (“dbFunctions.php”);
```

2.5.1 HTML-lomakkeen käsittely PHP-kielellä

Erittäin keskeinen osa toiminnallisuuden luomisesta PHP-kielellä web-sivulle on HTML-lomakkeen käsittely. Opinnäytetyössä lomakkeen käsittelyä PHP-kielellä

tarvittiin mm. tietokantojen ja tiedostojen muokkaamisessa.



Kuva 2.1 Yksinkertainen HTML-lomake

Kuvassa 2.1 on esitetty yksinkertainen HTML-lomake, joka sisältää kaksi tekstikenttää ja submit-napin eli tietojen lähetyspainikkeen.

```
<html>
<head>
</head>
<title>Example</title>
<body>
<form method = "POST">
TextField 1 <input type = "text" name = "tf1"/></br>
TextField 2 <input type = "text" name = "tf2"/></br>
<input type = "submit" name = "sub" value = "SUBMIT"/>
</form>

<?php
if(isset($_POST["sub"])){
$a = $_POST["tf1"];
$b = $_POST["tf2"];

echo $a . "</br>" . $b;

}
?>

</body>
</html>
```

Kuva 2.2. Kuvassa 2.1 näkyvän HTML-lomakkeen käsittely PHP-kielellä

Kuvassa 2.2 on esitetty, kuinka PHP-koodin avulla saadaan tulostettua kuvan 2.1 lomakkeen tekstikenttiin syötetyt tiedot. Aluksi ollaan määritelty lomakkeen metodiksi POST, jolloin tekstikenttiin syötetty data ei näy osoiterivillä. GET-metodia käyttämällä data näkyisi osoiterivillä /13/. FORM-elementissä ei ole määritelty PHP-koodin käsittelevää PHP-sivua, joten lomakkeen omaava sivu käsittelee PHP-koodin itse. Seuraavaksi on HTML-elementeillä luotu kaksi tekstikenttää ja lomakkeen lähetyspainike. Varsinaisen PHP-koodin sisällä on ensin tarkastettu funktiolla *isset()* onko lomakkeen lähetyspainiketta painettu ja jos näin on, niin tallennetaan muuttujaan *$a* tekstilaatikon1 sisältö ja muuttujaan *$b* tekstilaatikon2 sisältö. Sitten molemmat muuttujat tulostetaan sivulle allekkain käyttämällä komentoa *echo*.

2.6 MySQL

MySQL on avoimeen lähdekoodiin perustuva relaatiotietokantaohjelmisto, joka on tarkoitettu tiedon pysyvään tallentamiseen. Toimiakseen MySQL vaatii MySQL-palvelimen, joka pitää sisällään kaikki tietokannat. Tietokannat muodostuvat erilaisista tauluista, jotka itsessään sisältävät varsinaisen tallennetun datan. Tauluun voidaan määritellä haluttu määrä sarakkeita erilaisille ominaisuuksille /20/.

ID	Client	Sensor	Measurement	Status	Description
INT	Varchar(32)	Varchar(32)	Varchar(32)	On/Off(Varchar(32))	Varchar (32)

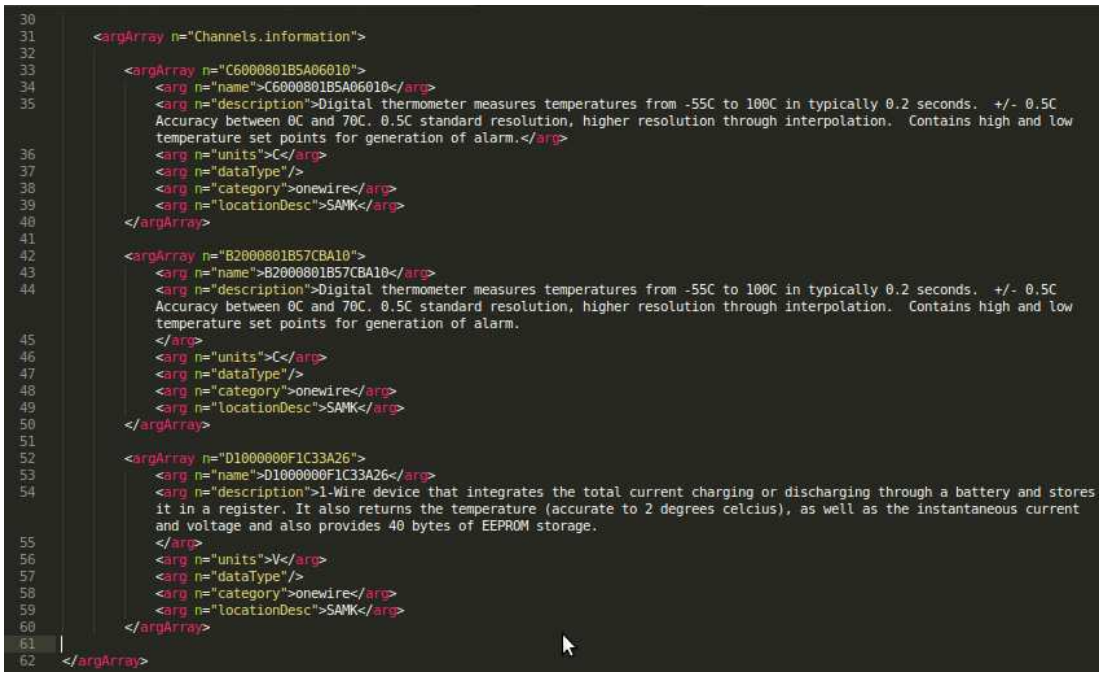
Taulukko 2.1. Opinnäytetyötä varten tehdyn lisätaulun rakenne

Taulukossa 2.1 kuvattu taulu “clients” sisältää kaikki clientit, anturit, antureiden ID-numerot, mittaustyypin, status-tilan ja anturin kuvauksen. Status-kenttää käytetään määrittelemään onko anturi sillä hetkellä määriteltynä clientin konfiguraatiotiedostossa. Tämän kentän ansiosta pystytään hakemaan konfiguraatiotiedostosta poistettujen antureitten mittaustuloksia, mutta samalla estetään jo poistetun anturin uudelleen poistaminen, koska sitä ei enää tulosteta antureiden käsittelyssä käytettävään alasvetovalikkoon. Mittaustyyppi puolestaan sisältää tiedon siitä minkälaista suuretta kyseisellä anturilla mitataan esim. lämpötila tai ilmankosteus. Muiden taulussa olevien kenttien käyttötarkoitus ilmenee suoraan otsikkokentästä. Tämä tietokanta on perustana kaikille opinnäytetyössä oleville elementeille joissa listataan clientejä tai antureita tietokannasta.

2.7 XML

XML on tekstityyppiä oleva tallennusmuoto, jonka tarkoituksena on esittää rakenteellista tietoa. Tämän opinnäyten kannalta oleellinen XML-muodossa oleva

tieto oli DEWIL-clientien konfiguraatiotiedostot. Erona HTML-merkkaukseen on se, että kaikki XML-elementit on välttämätöntä sulkea. /5/



Kuva 2.3. Antureiden tiedot XML-konfiguraatiotiedostossa

Kuvassa 2.3 nähdään kuinka XML-elementtejä on käytetty DEWIL-antureiden tietojen tallennuksessa. Ensimmäinen argArray-elementti pitää sisällään kaikki antureiden kuvaukseen käytettävät elementit. Elementille on annettu n-attribuutin kautta sitä kuvaava otsikko “Channels.information”.

Muut elementit on toteutettu niin, että aina argArray-elementin n-attribuuttina on annettu anturin nimi ja sen sisällä on arg-elementtejä joihin kuvaukset on kirjoitettu. Edellisten kohtien tapaan n-attribuuttiin on tallennettu kuvauksen otsikko.

Kuva 2.4. Mitattavan datan tiedot XML-konfiguraatiotiedostossa

Kuvassa 2.4 nähdään kuvan 2.3 konfiguraatiotiedoston loppuosa, jossa on määritelty käytettyjen kanavien määrä ja antureilla mitattavan datan tyyppi. Kanavilla tarkoitetaan tässä tapauksessa sitä, kuinka monta erilaista mittausta clientillä on tällä hetkellä mahdollista suorittaa. Elementit muodostetaan samaan tapaan kuin edellisessäkin kohdassa. Anturikohtaisen elementin sisälle on listattu arg-elementissä montako erilaista mittausta kyseisellä anturilla pystyy tekemään ja minkä tyyppisiä ne ovat. Kahdella ensimmäisella pystytään mittaamaan vain lämpötilaa ja viimeisellä sekä lämpötilaa että sähkövarausta.

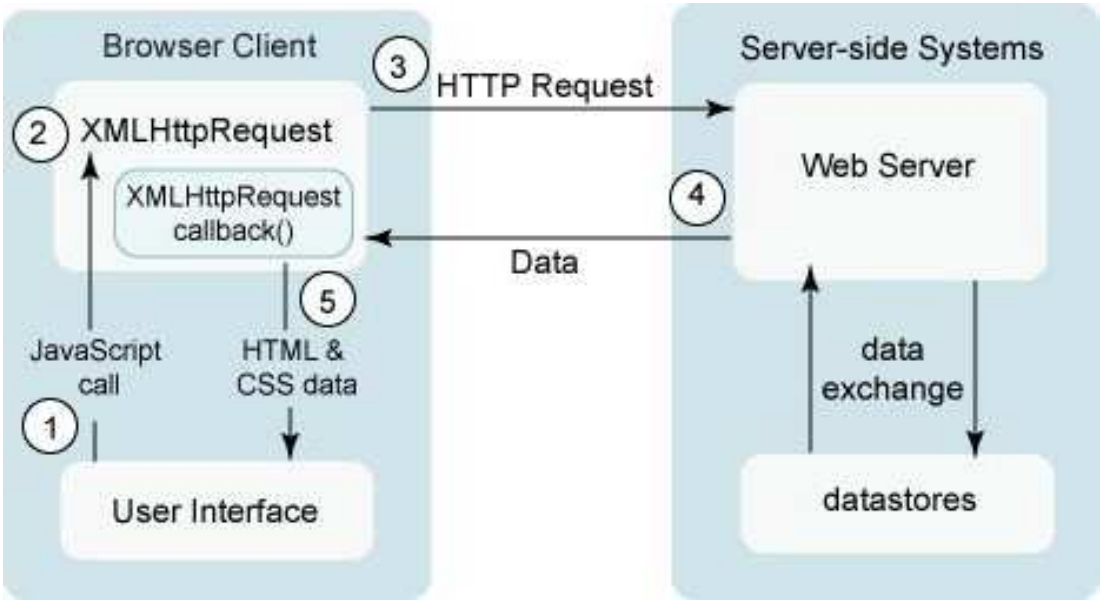
2.8 JavaScript

JavaScript on ohjelmointikieli, jolla on mahdollista lisätä web-sivulle dynaamisuutta. JavaScript-koodi suoritetaan asiakkaan selaimessa, mutta JavaScriptin saa myös välittämään tietoa palvelimelle käyttämällä AJAX-kutsuja. JavaScript-koodi sijoitetaan joko HTML-sivun HEAD-elementin sisälle kirjoitamalla tiedoston sisältävän hakemiston polku ja tiedostonimi. Toinen vaihtoehto sisällyttää JavaScript-koodi sivulle on laittaa HTML-sivun BODY-elementin sisälle SCRIPT-elementti ja kirjoittaa koodi siihen. /6/ Opinnäytetyössä valittiin jälkimmäisenä mainittu

