Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

OLLI KANGAS

# Datan keruu ja visualisointi

TIETOJENKÄSITTELYN TUTKINTO-OHJELMA
2021

Tekijä(t)
Kangas, Olli

Julkaisun laji
Opinnäytetyö, AMK

Päivämäärä
5/2021

Sivumäärä
23

Julkaisun kieli
suomi

Julkaisun nimi
**Datan keruu ja visualisointi**

Tutkinto-ohjelma
Tietojenkäsittelyn koulutusohjelma

Opinnäytetyön tarkoituksena on tarkastella datan kulkua aurinkopaneeleilta, tietokantaan ja sieltä esittelysivustolle. Työn tavoitteena on saada kerätty data esille siten, että siitä saadaan nopeasti ja tarkasti selville aurinkopaneeleiden tila ja, että analysoitava data on helposti luettavaa.

Työssä käydään läpi datan keräämiseen, tallentamiseen ja visualisointiin tarvittavat teknologiat ja työkalut. Opinnäytetyössä käsitellään koko sovelluksen kehittäminen yksityiskohtaisesti palvelinpuolelta käyttäjärajapintaan saakka.

Lopuksi tehdään yhteenveto, jossa käsitellään työn onnistuminen, mahdolliset jatkokehitysideat ja mitä työtä tehdessä opittiin.

Asiasanat
Data, visualisointi, web-sovellus

| Author(s)<br>Kangas, Olli | Type of Publication<br>Bachelor's thesis | Date<br>5/2021 |
|---|---|---|
| | Number of pages<br>23 | Language of publication:<br>Finnish |

Title of publication
**Data collecting and visualization**

Degree program
Business Information Systems

The purpose of the thesis is to examine the flow of data from solar panels, to database and from database to user interface of the application. The goal of the thesis is to make the data visible so that the data is easy to read and to analyze.

The thesis goes through the process of data collecting, saving the data to database and visualizing the data in a simple web application. The work covers technologies and tools used in developing the application in detail, from server side to user interface.

The synopsis will be the final part of the thesis. Synopsis will consist of summary of the work, further development ideas and what was learned developing the application.

Key words
Data, visualization, web application

# SISÄLLYS

# 1 JOHDANTO

Tässä opinnäytetyössä tarkastellaan datan kulkua aurinkopaneeleilta tietokantaan, tietokannasta API-rajapintaan ja lopuksi sivustolle, jossa data näytetään kaaviomuodossa. Opinnäytetyönä luodaan sovellus, joka toimii edellä mainitulla tavalla.

Sovelluksen tarkoituksena on tallentaa kerätty data tietokantaan, josta sitä voidaan kysellä ja näyttää kaaviossa. Kaaviosta tulisi saada selville aurinkopaneelin tila ja esiteltyä dataa tulisi pystyä analysoimaan.

Lukijalle selviää työn edetessä, miten data kerätään tietokantaan, miten se saadaan kyseltyä tietokannasta ja kuinka data lopuksi esitetään. Lukijalle selvitetään myös tarvittavat teknologiat ja miksi juuri kyseessä olevat teknologiat on valittu.

Aluksi työssä käydään läpi mitä datan hyödyntäminen ja visualisointi tarkoittaa, jonka jälkeen siirrytään sovelluksen kehittämiseen. Kehittäminen aloitetaan määrittelemällä sovelluksen tarkoitus. Ongelman määrittelyn jälkeen käydään läpi sovelluksen kehittäminen palvelinpuolelta käyttöliittymäpuolelle. Työtä lukiessa käydään ensin läpi teknologia, jota on käytetty, jonka jälkeen esitellään, miten kyseistä teknologiaa on käytetty.

# 2 YLEISTÄ

Datan visualisoinnilla tarkoitetaan datan saattamista muotoon, jossa sitä on helppo lukea. Visualisointi tapahtuu työkaluilla, joihin kuuluu muun muassa erilaiset kaaviot, kartat ja kuvaajat. Nykypäivänä datan visualisointi on tärkeä osa melkein jokaista alaa, koska visualisoitu data on helpompi lukea ja ymmärtää kuin pelkät numerot paperilla. Datan visualisoinnin perimmäinen tarkoitus onkin tehdä datasta ymmärrettävämpää.

