Onni Kytönummi

MITTAUSTIETOJEN JA VIDEOIDEN TALLENTAMINEN, KÄSITTELEMINEN JA JAKAMINEN

Opinnäytetyö
Tietotekniikan koulutusohjelma

2018

Kaakkois-Suomen ammattikorkeakoulu

Tekijä/Tekijät	Tutkinto	Aika
Onni Kytönummi	Insinööri (AMK)	Toukokuu 2018
Opinnäytetyön nimi Mittaustietojen ja videoiden tallentaminen, käsitteleminen ja jakaminen		37 sivua
Toimeksiantaja Kaakkois-Suomen ammattikorkeakoulu		
Ohjaaja Niina Mässeli		

Tiivistelmä

Tämän opinnäytetyön tarkoitus on käsitellä Mäkelänkankaan aurinko- ja tuulivoiman koulutus- ja tutkimuskeskittymän Webseuranta-verkkosivujen suunnittelua, toteutusta ja toiminnallisuutta. Toteutuksessa keskitytään mittaustiedon tallentamisen ja reaaliaikaisen videokuvan esityksen kehittämiseen sekä mittaustiedon käsittelemisen ja jakamisen optimointiin. Opinnäytetyötä voidaan käyttää dokumentaationa verkkosivujen asentamiseen ja ylläpitoon. Verkkosivujen tarkoituksena on antaa käyttäjälle keino seurata energiapuiston toimintaa ja voimantuottoa.

Verkkosivut kehitettiin WordPress-alustalle käyttäen hyödyksi alustan lisäosa- ja ohjelmointirajapintaa. Mittaustietoa varten luotiin koulun palvelimelle tietokanta ja WordPressiin lisäosa mittaustiedon automaattista tallentamista varten. Mittaustieto haettiin Empowerin mittaustietopalvelusta autentikoimalla palveluun. Haettu mittaustieto käsiteltiin ennen tietokantaan tallentamista. Lisäksi luotiin tapa tuoda projektia edeltävää mittaustietoa tietokantaan. Mittaustietotoiminnot myös optimoitiin käyttäjäystävällisemmiksi.

Mäkelänkankaan aurinko- ja tuulipuistoon asennettujen valvontakameroiden reaaliaikaista videokuvan esittämistä varten luotiin oma palvelin CentOS 7 -käyttöjärjestelmällä. Palvelimeen asennettiin FFmpeg-ohjelmistokokoelma ja videokuvan muuntoa varten tarvittavat encoder-ohjelmat. Ohjelmistokokoelmalla videokuva muunnettiin ja uudelleenohjattiin YouTube live-tapahtuma -palveluun.

Suurin osa projektiin asetetuista tavoitteista saavutettiin. Mittaustieto saatiin tallennettua luotettavasti tietokantaan, mikä oli projektin tärkein tavoite. Vanhempi mittaustieto saatiin myös onnistuneesti tallennettua tietokantaan. Suurin osa mittaustietotoiminnoista saatiin optimoitua onnistuneesti. Mittaustiedon jakamista ei kuitenkaan saatu optimoitua halutuksi haettaessa vuoden aikaista mittaustietoa. Reaaliaikaisen videokuvan esitys saatiin toteutettua, mutta palvelun signaalikatkeilun tuottamia ongelmia ei saatu ratkaistuksi.

Asiasanat

verkkosivusto, WordPress, tietokanta, PHP, FFmpeg

South-Eastern Finland University of Applied Sciences

Author (authors)	Degree	Time
Onni Kytönummi	Bachelor of Engineering	May 2018

Thesis Title

Storing, processing and distributing measurement data and videos

37 pages

Commissioned by

South-Eastern Finland University of Applied Sciences

Supervisor

Niina Mässeli

Abstract

The objective of the thesis study was to design, engineer and observe the functionality of website of Mäkenlänkangas education and research concentration of solar and wind power. Practical part of the thesis concentrated on storing measurement data, developing real-time video presentation and processing measurement data and distribution optimization. The thesis can be used as an installation and maintenance documentation for the website. The purpose of the website was to allow the user to follow the power plant operations and production.

The website was developed for WordPress platform utilizing its plugin and application programming interfaces. A database was developed for the measurements, and WordPress plugin was developed for automatic storing of measurements. The measurements were retrieved from Empower's measurement web service by authenticating to the service. The retrieved measurements were processed before they were stored in the database. In addition, means was developed for the older measurements to be stored in the database. Measurement functions were also optimized to make them more user-friendly.

A CentOS 7 operating system server was developed for presenting real-time video from security cameras that were installed at Mäkelänkangas solar power plant and wind farm, and FFmpeg software package with necessary encoders for encoding video were installed to the server. With the software package, video was encoded ja restreamed to YouTube Live Event service.

Most of the study's objectives were achieved. Measurement data was successfully stored in the database, which was the most important objective of the study. Older measurement data was also successfully stored in the database, and most of the measurement functions were successfully optimized. However, measurement distribution could not be optimized as desired when searching yearly data. Real-time video presentation was developed, but problems caused by the service's signal cuts could not be solved.

Keywords

website, WordPress, database, PHP, FFmpeg

SISÄLLYS

LYHENTEET & TERMIT

cron	Pohjautuu kreikkalaiseen sanaan chronos, joka tarkoittaa aikaa. Ajastuspalvelu Unix-pohjaisille käyttöjärjestelmille.
crontab	Lyhenne sanoista cron table. Asetustiedosto, josta cron käynnistää komentoja määritysten mukaan.
encoder	Ohjelma, joka muuttaa äänen tai kuvan eri muotoon. YouTuben live ja live-tapahtuma käyttää suomennettua termiä enkooderi.
FFmpeg	FF on lyhenne termistä fast forward ja mpeg on lyhenne termistä Moving Picture Experts Group. Ilmainen ohjelmistokokoelma multimediatiedostojen käsittelyyn.
JavaScript	Web-ympäristössä käytettävä dynaaminen komentosarjakieli.
MySQL	Relaatiotietokantaohjelmisto.
PHP	Lyhenne termistä PHP: Hypertext Preprocessor. Verkkokehitykseen suunniteltu ohjelmointikieli.
RTMP	Lyhenne termistä Real-Time Messaging Protocol. Adoben kehittämä tiedonsiirtoprotokolla multimedian suoratoistoa varten IP-verkoissa.
RTSP	Lyhenne termistä Real Time Streaming Protocol. Tiedonsiirtoprotokolla multimedian suoratoistoa varten IP-verkoissa. Toimii RTMP:n lailla.
SQL	Lyhenne termistä Structured Query Language. Tietokannan käsittelyyn käytetty ohjelmointikieli.
SSL	Lyhenne termistä Secure Sockets Layer. Tietoverkkosalausprotokolla.
striimausavain	YouTube liven ja live-tapahtuman käyttämä avain suoratoistoa varten.

1 JOHDANTO

Opinnäytetyön aiheena on Webseuranta-verkkosivujen eri toiminnallisuudet. Verkkosivuston päätarkoituksena on toimia aurinko- ja tuulivoiman koulutus- ja tutkimuskeskittymänä, josta voidaan kerätä arvokasta tutkimustietoa aurinko- ja tuulivoimaloiden tuotannosta ja toiminnasta. Aurinko- ja tuulivoiman tutkimuskeskittymää ja opinnäytetyön tuloksia voivat hyödyntää yritykset, koulut sekä yksityiset henkilöt.

Suomessa on kasvava tarve uusiutuville energianlähteille (Findikaattori 2017). Aurinko- ja tuulivoimaloiden kehityksen edistämiseksi on tärkeää tutkia voimaloista saatavaa tietoa, mutta Suomessa oli puutetta laajasti saatavilla olevista koulutus- ja tutkimuskeskittymistä (Xamk s.a.).

Opinnäytetyön aihe muodostui tarpeesta luoda koulutus- ja tutkimuskeskittymän Webseuranta-verkkosivut. Opinnäytetyön aihe saatiin alun perin Kymenlaakson ammattikorkeakoululta, joka fuusioitui opinnäytetyön aikana Mikkelin ammattikorkeakoulun kanssa Kaakkois-Suomen ammattikorkeakouluksi. Projektia johti Kaakkois-Suomen ammattikorkeakoulun merenkulun lehtori Kalle Suoniemi ja opinnäytetyön kirjallista osuutta ohjasi saman koulun tietotekniikan lehtori Niina Mässeli. Yhteistyössä oli mukana muun muassa Empower, jolta saatiin Mäkelänkankaan aurinko- ja tuulipuiston mittaustietoa. Verkkosivuston teossa oli mukana kaksi ohjelmoijaa ja yksi graafikko. Työtä toteutettiin pääasiassa Kaakkois-Suomen ammattikorkeakoulun pelilabrassa sekä toisinaan kotona.

Verkkosivujen toiminnallisuuksiin kuului tärkeimpänä mittaustiedon tallentaminen, käsitteleminen ja jakaminen. Lisäksi verkkosivuston kautta piti päästä näkemään reaaliaikaista videokuvaa Mäkelänkankaan aurinko- ja tuulipuistoon asennetuista valvontakameroista.

2 KEHITYSSUUNNITTELU

Projektin alussa keskusteltiin tulevista tavoitteista ja suunniteltiin projektin kulkua tavoitteiden saavuttamiseksi. Suunnittelua käytiin pääsääntöisesti viikoittaisissa palavereissa projektin vetäjän ja muiden ryhmäläisten seurassa. Toisinaan suunnittelua tehtiin sähköpostin välityksellä.

2.1 Tavoitteet

Projektin pääasialliseksi tavoitteeksi tuli luoda aurinko- ja tuulivoiman koulutus- ja tutkimuskeskittymä, mistä käyttäjä pystyy seuraamaan energiapuiston toimintaa ja voimantuottoa. Verkkosivuston tuli olla selkeäkäyttöinen ja monipuolinen. Luotujen ohjelmien ylläpidon tuli olla helppoa, sillä suurimmat kulut ohjelmistokehityksessä ovat yleensä ohjelmiston ylläpito (Haikala & Märijärvi 2004, 56).

Aurinko- ja tuulivoiman mittaustiedoille täytyi kehittää automaattinen tallentamisprosessi Empowerin mittaustietopalvelusta koulun omaan tietokantaan. Prosessin täytyi olla täysin omavarainen ja tietoturvallinen. Mittaustiedon tuli pysyä muuttumattomana keräysprosessin aikana tietokantaan.

Automaattisen tallentamisen lisäksi täytyi kehittää keino puskea projektia edeltävää mittaustietoa tietokantaan. Puskeminen ei saanut sotkea jo tietokannassa olemassa olevaa mittaustietoa tai vaikuttaa automaattiseen tallentamisprosessiin niin, että tietoa katoaisi.

Tietokanta suunniteltiin optimaaliseksi mittaustiedon helppoa ja nopeaa tallentamista ja jakamista varten. Tietoa, kuten esimerkiksi tarkkoja desimaalilukuja, ei saanut kadota taulukoiden muodon tai tallentamisen takia. Tietokanta suunniteltiin alusta asti huolella, sillä sen muokkaamisesta tulisi myöhemmin hankalaa, mikäli tallentamisprosessi olisi jo käynnissä tai tietoa haettaisiin jo verkkosivuilla. Mittaustieto täytyi pystyä hakemaan tietokannasta nopeasti, hidastamatta tietokannan muita prosesseja tai verkkosivustoa, josta tietoa haettaisiin. Tiedonhaun täytyi olla myös tietoturvallista.

Projektin viimeisessä vaiheessa kehitettiin keino esittää reaaliaikaista videokuvaa energiapuistosta. Tähän kehitettiin palvelin, joka pystyi vastaanottamaan, muokkaamaan ja uudelleenlähettämään energiapuistoon asennettujen kameroiden lähettämää videokuvaa.