lähestymistapa, koska JavaScript-koodilohkot eivät olleet yhdessäkään tiedostossa erityisen pitkiä.

2.8.1 AJAX

AJAX:n avulla pystytään lähettämään ja hakemaan tietoa palvelimelta tietoa, ilman että web-sivua tarvitsee välissä päivittää.



Kuva 2.5. AJAX-kutsun rakenne

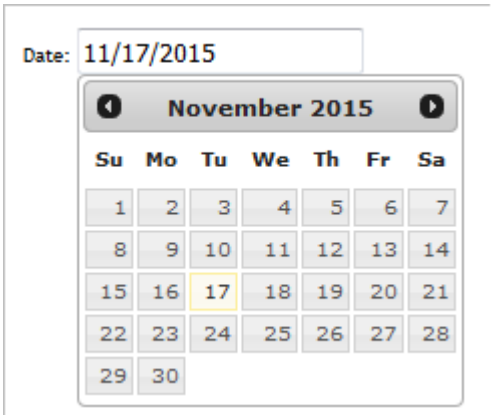
(Lähde: http://www.webstepbook.com/supplements/slides/ch10-ajax_xml.shtml)

Aluksi AJAX-kutsu pitää laukaista web-sivulta jollakin tapaa esim. klikkaamalla tai muuttaamalla alasvetovalikon tietuetta. Tämän jälkeen AJAX-kutsu lähetetään eteenpäin XMLHttpRequest-objektina palvelimelle, josta haluttu tieto haetaan. Kun palvelimelta on saatu haettua haluttu data XMLHttpRequest-objekti laukaisee nk. *callback()*-tapahtuman, joka tuo datan HTML-sivulle

2.8.2 jQuery ja jQueryUI

jQuery on JavaScript-kirjasto, jonka tarkoituksena on helpottaa JavaScript-ohjelmointia, varsinkin AJAX-kutsujen muodostamista /14/. Kirjaston asentaminen tapahtuu lataamalla jQuery.js-kirjasto ja sitten linkittämällä kirjaston osoite ja tiedostonimi HTML-tiedoston HEAD-elementin sisälle /15/.

jQueryUI-kirjasto on puolestaan tarkoitettu helpottamaan JavaScript-pohjaisten käyttöliittymien ohjelmointia. Kirjastolla on mahdollista luoda hyvinkin erilaisia toiminnallisuuksia käyttöliittymään, mutta opinnäytetyössä tarvittiin vain datepicker-ominaisuutta. Datepicker näyttää tekstilaatikkoa painamalla kalenterin, josta voidaan valita haluttu päivämäärä ja asettaa tekstilaatikon sisältö valituksi päivämääräksi /-merkein eroteltuna /16/.



Kuva 2.6 Esimerkki datepickerin toiminnasta
(Lähde: http://jqueryui.com/datepicker/).

2.8.3 Highcharts

Highcharts on JavaScriptillä kirjoitettu kirjasto, jonka avulla web-sivulle on mahdollista piirtää erilaisia kuvaajia. Highcharts tukee lukuisia erilaisia kuvaajatyyppejä ja se toimii kaikilla moderneilla mobiili ja pöytäkoneselaimilla /7/. Toimiakseen Highcharts vaatii highcharts.js ja jQuery.js tiedostot linkitettynä HTML-tiedoston HEAD-elementtiin. jQuery.js-kirjaston vaihtoehtoina on myös mainittu MooTools, Prototype ja Highchartsin oma itsenäinen framework /8/.
Opinnäytetyössäni käytin Highcharts-kirjastoa kahdenlaisen tiedon esittämiseen kuvaajan muodossa. Käyttäjä voi joko hakea tietokannasta mittaustuloksia tietyllä aikavälillä tunnin tarkkuudella, tai sitten käyttäjä voi tarkastella käynnissä olevan clientin mittausdataa reaaliaikaisesesta kuvaajasta.

3 SUUNNITTELU

Suunnitteluvaihe oli opinnäytetyössä melko lyhyt. Projekti-insinööri Sami Rantamäellä oli jo lähes valmis visio, mitä asioita käyttöliittymällä tulisi pystyä hoitamaan. Hän laati dokumentin, jossa oli esitetty käyttöliittymän keskeiset vaatimukset ja näiden pohjalta lähdettiin suunnittelemaan käyttöliittymän toteutusta. Suunnitteluvaiheeseen kuului käyttöliittymän graafinen suunnittelu ylimalkaisesti paperille ja sopivien menetelmien valitseminen. Ohjelmistotuotannon prosessimalleista spiraalimallia sovellettiin opinnäytetyössä. Spiraalimallissa tuotteesta tehdään aina pieni osa kerrallaan valmiiksi. Eri vaiheet jokaiselle osalle ovat vaatimusanalyysi, suunnittelu, koodaus ja testaus. Näitä vaiheita kutsutaan tehtäväalueiksi. Vaatimusmäärittelyssä selvitetään mitä asiakas tuotteelta haluaa. Suunnitteluvaiheessa määritellään miten asiakkaan vaatimukset toteutetaan. Riskianalyysissä kartoitetaan mahdolliset vaiheeseen sisältyvät riskit. Toteutus- ja testausvaiheessa tehdään edellisten määrittelyjen perusteella yksi tuotteen osa valmiiksi ja toistetaan seuraava tuotteen osa samalla kaavalla /19/.

Opinnäytetyössä poikkettiin perinteisestä spiraalimallin kaavasta sen verran, että vaatimusmäärittely tehtiin koko käyttöliittymälle jo ennen projektin aloittamista, edellä mainitun dokumentin muodossa. Jokaisessa eri toiminnallisuuden ohjelmoinnissa aloitettiin aina suunnitteluvaiheesta, josta edettiin toteutukseen ja testaukseen. Riskianalyysia ei varsinaisesti omana vaiheenaan ollut lainkaan, vaan sen sijaan eri toteutustapojen mahdollisia riskejä mietittiin toistuvasti toteutuksen edetessä.

4 TOTEUTUS

Työn toteutukseen käytettiin HTML-, CSS-, JavaScript-, PHP- ja MySQL-ohjelmointikieliä. HTML- ja CSS-kielillä toteutettiin käyttöliittymän visuaalinen ilme. JavaScriptiä käytettiin pääasiallisesti jQuery:n muodossa sekä AJAX-kutsuissa tilanteissa joissa sivun uudelleenlataaminen ei ole optimaalisin vaihtoehto. Lisäksi JavaScriptiä käytettiin kuvaajien piirtämiseen highcharts.js-kirjaston avulla sekä joidenkin yksittäisten ominaisuuksien toteuttamiseen.. PHP-ohjelmointikielellä toteutettiin kaikki liikenne varsinaisen DEWIL-ympäristön ja käyttöliittymän välillä. Kerätty data tallennettiin MySQL-tietokantoihin.

4.1 Ympäristön asentaminen

Koululta saatiin käyttöön fit-PC2 niminen pienikokoinen PC-kone, jolle ensimmäiksi asennettiin Linux Mint Rebecca-käyttöjärjestelmä. Seuraavaksi koneelle asennettiin Java-kehitysympäristö, sekä MySQL palvelin, jotta näiden jälkeen asennettava DEWIL-ympäristö toimisi. MySQL saadaan asennettua Linux Mintin omasta pakettivarastosta kirjoittamalla komentoriville komento:

