TAMPEREEN
AMMATTIKORKEAKOULU

IOT -MITTAUSJÄRJESTELMÄ JA DATAN VISUALISOINTI

Corelase

Roni Rantaeskola

Opinnäytetyö
Joulukuu 2017
Tieto- ja viestintätekniikka
Ohjelmistotekniikka

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tieto- ja viestintätekniikka
Ohjelmistotekniikka

RANTAESKOLA RONI:
IoT -mittausjärjestelmä ja datan visualisointi

Opinnäytetyö 29 sivua, joista liitteitä 0 sivua
Joulukuu 2017

Tulevaisuudessa kaikki tulee olemaan yhteydessä Internettiin muodossa tai toisessa. Internet of Things (IoT) eli asioiden Internet on mahdollistanut mm. teollisuuden automaation, IoT on nähnyt eksponentiaalista kasvua viimeisen muutaman vuoden aikana, IoT -laitteita arvioidaan olevan jopa 32 miljardia vuoteen 2020 mennessä.

Tämän opinnäytetyön aiheena on IoT -mittausjärjestelmä sekä datan visualisoiminen verkko portaalissa. Opinnäytetyö käsittelee IoT -järjestelmän mesh-verkkoteknologiaa ja rakennetta teoreettisella tasolla, datan visualisointi menetelmiä, verkko portaalin kehitystyötä, ja toteutuksen teknillisten päätösten tarkentamista.

Asiasanat: iot

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Programme in ICT Engineering
Software Engineering

RANTAESKOLA RONI:
IoT -measurement platform and data visualization web portal

Bachelor's thesis 29 pages, appendices 0 pages
December 2017

Eventually, in the not so far future, everything will be connected to the internet in one form or another. Internet of Things (IoT) has seen explosive growth these past few years, mainly because it is incredibly affordable to do. It costs a fraction of a dollar to add internet access to a product which will not only increase the value of the product but also add new use cases. There is estimated to be 32 billion IoT devices by 2020.

This thesis will take a look at a IoT measurement platform, which uses advanced mesh networking protocol WPC. WPC allows devices in the network to do local decision making in order to optimize and rearrange without the need of a central control server.
This thesis will also take a look at the used technologies in the making of the data visualization web portal and technical decisions on why said technologies were chosen.

Key words: iot, mesh network

SISÄLLYS

LYHENTEET JA TERMIT

TAMK	Tampereen ammattikorkeakoulu
IoT	Internet of Things (asioiden internet)
PHP	Ohjelmointi kieli jota käytetään enimmäkseen web-palvelinympäristössä.
jQuery	Tehokas opensource Javascript -kirjasto
WPC	Wirepas Connectivity on hajautettu radiokommunikointi protokolla.
Mesh-verkko	Mesh-verkko on verkon rakenne, jossa jokainen laite toimii solmu kohtana (solmuverkko), välittäen viestejä toisille solmuille.
Node	Ohjelmistoltaan kustomoitu RuuviTag anturi beacon, joka sisältää WPC toiminnallisuuden ja mittaa useita arvoja ympäristöstään, välittäen ne gatewaylle jatkuvasti optimoituvaa reittiä pitkin.
Gateway	Gateway, eli portti on mesh-verkoston piste joka mahdollistaa antureiden keräämän datan välittämisen ulkoisille data ja diagnostiikka -palvelimille. Gateway sisältää myös WPC toiminnallisuuden.
RSSI	Received Signal Strength Indicator, eli vastaanotetun signaalin voimakkuuden indikaattori on virran mittaus arvo vastaanotetussa radio signaalista.
Sanitointi	Käyttäjän syötteessä olevan tekstin käsittely siten että se käsitellään vain tekstinä. Menetelmiä ovat erikoismerkkejen poistaminen tai korvaaminen HTML entity koodeilla.

1 JOHDANTO

Tämän opinnäytetyön aiheena on IoT -mittausjärjestelmän web -portaalin kehitystyö Corelase Oy:lle. Opinnäytetyön tavoitteena on dokumentoida IoT -mittausjärjestelmää ja web -portaalissa käytettyjä teknologioita.

Web -portaalin tarkoituksena on visualisoida mittausjärjestelmästä kerättyä historiallista dataa sekä reaaliaikaista dataa, jonka visualisointiin käytetään JavaScript kirjasto HighChartsJS:ää. HighCharts on erittäin tehokas kaaviokirjasto, joka sisältää kaikki tarvittavat kaaviotyypit kuten viiva, pylväs ja ympyrä -kaaviot.
Web-portaaliin toteutetaan oma sisällön hallintajärjestelmä, jonka avulla eri asiakasyritysten käyttäjiä, mittausjärjestelmiä, kaavioita ja sivuja voidaan hallita.

Portaali sisältää kaavioiden tekoa varten työkalun joka listaa yrityksen mittausjärjestelmien laitteet, sekä niiden eri mittausarvotyypit, joista voidaan kasata toistaiseksi vain viivakaavioita.
Kaaviot lisätään sivulle dynaamisesti käyttäen SQL-kyselyllä saatua listaa sivun kaavioista, joka syötetään underscore.js kirjaston template -moottorille, jonka jälkeen jokaiselle kaaviolle tehdään kysely kaavion sisältämästä datasta, joka syötetään kaavioiden pistesarjoiksi.

Mittausjärjestelmä itsessään koostuu useista RuuviTag Bluetooth anturi beaconeista ja yhdestä tai useammasta Corelase gateway laitteesta, jonka tehtävä on välittää mittausjärjestelmän mittaamat tiedot ulkoverkkoon.
RuuviTag on pienikokoinen IoT -laite, joka mahdollistaa ympäristön arvojen mittaamisen. Mitattavia arvoja ovat mm. lämpötila, ilmanpaine, ilmankosteus ja kiihtyvyys.
Gateway on Bluetooth anturilla varustettu mikrokontrolleri, jonka tehtävä on välittää järjestelmän mittaamat arvot ulkoverkkoon, data palvelimelle. Gateway sisältää kosketusnäytön, mahdollisia tulevaisuuden laajennuksia varten, kuten verkon topologian visualisointi.

Mittausjärjestelmän laitteisto muodostaa paikallisen mesh -verkoston, käyttäen WPC protokollaa, joka mahdollistaa tehokkaan mesh -verkon luonnin millä tahansa radio laitteella,

taajuudesta ja rakenteesta riippumaatta. Bluetooth anturit menisivät tavanomaisesti jumiin, jos niitä olisi useampi kymmentä lähietäisyydellä, kun taas WPC:llä antureita voitaisiin laittaa pahvilaatikkoon toista sataa, ilman jumittumista.