2.2 WordPress

Keskiössä julkaisujärjestelmän valinnassa oli helppokäyttöisyys ja monipuolisuus sen kanssa työskentelevälle. Suunnittelun ja arvioinnin perusteella päädyttiin käyttämään WordPress julkaisualustaa sen tuomien etujen takia.

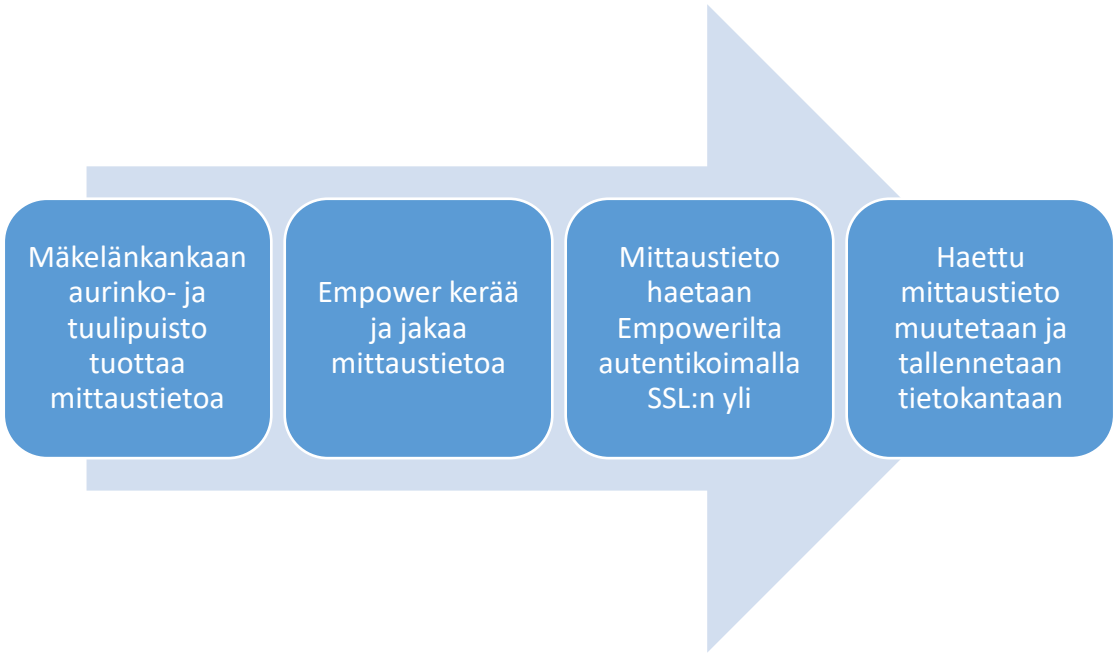
Alun perin erityisesti blogialustana tunnettu WordPress on nykyisin maailman suosituin julkaisualusta verkossa. Julkaisualustan suosioon ovat vaikuttaneet juuri edellä mainitut helppokäyttöisyys ja monipuolisuus. (Laukkarinen 2017.) Esimerkkeinä WordPressin tuomista eduista ovat yksinkertaisesti muokattavat teemat, monipuolinen lisäosarajapinta, mobiiliystävällisyys ja paljon muuta (Pitkänen 2016). Aurinko- ja tuulivoiman mittaustiedon käsittelyn kannalta tärkeimmät ominaisuudet WordPressissä olivat lisäosien hallinta sekä WordPressin oma ohjelmointirajapinta.

3 TOTEUTUS

Projektin työn toteutus jaettiin kolmeen pääosaan, jotka olivat mittaustiedon tallentaminen, mittaustiedon käsitteleminen ja jakaminen, ja reaaliaikaisen videokuvan esitys. Eri osia työstettiin joissain määrin samanaikaisesti, mutta pääsiassa niitä toteutettiin kyseisessä järjestyksessä töiden tärkeyden mukaan.

3.1 Mittaustiedon tallentaminen

Mittaustietoa varten rakennettiin sopiva tietokanta, jonne mittaustietoa tallennettiin ja josta pystyi haluttavaa mittaustietoa hakea. Ohjelmisto mittaustiedon tallentamiseen tehtiin luomalla WordPressiin tehtävään soveltuvan lisäosan. Tallentamisessa hyödynnettiin WordPressin tarjoamaa ohjelmistorajapintaa ja työkaluja. Mittaustieto haettiin Empowerin mittaustietopalvelusta autentikoimalla SSL:n yli. Ohjelmistokokonaisuus automatisoitiin lopulta koulun palvelimen cronilla.



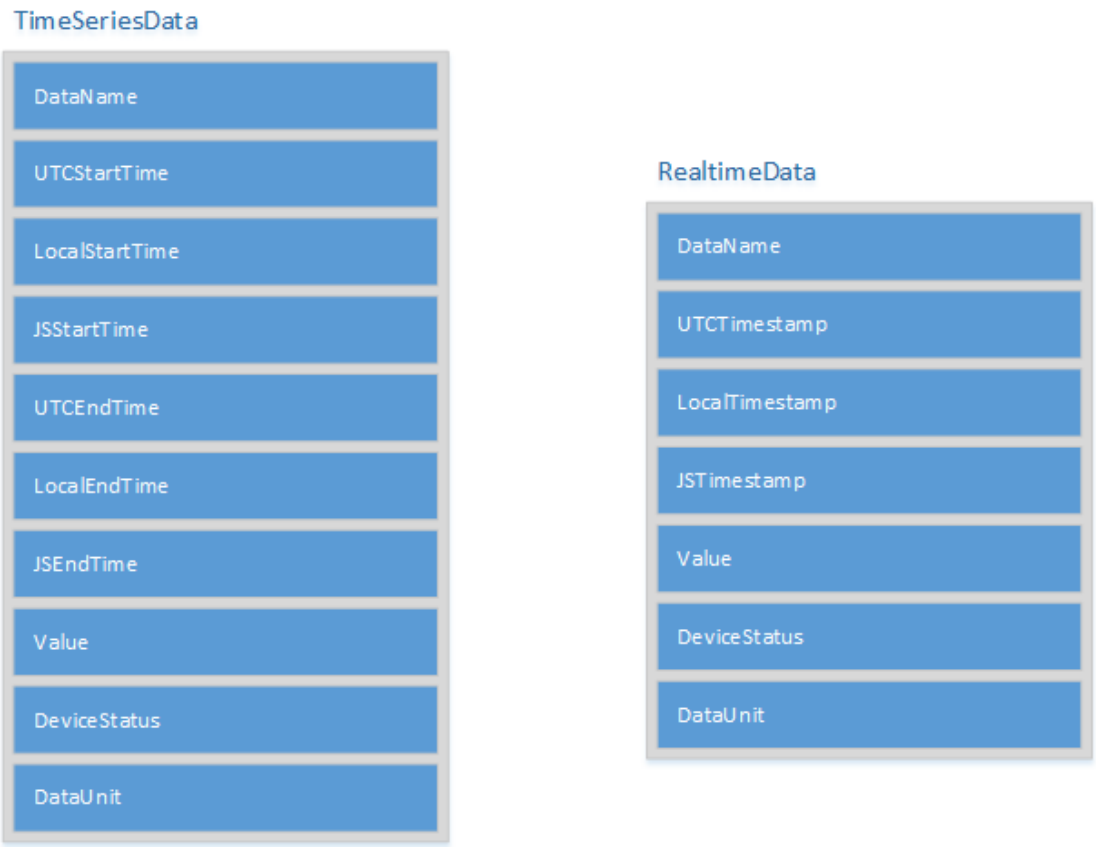
Kuva 1. Mittaustiedon tallennus yleisellä tasolla

Ylempi kuva esittää mittaustiedon tallennusprosessia yleisellä tasolla (Kuva 1). Empower kerää Mäkelänkankaan aurinko- ja tuulipuiston tuottamaa mittaustietoa, jota jaetaan mittaustietopalvelun kautta eteenpäin. WordPress lisäosa hakee mittaustiedon mittaustietopalvelusta autentikoimalla SSL:n yli. Haettu tieto muutetaan tietokantaan sopivaksi ja tallennetaan sinne.

3.1.1 Tietokannan rakentaminen

Tietokanta suunniteltiin alusta asti hyvin, jotta vältyttäisiin ongelmilta tulevaisuudessa. Huonosti suunniteltu tietokanta hankaloittaa tiedon tallentamista, käsittelyä ja jakamista (Hakala, Vainio & Vuorinen 2006, 345). Tietokantaa on hankala muokata myöhään silloin, kun tietoa on tallennettu jo paljon tai tietokanta on jo säännöllisessä käytössä, kuten projektin myöhemmässä vaiheessa todettiin.

Luodulle tietokannalle oli oleellista mallintaa siihen tallennettavaa tietoa. Helpoin ja aikaa säästävin ratkaisu projektin kannalta oli muodostaa taulukot Empowerin jo jaoteltuun tietomalliin. Empower jakaa mittaustietoa hakumenetelmien perusteella reaaliaikais- ja tuntipohjaisena. Mittaustiedot ovat edelleen jaoteltu eri kenttiin.



Kuva 2. Rakennemalli Empowerin jakamasta mittaustiedosta

Ylempi kuva mallintaa Empowerin jakaman mittaustiedon rakennetta ja siinä ilmenee, että mittaustiedot voitiin helposti jakaa kahteen eri tauluun (Kuva 2). Taulujen elementit nimettiin lähes sellaisenaan kuin mittaustietoja saatiin, mutta pienillä muokkauksilla. Sarkunin suositusten mukaisesti (2014) taulujen elementit oli hyvä nimetä käyttämällä pelkästään pieniä aakkosia. Pelkkien pienien aakkosten käyttö kuitenkin hankaloitti lukemista, mikäli elementtien nimien väleissä ei ollut välilyöntejä kuten Empowerin elementtien nimissä. Välilyöntien käyttö taulukoiden elementeissä hankaloitti SQL:n käyttöä (Sarkuni 2014). Ongelma kierrettiin lisäämällä välilyöntien sijaan alaviiva.

SQL:n VALUE-operaation takia elementin nimi ei voinut olla value. Elementti nimettiin data_value mahdollisten tulevien ongelmien välttämiseksi.

Rivien yksilöimiseksi taulukot tarvitsivat elementin, joka toimisi pääavaimena. Tuntipohjaiselle mittaustiedon elementille annettiin nimi ts_data_id ja reaaliaikaiselle rt_data_id. Elementin asetettiin lisäävän automaattisesti yhden numeroyksikön jokaiselle uudelle mittaustiedolle. Tulevan tiedon määrän takia elementin tietotyypiksi valittiin bigint(20), joka pystyy tallettamaan

18446744073709551615 positiivista lukua, kun se on määritelty etumerkittömäksi (Meloni 2003, 79).

Muiden elementtien tietotyypeiksi valittiin niihin tallennettavaa tietoa parhaiten edustaviksi. Koska Empowerilta haettavat mittaustiedot eivät sisältäneet koskaan tyhjiä tietoja, elementit rajattiin niin, että niihin ei saanut tyhjiä tietoja tallennettua.

Nopeaa tiedonhakua varten taulukoihin oli hyvä määritellä indeksi (Hakala ym. 2006, 345). Indeksi auttaa tiedon järjestämistä tietokannassa hakujen nopeuttamiseksi (Hovi 2004, 143). Projektin hakujen kannalta tehokkain tapa järjestää mittaustiedot oli antaa indeksi mittaustiedon otetulta päivämäärältä, sillä mittaustietoa haettiin usein päivämäärän perusteella.