```
sudo apt-get install mysql-server
```

Lisäksi tarvittiin yksi DEWIL:n käyttöön tuleva tietokanta, jonne kaikki sen keräämä data säilötään. Uuden tietokannan luomiseksi tarvitsee kirjautua ensin MySQL-palvelimelle komentoriviltä komennolla:

```
mysql –u root –p
```

Tämän jälkeen salasanan kirjoittamalla pääsee sisälle MySQL-palvelimeen. MySQL-palvelimen sisällä uuden tietokannan saa luotua käskyllä:

```
CREATE DATABASE tietokannan_nimi;
```

Java-kehitysympäristönä käytettiin OpenJDK-1.7.0-ympäristöä, jonka pystyi myös asentamaan helposti Linux Mintin omasta pakettivarastosta komennolla:

```
sudo apt-get install openjdk-7-jdk
```

DEWIL:n asentaminen tapahtui kopioimalla aiemmin toteutettu DEWIL- hakemisto Linux Mint-käyttäjän …/Documents/-hakemistoon. DEWIL-palvelin oli jo valmiiksi toimintakunnossa opinnäytetyötä varten ja siihen ei tarvinnut tehdä muutoksia. Myös

testauskäyttöön tuleva 1-Wire client oli jo suurilta osin valmis. 1-Wire clientiin tarvitsi tehdä vain pieni lisäys virheentarkistukseen, jolloin mittausta ei käynnistetä jos antureiden lukumäärä konfiguraatiossa ei täsmää kytkettyihin antureihin.

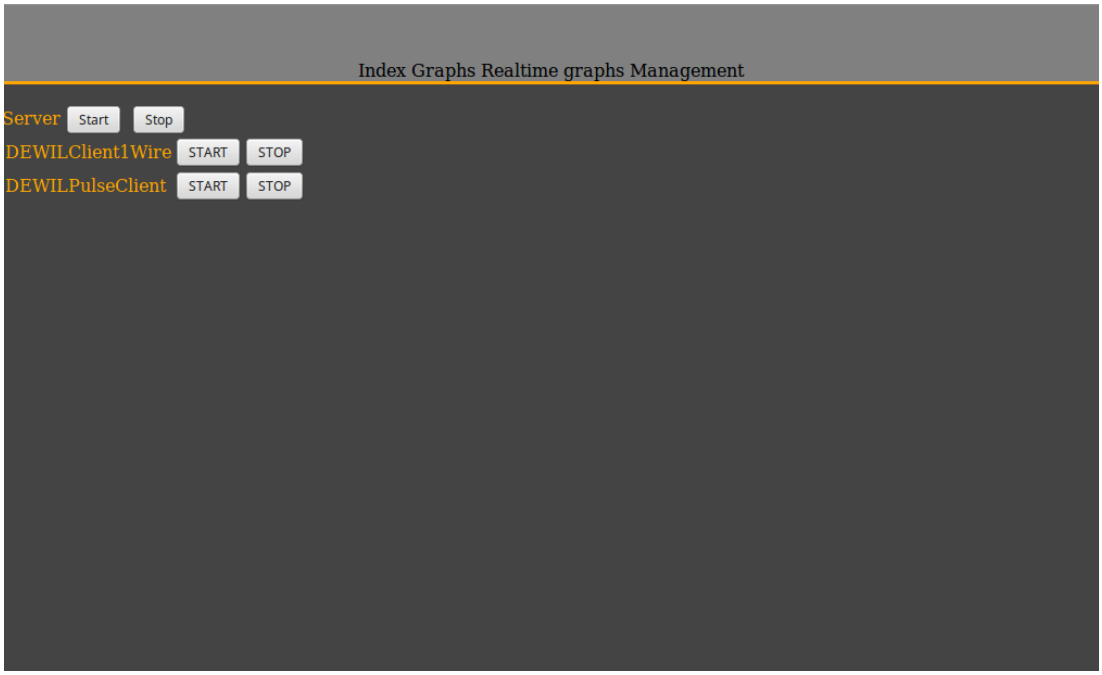
Käyttöliittymän toteutus vaati Apache2 web-palvelimen, sekä PHP-ympäristön asentamisen. Kummatkin olivat myöskin asennettavissa Linux Mintin omasta pakettivarastosta. Apache2 web-palvelin saatiin asennettua komennolla:

```
sudo apt-get install apache2
```

PHP-ympäristö taas saatiin asennettua komennoilla:

```
sudo apt-get install php5
sudo apt-get install libapache2-mod-php5
```

4.2 Clientien ja serverin käynnistäminen, sekä pysäyttäminen



Kuva 4.1. Käyttöliittymän pääsivu

Kuvassa 4.1 on esitetty käyttöliittymän pääsivu, joka on samalla tarkoitettu myös serverin ja eri clientien käynnistämiseen sekä pysäyttämiseen. Lista on generoitu suoraan tietokannasta saaduilla nimillä. Start-nappeja painamalla suoritetaan PHP-skripti, joka käynnistää GNU Screen-ohjelmassa shell-skriptin, joka käynnistää

palvelimen/clientin. GNU Screen on linux-apuohjelma, joka mahdollistaa usean virtuaalisen konsolin suorittamisen yhtäaikaisesti taustalla /10/. Shell-skriptillä tarkoitetaan tiedostoa, joka sisältää listan komentorivillä suoritettavia komentoja, jotka ajetaan automaattisesti listan mukaisessa järjestyksessä.

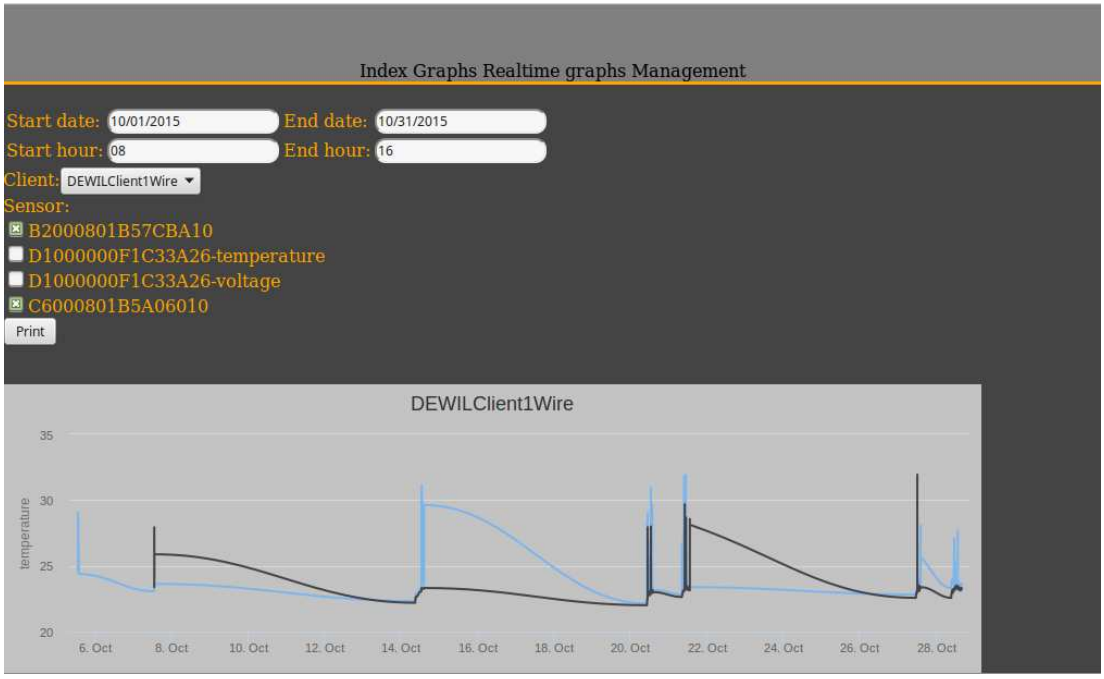
Stop-nappia painamalla käynnistetty GNU Screen suljetaan.