Perinteisesti mesh -verkolla tarkoittaa verkko topologiaa jossa verkon laitteet ovat yhteyksissä niin moneen muuhun verkon laitteeseen kuin mahdollista, ja verkon topologiaa pystyttäisiin muuttamaan lennosta, tai jos yksi yhteys katkeaa, voidaan data reitittää toista reittiä pitkin. Radioteknologiassa mesh -verkko on käytännössä sama, verkon laitteet kommunikoivat keskenänsä langattomasti, välittäen datan eri verkon pisteiden kautta mahdollisimman nopeasti ja optimaalisesti. Jos verkossa laite menee rikki, ei se vaikuta muiden laitteiden yhdistettävyyteen, sillä verkko alkaa reitittämään kyseiseen laitteeseen yhteydessä olevien laitteiden dataa jonkin muun verkon laitteen kautta.

2 TEHTÄVÄNANTO

Projektin tehtävänantona on toteuttaa IoT -mittausjärjestelmälle web portaali, joka sisältää oman sisällönhallintajärjestelmän.

Ensimmäisen portaalin tavoitteita

- Asiakasyrityksen luominen
- Käyttäjien rekisteröinti ja autentikointi
- Kaavioiden luonti ja hallinta
    - Node verkkojen merkkaaminen yritykselle
    - Nodejen hallinta
    - Nodejen nimeäminen
    - Node nimien ryhmitys
- Sivujen luonti
- Kaavioiden lisääminen sivuille

Asiakasyrityksille tehdään node verkostoja, joidenka nodet asiakasyrityksen admin voi nimetä ja ryhmittää ja luoda kaavioita nimetyistä nodeista (aluksi pelkästään viivakaavio).
Noden vaihtamisesta täytyy tehdä mahdollisimman helppoa, joten noden nimeämiseen käytetään erillistä kantaa, joka viittaa johonkin nodeen. Mitattu data syötetään mittaukset kantaan nimetyn noden mukaan, jolloin jos node täytyy vaihtaa, kaavioita ei tarvitse muokata eikä historiallista dataa tarvitse siirtää toiselle nodelle, vaan nimetyn noden node id vaihdetaan.

3 LAITTEISTO

Harjoitustyön IoT -mittausjärjestelmä on gatewaysta ja nodeista koostuva, itsenäisesti optimoituva mesh-verkko jossa jokainen node mittaa tietoa ja välittävät ne optimaalisinta reittiä pitkin gateway nodelle.

3.1 Node

Kuvissa (1) ja (2) on *Ruuvi Innovations Oy*:n RuuviTag (Node), avoimen lähdekoodin bluetooth anturi beacon, joka sisältää kiihtyvyys, lämpötila, suhteellinen ilmankosteus ja ilmanpaine -anturit.

Nodeen on ajettu Corelasen oma ohjelmisto, joka hyödyntää WirePas Oy:n Wirepas Connectivity (WPC) protokollaa, joka on alun perin kehitetty Tampereen Teknillisessä Yliopistossa (TTY). Ohjelmistoa asennettaessa nodelle määritetään node ja verkosto id. Nodet kommunikoivat vain samassa verkostossa olevien nodejen kanssa. WPC protokollan ansiosta noden elinkaari on arviolta 3-5 vuotta ilman että paristoa tai nodea täytyy vaihtaa, tämä perustuu nukkumiseen. Nodet nukkuvat suurimman osan ajata ja heräilevät aina 3 - 4 sekunnin välein ja vaihtavat datat naapurien kanssa. Mittaustietojen lähetystiheys on etukäteen ohjelmoitu ja se on tyypillisesti 1 - 10 min välein. Jatkuvassa käytössä tiedonsiirto nopeus on tyypillisesti 3 kbit/s.

KUVA 1. RuuviTag (Node)

KUVA 2. RuuviTag -piiri

3.1.1 RuuviTag raakadatan parsetus

Node lähettää data palvelimelle mittaus datan ”RAW” muodossa, joka on pitkä heksadesimaali merkkijono, esim. 28-cb-4d-01-00-94-09-00-00-2a-2f-7c-01-72-51-00-00-8c-00-00-00-30-00-00-00-ec-03-00-00.

Mittaus data voidaan purkaa käyttämällä implode funktiota väliviiva merkille, tämä luo taulukon kaikista erillisistä tavuista. Ensimmäinen arvo ei ole tavu vaan base10 numero, joka merkkaa tavujen määrän, sen voi ottaa pois splice funktiolla. RuuviTagin lähettämä tavumäärä on aina 28, ellei mittauksia ole jäänyt bufferiin, jos tavuja on siis enemmän kuin 28, tulee mittauksien parsetus toteuttaa yksinkertaisessa for loopissa, joka lukee 28 tavua kerralla.

Tavu taulukko sisältää tietyn määrän tavuja eri mittatyypeille, joidenka arvot täytyy käsitellä siten, että siitä saadaan todellinen luku (Taulukko 1). Mittausdata käyttää little endian LSB (least significant byte) formaattia, eli tavut luetaan oikealta vasemmalle.

Bytes	Type	Hex (LSB)	Modifier	Value
1-4	ID	cb-09-01-00	-	85451
5-8	°C	94-09-00-00	/100	24,52 °C
9-12	Pa	2a-2f-7c-01	/256/100000	~0,9733 Pa
13-16	% RH	72-51-00-00	/1024	20,36 %
17-20	X m/s^2	8c-00-00-00	/1000	0,14 m/s^2
21-24	Y m/s^2	30-00-00-00	/1000	0,048 m/s^2
25-28	Z m/s^2	ec-03-00-00	/1000	1,004 m/s^2

TAULUKKO 1. RAW datan parsetus

3.2 Gateway

Kuvassa (3) on Corelase Oy:n gateway 1.0, joka toimii vain datan välittäjänä, yhdessä verkostossa voi olla useampi gateway. Gateway koostuu STM32F7 sarjan mikrokontrollerista (Kuva 4), joka sisältää ARM Cortex-M7 prosessorin, 1MB flash muistia ja 340KB keskusmuistia, LCD -kosketusnäytöstä ja Bluetooth anturista, joka noden kaltaisesti hyödyntää WPC:tä.

KUVA 3. Gateway 1.0 laatikko

KUVA 4. Gateway -piiri

Gatewayn näytöllä, (kuva 5) on gatewayn vakiokäyttöliittymä, joka näytetään, kun gateway on vastaanottanut konfigurointi palvelimelta data ja/tai diagnostiikka palvelimien sijainnit. Tämän jälkeen gatewayn ainoa tehtävä on välittää node verkoston mittaama data ulkoisille palvelimille. Käyttöliittymä näyttää oleellista tietoa, kuten gatewayn ja data palvelimen ohjelmiston versio, ulkoisten palvelimien sijainnit, node id, verkosto id, naapureiden lukumäärän, talletettujen mittaus pakettien lukumäärän sekä näkyvistä nodeista lista, jossa listattu noden id, kanava, luotettavuus prosentti, RSSI arvon ja aika viimeisimmästä vastaanotetusta paketista.

Mikrokontrollerin tehokkuuden ja kosketusnäytön ansiosta gatewayn ohjelmistoa voidaan tulevaisuudessa laajentaa mm. visualisoimaan jonkin noden dataa tai mesh -verkon topologiaa reaaliajassa.

