Opinnäytetyö (AMK)

Tietotekniikan koulutusohjelma

Sulautetut ohjelmistot

2017

Sami Lindgren

LAITEREKISTERI

TURKU AMK
TURKU UNIVERSITY OF
APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tietotekniikan koulutusohjelma | Sulautetut ohjelmistot

2017 | 26

Jari-Pekka Paalassalo | Jimmy Lucchesi

Sami Lindgren

LAITEREKISTERI

Opinnäytetyön aiheena oli suunnitella ja ohjelmoida laiterekisteri, jonka tiedot tallentuvat tietokantaan. Laiterekisterin ohjelmointi toteutettiin Javalla ja tietokanta Linux-pohjaisena MySQL-tietokantana. Laiterekisterin tehtävänä on toimia helppona käyttöliittymänä tietokannan tietojen syöttämisessä, päivittämisessä sekä selaamisessa.

Lisäksi laiterekisteri mahdollistaa tietokantaan Arduino-pohjaisilla mittareilla kerättävän lämpötiladatan selaamisen ja tulostuksen.

Opinnäytetyönä tuloksena syntynyt sovellus on alustavasti otettu yrityksessä käyttöön. Lopputestauksen ja mahdollisesti vaadittavan validoinnin jälkeen se korvaa lopullisesti vanhan Excel-pohjaisen version.

ASIASANAT:

laiterekisteri, Arduino, ohjelmointi, tietokanta, sovellus

BACHELOR´S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information Technology | Embedded Software

2017 | 26

Jari-Pekka Paalassalo | Jimmy Lucchesi

Sami Lindgren

EQUIPMENT REGISTER

The aim of the thesis was to design and program an equipment register which saves the data inserted to a database. The program language used for the equipment register was Java and the database was actualized with MySQL server running on Linux. The function of the software is to act as an easy-to-use graphical user interface to insert, modify and browse the information recorded to the database.

Additionally the equipment register enables user to browse and print out data collected to the database by Arduino based digital thermometers.

The software formed as a result of the thesis has been implemented to be used in the company. After the final testing phase and the adding of possible additional features it will permanently replace the old Excel-based version.

KEYWORDS:

Equipment register, Arduino, programming, database, application

SISÄLTÖ

KUVAT

KÄYTETYT LYHENTEET

AWT	Abstract Window Toolkit, on Javan alkuperäinen piirto- ja graafisen käyttöliittymän toteuttava kirjasto.
CFR	Code of Federal Regulations on kodifikaatio USAn säännöksistä ja regulaatioista.
DNS	Domain Name System toimii Internetin nimipalvelujärjestelmänä ja muuntaa verkkotunnukset IP-osoitteiksi.
Ethernet	Verkkoteknologioiden perhe sekä protokolla, jota käytetään pakettikytkentäisissä lähiverkoissa ja kaupunkiverkoissa
FDA	U.S. Food and Drug Administration, Yhdysvaltain virasto joka laatii säädökset Yhdysvaltojen markkinoille.
IDE	Integrated development environment, ohjelmoijien työkaluksi suunniteltu ohjelma
ISO	International Organization for Standardization, kansainvälisiä standardeja tuottava järjestö.
JDBC	Java Database Connectivity, Javan rajapinta joka määrittelee miten asiakasohjelma pääsee käsiksi tietokantaan
LAMP	Linux Apache MySQL PHP, WWW-palvelimen muodostamiseen tarkoitettu kokoelma ohjelmia
MAC	Media Access Control on osajärjestelmä joka hoitaa verkon liikennöinnin ja varaamisen.
MDR	Medical Devices Regulation, EU:n lainsäädäntö, jossa kerrotaan miten lääkinnällisiä laitteita tulee valmistaa.
PHP	Hypertext Preprocessor on ohjelmointikieli joka on suunnattu web-kehitykseen.
PoE	Power over Ethernet, standardoitu systeemi jolla sekä virta, että data kulkee samaa parikaapelia pitkin.
URL	Uniform Resource Locator, web-osoite joka kertoo sen paikan tietokoneverkossa.

1 JOHDANTO

Pienet yritykset hyödyntävät Exceliä mitä moninaisimpiin tarkoituksiin. Yrityksen kasvaessa myös tietomäärät kasvavat ja Excelin käyttö tiedon käsittelyyn luo lähes hallitsemattomia kokonaisuuksia. Myös tiedostojen käyttäminen vaikeutuu yrityksen kasvaessa, kun useat ihmiset joutuvat muokkaamaan samaa tiedostoa eikä tämä onnistu yhtäaikaisesti. Lisäksi uudemmat Excel-versiot päivittävät tiedostomuodot ja voivat näin luoda ongelmia vanhempia versioita käyttäessä.

Tämän opinnäytetyön tarkoituksena on kehittää uusi tietokantaan pohjautuva ohjelmisto laiterekisterille sekä laite, joka mittaa lämpötiloja erilaisista kohteista (työtilat ja laitteet). Työn toimeksiantaja on lääkinnällisiä laitteita valmistava yritys BonAlive Biomaterials Oy. Bioalan yrityksissä laitteiden hallinta on osa viranomaisvaatimuksia ja siten rekisterin jatkuva käyttövalmius ja toimivuus on erittäin tärkeää, joka varmistetaan ohjelman validoinnilla.

Ohjelmiston tavoitteena on mahdollistaa rekisterin nopeampi toiminta sekä useamman yhtäaikaisen käyttäjän mahdollisuus muokata rekisterin tietoja. Ohjelmisto on pyritty luomaan helppokäyttöiseksi ja minimoimaan mahdolliset käyttäjien virheet esim. varmistamalla pakollisten kenttien täyttö sekä kokoamalla jäljitysketju automaattisesti. Rekisteriä pitää myös voida käyttää ilman tietoteknistä osaamista. Koulutus ja koulutusmateriaali sovelluksen käyttöön täytyy luoda tämä huomioiden. Sovellus korvaa vanhan Excel-pohjaisen menettelyn.

2 TUTKIMUSTYÖ

Laiterekisteri vaati graafisen käyttöliittymän helpottamaan sen jokapäiväistä käyttöä. Koska sulautettujen ohjelmistojen puolella graafisia käyttöliittymiä ei olla tehty, täytyi selvittää miten tällaisen luominen Javalla onnistuu. Javasta löytyi graafisen käyttöliittymän luontiin AWT sekä Swing -kirjastot, joita lähdettiin tutkimaan tarkemmin.