3.1.2 Autentikointi ja mittaustiedonhaku

Empowerin mittaustietopalvelu oli salattu SSL-salausprotokollalla. Pääsääntöisesti SSL-salaus verkkosivustolla auttaa suojaamaan yhteyden käyttäjän ja palvelimen välillä salaten siinä kulkevan tiedon. SSL lisää yhteyksien välistä tietoturvaa, mutta samalla hankaloittaa yhteyden muodostamista palveluun. (Assembla s.a.)

Autentikoinnin aloittamiseksi PHP:llä Empowerin mittaustietopalvelun osoite, käyttäjätunnus ja salasana tallennettiin erillisiin muuttujiin. Lisäksi luotiin uusi muuttuja $result, jolle annettiin arvo ’nil’. Autentikointiin vaadittavat tiedot laitettiin kontekstijonoon $context stream_context_create()-funktiolla. Lopullinen pyyntö palveluun ja mittaustiedonhaku merkkijonoon $result tapahtui funktiolla file_get_contents(). Virheiden havaitsemiseksi käytettiin try- ja catch-lohkoja keräämään poikkeuksia.

3.1.3 Mittaustiedon tallentaminen tietokantaan

Onnistuneen mittaustiedonhaun jälkeen merkkijonollinen mittaustieto muutettiin tietokannan kannalta sopivammaksi taulukoksi. Tämä tapahtui muuttamalla ensin merkkijono $result simplexml_load_string()-funktiolla objektiin $XML. XML-muotoisen objektin tiedot täytettiin uuteen taulukkoon $data_stack foreach()- ja array_push()-funktioilla.

```
if ( $result != 'nil' ) {
    $XML = simplexml_load_string( $result );
}

//var_dump( $XML );

//array for data
$data_stack = array();

//fill array with wanted values
foreach ( $XML->Data->TimeSeriesData as $TimeSeriesData ) {
    array_push ( $data_stack, array (
        $TimeSeriesData->DataName,
        $TimeSeriesData->UTCStartTime,
        $TimeSeriesData->LocalStartTime,
        $TimeSeriesData->JSStartTime,
        $TimeSeriesData->UTCEndTime,
        $TimeSeriesData->LocalEndTime,
        $TimeSeriesData->JSEndTime,
        $TimeSeriesData->Value,
        $TimeSeriesData->DeviceStatus,
        $TimeSeriesData->DataUnit
    ));
}
```

Kuva 3. Tuntipohjaisen mittaustiedon täyttö taulukkoon

Ylemmän kuvan mukaisessa täytössä indeksoituu yksittäiset mittaustiedot sekä niiden elementit (Kuva 3). Täytetystä taulukosta pystyi helposti myöhemmin valitsemaan halutut mittaustiedot tietokantaan tallentamista varten.

Tietokannan avaamiseksi tallennettiin halutut kirjautumisasetukset muuttujaan $connection msqli_connect()-funktiolla. for($i=0; $i<count($data_stack); $i++)-silmukalla lisättiin saadun mittaustietomäärän verran mittaustietoja haluttuun taulukkoon.

```
//insert array into database
for ( $i = 0; $i < count( $data_stack ); $i++ ) {

    if ( $data_stack[$i][0] != 'TESTIMUUTTUJA' ) {

        $sql = "INSERT INTO data (
            data_id, utc_start_time, local_start_time, js_start_time,
            utc_end_time, local_end_time, js_end_time, data_value, device_status, data_unit
            )
        VALUES (
            '".$data_stack[$i][0]."', '".$data_stack[$i][1]."', '".$data_stack[$i][2]."',
            '".$data_stack[$i][3]."', '".$data_stack[$i][4]."', '".$data_stack[$i][5]."',
            '".$data_stack[$i][6]."', '".$data_stack[$i][7]."', '".$data_stack[$i][8]."',
            '".$data_stack[$i][9]."'
        );";

        if ( $connection->query( $sql ) === TRUE ) {
            //echo "New record created successfully";
        } else {
            //echo "Error: " . $sql . "<br>" . $connection->error;
        }
    }
}
```

Kuva 4. for-silmukka mittaustiedon lisäämiseksi tietokantaan

Ylemmän kuvan mukaisen silmukan sisällä tarkastetaan ensin, onko lisättävää mittaustietoa jo tietokannassa vertaamalla lisättävää mittaustietoa taulukon tietoihin (Kuva 4). Tarkastuksen optimoimiseksi tietojen haku päättyy sen löydettäessä ensimmäisen vastaavan mittaustiedon. Duplikaatin löytyessä silmukka valitsee seuraavan lisättävän mittaustiedon. Mikäli duplikaatteja ei löydy, silmukka lisää uuden mittaustiedon tietokantaan. Silmukka jatkaa toimintaansa, kunnes se on käynyt lävitse kaikki lisättävät mittaustiedot.

Tallennusprosessin päätökseksi tietokantatulokset tyhjennettiin mysqli_free_result($result)-funktiolla ja yhteys tietokantaan suljettiin mysqli_close($connection)-funktiolla. Yhteys sulkeutuu yleensä automaattisesti skriptin käytyä loppuun, mutta funktion käyttö takaa sulkemisen säästäen suorituskykyä.

3.1.4 Tallentamisen automatisointi

Mittaustiedon tallentaminen tietokantaan tuli automatisoida, sillä uusia mittaustuloksia tuli jatkuvasti ja verkkosivuston tuli pystyä esittämään mahdollisimman todenmukaista ja ajankohtaista tietoa. Automatisointiin tarvittiin ajallista tehtävävuorottajaa, joka useimmissa Unix-pohjaisissa käyttöjärjestelmissä on cron (Guzel 2010). WordPressillä on oma cron-palvelu, mutta toisin kuin Unix-pohjainen cron, sillä pystyy ajoittamaan tehtäviä vain silloin, kun

verkkosivustolla käydään (McFarlin 2013). Tämä loi ongelmia, mikäli verkkosivustolla ei käynyt tiettyinä aikaväleinä, sillä tehtäviä ei käynnistynyt silloin. Ongelmien välttämiseksi WordPressin käyntiin perustuva cron päädyttiin poistamaan käytöstä ja ajastamaan se aktivoitumaan tietyin aikavälein koulun palvelimen cronilla.

WordPressin käyntiin perustuva cron poistettiin käytöstä lisäämällä tiedoston wp-config.php alkuun define('DISABLE_WP_CRON', true);. Palvelimella ajastettiin tämän jälkeen WordPressin cron aktivoitumaan tietyin aikavälein cronjob-ohjelmalla. Tämä tapahtui lisäämällä cronjob-ohjelmaan rivi /5 * * * wget -q -O - https://webseuranta.xamk.fi/wp-cron.php?doing_wp_cron. Tämä ajasti cronin aktivoitumaan viiden minuutin välein, mikä oli projektin kannalta sopiva aikaväli.

Automaattista tallentamista varten tallennusskriptit luotiin tehtäviksi ja ne ajastettiin. Tämä toteutui luomalla oma lisäosa WordPressiin. Lisäosa antoi työkalut tallentamisen helppoon käsittelyyn antamalla WordPressin oman ohjelmointirajapinnan käyttöön.

Lisäosan luonti tapahtui luomalla ensin kansio, jonne lisäosan kannalta tarpeelliset tiedostot laitettiin. Projektin lisäosakansio nimettiin measurement-data-manager. Kansion sisälle luotiin viisi PHP-tiedostoa: collect-data.php, collect-rt-data.php, data-push.php, measurement-data-manager.php, seurantadata-config.php. Tiedostoihin collect-data.php ja collect-rt-data.php tuli tuntipohjaisen ja reaaliaikaisen mittaustiedon keräysskriptit kutsuttaviin funktioihin. Tiedostoon data-push.php tuli vanhemman mittaustiedon puskemisen skripti kutsuttavaan funktioon. Tiedosto measurement-data-manager.php oli lisäosan pääskripti. Tiedosto seurantadata-config.php sisälsi tietokannan kirjautumiseen tarpeelliset asetukset. Jotta WordPress tunnisti lisäosan päätiedoston, measurement-data-manager.php-tiedoston alkuun lisättiin tarpeelliset määritykset.

```php
<?php
/** * Plugin Name: Measurement Data Manager
 * Description: Manages everything needed for collecting measurement data.
 * Author: Onni Kytönummi
 * Version: 1.0 */
/* Your code goes below here. */
```

Kuva 5. measurement-data-manager.php-tiedoston määritykset

Ylemmän kuvan mukaiset määritykset antoivat WordPressille tarpeelliset tiedot lisäosan toimivuudelle verkkosivustolla (Kuva 5). Määritykset myös antoivat lisäosan käyttäjälle lyhyen selostuksen lisäosan tiedoista. Määrittämisen jälkeen tiedostoon lisättiin halutut sisältävät tiedostot alemman kuvan mukaisesti include-lausekkeella (Kuva 6).

```php
// Includes
include 'collect-data.php';     // Function to collect timeseries data
include 'collect-rt-data.php';  // Function to collect realtime data
```

Kuva 6. measurement-data-manager.php-tiedoston sisällytettävät tiedostot

Tiedostojen sisältämät funktiot pystyttiin nyt tarvittaessa kutsumaan measurement-data-manager.php-tiedostossa. Ajastamista varten funktiot kiinnitettiin WordPressin toimintaan add_action()-funktiolla, kuten alemmassa kuvassa (Kuva 7).

```php
// Hooks timeseries data collecting -function to an action
add_action('scheduled_data_collecting', 'collect_data');
```

Kuva 7. WordPressin toimintaan kiinnitetty funktio

Ylemmän kuvan mukainen toiminta pystyttiin ajastamaan aktivoitumaan tietyin väliajoin wp_schedule_event()-funktiolla (Kuva 7). Ajastuksen pysäyttämiseen käytettiin wp_clear_scheduled_hook()-funktiota, jota voitiin tarvittaessa kutsua.

```php
// Activates scheduled timeseries data collecting when activating Measurement Data Manager
register_activation_hook(__FILE__, 'activate_data_collecting');

// Defines timeseries data collecting -function's schedule
function activate_data_collecting() {
    if (! wp_next_scheduled ( 'scheduled_data_collecting' )) {
        wp_schedule_event(time(), 'hourly', 'scheduled_data_collecting');
    }
}
```

Kuva 8. Toimintojen ajastaminen WordPressissä

Jotta ajastettu toiminta menisi päälle lisäosan käynnistettyä, ajastettu funktio määritettiin ylemmän kuvan mukaisesti register_activation_hook()-funktiolla (Kuva 8). Ajastuksen sulkemiseksi lisäosan poistuttua käytöstä käytettiin register_deactivation_hook()-funktiota.

Ajastettavien toimintojen helppoa tarkastelua varten WordPressiin ladattiin lisäosa WP Crontrol. Kyseinen lisäosa antoi myös helpon tavan hallita ajastettuja toimintoja sekä antoi mahdollisuuden luoda yksittäisiä tehtäviä, mikä auttoi vanhemman mittaustiedon puskemisen kanssa.

3.1.5 Vanhemman mittaustiedon puskeminen tietokantaan

Projektin aikana Empowerin mittaustietopalvelulle oli ehtinyt kertymään parin kuukauden aikaista tuntipohjaista mittaustietoa, joka tuli myös tallentaa koulun tietokantaan. Vanhan mittaustiedon puskeminen tietokantaan ei saanut häiritä automaattista tallentamisprosessia.