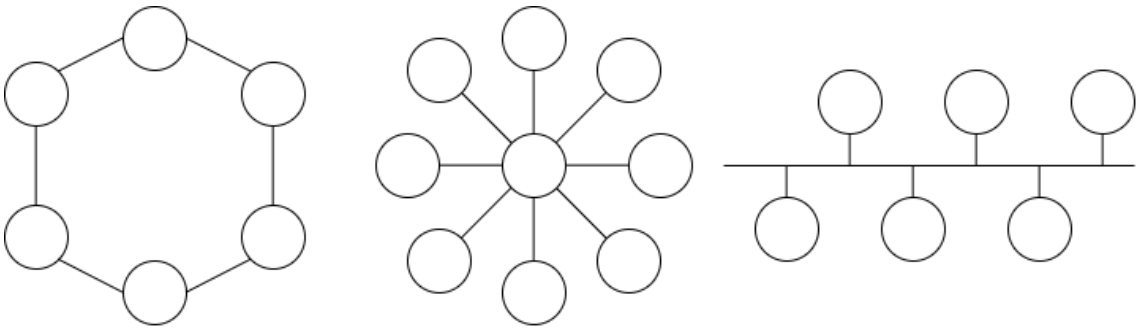
KUVA 5. Gatewayn kosketusnäyttö

4 MESH -VERKKO

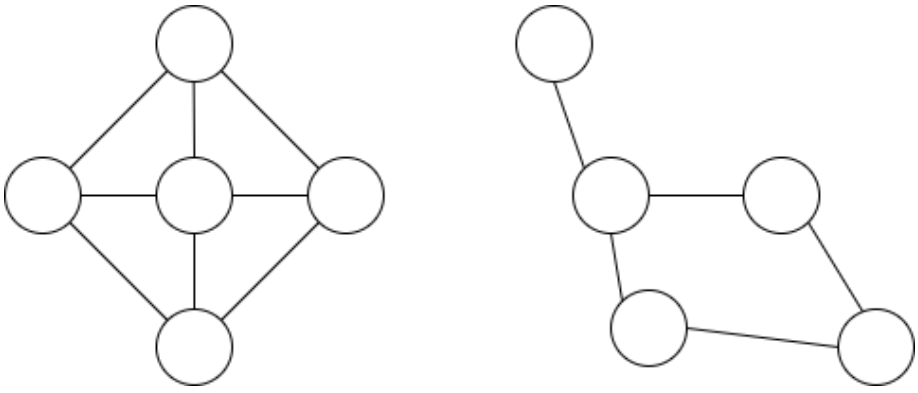
Mesh -verkko on verkon topologinen rakenne siinä missä rengas, väylä ja tähti -topologiat (Kuva 6). Mesh-verkko on edellä mainittuihin topologioihin verrattuna kuitenkin uniikki, sillä se optimoi verkostoa, dynaamisesti organisoimalla ja konfiguroimalla itseänsä (Kuva 7).

Mesh-verkko voidaan toteuttaa fyysisesti, jolloin jokainen verkon piste on johdotettu mahdollisimman moneen muuhun pisteeseen, tällöin yhden pisteen yhteyden kadotessa verkko voi jatkaa toimintaa normaalisti, ellei jonkin pisteen kaikki yhteydet muihin pisteisin katkea. Fyysisen mesh -verkon toteutus on tosin suhteellisen kallis ja aikaa vievä verrattuna muihin verkko-rakenteisiin. (howstuffworks.com)

Radiolaitteiden kehityksen ja halventuneiden tuotantohintojen mahdollistama modernimpi lähtökohta, joka eliminoi fyysisen toteutuksen ongelmat, on toteuttaa verkko langattomasti käyttäen radioteknologioita, jolloin verkon pisteitä voidaan lisätä ja poistaa vaivattomasti.



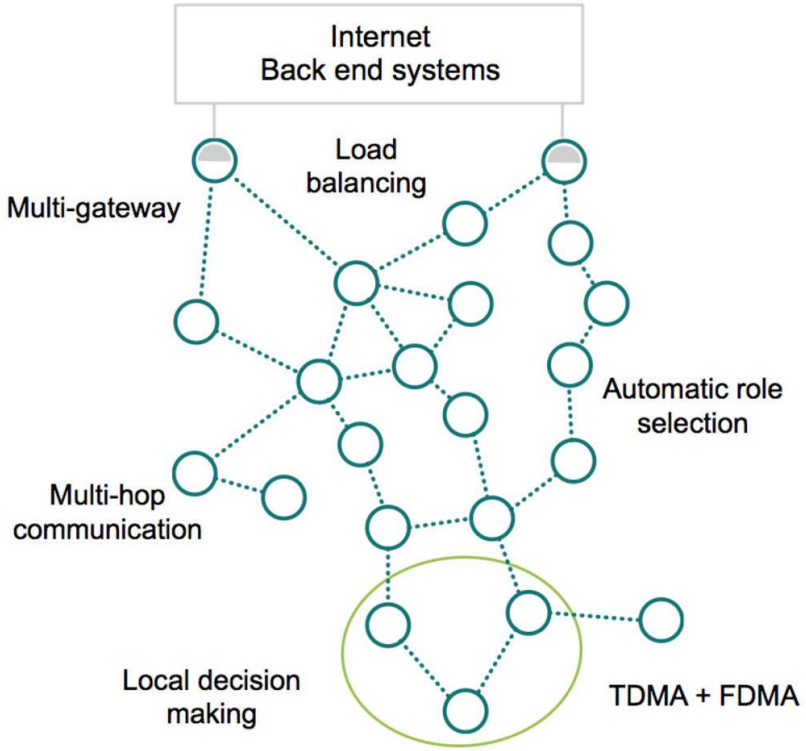
KUVA 6. Rengas, tähti ja väylä -topologiat



KUVA 7. Mesh-verkon organisoituminen

4.1 Wirepas Connectivity

IoT -mittausjärjestelmän kaikki laitteet käyttävät mesh-verkon luomiseen Wirepasin WPC -ohjelmistoa (kuva 8). WPC (Wirepas Connectivity) on hajautettu radiokommunikointi protokolla, jota voidaan käyttää missä tahansa laitteessa, millä tahansa radio piirisarjalla ja taajuudella, laitteisto ja taajuus riippumattomuuden ansiosta.
WPC mahdollistaa useamman gatewayn käytön, eli verkolla voi olla yksi tai useampi yhteys ulkoverkkoon. Verkko on itseorganisoituva, eli verkon pisteet valitsevat omat roolinsa ja tasapainottavat verkon kuormituksen automaattisesti ilman keskuspalvelun käyttöä. WPC tukee TDMA ja FDMA radioteknologioita.



