Lauri Hautala

Web-sovelluksen suunnittelu ja toteutus

Opinnäyteyö
Kevät 2018
SeAMK Tekniikka
Automaatiotekniikan tutkinto-ohjelma

SeAMK
SEINÄJOEN AMMATTIKORKEAKOULU
SEINÄJOKI UNIVERSITY OF APPLIED SCIENCES

SEINÄJOEN AMMATTIKORKEAKOULU

Opinnäytetyön tiivistelmä

Koulutusyksikkö: Tekniikan yksikkö

Tutkinto-ohjelma: Automaatiotekniikka

Suuntautumisvaihtoehto: Koneautomaatio

Tekijä: Lauri Hautala

Työn nimi: Web-sovelluksen suunnittelu ja toteutus

Ohjaaja: Petteri Mäkelä

Vuosi: 2018 Sivumäärä: 48 Liitteiden lukumäärä: 1

Opinnäytetyön aiheena oli web-sovelluksen suunnittelu ja toteutus. Työn toimeksiantaja oli Pme-Control Oy, yrityksellä oli tarve räätälöidylle tietojärjestelmälle, joka mahdollistaisi teknisten raporttien laatimisen ja hallinnoinnin ajasta ja paikasta riippumatta.

Työssä on kerrottu teoriaa mittauslaitteiden varmentamismenettelystä, joka antaa pohjan ohjelmiston vaatimuksille. Lisäksi on tehty ohjelmiston vaatimusmäärittely, jossa on mm. tutkittu miten EU:n uusi tietosuoja-asetus täytyy huomioida tietojärjestelmää suunniteltaessa. Työssä kerrotaan teoriaa tärkeimmistä käytetyistä tekniikoista: ASP.NET MVC, Azure cloud service ja Azure blob service. Lopuksi opinnäytetyössä tarkastellaan toteutettua ohjelmistoa ja sen ohjelmistoarkkitehtuuria.

Opinnäytetyön tuloksena saatiin toimiva web-sovellus. Sovellus vastaa pääosin sille asetettuja vaatimuksia, ja sovelluksen yritykselle tuomat hyödyt ovat selvästi nähtävissä.

Avainsanat: mittauslaite, varmennus, pilvipalvelu, ASP.NET, Azure, web-sovellus, Yleinen tietosuoja-asetus

SEINÄJOKI UNIVERSITY OF APPLIED SCIENCES

Thesis abstract

Faculty: School of Technology

Degree programme: Automation Engineering

Specialisation: Machine Automation

Author: Lauri Hautala

Title of thesis: Creating a web Application

Supervisor: Petteri Mäkelä

Year: 2018 Number of pages: 48 Number of appendices: 1

The purpose of this thesis was to design and develop a web application. The commissioner of this theses was Pme-Control Oy. The company needed a custom-made, mobile friendly application for managing technical reports in their business.

The thesis studied the verification methods intended for measuring instruments, which give the foundation for the application requirements. Also a software requirement specification of the application was made. In this section it was also studied how the new EU General Data Protection Regulation (GDPR) should be considered in the development of the application. The study included theory about the main software technologies used in the development, like ASP.NET MVC, Azure cloud service and Azure blob service. Finally the study concentrated on the developed application and the architecture of the application.

As a result of the thesis a web-application was developed. The application meets the mandatory requirements and the benefits for the company are obvious.

Keywords: Measuring instrument, legal metrology, verification, cloud service, ASP.NET, Azure, web application, General Data Protection Regulation

SISÄLTÖ

Kuva- ja taulukkoluettelo

Käytetyt termit ja lyhenteet

Ajax	(Asynchronous JavaScript And XML) on tekniikka, jota käytetään vuorovaikutteisten websovellusten luomiseen.
C#	Microsoft-yhtiön .NET-konseptia varten kehittämä ohjelmointikieli.
Html	Hyper Text Markup Language. Internetsivujen luomisessa käytettävä koodikieli.
JavaScript	Pääasiassa web-ympäristössä käytettävä dynaaminen komentosarjakieli.
Mittanormaali	Referenssi, johon vertaamalla voidaan tarkastaa mittalaitteita.
UML	UML-mallinnus (Unified Modeling Language) on graafinen mallinnuskieli.
Varmennus	Mittauslaitteelle tehtävä lakisääteinen tarkastus.
Varmennusaskelarvo(*e*)	Massaa kuvaava arvo, jota käytetään mittauslaitteiden luokituksissa ja varmennuksissa.
Versionhallinta	Järjestelmä tallentaa muutoksia tiedostoon tai joukkoon tiedostoja ja mahdollistaa tiettyihin versioihin paluun myöhemmin.

1 JOHDANTO

1.1 Työn tausta

Opinnäytetyö on tehty Pme-Conrol Oy:lle. Yrityksen toimiala on mittauslaitteiden tarkastaminen, tarkastustoiminta tapahtuu pääasiassa asiakkaiden kohteissa ympäri Suomea. Yritys tarvitsi mittaus- ja asiakastietojen hallintaan nykyaikaisen tietojärjestelmän. Työssä on suunniteltu ja toteutettu web-sovellus toimeksiantajan käyttöön.

1.2 Työn tavoite

Tavoitteena on luoda tietojärjestelmä toimeksiantajan käyttöön, millä pystytään korvaamaan kaikki nykyisin käytössä olevat mittaustietojen käsittelyyn käytettävät järjestelmät. Järjestelmän tulee helpottaa tarkastajien kentällä tekemää työtä sekä tarjota yrityksen johdolle tarvittavat tiedot raportointiin ja laskutukseen. Tavoitteena on myös luoda järjestelmä, joka ohjaa käyttäjää ja ennaltaehkäisee käyttäjää tekemästä virheitä, ja näin parantaa asiakastyytyväisyyttä sekä vähentää tietojen korjausten tarvetta. Järjestelmän tulee myös täyttää viranomaisten vaatimukset tiedon suojauksen, varmuuskopioinnin ja jäljitettävyyden osalta.

Työ on rajattu siten, että se käsittää ainoastaan mittaustietojen tallentamiseen ja raportointiin tarvittavat toiminnat sekä niihin välittömästi liittyvät rekisterit, kuten asiakasrekisterin ja mittanormaalirekisterin.

1.3 Työn rakenne

Ohjelmiston suunnittelun ja toteutuksen lisäksi opinnäyteyössä kerrotaan teoriaa ohjelmiston toteutukseen käytetyistä tekniikoista ja palveluista sekä ohjelmiston suunniteluun vaikuttaneista tekijöistä. Teoriaosuus koostuu kahdesta eri osiosta. Luvussa 2 kerrotaan yleistä asiaa mittauslaitteiden varmentamisesta ja hieman tar-

kemmin ei-automaattisten vaakojen varmennusmenettelystä. Luvussa 3 on ohjelmiston vaatimusmäärittely. Luvussa kerrotaan ohjelmiston tärkeimmistä toiminnoista ja vaatimuksista, sekä joistain yleisimmistä käyttötapauksista. Luvussa tutkitaan myös Euroopan unionin uuden tietosuoja-asetuksen tietojärjestelmälle asettamia vaatimuksia. Luvussa 4 käydään läpi käytettyjä palveluita ja tekniikoita, ja miten niitä on hyödynnetty sovelluksen toteuttamisessa.

Luvussa 5 kerrotaan ohjelmiston toteutuksesta. Luvussa käsitellään esimerkkien ja kuvioiden avulla ohjelmiston arkkitehtuuria ja lopuksi kerrotaan sovelluksen tärkeimmistä näkymistä. Viimeisessä luvussa pohditaan, miten projekti onnistui ja mitä opittiin.

1.4 Yritysesittely

Pme-Control Oy on erikoistunut mittauslaitteiden varmennus- ja tarkastustoimintaan, yritys on toiminut alalla vuodesta 1991. Pme-Control Oy on vuonna 2007 akreditoitu puolueeton ja riippumaton A-tyypin tarkastuslaitos. Yrityksen päätoimipaikka on Jurvassa ja toimialueena on koko Suomi, työntekijöitä on tällä hetkellä 5. (Hiipakka 2018.)

Toiminnan suurimpana osa-alueena on kaupankäynnissä käytettävien mittauslaitteiden varmentaminen. Varmennettavia laitteita ovat mm. polttonestemittarit kuten varastojen suurtehomittarit, jakeluautojen mittarit, sekä huoltamo- ja muut vähittäiskauppamittarit, sekä vaakojen varmennukset päivittäistavarakaupassa ja teollisuudessa. (Hiipakka 2018.)

2 MITTAUSLAITTEIDEN VARMENTAMINEN

Suomessa kaikkien mittauslaitteiden joita käytetään taloudellisen edun määrittämiseen kaupankäynnissä tai viranomaisten päätöksenteossa, on oltava mittauslaitelain vaatimusten mukaisia. Tällaisia kaikkien tuntemia laitteita ovat muun muassa hedelmävaaka ruokakaupassa tai polttonestemittari huoltoasemalla. Yleensä laitteet täytyy myös varmentaa säännöllisesti käytön aikana. Aiemmin varmennus tunnettiin termillä vakaus. Mittauslaitteita Suomessa saavat varmentaa ainoastaan Tukesin hyväksymät tarkastuslaitokset. Varmentamismenettelyllä pyritään varmistamaan mittauslaitteiden luotettava toiminta. (Tukes 16.1.2017 a.)

Mittauslaitteille tehdään usein ensivarmennus ennen käyttöönottoa sekä mahdollinen määräaikaisvarmennus tietyin väliajoin. Määräaikaisvarmennuksen taajuus vaihtelee eri laitetyypeittäin, kaikki mittauslaitteet eivät vaadi käytön aikaista varmentamista lainkaan. Taulukossa 1 on tietoa erilaisten mittauslaitteiden varmentamismenettelyistä. Käytön aikaisessa varmentamisessa tarkastetaan, että mittauslaite toimii luotettavasti, siihen ei ole tehty asiattomia muutoksia ja että asianmukaiset merkinnät ovat paikoillaan. Käytön aikainen varmennus tulee suorittaa myös aina, jos on syytä epäillä laitteen luotettavuutta tai sen merkinnät ovat vahingoittuneet. (Tukes 16.1.2017 b.)

Varmennuksessa hylätty mittauslaite voidaan huoltaa ja tämän jälkeen suorittaa uusintavarmennus. Jos mittauslaite ei täytä vaatimuksia se voidaan asettaa tilapäiseen käyttökieltoon. (Avi 16.1.2017.)

Suomessa mittauslaitteiden lainsäädäntö perustuu suurimmalta osin kansainvälisiin suosituksiin. Suosituksia laatii alan kansainvälinen kattojärjestö Organisation internationale de métrologie légale (OIML). (Rauhalaakso 2018.)

Taulukko 1. Erilaisten mittauslaitteiden varmentamismenettelyt ennen käyttöönottoa ja sen jälkeen

(Tukes 16.1.2017 b.)

Mittauslaitteiden vaatimukset	**Ennen käyttöönottoa**			**Käyttöönoton jälkeen**	
	Tyyppitarkastus ja varmentaminen (kansallinen tai EY)		MIDin tai NAWID:n mukaan		
	Tyyppitarkastus	Ensivarmennus	Vaatimustenmukaisuusvakuutus	Varmennus käytön aikana	Varmennusväli (vuotta)
Tavallinen kaupan vaaka (NAWI)	-	-	x	x	3
Automaattinen vaaka	x	x	x	x	2/3[2]
Punnus (>50 mg) Polttonestemittari		x	–[1]	x	3
Voiteluöljymittari			x	x	3
Säiliöautomittari (polttoneste)			x	x	2
Maitoauton mittari			x	x	3
Kuljetussäiliö	x	x	–[1]	-	
Pituusmitta			x	-	
Pituusmittari			x	x	3
Mitta-astia (≤5 l)			x	-	-
Torikaupan kapat ja lieriömitat		x		-	-
Pullonsuumittari	-	x	–[1]	x	3
Olut/siiderimittari	x	x	–[1]	x	3
Lämpöenergiamittari			x	tulossa	
Vesimittari			x	tulossa	
Sähköenergiamittari			x	tulossa	
Kaasumittari			x	tulossa	

[1] Nämä mittauslaitteet eivät ole mukana Euroopan unionin mittauslaitedirektiivissä MID:issä, joten niitä koskevat edelleen kansalliset tyyppihyväksyntä- ja varmennusvaatimukset.