Tämän lisäksi ohjelman piti noudattaa lääkinnällisten laitteiden valmistajille määrättyjä viranomaisvaatimuksia, jotka eivät olleet ennestään tuttuja. Tämän kirjoitushetkellä vaatimuksena oli täyttää FDA:n CFR Title 21 Part 11, joka sisältää Yhdysvaltojen viranomaisten vaatimuksia sähköisille dokumenteille (U.S. Food & Drug Administration 2017). ISO 13485:2016 Medical Devices – Quality Management Systems on kansainvälisen standardeja ylläpitävän järjestön vaatimuksia, joita monien valtioiden viranomaiset vaativat lääkinnällisten laitteiden valmistajia noudattamaan. ISO 13485:2016 sisältää vaatimuksia myös tietokoneohjelmistoille. Vastaavat vaatimukset löytyvät myös mm. Yhdysvaltojen laista CFR Title 21 Part 820 Quality System Regulation. Tarkempia ohjeita ohjelmistojen validoinnille löytyy esimerkiksi FDA:n General Principles of Software Validation; Final Guidance for Industry and FDA Staff ohjeistuksesta. Lääkinnällisille laitteille on Euroopan Unionin alueella oma direktiivi, joka korvataan tulevaisuudessa uudella asetuksella. Euroopan Parlamentti hyväksyi tämän asetuksen 5.4.2017. Tämä MDR-säädös astuu voimaan toukokuussa 2017 ja sillä on kolmen vuoden siirtymäaika. Asetuksessa ei ole suoria vaatimuksia kehitettävälle sovellukselle.

2.1 AWT- ja Swing-kirjastot

AWT on Javan alkuperäinen graafisen käyttöliittymän toteuttava kirjasto, joka käyttää käyttöjärjestelmän omia käyttöliittymäkomponentteja, jonka vuoksi sillä toteutetut ohjelmat näyttävät erilaisilta eri käyttöjärjestelmissä. Swing puolestaan käyttää Javan omia käyttöliittymäkomponentteja, ja se toimii mahdollisimman samalla tavalla käyttöjärjestelmästä huolimatta (Oracle Docs 2017). Koska jokainen Swing- komponentti kuitenkin pohjautuu AWT:hen, on sen mahdollista kytkeytyä käyttöjärjestelmän käyttöliittymäkomponenttien rakenteisiin ja seurata käyttäjän toimia kuten hiiren liikkeitä tai klikkauksia.

Ennen Java 6 Update 12:ta AWT ja Swing -komponenttien sekoittaminen aiheutti ongelmia, sillä AWT-komponentit ilmestyivät Swing-komponenttien päälle huolimatta niiden määritellystä järjestyksestä z-akselilla. Kyseinen päivitys kuitenkin korjasi tämän ongelman ja nykyään komponentteja voi rajoitetusti sekoittaa keskenään (Oracle 2017).

2.2 Viranomaisvaatimukset

2.2.1 FDA CFR Title 21 Part 11 -laki

CFR Title 21 Part 11 koskee sähköisiä dokumentteja ja sähköisiä tallenteita sekä niiden sähköisiä allekirjoituksia tai allekirjoituksia sähköisiin dokumentteihin. Yhdessä CFR Title 21 Part 820:n kanssa tämä laki asettaa vaatimuksia tallenteiden luomiselle, muokkaamiselle, ylläpidolle, arkistoinnille, palautukselle ja siirtämiselle. Siinä tapauksessa, että sähköiset allekirjoitukset ja niitä vastaavat sähköiset tallenteet ovat näiden vaatimusten mukaisia, sähköinen allekirjoitus tulkitaan samanarvoiseksi käsinkirjoitetun kanssa (U.S. Food & Drug Administration 2017).

Sovelluksen, joka sisältäen laitteiston ja ylläpidetyn dokumentaation, täytyy olla saatavilla FDA-tarkastuksissa. Sähköisiä tallenteita voidaan myös tarvittaessa toimittaa FDA:lle esim. myyntilupahakemuksen tai raportoinnin yhteydessä.

Sähköisiä allekirjoituksia käytettäessä Part 11 vaatii esimerkiksi käyttäjätunnuksen yhteydessä salasanan, joille täytyy varmistaa riittävät kontrollit niiden turvallisuuden ja oikeellisuuden osalta (U.S. Food & Drug Administration 2017). Tällaisia kontrolleja ovat esimerkiksi käyttäjätunnusten ja salasanojen yhdistelmän yksiöllisyys ja salasanojen säännöllinen vanhentuminen. Lisäksi vaaditaan proseduuri valtuutusten peruuttamiselle mahdollisissa katoamis- ja varkaustapauksissa.

2.2.2 ISO 13485:2016 -standardi

ISO 13485-standardin uusin versio (2016) vaatii validoimaan laadunhallintajärjestelmään liittyvät tietokoneohjelmistot. Validoinnit on tehtävä ennen käyttöönottoa ja jos tarpeen myös sen jälkeen kun sovellukseen on tehty muutoksia. Validoinnin on oltava tarkoituksenmukainen sovelluksen käyttöön liittyviin riskeihin nähden.

2.2.3 General Principles of Software Validation -ohjeistus

FDA:n General Principles of Software Validation ohjeistaa, miten tietyt laatujärjestelmän määräykset koskevat ohjelmistoja. Dokumentti listaa ominaisuuksia, jotka FDA katsoo hyväksyttäviksi ohjelmistojen validoinnissa, mutta ei aktiviteetteja tai tehtäviä joiden täytyy joka tapauksessa noudattaa lakia. Ohjeistus muun muassa kertoo, miten hyvä ohjelmistokehitys sisältäen esimerkiksi suunnittelua, verifikaatiota sekä testausta, auttaa ohjelmiston validoinnissa. Ohjelmiston käyttötarkoituksen ja sen luomien turvallisuusriskien perusteella ohjelmistokehittäjän täytyy määritellä minkälaisia tekniikoita ohjelmiston kehityksessä yhdistellään sekä paljonko työtä tämä vaatii. Jos ohjelmiston kehittää joku muu kuin lääkinnällisten laitteiden valmistaja itse, ei ohjelmistokehittäjä itse ole välttämättä vastuussa FDA-vaatimusten täyttämisestä. Tällaisissa tapauksissa osapuoli, joka joutuu FDA-vaatimuksia noudattamaan, on vastuullinen määrittelemään, mitä ylimääräisiä toimia joudutaan tekemään, jotta ohjelmisto saadaan validoitua tarkoitettuun käyttöön.

3 TYÖSSÄ KÄYTETYT TYÖKALUT

3.1 Ohjelmointiympäristö

Sovellus toteutettiin Javalla, joka on Sun Microsystemsin kehittämä (myöhemmin Oracle Corporation ostama) ohjelmointikieli. Koska Java on laitteistoriippumaton eikä se Swing-kirjastoa käytettäessä käytä käyttöjärjestelmän omia graafisia komponentteja, saatiin laiterekisteristä toimiva kokonaisuus kaikille yleisesti käytössä oleville käyttöjärjestelmille.