KUVA 8. Wirepas -verkko (https://wirepas.com/)

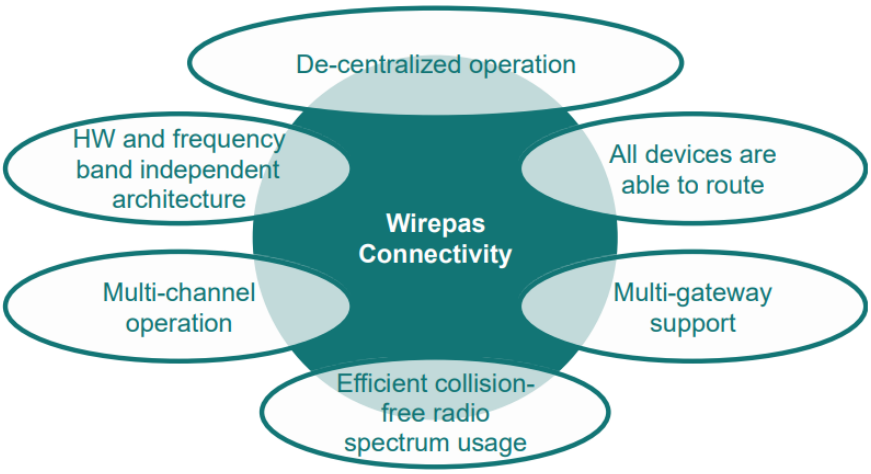
4.1.1 Keskeiset toimintaperiaatteet

Wirepas Connectivity on suurempi kokonaisuus toimintaperiaatteita (Kuva 9), kuten taajuus ja laitteisto riippumaton arkkitehtuuri.
Verkon jokainen laite voi reitittää tietoa jolloin jokainen laite on potentiaalinen verkon piste, jolloin verkon rakennetta tai laitteiden rooleja ei tarvitse erikseen suunnitella tai

ohjata, vaan kaikki tapahtuu automaattisesti, sillä laitteet ovat liitettävyyden näkökulmasta homogeenisiä.

Kaikki verkon logiikka on hajautettu verkon laitteille, jolloin päätökset tehdään paikallisesti, kuten parhaiden naapurien valinta, optimaalisimmat siirrosvirrat ja luotettavimmat taajuuskanavat. Hajautetun logiikan ansiosta keskushallintajärjestelmälle ei ole tarvetta ja protokolla toimii verkon koosta tai laitteiden sijainnista riippumatta.

WPC mahdollistaa useamman gatewayn käytön, jolloin verkon kapasiteettia voidaan laajentaa helposti, sillä verkko hajauttaa kuorman automaattisesti eri gatewayden välillä. Useampi gateway myös eliminoi potentiaalisen verkon suorituskykyongelman jolloin yhden gatewayn yhteyden katketessa verkko alkaa automaattisesti reitittämään dataa toiselle gatewaylle.



KUVA 9. WPC keskeiset toimintaperiaatteet (**https://wirepas.com/)**

5 VERKKOPORTAALI

Verkkoportaalin ideana on visualisoida IoT-mittausjärjestelmien mittaamaa dataa käyttäen erilaisia kaavioita, portaalin tulee myös sisältää helppokäyttöinen sisällön hallintajärjestelmä.

5.1 Lähtökohta

Projektin lähtökohtana on Bootstrap 3.3.x: lle rakennetun SmartAdmin alustan päälle rakennettu julkaisualusta (10). SmartAdmin on responsiivinen websovellus -alusta, jolle löytyy mm. AngularJS, HTML, React, .NET ja PHP versiot. SmartAdmin sisällyttää SmartUI komponentin, jolla voidaan helposti luoda mm. navigointi mukaan lukien breadcrumb, karuselli, datataulukko ja järjesteltävä widgetti.

Kuva 10. SmartAdmin - Portaalin lähtökohta

5.1.1 Navigaatio

SmartAdmin sisällyttää SmartUI -kirjaston, jonka config.ui.php tiedostoa käytetään sivupaneelin navigointi listan toteuttamiseen, kustomoidussa julkaisu alustassa config.ui tiedosto sisällytetään View luokassa, joka on vastuussa näyttöjen piirrosta ja näyttöjen muuttujien hallinnasta.

Navigaatiopaneelin on tarkoitus sisällyttää lista sivuista, nodeista, charteista, yrityksistä, jne. Esimerkki konfiguraatio tiedosto joka sisältää alilistan sivuista ja breadcrumbs mahdollisuuden.

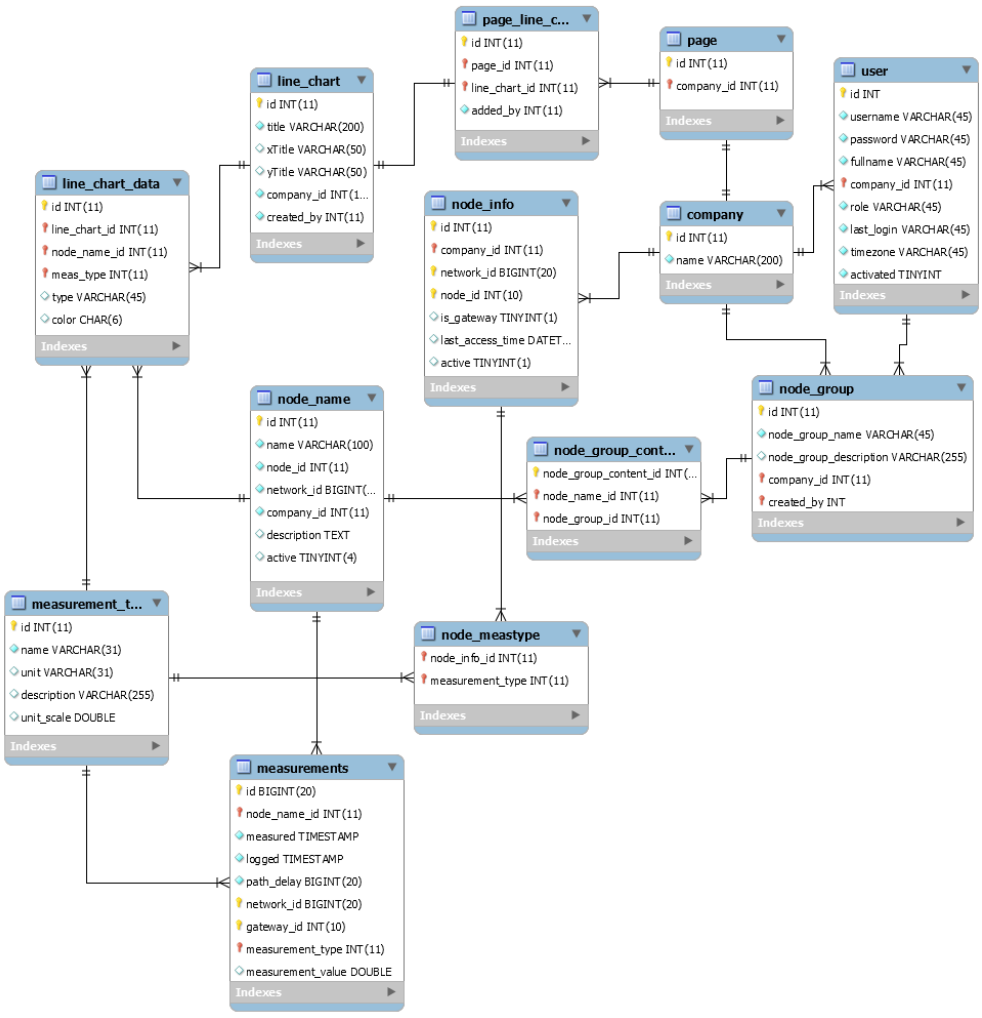
```php
<?php
// $this->view->breadcrumbs
$breadcrumbs = array(
  "Home" => APP_URL
);
// See if $this->view->list_of_pages is defiend on page controller.
$list_of_pages = isset($this->list_of_pages) ? $this->list_of_pages : "";
// Array of elements to be used to build the navigation panel
$listofpages = array();

// Loop through pages defined in view and append to array
if(is_array($list_of_pages)) {
  foreach ($list_of_pages as $list_of_page) {
    // Parse page name
    $page_name = strtolower(str_replace(' ', '_', $list_of_page->page_name)) . "-" . $list_of_page->page_id;
    $listofpages = array_merge($listofpages, array(
      $page_name => array(
        "id" => $list_of_page->page_id,
        "title" => $list_of_page->page_name,
        "icon" => "fa-bar-chart",
        "url" => APP_URL . "page/view/" . $list_of_page->page_id
      )
    ));
  }
}

// Add button for creating new page
$listofpages = array_merge($listofpages, array(
  "btn_add_new_page" => array(
    "title" => "Add new",
    "icon" => "fa-plus",
    "url" => APP_URL . "page/add"
  )
));

// Final page navigation used by SmartUI to render sidepanel
$page_nav = array(
  "home" => array(
    "title" => "Home",
    "icon" => "fa-home",
    "url" => APP_URL
  ),
  "pages " => array(
    "title" => "Pages",
    "icon" => "fa-file ",
    "sub" => $listofpages // Use list of pages as a sub list
  )
);
```