Puskemista varten luotiin data-push.php-tiedosto automaattisen mittaustiedon keräyksen lisäosan kansioon ja measurement-data-manager.php-tiedostoon lisättiin luotu tiedosto include-lausekkeella. data-push.php-tiedosto luotiin lähes samankaltaiseksi kuin collect-data.php-tiedosto, mutta mittaustiedon haussa vain määriteltiin itse haluttu aikaväli, jolta mittaustietoa haettiin. Mittaustiedon määrän takia aikavälin täytyi olla tarpeeksi pieni joka puskulle, jotta koulun palvelimella riitti tehot muihin samanaikaisiin toimintoihin. Kahden viikon aikaväli huomattiin sopivaksi kokeilujen perusteella, sillä pidemmällä aikavälillä pusku usein keskeytyi ennen kuin koko aikavälin mittaustiedot saatiin tallennettua. data-push.php-tiedoston mittaustiedon keräysfunktion nimettiin data_push().

Tiedostossa measurement-data-manager.php funktio data_push() kiinnitettiin WordPressin toimintaan add_action()-funktiolla. Toimintoa kutsuttiin manuaalisesti WordPressin lisäosasta WP Crontrol.

Puskun etenemistä seurattiin PuTTY:n kautta. Kahden viikon aikaisen tuntipohjaisen mittaustiedon puskemisessa tietokantaan meni noin kaksi tuntia. Mikäli tietokantaa olisi optimoitu aikaisemmin, puskemisessa olisi arviolta kulunut merkittävästi vähemmän aikaa. Onnistuneen puskun jälkeen aikaväliä data-push.php-tiedostossa muutettiin seuraavaan kahteen viikkoon ja puskemista toistettiin, kunnes kaikki vanhat mittaustiedot saatiin tallennettua tietokantaan.

3.1.6 Tallentamisen optimointi

Projektin aikana tallentamisprosessia piti optimoida kasvavan tietomäärän ja verkkosivuston kasvavan käytön takia. Tallentamisen optimointia hoidettiin tekemällä muokkauksia käytössä olleen mittaustiedon tallennuslisäosan kopioon. Vanha lisäosa korvattiin muokatulla kopiolla niin, että tallentamisprosessissa ei tullut taukoja, jossa tietoa olisi voinut jäädä pois.

Empowerin tarjoama mittaustieto ei aina ollut ajantasaista, joten tuntipohjaisessa tallentamisessa tallentui usein duplikaatteja tietokantaan. Duplikaattien tallentaminen tietokantaan vei prosessointitehoa ja kasvatti tietokannan kokoa turhaan, joten ne piti suodattaa tallentamisessa pois. Jokaista syötettävää mittaustietoa verrattiin yksitellen tietokannan tietoihin ennen tallentamista. Verrattaessa riitti, että tietokannasta haettiin vain verrattavan mittaustiedon nimi data_name ja mittaustiedon ottoaika. Näin saatiin kaikkein välttämättömimmät tiedot vertailua varten viemättä liikaa prosessointitehoa tiedon hakemisella. Haku laitettiin keskeytymään ensimmäisen tuloksen löydyttyä tietokannasta, minkä jälkeen verrattiin seuraavaa syötettävää mittaustietoa.

Empowerin tuntipohjaisissa mittaustiedoissa oli TESTIMUUTTUJA-niminen mittaustieto, jota ei tarvittu tallentaa tietokantaan. Tallentamisessa kyseinen mittaustieto suodatettiin vertaamalla syötettävän mittaustiedon nimeä siihen.

Eri vertailuja tallentamisessa nopeutettiin määrittelemällä SELECT-lausekkeella vain hakuihin tarpeellisia sarakkeita SELECT*-lausekkeen sijaan. Tähden käyttö lausekkeessa valitsi kaikki sarakkeet halutusta taulusta, mikä tarkoitti suuremman tietomäärän hakua, mikä hidasti vertailuja.

3.2 Mittaustiedon käsitteleminen ja jakaminen

Projektin edetessä mittaustiedon määrä kasvoi niin paljon, että se vaikutti vakavasti sen käsittelyyn ja jakamiseen. Sekä tietokantaa, että jakamista optimoitiin niin, että käyttäjä pystyi käyttämään palveluita ilman tarpeetonta odottamista.

3.2.1 Tietokannan optimointi

Mittaustiedon määrän kasvu tietokannassa projektin edetessä alkoi hidastamaan tietokannan kaikkia toimintoja merkittävästi. Hitaalla tietokannalla oli mahdollisuus hävittää tallennettavaa mittaustietoa ja jumittaa verkkosivustoa hakemalla tietoa. Tietokannan muotoa ja sisältöä täytyi muuttaa turvallisesti hävittämättä oleellista sisältöä. Varmuuskopiointia varten ts_data- ja rt_data-tauluille luotiin identtiset data_backup- ja rt_data_backup-taulut samalla tavalla kuin alkuperäiset taulut (Kuva 9).

```
CREATE TABLE ts_data_backup (
    ts_data_id bigint(20) unsigned not null auto_increment,
    data_name varchar(60) not null default 'DataName',
    utc_start_time bigint(14) not null default 19700101000000,
    local_start_time char(17) not null default '20000101000000+02',
    js_start_time bigint(20) not null default 0,
    utc_end_time bigint(14) not null default 19700101000000,
    local_end_time char(17) not null default '20000101000000+02',
    js_end_time bigint(20) not null default 0,
    data_value double(9,3) not null default 0.000,
    device_status int(1) not null default 0,
    data_unit varchar(30) not null default 'DataUnit',
    primary key (ts_data_id)
);
CREATE INDEX idx_js_start_time ON ts_data_backup (js_start_time);

INSERT INTO ts_data_backup (
    data_name, utc_start_time, local_start_time, js_start_time,
    utc_end_time, local_end_time, js_end_time, data_value, device_status, data_unit
)
SELECT
    data_name, utc_start_time, local_start_time, js_start_time,
    utc_end_time, local_end_time, js_end_time, data_value, device_status, data_unit
FROM ts_data
ORDER BY js_start_time;
```

Kuva 9. Esimerkki tietokannan varmuuskopioinnista

Ylemmän kuvan INSERT INTO -lausekkeella varmuuskopioitiin nykyisen tietokannan sisältö saman kuvan uusiin tauluihin (Kuva 9). Kopioitaessa SELECT-lausekkeesta ts_data_id ja rt_data_id jätettiin pois, sillä mittaustiedon yksittäisillä id-numeroilla ei ollut projektin kannalta vielä väliä. ORDER BY -lausekkeella mittaustiedot järjestettiin uuteen tauluun js_start_time ja js_timestamp

mukaan. Varmuuskopioitaessa ORDER BY ei ollut välttämätön lauseke, mutta se selkeytti uusien taulujen sisällön tarkastelua.

Tietokannan kokoon vaikutti projektin alussa tallentuneet lukuisat duplikaatit. Duplikaatit saattoivat myös aiheuttaa virheellistä tietoa haettaessa niitä (Hernandez 2000, 188–189). Duplikaattien poistaminen MySQL:n avulla teki hankalaksi ohjelmistoon liittyvät viittausongelmat poistaessa viitattua tietoa (MySQL 2018).

```
DELETE FROM ts_data
WHERE ts_data_id
NOT IN (
    SELECT ts_data_id
    FROM (
        SELECT ts_data_id
        FROM ts_data
        GROUP BY
            data_name, utc_start_time, local_start_time, data_value, device_unit
    )
);
```

Kuva 10. Esimerkki MySQL:n viittausongelmasta

Ylemmän kuvan mukaisella syntaksilla viitattava tieto poistuisi (Kuva 10), joten MySQL ei aja poistoa. Ongelma kierrettiin lisäämällä eri aliaksia viitattaviin tietoihin.

```
DELETE th
FROM ts_data th
WHERE ts_data_id
NOT IN (
    SELECT ts_data_id
    FROM (
        SELECT ts_data_id
        FROM ts_data
        GROUP BY
            data_name, utc_start_time, local_start_time, data_value, device_unit
    )
    x
);
```

Kuva 11. Duplikaattien poistamissyntaksi tuntipohjaiselle mittaustiedolle

Aliaksen lisääminen ylemmän kuvan mukaisesti varmisti, että ohjelma piti väliaikaisesti muistissa tiedot, joita se tarvitsi poistaakseen duplikaatit (Kuva 11). Kaikkien duplikaattien poistamiseen molemmista tauluista meni yhteensä noin puoli tuntia kuvan mukaisilla syntakseilla.

Eniten tietokannan toimintoja hidastava ongelma oli taulujen elementtien epäoptimaaliset tietotyypit. Projektin alussa taulujen ajankohtia esittävät elementit, kuten utc_start_time ja utc_timestamp, olivat char-tyyppiä, vaikka elementit sisälsivät ainoastaan numeroita. Monimerkkisen päivämäärän tallentaminen turhaan char-tyypiksi bigint sijaan kasvatti tallennukseen vaadittavan tiedon määrää merkittävästi, mikä samalla hidasti hakujen nopeuksia. Tyypillinen 14 merkkiä pitkä päivämäärä vei char-tyyppisenä tilaa 14 tavua, kun taas bigint-tyyppisenä korkeintaan 4 tavua.

```
ALTER TABLE rt_data
MODIFY COLUMN utc_timestamp bigint(14) zerofill not null default 19700101000000;
```

Kuva 12. Tietokannan elementtien tyyppien muuttaminen

Elementtien tyyppien vaihtaminen ylemmän kuvan tavalla (Kuva 12) pienensi tietokannan kokoa merkittävästi, mikä samalla nopeutti tiedonhakua merkittävästi. bigint-tyyppi antoi myös mahdollisuuden luoda elementistä indeksi, joka ryhmittää mittaustietoa päivämäärän mukaan, mikä helpotti tiedonhaussa.

```
CREATE INDEX idx_js_start_time ON ts_data_backup (js_start_time);
```

Kuva 13. Indeksin luominen elementistä

Projektin kannalta hyödyllisin indeksi oli JavaScript-muotoiset aloitusajat, sillä niissä oli tarkimmat ajat mittaustiedoille, ne olivat tiedostokooltaan muita taulujen aikaa ilmaisevia elementtejä pienempiä ja niitä haettiin graafeissa eniten. CREATE INDEX -lausekkeella luotiin js_start_time ja js_timestamp elementeistä indeksit, kuten ylemmässä kuvassa (Kuva 13).

3.2.2 Jakamisen optimointi

Tietokannan mittaustietoja jaettiin eteenpäin verkkosivuston graafeihin ja ladattaviin taulukoihin. Mittaustiedon määrän kasvaessa mittaustietojen hakeminen tietokannasta hidastui, mikä myös hidasti ja jumitti verkkosivuja. Tietokannan optimointi auttoi tiedon hakemisessa paljon, mutta hakemista täytyi nopeuttaa vielä lisää verkkosivuston käytön helpottamiseksi.

Alkuperäiset haut tietokannasta päivämäärän mukaan tehtiin hakemalla utc_start_time, utc_end_time ja utc_timestamp. Näiden päivämäärät sisälsivät

14 merkkiä, kun taas JavaScript-päivämäärät sisälsivät 13 merkkiä. Pienemmän merkkimäärän numeroa oli nopeampi hakea tietokannasta, joten haettava päivämäärä täytyi muuttaa ensin JavaScript-muotoon, jotta voitiin hakea merkkimäärältään pienemmät js_start_time, js_end_time ja js_timestamp.

```
// save dates into datetime variable
$startTime = strtotime($_POST['startTime_']); //'20161030130000';
$endTime = strtotime($_POST['endTime_']); //'20161030140000';
$startTimeLastYear = strtotime($_POST['startTimeLastYear_']); //'20161030130000';
$endTimeLastYear = strtotime($_POST['endTimeLastYear_']); //'20161030140000';

// convert dates into javascript
$start_time_js = date('U000',$startTime);
$end_time_js = date('U000',$endTime);
$start_time_last_year_js = date('U000',$startTimeLastYear);
$end_time_last_year_js = date('U000',$endTimeLastYear);
```

Kuva 14. DateTime-muuttujan konvertointi JavaScript-muotoon

Jotta saatu nykyinen aika voitiin konvertoida JavaScript-muotoon, se muutetiin ensin ylemmän kuvan esimerkillä Unix-pohjaiseksi aikaleimaksi strtotime()-funktiolla (Kuva 14). Unix-pohjainen aikaleima konvertoitiin tämän jälkeen JavaScript-muotoiseksi date()-funktiolla.

