TAUSTAJÄRJESTELMÄ AURINKOSÄHKÖVOIMALAN TUOTTAMAN DATAN VISUALISOIMISEEN

LAB-AMMATTIKORKEAKOULU
Tieto- ja viestintätekniikka (AMK)
Ohjelmistotekniikka
Syksy 2021
Otto Heikkinen

Tiivistelmä

Tekijä(t)	Julkaisun laji	Valmistumisaika
Heikkinen, Otto	Opinnäytetyö, AMK	Syksy 2021
	Sivumäärä 31	

Työn nimi

Taustajärjestelmä aurinkosähkövoimalan tuottaman datan visualisoimiseen

Tutkinto

Tieto- ja viestintätekniikka (AMK)

Tiivistelmä

Opinnäytetyössä toteutettiin taustajärjestelmä aurinkosähkövoimalan tuottaman energian visualisoimiseen. Työ toimii taustajärjestelmänä infonäyttö-sovellukselle, joka tulee Lem-Kem Oy:lle osaksi yrityksen myymiin avaimet käteen -aurinkosähköjärjestelmiin.

Aurinkosähköjärjestelmissä on Froniuksen Symo invertterit, joihin lisättävillä Datamanager-verkkokorteilla voimaloiden tuottama data lähetetään palvelimelle ja tallennetaan tietokantaan. Palvelimelle asennetaan node.js, jolla toteutetaan REST API -rajapinta käyttäen express.js-kirjastoa.

Rajapinnasta täytyy saada järjestelmien tuottama data haettua helposti visualisoitavassa muodossa. Voimalan tuotannon huipputehoa kuvataan infonäyttö-sovelluksessa päiväkohtaisesti 5 minuutin välein sekä kokonaistuotantoa päivä- ja kuukausikohtaisesti.

Taustajärjestelmä toimii usealle voimalalle, joten datan hakemiseen taustajärjestelmästä toteutetaan autentikointi. Kirjautumalla taustajärjestelmään käyttäjä autentikoidaan vain oman voimalansa dataan.

Lopputuloksena oli toimiva taustajärjestelmä. Aurinkovoimaloiden tuottama data tuotannosta saatiin tallennettua tietokantaan ja autentikoitu käyttäjä sai datan haettua rajapinnan kautta selkeästi JSON-muodossa, josta sen sai infonäyttö-sovelluksessa piirrettyä graafiin.

Asiasanat

taustajärjestelmä, aurinkosähkövoimala, Node.js, REST, API

Abstract

Author(s)	Type of publication	Published
Heikkinen, Otto	Bachelor's thesis	Autumn 2021
	Number of pages 31	

Title of publication

Background service for visualizing energy generated by a photovoltaic system

Name of Degree

Bachelor's Degree Programme in Information and Communications Technology

Abstract

The objective of this thesis was to implement a background service for visualizing energy generated by a photovoltaic system. This thesis is background service for InfoDisplay -application, which is going to be a part of photovoltaic systems sold by LemKem Oy.

Photovoltaic systems have Fronius Symo inverters to which the Datamanager communications centre added. The data generated by system can be sent to a server and stored in a database. Server is running a REST API built by Node.js and Express.js

The data generated by the system must be provided from the API in an easily visualizable form. On daily basis the maximum power output of the system is displayed in an InfoDisplay application every 5 minutes. Total production is displayed daily and monthly.

This thesis background service is for several photovoltaic system, so authentication is needed to retrieve data from the API. Login into the background system authenticates the user only for the data of her own system.

Final result of this thesis was a working background service. The data produced by the photovoltaic systems production was stored in the database and an authenticated user was able to retrieve the data clearly in JSON format.

Keywords

Background service, photovoltaic system, Node.js, REST, API

SISÄLLYS

1 JOHDANTO 1
2 AURINKOSÄHKÖVOIMALA 2
2.1 Aurinkosähkövoimalan toimintaperiaate 2
2.2 Fronius Symo 2
2.3 Datamanager 3
3 MYSQL 5
3.1 Yleisesti 5
3.2 ORM 5
3.3 Sequelize 5
3.3.1 Taulun mallintaminen 6
3.3.2 Kyselyt 7
4 NODE.JS 9
4.1 Yleisesti 9
4.2 Node Package Manager 9
4.3 Express 9
5 REST 12
5.1 Mitä REST tarkoittaa 12
5.2 Asiakas–palvelin 12
5.3 Tilattomuus 12
5.4 Välimuisti 12
5.5 Yhtenäinen rajapinta 12
5.6 Kerroksittainen järjestelmä 13
6 TAUSTAJÄRJESTELMÄN TOTEUTUS 14
6.1 Tavoite 14
6.2 Palvelimen asennus 14
6.3 Rajapinta 17
6.3.1 Kansiorakenne 18
6.3.2 Taulujen mallintaminen 18
6.3.3 Autentikointi 21
6.3.4 Datan vastaanottaminen 22
6.3.5 Päivä ja kuukauden kokonaistuotanto 23
6.3.6 Datan hakeminen rajapinnasta 24
6.4 Datamanagerin asentaminen ja tiedon lähettäminen 26

7 YHTEENVETO 28
LÄHTEET 29

LYHENTEET

API	Ohjelmointirajapinta (Application Programming Interface)
HTTP	Hypertekstin siirtoprotokolla (Hypertext Transfer Protocol)
IP	IP-protokolla (Internet Protocol)
JSON	JavaScript Object Notation
kWp	Aurinkosähköjärjestelmän nimellistehon yksikkö (kilowattipeak), 1 kWp vastaa Suomessa etelään suunnatuilla aurinkokennoilla noin 850 kilowattituntia vuodessa
NPM	Node Package Manager
ORM	Object-relational mapping
REST	Representional State Transfer

1 JOHDANTO

Suomessa arvio uusiutuvan energian osuudesta loppukulutuksesta vuoteen 2030 mennessä on 50 prosenttia (Työ- ja elinkeinoministeriö 2016). Vuonna 2017 sähköverkkoon liitetyn aurinkosähkön pientuotannon kapasiteetti oli 66 MW. Vuodessa se kasvoi 82 prosentilla 120 MW kapasiteettiin. Sähköverkkoon liitetyn pientuotannon osuus Suomen koko tuotantokapasiteetista on yli prosentin, josta 60 prosentin osuus on aurinkosähköä. (Energiavirasto 2019.)

Lem-kem Oy on vuonna 1985 perustettu perheyritys, joka palvelee yli 23 000 asiakasta valaistuksen, kiinteistönhoidon sekä teollisuuden kunnossapidon tuotteilla. Liikevaihto vuonna 2017 oli 12,7 milj. euroa, ja se työllistää 52 henkilöä sekä 30 alue-edustajaa. Vuonna 2016 perustettu Lem-Kem Energiatiimi keskittyy kiinteistöjen energiatehokkuuteen. Energiatiimi kartoittaa kiinteistön nykytilanteen ja toimittaa energiaa säästävät energiaratkaisut asennuksineen asiakkaalle. Ratkaisut energian säästöön kiinteistössä ovat: nykyaikaisen valaistuksen käyttö, sekä aurinkosähkön ja maalämmön hyväksikäyttö. (Lem Kem Oy 2020.)

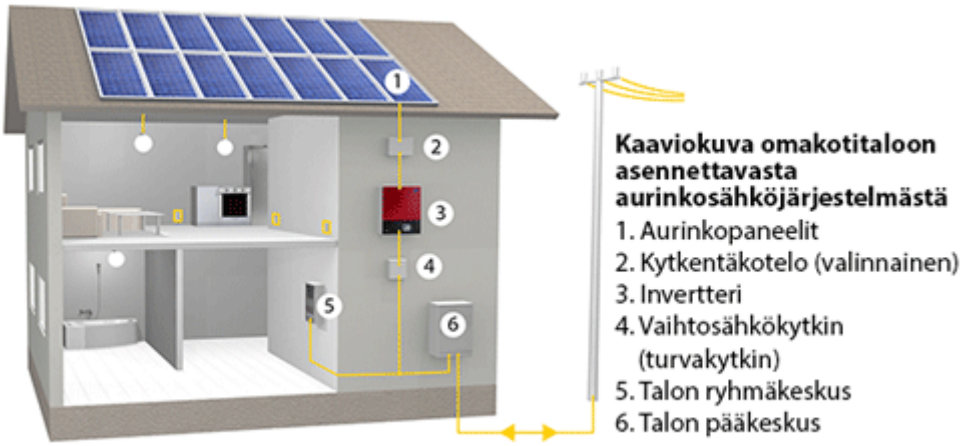
Opintäytetyön tavoitteena on rakentaa taustajärjestelmä aurinkosähköjärjestelmän tuottaman energian visualisoimiseen. Sovellus tulee toimiminaan taustajärjestelmänä infonäyttö-sovellukselle, josta tulee myytävä tuote lisänä Lem-Kemin myymiin avaimet käteen -aurinkosähköjärjestelmiin. Infonäytöt tullaan asentamaan kevään 2020 aikana kaikkiin Hämeenmaan osuuskauppoihin, joihin on aurinkovoimala asennettu. Asennettujen voimaloiden huipputehot ovat 70 - 500 kWp. Kokonaisuudessaan voimaloiden teho on 2400 kWp, mikä pienentää Hämeenmaan hiilijalanjälkeä 320 000 kilogrammaa vuodessa.

Tässä opinnäytetyössä selvitetään, kuinka Froniuksen Symo invertereiltä saadaan aurinkokennojen tuottama data lähetettyä omalle palvelimelle, jossa data tallennetaan tietokantaan. Lisäksi rakennetaan REST-arkkitehtuurinmallin mukainen rajapinta, josta data saadaan haettua halutussa muodossa. Rajapinnasta haettua dataa pitää pystyä visualisoimaan reaaliajassa päivä-, viikko- ja kuukausikohtaisesti.