[2] Pakkausten sisällön määrän tarkistamiseen käytettävät linjavaa'at varmennetaan 2 vuoden välein, muut automaattiset vaa'at 3 vuoden välein.

2.1 Varmennus ja kalibrointi

Termit kalibrointi ja varmennus mielletään usein samaksi asiaksi. Näin ei kuitenkaan ole. Kalibrointi on mittauslaitteen näyttämän vertaamista tunnettuun arvoon, se ei ota kantaa mittauslaitteen laatuun, joten mikä tahansa mittauslaite voidaan kalibroida. Kalibroinnille annetaan mittausepävarmuus, kun taas varmentamisessa annetaan varmennusvirheraja ja käyttövirheraja. Varmennusvirherajasta käytetään myös nimitystä suurin sallittu virhe (ssv). Lisäksi varmennetun laitteen erottelukyvyn tulee olla sellainen, ettei käyttäjälle tule väärää kuvaa mittauslaitteen tarkkuudesta. Esimerkiksi ei-automaattisilla vaaoilla tarkkuus on tyypillisesti 1–3 askelarvoa. Lisäksi luotettava mittaustulos on saatava aikaan yhdellä mittauksella. (Pusa, Riski & Ojanen-Saloranta 2017, 10, 13.)

Näillä säännöillä pyritään varmistamaan se, että mittaustekniikkaa tuntematonkin henkilö saa varmennetulla mittauslaitteella luotettavan tuloksen. Kalibrointi vastaavasti on aina palvelua ammattilaiselta ammattilaiselle. (Pusa, Riski & Ojanen-Saloranta 2017, 10, 13.)

2.2 Ei-automaattisten vaakojen varmentaminen

Vaa'at jaetaan kahteen ryhmään käyttötavan mukaan: automaattiset ja ei-automaattiset. Vaaka, joka vaatii käyttäjä toimenpiteitä mittaustuloksen todentamiseksi, määritellään ei-automaattiseksi vaa'aksi (OIML R76-1, 2006, 5).

Tässä luvussa kerrotaan esimerkin vuoksi hieman tarkemmin ei-automaattisten vaakojen varmentamisesta. Ei-automaattiset vaa'at ovat yksi yleisimmistä Pme-Control Oy:n varmentamista laitetyypeistä (Rauhalaakso 2018). Seuraavassa kerrotaan ei-automaattisten vaakojen suorituskyvyn mittaamisesta, tarkkuusluokista ja sallituista virheistä. Mainittujen testien lisäksi varmentamismenettelyyn kuuluu myös

muita tarkastettavia asioita ja testejä, joihin ei tässä syvennytä. Kuvassa 1 suoritetaan vaa'an varmennusta testipunnusten avulla.

Kuva 1. Vaa'an varmennus (Pme-Control [Viitattu 26.4.2018].)

2.2.1 Tarkkuusluokat ja virherajat

Ei-automaattiset vaa'at jaetaan neljään tarkkuusluokkaan. Taulukossa 2 on kerrottu kunkin tarkkuusluokan varmennusaskelarvo *e*, varmennusaskelten määrä sekä minimikapasiteetti.

Taulukko 2. Ei-automaattisten vaakojen tarkkuusluokat
(OIML R 76-1, 2006, 27.)

Tarkkuusluokka	Varmennusaskelarvo	Varmennusaskelten määrä		Minimi kapasiteetti
		minimi	maksimi	
I	**0,001 g ≤ *e****	**50 000****	**-**	**100 e**
II	**0,001 g ≤ e ≤ 0,05 g** **0,1 g ≤ *e***	**100** **5000**	**100 000** **100 000**	**20 e** **50 e**
III	**0,l g ≤ e ≤ 2 g** **5 g ≤ *e***	**100** **500**	**10 000** **10 000**	**20 e** **20 e**
IIII	**5 g ≤ *e***	**100**	**1000**	**10 e**

* Vaa'an, jonka *e* < 1mg varmentaminen ei normaaliolosuhteissa ole mahdollista testipainojen epävarmuudesta johtuen.

** Vaa'an jonka tarkkuusluokka on I ja *d* < 0,1, varmennusaskelten määrä voi olla pienempi kuin 50 000. (OIML R 76-1, 2006, 27.)

Taulukossa 3 on kerrottu suurimmat sallitut virheet suhteessa varmennusaskelarvoon *e* ja massaan *m*.

Taulukko 3. Suurimmat sallitut virheet
(OIML R 76-1, 2006, 30.)

Suurin sallittu virhe	Kuorma			
	I	II	III	III
± 0.5 *e* ± 1.0 *e* ± 1.5 *e*	0 ≤ *m* ≤ 50 000 50 000 < *m* ≤ 200 000 200 000 < *m*	0 ≤ m ≤ 5 000 5 000 < m ≤ 20 000 20 000 < m ≤ 100 000	0 ≤ m ≤ 500 500 < m ≤ 2 000 2 000 < m ≤ 10 000	0 ≤ m ≤ 50 50 < m ≤ 200 200 < m ≤ 1 000

2.2.2 Punnitustesti

Punnitustestissä vaaka kuormataan nollasta ylöspäin vaa'an maksimikuormaan saakka ja vastaavasti kuormaa poistamalla maksimista nollaan. Testissä käytetään vähintään viittä erisuuruista kuormaa. Testipainot valitaan siten, että ne sisältävät maksimi- ja minimikuormattavuuden, sekä ne kohdat, joissa suurin sallittu virhe vaihtuu, tai mahdollisimman lähellä näitä kohtia olevat pisteet. Minimiä ei kuitenkaan tarvitse mitata, jos se on pienempi kuin 100 mg. Testi suoritetaan siten, että lastatessa tai kuormatessa kuorma nousee tai laskee jatkuvasti. (OIML R 76-1, 2006, 88.)

2.2.3 Pyöristysvirheettömän virheen määrittäminen

Tutkittavan vaa'an näyttämää on pysyttävä todentamaan vähintään 1/5 e:n tarkkuudella. Jos vaa'assa on digitaalinen näyttölaite ja sen näyttämä ei ole luettavissa 1/5 e:n tarkkuudella, on vaa'an tarkka näyttämä selvitettävä näyttölaitteen näyttämän vaihtokohtia tarkastelemalla. (OIML R 76-1, 2006, 88.)

Vaihtokohtien tarkasteleminen tehdään lisäpainoja käyttämällä, näin saadaan selville virhe ennen näyttölaitteen tekemää pyöristystä. Lisäpainot ovat massaltaan 1/10 *e*. Tietyllä kuormalla *L*, saadaan näyttölaitteen näyttämä *I*. Lisäpainoja lastataan vaakalevylle kunnes näyttölaitteen näyttämä nousee seuraavaan suurempaan askelarvoon. Lisäpainojen *ΔL* ollessa vaakalevyllä saadaan todellinen näyttämä *P* käyttämällä seuraavaa kaavaa 1.

$$P = I + \frac{1}{2}\mathrm{e} - \Delta\mathrm{L} \qquad (1)$$

missä:

P = todellinen näyttämä

I = näyttölaitteen näyttämä

e = varmennusaskelarvo

ΔL = lisäpaino

Näin todellinen virhe saadaan käyttämällä kaavaa 2:

$$E = P - L \quad (2)$$

missä:

E = virhe ennen pyöristystä

P = todellinen näyttämä

L = kuorma

(OIML R 76-1, 2006, 88–89.)

.

3 OHJELMISTON VAATIMUSMÄÄRITTELY

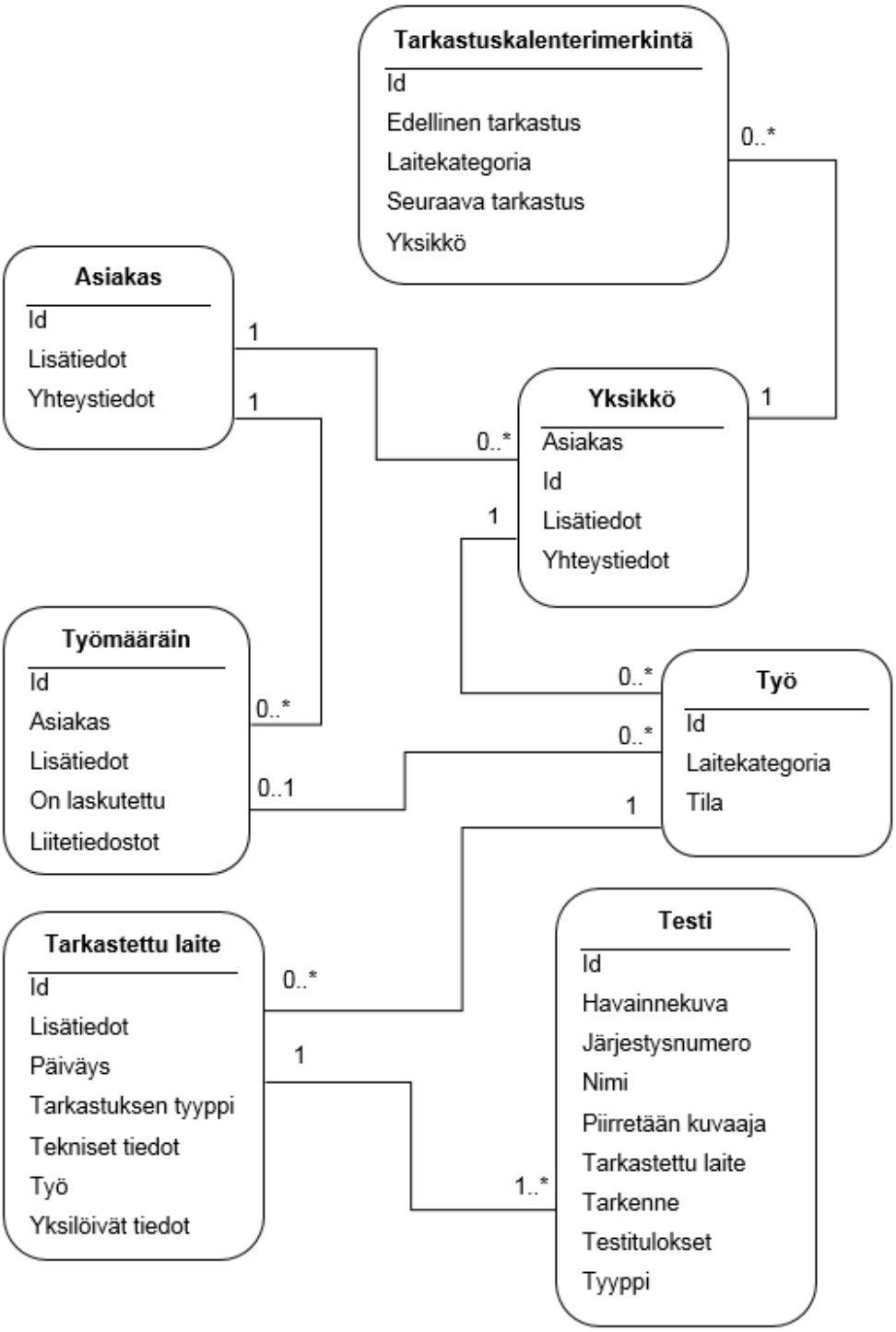
Ohjelmiston tärkein vaatimus on mittauslaitteiden tarkastustulosten ja muiden tietojen tallennus ja raporttien luominen. Toinen olennainen vaatimus on ylläpitää rekistereitä mittauslaitemalleista, asiakkaista, asiakkaan kohteista ja tarkastetuista mittauslaitteista sekä tarkastuksiin käytetyistä mittanormaaleista.

3.1 Toiminnalliset vaatimukset

Tässä luvussa kerrotaan ohjelmiston toiminnallisista vaatimuksista ja kuvataan yksinkertaistetusti tärkeimpien rekistereiden tietorakenteet.

3.1.1 Asiakasrekisteri

Asiakasrekisteri koostuu asiakkaista ja asiakkaan yksiköistä. Asiakkaalla voi olla useampi yksikkö, ja yksiköllä voi olla useampi työ. Sovelluksessa on taulukkomuotoinen näkymä, jossa näkyvät kaikki menneet ja tulevat tarkastukset. Kalenterissa näkyy milloin yksikössä on seuraava tarkastus, samalla yksiköllä voi olla useita laitekategorioita, joilla voi olla kaikilla eri tarkastusajankohdat, kuitenkin niin että yhdellä laitekategorialla on vain yksi ajankohta. Kuvassa 2 on kuvattu asiakasrekisterin tietorakenne.