Alkuperäisissä hauissa tehtiin myös useita kyselyitä tietokannasta eri päivämäärien takia. Useampi kysely lisäsi mittaustiedon hakuun menevää aikaa. Kyselyiden yhtenäistäminen tarkoitti, että hakuja tarvitsi tehdä vähemmän yksittäisiltä tauluilta, mikä nopeutti performanssia, vaikka tuloksissa joutui tehdä enemmän jaottelua.

```
//tuulivoima
$sql = "SELECT data_name, data_value, device_unit, js_timestamp
FROM rt_data
WHERE (
    js_timestamp BETWEEN ".$start_time_js." AND ".$end_time_js."
)
OR (
    js_timestamp BETWEEN ".$start_time_last_year_js." AND ".$end_time_last_year_js."
)";

// Perform the query
$result = mysqli_query( $connection, $sql );
if ( mysqli_num_rows($result) > 0 ){
    while ( $row = mysqli_fetch_assoc ($result) ){

        if (
            in_array ( $row['data_name'], $keywords ) && $row['js_timestamp']>=$start_time_js
        ){
            //echo "Match found";
            array_push (
                $dataStack, array (
                    $row['data_name'], $row['data_value'], $row['device_unit'], $row['js_timestamp']
                )
            );
        } else if (
            in_array ( $row['data_name'], $keywordsLastYear ) && $row['js_timestamp']<=$end_time_last_year_js
        ){
            //echo "Match found";
            array_push (
                $dataStack, array (
                    $row['data_name'].'_LASTYEAR', $row['data_value'], $row['device_unit'], $row['js_timestamp']
                )
            );
        }
    }
}
mysqli_free_result($result);
```

Kuva 15. Kyselyiden yhtenäistäminen ja tuloksien jaottelu

Kyselyitä samasta taulusta yhtenäistettiin ylemmän kuvan mukaisesti OR-operaattorilla (Kuva 15). Haetut tulokset vertailtiin ja jaoteltiin if- ja else if -lausekkeiden avulla eri $dataStack-taulukoihin. Jaottelun jälkeen tulokset tyhjennettiin mysqli_free_result($result)-funktiolla.

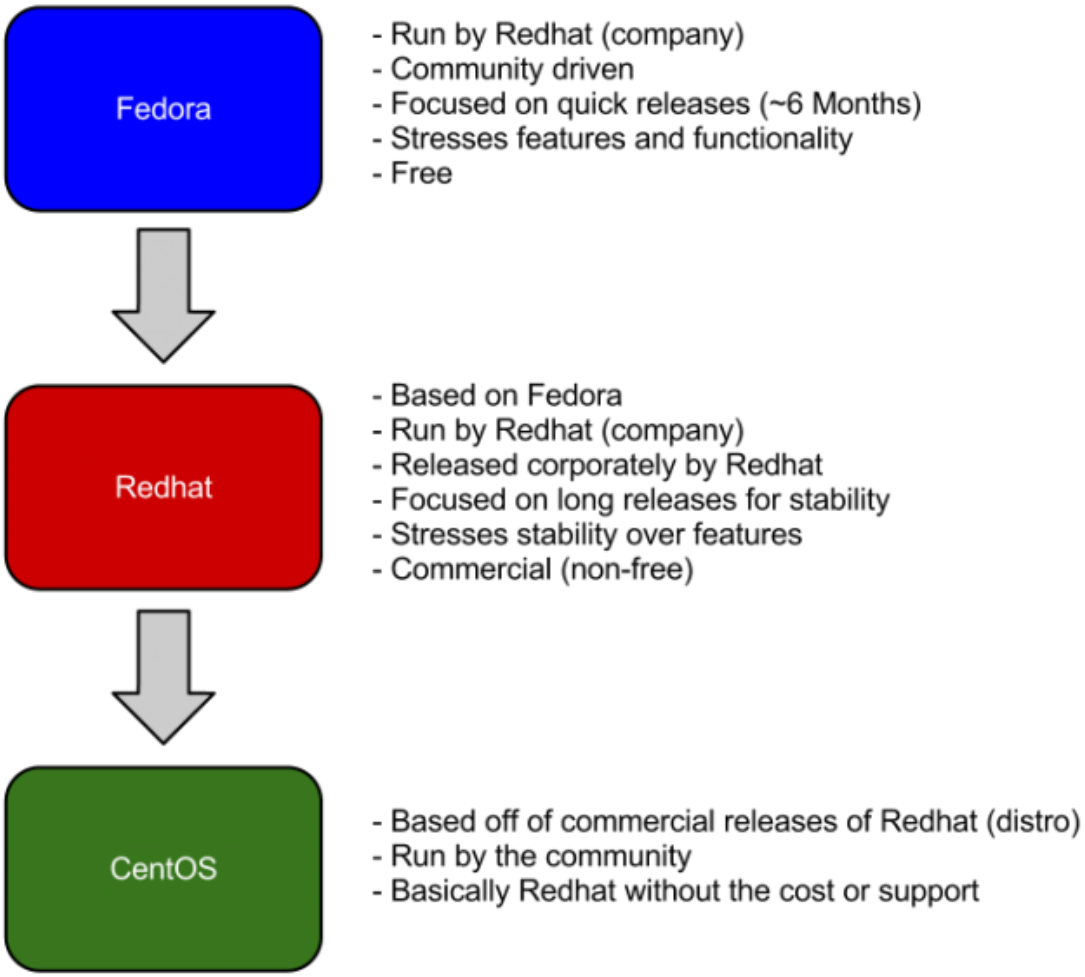
3.3 Reaaliaikaisen videokuvan esitys

Mäkelänkankaan aurinko- ja tuulipuistoon asennettujen valvontakameroiden reaaliaikaista videokuvaa tuli esittää Webseuranta-verkkosivujen kautta. Tehtävää varten luotiin oma palvelin CentOS 7 -käyttöjärjestelmällä, johon asennettiin videon käsittelyä ja uudelleenohjaamista varten FFmpeg-ohjelmistokokoelma. Käsitelty video lähetettiin YouTuben live-tapahtuma-palveluun.

3.3.1 CentOS 7

Reaaliaikaisen videokuvan esittämistä varten tarvittiin palvelin, joka pystyi ottamaan vastaan Mäkelänkankaan aurinko- ja tuulipuiston valvontakameroiden videokuvaa, käsittelemään se ja lähettämään käsitelty videokuva esitettäväksi.

Palvelinta varten täytyi valita käyttöjärjestelmä, joka sopi parhaiten tähän tarkoitukseen. Kulujen minimoimiseksi palvelimen käyttöjärjestelmän oli hyvä olla ilmainen.



Kuva 16. Käyttöjärjestelmien eroja

Palvelimen tärkein toiminta, videokuvan käsittely, vaati siihen sopivaa ohjelmaa. FFmpeg on laajasti käytössä oleva avoimen lähdekoodin ohjelmisto, joka sopii reaaliaikaisen videokuvan esittämiseen (Casey 2017). Ohjelmistokokoelmaa käytetään paljon Linux-jakelupaketeissa (Hong s.a.), joten päädyttiin käyttämään Linux-pohjaista palvelinta. Projektia varten oli käytössä joko Fedora- tai CentOS-käyttöjärjestelmät. Ylemmästä kuvasta ilmenee sekä Fedoran että CentOSin olevan ilmaisia käyttöjärjestelmiä (Kuva 16). CentOS on kuitenkin kahdesta vakaampi, joten sitä käytettiin projektissa.

3.3.2 FFmpeg

Aurinko- ja tuulivoimapuiston valvontakameroiden videokuvaa pystyi verkon kautta hakea vain RTSP-muotoisena ja kuvaa pystyi jakamaan kerralla vain rajalliselle määrälle. RTSP-muotoinen kuva ei myöskään ollut verkkosivusto-ystävällinen muoto sellaisenaan, vaan kuvan katsominen olisi vaatinut tarpeellisen lisäosan asentamista selaimeen (Flashphoner 2017). Koska videokuvaa saatettiin katsoa sellaisessa ympäristössä, jossa lisäosien asentaminen selaimeen ei olisi mahdollista tai kätevää, täytyi kuva muuntaa jo palvelimessa. Videokuvan jakoon kävi YouTuben live-tapahtuma-palvelu jakamisen kätevyyden takia, joten muutettu videokuva täytyi pystyä lähettämään palveluun.

FFmpeg on erityisesti Linux-jakelupaketeissa suosittu ohjelmistokokoelma (Hong s.a.), joka pystyy hoitamaan kyseisiä tehtäviä. FFmpegin luottoa nostattaa sen käyttö pohjana monissa muissa ohjelmissa kuten Handbrake, Plex ja Avanti (Casey & Suleman 2017).

FFmpegin asennus CentOS 7 -käyttöjärjestelmään vaati Nux-pakettivaraston asennuksen. Nuxin asennus vaati taas EPEL-pakettivaraston asennuksen. (Maruthamuthu 2016.)

```
sudo yum install https://dl.fedoraproject.org/pub/epel/epel-release-latest-6.noarch.rpm

sudo rpm --import http://li.nux.ro/download/nux/RPM-GPG-KEY-nux.ro
sudo rpm -Uvh http://li.nux.ro/download/nux/dextop/el7/x86_64/nux-dextop-release-0-1.el7.nux.noarch.rpm

yum repolist
```

Kuva 17. Pakettivarastojen asennus

Ylempi kuva esittää pakettivarastojen asennuksen. Pakettivarastojen asennus vaati komentojen ajon pääkäyttäjänä, mikä onnistui sudo-komennolla ennen varsinaista komentoa (Kuva 17). Komennolla yum install epel-release asennettiin EPEL-pakettivarasto. Nux-pakettivaraston asennukseen tarvittiin GPG-avain, joka tuotiin rpm -import -komennolla. Avaimen avulla Nux pystyttiin lataamaan onnistuneesti rpm -UVH -komennolla. Pakettivarastojen asennusten onnistuminen tarkastettiin yum repolist -komennolla.