2 AURINKOSÄHKÖVOIMALA

2.1 Aurinkosähkövoimalan toimintaperiaate

Aurinkosähkövoimalan toiminta perustuu auringon säteilyenergiaan. Auringon säteily koostuu fotoneista, jotka kuljettavat auringon säteilyenergian. Aurinkokennot on valmistettu puolijohdemateriaalista. Yleisin materiaali kennojen valmistamiseen on pii, jota käytetään joko yksi- tai monikiteisenä. Auringon säteiden osuessa aurinkokennoihin ne luovat varauksen puolijohteisiin. Varaus kulkee johtimia pitkin invertterille, joka muuntaa tasasähkön vaihtosähköksi. Invertterin mallista ja koosta riippuen se kytketään kiinteistön 3-vaiheverkon kaikkiin kolmeen vaiheeseen tai vain yhteen vaiheeseen, josta tuotettu sähkö menee laitteiden käytettäväksi. Jos vaiheella ei ole kuormaa ollenkaan, niin sähkö syötetään pääkeskuksen sähkömittarin kautta ulkopuoliseen sähköverkkoon. Kuvassa 1 on esitetty omakotitalon katolle asennetun aurinkosähkövoimalan toiminta. (Motiva 2017.)



Kuva 1. Aurinkosähkövoimalan toiminta(Salo Solar Oy 2020)

2.2 Fronius Symo

Fronius International GmbH on itävaltalaisen Günther Froniuksen perustama yritys 1950-luvulla, joka on erikoistunut auton akkujen lataukseen. Myöhemmin yritys alkoi valmistamaan elektroniikkaa ja hitsauskomponentteja. Vuonna 1992 yritys laajensi aurinkoenergiaan. Nykyään Fronius jakautuu kolmeen sektioon: hitsaukseen, aurinkoenergiaan sekä akkujen lataukseen. (Fronius International GmbH 2020.)

Fronius Symo (Kuva 2) on Froniuksen vuonna 2013 lanseerattu invertteri aurinkokennojen tuottaman sähköenergian muuntamiseksi kiinteistön 3-vaiheiseen sähköverkkoon. Eri tehoisia invertteriä on huipputeholtaan 3 kilowatista 20 kilowattiin. M-merkinnällä varustetut invertterit sisältävät kaksi lataussäädintä, mikä mahdollistaa kahden

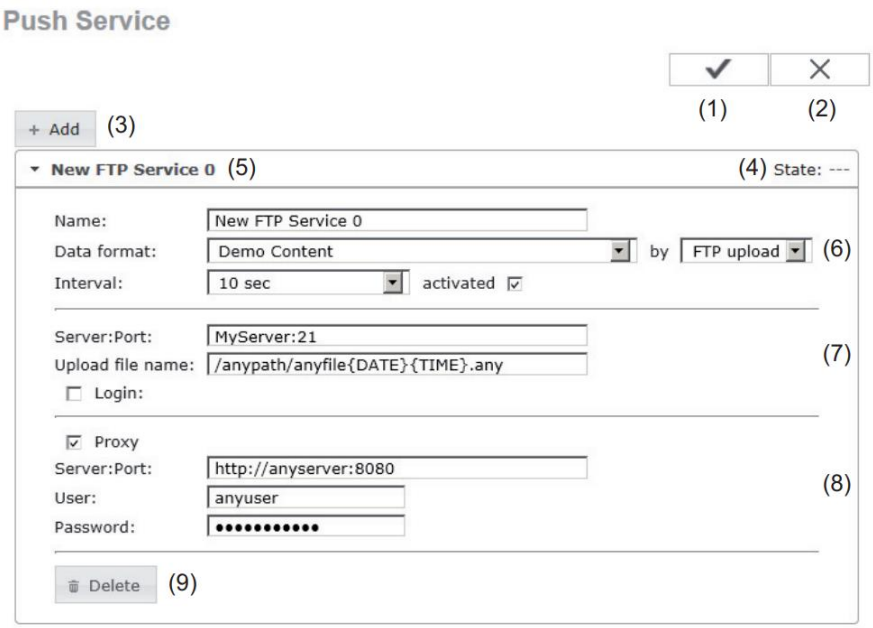
aurinkokennoketjun kytkemisen. Kahden ketjun yhdistämisen hyöty tulee siitä, että toinen ketju jatkaa tuottamista, vaikka toinen ketju olisi varjossa. Jos talon harja on itä-länsi suuntainen, niin toinen ketju asennetaan talon itäpuolelle keräämään aamupäivän auringon energian ja toinen ketju talon länsipuolelle keräämään iltapäivän auringon energian. (Fronius International GmbH 2020.)

Kuva 2. Fronius Symo invertteri (Fronius International GmbH 2020)

2.3 Datamanager

Fronius Symon saa kytkettyä internetiin siihen lisättävällä Datamanager-verkkokortilla joko langattomaan lähiverkkoon tai lähiverkkoon Ethernet-kaapelilla. Kun laite on yhdistetty internetiin se lähettää aurinkosähkövoimalan tietoja suoraan Froniuksen omaan Solar.web-portaaliin, joka tarjoaa käyttäjälle jatkuvan yleiskatsauksen järjestelmän toiminnasta ja tuotannosta. Optimaalisen järjestelmän seurannan ja Datamanagerin konfiguroimisen voi tehdä Datamanagerin integroidun verkkopalvelimen web-käyttöliittymässä. Datamanagerin voi yhdistää rinnakkain Solar.web-portaaliin ja kolmannen osapuolen järjestelmien kanssa integroiduilla Modbus RTU Sunspec-, Modbus TCP Sunspec-, Fronius API- ja Pushservice-rajapinnoilla. (Fronius International GmbH 2020.)

Pushservicen määrittäminen tapahtuu Datamanagerin web-käyttöliittymässä asetusvalikossa kohdasta Pushservice (Kuva 3). Pushservicen avulla voidaan lähettää järjestelmän dataa invertteriltä halutulle palvelimelle. Erilaisia dataformaatteja on saatavilla XML ja JSON. Tuetut protokollat ovat http POST ja FTP. Palveluita voi määritellä kymmenen kappaletta, ja jokainen palvelu voi käyttää eri protokollia, dataformaatteja ja ulkoisia palvelimia.



Kuva 3. Pushservicen käyttöliittymä

3 MYSQL

3.1 Yleisesti

MySQL on ruotsalaisen MySQL AB-ohjelmistoyrityksen vuonna 1995 julkaistu avoimen lähdekoodin tietokannan hallintajärjestelmä (Wikipedia 2019). Se on suosituin avoimen lähdekoodin tietokannan hallintajärjestelmä, jota käyttää muun muassa Facebook, Twitter ja Youtube. MySQL käyttää GPL (GNU General Public License) -lisenssiä, jolloin sitä voi käyttää ilman maksua.

SQL (Structured Query Language) on yleisin standardisointunut kieli tietokannan käsittelyyn. SQL on määritelty ANSI/ISO SQL -standardin mukaan. SQL-standardi on kehittynyt vuodesta 1986 lähtien, ja siitä on useampi versio olemassa. Ohjelmointiympäristöstä riippuen SQL:ää syötetään suoraan, sulautettuna SQL-lauseena toiseen ohjelmointikieleen, tai käyttäen ohjelmointirajapintaa, joka piilottaa sisäänsä SQL-syntaksin. (Oracle 2020a.)

MySQL-tietokannat ovat relaationaalisia. Relaatiotietokannassa data tallennetaan ennemmin useaan erilliseen tietokantatauluun, kuin yhteen isoon tauluun, jossa on kaikki. Loogiset mallit olioineen, kuten tietokannat, taulut, näkymät, rivit ja sarakkeet, tarjoavat joustavan ohjelmointiympäristön, jossa määritellään sääntöjä, jotka ohjaavat eri tietokenttien välisiä suhteita. (Oracle 2020b.)

3.2 ORM

Lyhenne ORM tulee sanoista Object-relational mapping. Se on useimmiten kirjasto, joka hoitaa datamuunnokset ohjelman kielen ja tietokannan välillä. Sen tarkoitus on käyttää relaatiotietokantaa ja tehdä kyselyitä datan tallentamiseen, hakemiseen, päivittämiseen ja poistamiseen olio-ohjelmointikielellä. Relaatiotietokannat käyttävät datan tallentamiseen datatauluja, jotka koostuvat riveistä ja sarakkeista. ORM muuntaa datan olioista sopivaksi tietokantaan ja toisinpäin. SQL on tehokas ja monipuolinen kieli, mutta jokseenkin vaikea kehittäjälle, joka ei ole perehtynyt siihen kunnolla. ORM:n avulla voidaan tehdä samat kyselyt käyttäen sitä tuttua kieltä, jota on tottunut käyttämään. (Hoyos 2018.)

3.3 Sequelize

Sequelize on lupauksiin (promise) perustuva Node.js ORM -työkalu seuraaville tietokannoille: Postgres, MySQL, MariaDB, SQLite ja Microsoft SQL Server. Versio 5 julkaistiin maaliskuussa 2019, joka sisältää viralliset TypeScript-tyypitykset. (Sequelize 2020.)

Sequelize-kirjasto asennetaan joko npm- tai yarn-paketinhallintatyökalua käyttämällä. Tietokantaan yhdistäminen tapahtuu luomalla Sequelizesta instanssi (Kuva 4). Instanssia luotaessa sille täytyy antaa parametreina host, käyttäjänimi, salasana, tietokannan nimi ja tietokannan tyyppi. Parametrit voi antaa joko muodostimessa tai antamalla sille yhden rivin merkkijono, joka sisältää kaikki parametrit. (Sequelize 2020.)

```
// Instanssin luominen muodostimessa
const sequelize = new Sequelize('tietokannanNimi',
  'käyttäjänimi', 'salasana', {
  host: 'localhost',
  dialect: 'mysql'
});

// Instanssin luominen yhden rivin merkkijonolla
const sequelize = new Sequelize(
  'mysql://user:esim@esimerkki.com:5432/tietokannanNimi'
);
```

Kuva 4. Sequelize instanssin luominen

3.3.1 Taulun mallintaminen

Mallin luominen tapahtuu luomalla lapsi luokasta Sequelize.Model. Mallin luomiseen on kaksi oikeaa tapaa. Ensimmäinen on Sequelize.Model.Init-metodia kutsumalla. Init-metodille annetaan oliona tietokantataulun attribuutit ja asetukset. Attribuuttiolio sisältää taulun sarakkeet ja niiden tietotyypit. Toinen tapa on kutsua metodia sequelize.define. Define-metodia kutsuttaessa se kutsuu sisäisesti Model.Init-metodia. Attribuutteja annettaessa jokaiselle sarakkeelle täytyy antaa tietotyyppi (Kuva 5).