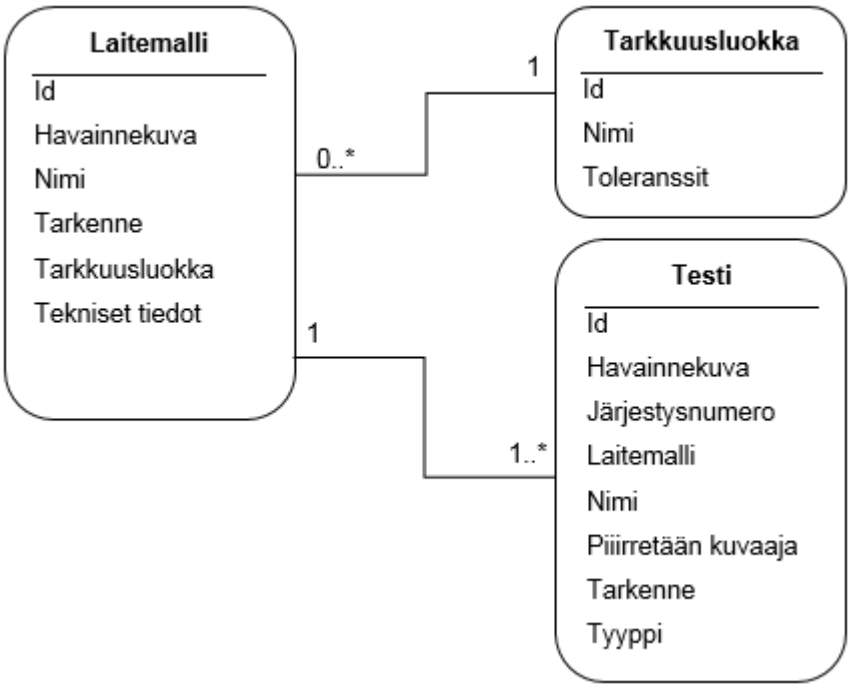
Kuva 2. Asiakasrekisterin tietorakenteen UML-malli

3.1.2 Mittauslaitemallirekisteri

Tarkastettaville laitteille tehtävät mittaukset ja mittausvirherajat ovat aina riippuvaisia tarkastettavasta laitteesta, tähän vaikuttavat mm. laitteen mittausalue, tarkkuusluokka ja tyyppihyväksyntä. Uutta laitemallia luotaessa järjestelmään syötetään

kaikki tarvittavat tekniset tiedot ja näiden tietojen perusteella määritetään laitteelle asetusten mukaiset testit. Sovellus laskee mittausvirherajat annetun tarkkuusluokan mukaan. Kun laitemalli on kerran tallennettu, se on jatkossa valittavissa listalta tarkastustoimenpiteen yhteydessä.

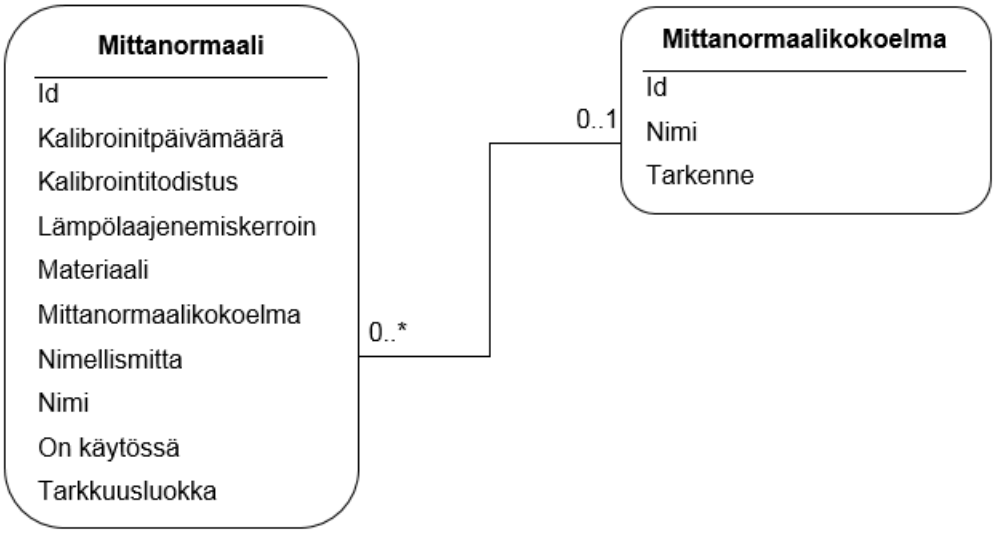
Laitemallin tietoihin voidaan lisätä myös muita tietoja ja ohjeita, jotka helpottavat käyttäjän työtä jatkossa. Sovelluksessa on ominaisuus, jonka avulla voidaan tarvittaessa piirtää havainnekuva vaa'an kuormauksesta. Kuvasta voidaan nähdä miten vaaka tulee kuormata testaustilanteessa. Havainnekuvia on mahdollista tehdä yksi jokaista testiä kohden. Lisäksi on mahdollista tehdä yksi ylemmän tason havainnekuva laitemalla kohden, joka kuvaa kaikkia laitemallin testejä. Kuvassa 3 on kuvattu laitemallirekisterin tietorakenne pääpiirteissään.



Kuva 3. Mittauslaitemallirekisterin tietorakenteen UML-malli

3.1.3 Mittanormaalirekisteri

Tarkastuksissa käytetyt mittanormaalit tallennetaan mittanormaalirekisteriin. Jokaiselle laitekategorialle on oma rekisterinsä. Kuvassa 4 on kuvattu rekisterin tietorakenne pääpiirteissään. Tietorakenne vaihtelee hieman eri laitekategorioiden välillä. Mittanormaaleista voidaan luoda myös kokoelmia.

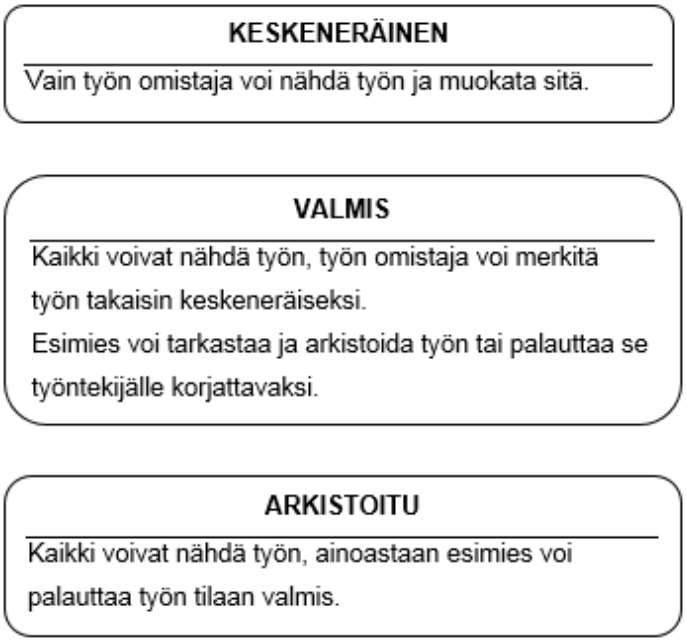


Kuva 4. Mittanormaalirekisterin tietorakenteen UML-malli

3.1.4 Työn hallintamekanismi

Järjestelmään luodaan kolmivaiheinen työn hallintamekanismi. Kuvassa 5 on esitetty työn hallinnan vaiheet. Eri vaiheiden avulla pystytään valvomaan työn kulkua siten, että esimies tietää, milloin työ on valmis, lisäksi tällä varmistetaan, että työntekijä ei muokkaa työtä, jos se on jo arkistoitu esimiehen toimesta. Vain yksi käyttäjä voi kerrallaan muokata työtä, tätä käyttäjää kutsutaan työn omistajaksi.

Kun uusi työ luodaan, se on tilassa keskeneräinen. Kun työntekijä on saanut työn valmiiksi, se merkitään tilaan valmis, tämän jälkeen esimiehellä on mahdollisuus merkitä työ tilaan arkistoitu. Työn siirtyessä tilaan arkistoitu se siirtyy automaattisesti työmääräimelle, yhdellä työmääräimellä voi olla useita saman asiakkaan töitä. Työmääräimellä on merkintä siitä, onko se laskutettu vai ei. Työmääräin menee aina tilaan laskuttamatta, kun sille lisätään uusi työ. Jos työ merkitään takaisin tilaan valmis, se poistuu automaattisesti työmääräimeltä.



Kuva 5. Työn tilat

3.1.5 Uuden työn luominen

Kun käyttäjä aloittaa uuden tarkastustoimenpiteen, hän luo järjestelmään uuden työn. Työ on aina laitekategoriakohtainen, esim. jos samassa kohteessa on kahden laitekategorian laitteita, luodaan kaksi eri työtä. Jos asiakkaan tai tarkastettavan laitemallin tietoja ei ole järjestelmässä entuudestaan, lisätään nämä ennen uuden laitteen lisäämistä.

Kun uusi työ on luotu, lisätään työlle uusi tarkastettava laite. Laitteen lisäämisen yhteydessä käyttäjä valitsee laitteen mallin, ja sovellus hakee mittauslaitemallirekisteristä kyseisen laitemallin tiedot. Tämän jälkeen käyttäjä täyttää lomakkeelle tarkastettavan laitteen yksilöivät tiedot, mittaustulokset ja mahdolliset muut tiedot kuten ympäristöolosuhteet, viritettiinkö laitetta tarkastuksen yhteydessä tai huomattiinko jotain poikkeavaa. Lisäksi käyttäjä valitsee tarkastuksen tyypin: varmennus, kalibrointi tai tarkastus. Sovellus vaihtaa lomakkeen valitun tyypin mukaiseksi. Yhdelle työlle voidaan lisätä tarvittava määrä laitteita.

Käyttäjä voi tarkastella työn yhteenvetonäkymää, jossa näkyy kaikki työlle kuuluvat tarkastetut laitteet. Yhteenvedossa näkyy vähintäänkin seuraavassa luettelossa olevat tiedot. Tiedot vaihtelevat hieman laitekategoriasta ja tarkastuksen tyypistä riippuen.

Yhteenvedossa vähintään näkyvät tiedot:

- Laitemalli
- Onko kaikki pakolliset tiedot täytetty
- Laitteen sijainti, esimerkiksi osasto tai mittarin kenttänumero
- Maksimikapasiteetti
- Mahdollinen tarkkuusluokka
- Läpäisikö laite tarkastuksen hyväksytysti.

Kun käyttäjä on saanut työn valmiiksi, hän merkitsee työn tilaan valmis. Tässä vaiheessa sovellus luo tiedoista pdf-muotoisen raportin, joka tallentuu tietokantaan.

3.1.6 Työn arkistoiminen

Kun työntekijä on merkinnyt työn valmiiksi, tulee tästä ilmoitus esimiehelle. Esimies tarkastaa, että raportit on asianmukaisesi tehty ja tarvittaessa palauttaa työn työntekijälle korjattavaksi. Kun työ on valmis arkistoitavaksi, esimies merkitsee sen tilaan valmis. Samalla työ näkyy esimiehen valitsemassa työmääräimessä. Lopuksi hän laskuttaa työn ja merkitsee työmääräimen laskutetuksi.

3.1.7 Tarkastuspöytäkirjat

Sovellus muodostaa työstä aina pdf-muotoisen tarkastuspöytäkirjan. Yhdestä työstä luodaan yksi pdf-tiedosto. Pdf-tiedosto pyritään tekemään siten, että tulosteella on yksi tarkastettu laite per sivu. Pöytäkirjat tallennetaan sovelluksen tietokantaan myöhempää käyttöä varten, tiedostojen tulee olla helposti löydettävissä ja ladattavissa. Pöytäkirjoja luovutetaan tarvittaessa asiakkaille ja viranomaisille, niiden ulkoasun tulee olla siisti ja yhdenmukainen. Mallina käytetään yrityksen olemassa olevia pöytäkirjamalleja.

3.2 Käyttäjät ja käyttäjäryhmät

Ohjelmistossa on kolme eri käyttäjäryhmää: ylläpitäjä, esimies ja työntekijä. Eri käyttäjäryhmillä on pääasiassa yhtenevät käyttöoikeudet. Taulukossa 4 on esitetty ne toiminnot, joiden käyttöoikeus poikkeaa eri käyttäjäryhmien välillä.