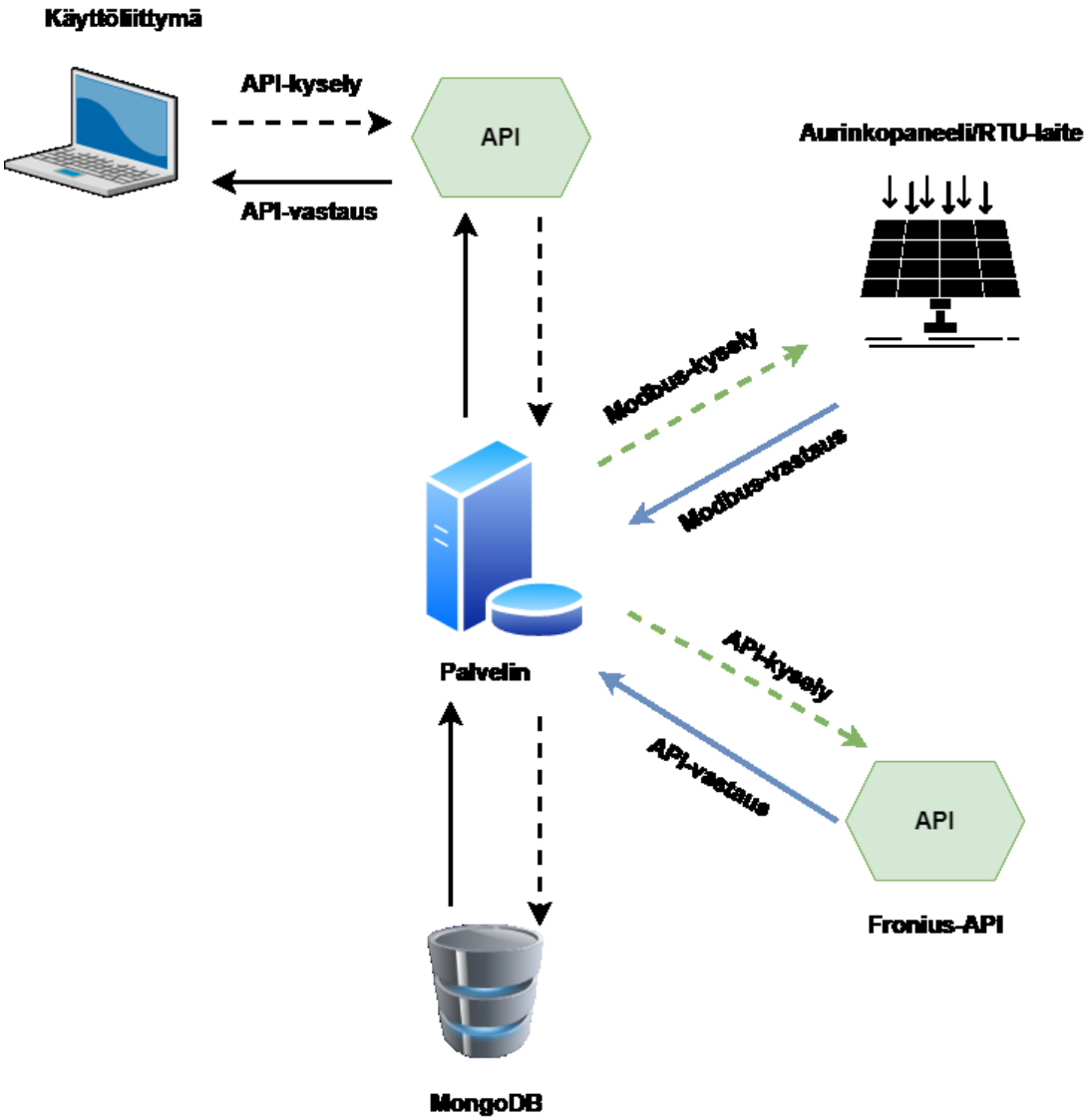
Visualisointi kuitenkin on harvoin pelkästään kaavioiden tekemistä. Monesti visualisointi on ikään kuin tarinan kerrontaa. Korostetaan tärkeimmät asiat voimakkailla väreillä, jotta huomio kiinnittyy oikeisiin paikkoihin. Data täytyy myös siistiä, jotta kaikki ylimääräinen jää pois, eikä sekoita pääasioita. Kuitenkin datan visualisointi on yksinkertaisimmillaan hyvinkin helppoa, pelkästään Excel:iä käyttämällä voidaan saada aikaan näyttäviä kaavioita ja grafiikoita. Tästä syystä jokaisen olisi hyvä olla perillä edes jossain määrin siitä, mitä datan visualisointi tarkoittaa. (Tableau 2021.)

Välillä on vaikea hahmottaa, miten datan visualisointi näkyy arkielämässä. Esimerkkeinä datan visualisoinnista voisi olla sääkartat, erilaiset aktiivisuusmittarit ja henkilövaaka. Jokaisessa esimerkissä kerätään data, ja esitellään se helposti luettavassa muodossa.

# 3 SOVELLUKSEN KEHITTÄMINEN

## 3.1 Sovelluksen tarkoitus

Sovelluksen tarkoitus on kerätä dataa SMA- ja Fronius-merkkisistä aurinkopaneeleista. Data sisältää muun muassa hetkellisen virran, päivän kokonaistuoton, vuoden kokonaistuoton ja paneelin tuoton asennuksesta lähtien. Data on tarkoitus tallentaa tietokantaan, jonka jälkeen se voidaan visualisoida web-sivustolle kaaviomuotoon. Kaavion tarkoitus on antaa nopeasti ja selkeästi yleiskuva paneelin sen hetkisestä tilanteesta ja kaavion dataa tulisi pystyä analysoimaan. Aloitetaan käymällä läpi sovelluksen suunniteltu toiminta kuvana. (Kuva 1)




Kuva 1. Sovelluksen suunniteltu toiminta.

Palvelin kyselee aurinkoenergiajärjestelmiltä dataa säännöllisin välein ja tallentaa datan tietokantaan. Tallentamisen jälkeen palvelin pyytää tietokannasta datan ja reitittää sen API:in, josta käyttäjä voi API-kyselyillä pyytää datan kaavioon. Kaavio esitetään käyttäjälle yksinkertaisessa web-sovelluksessa.

## 3.2 Palvelinpuoli

Palvelinpuolella (backend) tarkoitetaan käyttäjälle näkymätöntä osaa sovelluksesta. Kaikki mitä käyttäjä näkee, on osa käyttöliittymää (frontend) ja kaikki muu kuuluu palvelinpuolelle. Esimerkiksi tässä työssä datan keruu ja tallentaminen ovat osa palvelinpuolta.