	Sequelizen tietotyyppi	Tietokannan tietotyyppi
Merkkijono	DataTypes.STRING // VARCHAR(255)	VARCHAR(255)
	DataTypes.STRING(1234)	VARCHAR(1234)
	DataTypes.STRING.BINARY	VARCHAR BINARY
	DataTypes.TEXT	TEXT
	DataTypes.TEXT('tiny')	TINYTEXT
	DataTypes.CITEXT	CITEXT PostgreSQL and SQLite only.
Numerot	DataTypes.INTEGER	INTEGER
	DataTypes.BIGINT	BIGINT
	DataTypes.BIGINT(11)	BIGINT(11)
	DataTypes.FLOAT	FLOAT
	DataTypes.FLOAT(11)	FLOAT(11)
	DataTypes.FLOAT(11, 10)	FLOAT(11,10)
	DataTypes.REAL	REAL PostgreSQL only.
	DataTypes.REAL(11)	REAL(11) PostgreSQL only.
	DataTypes.REAL(11, 12)	REAL(11,12) PostgreSQL only.
	DataTypes.DOUBLE	DOUBLE
	DataTypes.DOUBLE(11)	DOUBLE(11)
	DataTypes.DOUBLE(11, 10)	DOUBLE(11,10)
	DataTypes.DECIMAL	DECIMA
	DataTypes.DECIMAL(10, 2)	DECIMAL(10,2)
päivämäärä	DataTypes.DATE	DATETIME for mysql / sqlite, TIMESTAMP WITH TIME ZONE for postgres
	DataTypes.DATE(6)	DATETIME(6) for mysql 5.6.4+. Fractional seconds support with up to 6 digits of precision
	DataTypes.DATEONLY	DATE without time
Tottusarvomuuttuja	DataTypes.BOOLEAN	TINYINT(1)

Kuva 5. Tietotyypit

Sequelize etsii tietokannasta automaattisesti annetun taulunnimen monikkomuotoa, jos asetusta ei erikseen poisteta. Esim. taulun nimeksi annettaessa user. Sequelize etsii tietokannasta taulua nimeltä users. Sequelize luo myös automaattisesti tauluun aikaleima-sarakkeet luontiajalle sekä muokkausajalle. Mallien synkronointi luo taulut tietokantaan, jos tauluja ei ole vielä luotu. Synkronointi suoritetaan kutsumalla Sequelizen metodia sync(). Se synkronoi automaattisesti kaikki luodut mallit. (Sequelize 2020.)

3.3.2 Kyselyt

Kyselyt tehdään kutsumalla luodun mallin metodeja. Seguelizessa metodeja kyselyihin on findOne, findAll, create, destroy, update. Tehtäessä kyselyä findAll- ja findOne-metodeilla sekä tuhottaessa tai päivittäessä useampaa merkintää voidaan käyttää where-objektia kyselyn tulosten tarkentamiseen. Where-objekti koostuu yleensä atribuutti–arvo -pareista, mutta on myös mahdollista käyttää JA/TAI-operaattoreita. Kuvassa 6 on kaksi esimerkkiä where-objektin käytöstä. Ensimmäisessä esimerkissä kyselyllä haetaan kaikki rivit tietokantataulusta, joilla authorId on yhtä kuin 12 ja status on yhtä kuin ”active”. Toisessa esimerkissä haetaan kaikki rivit, joilla authorId on joko 12 tai 13. (Sequelize 2020.)

```
Post.findAll({
  where: {
    authorId: 12,
    status: 'active'
  }
});
// SELECT * FROM post WHERE authorId = 12 AND status = 'active';

Post.findAll({
  where: {
    [Op.or]: [{authorId: 12}, {authorId: 13}]
  }
});
// SELECT * FROM post WHERE authorId = 12 OR authorId = 13;
```

Kuva 6. Where-objektin käyttö kyselyissä (Sequelize 2020)

Uuden merkinnän luominen tietokantatauluun tapahtuu kutsumalla luodun mallin create-metodia (Kuva 7). Sille annetaan attribuuttina olio, jonka sisältämä data merkintään halutaan lisätä. (Sequelize 2020.)

```
User.create({
  firstName: "Jane",
  lastName: "Doe"
});
```

Kuva 7. Merkinnän tekeminen Sequelizella

4 NODE.JS

4.1 Yleisesti

Ennen kaikki JavaScript-koodi suoritettiin selaimessa, mikä kuormittaa selainta ja hidastaa selaimen käyttökokemusta. Vuonna 2009 julkistettiin Node.js jonka avulla JavaScriptin suorittaminen voidaan tehdä palvelimella. Node.js on ohjelmointikehys JavaScript-applikaatioiden kirjoittamiseen palvelinpuolelle. Se on rakennettu Google Chromen V8 JavaScript-moottorin päälle. Se on suunniteltu skaalautuvien web-applikaatioiden luomiseen sen asynkronisen tapahtumapohjaisen ajon myötä. Node.js on usein käytetty taustajärjestelmien luomiseen, ja se keskustelee asiakassovelluksen kanssa ohjelmointirajapinnan kautta. (Henderson 2019)

Vaikka Node.js suorittaa tapahtumat yksisäikeisessä silmukassa ja voi suorittaa vain yhtä tapahtumaa kerrallaan, se käyttää tapahtumapohjaista esteetöntä (Non-Blocking) I/O-mallia. Esteetön I/O-malli mahdollistaa, että Node.js voi suorittaa tapahtumia rinnakkaisesti. Esim. Jos node.js lähettää kyselyn tietokantaan, se ei jää odottamaan vastausta, vaan alkaa suorittamaan seuraavaa tehtävää. Tämä mahdollistaa tuhansien yhteyksien hallitsemisen yhdellä palvelimella. (node.js 2020a.)

4.2 Node Package Manager

Npm eli Node Package Manager on Node.js:n paketinhallintatyökalu, jota ajetaan komentoriviltä. Npm luotiin alun perin Node.js-pakettien lataamiseen ja riippuvuussuhteiden hallintaan ja nykyään sitä voi käyttää myös versionhallintaan ja komentojen tekemiseen.

Jos projektilla on package.json-tiedosto, se pitää sisällään listan npm-paketeista, joista projekti on riippuvainen. Npm asentaa kaikki projektiin tarvittavat paketit ajamalla komennon ”npm install”. Uusien pakettien asentaminen tehdään lisäämällä komennon perään paketin asennettavan paketin nimi. (node.js 2020b.)

4.3 Express

Express.js on ohjelmointikehys Node.js:lle. Se on suunniteltu rajapintojen ja web-sovellusten luontiin tarjoten kokoelman ominaisuuksia. Expressillä on minimaalinen käyttöliittymä, mutta se sisältää kaikki tarvittavat työkalut applikaatioiden luontiin. Sovelluksiin on myös lisättävissä laaja valikoima moduuleja Node.js:n paketinhallintatyökalun avulla.

Expressin yksi tärkeimmistä ominaisuuksista on väliohjelmistot (middleware). Väliohjelmisto-funktioilla pääsee käsiksi rajapintaan tulleiden kutsujen kiertokulkuun, jotka voidaan

suorittaa kutsun tullessa rajapintaan. Väliohjelmisto pääsee käsiksi kutsun pyyntö (request) -objektiin sekä vastaus (response) -objektiin. Väliohjelmisto voi tehdä muutoksia näihin objekteihin, lähettää kutsun takaisin kysyjälle ja kutsua uuden väliohjelmiston.

Expressillä voi käyttää erityyppisiä väliohjelmistoja: applikaatiotason, reititintason, virheiden käsittelyn, sisään rakennettuja ja kolmannen osapuolen. Applikaatio- ja reititintason väliohjelmisto toimivat muuten samalla tavalla, mutta applikaatiotason väliohjelmistot liitetään Expressistä luotuun instanssiin, kun taas reititintason väliohjelmisto liitetään reitittimestä luotuun instanssiin. Alla olevassa kuvassa (Kuva 8) väliohjelmisto suoritetaan aina, kun applikaatiolle lähetetään kutsu.

```
var app = express()

app.use(function (req, res, next) {
  console.log('Time:', Date.now())
  next()
})
```

Kuva 8. Applikaatiotason väliohjelmisto (expressjs.com 2020c.)

Väliohjelmistot voidaan applikaatiotasolla myös sitoa tiettyyn polkuun. Väliohjelmisto suoritetaan vain niille kutsuille, jotka lähetetään tiettyyn polkuun (Kuva 9).

```
app.use('/user/:id', function (req, res, next) {
  console.log('Request Type:', req.method)
  next()
})
```

Kuva 9. Polkuun sidottu väliohjelmisto (expresjs.com 2020c.)

Expressillä on sisään rakennettuja väliohjelmistofunktioita. Kun express 4.x julkaistiin, monet sisään rakennetut väliohjelmistot siirrettiin erillisiin paketteihin, mutta käyttäjien pyynnöstä osa taas palautettiin 4.16.0-version julkaisussa (expresjs.com 2020). Expressillä uusimmassa versiossa on sisään rakennettuja väliohjelmistoja: express.static staattisten tiedostojen palvelemiseen, express.json JSON-rungon jäsentämiseen pyynnöissä ja express.urledcoded URL-koodatun rungon jäsentämiseen pyynnöissä.

Expressiin saa lisättyä toiminnallisuutta kolmannen osapuolen väliohjelmistoilla. Kolmannen osapuolen väliohjelmistoja voidaan asentaa Node.js:n paketinhallintatyökalulla. Kuvassa 10 on listattuna express-tiimin ylläpitämät väliohjelmistomoduulit.

Middleware module	Description	Replaces built-in function (Express 3)
body-parser	Parse HTTP request body. See also: body, co-body, and raw-body.	express.bodyParser
compression	Compress HTTP responses.	express.compress
connect-rid	Generate unique request ID.	NA
cookie-parser	Parse cookie header and populate `req.cookies`. See also cookies and keygrip.	express.cookieParser
cookie-session	Establish cookie-based sessions.	express.cookieSession
cors	Enable cross-origin resource sharing (CORS) with various options.	NA
csurf	Protect from CSRF exploits.	express.csrf
errorhandler	Development error-handling/debugging.	express.errorHandler
method-override	Override HTTP methods using header.	express.methodOverride
morgan	HTTP request logger.	express.logger
multer	Handle multi-part form data.	express.bodyParser
response-time	Record HTTP response time.	express.responseTime
serve-favicon	Serve a favicon.	express.favicon
serve-index	Serve directory listing for a given path.	express.directory
serve-static	Serve static files.	express.static
session	Establish server-based sessions (development only).	express.session
timeout	Set a timeout period for HTTP request processing.	express.timeout
vhost	Create virtual domains.	express.vhost

Kuva 10. Kolmannen osapuolen väliohjelmistot (expresjs.com 2020a.)