Koodi 1. Esimerkki SmartUI konfiguraatio tiedosto

5.2 Tietokanta

Tietokanta oli jo osittain toteutettu, oheisessa kuvassa (11) on kannan rakenne, jollaisen itse toteuttaisin projektin tarpeet huomioon ottaen. Kanta (11) sisällyttää yksinkertaisen rakenteen sivujen ja kaavioiden sisällön hallinnalle ja nodejen nimeämiselle, ja ryhmittämiselle.

Suunnitelman kuva kuvastaa relaatiokantaa, jo toteutettu kannan rakenne ja ei ole relaatiokanta, sillä kannan moottori ei tue sitä. Relaatio ominaisuus jätettään siis implementoimatta ensimmäisestä versiosta, sillä kannan moottori tulisi vaihtaa esim. InnoDB:ksi ja kantaan täytyisi lähteä käsin tekemään foreign-key linkityksiä. Relaatiokantaan integroitumista ei myöskään nähdä tarpeelliseksi vielä ensimmäisessä versiossa, sillä tuote halutaan mahdollisimman nopeasti ulos.



KUVA 11. Tietokanta -schema

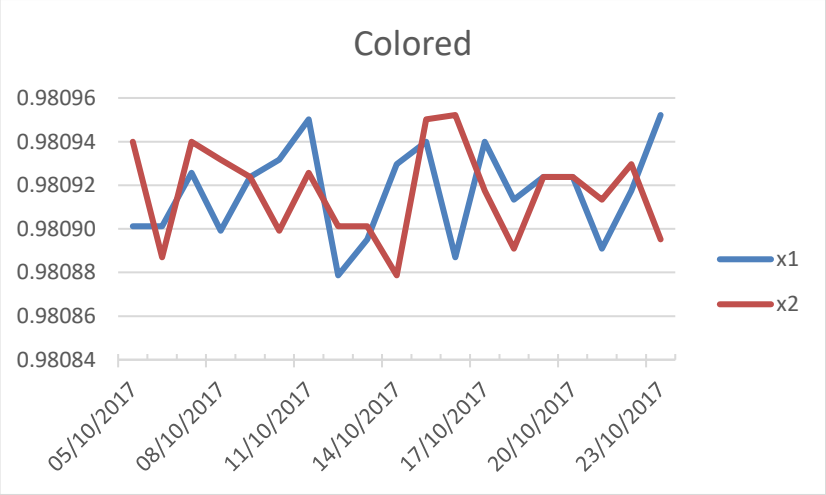
6 DATAN VISUALISOINTI

Projektissa datan visualisointia varten valittiin JavaScript kirjasto HighCharts, joka on monipuolinen ja tehokas kirjasto kuvaajien piirtoa varten. HighCharts mahdollistaa mm. viiva, alue, piste, pylväs, palkki ja ”mittari” -kaavioiden piirron.
Datan visualisointiin päätettiin aluksi käyttää vain viiva, alue ja mittari -kaavioita.
Kuvaaja voi sisältää useamman eri tyylisen data sarjan, esimerkiksi yhden aluekaavion ja loput viivakaavioita. Mahdollinen käyttötapaus eri kaaviotyyppien sekoittamisesta esim. lämpötila viivakaavion ja sademäärä pylväskaavio.

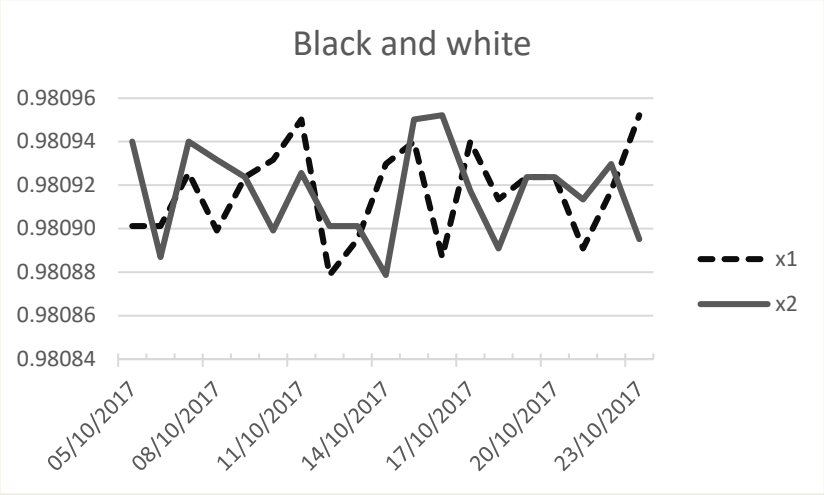
6.1 Viiva- ja aluekaaviot

Viivakaavio on kaavio, joka esittää datan sarjaa datapisteillä jotka on yhdistetty suorilla viivoilla. Viivakaavio tyypillisesti esitetään x ja y -akseli arvoilla, jossa datapisteet kuvastavat x-akselin arvon muutosta y-akselin suhteen.