```
if(isset($_POST['startbtn'])){
    if($_SESSION['server_on'] == 1)
    {
        $startedclient = $_POST['idstart'];
        shell_exec('sudo screen -dmS ' . $startedclient . '_screen  /home/' . $get_current_user() . '/Documents/DEWIL/apps/' . $startedclient . '/' . $startedclient . '.sh');
    }
    else{
        echo "SERVER IS NOT ON!";
    }
}
if(isset($_POST['stopbtn'])){
    if($_SESSION['server_on'] == 1)
    {
        $stoppedclient = $_POST['idstop'];
        shell_exec('sudo screen -X -S ' . $stoppedclient . '_screen quit');
    }
    else{
        echo "SERVER IS NOT ON!";
    }
}
```

Kuva 4.2. Clientin käynnistys ja pysäytys

Kuvasta 4.2 nähdään ohjelmakoodi, jolla clientin käynnistäminen ja pysäyttäminen ollaan toteutettu. Aluksi tarkastetaan onko start- tai stop-nappia painettu ja tämän jälkeen varmistetaan onko serveri käynnissä. Jos serveriä ei ole vielä käynnistetty tulostetaan virheilmoitus. Jos serveri on käynnissä, luodaan clientin niminen GNU Screen tai lopetetaan jo käynnissä oleva GNU Screen.

4.3 Kuvaajan piirtäminen aikavälillä



Kuva 4.3. Kuvaajan piirtäminen halutulla aikavälillä.

Kuvaajan piirtämiseen aikavälillä on käytetty tavallista HTML-lomaketta, johon syötetään tekstikenttiin halutut päivämäärät ja kellonajat, sekä alasvetovalikosta valitaan haluttu client ja rastitetaan valintalaatikoista halutut anturit. Päivämäärän valinta on toteutettu jQueryUI-kirjastolla, jolloin tekstikenttää painamalla aukeaa kalenteri, josta päivämäärä voidaan valita.

Alasvetovalikon clientit tulostetaan suoraan tietokannasta ja valintalaatikot on toteutettu AJAX-tekniikalla niin, että kun clientiä vaihdetaan niin valintalaatikoiden sisältö muuttuu valitun clientin antureiksi. Alasvetovalikossa näkyvien antureiden nimet tulostetaan taulukossa 2.1 esitetyn “clients” taulun description-sarakkeesta koska samannimisellä anturilla voidaan 1-wiren tapauksessa mitata eri mittaustyyppiä olevia mittauksia ja näin ollen pelkästä anturin nimestä ei voida mittaustyyppiä selvittää.

```
<script>
$(document).ready(function(){
    getData();
});
function getData(){
    var xhr;
    var selection = $("#cl").val();
    var resultValue;
    xhr = new XMLHttpRequest();

    xhr.onreadystatechange = function(){
        if(xhr.readyState == 4 && xhr.status == 200){
            resultValue = xhr.responseText;
            console.log(resultValue);
            document.getElementById("cbox").innerHTML = resultValue;
        }
    }
    xhr.open("POST", 'getDescriptions.php', true);
    xhr.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
    var stringToSend = "cl="+selection;
    xhr.send(stringToSend);

}
</script>
```

Kuva 4.4. AJAX-kutsu, jolla haetaan antureiden nimet alasvetovalikkoon sen perusteella mikä clienteistä on valittuna

Kuvaajan piirtäminen toteutettiin highcharts.js ja jQuery.js JavaScript-kirjastoja käyttämällä. Sekaannusten välttämiseksi kuvaajaan on mahdollista piirtää samaan aikaan vain samaa mittaustyyppiä olevien anturien dataa. Mittaustyyppien määrä pystytään tarkistamaan ehtolauseella:

```
if(count(array_unique($measurement_types)) == 1)
```

jolloin ehtolauseen sisältö eli kuvaajan piirtäminen suoritetaan vain jos mittaustyyppejä on yksi. Muuten tulostetaan else-lausekkeen sisältä virheilmoitus.

Myös antureiden valitsematta jättäminen tulostaa virheilmoituksen. Tämä toteutettiin ehtolauseella:

```
if(isset($_POST['sensors']))
```

jolla tarkistetaan että ainakin yksi valintalaatikoista on valittuna.

```
    echo "
        <script type = \"text/javascript\">

        $(function() {



        $('#placeholder').highcharts({
            chart: {
                type: 'spline',
                backgroundColor: '#C3C3C3'
            },
            xAxis: {
                type: 'datetime'
            },
            yAxis: {
                title: {
                    text: '" . getMeas($sensor_names[0]) . "'
                }
            },
            title: {
                text: '" . $client . "'
            },
            " . $generateSeries . "
        });
    });

</script></br>
    ";
```

Kuva 4.5 PHP-koodi kuvaajan piirtämiseen aikavälillä

Kuvaajan piirtäminen on toteutettu tulostamalla PHP-koodin sisällä JavaScript-koodia. Piirtäminen on toteutettu tällä tavalla, koska näin pystytään laittamaan sekaan myös PHP-muuttujien arvoja.

Kuvaajan x-akselilla on kuvattuna käyttäjän syöttämä aikaväli ja y-akselilla antureiden mittaamat arvot. Highcharts käyttää näiden arvojen esittämiseen JSON- eli JavaScript Object Notation-muotoista dataa. Tämä JSON-data on ennen piirtämistä tallennettu *$generateSeries*-nimiseen PHP-muuttujaan. Kuvaajan otsikkona käytetään valitun clientin nimeä ja y-akselin otsikkona on *getMeas()*-funktiota käyttämällä tietokannasta haettu valittu mittaustyyppi.

```
if(count(array_unique($measurement_types)) == 1)
{


if(count($sensors) == 1){
    $generateSeries = "series: [{ name: '" . $sensor_names[0] . "', data: " . $json[0] . "}]";
}
else{

    $seriesArray = array();


    $seriesArray[0] = "series: [{ name: '" . $sensor_names[0] . "', data: " . $json[0] . "}";

    for($j = 1; $j<count($sensors); $j++){
        $seriesArray[$j] = ", { name: '" . $sensor_names[$j] . "', data: " . $json[$j] . "}";


    }
    $seriesArray[count($sensors)] = "]";
    $generateSeries = implode("", $seriesArray);




}
```

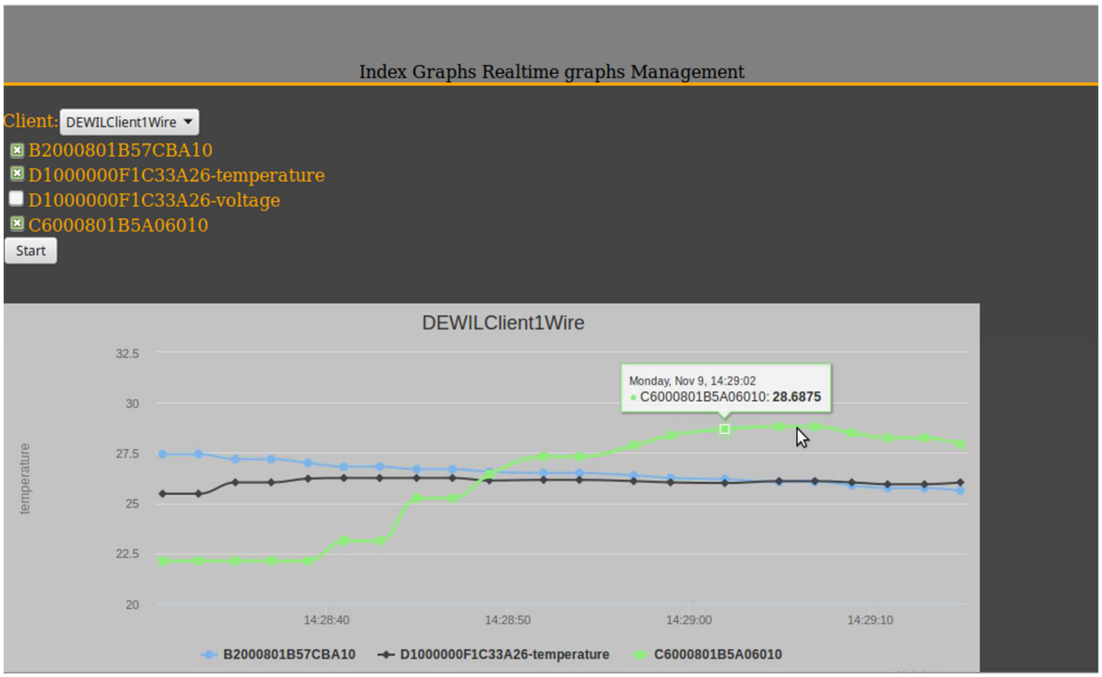
Kuva 4.6 *$generateSeries*-muuttujan määrittely

Kuvasta 4.6 nähdään kuinka kuvaajan piirtämisen kannalta keskeinen muuttuja *$generateSeries* on määritelty. Jos vain yksi anturi on valittu, tallennetaan muuttujaan vain sen anturin nimi sekä anturin mittausdata JSON-muodossa.

Jos taas antureita on valittuna useampi, tallennetaan ensin jokaisen anturin edellämainitut datat taulukkoon, ja tästä saadaan kuvaajan piirtoon kelpaavaa dataa käyttämällä PHP-funktiota *implode()*. *Implode()*-funktio hajoittaa taulukon rakenteen niin, että taulukon jokaisen alkion sisältö laitetaan peräkkäin yhteen tekstijonoon. Alkioiden erottamiseen käytettävä merkki annetaan funktion ensimmäisenä parametrinä /18/.

Kuvaaja itsessään piirretään HTML-sivulla määritetyn div-elementin sisälle.

4.4 Reaaliaikainen kuvaaja



Kuva 4.7. Kuvaajan piirtäminen reaaliajassa

Kuvassa 4.7 nähdään kuvankaappaus valittujen antureiden datan piirtämisestä reaaliajassa. Reaaliaikaisen kuvaajan piirtämisessä ehdot ovat samat aikavälilläkin piirtämisessä mutta lisäksi varmistetaan, että piirrettävien antureiden käyttämä client on käynnissä.

Itse reaaliaikaisen kuvaajan tekninen toteutus poikkeaa jonkin verran edellisen kappaleen kuvaajasta, koska piirtämiseen tarvitaan AJAX-kutsuja.