FFmpeg voitiin pakettivarastojen asennuksen jälkeen asentaa suoraan, mutta videonmuuntoon ja lähettämiseen vaadittiin tiettyjä encoder-ohjelmia, joten ne asennettiin ensin. Valvontakameroiden videokuvan muuntoa varten tarvittavat

encoder-ohjelmat olivat NASM, Yasm, libx264, libx265, libfdk_aac, libmp3lame, libopus, libogg, libvorbis ja libvpx. Encoder-ohjelmia varten luotiin ffmpeg_sources-kansio home-kansioon mkdir-komennolla. Encoder-ohjelmien paketit haettiin ja purettiin luotuun kansioon. Puretut encoder-ohjelmat konfiguroitiin ja asennettiin. Encoder-ohjelmien asennuksen jälkeen FFmpeg asennettiin encoder-ohjelmien kanssa, kuten alemmassa kuvassa (Kuva 18).

```
cd ~/ffmpeg_sources
curl -O -L https://ffmpeg.org/releases/ffmpeg-snapshot.tar.bz2
tar xjvf ffmpeg-snapshot.tar.bz2
cd ffmpeg
PATH="$HOME/bin:$PATH" PKG_CONFIG_PATH="$HOME/ffmpeg_build/lib/pkgconfig" ./configure \
  --prefix="$HOME/ffmpeg_build" \
  --pkg-config-flags="--static" \
  --extra-cflags="-I$HOME/ffmpeg_build/include" \
  --extra-ldflags="-L$HOME/ffmpeg_build/lib" \
  --extra-libs=-lpthread \
  --extra-libs=-lm \
  --bindir="$HOME/bin" \
  --enable-gpl \
  --enable-libfdk_aac \
  --enable-libfreetype \
  --enable-libmp3lame \
  --enable-libopus \
  --enable-libvorbis \
  --enable-libvpx \
  --enable-libx264 \
  --enable-libx265 \
  --enable-nonfree
make
make install
hash -r
```

Kuva 18. FFmpegin asennus encoder-ohjelmien kanssa

FFmpegin pakattu asennuskansio tuotiin curl -O -komennolla ja purettiin tar xjvf -komennolla. FFmpegin konfiguroitiin määrittelemällä ensin PKG_CONFIG_PATH sijainniksi $HOME/ffmpeg_build/lib/pkgconfig. Konfiguroidessa tarvittavat encoder-ohjelmat asetettiin aktiivisiksi -enable-asetuksella. Valmis konfiguraatio koottiin make-komennolla ja asennettiin make install -komennolla. Valmiin asennuksen jälkeen tarkistemuistin tyhjennettiin hash -r -komennolla. FFmpegin toimivuutta testattiin lopuksi ffmpeg-komennolla.

3.3.3 Reaaliaikaisen videokuvan muunto ja uudelleenohjaus

FFmpegin asennuksen jälkeen luotiin skriptit reaaliaikaisen videokuvan vastaanottamiselle, muuntamiselle ja uudelleenohjaamiselle YouTube live-tapahtuma -palveluun. Skripteille luotiin käyttäjän bin-kansioon uusi kansio nimeltä ffmpeg-scripts komennolla mkdir. Uuteen kansioon luotiin YouTube live-tapahtumaan tarvittaville striimausavaimille lähdeskripti livestreamkeys.sh Vi-ohjelmalla. Tiedostoon lisättiin valmiiksi muuttujat livestreamkey1, livestreamkey2

ja niin edelleen kuutta kameraa kohti. Muuttujiin lisättiin myöhemmin striimausavaimet. Komennolla chmod +x tiedostoille annettiin käynnistysoikeus. Reaaliaikaisen videokuvan vastaanottamista, muuntoa ja uudelleenohjausta varten tehtiin aluksi yksi skripti nimeltä camera1.sh, jolle annettiin myös käynnistysoikeus.

```
source /home/kytonummi/bin/ffmpeg-scripts/livestreamkeys.sh

~/bin/ffmpeg \
-re \
-rtsp_transport tcp \
-i "rtsp://<käyttäjätunnus>:<salasana>@<ip>:<portti>/cam/realmonitor?channel=1&subtype=1" \
-f lavfi \
-i anullsrc=channel_layout=stereo:sample_rate=44100 \
-c:a libmp3lame \
-ab 128k \
-ar 44100 \
-c:v copy \
-threads 2 \
-bufsize 512k \
-f flv "rtmp://a.rtmp.youtube.com/live2/$livestreamkey1"

```

Kuva 19. camera1.sh-tiedoston skripti

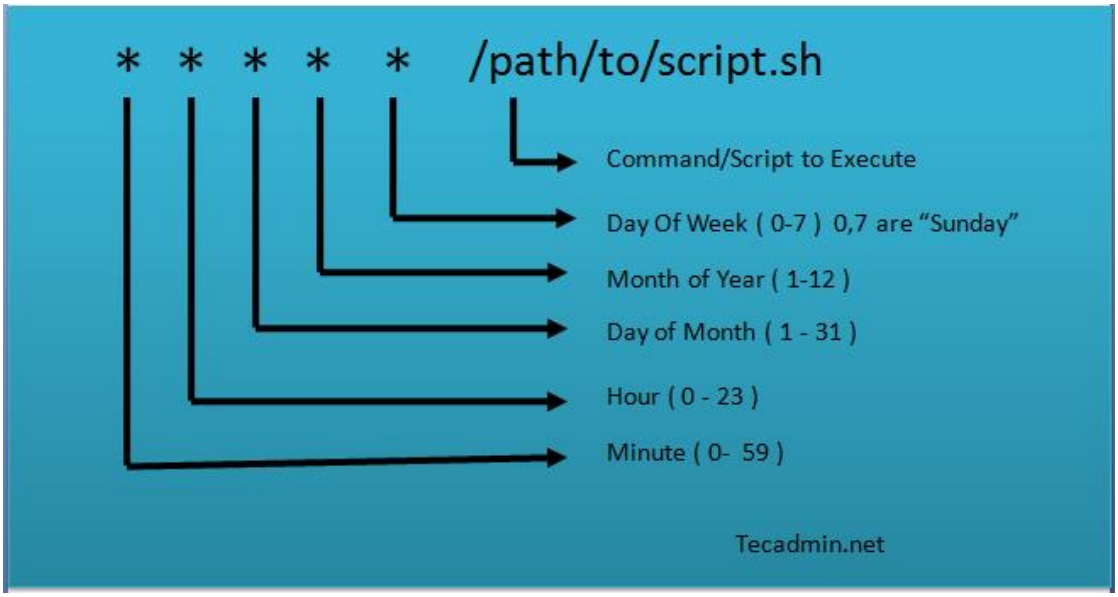
Ylemmän kuvan camera1.sh-tiedoston skriptin alkuun lisättiin lähteeksi source-komennolla juuri luotu livestreamkeys.sh (Kuva 19). Tiedosto camera1.sh pystyi tällä hakemaan livestreamkeys.sh-tiedoston striimausavainmuuttujat. Jotta camera1.sh-tiedosto pystyi käyttämään ffmpeg-komentoa, siinä täytyi olla määriteltynä ohjelman tarkka sijainti. Videokuvan vastaanottoa, muuntoa ja uudelleenohjausta varten tehtiin kuvan mukaiset määrityksiä skriptille.

Valinnalla -re FFmpeg asetettiin lukemaan saadun videokuvan natiivilla kuvanpäivityksellä. -rtsp_transport tcp -valinnalla FFmpeg asetettiin siirtämään videokuvaa TCP-protokollalla. -i-valinnalla määriteltiin sisääntulokanava, joka oli tässä tapauksessa ensimmäisen kameran RTSP-linkki. Vaikka kameroista saatu videokuva oli äänetöntä, se täytyi silti määritellä, jotta kuvaa pystyi lähettämään eteenpäin YouTube live-tapahtumaan. -f lavfi -valinnalla määritettiin sisääntulevan äänen muodoksi lavfi. -i-valinnalla määritettiin näytteenottotaajuudeksi 44100 Hz. -c:a-valinnalla määritettiin äänen käsittelyyn käytettäväksi encoder-ohjelmaksi libmp3lame. -ab-valinnalla määritettiin äänen lähteväksi bittinopeudeksi 128 kbit/s ja -ar-valinnalla äänen lähteväksi näytteenottotaajuudeksi 44100 Hz. -c:v copy -valinnalla saatu videokuvan kopioitiin sellaisenaan siirtämistä varten. -threads-valinnalla määriteltiin käytettävien prosessorien säikeiksi kaksi. -bufsize-valinnalla määriteltiin puskurointikooksi 512 kbit. -f flv -valinta määritti lähtevän videon muodoksi flv. Lopuksi määriteltiin

muunnetun videokuvan kohdelinkki, joka oli YouTube live-tapahtumasta saatu RTMP-linkki. Linkin lopussa olevan striimausavaimen tilalle laitettiin muuttuja $livestreamkey1, jonka tiedot haetaan livestreamkeys.sh-tiedostosta.

Valmiin skriptin toimivuutta testattiin kirjoittamalla komentokehotteeseen skriptin tarkka sijainti. Toimivasta skriptistä tehtiin kopiot toisille kameroille cp-komennolla. Kopioiduista skripteistä muutettiin tiedostonimi vastaamaan kameroiden numeroita. Lisäksi skriptien channel- ja livestreamkey-numerot vaihdettiin myös vastaamaan kameroiden numeroita.

Ennen skriptien ajon automatisointia kameraskriptit lukittiin, jotta samat skriptit eivät käynnistyisi vahingossa päällekkäin edellisten ollessa päällä jo taustalla. Skriptien ajo päällekkäin vei ylimääräisiä resursseja sekä palvelinlaitteelta että kameroilta. Skriptit lukittiin luomalla tyhjä lock-tiedosto, jolla oli käynnistysoikeus, jokaista kameraa kohden. Tiedostot käynnistyvät samanaikaisesti skriptien ajon aikana estäen päällekkäisiä ajoja, mikäli lock-tiedostot olivat silloin käynnissä. Skriptit laitettiin ajoon palvelimen taustalle cronin avulla. Crontab avattiin komennolla crontab -e.



Kuva 20. Ajan esitysmuoto crontabissa

Crontabin toimivuus perustui viisiosaiseen aikakenttään ja käynnistettävään komentoon, kuten ylemmässä kuvassa on esitetty (Kuva 20). Aikakenttä määritti komennon käynnistyksen ajankohdan. Aikakentästä ensimmäinen parametri vastasi minuuttia, toinen tuntia, kolmas päivää, neljäs kuukautta ja viides

viikonpäivää. Tähdellä merkattiin parametrit, joita ei tarvinnut huomioida. (Peltomäki & Linjama 1999, 370.) Projektin kannalta tärkein määritettävä osa oli tunti, jolla asetettiin skriptit käynnistymään kello 6.00 ja 18.00. Skriptin tuli käynnistyä pariin kertaan päivässä signaalikatkojen varalta.

```
PATH=/home/kytonummi/bin/ffmpeg-scripts

#* * * */1 * root /usr/bin/flock -w 1 /home/kytonummi/bin/ffmpeg-scripts/camera1.lock /home/kytonummi/bin/ffmpeg-scripts/camera1.sh
#* * * */1 * root /usr/bin/flock -w 1 /home/kytonummi/bin/ffmpeg-scripts/camera2.lock /home/kytonummi/bin/ffmpeg-scripts/camera2.sh
#* * * */1 * root /usr/bin/flock -w 1 /home/kytonummi/bin/ffmpeg-scripts/camera3.lock /home/kytonummi/bin/ffmpeg-scripts/camera3.sh
#* * * */1 * root /usr/bin/flock -w 1 /home/kytonummi/bin/ffmpeg-scripts/camera4.lock /home/kytonummi/bin/ffmpeg-scripts/camera4.sh
#* * * */1 * root /usr/bin/flock -w 1 /home/kytonummi/bin/ffmpeg-scripts/camera5.lock /home/kytonummi/bin/ffmpeg-scripts/camera5.sh
#* * * */1 * root /usr/bin/flock -w 1 /home/kytonummi/bin/ffmpeg-scripts/camera6.lock /home/kytonummi/bin/ffmpeg-scripts/camera6.sh

#* * * */1 * root /usr/bin/flock -w 1 /home/kytonummi/bin/ffmpeg-scripts/camera1backup.lock /home/kytonummi/bin/ffmpeg-scripts/camera1backup.sh
#* * * */1 * root /usr/bin/flock -w 1 /home/kytonummi/bin/ffmpeg-scripts/camera2backup.lock /home/kytonummi/bin/ffmpeg-scripts/camera2backup.sh
#* * * */1 * root /usr/bin/flock -w 1 /home/kytonummi/bin/ffmpeg-scripts/camera3backup.lock /home/kytonummi/bin/ffmpeg-scripts/camera3backup.sh
#* * * */1 * root /usr/bin/flock -w 1 /home/kytonummi/bin/ffmpeg-scripts/camera4backup.lock /home/kytonummi/bin/ffmpeg-scripts/camera4backup.sh
#* * * */1 * root /usr/bin/flock -w 1 /home/kytonummi/bin/ffmpeg-scripts/camera5backup.lock /home/kytonummi/bin/ffmpeg-scripts/camera5backup.sh
#* * * */1 * root /usr/bin/flock -w 1 /home/kytonummi/bin/ffmpeg-scripts/camera6backup.lock /home/kytonummi/bin/ffmpeg-scripts/camera6backup.sh

0 6,18 * * * /sbin/shutdown -r +10
```