Alussa graafisen käyttöliittymän luomisessa haluttiin saada kaikki sen komponentit paikalleen mahdollisimman helposti. Edellämainitun vuoksi ohjelmointiympäristöksi valittiin Eclipse, johon on olemassa WindowBuilder -niminen liitännäinen. WindowBuilder on koostettu SWT Designerista ja Swing Designerista ja sillä on erittäin helppo luoda graafisia käyttöliittymiä Java-sovelluksiin, tarvitsematta itse kirjoittaa koodia (Eclipse 2017). Kuitenkin WindowBuilder on vain liitännäinen Eclipseen joten se ei ole tuettu jokaisessa versiossa, keväällä 2017 uusin Eclipsen versio, jossa liitännäinen on tuettuna, on Mars.2.

Eclipse myös helpottaa koodin kirjoittamista Javalla huomattavan paljon, sillä siitä löytyy esimerkiksi pikakomennot tarvittavien kirjastojen lisäämiselle, se näyttää parametrit funktioille, myös omien funktioiden kohdalla, sekä antaa informaatiota funktioista ohjelmointiympäristössä itsessään. Myös luokkien uudelleen nimeämisessä eli refaktoroinnissa Eclipse muuttaa kaikki tarvittavat kohdat koko projektissa ilman tarvetta käyttäjän toimille.

3.2 MySQL server ja Ubuntu Server

Tietokantojen vaihtoehtoja tutkiessa päädyttiin käyttämään MySQL serveriä. Tähän päätökseen vaikutti mahdollisuus asentaa kyseinen ohjelma sekä Windows, Mac OS että UNIX ympäristöön, sillä lopullisesta käyttöjärjestelmästä jolle tietokanta tultaisiin asentamaan ei ollut vielä tietoa tässä vaiheessa (MySQL 2017). Tämän lisäksi MySQL mahdollistaa usean yhtäaikaisen käyttäjän pääsyn tietokantaan, sekä pystyy käsittelemään suuren määrän tietoa nopeasti ja luotettavasti.

Tietokannan valitsemisen jälkeen, se piti asentaa jollekin käyttöjärjestelmälle. Koska lopullisesta käyttöjärjestelmästä ei ollut tietoa, päädyttiin MySQL server asentamaan Ubuntu server 12.04:lle. Tähän johti tekijän aiempi kokemus Ubuntu serveristä ja MySQL:n käytöstä kyseisellä käyttöjärjestelmällä, sekä käyttöjärjestelmän keveys verrattuna esimerkiksi Windows-ympäristöön, sillä testiympäristöä varten annetun tietokoneen laskentateho ja muistin määrät olivat matalat. Koska Ubuntussa yhdellä komennolla saa asennettua koko LAMP-ympäristön kokoonpano asennettiin sen palvelimelle, tähän johti tekijän aiempi kokemus Arduinon mahdollisuuksista lähettää tietoa tietokantaan, joka onnistuu helpoiten verkon yli PHP-skriptillä.

Lopulliseksi käyttöjärjestelmäksi tuli lopulta Ubuntu server 10.04.1, sillä saatiin selville, että yrityksestä löytyi jo sellainen palvelinkäyttöön asennettuna. Tämän lisäksi palvelimelle oli valmiiksi asennettu MySQL server muuta käyttöä varten, joten palvelimelle ei tarvinnu kuin luoda tietokanta sovellusta varten.

3.3 Arduino ja Arduino IDE

Mikrokontrollereita oli tarjolla lähes loputtomalta tuntuva määrä. Suurin osa niistä olisi soveltunut lämpömittarin tehtävään mainiosti, mutta kun tietoa täytyi siirtää verkon yli tietokantaan, soveltuvien mikrokontrollien määrä pieneni selkeästi. Näistä lähes jokaiseen olisi saanut ostettua erillisen modulaarisen piirilevyn, esim. Arduino kutsuu näitä nimellä Shields, mutta koska itse mittarin fyysisestä koosta haluttiin mahdollisimman pieni olisivat kyseiset tuotteet lisänneet mikrokontrollerin korkeutta huomattavasti. Listaa hieman karsittuna saatavuuden, hinnan ym. osalta jäljelle jäivät Raspberry Pi sekä Arduino Yun, joista molemmat tukevat myös PoE:a, mutta vain erillisen moduulin kautta.

Työtä tehtäessä Arduino Yun:lle PoE-moduulin saatavuus Suomessa oli huomattavasti Raspberry Pitä parempi, joten se valittiin työssä käytettäväksi mikrokontrolleriksi. Vaikka PoE-moduulin lisääminen nostaakin Arduinon fyysistä kokoa, se haluttiin optioksi, jos mittareita laitettaisiin paikkoihin, joissa sähköverkosta virran saaminen olisi työläämpää kuin ethernetin yli. Tämän vuoksi työssä käytettävät ethernet-kaapelit olivat vähintään cat5 luokitusta, sillä se on PoE-standardin oletus (wikipedia 2017).

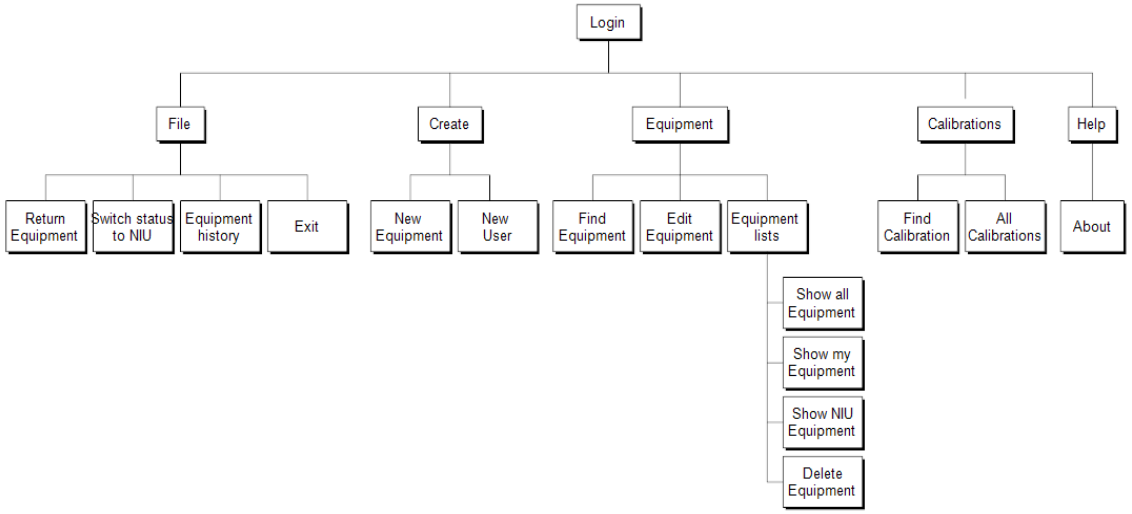
Koska käytössä oli Arduino, oli Arduino IDE:n valitseminen lähes itsestäänselvyys. Arduino IDE toimii kaikilla yleisesti käytössä olevilla käyttöjärjestelmillä, ja se tekee

koodin lähettämisestä sekä ajamisesta millä tahansa Arduino-mikrokontrollerilla helppoa (Arduino 2017).