Taulukko 4. Käyttäjäryhmien poikkeavat käyttöoikeudet

Toiminto	Ylläpitäjä	Esimies	Työntekijä
Työn tilamuutos valmis ➔ arkistoitu	**x**	**x**	
Työn tilamuutos arkistoitu ➔ valmis	**x**	**x**	
Ilmoitus työn valmistumisesta		**x**	

3.3 Ei-toiminnalliset vaatimukset

Ohjelmiston tulee olla suomenkielinen ja helppokäyttöinen. Ohjelmistoa tulee voida käyttää työpöytä- ja mobiililaitteilla. Järjestelmällä on alkuvaiheessa alle 10 käyttäjää, määrän on arvioitu pysyvän melko pienenä myös lähitulevaisuudessa. Tällä hetkellä ei ole nähtävissä tarvetta järjestelmän integroimiseksi muihin järjestelmiin.

Vain tunnistautuneet käyttäjät voivat käyttää ohjelmistoa. Ohjelmiston tulee täyttää viranomaisten asettamat vaatimukset tiedon suojauksen ja jäljitettävyyden osalta. Ohjelmisto tulee validoida ennen käyttöönottoa sekä aina jos ohjelmistoon tehdään muutoksia. Validointi koskee erityisesti niitä ohjelmiston osia, jotka vaikuttavat mittaustulosten oikeellisuuteen. Validoinnista laaditaan validointisuunnitelma ja tulokset dokumentoidaan asianmukaisesti.

3.4 Euroopan unionin tietosuoja-asetus ja sen asettamat vaatimukset tietojärjestelmälle

Suomen ja Euroopan unionin tietosuojalainsäädäntö on uudistumassa. Tietosuoja-asetusta (General Data Protection Regulation, GDPR) sovelletaan 25.5.2018 alkaen. (Tietosuojavaltuutetun toimisto, 23.3.2018.)

Tietosuoja-asetuksen tarkoituksena on vastata uusin haasteisiin, jotka digitalisaatio ja globalisaatio asettavat henkilötietojen tietosuojalle. Vahvistamalla jäsenmaiden välistä luottamusta asetus pyrkii poistamaan digitaalitalouden kehityksen esteitä Euroopan Unionin sisämarkkinoilla. Tietosuoja-asetuksen tarkoituksena on vahvistaa rekisteriin kerättyjen henkilöiden oikeutta vaikuttaa siihen, millaista tietoa heistä tallennetaan ja miten tietoja käytetään, asetus myös säätää rekisteröidylle oikeuden tulla unohdetuksi. (Talus, Autio, Hänninen, Pihamaa & Kantonen, 2017, 23, 25.)

Tietosuoja-asetuksen noudattamista pyritään tukemaan eri toimenpitein: Rekisterinpitäjä voidaan määrätä asetusten vastaisesta tietojen käsittelystä sakkoihin, tai se voidaan määrätä tekemään tietojen käsittelyyn liittyviä korjaavia toimenpiteitä. (Talus ym. 2017, 23.)

Organisaatioiden on varmistettava, että heidän käyttämät tietojärjestelmät ja prosessit vastaavat asetuksen vaatimuksia rekisteröityjen henkilötietojen osalta. Nykyinen henkilötietolaki on jo monilta osin yhtenevä uuden tietosuoja-asetuksen kanssa, mutta asetus on yksityiskohtaisempi. (Talus ym. 2017, 23.)

Opinnäyteyössä toteutettavaan järjestelmään tallennetaan ainoastaan järjestelmän käyttäjien eli toimeksiantajayrityksen työntekijöiden henkilötietoja. Järjestelmään tallennettavat asiakkaat ovat poikkeuksetta yrityksiä, eivät luonnollisia henkilöitä. Tulevaisuudessa on mahdollista, että järjestelmään kerätään myös muita henkilötietoja, kuten asiakkaiden tai heidän edustajien henkilötietoja, joten asetuksen vaatimukset tulee huomioida järjestelmän toteutuksessa.

4 KÄYTETYT TEKNIIKAT JA PALVELUT

Tässä luvussa kerrotaan lyhyesti ohjelmiston toteutuksessa käytetyistä tekniikoista ja palveluista.

Työn alkuvaiheessa päätettiin käytettävät tekniikat. Tekniikaksi valikoitui Microsoft ASP.NET MVC ja Microsoft SQL. Tekniikkoihin päädyttiin, koska työn tekijällä sekä yhteistyökumppaneilla oli kokemusta näistä tekniikoista. Valittujen tekniikoiden myötä kehitysympäristöksi tuli Microsoft Visual Studio ja Microsoft SQL Management Studio.

Lisäksi tietokantakyselyissä päädyttiin käyttämään Entity Framework Database First -sovelluskehystä. Tekniikka valittiin, koska se vastasi parhaiten vaatimuksia. Käyttöliittymän toteuttamisessa on käytetty apuna Bootstrap-sovelluskehystä. Bootstarp valittiin koska se on yksi maailman käytetyimmistä HTML-, CSS- ja JavaScript -sovelluskehyksistä ja siksi sen käyttöön löytyy paljon tukea ja ohjeita.

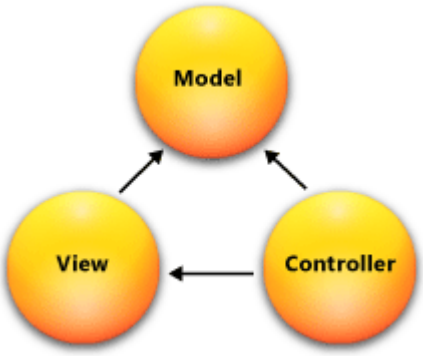
Sovelluksen julkaisualustaksi valittiin Microsoft Azure ja tiedostojen tallentamiseen käytettiin Azure Blob Storage -palvelua. Julkaisualusta valittiin, koska siitä oli aiempaa kokemusta. Blob Storage -palvelun käyttöön päädyttiin, koska se vastasi parhaiten vaatimuksia ja oli parhaiten yhteensopiva muiden valittujen tekniikoiden kanssa. Ohjelmiston versionhallinta toteutettiin Git-versionhallintaa käyttäen. Git-versionhallinan valintaan vaikutti tekijän kokemus järjestelmän käytöstä opiskelujen aikana.

Sovelluksen erilaiset lomakkeet toteutettiin pääosin käyttämällä perinteisiä web-lomakkeita, mutta myös asynkronista AJAX-tekniikkaa on käytetty tietojen tallentamisessa ja näkymien päivittämisessä.

4.1 ASP.NET MVC

ASP.NET MVC on Microsoftin sovelluskehys web-sovelluksen käyttöliittymän toteuttamiseen. Model-View-Controller-arkkitehtuuri jakaa sovelluksen kolmeen pääkomponenttiin: model, view ja controller. (Microsoft [Viitattu 1.4.2018].) Tässä työssä on käytetty ASP.NET MVC 5 -sovelluskehystä.

Model on luokka, joka sisältää käyttöliittymän näkymän tietomallin. Usein model-olio vastaanottaa tiedot tietokannasta, välittää ne näkymälle ja vastaavasti välittää näkymältä tulevat tiedot tietokantaan. View on komponentti, joka näyttä tiedot käyttäjälle HTML-sivun muodossa. View rakentaa html-sivun tyypillisesti model-komponentin tiedoista. Controller on luokka, joka käsittelee käyttäjän pyynnöt, valitsee näytettävän näkymän ja välittää tiedot model-olion muodossa näkymälle. Kuvassa on 6 on havainnollistettu MVC-arkkitehtuuri. (Microsoft [Viitattu 1.4.2018].)



Kuva 6. MVC arkkitehtuuri (Microsoft [Viitattu 1.4.2018].)

4.2 Azure

Azure on Microsoftin tarjoama pilvipalvelu, joka tarjoaa IT-ammattilaisille palvelua erilaisten sovellusten kehittämiseen, julkaisuun ja hallintaan (Microsoft [Viitattu 2.4.2018]).

4.3 Azure Cloud Service

Azure tarjoaa useita erilaisia vaihtoehtoja web-sovelluksen julkaisemiseen kuten:

- Azure App Service
- Virtual Machines
- Service Fabric
- Cloud Services (Microsoft 7.7.2016).

Tämän työn julkaisualustaa valittaessa mietittiin lähinnä Azure App Service -palvelun ja Azure Cloud Service -palvelun välillä. Tässä luvussa kerrotaan Azure Cloud Service -palvelusta ja verrataan sitä Azure App Service -palveluun.

Azure Cloud Service on Platform-as-a-Service (PaaS) -tyyppinen palvelu. Aivan kuten Azure App Service, myös tämä palvelu on suunniteltu helposti skaalautuvien sovellusten toteuttamiseen. Molemmat palvelut toimivat Azure-virtuaalikoneilla. (Microsoft 19.4.2017.)

Azure Cloud Service tarjoaa vapaamman pääsyn virtuaalikoneelle verrattuna App Service -palveluun. Virtuaalikoneelle voidaan asentaa ohjelmia ja sinne voidaan ottaa etäyhteys, App Service -palvelussa tämä ei ole mahdollista. Toisaalta vapauden myötä palvelun helppokäyttöisyys kärsii. Tyypillisesti web-sovellus on nopeampi ja helpompi julkaista App Service -palvelun avulla. Voidaankin todeta, että palveluksi kannatta valita App Service, jos ei todella tarvitse Cloud Servicen tarjoamia vapauksia. (Microsoft 19.4.2017.)

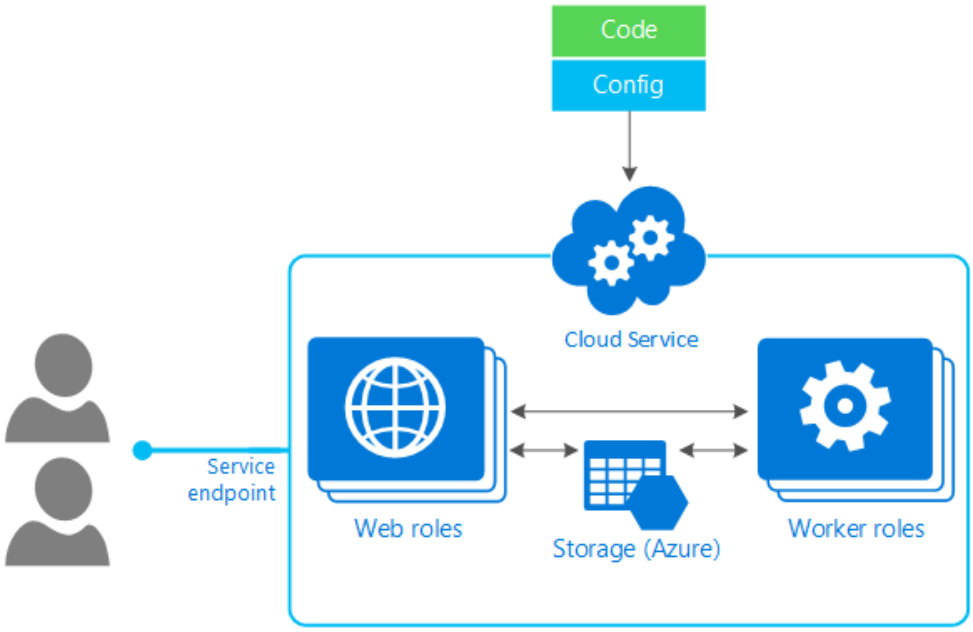
Azure Cloud Service -palvelussa on kaksi eri roolia. Roolien ero on siinä, miten ne toimivat virtuaalikoneella.

- Web: Julkaisee sovelluksen automaattisesti IIS-ohjelmiston avulla.
- Worker: Toimii itsenäisesti eikä käytä IIS-palvelua. (Microsoft 19.4.2017.)

Esimerkiksi yksinkertainen web-sovellus voi käyttää vain yhtä web-roolia, jos useammalle roolille ei ole tarvetta. Monimutkaisempi sovellus taas voi käyttää kahta

roolia esimerkiksi siten, että web-roolia käytetään käyttäjiltä tulevien pyyntöjen käsittelemiseen ja pyynnöt välitetään edelleen worker-roolille jatkokäsiteltäviksi. (Microsoft 19.4.2017.)

Cloud service -sovelluksessa kaikkien virtuaalikoneiden tulee toimia samassa Cloud Service -palvelussa. Arkkitehtuuri on havainnollistettu kuvassa 7. Käyttäjät voivat ottaa yhteyden sovellukseen yhden julkisen IP-osoitteen avulla, ja alusta jakaa pyynnöt automaattisesti eri virtuaalikoneille. Alusta pyrkii skaalamaan sovelluksen käytössä olevia koneita niin että laitteisto-ongelmilta vältytään.