5 REST

5.1 Mitä REST tarkoittaa

REST on lyhenne sanoille Representational State Transfer. Se on vuonna 2000 Roy Fieldingin väitöskirjassa esitetty arkkitehtuurimalli www-sovelluspalveluiden luomiseen. Sovelluksia, jotka noudattavat REST-arkkitehtuurimallia, kutsutaan RESTful www-sovelluspalveluiksi. REST-arkkitehtuurimalli määrittelee viisi ohjesääntöä, joita täytyy noudattaa. Ohjesäännöt ovat: asiakas–palvelin, tilattomuus, välimuisti, yhtenäinen rajapinta ja kerroksittainen järjestelmä.

5.2 Asiakas–palvelin

Asiakas–palvelin -ohjesäännön tarkoitus on eritellä asiakas ja palvelin omiin komponentteihinsa. Molemmilla komponenteilla on omat tehtävänsä: palvelin tarjoaa palveluita ja kuuntelee pyyntöjä näihin palveluihin, asiakas lähettää pyynnöt palvelimelle. Asianmukainen tehtävien jakaminen yksinkertaistaa palvelinkomponenttia ja tekee sovelluksesta skaalautuvamman. Erittely mahdollistaa myös komponenttien itsenäisen kehittämisen.

5.3 Tilattomuus

Tilattomuus-ohjesäännön tarkoituksena on, että jokainen pyyntö asiakkaalta sisältää tarvittavat tiedot palvelimelle kyselyn ymmärtämiseen ja käsittelyyn. Istunnon tila pidetään kokonaan asiakkaalla, eikä sitä pidetä ollenkaan palvelimella.

5.4 Välimuisti

Välimuisti-ohjesäännön tarkoituksena on vähentää turhaa liikennettä asiakkaan ja palvelimen välille. Pyyntöjen vastaukset voidaan tallentaa välimuistiin, jolloin samaa resurssia haettaessa voidaan tuoda tieto välimuistista.

5.5 Yhtenäinen rajapinta

Yhtenäinen rajapinta -ohjesäännön tarkoituksena on, että kaikki pyynnöt asiakkaalta ovat samanlaisia riippumatta, onko asiakkaana web-selain, puhelinapplikaatio tai mikä tahansa muu. Yhtenäinen rajapinta -ohjesääntö koostuu neljästä ehdosta: (Avraham 2017.)

1. Resurssi täytyy identifioida asiakkaan tehdessä pyyntö palvelimelle.

2. Palvelimelta tulevassa vastauksessa täytyy olla riittävästi informaatiota asiakkaalle, jotta se voisi muokata resurssia.
3. Jokaisen pyynnön täytyy sisältää riittävästi informaatiota palvelimelle suorittaakseen pyynnön ja jokaisen vastauksen täytyy sisältää riittävästi informaatiota asiakkaalle ymmärtääkseen vastauksen.
4. Palvelin voi tiedottaa asiakasta, kuinka se pystyy muokata tilaansa. Esimerkiksi: asiakkaan kysyessä käyttäjänimeä palvelin voi palauttaa käyttäjänimen lisäksi linkin, kuinka vaihtaa käyttäjänimeä tai poistaa sen.

5.6 Kerroksittainen järjestelmä

Kerroksittainen järjestelmä -ohjesäännöllä tarkoitetaan, että asiakkaan pyytäessä palvelimelta resurssin tilaa, pyyntö voi kulkea usean palvelimen kautta. Palvelimille voidaan lisätä eri tasoja, kuten suojaustason ja välimuistitason. Tasoja voidaan lisätä ja poistaa välistä ilman, asiakkaan tai palvelimen ohjelmiston muokkaamista, eikä asiakas tiedä, että kuinka monta tasoa asiakkaan ja pyyntöön vastaavan palvelimen välillä on. (Avraham 2017.)

6 TAUSTAJÄRJESTELMÄN TOTEUTUS

6.1 Tavoite

Opintäytetyön tavoitteena oli rakentaa REST-arkkitehtuurin mukainen taustajärjestelmä aurinkosähköjärjestelmän tuottaman energian visualisoimiseen infonäytöllä. Taustajärjestelmästä pitää pystyä hakemaan käyttäjäkohtaisesti dataa reaaliajassa siten, että aurinkosähkövoimalan tuottoa pystyään kuvaamaan päivä-, viikko- ja kuukausikohtaisesti. Data pitää saada sellaisessa muodossa, että se pystytään piirtämään kaavioon.

Invertteri itsessään ei tallenna muuta dataa tuotannosta kuin päivän, vuoden ja elinkaaren kokonaistuotannon. Dataa täytyy siis tallentaa tietokantaan, josta infonäyttö-sovellus saa haettua halutut tiedot tuotannosta. Tietokannasta pitää pystyä hakemaan hetkellistä tehoa, viimeisen viikon tuotantoa päiväkohtaisesti viiden minuutin tarkkuudella ja vuosituotantoa kuukausikohtaisesti. Työstä tulee taustajärjestelmä usealle voimalalle, joten data täytyy autentikoida, jotta tietokannasta voidaan hakea haluttu data.

6.2 Palvelimen asennus

Opinnäytetyön toteutuksen alkaessa tammikuussa oli vielä talvi, joten aurinkokennojen päällä oli vielä lunta ja aurinko paistoi liian matalalta tuottaakseen tarvittavan määrän energiaa invertterin käynnistämiseen. Työn tekeminen täytyi siis aloittaa palvelimen asennuksella. Palvelimella on tarkoitus toteuttaa rajapinta Froniuksen invertterin lähettämän datan vastaanottoon ja tallentamiseen sekä välittämiseen asiakassovellukselle. Palvelimelle asennetaan myös asiakasohjelma datan visualisoimiseksi infonäytölle. Lem-Kemillä itsellään ei ole omaa palvelinta, joten palvelin päädyttiin vuokraamaan Hetzneriltä.

Hetzner Online on vuonna 1997 perustettu saksalainen yritys. Yritys on erikoistunut webpalveluntarjontaan ja datakeskusten ylläpitoon. Yrityksellä on tietokeskuksia Saksassa sekä Suomessa. Työtä varten vuokrattiin yksi virtuaalinen palvelin Suomen tietokeskukselta, jonne infonäyttö-sovellus ja rajapinta asennetaan.

Palvelimen vuokraamiseen Hetzneriltä luodaan käyttäjä Heznerin sivustolle. Käyttäjän luonnin yhteydessä täytyy todentaa henkilöllisyys. Todentaminen tapahtuu lähettämällä heille kuva voimassa olevasta henkilöllisyystodistuksesta. Tässä tapauksessa lähetettiin kuva ajokortista ja vahvistus Hetznerin palvelujen käyttöön saapui seuraavana päivänä. Käyttäjätunnuksen vahvistuksen jälkeen sivustolle luodaan projekti, jonne päästään lisäämään palvelin.

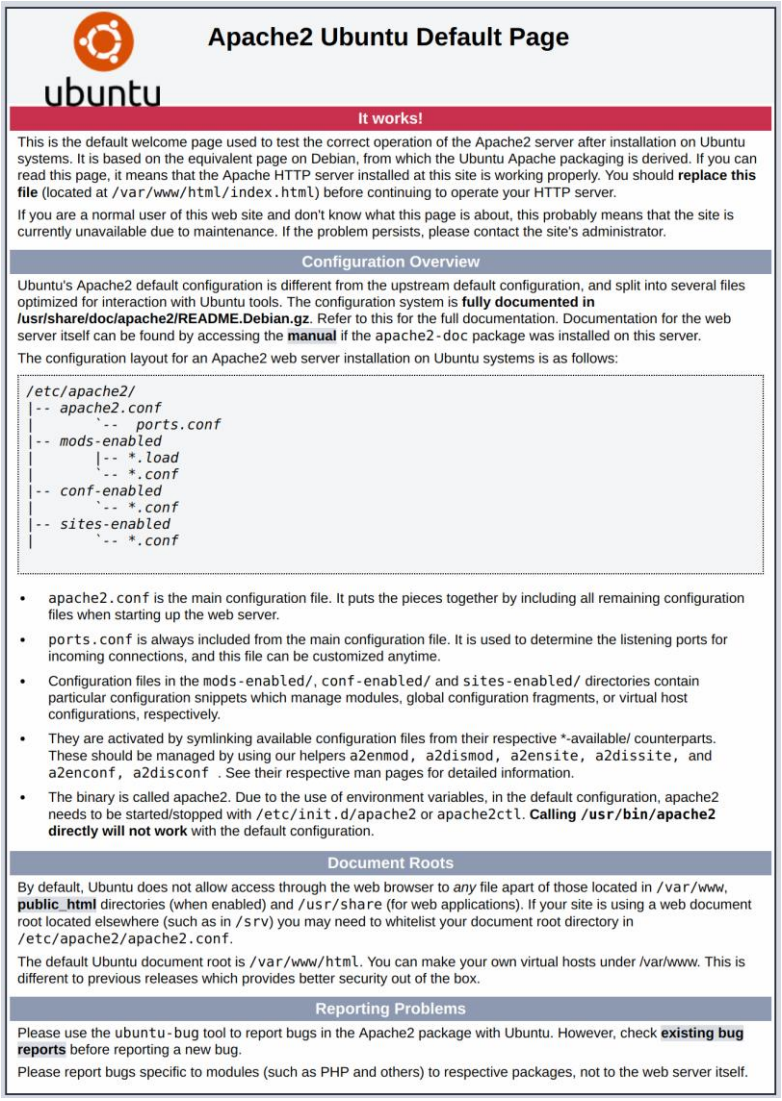
Palvelinta lisättäessä valitaan sijainti. Sijainniksi valittiin Suomen konesali Helsingissä. Vaihtoehtoina sijainnille ovat Saksan konesalit Nurenbergissa ja Falkensteinissa. Käyttöjärjestelmän valinnaksi on sivustolla valmiit levykuvat. Valittavissa ovat Ubuntu 18.04, Fedora 31, Debian 10 ja CentOS 8. Käyttöjärjestelmäksi valikoitui Ubuntu 18.04, koska sen käyttämisestä löytyi aikaisempaa kokemusta. Seuraavaksi valitaan laitteiston tyyppi. Toteutukseen riitti halvin vaihtoehto, mikä sisältää yhden virtuaalisen ytimen, 2 Gt keskusmuistia, 20 Gt SSD levytilaa sekä 20 Tt tietoliikennettä kuukaudessa. Palvelin vuokrataan käyttöön hyväksymällä asetukset, jolloin valittu levykuva asentuu palvelimelle.