4 TYÖN TOTEUTUS

4.1 Sovelluksen suunnittelu

Ennen laiterekisterin ohjelmoinnin aloitusta piti hahmottaa minkälainen sovelluksesta halutaan. Kuvassa 1 on suunnittelun aikana hahmoteltu kokonaisuus, jossa sovelluksen osat on merkitty. Keskeisimmiksi asioiksi valittiin helppokäyttöisyys ja käyttäjäystävällisyys, itse sovelluksen ulkonäölle ei laitettu suurta painoa, sillä sitä pystyisi muuttamaan haluttaessa myös jälkikäteen.



Kuva 1. Käyttöliittymän suunniteltu kokonaisuus.

Itselläni ei sovelluskehityksestä ole entuudestaan kokemusta, joten helppokäyttöisyyttä ja käyttäjäystävällisyyttä lähdettiin tavoittelemaan mahdollisimman monen asian automatisoinnilla tai tiettyjen tapahtumien estämisellä. Yksi tällainen tapahtuma on viranomaisvaatimusten edellyttämä kaikkien vaadittavien kenttien huomioiminen. Tyhjä kenttä tulkitaan tiedon puuttumiseksi ja siksi jokaisessa vaadittavassa kentässä on oltava jotakin sisältöä. Jos vaadittava informaatio ei ole sovellettavissa kyseiselle laitteelle, huomiointiin riittää pelkkä ”-” tai ”na”. Tämän vuoksi lähdettiin suunnittelussa siitä, ettei ohjelma anna tallentaa tietoja tietokantaan ennenkuin kaikissa kentissä olisi informaatiota. Koska välilyönnit näyttävät tietokoneella tyhjäksi jäteltyltä kentältä, rajattiin myös pelkästään niiden käyttö pois. Ohjelman kaatuminen ei myöskään anna vaikutelmaa helppokäyttöisyydestä ja tämän vuoksi sovelluksen täytyy pystyä

käsittelemään kaikki mahdolliset virheet sekä antaa sitä käyttävälle työntekijälle luettavissa oleva virheilmoitus.

4.2 Sovelluslogiikka

Sovellukselta edellytettiin vuorovaikutusta käyttäjän kanssa. Esimerkiksi tietokantaan tallentaminen nappia painamalla ei anna käyttäjälle minkäänlaista tietoa tehtävän onnistumisesta. Myös asioiden onnistuessa käyttäjän olisi hyvä saada selkeä ilmoitus, jotta ei jää epäselvyyttä etenikö kyseinen tehtävä halutulla tavalla. Tietenkään tätä ei kannata soveltaa asioihin, joissa käyttäjän toimet aiheuttavat sovelluksessa jonkinlaista näkyvää toimintaa, esimerkiksi uuden ikkunan aukeamisen. Myös liiallinen uusien ikkunoiden avaaminen ja siitä lopulta aiheutuva kaaos haluttiin välttää, joten lähes kaikki uudet ikkunat jotka sovelluksessa suunniteltiin aukeavan, myös sulkeutuvat itsekseen, kun käyttäjän on suorittanut niissä toimintonsa. Poikkeuksen tähän muodostavat muutamat ikkunat, jotka on suunniteltu niin, että käyttäjä voi suorittaa monta perättäistä toimintoa niissä. Tällaisia ikkunoita on esimerkiksi pääikkuna, joka sisältää alasvetovalikoita joiden alta löytyy ohjelman eri toiminnallisuudet sekä laitteiden lisäämiseen tarkoitettu ikkuna. Pääikkunan sammuttaminen myös sulkee kaikki avoinna olevat ikkunat ja kirjaa käyttäjän automaattisesti ulos ohjelmasta.