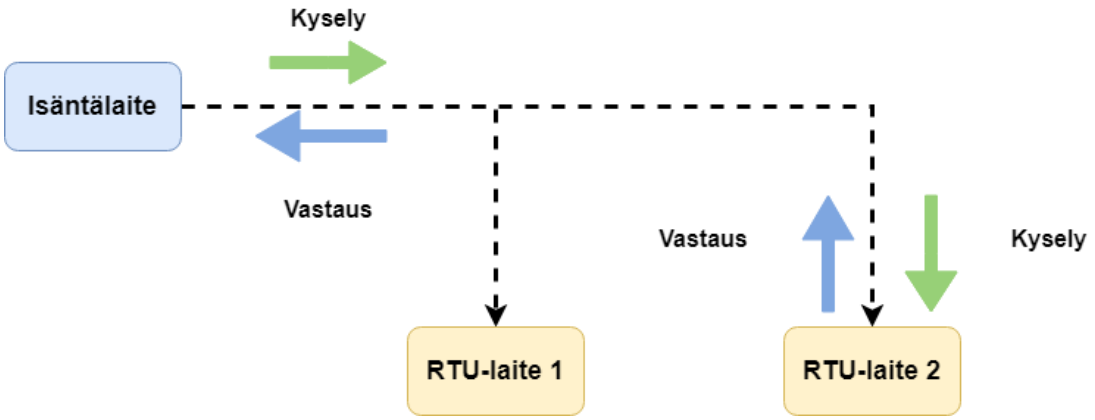
### 3.2.1 API- ja Modbus-kyselyt

Datan lähteenä toimii aurinkopaneelit ja niihin liitetyt muuntimet (inverter). Paneeleita ja muuntimia on monia eri merkkejä ja malleja, tässä työssä pitäydytään SMA Sunnyboy- ja Fronius-merkkisissä aurinkopaneeleissa. SMA-paneelit käyttävät Modbus-protokollaa ja Fronius-merkkiset paneelit käyttävät API-rajapintoja (Application Programming Interface). API on rajapinta, joka sallii kahden eri sovelluksen tietojen vaihdon keskenään. Sovellukset siis keskustelevat keskenään. Tässä tapauksessa Fronius-paneelin ohjaussivusto ylläpitää API:ta joka sisältää JSON-muotoista dataa aurinkopaneeliin liittyen ja opinnäytetyössä kehitetyn palvelimen tehtävä on kysellä kyseiseltä API:lta tietoja aurinkopaneelista. (MDN contributors 2020.)

Datan keruu onnistuu erilaisilla kyselyillä. Fronius-paneelien datan keruu onnistuu yksinkertaisilla API-kyselyillä. API-kyselyllä tarkoitetaan toimenpidettä, jossa ohjelmallisesti pyydetään dataa API:lta. Jos kysely on onnistunut, API vastaa lähettämällä kyseliyn datan vastaanottajalle, jonka jälkeen data voidaan tallentaa tietokantaan. API:t antavat yleensä vastauksen JSON-muodossa.

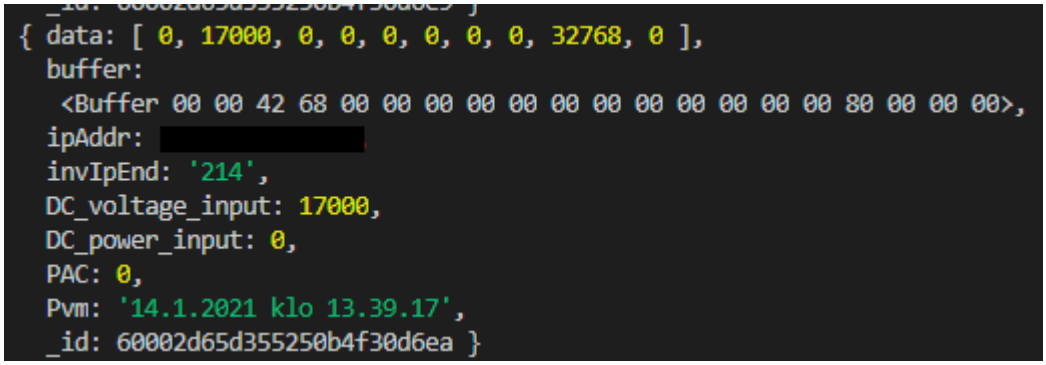
Modbus-kyselyissä vaaditaan erilaisia parametreja, kun taas API-kyselyissä tarvitaan vain API:n osoite. Aloitetaan ensin selvittämällä mitä Modbus tarkoittaa.

Modbus on vuonna 1979 Modiconin julkaisema sarjaliikenneprotokolla. Modbus toimii isäntä/orja-periaatteella. Orjalaitteesta voidaan myös käyttää termiä RTU (Remote Terminal Unit). Standardi Modbus-verkko sisältää yhden isäntälaitteen ja 1-247 RTU-laitetta (Kuva 2). Isäntä lähettää käskyjä RTU-laitteelle, esimerkiksi käskyn lähettää dataa. Isännän käsky sisältää RTU-laitteen osoitteen, joka on numero 1-247 välillä. Käsky pitää sisällään myös funktiokoodin, joka määrittelee, luetaanko vai kirjoitetaanko rekisteriin. Käskyn viimeinen osa on datakenttä, johon määritellään dataosoite (data field), jonka avulla RTU-laite tietää mistä ja mihin asti rekistereitä luetaan. (Schneider electronics www-sivut 2021.)




Kuva 2. Modbus-järjestelmä kaaviona, kahdella RTU-laitteella.

Kyseessä olevassa tapauksessa isäntälaitteena toimii virtuaalitietokone, josta kyselyt lähetetään ja RTU-laitteena aurinkosähköjärjestelmien muuntimiin liitetyt Modbus-laitteet. Data saapuu isäntälaitteelle muokattuna luettavampaan muotoon (Kuva 3).




Kuva 3. RTU-laitteelta saatava data.

### 3.2.2 Datan tallennus tietokantaan

Tietokantana käytettiin MongoDB:tä. Seuraavaksi lyhyt kuvaus tietokannasta, jonka jälkeen käydään läpi datan tallennus.

MongoDB on kaikkein suosituin NoSQL-dokumenttitietokanta. Tietokantaan tallennettava dokumentti on muotoa field:value (vapaasti suomennettuna kenttä:arvo). Arvoina voi toimia toiset dokumentit ja taulukot (array). NoSQL-tietokannalla tarkoitetaan ei relaatiotietokantoja. MongoDB dokumentit muistuttavat JSON-objekteja. Myös tietokantakyselyt suoritetaan JSON-muodossa, mikä helpottaa MongoDB:n

kanssa työskentelyä. MongoDB-tietokannat koostuva kokoelmista (collection). Kokoelmat vastaavat relaatiotietokantojen tauluja. MongoDB:n pääominaisuuksia ovat tehokkuus, varsinkin kyselyjen nopeus, monimuotoinen kyselykieli (query language) ja skaalautuvuus. Kyselykielellä tarkoitetaan syntaksia, jolla tietokannalta kysellään dokumentteja. NoSQL-tietokannat ovat yleisesti ottaen käytännöllisiä suurien datamäärien tallentamiseen, varsinkin jos tallennettava data ei sovi perinteiseen SQL-malliin (rivit, sarakkeet, relaatiot). MongoDB:stä on saatavilla Community- ja Enterprise-versiot. Yhteisöversiossa (Community) on käytännössä samat ominaisuudet, erona se, että Enterprise-versiossa on mukana täysi tuki virhetilanteen sattuessa. (MongoDB contributors 2021.)