Kuva 7. Cloud Service (Microsoft 19.4.2017.)

4.3.1 Ylläpito ja skaalaus

Sovelluksen julkaisun yhteydessä palvelulle annetaan konfigurointitiedosto, jossa määritetään paljonko sovellus tarvitsee suorituskykyä. Esimerkkisi voidaan määrittää, että tarvitaan kolme web-roolin instanssia ja kaksi worker-roolin instanssia. Lisäksi määritellään minkä kokoisia näiden instanssien tulisi olla. Konfiguroinnin poh-

jalta Azure luo tarvittavat koneet sovelluksen käyttöön. Jos sovelluksen tarpeet myöhemmin muuttuvat, voidaan koneiden määrää ja tehoa muuttaa. (Microsoft 19.4.2017.)

Tyypillisesti sovelluksen julkaisu Azure cloud service -palvelussa tehdään kahdessa vaiheessa. Ensimmäisessä vaiheessa kehittäjä lataa sovelluksen palveluun, mutta sitä ei vielä julkaista käyttäjille (staging area). Toisessa vaiheessa, kun kehittäjä on valmis julkaisemaan sovelluksen, hän vaihtaa sovelluksen tuotantotilaan (production area). Vaihto tehdään Azure portal -palvelun avulla ja se voidaan tehdä ilman että palvelun käyttö häiriintyy. (Microsoft 19.4.2017.)

4.3.2 Automaattinen monitorointi

Azure cloud service pyrkii jatkuvasti tunnistamaan sovelluksessa tapahtuneet virheet. Esimerkiksi jos palvelin fyysisesti sammuu, palvelu käynnistää virtuaalikoneet toisella palvelimella. Se tunnistaa myös, jos yksittäinen virtuaalikone tai sovellus kaatuu ja käynnistää ne uudelleen. (Microsoft 19.4.2017.)

Edellä mainitut asiat tulee ottaa huomioon myös suunniteltaessa sovellusta Azure cloud service -palveluun. Palvelun PaaS-luonteesta johtuen kaikkien sovellusten tulisi toimia oikein myös siinä tilanteessa, jos sovellus yllättäen kaatuu. Sovelluksen ei tulisi tallentaa tilaansa tai muita tietoja virtuaalikoneen tiedostojärjestelmään. Kaikki virtuaalikoneelle kirjoitettu tieto häviää, jos kone kaatuu. Siksi kaikki pysyväksi tarkoitettu tieto tulisi tallentaa ulkopuoliseen palveluun. (Microsoft 19.4.2017.)

4.3.3 Cloud Service -palvelun käyttö opinnäytetyössä

Työssä päädyttiin käyttämään Cloud Services -palvelua koska, se vastaa parhaiten sovelluksen vaatimuksia. Alussa sovellus julkaistiin Azure App Service -palvelussa, mutta myöhemmin alusta jouduttiin vaihtamaan, koska App Service ei mahdollista SelectPdf-ohjelman käyttöä (SelectPdf [Viitattu 13.4]). Sovelluksessa käytetään ainoastaan yhtä web-role-instanssia, koska useamman roolin käytölle ei ole esiintynyt tarvetta.

4.4 Azure SQL Database

Azure SQL Database on yleiskäyttöinen PaaS-tyyppinen tietokantapalvelu, joka tukee eri tietomalleja kuten, relaatio, JSON ja XML. Azure SQL Database käyttää samaa perustaa Microsoft SQL Server tietokantamoottorin kanssa ja näin siinä on tietokannan tasolla lähes samat ominaisuudet kuin Microsoft SQL Server -ohjelmistossa. (Microsoft 3.7.2017.)

Azure SQL Database tukee sovellusten kehittämistä eri ohjelmointikielillä kuten, Python, Java, Node.js, PHP, Ruby, ja .NET. Sitä voidaan käyttää MacOS-, Linux- ja Windows-käyttöjärjestelmissä. Azure SQL Database tukee samoja kirjastoja kuin Microsoft SQL Server. (Microsoft 3.7.2017.)

Azure SQL Database palvelua voidaan käyttää useilla eri työkaluilla, kuten Azure portal, SQL Server Management Studio tai Visual Studio -sovelluskehittimen SQL Server Data Tools -työkalujen avulla. (Microsoft 3.7.2017.)

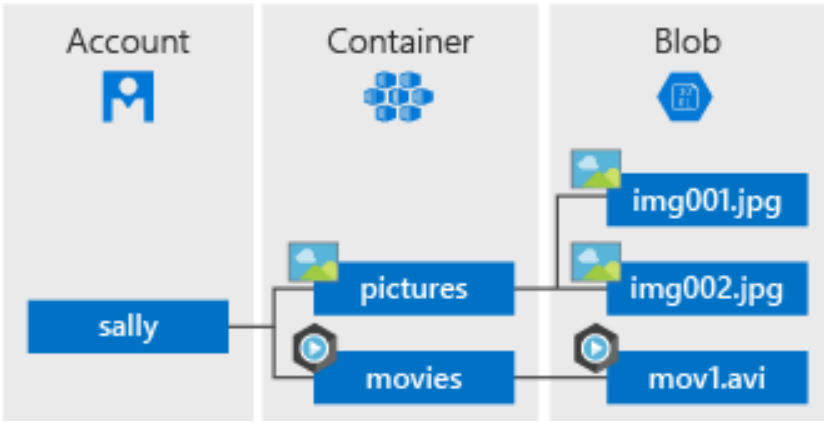
Azure SQL Database luo automaattisesti varmuuskopion tietokannasta ja hajauttaa varmuuskopiot maantieteellisesti eri paikkoihin. Varmuuskopiot säilytetään tavallisesti vähintään seitsemän päivän ajan, mutta tarvittaessa varmuuskopioita voidaan säilyttää jopa kymmenen vuotta. (Microsoft 4.4.2018.)

4.5 Azure blob storage

Azure Blob Storage on pilvitallennuspalvelu, jonka avulla voidaan tallentaa isoja määriä jäsentelemätöntä tietoa, kuten tekstiä tai binääridataa. Tietoon päästään käsiksi käyttämällä URL-osoitetta, REST-rajapintaa tai jotain useista Azure SDK storage client -kirjastoista. Client-kirjastoja on saatavilla useille ohjelmointikielille, kuten Node.js, Java, PHP, Ruby, Python, ja C#. Tieto voidaan asettaa julkiseksi kaikkien saataville tai se voidaan tarjota vain tunnistetuille käyttäjille. (Microsoft 27.3.2018.)

Blob Storage -tietokanta koostuu seuraavista komponenteista: käyttäjätili (account), container ja blob. Komponentit on kuvattu kuvasa 8. Container-tietueet ovat ikään kuin kansioita, joiden avulla tietoa voidaan jäsennellä. Jokaisen blob-tiedoston täytyy olla container-tietueen sisällä. Yksi käyttäjätili voi sisältää äärettömän määrän

container-tietueita ja container voi sisältää äärettömän määrän blob-tiedostoja. Blob voi olla mikä tahansa tiedosto. Azure-palvelussa on tarjolla kolme erilaista blob-tyyppiä: block blobs, append blobs ja page blobs. (Microsoft 27.3.2018.)



Kuva 8. Blob tietorakenne (Microsoft 19.4.2017.)

Block blobs sopii parhaiten teksti- ja binääritiedostojen, kuten pdf-dokumenttien tai mediatiedostojen tallentamiseen. Yksi blob block voi sisältää maksimissaan 50 000 blob-tiedostoa ja yksi blob voi olla kooltaan maksimissaan 100 MB. (Microsoft [Viitattu 8.4.2018].)

Append blob -tiedostot ovat hyvin saman kaltaisia kuin blob block-tiedostot. Ne on kuitenkin optimoitu lisäystoiminnoille, joten ne toimivat hyvin lokitietojen tallentamiseen tai vastaavan tyyppiseen käyttöön. Yksi block blob voi sisältää 50 000 block-tietuetta ja yhden tiedoston maksimikoko on 4 MB. (Microsoft [Viitattu 8.4.2018].)

Page blob voi olla maksimissaan 1 TB:n kokoinen ja se on parhaimmillaan sovelluksissa, joissa suoritetaan jatkuvaa lukemista ja kirjoittamista. Azure-virtuaalikoneet käyttävät bage blob -tiedostoja käyttöjärjestelmänä ja kiintolevyinä. (Microsoft [Viitattu 8.4.2018].)

Blob Storage ei perustu hierarkkiseen tietorakenteeseen, joten siinä ei ole perinteistä kansiorakennetta. Storageen voidaan kuitenkin luoda virtuaalisia hakemistoja, joiden avulla tieto voidaan järjestää kuten perinteisessä hakemistorakenteessa. Virtuaalisia hakemistoja voidaan luoda käyttämällä /-merkkiä tiedostojen nimissä. (Microsoft [Viitattu 9.4.2018].)

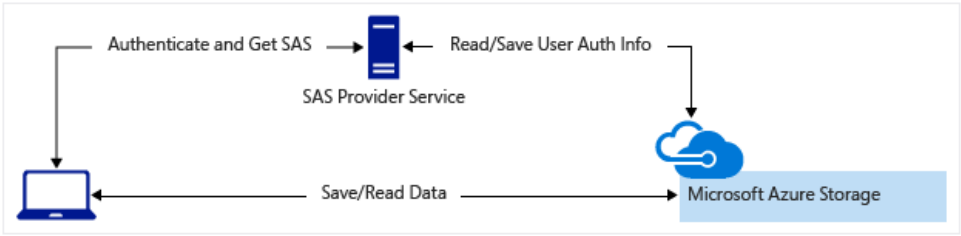
Blob Storage tietoja, kuten tiedostoja ja virtuaalisia hakemistoja, voidaan tarkastella joko selaimen kautta tai Azure Storage Explorer -työpöytäsovelluksen avulla.

4.5.1 Blob storage -palvelun käyttö opinnäytetyössä

Tässä työssä Blob storage -palvelua käytetään pääasiassa pdf-tiedostojen ja erilaisten kuvien tallentamiseen. Työssä päädyttiin käyttämään Block blobs -palvelua koska se sopii parhaiten edellä mainittujen tiedostojen tallentamiseen.

Sovelluksessa on oma luokka, joka hoitaa tiedoston tallentamisen blob storage -palveluun. Samalla tiedostoon liittyvät tiedot tallennetaan SQL-tietokantaan. Näin tiedostoille voidaan helposti luoda relaatioita tietokannan tietoihin. Sovelluksen kautta luodut tiedostot tallennetaan aina saman container-tietueen alle, ja tiedoston nimessä on aina id, joka vastaa tiedoston id:tä tietokannassa. Kun tietokannassa id on määritelty uniikiksi tiedoksi, voidaan varmistua, että tiedoston nimi on aina uniikki, eikä konflikteja pääse syntymään. Sovelluksen käytössä on myös toinen container, johon kehittäjä voi tallentaa staattista sisältöä.

Tiedostojen lukemisessa käytetään apuna shared access signature (SAS) -tekniikkaa. SAS on tekniikka, jonka avulla storage-tilin tiedostoja voidaan jakaa asiakasohjelman kanssa ilman, että käyttäjätunnuksia annetaan asiakassovelluksen käyttöön (Microsoft 18.4.2017). SAS-tekniikan arkkitehtuuri on kuvattu kuvassa 9. SAS on turvallinen tapa jakaa tietoja vaarantamatta tilin käyttäjätunnuksia (Microsoft 18.4.2017).



Kuva 9. SAS arkkitehtuuri (Microsoft 18.4.2017.)

Tässä työssä SAS-tekniikkaa käytettiin lähinnä siksi, että sen avulla voidaan tiedostot ladata blob storage -palvelusta suoraan käyttäjälle, ilman että tietoja reititetään sovelluksen palvelimen kautta. Näin tiedostojen lataaminen on nopeampaa, eikä rasita turhaan sovelluksen palvelinta. Kun käyttäjä haluaa avata sovelluksesta tiedoston, avaa hän linkin blob storage -tiedostoon. Linkki on voimassa määräajan ja se voi sallia tilanteesta riippuen käyttäjälle oikeudet lukea, muokata tai poistaa tiedosto.

4.6 Muut sovelluskehykset ja lisäosat

Sovelluksen toteuttamisessa on käytetty lukuisia lisäosia, joista monet ovat avoimen lähdekoodin JavaScript-kirjastoja tai lisäosia.