Palvelimen komentorivikäyttöliittymään pääsee SSH-yhteydellä. SSH-asiakasohjelmana käytössä oli PuTTY. Kirjautumiseen tarvittavat pääsytiedot sai sähköpostilla palvelimen oston jälkeen. Ensimmäisen kirjautumisen jälkeen vaihdettiin root-käyttäjän salasana ja tehtiin uusi käyttäjä.

Palvelimelle asennetaan LAMP(Linux, Apache, MySQL, PHP) kokoelma. Palvelimelta löytyy jo Linux, joka tuli palvelimen oston yhteydessä. Apachen, MySQL:n ja PHP:n asennukseen käytetään Ubuntun paketin hallintatyökalua. Paketit päivitetään komennolla ”sudo apt update” ja haluttu paketti asennetaan komennolla ”sudo apt install ’haluttu paketti’ ”.

Apachen asennuksessa sille sallitaan HTTP- ja HTTPS-tietoliikenne määrittämällä palomuurin asetuksia. MySQL:n asennuksessa tulee huomioida se, että asennuksessa on haavoittuvaisuuksia. Haavoittuvuudet voidaan korjata ajamalla shell-skripti, joka tulee asennuksen mukana. Sillä voidaan asettaa root-käyttäjälle salasana, poistaa nimettömät käyttäjät ja poistaa testitietokanta, johon oletuksena pääsee nimettömät käyttäjät käsiksi. Seuraavaksi kirjaudutaan tietokantaan ja vaihdetaan root-käyttäjän todentaminen. Se käyttää oletuksena todennukseen auth_socket-lisäosaa, mutta vaihdetaan se käyttämään salasanaa.

Apachen toimivuus testattiin syöttämällä palvelimen julkinen ip-osoite selaimen osoitekenttään. Apachen toimiessa selaimen pitäisi avata Ubuntu 18.04 Apachen oletusverkkosivun (Kuva 11).



Apache2 Ubuntu Default Page

ubuntu

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Ubuntu tools. The configuration system is **fully documented in /usr/share/doc/apache2/README.Debian.gz**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the `apache2-doc` package was installed on this server.

The configuration layout for an Apache2 web server installation on Ubuntu systems is as follows:

```
/etc/apache2/
|-- apache2.conf
|       `--  ports.conf
|-- mods-enabled
|       |-- *.load
|       `-- *.conf
|-- conf-enabled
|       `-- *.conf
|-- sites-enabled
|       `-- *.conf
```

- `apache2.conf` is the main configuration file. It puts the pieces together by including all remaining configuration files when starting up the web server.
- `ports.conf` is always included from the main configuration file. It is used to determine the listening ports for incoming connections, and this file can be customized anytime.
- Configuration files in the `mods-enabled/`, `conf-enabled/` and `sites-enabled/` directories contain particular configuration snippets which manage modules, global configuration fragments, or virtual host configurations, respectively.
- They are activated by symlinking available configuration files from their respective *-available/ counterparts. These should be managed by using our helpers `a2enmod, a2dismod, a2ensite, a2dissite,` and `a2enconf, a2disconf` . See their respective man pages for detailed information.
- The binary is called apache2. Due to the use of environment variables, in the default configuration, apache2 needs to be started/stopped with `/etc/init.d/apache2` or `apache2ctl`. **Calling `/usr/bin/apache2` directly will not work** with the default configuration.

Document Roots

By default, Ubuntu does not allow access through the web browser to *any* file apart of those located in `/var/www`, **public_html** directories (when enabled) and `/usr/share` (for web applications). If your site is using a web document root located elsewhere (such as in `/srv`) you may need to whitelist your document root directory in `/etc/apache2/apache2.conf`.

The default Ubuntu document root is `/var/www/html`. You can make your own virtual hosts under /var/www. This is different to previous releases which provides better security out of the box.

Reporting Problems

Please use the `ubuntu-bug` tool to report bugs in the Apache2 package with Ubuntu. However, check **existing bug reports** before reporting a new bug.

Please report bugs specific to modules (such as PHP and others) to respective packages, not to the web server itself.

Kuva 11. Ubuntu 18.04 Apachen oletusverkkosivu

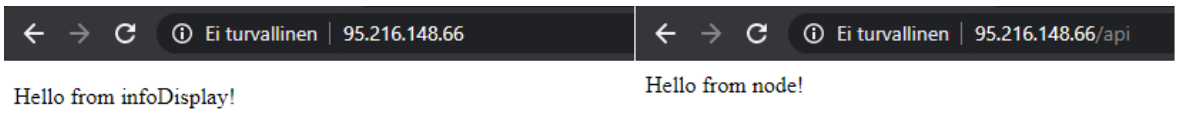
Palvelimella on kuitenkin tarkoitus olla infonäyttö-sovellus, sekä rajapintaa datan hakemiseen, joten Apachelle määritellään käänteinen välityspalvelin (reverse proxy). Käänteisen välityspalvelimen avulla saadaan samalta palvelimelta haettua infonäyttö-sovellus web-selaimeen, sekä halutut kutsut ohjattua rajapintaan. Käänteisen välityspalvelimen määrittelemiseksi Apachelle täytyy sallia proxy- ja proxy_http -moduulit. Käänteinen välityspalvelin määritellään Apachen virtuaalipalvelun määrittelyn yhteydessä (Kuva 12).

```
<VirtualHost *:80>
  ServerName fronius.devaus.dy.fi

  DocumentRoot /var/www/html/infoDisplay

  ProxyRequests Off
  ProxyPreserveHost On
  ProxyVia Full
  <Proxy *>
   Require all granted
  </Proxy>

  <Location /api>
    ProxyPass http://127.0.0.1:8080
    ProxyPassReverse http://127.0.0.1:8080
  </Location>

  <Directory "/var/www/html">
    AllowOverride All
  </Directory>
</VirtualHost>
```

Kuva 12. Apachen virtuaalipalvelun määrittely

Virtuaalipalvelun määrittelyssä kutsut ohjataan /var/www/html/infoDisplay-kansioon, jossa suoritetaan infonäyttö-sovellusta. Seuraavaksi määritellään käänteinen välityspalvelin, jonka avulla /api-reittiin lähetetyt kutsut ohjataan 8080-porttiin, jonne rajapinta asennetaan.

Testitarkoitukseen tehtiin infoDisplay-kansion alle index.html-tiedosto, joka tulostaa ”Hello from infoDisplay” ja api-kansion alle asennettiin Node.js ja tehtiin yksinkertainen sovellus, joka kuuntelee porttia 8080 ja tulostaa ”/api”-reitistä ”Hello from node!”. Laittamalla selaimeen serverin ip-osoitteen, niin index.html sivu avautui ja kun ip-osoitteen perään laittoi /api, niin tulostui ”hello from node!” (Kuva 13). Apachen asetukset olivat siis oikein, joten voitiin siirtyä rajapinnan rakentamiseen.



Kuva 13. Käänteisen välityspalvelimen testaus

6.3 Rajapinta

Palvelimelle asennetaan rajapinta, jonka tehtävänä on tallentaa aurinkosähkövoimalan tuotannon data tietokantaan ja hakea dataa tietokannasta infonnäyttö-sovellukselle

visualisoitavaksi. Rajapinta toteutetaan Node.js ja Express.js kirjastoa käyttämällä. Tietokantana toimii MySQL ja ORM-työkaluna Sequelize.

Dataa täytyy saada haettua rajapinnasta siten, että aurinkosähkövoimalan tuotantoa pystytään esittämään graafeissa. Voimalan huipputehoa kuvataan päiväkohtaisesti 5 minuutin välein ja voimalan kokonaistuotantoa kuvataan päivä ja kuukausikohtaisesti.

Testausvaiheessa palvelimelle asennettiin jo Node.js api/-kansioon. Noden avuksi asennetaan Express.js-kirjasto helpottamaan ja selkeyttämään koodia. Express väliohjelmistojen avulla saadaan myös autentikoitua palvelimelle tulevat kutsut.

6.3.1 Kansiorakenne

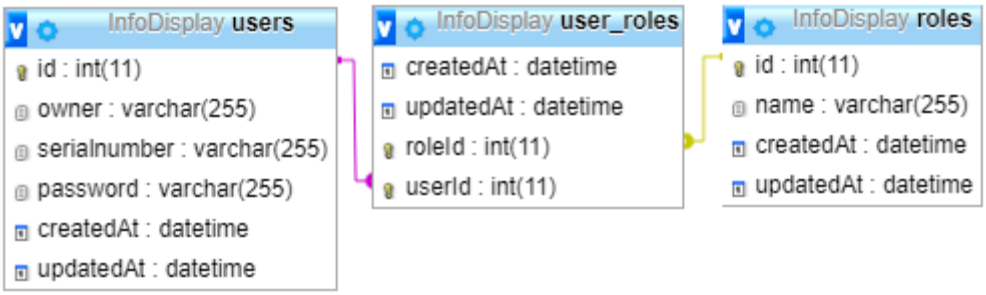
Kansiorakenne koostuu seuraavista hakemistoista:

- Config
- Routes
- Middleware
- Controllers
- Models.

Config-kansiossa määritellään asetuksia kuten tietokannan määrittely, joita tarvitaan useammassa kohtaa koodia. Näin koodia ei tarvitse kirjoittaa useampaan kertaan, vaan voidaan tuoda koodi täältä. Routes-kansiossa on määritelty rajapinnan reititys, jossa on määrätty päätepisteet rajapinnan resursseille. Controllers-kansiossa tehdään funktiot. Models-kansiossa luodaan tietokantataulujen mallit.