```
function requestData() {
        $.ajax({
            url: 'livedata.php',
            success: function(point) {

                if(count > 0 && oneMeasType){

                var series = chart.series[0],
                    shift = series.data.length > 20;


                var values = eval(point);
                for(var i = 0; i<count; i++){
                chart.series[i].addPoint([values[0], values[i+1]], true, shift);

                }



                setTimeout(requestData, 1000);

               }

            },
            cache: false
        });
    }


    $(document).ready(function() {

        chart = new Highcharts.Chart({
            chart: {
                renderTo: 'placeholder',
                defaultSeriesType: 'spline',
                backgroundColor: '#C3C3C3',
                events: {
                    load: requestData
                }
            },
            title: {
                text:  getClientName()
            },
            xAxis: {
                type: 'datetime',
                tickPixelInterval: 150,
                maxZoom: 20 * 1000
            },
            yAxis: {
                minPadding: 0.2,
                maxPadding: 0.2,
                title: {
                    text: getMeasTypeStorage(),
                    margin: 80
                }
            },
            series: getSeries()

        });

    });
```

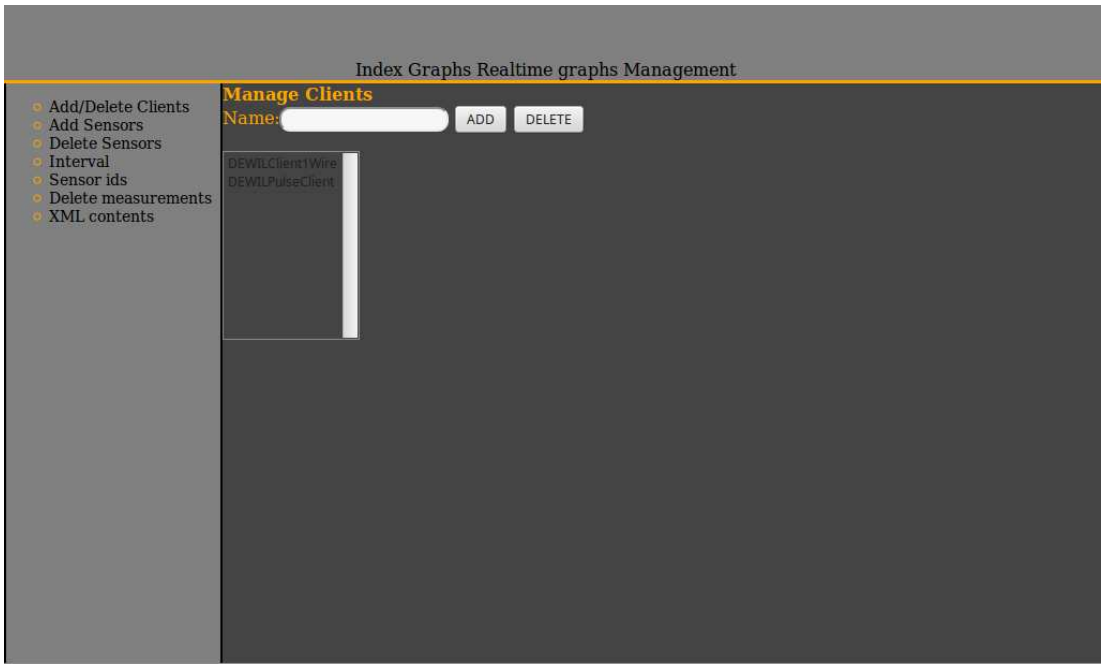
Kuva 4.8. JavaScript-koodi reaaliaikaisen kuvaajan piirtämiseen.

Reaaliaikaisen kuvaajan piirtämisessä tarvitsee hakea AJAX-kutsulla tietyin aikavälein dataa tietokannasta. Nämä määrittelyt tehdään kuvassa 4.7 näkyvässä funktiossa *requestData()*. Funktion url-kohdassa on mainittu datan hakemisen käsittelevä PHP-tiedosto. For-silmukan sisällä alustetaan valittu määrä käyriä ja annetaan niille kannasta luetut arvot. *setTimeout()*-funktion toisessa parametrissa annetaan haluttu kuvaajan päivitysaikaväli, tässä tapauksessa 1000ms.
Itse kuvaajan piirtäminen tapahtuu samaan tapaan kuin aiemminkin, mutta tein erilaisia JavaScript-funktioita kuvaajan eri tietojen näyttämiseen.

4.5 Hallintapaneeli

Käyttöliittymän hallintapaneelista pystytään hoitamaan kaikki asiat, jotka vaativat tiedostojen tai tietokantojen sisällön käsittelemistä. Käsittelyllä tarkoitetaan tietojen lisäämistä, tietojen poistamista, tietojen muokkaamista tai tietojen tulostamista tiedostoista tai tietokannoista.

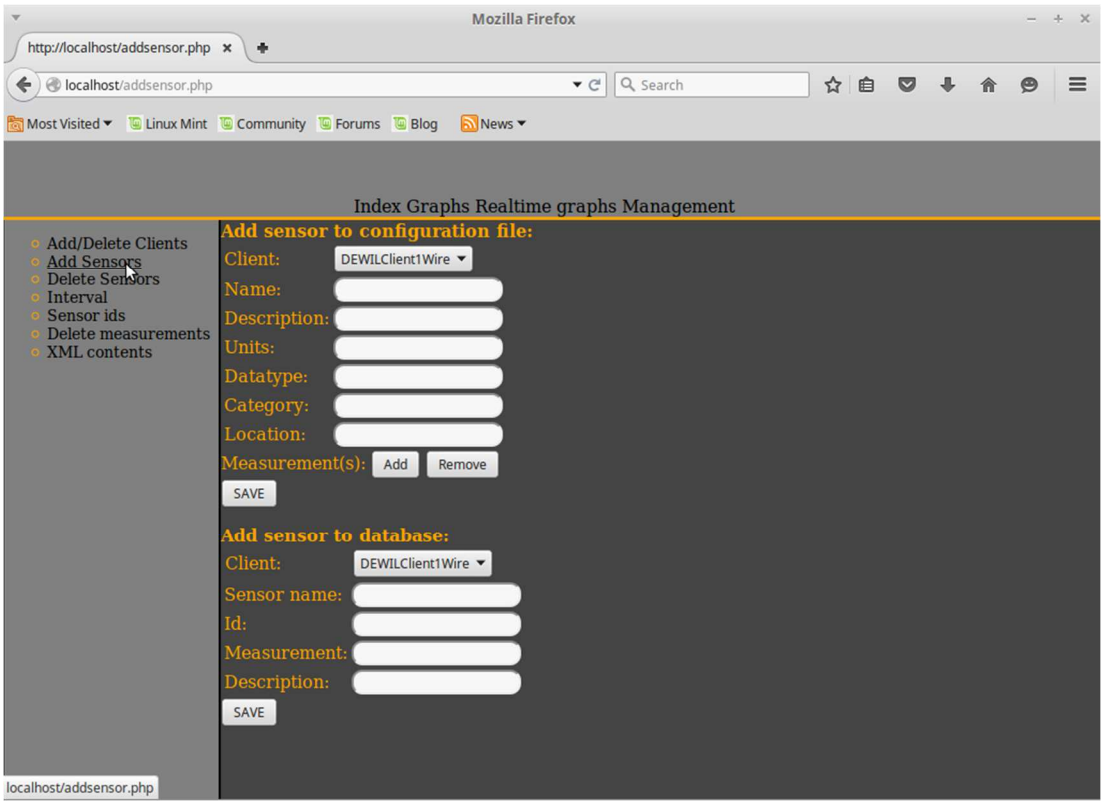
4.5.1 Clientin lisääminen



Kuva 4.9. Uuden clientin lisääminen/poistaminen

Kuvassa 4.9 on otettu kuva hallintapaneelin sivusta, jolla on mahdollista lisätä “Clients”-tauluun uusia clientejä tai poistaa vanhoja. Kyseessä on yhden tekstikentän ja kahden napin sisältävä HTML-lomake. ADD-nappia painamalla kutsutaan funktiota *addClient()*, joka hoitaa clientin tallentamisen kantaan. DELETE-nappia painamalla kutsutaan funktiota *deleteClient()*, joka hoitaa clientin poistamisen tietokannasta. Lomakkeen alapuolella on listattu tämänhetkiset clientit tietokannasta. Samannimisen clientin lisäämisyritys tai ei-olemassaolevan clientin poistoyritys tulostaa virheilmoituksen ja mitään ei tehdä.

4.5.2 Antureiden lisääminen/poistaminen



Kuva 4.10. Uuden anturin lisääminen.

Uuden anturin lisäämisessä ollaan käytetty kahta erillistä HTML-lomaketta, joilla toisella generoidaan XML-muotoiseen konfiguraatiotiedostoon uudet XML-elementit lomakkeeseen syötetyillä tiedoilla. Toisella lomakkeella lisätään uusi anturi tietokannan tauluun “clients”.

Konfiguraatiotiedostoon tallentavassa lomakkeessa kaikkiin kenttiin tulee kirjoittaa jotakin ja tämä on varmistettu kirjoittamalla tekstilaatikkoelementin määrittelyyn sana *required*. Antureilla voi olla mahdollista mitata usean tyyppistä dataa ja painamalla ADD-nappia luodaan aina yksi uusi tekstikenttä, johon voidaan syöttää mittaustyyppi.

REMOVE-nappia painamalla poistetaan kaikki ADD-napilla luodut tekstikentät.