Bootbox.js on JavaScript-kirjasto jonka avulla voidaan helposti luoda ohjelmoitavia Bootstrap modal-ikkunoita

Bootstrap on avoimen lähdekoodin sovelluskehys HTML-, CSS- ja JavaScript-kehittämiseen

Bootstrap-datepicker on JavaScript-lisäosa, jonka avulla voidaan luoda helppokäyttöisiä päivämäärävalitsimia

Datatables.js on JavaScript-kirjasto, jonka avulla voidaan luoda vuorovaikutteisia HTML-taulukoita

Fabric.js on JavaScript HTML5 canvas -kirjasto. Sen avulla voidaan luoda kuvankäsittely- ja piirtotyökaluja

Nlog on .NET-sovelluskehykselle kehitetty ilmainen alusta lokitietojen tallentamiseen

Rangeslider.js on Jquery-lisäosa, jonka avulla voidaan luoda HTML5-liukusäätimiä

Select2 on JavaScript lisäosa, jonka avulla voidaan luoda helppokäyttöisiä ja muunneltavia valintatyökaluja

Sortable on JavaScript-kirjasto vedä ja pudota -tekniikalla järjestettävien listojen luomiseen

Typeahead.js on JavaScript-kirjasto, jonka avulla voidaan luoda avustava tekstin-syöttö tekstikenttien täyttämiseen

5 OHJELMISTON TOTEUTUS

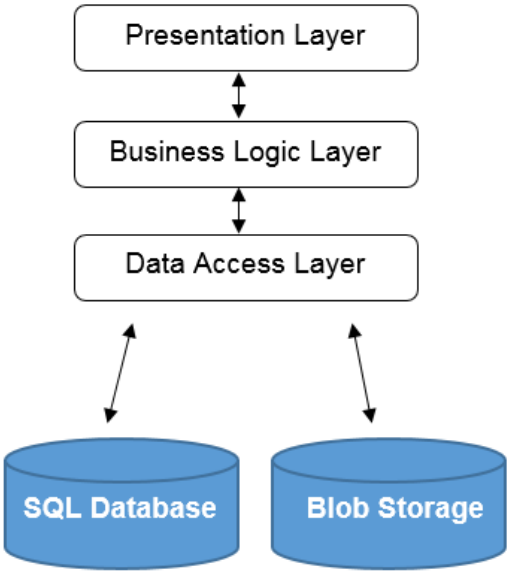
5.1 Arkkitehtuuri

Nykyaikaisessa ohjelmistokehityksessä ohjelmiston rakenne jaetaan käytännössä aina eri kerroksiin, joista kullakin on oma tehtävänsä. Kerrosten tarkoitus on paketoida kokoelma toiminnallisuuksia sovitun rajapinnan taakse. Kerrokset jäsentelevät erityyppiset asiat eri kerroksiin ja samalla mahdollistavat saman järjestelmän eri komponenttien yhteensopivuuden. Olio-ohjelmoinnissa kerros on käytännössä kokoelma luokkia, jotka toteuttavat annetun tehtävän. (Esposito 2016, 164.)

Tässä työssä on käytetty arkkitehtuurina kolmikerrosmallia. Arkkitehtuuri on kuvattu kuvassa 10. Ylimpänä on esityskerros (presentation layer), jonka tehtävänä on sovelluksen käyttöliittymän toteuttaminen. Esityskerros on totutettu ASP.NET MVC -sovelluskehyksen avulla, jonka sisäisestä arkkitehtuurista on kerrottu luvussa 4.1. Yksi esityskerroksen osa on verkkoselaimella suoritettava JavaScript-koodi, jonka tehtävänä on välittää käyttäjän pyyntöjä palvelimelle ja manipuloida HTML-sivua. Tässä sovelluksessa verkkoselaimella suoritettava koodin määrä on pyritty pitämään suhteellisen vähäisenä.

Toimintalogiikkakerros (business logic layer) on taso, jossa tapahtuu sovelluksen varsinainen toimintalogiikka. Toimintalogiikkakerros on luokkakirjasto, jonka luokat käyttävät tietomalleina pääosin tietokerroksen (data access layer) luokkia.

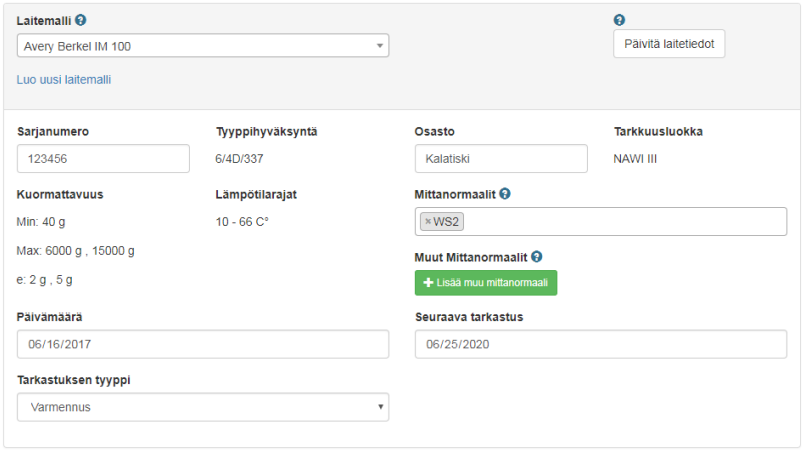
Tietokerros huolehtii tiedon lukemisesta ja kirjoittamisesta tietokantaan ja muihin ulkopuolisiin tietovarastoihin. Sovelluksen tietokantakommunnikoinnissa on käytetty apuna Entity Framework -sovelluskehyksen database first -tekniikkaa. Entity Framework generoi automaattisesti kustakin tietokannan taulusta luokan, joka on abstraktio taulusta. Näin tietokantataulun tietuetta voidaan käsitellä oliona ohjelmiston toimintalogiikkakerroksessa.



Kuva 10. Sovelluksen arkkitehtuuri

5.2 Tiedon tallentaminen

Sovelluksessa on käytetty erityyppisiä tallennusmetodeita käyttäjän syöttämien tietojen tallentamiseksi tietokantaan. Tässä luvussa kerrotaan esimerkkinä miten vaa'an tarkastuslomakkeen laitetiedot tallennetaan tietokantaan. Kuvakaappaus laitetiedot-paneelista on nähtävissä kuvassa 11. Tarkastuslomakkeen tietojen tallennus on toteutettu asynkronisesti siten, että tieto tallennetaan tietokantaan aina, kun yksittäisen kentän arvo muuttuu.



Kuva 11. Kuvakaappaus verkkoselaimelta sovelluksen laitetiedot-paneelista

HTML sivun elementteihin on asetettu jQuery .change() -toiminto, joka kutsuu saveInstrumentInfo-JavaScript-funktiota aina, kun kentän arvo muuttuu. Funktio on esitetty kuvassa 12. Funktio saa parametreina kentän arvon, id:n ja muut tarvittavatat tiedot. Funktio tekee palvelimelle HTTP-POST-pyynnön. Pyyntö kutsuu JobController-luokan SaveInstrumentProperty-metodia, joka saa parametreinä edelleen samat muuttujat, metodin koodi on nähtävissä kuvassa 12.

```
function saveInstrumentInfo(id, key, value, input, detailId) {

    $.post("/WeighingInstruments/Job/SaveInstrumentProperty",
    {
        id: id,
        key: key,
        value: value,
        detailId: detailId,
        beforeSend: function () {
            input.parent().addClass("has-error");
        },
    },
    function (data) {
        if (data.success) {
            input.parent().removeClass("has-error");
        } else {
            input.parent().addClass("has-error");
        }
    });
};
```

Kuva 12 JavaScript-tallennusfunktio 1

```
0 references
public JsonResult SaveInstrumentProperty(long id, string key, string value, long detailId = 0)
{
    logger.Info("{0}: Browser:{1}, Parameters: {2}, {3}, {4}",
        Helper.GetCurrentClassAndMethod(), Request.Browser.Type, id, key, value);

    var dp = new WeighingInstrumentVerificationJobDataProvider(CurrentUser);
    dp.SaveInstrumentProperty(id, key, value, detailId);

    return Json(new
    {
        success = true,
        message = "saved"
    });
}
```

Kuva 13. C#-tallennusmetodi 1

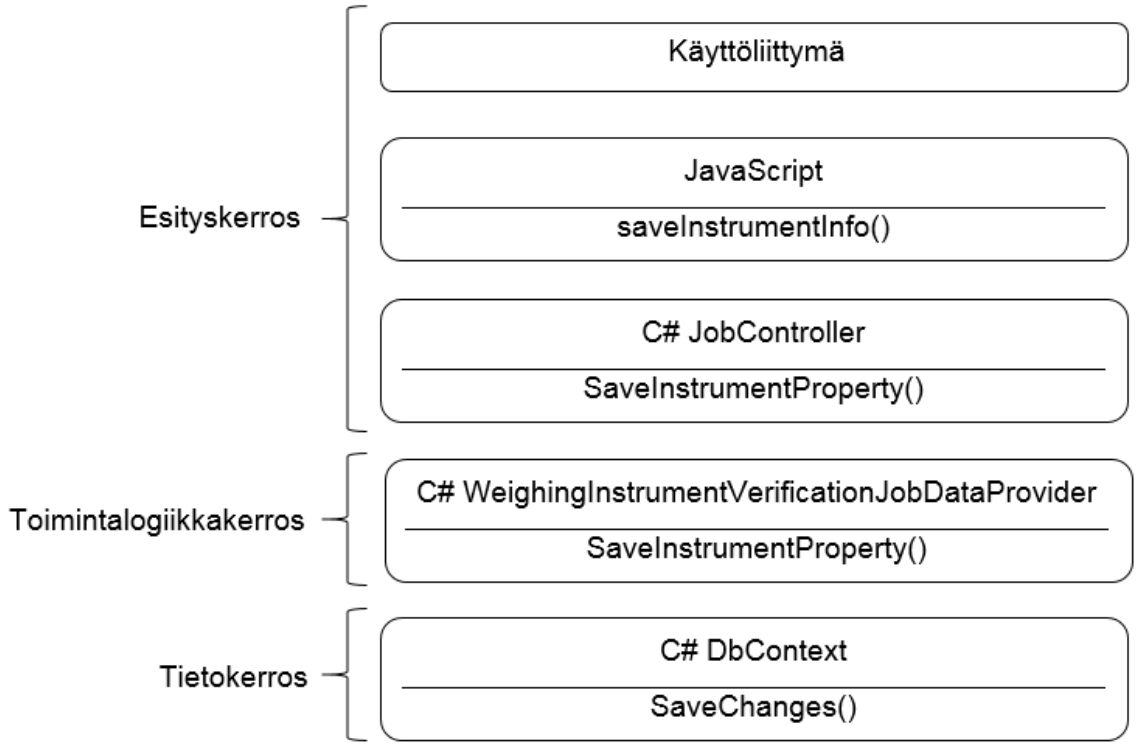
Kun pyyntö on saapunut kuvassa 13 esitetyn JobController-luokan SaveInstrument-Property-metodile, se kutsuu edelleen WeighingInstrumentVerificationJobDataPro-

vider-luokan samannimistä metodia, joka on esitetty kuvassa 14. Tämä metodi tekee valinnan, mihin tieto tallennetaan ja välittää tiedot edelleen tietokerrokselle, joka huolehtii tiedon kirjoittamisesta tietokantaan.

```
1 reference
public void SaveInstrumentProperty(long instrumentId, string name, dynamic value, long detailId)
{
    using (var db = new InstCertDbContext())
    {

        var instrument = db.WeighingInstruments.SingleOrDefault(d => d.Id == instrumentId);
        if (instrument == null)
            throw new ArgumentOutOfRangeException("instrumentId");

        switch (name)
        {
            case "Department":
                instrument.Department = value;
                break;
            case "SerialNumber":
                instrument.SerialNumber = value;
                break;

           // koodia lyhennetty esimerkin vuoksi

            default:
                throw new ArgumentOutOfRangeException("name");
        }

        instrument.ModificationDate = DateTime.Now;
        instrument.ModifiedBy = CurrentUser.UserName;

        db.SaveChanges();
    }
}
```

Kuva 14. C#-tallennusmetodi 2

Esimerkin tallennustoiminnon kulku sovelluksen arkkitehtuurissa on kuvattu kuvassa 15.