Palvelinpuolen ohjelmointikielenä toimi Node.js. Node.js on avoimen lähdekoodin asynkroninen tapahtumapohjainen JavaScript-ajoympäristö, eli ei varsinaisesti ohjelmointikieli, vaikka ohjelmointikielenä siitä useasti puhutaan. Node.js avulla on mahdollista luoda palvelin- ja verkkosovelluksia. Node.js:n kanssa on mahdollista käyttää monia erilaisia kirjastoja ja ohjelmistokehyksiä, näistä suosituimpana mainittakoon Express.js ja Mongoose. Node.js toimii ristiin kaikilla alustoilla. Node.js ominaisuuksiin kuuluu asynkronisuus, tapahtumapohjaisuus ja nopeus. Asynkronisuus tarkoittaa sitä, että ohjelma ei jää koskaan odottelemaan esimerkiksi API-kyselyiden vastauksia, vaan siirtyy heti kyselyn tehtyään seuraavaan osaan ajoa. Node.js syntaksi on samankaltainen kuin JavaScriptissä. (Node.js contributors 2021.)

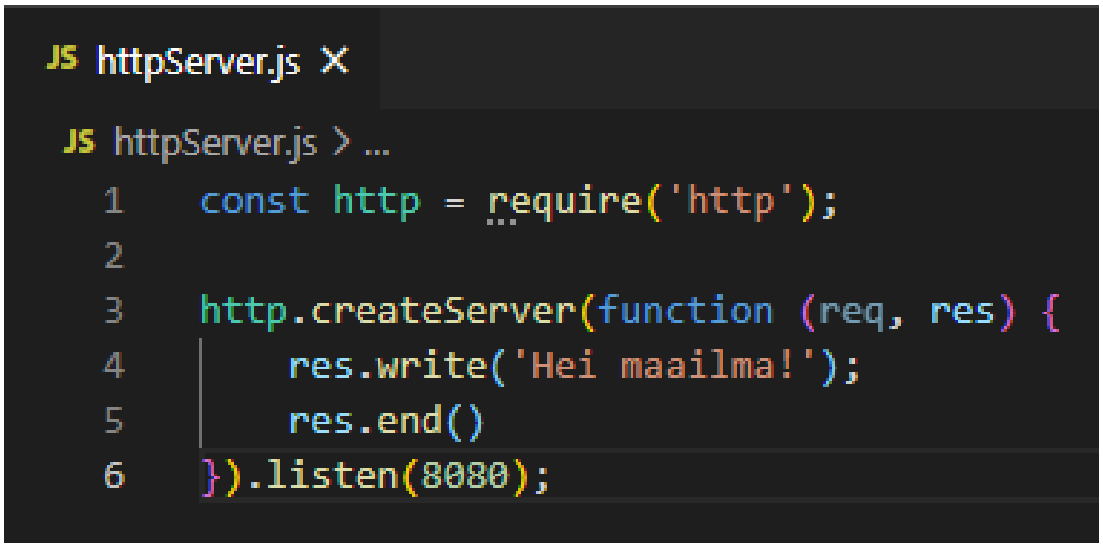
Kun Modbus tai API on vastannut datakyselyyn onnistuneesti, saadaan vastauksesi JSON-muotoista dataa, joka voidaan MongoDB:n sisäisellä toiminnolla tallentaa tietokantaan suoraan. (Kuva 4)


Kuva 4. Datan tallentaminen tietokantaan.

Rivillä yksi luodaan muuttuja MongoClient, joka toimii moduulin määrittelynä. Rivillä kaksi yhdistetään tietokantaan, jossa mongoConn-muuttuja on tietokannan URI-osoite. Rivillä neljä ja viisi määritellään minkä nimiseen tietokantaan ja kokoelmaan data tallennetaan. Funktiolla insertMany suoritetaan muuttujien tallentaminen tietokantaan. Riveillä kuudesta yhdeksään on muuttujia, jotka sisältävät JSON-muotoista dataa.

### 3.2.3 Datan haku ja palvelimen luonti

Seuraavaksi luodaan palvelin, joka isännöi API:ta. Palvelimeksi riittää yksinkertainen HTTP-palvelin (Hypertext Transfer Protocol). Palvelimen tarkoitus on ylläpitää JSON-muotoista dataa, jotta käyttäjä voi kysellä sitä käyttöliittymä puolelta. Node.js palvelimen luominen on yksinkertaista ja onnistuu muutamalla rivillä koodia. (Kuva 5)




Kuva 5. Yksinkertainen esimerkki palvelin.

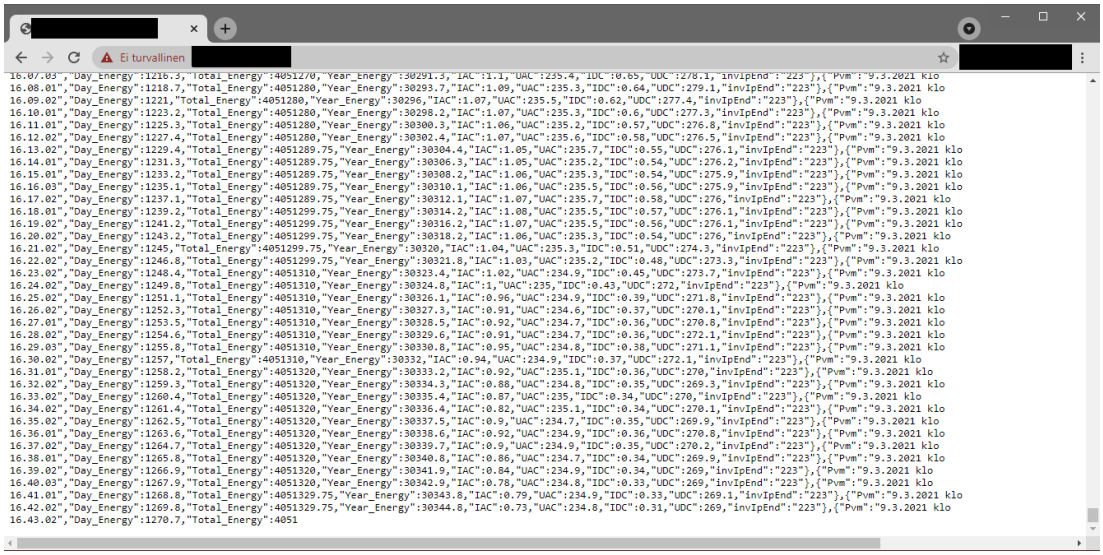
Rivillä kolme käytetään Node.js sisäistä toimintoa createServer, joka luo palvelin objektin. Objektin luomisen jälkeen rivillä kuusi määritellään porttinumero, jota palvelin kuuntelee. Palvelin käynnistetään antamalla komentokehotteessa käsky node ”palvelintiedostonimi”.