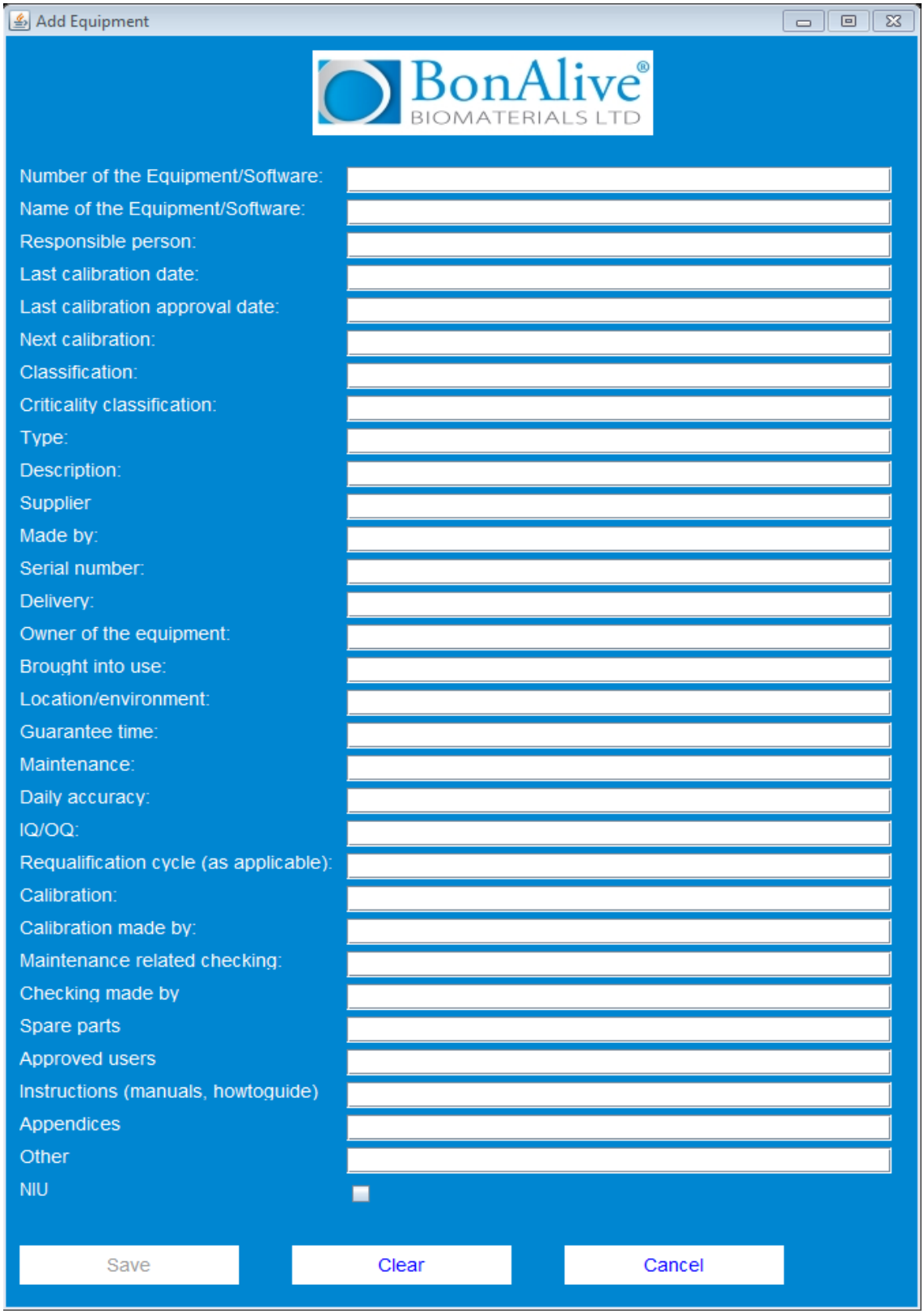
Sovelluksella yksittäisten käyttäjien identifiointi on yksi viranomaisvaatimuksista, joten vaikkakin tämä on yksinkertainen toteuttaa täytyy se silti löytyä ohjelmasta. Yksi erittäin tärkeä viranomaisvaatimus on vaatimus identifioida kaikki käyttäjät, joten sovelluksessa tulee olla proseduuri sisäänkirjautumiselle. Tämän lisäksi sovelluksen haluttiin pystyvän identifioimaan henkilö, joka on suorittanut muokkauksia tietokannan tietoihin (audit trail). Vaikka kirjanpito eri käyttäjien suorittamista muokkauksista olisi ollut helppoa, päätettiin asia suorittaa hieman eri näkökulmasta. Sen sijaan, että jokaisen käyttäjän toimia seurattaisiin, rajoitettiin käyttäjien oikeuksia niin etteivät he voi muokata muita tietoja kuin niitä, missä heidät on nimetty vastuuhenkilöksi. Näin parannettiin myös sovelluksen käyttäjäystävällisyyttä, sillä käyttäjät eivät voi vahingossa muokata vääriä tietoja muiden työntekijöiden vastuulla olevaan informaatioon. Tietojen selaamisoikeus kuitenkin säilyi kaikilla, joilla on tunnukset sovellukseen.

4.3 Sovelluksen ohjelmointi

4.3.1 Graafinen käyttöliittymä

Kun sovellusta lähdettiin ohjelmoimaan, aloitettiin muutamien tärkeimpien osien muodostamisesta käyttäen WindowBuilderia. Koska tarkoituksena oli luoda yksi pääikkuna josta olisi pääsy kaikkiin muihin toimintoihin, luotiin jokainen ikkuna omaksi luokakseen. Näin jokainen sovelluksen ikkuna pystyttäisiin luomaan sen kutsumalla luokan konstruktoria, ja samalla konstruktoriin voi syöttää mahdolliset parametrit jotka vaikuttavat ikkunan muodostamiseen. Näihin osiin kuuluivat muunmuassa pääikkuna, kirjautumisikkuna, tietojensyöttöön tarkoitettu lomake sekä uusien käyttäjien lisäykseen tehty lomake. Pääikkuna koostui yrityksen logosta, sekä pudotusvalikoista, joiden alle tulisi lisää ominaisuuksia. Aluksi pudotusvalikoista löytyi painikkeet sovelluksen sulkemiseen, uusien käyttäjien ja tiedon syöttöön tarkoitettujen lomakkeiden avaamiseen sekä tietokannan selaamisikkunan avaamiseen.

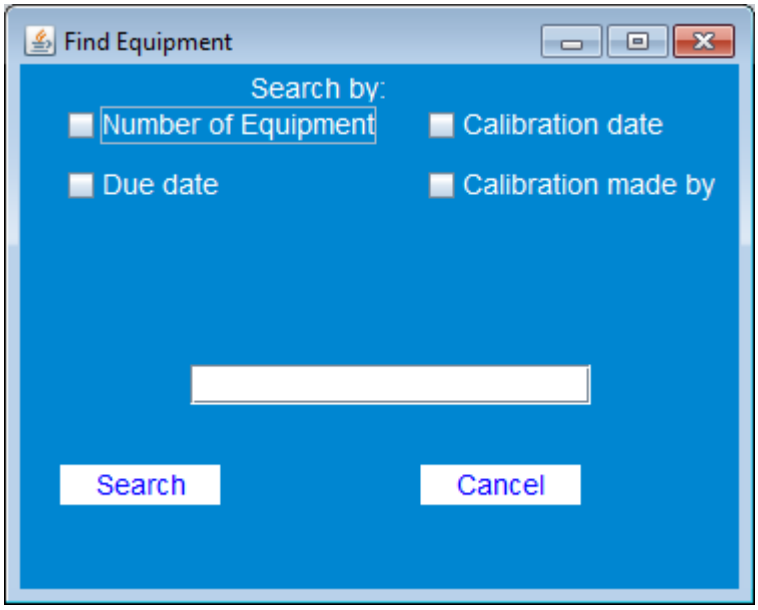
Kuvassa 2 näkyvään tiedonsyöttölomakkeeseen luotiin tekstinsyöttökentät kaikille halutuille kohdille ja näiden viereen tuli lyhyt kuvaus mitä kenttään kirjoitetaan. Lisäksi lomakkeesta löytyi painikkeet tallentamiselle, lomakkeen sulkemiselle ja lomakkeentekstikenttien tyhjentämiseen.



Kuva 2. Laiterekisterin tiedonsyöttölomake.

Käyttäjien luontiin rakennettu lomake muodostettin tiedonsyöttölomakkeen pohjalta, mutta tekstikenttien tyhjennys -painike jätettiin luomatta, sillä kenttiä tuli vain 4 kappaletta: käyttäjätunnus, sähköposti sekä salasana kahteen kertaan.

Myöhemmässä vaiheessa sovellukselle luotiin kirjautusmisikkuna, joka yksinkertaisuudessaan sisälsi vain kaksi tekstikenttää sekä sisäänkirjautumis- että sammutuspainikkeet. Viimeiseksi muodostettiin vielä kuvassa 3 näkyvä hakutoiminnon mahdollistava ikkuna.