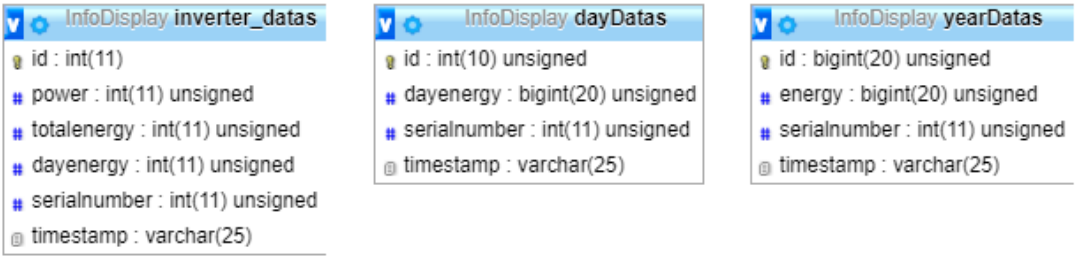
6.3.2 Taulujen mallintaminen

MySQL tietokantaan luodaan taulut käyttäjille sekä invertterien datalle. Käyttäjiä tehdessä niille luodaan roolit, jolla asiakassovelluksessa voidaan määritellä tarjottava sisältö. Esimerkiksi perus käyttäjällä on oikeus nähdä oman aurinkovoimalan tuottama energia, mutta admin-oikeuksilla oleva käyttäjä voi nähdä kaikkien voimaloiden tuotannon ja sillä on oikeus lisätä sekä poistaa käyttäjiä. Käyttäjille tarvitaan MySQL-tietokantaan 3 taulua. Ensimmäiseen tauluun tallennetaan käyttäjät, toiseen mahdolliset roolit ja kolmanteen relaatiot käyttäjien ja roolien välille (Kuva 14).



Kuva 14. Käyttäjät ER-kaavio

Invertterien tuottamalle datalle luodaan taulu, jonne tallennetaan kaikkien voimaloiden lähettämä data. Kuukauden ja päivän kokonaistuotoille tehdään omat taulunsa, jotta asiakassovellukseen saadaan haettua data helposti esitettävässä muodossa. Kuvassa 15 on esitetty tietokantataulujen rakenteet.



Kuva 15. ER-kaavio invertterien tuottamalle datalle

Mallit tietokantatauluille luodaan Sequelizen define-metodilla. Invertterien tuottamalle datalle luodaan malli, joka sisältää kaikkien taustajärjestelmään liitettyjen voimaloiden datan. Mallille annetaan ominaisuuksiksi power, totalenergy, dayenergy, serialnumber ja timestamp. Sequelize luo automaattisesti tietokantaan tauluihin sarakkeet createdAt ja updatedAt, mutta ne voidaan estää antamalla mallin luonnissa timestamp-ominaisuudelle arvon false (Kuva 16).

```
module.exports = (sequelize, Sequelize) => {
    const PushData = sequelize.define('inverter_datas', {
        power: {
            type: Sequelize.INTEGER.UNSIGNED
        },
        totalenergy: {
            type: Sequelize.INTEGER.UNSIGNED
        },
        dayenergy: {
            type: Sequelize.INTEGER.UNSIGNED
        },
        serialnumber: {
            type: Sequelize.INTEGER.UNSIGNED
        },
        timestamp: {
            type: Sequelize.STRING,
            allowNull: false
        }
    }, {
        timestamps: false
    })
    return PushData;
}
```

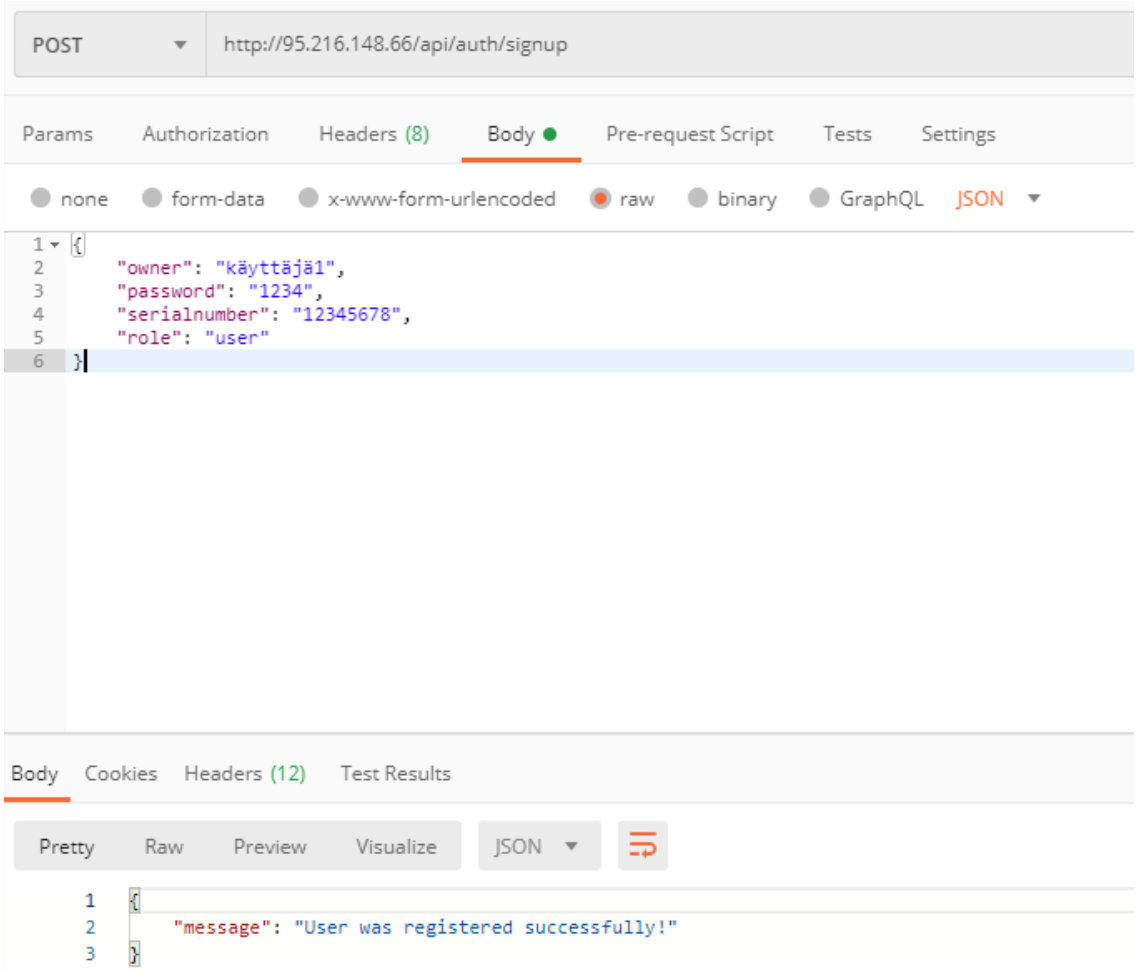
Kuva 16. Invertterien datan tietokantataulun malli

Käyttäjät-taulun mallille annetaan ominaisuuksiksi omistaja, salasana ja invertterin sarjanumero, jossa omistaja toimii käyttäjänimenä. Roolit-taululle tehdään myös malli, jolle annetaan ominaisuudet id ja nimi. Näiden välille luodaan relaatio, jossa käyttäjän viiteavaimena toimii userId ja roolien viiteavaimena roleId. Relaatiot luodaan Sequelizen mallin belongsToMany-metodilla. Sille annetaan ominaisuudet through, foreignKey ja otherKey, joka luo uuden mallin, jossa on relaatiot yhdistetyssä taulussa. Malli saa nimekseen ominaisuuden through-arvon. ForeignKey ja otherKey-ominaisuuksiin lisätään yhdistettävien tietojen viiteavain sekä toinen avain.

6.3.3 Autentikointi

NodeJS web-applikaatio toimii useamman invertterin taustajärjestelmänä, joten datat täytyy saada eroteltua toisistaan asiakassovellukselle. Jotta datat saa eroteltua toisistaan, täytyy asiakassovellukseen kirjautua. Palvelu kirjautumiseen tehdään JSON Web Tokenia(JWT) hyödyntämällä. Jotta tietokannasta saadaan haluttua dataa, luodaan access token, joka tallennetaan selaimen local storageen. Accesstokenin erääntyminen poistetaan käytöstä, joten kerran kirjautumalla saadaan oikeus aina tiettyyn dataan. Poikkeuksena on selaimen local storagen tyhjentyminen, joka tapahtuu, jos käyttäjä poistaa selaimen selaustiedot. Local storagen voi myös tyhjentää sivustokohtaisesti kehittäjän työkaluilla.

Käyttäjän tekemiselle on määrätty päätepiste ”/auth/signup”. Päätepisteelle täytyy lähettää POST-pyyntönä JSON objekti (Kuva 17). Objektissa täytyy olla käyttäjänimi, salasana, rooli sekä invertterin sarjanumero. Pyyntö ohjataan käsittelijään, jossa luodaan käyttäjä tietokantaan. Käyttäjille on kolme taulua tietokannassa. Ensimmäiseen tallennetaan lähetetyn objektin ominaisuudet, toisessa taulussa on roolit ja kolmanteen luodaan relaatiot roolien ja käyttäjien välille. Mahdollisia rooleja on user ja admin. Taulut ja relaatiot tehdään Seguelizella. Relaatiot muodostetaan monen suhde moneen. Käyttäjällä voi olla useampi rooli ja samalla roolilla voi esiintyä useampi käyttäjä.



Kuva 17. Käyttäjän tekeminen

6.3.4 Datan vastaanottaminen

Datan vastaanottamiselle määrättiin päätepiste /api/inverterdata/receive. Päätepiste kuuntelee HTTP POST -pyyntöjä ja ohjaa ne käsittelijään. Käsittelijässä otetaan kiinni pyynnön rungosta(body), jossa kaikki invertterin lähettämä data on. Tarvittavat tiedot invertteriltä ovat hetkellinen teho, päivän tuotto, kokonaistuotto, invertterin sarjanumero sekä aikaleima. Jos järjestelmässä on useampi invertteri linkitettynä niin JSON, jossa tiedot tulevat, sisältää useamman arvon jokaista tehoa tai tuotantoa koskevaa dataa kohdin. Tietoa ei tarvitse eritellä, että miten kukin invertteri tuottaa. Niiden kokonaistehon ja kokonaistuotannon laskeminen riittää. Kokonaistehon ja kokonaistuotannon sai helposti yksinkertaisella forEach-silmukalla, joka käy jokaisen arvon läpi ja lisää arvon muuttujaan. Aikaleiman sai helposti JSON:ista muuttujaan, koska riippumatta invertterien määrästä järjestelmässä aikaleima-arvoja oli vain yksi. Invertteri itsessään ei lähetä sarjanumeroaan, joten sarjanumero täytyi lähettää kyselyparametrina invertterin ja datamanagerin määrittelyn

yhteydessä. Kun sarjanumero tulee kyselyparametrina, sen arvo on helppo siirtää muuttujaan.