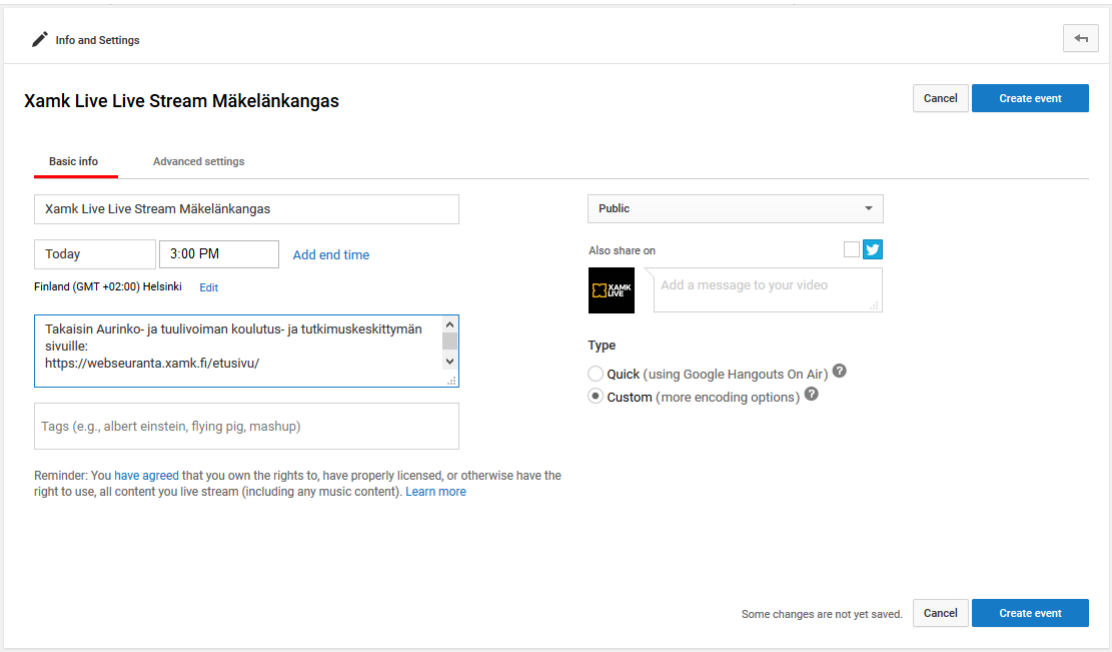
Kuva 21. Skriptien ajastus crontabilla

Ajettaviksi komennoiksi laitettiin flock-ohjelma, joka käynnisti kameraan liittyvän lock-tiedoston, ja kameran skriptitiedosto, kuten ylemmässä kuvassa on esitetty (Kuva 21). Jotta crontab pystyi löytämään flock-ohjelman ja skriptin, niille piti määrittää tarkat sijainnit. Valmiista rivistä kopioitiin rivit muille kameroille.

3.3.4 YouTube live-tapahtuma

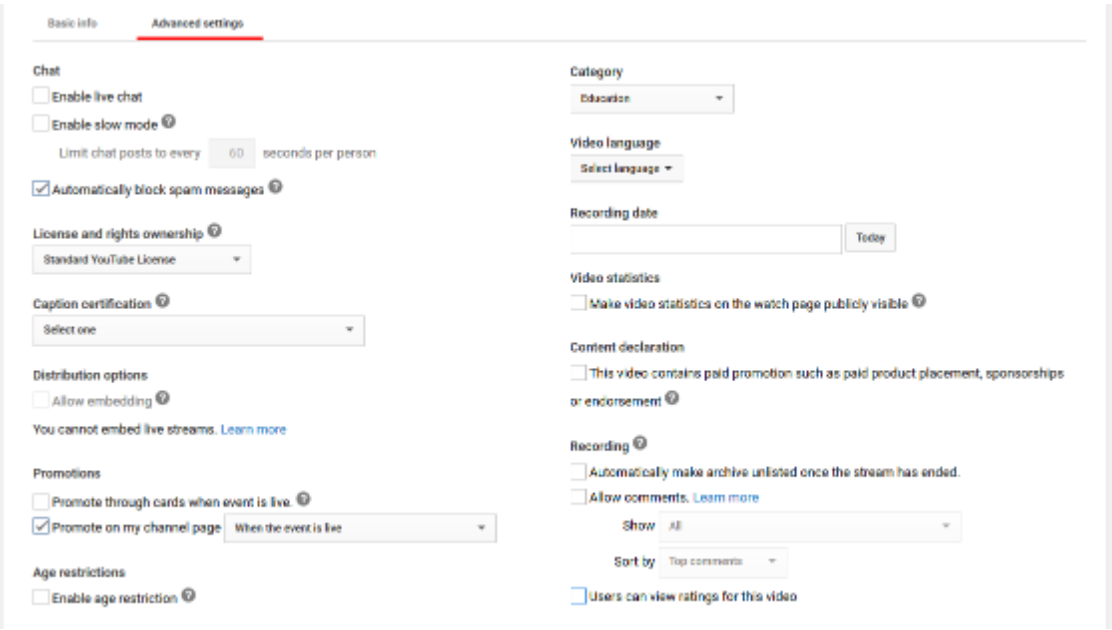
Suoratoistoa varten tarvittiin palvelu, joka pystyi esittämään reaaliaikaista videokuvaa ilmaiseksi. Palvelun kautta piti olla myös helppo seurata videokuvaa tarvitsematta ladata ylimääräisiä lisäosia selaimeen. YouTube live-tapahtuma-palvelu valittiin projektia varten juuri näiden ominaisuuksien takia. YouTubella oli yli miljardi käyttäjää (Frangoul 2018). Tämä takasi sen, että palvelu oli myös käyttäjälle luotettava ja tutun tuntuinen.

Palvelun käyttöä varten koulu loi YouTube-tunnuksen, joka jouduttiin verifioimaan, jotta YouTuben suoratoisto- ja live-tapahtuma-palveluita voitiin käyttää. Suoratoistaminen myös sallittiin YouTuben suoratoistoasetuksista. Salliminen antoi luvan lisätä uusia suoratoistoja tai live-tapahtumia. Suoratoiston tapahtumat-asetuksesta lisättiin uusi live-tapahtuma.



Kuva 22. Perustietojen lisääminen uuteen live-tapahtumaan

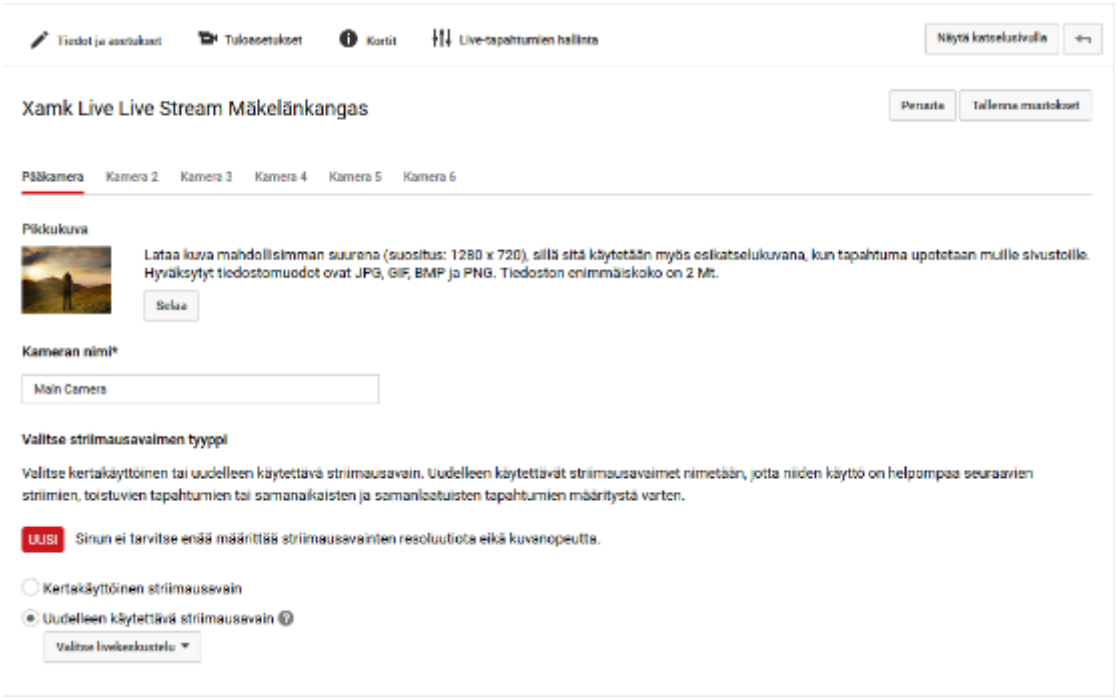
Perustietovälilehteen lisättiin tapatuman nimi, kuvaus ja aloitusajankohta ylemmän kuvan mukaisesti (Kuva 22). Kuvauksessa täytyi olla linkki takaisin Webseuranta-verkkosivuille. Tyypiksi valittiin muokattu, jotta palvelu pystyisi käyttämään FFmpegin muuttamaa videokuvaa.



Kuva 23. Live-tapahtuman lisäasetukset

Lisäasetuksetvälilehdestä poistettiin ylemmän kuvan mukaisesti live chatti, kommentit ja videotilastojen julkinen näyttö katselusivuilla, sillä niitä ei projektissa tarvittu (Kuva 23). Arvoksi valittiin Koulutus, sillä projektin reaaliaikaista videokuvaa käytettiin pääasiassa koulutuskäyttöön. Muut asetukset jätettiin

oletuksiksi. Tapahtuman tallentaminen avasi vaihtoehdot tuloasetusten valitsemiseen.



Kuva 24. Tuloasetukset

Tuloasetuksissa lisättiin kameroita ylemmän kuvan mukaan, kunnes niitä oli kuusi. Striimausavaimen tyypiksi valittiin Uudelleen käytettävä, jotta striimausavaimia ei tarvitsisi muuttaa palvelimen livestreamkeys.sh-tiedostossa suoratoiston mahdollisissa uudelleenkäynnistämisissä (Kuva 24). Jokaista kameraa varten lisättiin uusi livekeskustelu, jotka nimettiin kameroiden mukaan. Myös livekeskustelujen kuvauksiin laitettiin live-tapahtuman kuvaus. Jokaiselle livekeskustelulle valittiin suurimmaksi jatkuvaksi bittinopeudeksi ”500–2 000 kb/s (480p)”, sillä se sopi parhaiten FFmpegin lähettämään videokuvaan. Jokaiselle kameralle valittiin juuri luodut livekeskustelut ja encoder-ohjelmaksi Muu enkooderi, jotta palvelu pystyisi käyttämään FFmpegin muuttamaa videokuvaa. Tämä antoi suoratoistolinkit ensisijaiselle- ja varapalvelimelle sekä striimausavaimen, joka kopioitiin palvelimen livestreamkeys.sh-tiedostoon. Kopioinnin jälkeen palvelin käynnistettiin uudelleen videokuvan lähetyksen aloittamiseksi YouTuben live-tapahtumaan.