```
<script>
$(document).ready(function(){
    var iCnt = 0;
    var boxes = $(document.createElement('div')).css({
        padding: '5px', margin: '20px', width: '170px', border: '1px dashed'
    });
    $('#btAdd').click(function(){

    if(iCnt <= 19){
        iCnt = iCnt + 1;

        $(boxes).append(iCnt + ': <input type =text name=tb[] class="textboxes" id=tb' + iCnt + '' +'value=TextElement' + iCnt +
            '" required />');

        $('#meas').after(boxes);


    }
    });

    $('#btRemove').click(function(){

            $(boxes).empty();
            $(boxes).remove();
            $('#btAdd').removeAttr('disabled');
            $('#btAdd').attr('class', 'bt');
            iCnt = 0;

    });
});

</script>
```

Kuva 4.11. JavaScript-koodi, jolla mahdollista luoda/poistaa tekstikenttiä.

SAVE-nappia painamalla, pystytään tallentamaan syötetyt tiedot XML-tiedostoon ja seuraavana on kuvattu kuinka tämä teknisesti toteutuu.

```php
if(isset($_POST['save'])){
            $client = $_POST['clients'];
            $sensorname = $_POST['name'];
            $measArray = $_POST['tb'];

            $xml = new DOMDocument();


            $xml->load('/home/' . $get_current_user() . '/Documents/DEWIL/apps/' . $client. '/dewil.xml');
            $searchNode = $xml->getElementsByTagName("argArray");

            foreach($searchNode as $searchNode){
            $value = $searchNode->getAttribute('n');

                    if($value == 'Channels.information'){

                        $elementCi = $xml->createElement('argArray', '');
                        $elementCi->setAttribute('n', $_POST['name']);
                        $searchNode->appendChild($elementCi);

                        $name = $xml->createElement('arg', $_POST['name']);
                        $name->setAttribute('n', 'name');
                        $elementCi->appendChild($name);

                        $desc = $xml->createElement('arg', $_POST['desc']);
                        $desc->setAttribute('n', 'description');
                        $elementCi->appendChild($desc);


                        $units = $xml->createElement('arg', $_POST['units']);
                        $units->setAttribute('n', 'units');
                        $elementCi->appendChild($units);

                        $data = $xml->createElement('arg', $_POST['data']);
                        $data->setAttribute('n', 'dataType');
                        $elementCi->appendChild($data);

                        $cat = $xml->createElement('arg', $_POST['cat']);
                        $cat->setAttribute('n', 'category');
                        $elementCi->appendChild($cat);

                        $loc = $xml->createElement('arg', $_POST['loc']);
                        $loc->setAttribute('n', 'locationDesc');
                        $elementCi->appendChild($loc);


                    }
```

Kuva 4.12. PHP-koodi, jolla pystytään lisäämään anturin tiedot XML-tiedostoon

```
                    if($value == 'Channels.used'){

                        $elementCu = $xml->createElement('argArray', '');
                        $elementCu->setAttribute('n', $_POST['name']);
                        $searchNode->appendChild($elementCu);

                        for($i = 0; $i<count($measArray); $i++){
                        $meas = $xml->createElement('arg', $measArray[$i]);
                        $meas->setAttribute('n', ($i+1));
                        $elementCu->appendChild($meas);
    }
                    }

            }
                    $searchChans = $xml->getElementsByTagName("arg");

                        foreach($searchChans as $searchChans){

                            $value = $searchChans->getAttribute('n');

                            if($value == 'numberOfChannels'){
                                $currentChannels = $searchChans->nodeValue;
                                $newChannels = $currentChannels+count($measArray);
                                $searchChans->nodeValue = $newChannels;

                            }
                        }

    $xml->save('/home/' . $get_current_user() . '/Documents/DEWIL/apps/' . $client. '/dewil.xml');
    sensorStatusOn($sensorname);
}
```

Kuva 4.13 PHP-koodi, jolla pystytään asettamaan mittausten määrä sekä mittausten tyypit XML-tiedostoon. Jatkoa kuvan 4.8. koodille

Kuvissa 4.12 ja 4.13 on esitetty PHP-koodi, jolla saadaan lomakkeeseen syötetyt tiedot kirjoitettua konfiguraation vaatimien XML-elementtien sisälle.

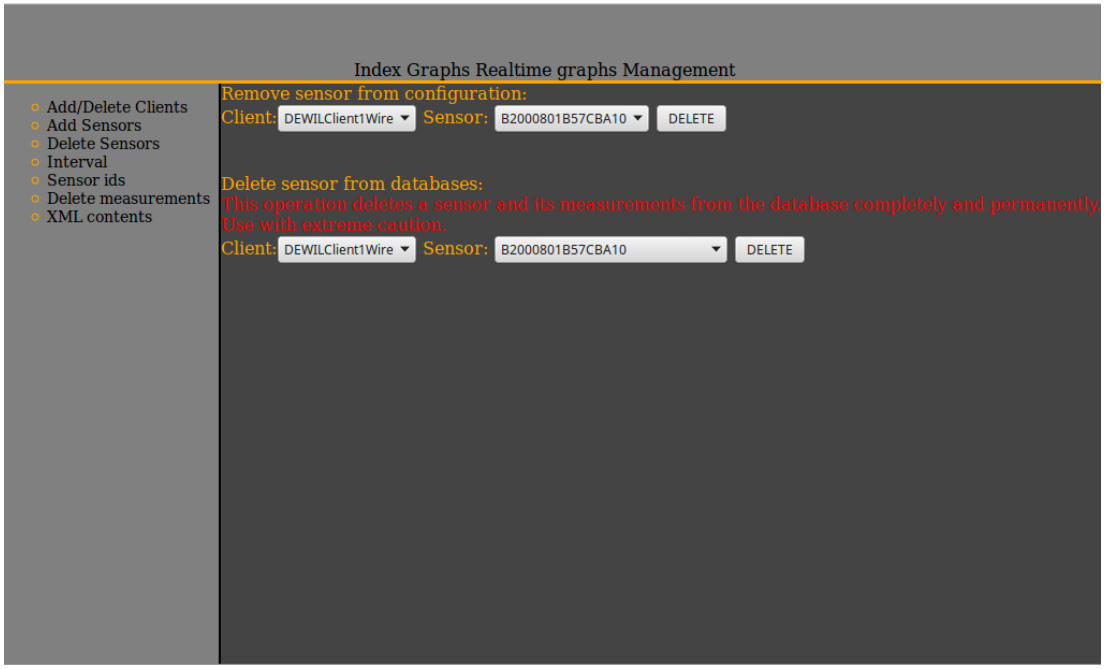
Aluksi luodaan uusi DOMDocument-objekti, jonka sisälle haluttu konfiguraatiotiedosto ladataan. Tämän jälkeen tiedostosta etsitään argArray-elementti, jonka n-attribuuttina on “Channels.information”. Tämän elementin sisälle puolestaan luodaan jokaiselle lomakkeen kohdalle oma arg-elementtinsä, jonka n-attribuuttina on elementin sisältöä kuvaava otsikko ja elementin sisälle tulee lomakkeeseen syötetty teksti.

Seuraavaksi etsitään argArray-elementti, jonka n-attribuuttina on “Channels.used”. Tämän elementin sisälle luodaan uusi argArray-elementti, jonka n-attribuuttina on anturin nimi. Edelleen viimeksi luodun argArray-elementin sisälle luodaan arg-elementtejä niin monta kuin anturilla on erilaisia mittaustyyppejä. N-attribuuttina on järjestysnumero ja elementin sisältönä mittaustyypin kuvaus.

Viimeisenä muutetaan “numberOfChannels”-elementin sisältö, joka kertoo montako eri kanavaa eli mittausta konfiguraatiossa on. Tämä onnistuu lisäämällä vain vanhaan arvoon uuden anturin mittaustyyppien lukumäärä. Lopuksi muutokset tallennetaan XML-konfiguraatiotiedostoon.

Antureiden konfiguraatioon lisäämisen jälkeen tarvitsee anturit omaava client käynnistää ja tarkistaa hallintapaneelin “Sensor ids”-sivulta uusien antureiden ID-numerot. Tämä menettely johtuu siitä syystä että lomake, jolla anturi tallennetaan tietokantaan kysyy anturin ID-numeroa, joka ei ole välttämättä tiedossa ennen kuin anturilla ollaan käynnistetty mittaus. Näin ollen se pitää poimia antureiden ID-numerot tulostavalta sivulta talteen.

Tämän jälkeen täytetään anturin tiedot tietokantaan lisäävään lomakkeeseen ja SAVE-nappia painamalla tallennetaan uusi anturi kantaan. Tässäkin tapauksessa on eliminoitu mahdollisuus jättää kenttiä tyhjiksi tekstilaatikon elementissä käytetyllä required-määrittelyllä.



Kuva 4.14. Anturin poistaminen.

Kuvassa 4.14 on kuvattuna antureiden poistamiseen tarkoitettu sivu. Sivulla on kaksi lomaketta, joista kumpikin koostuvat pelkistä alasvetovalikoista ja napista. Toisella lomakkeella voidaan poistaa anturi XML-konfiguraatiotiedostosta ja toisella lomakkeella voidaan poistaa anturin kaikki olemassa olevat tiedot kaikista tietokannoista. Kummassakin tapauksessa poistaminen varmistetaan ensin JavaScriptillä toteutetulla ponnahdusikkunalla vahinkojen välttämiseksi.