Kun palvelin on luotu ja käynnistetty, täytyy data myös saada haettua, jotta se voidaan visualisoida. Datan hakeminen tietokannasta tapahtuu myös MongoDB:n sisäänrakennetuilla toiminnoilla. (Kuva 6)


Kuva 6. Datan reititys ja haku tietokannasta

Tietokantaan yhteyden luominen tapahtuu samalla tavalla, kuin datan tallennuksessakin. (Kuva 4) Erona on reitin luominen app.get-toiminnolla ja dataa ei tallenneta, vaan haetaan tietokannasta find-toiminnolla. Find-toiminto hakee tietokannasta parametrien

mukaisesti datan, ja rivillä kymmenen data lähetetään osoitteeseen /apiRoute. Kun edellä mainitut toimenpiteet ovat tehty, on yksinkertainen API käyttövalmiina. (Kuva 7)




Kuva 7. Reitin /apiRoute sisältämä API.

Osoitteesta /apiRoute löytyy JSON-muodossa olevaa dataa. Riveillä näkyy muun muassa hetkellinen virta, päivän kokonaistuotto ja päivämäärä.

## 3.3 Käyttöliittymä

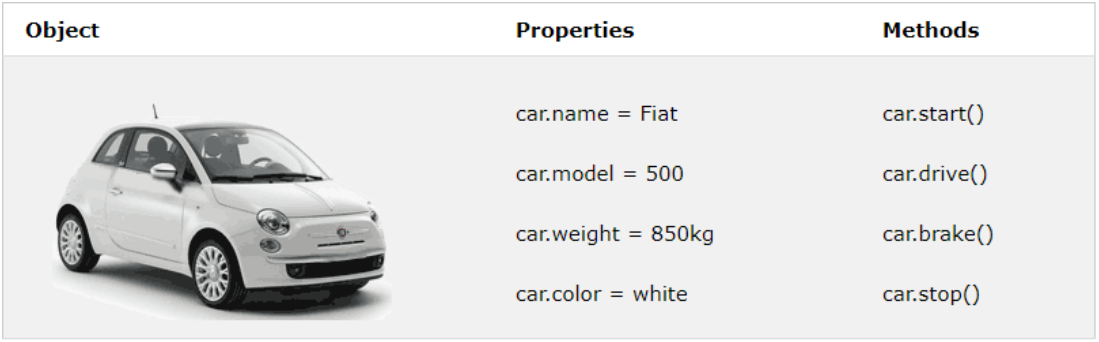
Käyttöliittymän (frontend) vaatimuksena oli selkeä luettavuus, yksinkertaisuus ja helppokäyttöisyys. Käyttöliittymältä vaadittiin myös laitteen, eli aurinkopaneelin, valitsemiseen tarvittava valikko ja aikavälin valitsemiseen käytettävä valikko. Kaavion luomiseen käytettiin ChartJS-kirjastoa ja aikaväli valitsimeen jQuery-kirjastoa. Esitellään käytetyt teknologiat, jonka jälkeen käydään läpi web-sovelluksen rakenne ja sen kehittäminen.

### 3.3.1 Web-ohjelmoinnin perusosat

Jotta web-ohjelmoinnin voi aloittaa, ohjelmoijan täytyy tuntea web-sovellusten perusteknologiat; HTML, CSS ja JavaScript ovat ne peruspalikat, joilla web-ohjelmoiminen aloitetaan.

HTML, eli Hypertext Markup Language, on merkintäkieli, jolla luodaan web-sivustojen perusosat ja niiden sisältö. HTML koostuu elementeistä, joilla muodostetaan web-sivusto. CSS, eli Cascading Style Sheets, on tyylittelykieli, jolla muotoillaan HTML-elementtejä. CSS avulla pystytään muuttamaan HTML-elementtien kokoa, väriä ja muotoa. CSS kuuluu myös web-ohjelmoinnin peruskäsitteisiin. (MDN contributors 2021.)

JavaScript, useasti pelkkä JS, on ohjelmointikieli. JavaScript on dynaamisesti tyypitetty, eli tyyppi tarkistetaan ja päätetään ajon aikana. Tyypillä tarkoitetaan jonkin muuttujan tai funktion palautuksen tyyppiä, esimerkiksi numero tai merkkijono. Dynaamisen tyypityksen vastakohta on staattinen tyypitys, jossa tyyppi määritellään ennen ajoa. JavaScript on myös oliopohjainen ohjelmointikieli. Oliopohjaisella ohjelmoinnilla tarkoitetaan ohjelmointitapaa, jossa ongelmien ratkaisuun käytetään olioita. Olio (engl. object) on osa ohjelmistoa, joka sisältää ominaisuuksia ja metodeja. (Kuva 8)




Kuva 8. Auto-olio kuvana. (W3Schools, 2021a)

Kuvassa yhdeksän esitellään auto-oliolle kuuluvia ominaisuuksia ja metodeja. Metodit ovat oliolle kuuluvia toimintoja. Kaikilla auto-olioilla on samat ominaisuudet, mutta ominaisuuksien arvot vaihtelevat oliokohtaisesti. JavaScript:ia on yleisesti käytetty web-ohjelmoinnissa, mutta sitä voidaan hyödyntää myös palvelinpuolen ohjelmoinnissa, esimerkiksi Node.js-ajoympäristössä, pelikehityksessä, työpöytä- ja mobiilisovellusten kehittämisessä. (MDN contributors 2021.)