Live-tapahtumien hallinnasta tarkastettiin signaalintulo palvelimelta. Kaikkien kuuden kameran signaalin toimiessa saatu videokuva esikatseltiin toimivuu-

den varmistamiseksi. Esikatselun jälkeen tuloasetuksia ei voinut enää muuttaa. Lähetys aloitettiin kaikkien videokuvien toimiessa esikatselusoittimessa. Katselusivuilta kopioitiin jakolinkit Webseuranta-verkkosivun etusivulle. Alun perin oli tarkoituksena upottaa live-tapahtuma Webseuranta-verkkosivuille, mutta upottamismahdollisuus olisi vaatinut YouTube-tunnusten linkkauksen AdSense-tunnuksen kanssa ja mainosten käynnistämisen live-tapahtumassa (Kolm 2016). Projektin johdon kanssa päätettiin luopua upotuksesta.

Suoratoistoa varten todettiin hyväksi luoda varapalvelin signaalikatkojen varalta. Mikäli signaali katkeaisi live-tapahtumaan hetkeksi, live-tapahtuma päättyy (Rexer 2014). Uudelleenkäynnistämiseksi palvelin täytyisi käynnistää uudelleen, YouTubessa täytyisi luoda uusi live-tapahtuma ja Webseurannan verkkosivun linkit uuteen live-tapahtumaan tulisi vaihtaa. Varapalvelin lisäisi varmuutta signaaliin, pitäen live-tapahtuman aktiivisena, jotta uudelleenkäynnistämistä ei tarvitsisi. Varapalvelin voi sisältää samat asetukset kuin pääasiallinen palvelin, mutta kameroiden skriptien ulostulolinkit täytyy muuttaa muotoon ”rtmp://b.rtmp.youtube.com/live2?backup=1/<livestreamavain>”. Varapalvelimen kautta olisi voinut suoratoistaa samaa kamerakuvaa toiseen kertaan tai vaihtoehtoisesti esitallennettua videota luupissa. Projektin aikana ei kuitenkaan luotu varapalvelinta.

4 PÄÄTELMÄT JA LOPPUYHTEENVETO

Projektin tavoitteena oli luoda helposti seurattava verkkosivustokokonaisuus, josta tuli aurinko- ja tuulivoiman koulutus- ja tutkimuskeskittymä. Projektin työstämiseen annettiin hyvin avoimet kädet, sillä verrattavaa materiaalia sen tapaisesta verkkosivustokokonaisuudesta ei ollut paljon tarjolla. Tämä antoi vapaamman lähestymistavan suunnittelun kannalta, mutta ongelmatilanteiden kanssa täytyi usein itse keksiä omat ratkaisumenetelmät, mikä hidasti työtä merkittävästi. Itselläni oli jonkin verran kokemusta PHP:stä ja tietokannoista kurssien lisäksi aikaisemmasta projektityöstäni, mutta esimerkiksi WordPressistä ja reaaliaikaisten videokuvien uudelleenohjaamisessa minulla ei ollut mitään kokemusta. Kursseissa ei myöskään käyty tietokantojen optimointia luontivaiheessa kovin syvällisesti, mikä aiheutti projektissa ongelmia, kun mittaus-

tiedon määrä kasvoi suureksi. Kuitenkin esimerkiksi WordPressin oma dokumentaatio ja käyttäjien kirjoittamat neuvot Stack Overflowssa auttoivat monissa ongelmatilanteissa.

Projekti alkoi hitaasti, sillä työskentelymateriaalin saanti myöhästyi paljon ja niiden tilalla meille jaettiin paljon turhia dokumentteja ja tietoa, joita ei projektissa koskaan tullut käytettyä. Odottaessa materiaalin saantia meille jäi aikaa tutustua käytettäviin menetelmiin, mutta esimerkiksi tietokantaa ei voinut suunnitella ilman, että tiesi millaista mittaustietoa tulimme saamaan. Epävarmat tekijät hankaloittivat oikean tiedon etsimistä, mikä hukkasi paljon aikaa. Kun vihdoin saimme tarvittavat materiaalit, projekti eteni mielestäni kiitettävällä tahdilla, mutta alun perin suunniteltu puolen vuoden toteutus ei onnistunut enää.

Mittaustiedon tallentamisessa minulle tuotti eniten ongelmia vähäinen ymmärrykseni WordPressin kanssa. Oppiessani lopulta WordPressin rajapintatyökalut, lisäosien käytön ja cronin toimivuuden, sain kuitenkin mittaustiedon tallennuksen toimimaan oletetusti helpommin ja luotettavammin kuin mitä olisin siitä saanut ilman WordPressiä.

Mittaustiedon käsittelyn ja jakamisen optimoinnissa minun tuli tulkita toisen skriptiä. Onnistuin ajan kanssa tulkitsemaan sitä sen verran, että sain tietokannasta hakua optimoitua, mutta puutteelliset kommentit hidastivat tulkintaa. Tämä sai minut vahvemmin arvostamaan kommenttien tärkeyttä skripteissä. Sain kaiken kaikkiaan jaon optimoitumaan niin, että päivän, viikon ja kuukauden väliset ajankohtatiedot mittauksista tulivat diagrammeihin näkyviin lähes välittömästi, mutta valittaessa vuoden mittaustiedot hidastivat selainta jonkin verran tietoa ladattaessa. Projektin aikana en löytänyt keinoa nopeuttaa varsinkaan vuoden mittaustiedon hakua entisestään.

Reaaliaikaisen videokuvan esityksen toteuttamisessa hankaloitti koulun henkilökunnan kesäloma, mikä rajoitti palvelinlaitteiden ja käyttöjärjestelmien saantia. Tein koulun vanhasta koneesta väliaikaisen palvelimen, mikä ei ollut omasta mielestäni yhtä hyvä vaihtoehto kuin virtuaalipalvelin, koska tekemieni tietojen kopiointi toiseen palvelimeen oli hankalaa. Kuitenkin lopulta saatuani

virtuaalipalvelimen työtä varten, pystyin asentamaan FFmpegin ja tarvittavat encoder-ohjelmat ilman tarpeettomia asennuksia, joita tein kokeillessani.

Projekti opetti minulle ongelmien ratkontaa varsinkin ison tietomäärän käsittelyn kanssa ja hyviä optimointitapoja. Näitä oppeja pystyn mahdollisesti käyttämään tulevaisuudessa suunnitellessani uusia ohjelmistojärjestelmiä, joissa optimoiminen on hyvin tärkeää.

LÄHTEET

Assembla s.a. Configuring Certificate-Based Authentication. WWW-dokumentti. Saatavissa: https://cornerstone.assembla.com/cornerstone/helpbook/pages/getting-started/client-certificates.html [viitattu 4.4.2018].

Casey, L. 2017. ffmpeg, The Ultimate Swiss Army Knife. Blogi. Päivitetty 17.8.2017. Saatavissa: https://www.caseyliss.com/2017/8/17/ffmpeg [viitattu 15.3.2018].

Findikaattori. 2017. Uusiutuvat energialähteet. WWW-dokumentti. Päivitetty 8.12.2017. Saatavissa: http://findikaattori.fi/fi/89 [viitattu 14.3.2018].

Flashphoner. 2017. 7 ways to stream RTSP on the page. Blogi. Päivitetty 5.6.2017. Saatavissa: https://flashphoner.com/7-ways-to-stream-rtsp-on-the-page/ [viitattu 5.4.2018].

Frangoul, A. 2018. With over 1 billion users, here's how YouTube is keeping pace with change. Blogi. Päivitetty 14.3.2018. Saatavissa: https://www.cnbc.com/2018/03/14/with-over-1-billion-users-heres-how-youtube-is-keeping-pace-with-change.html [viitattu 6.4.2018].

Guzel, B. 2010. Scheduling Tasks with Cron Jobs. Blogi. Päivitetty 26.1.2010. Saatavissa: https://code.tutsplus.com/tutorials/scheduling-tasks-with-cron-jobs--net-8800 [viitattu 4.4.2018].

Haikala, I. & Märijärvi, J. 2004. Ohjelmistotuotanto. 10. painos. Helsinki: Talentum.

Hakala, M., Vainio, M. & Vuorinen, O. 2006. Tietoturvallisuuden käsikirja. Porvoo: WS Bookwell.

Hernandez, M. 2000. Tietokannat – suunnittelu käytännössä. Jyväskylä: Gummerus.

Hong, K. s.a. FFmpeg on Linux. WWW-dokumentti. Saatavissa: http://www.bogotobogo.com/VideoStreaming/ffmpeg_transcoding_audio_video_demuxing_muxing.php [viitattu 15.3.2018].

Hovi, A. 2004. SQL-opas. Jyväskylä: Docendo.

Kolm, P. 2016. Youtube live stream event embedding not allowed. Keskusteluryhmän artikkeli. Päivitetty 10.8.2016. Saatavissa: https://productforums.google.com/d/msg/youtube/tiz7aETkcpY/K6uKfnUABwAJ [viitattu 20.3.2018].

Laukkarinen, R. 2017. WordPress-sivut – räätälille vai markettiin? Blogi. Päivitetty 2.8.2017. Saatavissa: https://www.dude.fi/wordpress-sivut-raatalille-vai-markettiin [viitattu 14.3.2018].

Maruthamuthu, M. 2016. Install / Enable Nux Dextop Repository on RHEL, CentOS & SL. Blogi. Päivitetty 31.1. 2016. Saatavissa: https://www.2daygeek.com/install-enable-nux-dextop-repository-on-centos-rhel-scientific-linux/

[viitattu 5.4.2018].

McFarlin, T. 2013. Properly Setting Up WordPress Cron Jobs. Blogi. Päivitetty 18.7.2013. Saatavissa: https://tommcfarlin.com/wordpress-cron-jobs/ [viitattu 4.4.2018].

Meloni, J. 2003. MySQL : Trainer Kit. Helsinki: Edita.

MySQL. 2018. 13.2.2 DELETE Syntax. WWW-dokumentti. Päivitetty 4.4.2018. Saatavissa: https://dev.mysql.com/doc/refman/5.7/en/delete.html [viitattu 5.4.2018].

Peltomäki, J. & Linjama, T. 1999. Linux : Linux-käyttäjän peruskirja. Porvoo: WSOY.

Pitkänen, A. 2016. Miksi tehdä kotisivut WordPressillä? Blogi. Päivitetty 25.2.2016. Saatavissa: https://www.digimarkkinointi.fi/blogi/miksi-tehda-kotisivut-wordpressilla [viitattu 14.3.2018].

Rexer, B. 2014. Preventing Youtube from ending event if stream stops. Keskusteluryhmän artikkeli. Päivitetty 11.7.2014. Saatavissa: https://productforums.google.com/d/msg/youtube/zDYo-m_XPes/H9b2tbt8_YcJ [viitattu 6.4.2018].

Sarkuni, S. 2014. How I Write SQL, Part 1: Naming Conventions. Blogi. Päivitetty 16.2.2014. Saatavissa: https://launchbylunch.com/posts/2014/Feb/16/sql-naming-conventions/ [viitattu 4.4.2018].

Suleman, M. 2017. 5 Free GUI For FFmpeg. WWW-dokumentti. Päivitetty 23.4.2017. Saatavissa: http://www.ilovefreesoftware.com/23/featured/free-gui-ffmpeg.html [viitattu 15.3.2018].

Xamk s.a. Aurinko- ja tuulivoiman koulutus- ja tutkimuskeskittymä. WWW-dokumentti. Saatavissa: https://www.xamk.fi/tutkimus-ja-kehitys/aurinko-ja-tuulivoiman-koulutus-ja-tutkimuskeskittyma/ [viitattu 14.3.2018].

KUVALUETTELO

Kuva 24. Tuloasetukset.