Viivakaavio voi sisältää useita eri viivasarjoja, jotka erotellaan joko eri väreillä (Kaavio 1) tai jos kaavio tulostetaan mustavalkoisena, eri harmaiden sävyjen lisäksi erottelussa tulisi käyttää kuviollisia viivoja (Kaavio 2).



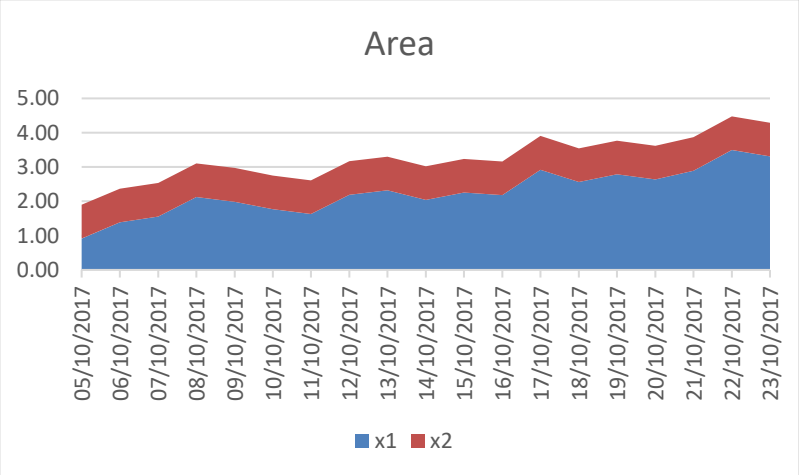
Kaavio 1. Värjätty viivakaavio



Kaavio 2. Mustavalkoinen viivakaavio

Aluekaavio on rakenteeltansa samankaltainen kuin viivakaavio, mutta kuten kaavion nimi vihjaa, sen alue on värjätty. Yksittäisenä data sarjana kaaviossa aluekaaviolla ei ole suurta merkitystä, sillä se on lähtökohtaisesti vain nätti viivakaavio.

Aluekaavioita tulisi käyttää esimerkiksi arvojen summaamiseen, jossa jokainen sarja lisää kokonais-sarjan x arvoon (Kaavio 3). Mahdollinen käyttötapaus on esimerkiksi portfolio visualisointi, jossa aluekaavion eri sarjat merkkaavat omistuksia, velvollisuuksia, jne. Projektissa aluekaavio tyyppiä tullaan toistaiseksi käyttämään vain tyyliteltynä viivakaaviona.

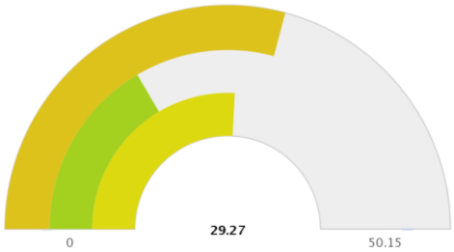


Kaavio 3. Aluekaavio

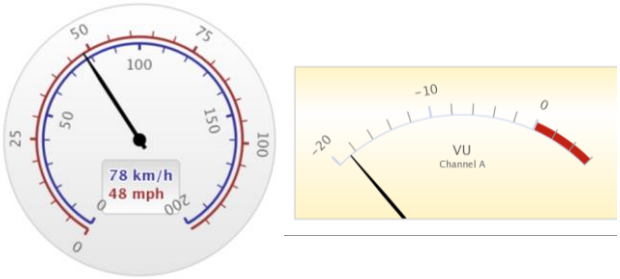
6.2 Mittarikaavio

Mittarikaavio eli ”gauge” on hiukan eksoottisempi ja epätavanomaisempi menetelmä visualisoida dataa. Mittarikaavio kartoittaa viimeisimmän mitatun pisteen arvon, määritetyn numerovälille. Kaavio voidaan värjätä vertaamalla mitattua arvoa kyseiseen numeroväliin.

Solid mittarikaaviota tullaan käyttämään projektissa mm. lämpötilojen visualisoinnissa. Mittarikaavio voi esittää autopeleistä tuttua vauhtimittaria (Kuva 12) tai tavanomaisempia mittari ulkoasuja kuten paine/nopeus -mittari ja jännitemittari (Kuva 13).



Kuva 12. HighCharts solid gauge



Kuva 13. Perinteisempiä mittari tyylejä. (www.highcharts.com)

7 SIVUJEN SISÄLLÖN HALLINTA

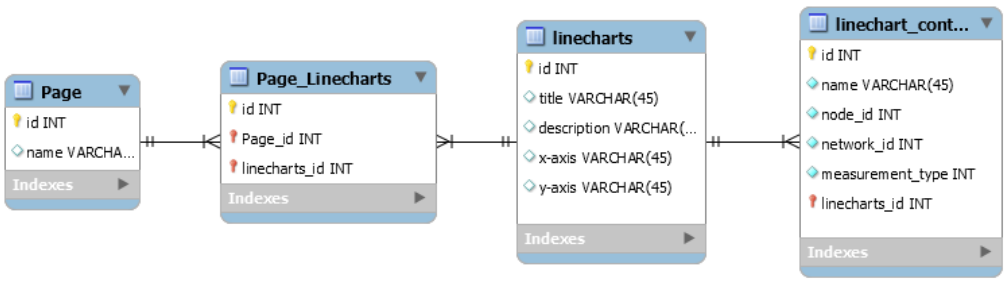
Sisällön hallinnalla tarkoitetaan menetelmää, jonka avulla voidaan dynaamisesti luoda, muokata, ja poistaa sisältöä palvelussa. Tavanomaisin menetelmä on käyttää sisällön hallintajärjestelmää, joka tunnetaan termillä CMS (content management system).
Sisällön hallintajärjestelmät tarjoavat yksinkertaisen ”hallitsija” portaalin jossa voidaan käydä luomassa sivuja ja muokata kyseisten sivujen sisältöä, järjestelmän omalla editorilla. Mainittavia sisällön hallintajärjestelmiä ovat mm. Drupal, Typo3, ja WordPress.

Projekti vaatii alkuun järjestelmän, jolla voidaan lisätä, muokata, ja poistaa kaavioita sivuilta. Ns. ”out of the box” sisällön hallintajärjestelmän sijaan siitä halutaan oman tuntuinen. Projektiin päädyttiin toteuttamaan oma sisällönhallintajärjestelmä, portaalin sisälle, jonka eri osa-alueet ovat lukittuna tiettyjen käyttäjärooli arvojen taakse.