Kuva 3. Hakutoiminnon ikkuna.

Ikkuna sisältää muutaman valintapainikkeen ja painikkeen sekä tekstikentän johon haettava informaatio syötettäisiin.

4.3.2 Ikkunoiden toiminnallisuudet

Sovelluksien osien toiminnallisuutta lähdettiin toteuttamaan pala kerrallaan edeten kronologisessa järjestyksessä. Täten ensimmäisenä järjestyksessä oli kirjautumisikkuna, koska kirjautumisen tiedot piti saada tietokannasta täytyi ensimmäisenä luoda tietokantayhteys. Tietokantayhteys toteutettiin Javan DriverManagerin getConnection()-funktiolla, joka löytyy kielen sql-kirjastosta. Parametreinä funktioon annetaan tietokannan URL, käyttäjänimi ja salasana. DriverManager on osa JDBC kirjastoa, joka tarjoaa rajapinnat tietokantayhteydelle,

lausekkeiden luomiseen, kyselyiden suorittamiseen sekä tallenteiden tarkasteluun ja muokkaamiseen. (tutorialspoint 2017). Painikkeiden toimintaan käytettiin AWT:n event-kirjastosta löytyvää ActionListeneriä, tarkemmin tämän actionPerformed-metodia, jota kutsutaan aina käyttäjän tehdessä toimia määritellyille objekteille. Kuvasta 4 nähdään miten kyseistä metodia voi soveltaa esimerkiksi painikkeiden käytössä, kuvassa login ja cancel ovat JButton luokan olioita.

```
public void actionPerformed(ActionEvent ae)
{
    if(ae.getSource().equals(login))
    {
        //code to execute if user clicks on Login button
    }

    else if(ae.getSource().equals(cancel))
    {
        //code to execute if user clicks on Cancel button
    }
}
```

Kuva 4. actionPerformed metodin käyttö.

Edellämainittuja funktioita käyttäen käyttäjän painaessa kirjautumispainiketta, sovellus pystyy ottamaan yhteyden tietokantaan, hakemaan sieltä käyttäjänimen yhteydestä salasanan ja testaamaan käyttäjän kirjoittamaa salasanaa tietokannasta löytyvään. Jos käyttäjänimeä ei tietokannasta löydy sovellus ilmoittaa virheestä käyttäjätunnuksessa tai salasanassa, samoin jos käyttäjänimen yhteydessä oleva salasana ei vastaa käyttäjän kirjoittamaa. Jos molemmat osat kuitenkin löytyvät tietokannasta, kutsutaan seuraavan ikkunan konstruktoria ja hävitetään kirjautumisikkuna. Koska käyttäjätunnusta tullaan tarvitsemaan myöhemmässä vaiheessa ohjelmaa, tallennetaan se tässä vaiheessa muuttujaan. Muuttujan arvon saa myöhemmässä vaiheessa luomalla LoginWindow-luokasta olion ja kutsumalla userName()-funktiota.

Koska pääikkunan tarkoitus oli vain kutsua muiden luokkien konstruktoreita tai suljettaessaan hävittämään kaikki muutkin ohjelman ikkunat. Tästä syystä ikkunan tekemisessä ei tarvinnut kuin soveltaa edellisessä luvussa 4.2.1 mainittua ActionListeneriä kaikkiin ikkunaan luotujen alasvetovalikoiden painikkeisiin.

Seuraavana vuorossa oli tietojensyöttölomake sekä käyttäjien syöttö-lomake. Käyttäjien syötön osalta tämä oli yksinkertainen tehtävä, sillä tarvittiin vain soveltaa Javan sql kirjaston Connection, Statement sekä PreparedStatement -objekteja ja niiden fuktioita syöttämään lomakkeen tekstikentistä saadut tiedot tietokantaan. Koska tässä vaiheessa huomattiin tarvittavan myös kirjautumisikkunassa käytetyn DriverManagerin funktioita, päätettiin tietokantayhteyksien muodostamisesta luoda oma luokkansa, jolloin samaa koodia ei tarvitse kopioida jokaiseen luokkaan jossa sitä tultaisiin tarvitsemaan. Molempien lomakkeiden kaikkien painikkeiden toiminnallisuudet luotiin taas ActionListeneriä käyttämällä. Koska tiedonsyöttölomakkeessa kaikissa kentissä täytyi olla tietoa, ennenkuin käyttäjä saisi tallennuspainikkeen käyttöönsä, täytyi muodostaa oma DocumentListener-rajapintaa käyttävä luokka.

```
class MyDocumentListener implements DocumentListener {
@Override
public void changedUpdate(DocumentEvent e) {
    change();
}

@Override
public void insertUpdate(DocumentEvent e) {
    change();
}

@Override
public void removeUpdate(DocumentEvent e) {
    change();
}
```

Kuva 5. DocumentListener-rajapinnasta löytyviä metodeja.

Kuvassa 5 nähdään metodit, joilla pystytään seuraamaan tässä tapauksessa tekstikentässä tapahtuvia lisäyksiä, poistoja tai muutoksia. Tämän lisäksi kenttien tiedot piti testata tietoa sisältäviksi. Pelkkä Javan equals(””) funktio ei riittänyt, sillä se antaisi mahdollisuuden syöttää välilyöntejä järjestelmään. Javasta kuitenkin löytyy trim() funktio, joka poistaa ylimääräiset välilyönnit tekstin alusta ja lopusta. Kuvassa 5 kutsuttavaan change()-funktioon on määritelty trim()- ja equals()-funktiota käyttäen kaikkien tekstikenttien sisältöä tarkkaileva tarkistus, joka onnistuu vain, jos kaikkiin kenttiin on lisätty tekstiä. Jos tarkitus onnistuu, se aktivoi tallennus-painikkeen, mutta epäonnistuessaan se lukitsee kyseisen painikkeen.