```
if(isset($_POST['delete'])){
    $client = $_POST['clients'];
    $sensor = $_POST['sensors'];

    SensorStatusOff($sensor);

    $xml = new DOMDocument();

    $xml->load('/home/' . $get_current_user() . '/Documents/DEWIL/apps/' . $client. '/dewil.xml

    $searchChans = $xml->getElementsByTagName("argArray");

    foreach($searchChans as $searchChans){
        $value = $searchChans->getAttribute('n');


                if($value == 'Channels.used'){

                    $searchArgs = $searchChans->getElementsByTagName('argArray');

                    foreach($searchArgs as $searchArgs){

                        $argValue = $searchArgs->getAttribute('n');

                        if($argValue == $sensor){
                            $searchMeas = $searchArgs->getElementsByTagName('arg');
                            $measurements = $searchMeas->length;

                        }
                    }
                }
    }

    $searchNode = $xml->getElementsByTagName("argArray");

        foreach($searchNode as $searchNode){
            $value = $searchNode->getAttribute('n');

                if($value == $sensor){
                    $searchNode->parentNode->removeChild($searchNode);
                }
        }

        $searchNoc = $xml->getElementsByTagName("arg");

        foreach($searchNoc as $searchNoc){

                $valueNoc = $searchNoc->getAttribute('n');

                if($valueNoc == 'numberOfChannels'){
                    $currentChannels = $searchNoc->nodeValue;
                    $newChannels = $currentChannels-$measurements;
                    $searchNoc->nodeValue = $newChannels;
                 }
        }

$xml->save('/home/' . $get_current_user() . '/Documents/DEWIL/apps/' . $client. '/dewil.xml');
}
```

4.15. Halutun anturin poistaminen XML-konfiguraatiotiedostosta

Kuvassa 4.15. nähdään anturin poistamisen tekninen toteutus. Aluksi avataan XML-tiedosto DOMDocument-objektiin, kuten anturin lisäämisessäkin tehtiin. Poistaminen tapahtuu etsimällä argArray-elementit, joiden n-attribuuttina on anturin nimi ja sen jälkeen poistetaan tämä elementti ja kaikki sen lapsielementit. Lopuksi vähennetään “numberOfChannels”-elementin sisältöä niin monella yksiköllä kuin mittaustyyppejä anturin mukana poistui ja XML-tiedosto tallennetaan.

```
if(isset($_POST['deleteDB'])){
    $sensor_id = getIdAndDelete($_POST['sensorsDB']);
    deleteFromDEWIL($sensor_id);
    echo $sensor_id . " DELETED";
}
```

4.16. Halutun anturin tietojen poistaminen pysyvästi tietokannoista.

Kuvassa 4.16. kuvattu koodi, jolla pystytään poistamaan haluttu anturi, sekä “clients”-taulusta että DEWIL:n käyttämistä tietokannoista. Tätä ominaisuutta tulee käyttää äärimmäisellä varovaisuudella, koska kaikki anturia koskeva tieto poistetaan.

4.5.3 Mittausvälin säätäminen

Kuva 4.17. Mittausaikavälin säätäminen.

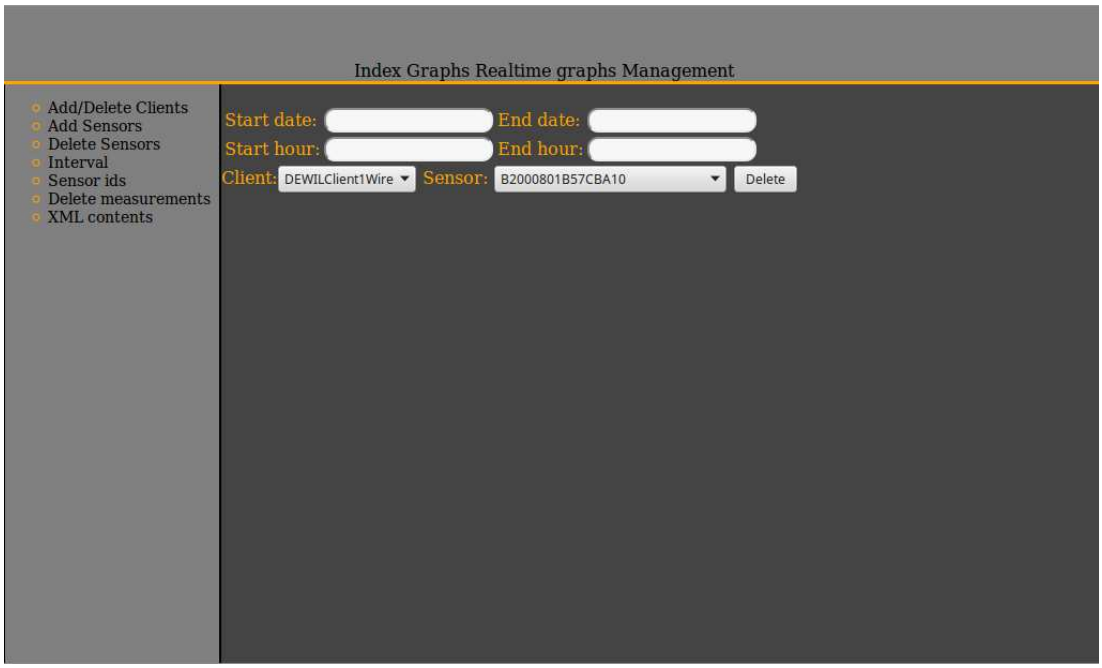
Mittausvälillä tarkoitetaan aikaa, kuinka usein DEWIL hakee mitattua dataa kyseiseltä clientillä. Yksikkönä on käytetty sekuntia. Tälle on konfiguraatiotiedostossa oma elementtinsä, jonka sisältöä pystytään muokkaamaan kuvassa 4.17 esitetyllä HTML-lomakkeella. Alasvetovalikosta valitaan haluttu client ja asetetaan tekstilaatikkoon uusi aika.

4.5.4 Antureiden ID-numeron tulostaminen.

Kuva 4.18. Lista antureista ja niiden ID-numeroista.

Kuvassa 4.18 olevalla sivulla tulostetaan DEWIL:n käyttämästä tietokannasta kaikki anturit ja niiden ID-numerot. Koska anturin ID-numeroita ei voida varmuudella tietää ennen clientin käynnistämistä ja näin ollen anturia ei pystytä heti lisäämään “clients”-tauluun niin tätä sivua tarvitaan oikean ID-numeron löytämiseen.

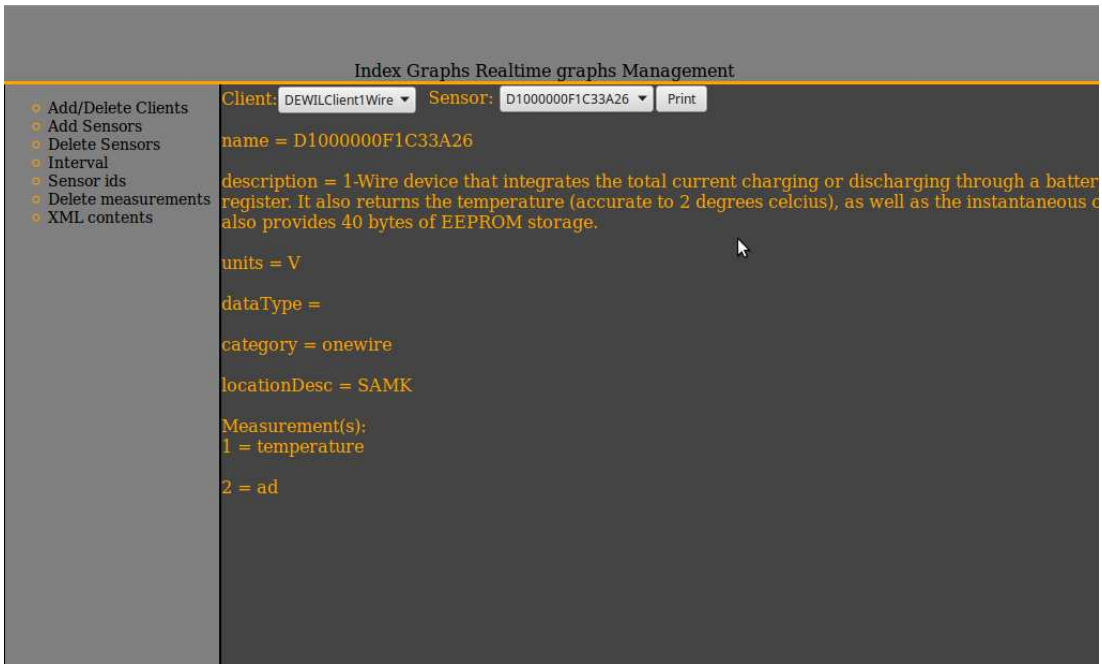
4.5.5 Antureiden datan poistaminen tietokannasta valitulta aikaväliltä



Kuva 4.19 Datan poistaminen valitulta aikaväliltä

Kuvassa 4.19 on esitetty sivu, jolla pystyy poistamaan tietokannasta antureilla mitattua dataa valittujen ajankohtien väliltä. Päivämäärien lukemiseen käytettiin samaa jQueryUI-tekniikkaa, kuin kuvaajan piirtämiseen aikavälillä. Ajankohtien lisäksi alasvetolaatikoista valitaan haluttu client sekä haluttu anturi. Delete-nappia painamalla kutsutaan ensin funktiota *getID()*, jonka avulla saadaan selvitettyä anturin ID-numero ja tämän jälkeen saadun ID-numeron perusteella suoritetaan MySQL-tietokantakysely, joka poistaa mittausdatan valittujen ajankohtien väliltä.

4.5.6 XML-tiedostojen sisällön tulostus



Kuva 4.20. Anturin tietojen tulostaminen XML-tiedostosta.

Koska XML-konfiguraatiotiedostoihin loppukäyttäjän ei tule päästä käsiksi suoraan, vaaditaan myös jokaisen anturin tietojen tulostamiselle XML-tiedostosta oma sivunsa, joka on esitetty kuvassa 4.19.