7.1 Esimerkki kanta

Sivun sisällön piirtäminen kannasta on suhteellisen suoraviivaista, otetaan esimerkkinä viivakaavio. Kantaan tarvitaan vain muutama taulu, ja kaikki piirtoon tarvittava sisältö vaatii vain kaksi kyselyä,

Kuvassa 14 on esimerkki eri kantojen välisistä suhteista



Kuva 14. Esimerkki sivun sisällönhallintakannan rakenteesta

Kyseisestä kannasta voidaan hakea kaikki sivulle kuuluvat viivakaaviot oheisella kyselyllä:

```
SELECT linecharts.* FROM linecharts, page_linecharts
WHERE linecharts.id = page_linecharts.linechart_id
```

```
AND page_linecharts.page_id = 1;
```

Koodi 2. Viivakaavioiden kysely

ja kaavioiden eri nodet sekä mittaustyypit seuraavasti:

```
SELECT linechart_content.* FROM linechart_content
WHERE linechart_content.linecharts_id IN (
    SELECT line_charts.id FROM linecharts, page_linecharts
    WHERE linecharts.id = page_linecharts.linechart_id
    AND page_linecharts.page_id = 1
);
```

Koodi 3. Viivakaavioiden sisällön kysely

Edellä mainitut kyselyt voidaan yhdistää seuraavanlainen JSON objekti

```
[
   {
      title: "chart-title",
      desc: "chart-description",
      x: "x-axis",
      y: "y-axis",
      nodes: [
         {
            name:"node-name",
            node_id: 100,
            network_id: 200,
            measurement_type: 1
         },
         ...
      ]
   },
   ...
]
```

Koodi 4. Esimerkki JSON rakenne kaavioista

7.2 Esimerkki piirto underscore.js kirjastolla

Underscore.js kirjasto sisältää template -moottorin, jolla voidaan dynaamisesti luoda monimutkaisia DOM elementtejä. Underscore käyttää nimensä mukaisesti alaviiva prefixiä kaikissa toiminnoissaan, esim. `_.each()`.

Highcharts vaatii uniikilla id:llä varustetun div elementin, johon kaavio piirretään. Sivua ladattaessa luetaan template (koodi 5) käyttäen jQueryn get funktiota, jonka jälkeen siitä luodaan uusi underscore template, jolle välitetään kaavio JSON objekti. Underscore käsittelee ja palauttaa DOM tekstijonon, joka lisätään sivulle näkymättömänä ja häivytetään näkyväksi, joka lisää responsiivisuuden tunnetta. (koodi 6)

```
<% _.each(data, function(chart, index){ %>
   <div id="page_chart_<%=chart.chart_id%>"></div>
<% }); %>
```

Koodi 5. underscore template kaavioi säiliöiden piirrolle

```
$.get("./templates/graph_template.html", function(res) {
   let chart_template = _.template(res);
   $(chart_template({data: self.graphs})).hide().appendTo(
      $("#page-container")
   ).fadeIn("normal");
});
```

Koodi 6. Esimerkki underscore template kutsu

```
new Highcharts.Chart({
    chart: {
        type: "line",
        renderTo: "page_chart" + chart.chart_id
    },
    title: { text: chart.title},
    xAxis: {
        type: 'datetime',
        title: {
            text: chart.x
        }
    },
    yAxis: {
        title: {
            text: chart.y
        }
    },
    series: measurements
});
```

Koodi 7. Esimerkki highcharts konstruktori

8 TIETOTURVAA

8.1 Käyttäjän oikeuksien tarkastus

Otetaan esimerkkinä sivu www.example.com/profile/3213, jossa 3213 on käyttäjän id ja sivulla voi muokata kyseisen käyttäjän tietoja. Joku käyttäjä voi uteliaisuuttaan muuttaa numeron arvoa ja jos käyttäjän oikeuksia ei tarkisteta, pääsee kyseinen henkilö muokkaamaan muiden käyttäjien tietoja.
Jokaisen pyynnön yhteydessä tulisi aina tarkistaa ehto lausekkeilla, että käyttäjä täyttää tietyt kriteerit ennen kuin sivua piirretään tai kantaan tehdään minkäänlaisia muutoksia.
Tämä voidaan toteuttaa helposti sessio muuttujalla, johon kirjautumisen yhteydessä asetetaan käyttäjän id ja sivua ladattaessa varmistetaan, että sessiossa olevan id:n arvo on sama kuin parametrina vastaanotettu. Jos kantaan tehdään mitään muutoksia, tulee kyselyihin määrittää esim. ”WHERE käyttäjä = :session_käyttäjä”.

8.2 SQL injektio

Jos käyttäjän sallitaan antaa tietoa, jota tullaan käyttämään kanta kyselyissä, tulee kaikki syötteet sanitoida, ja käsitellä vain ja ainoastaan tekstinä.
Otetaan esimerkkinä tilanne (koodi 4), jossa kirjautumisen yhteydessä kantaan tehdään kirjautumiskysely käyttäjän syöttämien arvojen perusteella. Jos hyökkääjä asettaisi kirjautumiskentässä käyttäjänimeksi *’ or 1=1;* kannasta etsittäisiin käyttäjänimeä, joka on tyhjä tai 1=1. SQL ei välitä mitä puolipisteen jälkeen tulee ja 1 on 1, jolloin hyökkääjä pääsee kirjautumaan useimmiten kannan ensimmäisenä käyttäjänä, joka useimmiten on root -käyttäjä.

```
$pdo->query("SELECT * FROM users WHERE
            name = '" . $_POST['name'] . "' AND
            password = '" . password_hash(
                $_POST['pw'],
                PASSWORD_DEFAULT,
                array('cost' => 9 ))
            . "'
);
```

Koodi 8. Esimerkki huonosta kirjautumiskyselystä

SQL injektiot on todella helppo välttää käyttämällä prepare lauseketta (koodi 5) jolloin ensin SQL palvelimelle välitetään kyselyn rakenne jossa data on korvattu esim :pass tekstillä, johon asetettua arvoa kanta käsittelee vain ja ainoastaan tekstinä ja sen jälkeen kyselyn data, execute kutsulla.

```
$q = $pdo->prepare("
     SELECT * FROM users WHERE name = :name AND password = :pass
");
$q->execute(array(
    ":name" => $_POST['name'],
    ":pass" => password_hash(
                  $_POST['pw'],
                  PASSWORD_DEFAULT,
                  array('cost' => 9)
                )
));
```

Koodi 9. Käyttäjän syötteen sanitointi SQL prepare lausekkeella

8.3 XSS

XSS eli Cross Site Scripting on hyökkäys, jossa hyökkääjä pystyy ajamaan sivulla omaa koodia, käyttämällä esimerkiksi forumin kommentti tekstikenttää. PHP XSS hyökkäykset on erittäin helppo välttää, älä ikinä koske eval funktioon!