4.3.3 Tietokantarajapinta

Lukuunottamatta tietokannan selausikkunoita, muut sovelluksen osat muodostuivat pitkälti edellämainituista objekteista, eivätkä tuoneet uutta ohjelman toimivuuden kannalta. Selausikkunat kuitenkin toteutettiin käyttämällä JTable luokkaa, joka on osa Javan Swing kirjastoa. JTable mahdollistaa tiedon näyttämisen soluista koostuvana 2-ulotteisena taulukkona, samaan tapaan kuin esimerkiksi Microsoft Excel. Tietokantaa luotaessa kolumnien nimet lyhennettiin ymmärrettäviksi, mutta kuitenkin helpommin kirjoitettaviksi tietokantaan luotavien syöttöjen ja hakujen vuoksi. Tämän takia kolumnien otsikoita ei saanut suoraan tietokannasta, vaan ne piti kovakoodata sovellukseen. Koska sovellus ei ainakaan nykyisessä muodossaan salli uusien kenttien luontia tietokantaan, ei kolumnien otsikoiden kovakoodaus ollut merkittävä puute. Taulukon riveihin tuleva data saatiin haettua kuvassa 6 näkyvillä sisäkkäisillä silmukoilla, joista ulommainen haki tietokannasta halutun kohdan, sisempi tallensi kaiken informaation Vector<Object> tyyppiseen olioon, ja lopuksi ulompi silmukka lisäsi informaation yhtenä kokonaisuutena toiseen samantyyppiseen, rowData-nimiseen olioon.

```
while (rs.next())
{
    Vector<Object> row = new Vector<Object>(columncount);

    for (int j = 1; j < columncount; j++)
        {
            row.addElement(rs.getObject(j));
        }
    rowData.addElement(row);
}
```

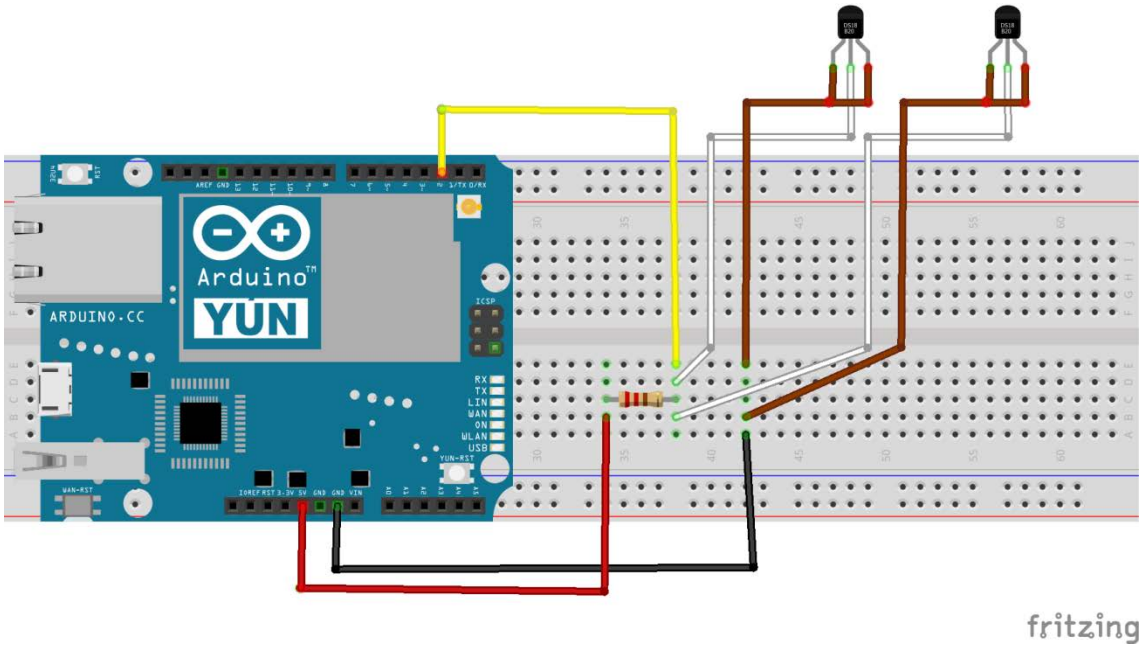
Kuva 6. Datan hakeminen tietokannasta taulukon riveihin.

Kuvan 6 muuttujat joita ei ole alustettu kuvassa ovat rs, joka on ResultSet rajapinnan olio, ja se saa arvokseen Statement-rajapinnasta löytyvän executeQuery()-funktion arvon. Funktioon on parametreina annettu käyttäjän valitsema SQL-query. Columncount on int tyyppinen muuttuja, jonka arvo on ResultSetMetaData-rajapinnan getColumnCount()-funktion arvo lisättynä yhdellä. RowData ja aiemmin kolumnien otsikoista luotu Vector<Object> olio syötettiin parametreina DefaultTableModel-olioon joka taasen annettiin parametrina JTable luokasta luotuun ”table”-nimiseen olioon. Table-olio lisättiin JScrollPanella muodostettuun olioon, sillä JScrollPanea käyttämällä saadaan tarvittaessa muodostettua horisontaalinen sekä vertikaalinen vierityspalkki.

Lopuksi ikkunaan lisättiin vielä painike tulostukselle. Painike kutsuu yksinkertaisesti JTable:sta löytyvää print()-funktiota joka mahdollistaa koko taulukon tulostuksen. Näin muodostettu taulukko kuitenkin venyy horisontaalisesti melko pitkäksi, joka hankaloittaa yksittäisen laitteen tietojen katselua ja tulostusta. Tästä syystä käyttäjän hakiessa vain yhtä laitetta sen laitenumerolla, joka identifioi yksittäisen laitteen, taulukko kääntyy niin että kolumneja on kaksi ja informaation otsikot rivitetään vasemmanpuoleiseen kolumniin ja itse informaatio oikeanpuoleiseen. Tällä tavalla yhden laitteen tiedot mahtuvat tulostettaessa yleensä yhdelle A4-kokoiselle paperille, ja niiden selaamisen tietokoneelta helpottuu huomattavasti.

4.4 Mikrokontrollerin ohjelmointi ja tietokantayhteys

Arduinoon yhdistettynä lämpötila-anturina käytettiin Dallasin DS18B20 -lämpötila-antureiden johdollista versiota. Koska jokaisella kyseisellä anturilla on oma uniikki tunnisteensa, ne voidaan kytkeä yhteen Arduinon porttiin, kuten kuvassa 7 näkyy, käyttäen OneWire.h sekä DallasTemperature.h -kirjastoja.



Kuva 7. Arduino Yun kytkentä.

Näin anturien mittaaman datan pystyi lukemaan getTempCByIndex() funktiolla, jonka parametrina annetaan anturin numero, alkaen nollasta. Saatu lämpötila-arvo tallennettiin float -tyyppiseen muuttujaan.

Arduino ei itsessään pysty syöttämään dataa tietokantaan, joten tietokantayhteys jouduttiin toteuttamaan mutkan kautta. Koska Arduinolla pystyy luomaan http-pyyntöjä Ethernet.h-kirjastoa käyttäen, yhdistettiin Arduino samaan sisäverkkoon tietokantapalvelimen kanssa. Tähän käytettiin Ethernet.begin()-funktiota, jonka parametreina annettiin Arduinon MAC sekä haluttu ip-osoite sekä vapaaehtoinen DNS-osoite. Kun Arduino oli verkossa, yhdistettiin se palvelimeen connect()-funktiolla, johon annettiin palvelimen ip-osoite sekä portti parametreina. Tämän jälkeen jäljelle jäi http-pyyntönä lähettää lämpötiladata palvelimelle.