Datan tallentaminen tietokantaan tapahtuu Sequelizella. Tietokantaan on luotu taulu invertterin datalle ja se on mallinnettu Sequelizella. Mallin nimi on PushData, joten kutsutaan sen create-metodia, jolla voidaan lisätä merkintä tietokantaan (Kuva 18). Create-metodi tarvitsee atribuuteiksi olion, joka sisältää tiedon joka tauluun halutaan tallentaa. Oliolle asetetaan arvoiksi muuttujat, joilla on haluttu data.

```
PushData.create({
  power: PAC,
  dayenergy: dayEnergy,
  totalenergy: totalEnergy,
  serialnumber: sn,
  timestamp: timestamp
})
```

Kuva 18. Merkinnän tekeminen sequelizella

6.3.5 Päivä ja kuukauden kokonaistuotanto

Rajapinnasta täytyy saada haettua kuukauden kokonaistuotantoa vuoden ajalta ja päivän kokonaistuotantoa viikon ajalta. Kuukauden kokonaistuotannolle tehtiin oma tietokantataulu, jonne tallennetaan jokaisen invertterin kuukauden kokonaistuotanto, ja päivitetään sitä arvoa joka päivä. Kuukauden ensimmäisenä päivänä tehdään uusi merkintä jokaiselle invertterille. Jokaiselle merkinnälle tallennetaan tauluun kokonaistuotanto wattitunteina, invertterin sarjanumero sekä aikaleima muodossa ”vvvv/kk”. Koska invertteri itsessään ei lähetä kuukauden tuottoa, täytyy se laskea hakemalla edellisen kuukauden viimeisestä merkinnästä kokonaistuotanto ja vähentää se uusimman merkinnän kokonaistuotannosta. Päivän tuotannolle tehtiin myös oma taulunsa, jonne tallennetaan jokaisen invertterin päivän kokonaistuotanto, sarjanumero ja aikaleima muodossa ”vvvv/kk/pp”.

Node-applikaatioon tehtiin erillinen tiedosto, joka sisältää funktiot kuukauden kokonaistuotannon laskemiseen ja tallentamiseen sekä päivän kokonaistuotannon tallentamiseen ja päivittämiseen. Funktioiden täytyy päivittää kaikkien invertterien data, joten ne hakevat ensin tietokannasta käyttäjät-taulusta käyttäjien määrän, ja suorittaa funktiot niin monta kertaa for-silmukassa, kuin on käyttäjiä. Kuukauden kokonaistuotantoa laskiessa funktio tekee kyselyn invertterien pushData-tauluun. Kyselyä tehtäessä Sequelizella, sille annetaan attribuutteja, joilla määritellään, että haetaan edellisen kuukauden viimeisen päivän viimeistä merkintää ja invertterin sarjanumerona on käyttäjät-taulusta haettu sarjanumero (Kuva 19). Seuraavaksi haetaan pushData-taulusta viimeisin merkintä samalla

sarjanumerolla ja miinustetaan siitä edellisen kuukauden kokonaistuotanto. Näin saadaan tämänhetkisen kuukauden tuottama energia. Kun kuukauden tuottama energia on selvillä, täytyy se tallentaa tietokantaan omaan tauluunsa. Tallentaminen tapahtuu sille luodun mallin create-metodilla. Tämä funktio ajetaan kerran kuukaudessa kuun ensimmäisenä päivänä. Funktion ajamiseen kerran kuukaudessa käytetään node-cron -kirjastoa. Sillä voidaan ajastaa tehtäviä. Kirjaston avulla ajoitetaan toinen funktio, joka suoritetaan kerran päivässä. Toinen funktio toimii muuten samalla tavalla kuin ensimmäinen, mutta uuden merkinnän sijaan se päivittää tämän kuukauden arvoa. Merkinnän päivittämiseen Sequelizesta löytyy metodi update().

```
InvPushData.findOne({
    where: {
        timestamp: {
            [Op.regexp]: `${y}-${m}`
        },
        serialnumber: data[i].dataValues.serialnumber
    },
    order: [['id', 'DESC']]
})
```

Kuva 19. Kyselyn attribuutit

6.3.6 Datan hakeminen rajapinnasta

Datan hakemiseen rajapinnasta määriteltiin päätepisteet päivän huipputeholle, päivän ja kuukauden kokonaistuotannolle, sekä viimeisimmälle merkinnälle. Päätepisteet kuuntelevat HTTP GET -pyyntöjä.

Viimeisintä merkintää haettaessa lähetetään pyyntö api/last-reittiin. Pyyntöön tarvitaan kyselyparametrina invertterin sarjanumero. Pyyntö ohjataan käsittelijään, jossa data haetaan tietokannasta käyttäen Sequelizella luotua pushData mallia. Tietokantataulusta saadaan haettu yksi merkintä käyttämällä pushData-mallin findOne metodia. Metodille annetaan ehdoiksi, että haetaan vain rivit, joissa on serialnumber-sarakkeessa halutun invertterin sarjanumero ja data järjestetään id-sarakkeen mukaan laskevassa järjestyksessä (Kuva 20). Metodi luo kyselyn tietokantaan, joka palauttaa taulun viimeisimmän rivin. Rivi sisältää aurinkosähkövoimalan hetkellisen tehon, aikaleiman, sarjanumeron ja päivän sekä elinkaaren kokonaistuotannon. Kaikki arvot lähetetään JSON-muodossa asiakassovellukselle kutsun vastauksena.

```
PushData.findOne({
    where: {
        serialnumber: req.query.sn
    },
    order: [['id', 'DESC']]
})
```

Kuva 20. Ehdot viimeisimmän merkinnän hakemiseen

Päivän huipputehoa haettaessa lähetetään pyyntö api/daydata-reittiin. Pyyntöön tarvitaan kyselyparametreina päivämäärä sekä invertterin sarjanumero. Pyyntö ohjataan käsittelijään, jossa data haetaan tietokannasta Sequelisen pushData-mallin findAll-metodilla. Metodille annetaan ehto, että haetaan vain data, jossa on halutun invertterin sarjanumero ja säännöllisenä lausekkeena (regular expression) on päivämäärä, miltä data halutaan hakea (Kuva 21). Metodi luo kyselyn tietokantatauluun, joka palauttaa JSON-muodossa kaikki kyseisen invertterin merkinnät kyseiseltä päivältä

```
PushData.findAll({
    where: {
        timestamp: {
            [Op.regexp]: `${req.query.date}`
        },
        serialnumber: req.query.sn
    }
})
```

Kuva 21. Ehdot päivän huipputehon hakemiseen

Merkintöjä on tietokannassa kymmenen sekunnin välein, mutta asiakassovellukselle halutaan palauttaa huipputehot 5 minuutin välein. Jokaisessa merkinnässä on mukana aikaleima, joten huipputehoarvot voitaisi poimia JSON:ista aikaleiman mukaan viiden minuutin välein. Ongelmana olisi ikävät piikit infonäyttösovelluksen graafissa, jos arvo poimittaisiin hetkeltä, jolloin aurinko on ollut varjossa. Ongelman ratkaisuksi tehdään for-silmukka, joka laskee keskiarvon kaikista päivän arvoista 30 arvon välein. Näin saadaan keskiarvo huipputeholle viiden minuutin ajanjaksolta. Samalla otetaan aina 30. arvon aikaleima ja pyöristetään se lähimpään viiteen minuuttiin. Keskiarvo ja aikaleima talletetaan taulukkoon. Taulukko palautetaan asiakassovellukselle silmukan käytyä kaikki arvot läpi, jolloin taulukko pitää sisällään halutun invertterin huipputehon halutulta päivältä viiden minuutin välein aikaleimoineen.

Päivän ja kuukauden kokonaistuotannon hakeminen rajapinnasta on helpompaa, koska ne ovat omissa tietokantatauluissan. Taulut sisältävät sarakkeet: id, energy, serialnumber

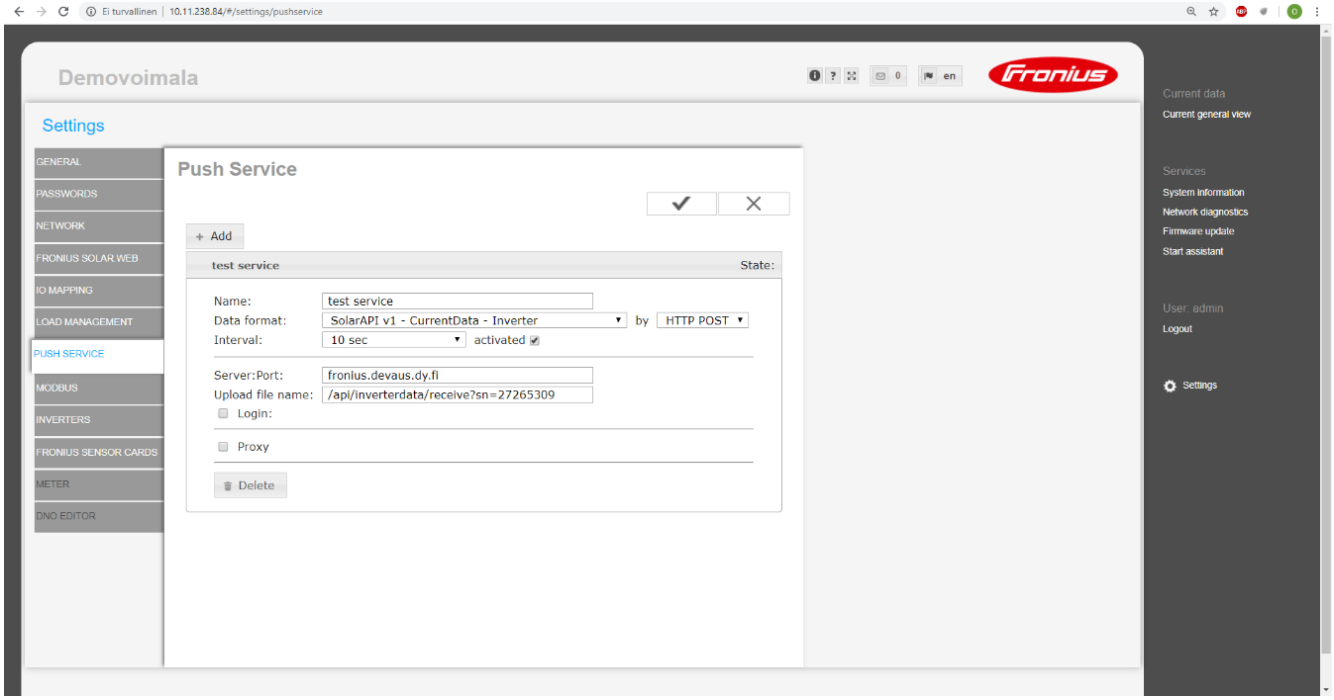
ja timestamp. Halutun datan hakemiseen rajapinnasta pyyntöön annetaan halutun invertterin sarjanumero ja päivämäärä, jolta data halutaan.