### 3.3.2 Käyttäjärajapinnan suunnittelu

Rajapinnan vaatimuksena oli helppokäyttöisyys ja yksinkertaisuus, joten käyttäjärajapintaan luotiin kaksi valitsinta, kalenterivalitsin datan aikavälille ja kaavio. Ensimmäisestä valitsimesta voidaan valita intervalli datapisteille, esimerkiksi minuutti tai puoli tuntia, toisesta valitsimesta valitaan laite, jota halutaan tarkastella ja kalenterista voidaan valita aikaväli datalle, esimerkiksi päivä tai viikko.

### 3.3.3 Valitsimet

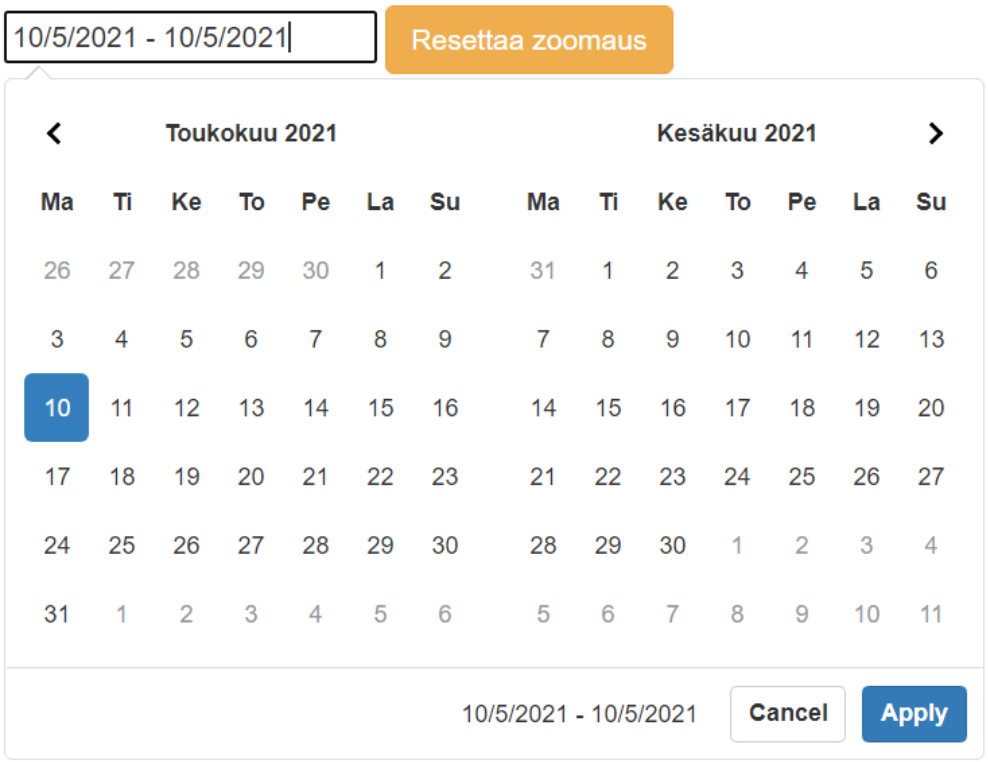
Valitsimet luotiin HTML-merkintäkielellä ja ne muotoiltiin CSS-tyylittelykielellä. Muotoiluun käytettiin Bootstrap:ia. Bootstrap on CSS-ohjelmistokehys jolla saadaan helposti ja nopeasti aikaseksi näyttäviä ja täysin responsiivisia HTML-elementtejä. Intervalli- ja laitevalitsimet ovat tehty samalla tavalla, käyttäen list-elementtiä. (Kuva 9) Lista-elementteihin on tallennettu funktioita ja arvoja, joilla saadaan valittua tietyt laitteet tai tietty aikaväli datapisteille. (Bootstrap team 2021.)

```
50
51  <div class="btn-group">
52    <button type="button" class="btn btn-warning dropdown-toggle" data-toggle="dropdown">
53      Intervalli
54    </button>
55    <div class="dropdown-menu">
56      <li onclick="minute()" id="minute"><a href="#">Minuutti</a></li>
57      <li onclick="halfHour()" id="half"><a href="#">Puoli tuntia</a></li>
58    </div>
59  </div>
60
61  <div class="btn-group">
62    <button type="button" class="btn btn-warning dropdown-toggle" data-toggle="dropdown">
63      Laitteet
64    </button>
65    <div class="dropdown-menu">
66      <li><a href="#">223</a></li>
67    </div>
68  </div>
```

Kuva 9. HTML-kielellä toteutetut valitsimet. Rivillä 51 määritetään div-elementti, joka sisältää valitsimen. Rivillä 52 on button-elementti, joka on muotoiltu Bootstrap:in avulla. Alasvetovalikko (dropdown menu) sisältää funktiot, joilla määritetään intervalli datapisteille. Alempi alasvetovalikko (rivit 61-68) on muuten samanlainen, mutta valikko sisältää arvon 223, jolla määritellään tarkasteltava laite.

Kalenterivalitsimeksi otettiin valmis JavaScript komponentti, Daterangepicker. Komponentti luo alasvetovalikon, johon aukeaa kalenteri, josta käyttäjä voi valita aikavälin. (Kuva 10) Komponentti on kehitetty jQuery:lla. (Daterangepicker www-sivut 2021.)

JQuery on JavaScript-kirjasto. Kirjaston tarkoituksen on tehdä JavaScriptin kirjoittamisesta helpompaa ja saada vähemmällä kirjoittamisella enemmän aikaan. JQuery sitoo monet JavaScript:in yleiset tehtävät ja funktiot yksinkertaisiin metodeihin, joita voidaan kutsua yhdellä rivillä koodia. JQuery siis yksinkertaistaa JavaScriptin käyttöä. (W3Schools 2021b)




Kuva 10. Näkymä kalenterivalitsimesta. Käyttäjän valitsema aikaväli sijoitetaan ChartJs-kaavion X-akselille.