```
if(isset($_POST['print'])){
    $client  = $_POST['clients'];
    $sensor = $_POST['sensors'];
    $xml = new DOMDocument();
    $xml->load('/home/' . $get_current_user() . '/Documents/DEWIL/apps/' . $client. '/dewil.xml');

    $searchNode = $xml->getElementsByTagName("argArray");

    foreach($searchNode as $searchNode){
        $value = $searchNode->getAttribute('n');
        if($value == 'Channels.used'){
            echo "Measurement(s): <br>";
        }
        if($value == $sensor){
            $searchArgs = $searchNode->getElementsByTagName("arg");
            foreach($searchArgs as $searchArgs){
                $valueArg = $searchArgs->getAttribute('n');
                echo $valueArg . " = " . $searchArgs->nodeValue . "<br><br>";
            }
        }

    }
}
```

Kuva 4.21 PHP-koodi XML-tiedoston sisällön tulostamiseen.

XML-tiedoston tulostuksen tekninen toteutus on esitetty kuvassa 4.20. XML-tiedostosta haetaan anturin nimellä otsikoitu elementti ja tulostetaan sekä otsikon nimi että elementin sisältö eri riveille, erotettuna =-merkillä.

5 TESTAUS

Testausta suoritettiin jo suunnitteluvaiheessa päätetyn sovelletun spiraalimallin mukaisesti. Jokainen uusi toiminnallisuus testattiin heti toteutuksen jälkeen ja varmistettiin, että se toimii odotetusti. Vasta tämän jälkeen siirryttiin seuraavan toiminnallisuuden toteutukseen.

Anturien lisäämistä ja poistamista pystyttiin testaamaan keksityillä arvoilla, joiden avulla pystyttiin tarkastamaan tuleeko uusi anturi oikeaan kohtaan ja oikeisiin elementteihin konfiguraatiotiedostoon. Vastaavasti anturia poistaessa tarkistettiin, että vain halutun anturin kaikki ominaisuudet poistuivat konfiguraariosta, eikä mitään enempää tai vähempää. Vastaavalla tavalla pystyttiin testaamaan myös tietojen lisäämistä ja poistamista tietokannasta.

Kuvaajien piirtämistä pystyttiin testaamaan ajamalla DEWIL-ympäristöä ja 1-Wire clientiä suoraan Linux Mintin komentoriviltä, jolloin clientin mittaukset ovat koko ajan käynnissä ja ne tallennetaan DEWIL:n omaan tietokantaan. Kuvaajat piirrettiin tämän tietokantaan tallennetun datan perusteella.

Lopullinen testaus anturin poistamiselle ja lisäämiselle tehtiin niin, että poistettiin käyttöliittymästä käsin yksi lämpötilaa mittaava 1-Wire anturi. Poistaminen tehtiin hallintapaneelin kautta sekä konfiguraatiotiedostosta, että tietokannoista. Tästä seurasi, että anturi oli samankaltaisessa tilassa kuin sitä ei olisi koskaan aikaisemmin kytkettykään ympäristöön. Tämän jälkeen anturi lisättiin takaisin ympäristöön, aikaisemmin mainittujen vaiheiden mukaisesti ja tarkistettiin, että uudella anturilla pystytään tekemään käyttöliittymässä kaikki vaaditut operaatiot. Loput löytyneet virheet korjattiin tässä vaiheessa.

Kokonaan uuden clientin lisäämistä testattiin aikaisemmin tekemäni Arduino Uno-mikrokontrollerin pohjalle rakennetulla pulssien mittausclientillä. Client sisälsi valmiiksi kaikki anturit määriteltyinä konfiguraatiotiedostossa, joten testaus aloitettiin ensin lisäämällä uuden clientin tiedostot DEWIL:n clientien käyttämään kansioon. Tämän jälkeen lisättiin client ja sen anturit tietokantaan ja tarkistettiin, että kaikki

tallentui oikein. Seuraavaksi tarkistettiin, että client pystytään käynnistämään ja sammuttamaan, sekä piirtämään kuvaajat tietokantaan tallennetusta datasta. Tässä testausvaiheessa virheitä ei enää tullut, koska edellisessä testausvaiheessa suurin osa käyttöliittymän virheistä oli saatu jo korjattua.

6 YHTEENVETO

Valmis opinnäytetyö tuli täyttämään kaikki toiminnallisuudet, jotka oli listattu etukäteen tehdyssä vaatimusmäärittelyssä. Käyttöliittymän ohjelmakoodista tuli paikoittain pitempi kuin olin suunnitellut, joten koodia voidaan vielä muokata joistakin kohdin helpommin ylläpidettäväksi. Myös käyttöliittymän kehitystyö jatkuu muutenkin, koska DEWIL-ympäristöönkin tulee vielä opinnäytetyön valmistumisen jälkeen mahdollisia muutoksia ja nämä täytyy päivittää aina myös käyttöliittymään. Myös graafiselle käyttöliittymälle tehtiin alustavasti vain selkeä pohja, jonka päälle on mahdollista alkaa rakentamaan näyttävämpää kokonaisuutta.

Opinnäytetyön toteutuksessa en törmännyt suuriin ongelmiin vaan kaikki ongelmakohdat sain ratkaistua tutkimalla ongelmaa internetin avulla ja soveltamalla mahdollisia ratkaisuja opinnäytetyössäni. Erityisesti internetissä olevien erilaisten ohjelmointiin liittyvien keskustelupalstojen läpikäyminen oli auttavaista ja opettavaista.

LÄHTEET

1 Fit-PC www-sivusto, viitattu 10.11.2015 saatavilla:
http://www.fit-pc.com/web/products/fit-pc2/

2 Sami Rantamäki. 2014. Perehdytysmateriaali.

3 W3C HTML & CSS www-sivusto, viitattu 10.11.2015 saatavilla:
http://www.w3.org/standards/webdesign/htmlcss

4 Webplatform The Basics of HTML www-sivusto, viitattu 10.11.2015 saatavilla:
https://docs.webplatform.org/wiki/guides/the_basics_of_html

5 W3C XML Essentials www-sivusto, viitattu 10.11.2015 saatavilla:
http://www.w3.org/standards/xml/core

6 W3Schools JavaScript Where To www-sivusto, viitattu 10.11.2015, saatavilla:
http://www.w3schools.com/js/js_whereto.asp

7 Highcharts Product www-sivusto, viitattu 10.11.2015, saatavilla:
http://www.highcharts.com/products/highcharts

8 Highcharts Installation www-sivusto, viitattu 10.11.2015 saatavilla:
http://www.highcharts.com/docs/getting-started/installation

9 PHP.net What is PHP? www-sivusto, viitattu 15.11.2015 saatavilla:
http://php.net/manual/en/intro-whatis.php

10 GNU Screen www-sivusto, viitattu 15.11.2015 saatavilla:
https://www.gnu.org/software/screen/

11 What is Shell Scripting? www-sivusto, viitattu 15.11.2015 saatavilla:
http://www.freeos.com/guides/lsst/ch01sec09.html

12 PHP.net What can PHP do? www-sivusto, viitattu 17.11.2015 saatavilla:
http://php.net/manual/en/intro-whatcando.php

13 PHP.net manual $_GET www-sivusto, viitattu 17.11.2015 saatavilla:
http://php.net/manual/en/reserved.variables.get.php

14 jQuery.com jQuery API www-sivusto, viitattu 17.11.2015 saatavilla:
http://api.jquery.com/

15 jQuery.com How jQuery works? www-sivusto, viitattu 17.11.2015 saatavilla:
http://learn.jquery.com/about-jquery/how-jquery-works/

16 jQueryUI.com Support Center www-sivusto, viitattu 17.11.2015 saatavilla:
http://jqueryui.com/support/

17 A typical Ajax request www-sivusto, viitattu 17.11.2015 saatavilla:
http://www.webstepbook.com/supplements/slides/ch10-ajax_xml.shtml

18 PHP.net implode() www-sivusto, viitattu 17.11.2015 saatavilla:

http://php.net/manual/en/function.implode.php

19 Ohjelmistotuotannon prosessimallit www-sivusto, viitattu 17.11.2015 saatavilla: http://www.cs.helsinki.fi/u/taina/ohtu/s-2000/luennot/prosessi/kaikki.html

20 DigitalOcean A Basic MySQL tutorial www-sivusto, viitattu 18.11.2015 saatavilla:

https://www.digitalocean.com/community/tutorials/a-basic-mysql-tutorial