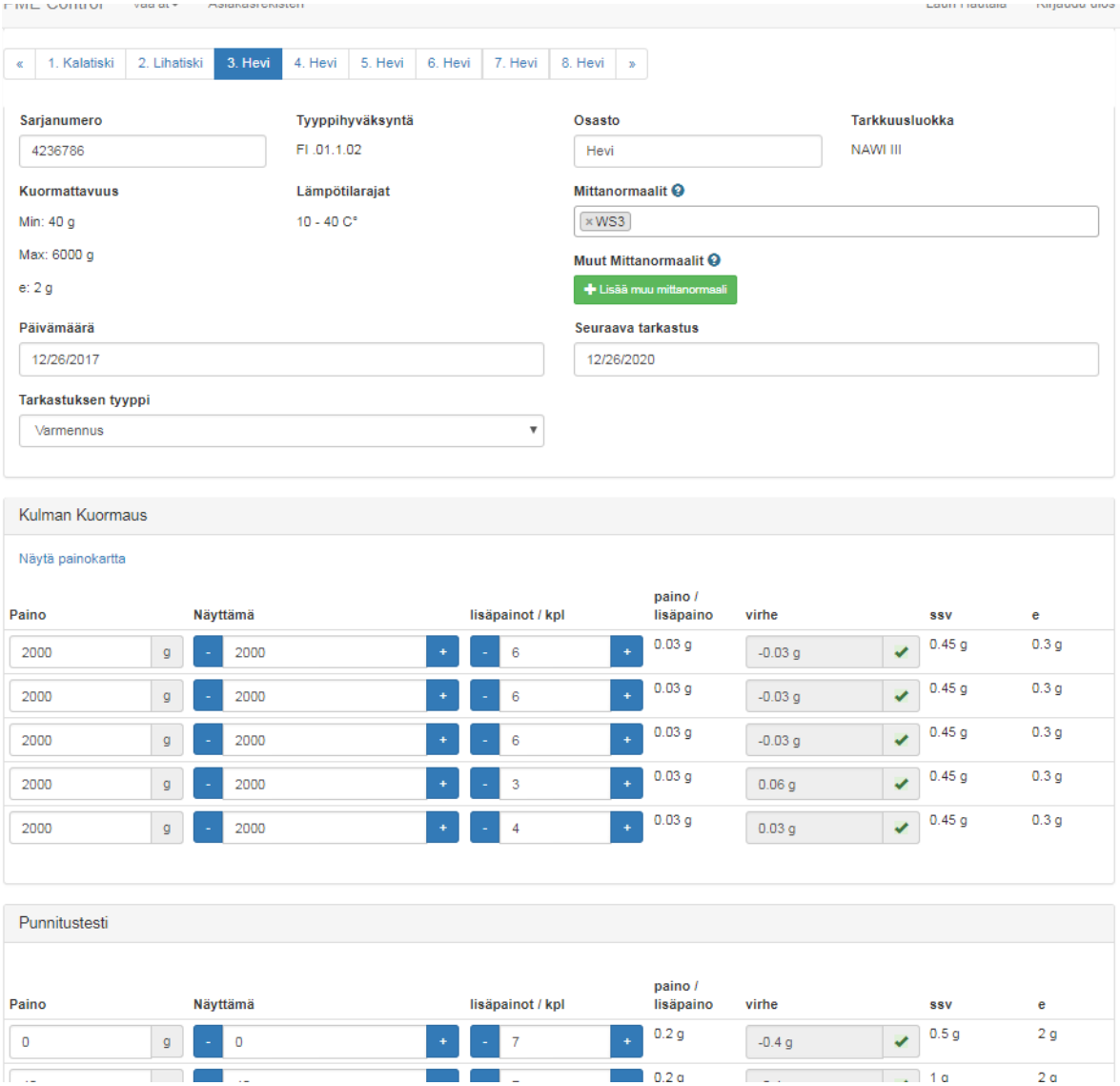
Kuva 15. Tallennuksen kulku arkkitehtuurin kerroksissa

5.3 Sovelluksen näkymät

Sovellus koostuu lukuisista eri näkymistä, tässä luvussa kerrotaan joistain oleellisimmista näkymistä ja niiden toiminnoista.

5.3.1 Vaa’an tarkastuslomake

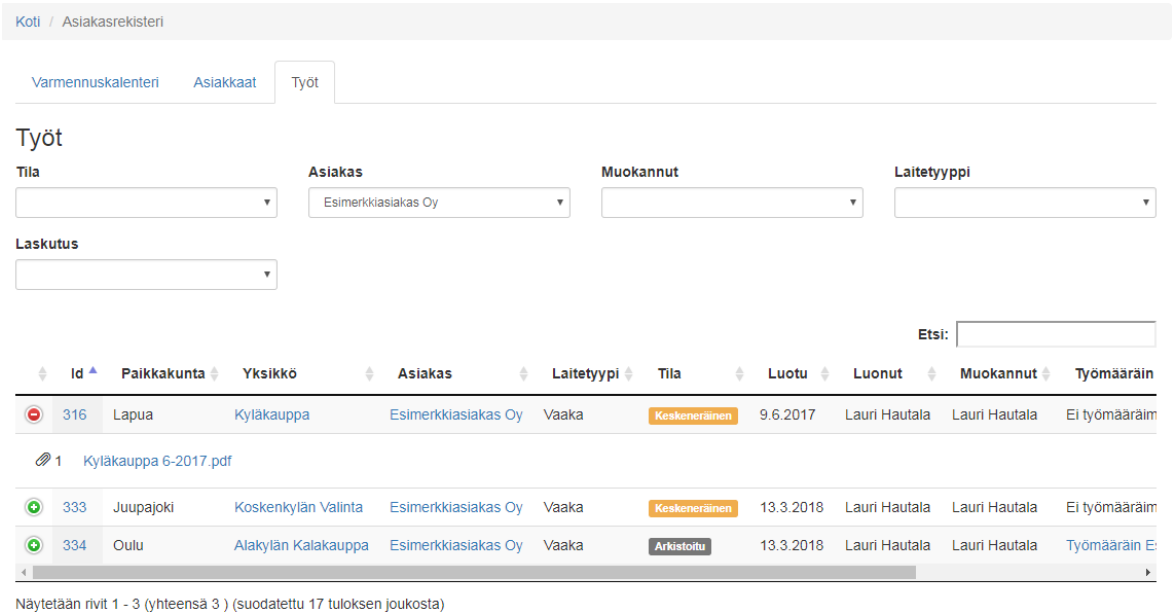
Vaa’an tarkastuslomakkeella, joka on esitetty kuvassa 16, käyttäjä syöttää lomakkeelle tarvittavat tekniset tiedot, käytetyt mittausvälineet ja mittaustulokset. Mittaustulosten syöttämisen helpottamiseksi on tekstikentän vierelle lisätty painikkeet, joilla tulosta voi muuttaa askel kerrallaan ylös- tai alaspäin. Käyttäjän ei tarvitse erikseen tallentaa lomakkeen tietoja, vaan tiedot tallentuvat automaattisesti aina, kun niitä muokataan.



Kuva 16. Kuvakaappaus mittaustietolomakkeesta

5.3.2 Asiakasrekisteri

Asiakasrekisterin tietoja voi tarkastella monella eri tapaa. Asiakasrekisteri sisältää tällä hetkellä kolme erityyppistä listausta: varmennuskalenteri, asiakkaat ja työt. Jokainen näkymä löytyy omalta välilehdeltään. Tietojen etsimisessä voidaan käyttää apuna erilaisia suodattimia sekä hakutoimintoa, kuten kuvasta 17 voidaan nähdä. Lisäksi jokaisella asiakkaalla on oma näkymä, jossa on listattu asiakkaalle kuuluvat yksiköt, työt ja työmääräimet. Myös yksiköillä on samankaltaiset näkymät. Töiden listauksesta päästään tarkastelemaan työn liitetiedostoja suoraan +-ikonia klikkaamalla, kuten kuvasta 17 ilmenee.



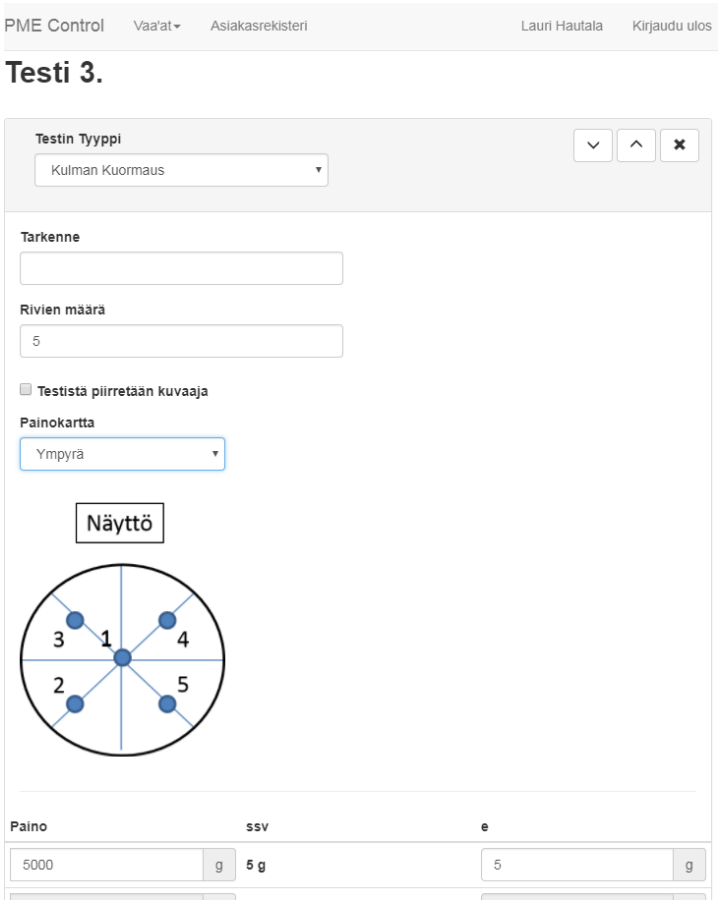
Kuva 17. Kuvakaappaus asiakasrekisteristä

5.3.3 Laitemallin hallinta

Laitemallin hallinnassa käyttäjä voi määritellä uuden laitemallin tai muokata olemassa olevaa. Jokaiselle laitekategorialle on hieman erityyppinen näkymä. Näkymässä määritellään laitteen tekniset tiedot ja laitteelle tehtävät testit. Testien määrä voi vaihdella, ja jokaisen testin yksityiskohdat voidaan määritellä erikseen.

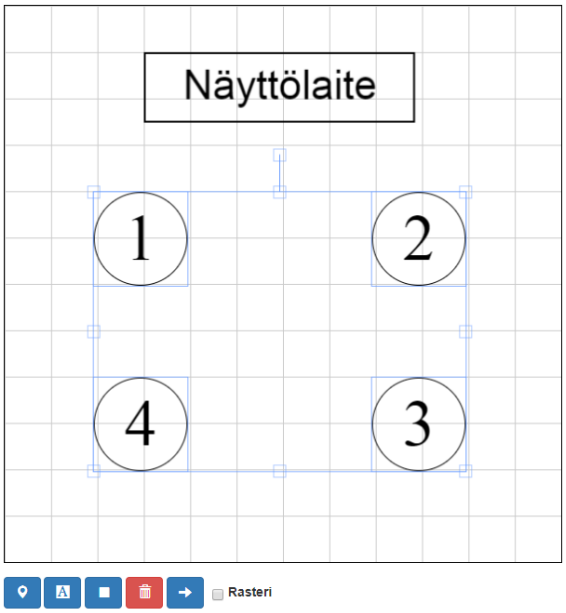
Laitemallin hallinnassa on myös toiminto, jonka avulla sovellus etsii käyttäjän antamien tietojen perusteella mahdolliset samankaltaiset laitemallit tietokannasta ja kopio valitun laitemallin testit uudelle laitteelle. Kopioinnin jälkeen käyttäjä voi vielä halutessaan muokata testejä.

Kuvassa 18 on näkymä vaakamallin yksittäisen testin määrittelystä.



Kuva 18. Kuvakaappaus vaa’an laitemallin hallintanäkymästä

Vaakamallin määrittelyssä käyttäjä voi valita ennalta määritellyistä kuvista mittauspisteiden sijoittelusta kertovan havainnekuvan. Vaihtoehtoisesti käyttäjä voi piirtää oman kuvan. Kuvassa 19 on kuvakaappaus havainnekuvan piirtämisestä.