6.4 Datamanagerin asentaminen ja tiedon lähettäminen

Jos järjestelmässä on useampi invertteri, niin niiden datamanager-kortit yhdistetään toisiinsa lähiverkkokaapelilla DATCOM-liitännän avulla. Yhteen datamanagereista määritellään Pushservice-palvelu, jolla data voidaan lähettää halutulle palvelimelle.

Datan lähettämiseen tarvitsee kirjautua halutun kohteen invertterin datamanagerin web-käyttöliittymään, joka löytyy web-selaimella laitteen lähiverkon osoitteella. Käyttöliittymästä löytyy asetuksista PUSH SERVICE -kohta. Palveluita tarvitaan vain yksi, joka lähettää kaikkien linkitettyjen invertterien datat eteenpäin.

Kuvassa 22 on Pushservicen määrittely. Datan pystyy lähettämään joko FTP tai HTTP POST -protokollan avulla. Tässä työssä päädyttiin HTTP POST -protokollaan, koska se sopii yhteen taustajärjestelmän rajapinnan kanssa. Datan lähettämisen intervalli voidaan säätää minimissään kymmeneen sekuntiin ja maksimissaan kahteen tuntiin. Jotta päästäisiin seuraamaan aurinkosähköjärjestelmän tuotantoa mahdollisimman reaaliajassa, intervalli asetettiin kymmeneen sekuntiin. Kutsut ohjattiin palvelimelle, jossa taustajärjestelmä pyörii ja päätepisteeseen, joka tehtiin invertterien datan vastaanottamiseen. Koska taustajärjestelmän täytyy toimia usealle voimalalle, kutsuun lisätään invertterin sarjanumero kyselyparametrina. Sarjanumeron avulla voidaan erotella datat toisistaan.



Kuva 22. Pushservicen määrittely

Dataformaattina oli SolarAPI v1 – CurrentData -Inverter, joka sisältää JSON-muodossa invertterin tilan, mahdolliset vikailmoitukset, aikaleiman, tuotetun tehon sekä tuotetun energian tänään, tänä vuonna ja koko elinkaaren aikana (Kuva 23).


```
{
    "Body": {
        "DAY_ENERGY": {
            "Unit": "Wh",
            "Values": {
                "1": 465
            }
        },
        "PAC": {
            "Unit": "W",
            "Values": {
                "1": 292
            }
        },
        "TOTAL_ENERGY": {
            "Unit": "Wh",
            "Values": {
                "1": 5478750
            }
        },
        "YEAR_ENERGY": {
            "Unit": "Wh",
            "Values": {
                "1": 22913
            }
        }
    },
    "Head": {
        "RequestArguments": {
            "Query": "Inverter",
            "Scope": "System"
        },
        "Status": {
            "Code": 0,
            "Reason": "",
            "UserMessage": ""
        },
        "Timestamp": "2020-02-20T10:58:40+02:00"
    }
}
```


Kuva 23. SolarAPI v1 – CurrentData -Inverter

7 YHTEENVETO

Tavoitteena oli luoda taustajärjestelmä aurinkosähkövoimalan tuottaman energian visualisoimiseen. Sovellus toimii taustajärjestelmänä infonäyttö-sovellukselle, joka kuvaa voimalan tuotantoa. Voimalan hetkellistä tuotantoa haluttiin tarkastella reaaliajassa, sekä sen kokonaistuotantoa haluttiin visualisoida graafeissa päivä, viikko ja kuukausikohtaisesti.

Tavoitteeseen päästiin ja taustajärjestelmä päästiin asentamaan yrityksen oman konttorin aurinkosähkövoimalaan, sekä yhden yhteistyökumppanin voimalaan. Infonäytöt ovat olleet käytössä huhtikuun ajan ongelmitta.

Infonäyttö-sovellusten piti tulla kevään aikana asennettaviksi kaikkiin Hämeenmaan osuuskauppoihin, joihin on Lem-Kemin toimesta asennettu voimalat vuonna 2019. Sovelluksia ei kuitenkaan asennettu vielä COVID-epidemian puhjetessa. Taustajärjestelmän testaaminen jäi siis pois isommassa mittakaavassa.

Jokainen taustajärjestelmään kytketty voimala lähettää HTTP POST -kutsuna dataa palvelimelle 10 sekunnin välein. Kaikki voimalat on tarkoitus liittää samaan palvelimeen ja tietokantaan, joten työn seuraava vaihe on selvittää palvelimen, rajapinna ja tietokannan kapasiteetit ja ratkaista ongelmat mitä tulee vastaan voimaloiden määrän kasvaessa.

Seuraavaksi tutkitaan voiko taustajärjestelmää integroida kiinteistöautomaatioon. Datamanager verkkokortti tukee myös Modbus RTU ja Modbus TCP -protokollaa. Protokollien avulla saadaan invertteri keskustelemaan valmiiksi olemassa olevan kiinteistöautomaatiojärjestelmän kanssa. Aurinkosähkövoimalan ollessa vikatilassa voitaisiin virheilmoitus lähettää kiinteistön huoltajalle ja valvontakeskukseen.

LÄHTEET

Salo Solar Oy 2020. Aurinkopaneelit omakotitaloon. [viitattu 15.5.2020] Saatavissa: http://arevasolar.fi/fi/aurinkopaneelit-omakotitaloon

Energiavirasto 2019. Aurinkosähkön tuotantokapasiteetti lisääntyi 82 % vuodessa [viitattu 2.5.2020]. Saatavissa: https://energiavirasto.fi/tiedote/-/asset_publisher/aurinkosahkon-tuotantokapasiteetti-lisaantyi-82-vuodessa

Expressjs.com 2020a. Express middleware [viitattu 09.05.2020] Saatavissa: https://expressjs.com/en/resources/middleware.html

Expressjs.com 2020b. Release Change Log. [viitattu 09.05.2020] Saatavissa: https://expressjs.com/en/changelog/4x.html

Expressjs.com 2020c. Writing middleware for use in Express apps. [viitattu 09.05.2020] Saatavissa: http://expressjs.com/en/guide/writing-middleware.html

Fielding, R. 2000. Architectural Styles and the Design of Network-based Software Architectures [viitattu 10.05.2020]. Saatavissa: https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

Fronius International GmbH 2020a. First steps of Fronius. [viitattu: 26.04.2020] Saatavissa: https://www.fronius.com/en/about-fronius/history?

Fronius International GmbH 2020b. Fronius Symo. [viitattu: 26.04.2020] Saatavissa: https://www.fronius.com/en/photovoltaics/products/all-products/inverters/fronius-symo/fronius-symo-3-0-3-s

Hoyos 2017. What is ORM and Why You Should Use it. [viitattu: 24.04.2020] Saatavissa:https://blog.bitsrc.io/what-is-an-orm-and-why-you-should-use-it-b2b6f75f5e2a

Henderson 2019. What is NodeJS and Why You need to learn it. [viitattu 27.04.2020]. Saatavissa: https://medium.com/@michaelhenderson/what-is-nodejs-and-why-you-need-to-learn-it-f0760ba9a76a

Motiva 2017. Auringosta sähköä. [viitattu 02.05.2020]. Saatavissa https://www.motiva.fi/ratkaisut/uusiutuva_energia/aurinkosahko/aurinkosahkon_perusteet/auringosta_sahkoa

Node.js 2020a. Introduction to Node.js. [viitattu 05.05.2020] Saatavissa: https://nodejs.dev/introduction-to-nodejs

Node.js 2020b. An introduction to npm package manager. [viitattu 05.05.2020] Saatavissa: https://nodejs.dev/an-introduction-to-the-npm-package-manager/

Oracle 2020a. [viitattu 31.03.2020]. Saatavissa: https://www.oracle.com/mysql/

Oracle 2020b. What is MySQL? [viitattu 31.03.2020]. MySQL. Saatavissa: https://dev.mysql.com/doc/refman/8.0/en/what-is-mysq l.html

Patel 2018. What exactly is Node.js. [viitattu 27.04.2020]. Saatavissa: https://medium.com/free-code-camp/what-exactly-is-node-js-ae36e97449f5

Sequelize 2020. [viitattu 24.04.2020] Saatavissa: https://sequelize.org/master/manual/model-basics.html

Avraham 2017. What is REST – A Simple Explanation for Beginners [viitattu 10.05.2020] Saatavissa: https://medium.com/extend/what-is-rest-a-simple-explanation-for-beginners-part-2-rest-constraints-129a4b69a582

Työ- ja elinkeinoministeriö 2016. Valtioneuvoston selonteko kansallisesta energia- ja ilmastostrategiasta vuoteen 2030 [viitattu 02.05.2020] Saatavissa: https://tem.fi/documents/1410877/3570111/Kansallinen+energia-+ja+ilmastostrategia+vuoteen+2030+24+11+2016+lopull.pdf/a07ba219-f4ef-47f7-ba39-70c9261d2a63/Kansallinen+energia-+ja+ilmastostrategia+vuoteen+2030+24+11+2016+lopull.pdf