### 3.3.4 Kaavio

Valitsimien luonnin jälkeen kaavio tarvitsee pohjan. Pohjaksi riittää kaavion kokoa säätelevä div-elementti ja canvas-elementti johon kaavio piirretään. (Kuva 11)

```
<div id="chartContainer">
  <canvas id="chart"></canvas>
</div>
```

Kuva 11. Tarvittavat HTML-elementit.

Div-elementti, eli HTML Content Division element, on HTML peruselementti, jolla jaetaan web-sovellus osiin. Tässä tapauksessa div-elementti auttaa säätelemään kaavion kokoa. Canvas-elementti toimii alustana 2D ja 3D muotojen ja bittikarttakuvien renderöintiin ja muotoiluun. Canvas-elementtiä tarvitaan itse kaavion piirtämiseen. (MDN contributors 2021d.)

Kaavioon tarvitaan dataa. Axios.get-funktiolla saadaan kyseltyä edellä luodulta Node.js-palvelimen isännöimältä API:lta dataa käyttöliittymään. (Kuva 12)

```
axios.get('https://apiAddress/apiRoute')
    .then(response => {
        console.log(response);
    });
```

Kuva 12. Axios-kirjaston avulla toteutettu datan haku.

Axios.get-komennolle annetaan parametriksi osoite, josta data halutaan hakea. Axios palauttaa vastauksen kyseltävän kohteen Content-Type header:in mukaan. Tässä tapauksessa se on määriteltynä palvelimella application/json:ksi, jolloin axios osaa palauttaa vastauksen JSON-muodossa.

Kun pyyntöön on vastattu, tallennetaan data muuttujaan, tässä tapauksessa muuttujan nimi on pacRange, lisäkäsittelyä varten. Luotu muuttuja lisätään kaavion dataset-objektiin. (Kuva 13) Muuttujan pacRange lisäksi datasetille annetaan arvoja, joilla muotoillaan viivakaaviota.

```
var oneChartDataset1 = {
    label: "PAC",
    fill: false,
    lineTension: 0,
    backgroundColor: "rgba(0,250,0)",
    borderColor: "rgba(30,250,50)",
    borderCapStyle: 'butt',
    borderDash: [],
    borderDashOffset: 0.0,
    borderJointStyle: 'round',
    data: pacRange,
};
```

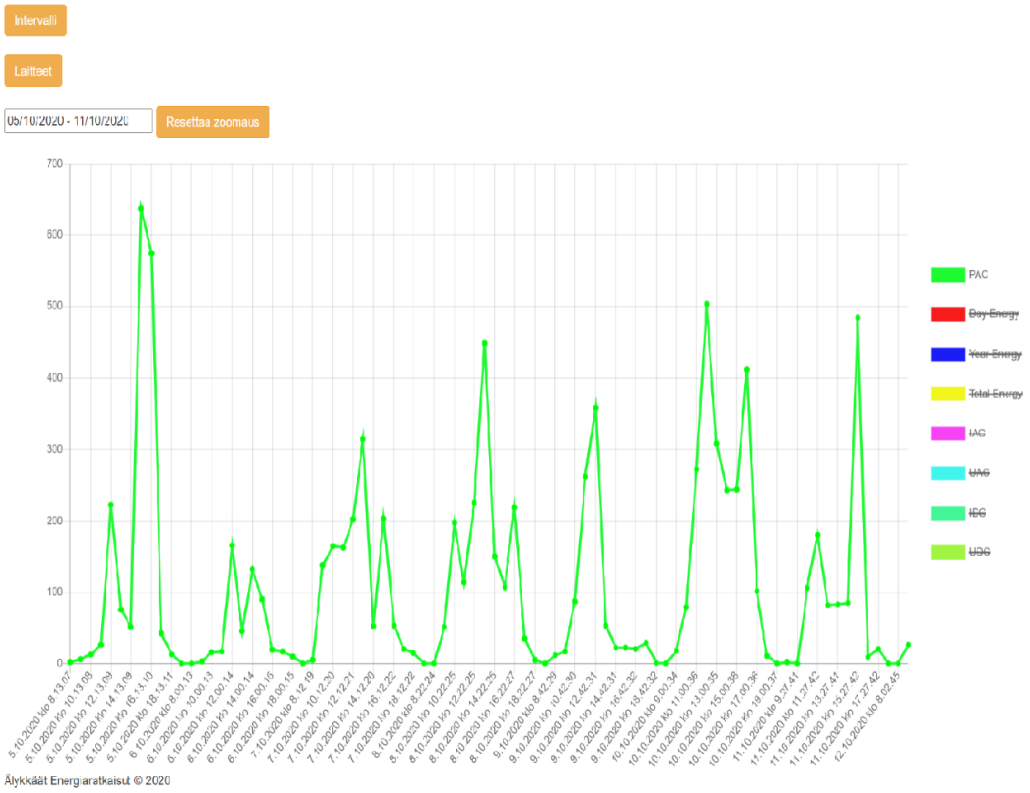
Kuva 13. Kuvan oneChartDataset1-objekti sisältää muuttujan pacRange. Muuttuja pacRange sisältää API:sta haetun datan.

Kun kaavioon tarvittava data on valmiina, tarvitaan itse kaavio. Tässä tapauksessa kaavio on muodostettu funktiossa (Kuva 14), kaavion kuitenkin pystyy piirtämään myös muuttujien avulla. Kaavion piirtämisen funktiossa etuna on se, että kaavio piirtyy vasta kun funktio on suoritettu. Muuten kaavio piirtyy heti sivuston avautuessa. Piirtofunktioon määritellään myös asetuksia muuttujassa oneChartOptions. Asetusmuuttuja sisältää erilaisia tyylittelyjä ja lisäosia, kuten selitteen sijoittelu ja kartan suurennus (zoom) lisäosa.

```
1  var canvas = document.getElementById("oneChart");
2  var ctx2 = document.getElementById('oneChart').getContext('2d');
3
4  function drawOneChart() {
5
6      window.oneChart = new Chart(ctx2, { type: 'line', data: oneChartData1, options: oneChartOptions })
7
8  };
```

Kuva 14. Rivillä 1 ja 2 määritellään HTML-elementit JavaScript-muuttujiksi, jotta niiden käsittely olisi helpompaa. Rivillä neljä luodaan funktio drawOneChart. Rivillä kuusi määritellään funktion toiminta, eli piirretään new Chart, jonka parametreina on kaavion tyyppi, data, joka määriteltiin aiemmin ja options, jonka arvona muuttuja oneChartOptions.