Kuva 19. Kuvakaappaus vaa'an havainnekuvan piirrosta

5.3.4 Tarkastuspöytäkirja

Jokaisesta tarkastetusta laitteesta luodaan lopuksi tarkastuspöytäkirja pdf-muodossa. Sovellus generoi pdf-tiedoston automaattisesti, kun työ merkitään valmiiksi. Liitteessä 1 on esimerkki tarkastuspöytäkirjasta.

6 YHTEENVETO JA POHDINTA

Työn tavoitteena oli luoda nykyaikainen tietojärjestelmä toimeksiantajan käyttöön. Työhön ryhdyttäessä ei opinnäytetyön tekijällä ollut aiempaa kokemusta web-sovelluksen toteuttamisesta, ja kokemus ohjelmoinnista ylipäätään perustui pitkälti opiskeluprojekteihin. Alkuvaiheessa saatu tuki kokeneemmilta tekijöiltä olikin äärimmäisen tärkeää ja loi hyvän pohjan projektille.

Työssä ei kohdattu mitään yksittäistä isoa ongelmaa, mutta nykyaikaisen, responsiivisen ja mobiiliystävällisen käyttöliittymän toteuttaminen vaati enemmän työtä, kuin oli ehkä alun perin arvioitu. Työ oli todella vaativa ja opetti tekijälleen paljon eri tekniikoita ja ohjelmointikieliä, kuten C#, JavaScript, Html, Azure-pilvipalvelut, GIT-versionhallinta ja SQL-tietokannat. Lisäksi tuli paljon tietämystä tietosuoja-asioista, sekä mittauslaitteisiin liittyvästä lainsäädännöstä.

Tällä hetkellä ohjelmisto on toimeksiantajalla päivittäisessä käytössä. Sovelluksen tuomat edut verrattuna aiemmin käytössä olleisiin laskentataulukoihin ovat kiistattomat. Joiltain osin sovellus vaatii vielä jatkokehitystä, koska matkan varrella on huomattu uusia vaatimuksia, joita ei tiedetty tai osattu nähdä alkuperäistä vaatimusmäärittelyä tehtäessä. Työlle asetetut tavoitteet saavutettiin pääosin ja lopputulokseen voidaan olla tyytyväisiä.

LÄHTEET

Avi. 16.1.2017. Mittauslaitevalvonta [www-dokumentti]. Aluehallintovirasto. [Viitattu 30.4.2018]. Saatavissa: https://www.avi.fi/web/avi/mittauslaitevalvonta

Esposito, D. 2016. Modern Web Development: Understanding domains, technologies, and user experience. [Verkkokirja]. Redmond, Washington, USA: Microsoft Corporation. [Viitattu 16.5.2018]. Saatavana Microsoft press-palvelusta. Vaatii käyttöoikeuden.

Hiipakka, J. 2018. Toimitusjohtaja. Pme-Control Oy. Sähköpostihaastattelu 29.5.2018.

Microsoft. 18.4.2017. Using shared access signatures (SAS). [www-dokumentti]. Microsoft Corporation. [Viitattu 10.4.2018]. Saatavissa: https://docs.microsoft.com/en-us/azure/storage/common/storage-dotnet-shared-access-signature-part-1

Microsoft. 19.4.2017. Should I choose Azure Cloud Services or something else? [www-dokumentti]. Microsoft Corporation. [Viitattu 3.4.2018]. Saatavissa: https://docs.microsoft.com/en-us/azure/cloud-services/cloud-services-choose-me

Microsoft. 27.3.2018. Introduction to object storage in Azure. [www-dokumentti]. Microsoft Corporation. [Viitattu 7.4.2018]. Saatavissa: https://docs.microsoft.com/en-us/azure/storage/blobs/storage-blobs-introduction

Microsoft. 3.7.2017. What is the Azure SQL Database service? [www-dokumentti]. Microsoft Corporation. [Viitattu 5.4.2018]. Saatavissa: https://docs.microsoft.com/en-us/azure/sql-database/sql-database-technical-overview

Microsoft. 4.4.2018. Learn about automatic SQL Database backups. [www-dokumentti]. Microsoft Corporation. [Viitattu 6.4.2018]. Saatavissa: https://docs.microsoft.com/en-us/azure/sql-database/sql-database-automated-backups

Microsoft. 7.7.2016. Azure App Service, Virtual Machines, Service Fabric, and Cloud Services comparison. [www-dokumentti]. Microsoft Corporation. [Viitattu 4.4.2018]. Saatavissa: https://docs.microsoft.com/en-us/azure/app-service/choose-web-site-cloud-service-vm

Microsoft. Ei päiväystä. ASP.NET MVC Overview. [www-dokumentti]. Microsoft Corporation. [Viitattu 1.4.2018]. Saatavissa: https://msdn.microsoft.com/en-us/library/dd381412(v=vs.108).aspx

Microsoft. Ei päiväystä. Naming and Referencing Containers, Blobs, and Metadata. [www-dokumentti]. Microsoft Corporation. [Viitattu 9.4.2018]. Saatavissa: https://docs.microsoft.com/en-us/rest/api/storageservices/naming-and-referencing-containers--blobs--and-metadata

Microsoft. Ei päiväystä. Understanding Block Blobs, Append Blobs, and Page Blobs. [www-dokumentti]. Microsoft Corporation. [Viitattu 8.4.2018]. Saatavissa: https://docs.microsoft.com/en-us/rest/api/storageservices/understanding-block-blobs--append-blobs--and-page-blobs

Microsoft. Ei päiväystä. What is Azure? [www-dokumentti]. Microsoft Corporation [Viitattu 2.4.2018]. Saatavissa: https://azure.microsoft.com/en-gb/overview/what-is-azure

OIML R76-1. 2006. Non-automatic weighing instruments [verkkojulkaisu]. Organisation Internationale de Métrologie Légale. [Viitattu 26.4.2018]. Saatavissa: https://www.oiml.org/en/files/pdf_r/r076-p-e06.pdf

Pme-Control. Ei päiväystä. Huoltoasemamittarit ja vaa'at [verkkojulkaisu]. Pme-Control Oy [Viitattu 26.4.2018]. Saatavissa: http://www.pme-control.com/07/index.php?sivu=huoltoasemamittarit

Pusa, A., Riski, K. & Ojanen-Saloranta, M. (toim.) 2017. Vaakojen kalibrointiopas. [Verkkokirja]. Espoo: Teknologian tutkimuskesus VTT Oy. [Viitattu 1.5.2018]. Saatavissa: http://www.vtt.fi/inf/pdf/technology/2017/T286.pdf

Rauhalaakso, J. 2018. Tarkastusinsinööri. Pme-Control Oy. Puhelinhaastattelu 27.4.2018.

SelectPdf. Ei päiväystä. Deployment to Windows Azure [www-dokumentti]. SelectPdf. [Viitattu 13.4.2018]. Saatavissa: https://selectpdf.com/docs/Deployment.html

Talus, A., Autio, E., Hänninen, A., Pihamaa, H. & Kantonen, S. 2017. Selvityksiä ja ohjeita 4:2017, Miten valmistautua EU:n tietosuoja-asetukseen [Verkkokirja]. Helsinki: Oikeusministeriö ja tietosuojavalutuutetun toimisto. [Viitattu 8.5.2018]. Saatavissa: http://www.tietosuoja.fi/material/attachments/tietosuojavaltuutettu/tietosuojavaltuutetuntoimisto/oppaat/1Em8rT7IF/Miten_valmistautua_EUn_tietosuoja-asetukseen.pdf

Tietosuojavaltuutetun toimisto. 23.3.2018. EU:n tietosuojauudistus [www-dokumentti]. Tietosuojavaltuutetun toimisto. [Viitattu 7.5.2018]. Saatavissa: http://www.tietosuoja.fi/fi/index/euntietosuojauudistus.html

Tukes. 16.1.2017 a. Mittauslaitteiden varmentamisen menettely [www-dokumentti]. Turvallisuus- ja kemikaalivirasto (Tukes). [Viitattu 30.4.2018]. Saatavissa: http://www.tukes.fi/fi/Toimialat/Mittauslaitteet/markkinoille-saattaminen

Tukes. 16.1.2017 b. Mittauslaitteiden varmentamisen menettely. [www-dokumentti]. Turvallisuus- ja kemikaalivirasto (Tukes). [Viitattu 1.5.2018]. Saatavissa: http://www.tukes.fi/fi/Toimialat/Mittauslaitteet/Mittauslaitteen-vaatimukset

LIITTEET

Liite 1. Tarkastuspöytäkirja

Varmennustodistus Nro M18 M04 - 123456

Päivämäärä 5.4.2018

Asiakas	Testiasiakas Oy
Yksikkö	Testiasiakas Seinäjoen Tehdas
Y-tunnus	1-123456
Osoite	Testikatu 3 60100 Seinäjoki

Osasto	Pakkaamo
Laite	Bizerba SC II 500
Sarjanumero	123456
Tyyppihyväksymä	D10-09-010
Tarkkuusluokka	NAWI III
Kuormattavuus	Min: 40 g Max: 6000 g , 15000 g
Askelarvo	e: 2 g , 5 g
Lämpötilarajat	10 - 40 C°

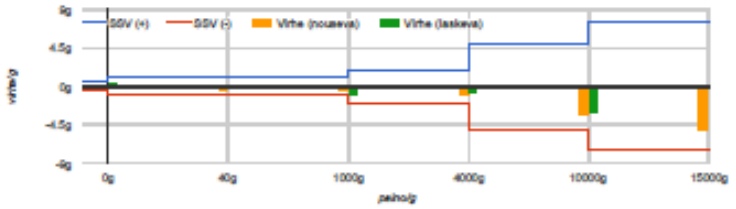
Kulman Kuormaus

Paino	Näyttämä	lisäpainot / kpl	paino / lisäpaino	virhe	e
5000 g	5000 g	4	0.2 g	0.2 g	2 g
5000 g	5000 g	4	0.2 g	0.2 g	2 g
5000 g	5000 g	3	0.2 g	0.4 g	2 g
5000 g	5000 g	7	0.2 g	-0.4 g	2 g
5000 g	5000 g	6	0.2 g	-0.2 g	2 g

Punnitustesti

Paino	Näyttämä	lisäpainot / kpl	paino / lisäpaino	virhe	e
0 g	0 g	6	0.2 g	-0.2 g	2 g
40 g	40 g	7	0.2 g	-0.4 g	2 g
1000 g	1000 g	7	0.2 g	-0.4 g	2 g
4000 g	4000 g	10	0.2 g	-1 g	2 g
10000 g	9995 g	2	0.5 g	-3.5 g	5 g
15000 g	14995 g	5	0.5 g	-5 g	5 g
10000 g	9995 g	1	0.5 g	-3 g	5 g
4000 g	4000 g	9	0.2 g	-0.8 g	2 g
1000 g	1000 g	10	0.2 g	-1 g	2 g
40 g	40 g	5	0.2 g	0 g	2 g
0 g	0 g	3	0.2 g	0.4 g	2 g

Hystereesi



Toistokykytesti

Paino	Näyttämä	lisäpainot / kpl	paino / lisäpaino	virhe	e
12000 g	11995 g	3	0.5 g	-4 g	5 g
12000 g	11995 g	1	0.5 g	-3 g	5 g
12000 g	12000 g	8	0.5 g	-1.5 g	5 g

Muut tiedot

Tarkastuksessa käytetyt mittanormaalit

Punnussarja F1, Kalibrointitodistus K019-12019

Mittauslaitetta ei viritetty tarkastuksen yhteydessä

Mittausolosuhteet:

Lämpötila noin 20 °C

Päätelmä

Mittalaite täyttää vaatimukset.
Seuraava varmennus: 05.04.2021

Laitteisto on varmennettu standardin OIML R76 mukaisesti kansallisesti jäljitettävillä menetelmillä. Varmennusperuste on PME vaakatyöohjeet. Mittausepävarmuus 1/3 varmennusvirherajasta. Varmennus on voimassa, kun laitteisto on tyyppihyväksyntäpäätöksen mukainen ja sinetit ehjät. Oikaisuvaatimukset: joni.hilpakka@pme-control.com tai puhelimitse +358400162165

Tarkastajan allekirjoitus ja nimen selvennys

Lauri Hautala

PME-Control Oy
Luhtatie 4 A 3
66300 Jurva

Puhelin +358162 165
www.pme-control.com
Y-tunnus: 2061463-2

FINAS