Tällöin tieto oli kuitenkin vasta palvelimelle, joten palvelimelle luotiin lyhyt PHP-skripti, joka käsitteli Arduinon lähettämän http-pyynnön ja tallensi pyynnön mukana tulleen datan MySQL-tietokantaan.

4.5 Tulokset

Opinnäytetyön tuloksena syntyi ohjelma, jota on mahdollista käyttää laiterekisterinä yrityksessä, jolle se tuotettiin. Ohjelmaa on testidataa käyttämällä testattu, ja sen kaikki osa-alueet ovat toimineet halutulla tavalla. Syötettäessä dataa tiedonsyöttölomakkeeseen pysyy ”Save”-painike estettynä kunnes jokaisessa kentän kohdassa on näkyvää dataa. Kun lomake on täynnä pystyy kyseistä painiketta painamaan, ja tällöin data siirtyi tietokantaan kuten oli haluttu. Data siirtyi myös automaattisesti jäljitysketjuun sekä sai aikaleiman milloin syöttö tapahtui. Muokkaustoimintoa käyttäessä data haettiin tiedonsyöttölomakkeeseen tietokannasta, jossa virallinen tieto sijaitsi. Myös tällöin ”Save”-painike pysyi estettynä, jolloin jäljitysketjuun ei tule ylimääräisiä merkintöjä vahingossa. Painike aktivoituu vasta kun tietoja muutetaan, ja jos kentän tyhjentää se estyy uudelleen. Myös muokkaustoimintoa käyttäessä data siirtyi automaattisesti jäljitysketjuun ja sai aikaleiman. Samoin kävi kun kohteena ollut data poistettiin järjestelmästä, tällöin jäljitysketjuun tuli aikaleima kohtaan milloin tieto poistettiin. Saman numeron omaavia laitteita ei myöskään järjestelmään pystynyt syöttämään vaan tämä johti error-koodiin sekä selkokieliseen selitykseen. Tulostus toimii yhden laitteen tiedoilla hyvin, mutta jos koko rekisterin tiedot tulostettiin oli lopputulos hieman hankalahko lukea sillä sarakkeita on monta. Käyttäjien lisäys sovellukseen onnistui myös, sekä ”Admin”-määrittely käyttäjätunnuksille poisti tai lisäsi niille käyttöoikeuksia sovelluksen sisällä. Käyttäjätunnusten luonti ei tullut osaksi jäljitysketjua, sillä se ei ollut vaatimuksena, eikä käyttäjätunnuksia pysty sovelluksen

kautta muuttamaan vielä tällä hetkellä. ”NIU”-statuksen (Not In Use) määrittely laitteille sai ne näkymään kyseisellä listalla, sekä poisti ne käytössä olevien laitteiden listalta. Myös tästä jäi aikaleimallinen jälki jäljitysketjuun.

Arduino-pohjainen lämpömittari lähetti mittausdatansa tunnin välein tietokantaan, josta sen pystyi sovelluksella lukemaan tai tulostamaan. Testausta varten tämä aikaväli lyhennettiin muutamaan sekuntiin, jolloin nähtiin mittarien toimivan kun antureita lämmitettiin sormien välissä tai painettiin kylmää pulloa vasten. Mittareita ei kuitenkaan ole kalibroitu, joten niiden tarkkuudesta ei tämän opinnäytetyön puitteissa ollut tietoa, mutta näennäisesti lämpömittarin arvot olivat kuitenkin järkeviä.

5 LOPUKSI

Kehitystyön lopputuloksena syntyi toimiva sovellus tietokantapohjaiselle laiterekisterille, joka toimii ilman käyttöjärjestelmästä johtuvia rajoitteita. Lisäksi mikrokontrollerista rakennettu lämpömittari saatiin toimimaan halutulla tavalla. Ainakin jatkokehityksen kautta sovelluksen parantamiselle on luultavasti tarvetta kunhan se yrityksessä saadaan kaikkien käytettäväksi.

Sovellus tukee tällä hetkellä vain nykyistä implementaatiota laiterekisteriin vaadittavista tiedoista. Vaikka sovelluksen ohjelmakoodiin on helppo lisätä tai poistaa osia, jotka mahdollistavat laajemman tai suppeamman tietokannan käyttöä, ei sovellus taivu tämän muodostamiseen. Myös tietokannan taulukoiden muokkaaminen on vielä sovelluksen ulkopuolella ja ne on tehty suoraan SQL:lla tietokantaan. Nämä ainakin ovat mahdollisia jatkokehityksen kohteita tulevaisuudessa. Myös mahdollisuus lähettää sähköpostia käyttäjälle esimerkiksi kalibrointien umpeutuessa voisi osoittautua hyödylliseksi.

Mikrokontrollerin ominaisuuksia ei voi ajon aikana muuttaa lainkaan, vaan kaikki muutokset joudutaan tekemään itse ohjelmakoodiin. Lisäksi jos mittareita tai kontrollereita halutaan enemmän kuin nykyiset 2 mittaria ja 1 kontrolleri, joudutaan palvelimen PHP-komentosarjaa muokkaamaan. Myöskään mittareiden identifiointia ei voi muokata ilman laiterekisterisovelluksen muokkausta, joten ainakin näissä olisi varaa jatkokehitykselle mikrokontrollerin osalta.

LÄHTEET

Arduino 2017. Getting Started with Arduino and Genuino products. Viitattu 2.5.2017 https://www.arduino.cc/en/Guide/HomePage

U.S. Food & Drug Administration 2017. CFR - Code of Federal Regulations Title 21. Viitattu 2.5.2017 https://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfcfr/CFRSearch.cfm?CFRPart=11

Eclipse 2017. WindowBuilder – is a powerful and easy to use bi-directional Java GUI designer. Viitattu 2.5.2017 https://eclipse.org/windowbuilder/

MySQL 2017. Supported Platform: MySQL Database. Viitattu 2.5.2017 https://www.mysql.com/support/supportedplatforms/database.html

Oracle Docs 2017. Swing. Viitattu 2.5.2017 http://docs.oracle.com/javase/8/docs/technotes/guides/awt/

Oracle 2017. Update release notes. Viitattu 2.5.2017 http://www.oracle.com/technetwork/java/javase/6u12-137788.html

tutorialspoint 2017. JDBC Introduction. Viitattu 2.5.2017 https://www.tutorialspoint.com/jdbc/jdbc-introduction.htm

wikipedia 2017. Power over Ethernet. Viitattu 2.5.2017 https://en.wikipedia.org/wiki/Power_over_Ethernet