JavaScript on oma tarinansa, varsinkin jos käyttäjän syötettä käytetään sivun sisällön piirtämiseen. Esimerkkinä aiemmin mainittu forumin kommentti tekstikenttä, oletetaan että palvelimella sivua ladattaessa kysellään kaikki kommentit ja ne piirretään for loopissa esim. echo *”
$message”;*.
Jos käyttäjän syötettä ei sanitoida, voi hyökkääjä kirjoittaa kommentiksi esim. *<script>alert(”XSS”);</script>* jolloin käyttäjien selaimet hyvillä mielin käsittelevät kyseisen koodin JavaScriptinä. Vaikka kommenttikenttä sisältäisi merkki rajoituksen, se ei vaikuttaisi mitenkään hyökkäyksen vakavuuteen, sillä hyökkääjä voi linkittää ulkoisen JavaScript tiedoston *script* elementin *src* parametrilla.

PHP on tehnyt käyttäjän syötteen sanitoinnin niin helpoksi htmlspecialchars funktiolla, että ei ole mitään syytä, etteikö sitä oltaisi implementoitu. Kommenttien piirto olisi siis toteutettu seuraavasti.

echo *”
htmlspecialchars($message, ENT_QUOTES, 'UTF-8', FALSE) ”;*

Jos sivulla käytetään jotain JavaScript kirjastoa, joka lukee jonkin DOM elementin tekstin arvon ja käyttää sitä muun sisällön luomiseen, tulee sekin sanitoida. Esimerkiksi SmartAdminin JarvisWidget ottaa widgetin otsikko DOM elementin tekstin SmartNotifications poisto varmennus modaaliin ilman sanitointia, jolloin selain voi taas tulkita sen JavaScriptinä.

JavaScript ei sisällä valmista funktiota, joten se täytyy toteuttaa itse tai ladata jokin kirjasto sitä varten. Projektia varten toteutettiin jQuerylle laajennus (koodi 6), jolla voidaan sanitoida käyttäjien syöttämä teksti.

```
(function ($) {
    // do not overwrite the namespace, if it already exists
    $.Escape = $.Escape || {};

    // entityMap for regex match
    const entityMap = {
        '&': '&amp;',
        '<': '&lt;',
        '>': '&gt;',
        '"': '&quot;',
        "'": '&#39;',
        '/': '&#x2F;',
        '`': '&#x60;',
        '=': '&#x3D;'
    };
    $.Escape.Html = function (string) {
        return String(string).replace(/[&<>"'`=\/]/g, function (s) {
            return entityMap[s];
        });
    }
})($);
```

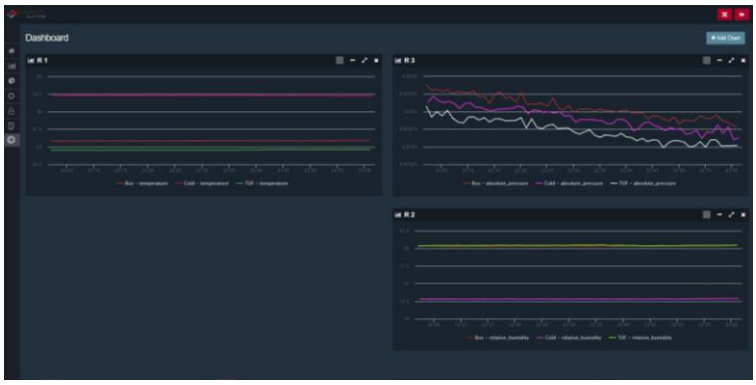
Koodi 10. jQuery teksti sanitointi laajennus

Jotta JarvisWidgetin haavoittuvuus saadaan korjattua, sen koodissa täytyy vain korvata

```
+d.o.labelDelete+' "'+f+'"'
+d.o.labelDelete+' "'+$.Escape.Html(f)+'"'
```

9 POHDINTA

Opinnäytetyön aiheena on voimassa oleva kehitystyö, joka tekee juuri sitä mitä nimi vihjaa, kehittyy. Tämä on hiukan monimutkaistanut opinnäytetyön kirjoitusprosessia, sillä suunnitelmat voivat muuttua suhteellisen nopeaan tahtiin.
Olen oppinut tämän opinnäytetyön aikana paljon eri teknologioista, joihin en olisi itsenäisesti todennäköisesti koskaan tullut tutustumaan, opin myös paljon käyttäjien autentikoinnista ja verkkopalveluiden turvaamisesta.
Aluksi halusin kirjoittaa opinnäytetyössä enemmän datan visualisoinnin säännöistä ja parhaista menetelmistä, mutta vapaa-ajan puutteen vuoksi tämä jäi toissijaiseksi.
Kaikki tehtävänanto kappaleessa mainitut ominaisuudet on implementoitu (kuva 14).



KUVA 14. Tämänhetkinen web-portaalin tilanne

Jatkokehityksenä portaaliin tullaan lisäämään eri kaaviotyyppejä ja parantamaan käyttäjäkokemusta, siten että sisällön hallintaprosessi on mahdollisimman intuitiivista.
Kaavioiden piirto voitaisiin toteuttaa underscore.js kirjaston sijaan käyttämällä PHP:n foreach toiminnallisuutta, sillä underscore hidastaa sivun ensimmäisen merkityksellisen piirron aikaa ilman merkityksellisiä hyötyjä.

Sivujen sisältö tulee koostumaan useasta erilaisesta sisältötyypistä, joita täytyy erikseen kysellä kannasta. Sen sijaan että kantaan toteutettaisiin erillinen taulukko kaikille sivun eri kaavio tyypeille (viiva, pylväs, mittari, jne.) ja koodissa kyseltäisiin erikseen kyseisiä kantoja, voitaisiin toteuttaa polymorfinen assosiaatio kanta, josta voitaisiin kerralla kysellä joka ikistä sivulle kuuluvaa kaaviota kerralla, tyypistä riippumatta.

IoT -markkina on suuressa kasvussa eikä loppua näy, uskon että helppokäyttöiselle mittausjärjestelmälle on käyttöä niin yksityiselle kuin yritysasiakkaalle.

Yksityinen henkilö voi käyttää järjestelmää esimerkiksi kodin automatisoinnissa ja valvonnassa. Yritysmaailmassa vain mielikuvitus on rajana, käytännön käyttötarkoitus jonka näen ajankohtaisena, on esim. rokote ja lääke -lähetykset, joidenka täytyy olla koko kuljetuksen ajan tietyssä lämpötilarajassa, etteivät ne mene pilalle. Kuljetusautosta voitaisiin tehdä kulkeva mittausjärjestelmä laittamalla kaikkiin lähetyslaatikoihin yksittäiset nodet, jolloin kuljetuksesta saataisiin historiallista tietoa ja tuotteiden laatu voidaan taata.

LÄHTEET

Wirepas Connectivity. Luettu 15.10.2017
https://wirepas.com/connectivity/technology/

RuuviLab. Luettu 15.10.2017
https://lab.ruuvi.com/

SmartAdmin UI demo. Käyty 8.12.2017
http://wrapbootstrap.com/preview/WB0573SK0

HighCharts. Käyty 15.11.2017
https://www.highcharts.com/docs/

How stuff works. 14.12.2017
https://computer.howstuffworks.com/how-wireless-mesh-networks-work.htm

Meshdynamics. 14.12.2017
http://meshdynamics.com/