Kun edellä mainitut toimenpiteet ovat suoritettu, kaavio piirtyy sivustolle (Kuva 15). Sivuston vasemmassa yläreunassa näkyy valitsin Intervalli, jolla määritellään datapisteiden tiheys ja valitsin Laitteet, josta voidaan valita eri muuntimia ja laitteita, joiden dataa tarkastellaan. Alimmaisena on kalenterivalitsin, josta valitaan päivämäärät, joita halutaan tarkastella. Kaaviossa on myös tarkennustoiminto, jonka pystyy palauttamaan alkutilaan napsauttamalla Resettaa zoomaus -painiketta. Painikkeet on muotoiltu Bootstrap-kirjastolla.




Kuva 15. Kaavio piirrettynä sivustolle. Kaavio näyttää muuntimen tehon viikon ajalta.

Kaavion oikealta puolelta voidaan valita tarkasteltava data, muun muassa kaaviossa näkyvä hetkellinen teho, päivän kokonaistuotto ja vuoden kokonaistuotto. Kaaviota tarkastellessa pystytään päättelemään, mihin kellonaikaan aurinkoenergiaa on kertynyt eniten, eli milloin aurinko on paistanut ja milloin ei.

# 4 YHTEENVETO

Vaatimuksina web-sovellukselle oli datan luettavuus yhdestä selkeästi paikasta. Web-sovellus onnistuu vastaamaan vaatimuksiin hyvin. Kaavion avulla pystytään tarkastelemaan monenlaista dataa monista eri muuntimista. Jo yhdellä silmäyksellä näkee pääpiirteittäin mitä muuntimella tapahtuu. Myös aikavälin valinta osuus onnistui hyvin, vaikka JavaScriptin päiväyksiin liittyvät toiminnot ovat yleisesti todettu melko hankaliksi toteuttaa.

Käytetyt modernit teknologiat vastaavat hyvin nykyajan vaatimuksia, ja mahdollinen jatkokehitys on dokumentaation puolesta erittäin mahdollista. Työhön on mahdollista lisätä tulevia aurinkosähköjärjestelmiä ja niiden datan keruuohjelmistoja ja tallentaa samaan paikkaan kuin nykyisetkin.

Jatkokehityksenä sovelluksen palvelinpuolta voisi parannella. Sovellus on tällä hetkellä melko hidas, sillä käyttäjäpuoli hakee todella suuren määrän dataa, mikä ei ole suotuisin vaihtoehto. Datan voisi jo palvelinpuolella jaotella pienempiin osiin, joita käyttäjäpuoli voisi kysellä.

Työtä tehdessä vastaan tuli paljon asioita ja teknologioita, joista en ollut aiemmin kuulutkaan, kuten esimerkiksi Modbus-protokolla. Ennalta tutut teknologiat olivat webohjelmoinnin perustyökalut, joten opiskeluun ja uusien asioiden haltuun ottamiseen täytyi käyttää aikaa. Ajankäyttö kuitenkin maksoi itsensä takaisin, ja tunnen itseni nyt sovelluksen kehittäneenä olevani paljon valmiimpi ohjelmoijana, kuin työn aloittaessani. Työn aikana opin myös ohjelmistokehitys työhön kuluvan ajan suunnittelusta

ja käytöstä paljon. Monta kertaa toiminnon tai ohjelmiston osan, jonka kehittämiseen kuvittelin kuluvan aikaa muutaman tunnin, saattoi venyä muutaman päivän mittaiseksi työksi.

Kaiken kaikkiaan työ on ollut hyvin opettavainen ja mielenkiintoinen ja olen tyytyväinen lopputulemaan.

# LÄHTEET


Axios contributors. 2021. Getting Started. Viitattu 10.5.2021.

https://axios-http.com/docs/intro

Bootstrap team. 2021. Build fast, responsive site with Bootstrap. Viitattu 12.1.2021.

https://getbootstrap.com/

Chart.js contributors. 2021. Chart.js. Viitattu 6.5.2021.

https://www.chartjs.org/

MDN contributors. 2020a. Introduction to web APIs.

https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Client-side_web_APIs/Introduction

MDN contributors. 2021b. HTML: HyperText Markup Language. Viitattu 6.5.2021.

https://developer.mozilla.org/en-US/docs/web/HTML

MDN contributors. 2021c. About JavaScript. Viitattu 6.5.2021.

https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript

MDN contributors. 2021d. Canvas API. Viitattu 10.5.2021.

https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API

MongoDB contributors. 2021. Introduction to MongoDB. Viitattu 11.5.2021.

https://docs.mongodb.com/manual/introduction/

Node.js contributors. 2021. About Node.js.

https://nodejs.org/en/about/

Schneider electronics www-sivut. 2021. What is Modbus and How does it Work.

https://www.se.com/us/en/faqs/FA168406/

Tableau. 2021. Data visualization beginner's guide: a definition, examples, and learning resources. Viitattu 11.5.2021.

https://www.tableau.com/learn/articles/data-visualization

Tutorialspoint. 2021. Node.js – Introduction. Viitattu 11.5.2021.

https://www.tutorialspoint.com/nodejs/nodejs_introduction.htm

W3Schools. 2021a. JavaScript Objects. Viitattu 7.5.2021.

https://www.w3schools.com/js/js_objects.asp

W3Schools. 2021b. jQuery introduction. Viitattu 10.5.2021.

https://www.w3schools.com/jquery/jquery_intro.asp