

Tuotannon IoT -laitteiden keskitetty hallinta

Gateway -laitteiden keskitetty ohjelmistopäivitys

LAB-ammattikorkeakoulu

Insinööri (AMK), Tieto- ja viestintäteknikka

2021

Kimi Nygren

Tiivistelmä

Tekijä(t) Nygren, Kimi	Julkaisun laji Opinnäytetyö, AMK Sivumäärä 30	Valmistumisaika Kevät 2021
Työn nimi Tuotannon IoT -laitteiden keskitetty hallinta Gateway -laitteiden keskitetty ohjelmistopäivitys		
Tutkinto Insinööri (AMK)		
Toimeksiantajan nimi, titteli ja organisaatio Markus Järvinen, mScales tuotetiimin vetäjä, Lahti Precision Oy		
Tiivistelmä <p>Opinnäytetyön tavoitteena oli kehittää tuotannon IoT laitteiden hallintaa ja työn pääpainona oli ohjelmistoversioiden jakelun automatisointi. Tavoitteena oli kehittää MVP mallin mukainen järjestelmä hajautettujen IoT laitteiden ohjelmistopäivitysten automatisoinnista. Järjestelmä kehitettiin Lahti Precision Oy:lle osaksi mScales digitaalista punnituspalvelua.</p> <p>Järjestelmä koostui kolmesta pienemmästä osa-alueesta: CI/CD -putkistosta, verkkopalvelusta, sekä IoT-laitteiden päivittäjäohjelmasta. Nämä kolme erillistä järjestelmän osaa toimivat keskenään lähes täysin automaattisesti ja vain hallittavuuden kannalta pakolliset asiat on jätetty manuaalisesti operoitavaksi.</p> <p>Lopputuloksen oli minimaalinen, mutta toimiva järjestelmä, mistä voidaan hallita hajautettujen IoT-laitteiden ohjelmistoversioita hallitusti ja kootusti. Työn aikana kerättiin jatkokehitys ideoita, ja niitä lähdetään kehittämään lähitulevaisuudessa.</p>		
Asiasanat mScales, IoT, pilvipalvelu		

Abstract

Author(s) Nygren, Kimi	Type of Publication Bachelor's thesis	Published Spring 2021
	Number of Pages 31	
Title of Publication Centralized management of industrial IoT-devices Centralized software updating of gateway devices		
Name of Degree Bachelor's Degree Programme in Information and Communications Technology		
Name, title and organization of the client Markus Järvinen, mScales production manager, Lahti Precision Oy		
Abstract <p>The topic of the thesis was centralized management of industrial IoT devices, and the focus of the work was the automation of the distribution of software versions. The system was made for Lahti Precision Oy and as part of their mScales digital weighing service.</p> <p>The system consisted of three smaller components: CI/CD -pipeline, cloud service and the IoT device updater program. These three components cooperate almost fully automatically and only the mandatory matters have been left to be operated manually.</p> <p>The result was a minimalistic but functional system from which software versions of distributed IoT devices can be managed in a controlled and aggregated manner. During the work, further development ideas were collected and will be developed in the near future.</p>		
Keywords mScales, IoT, Cloud computing		

Sisällys

1	Johdanto.....	1
2	Esineiden internet.....	2
2.1	Yleistä.....	2
2.2	IoT laite.....	2
2.3	Teollisuuden Internet	4
2.4	Tulevaisuus	4
3	DevOps	5
3.1	Yleistä.....	5
3.2	Ketterä kehitys.....	6
3.3	CI/CD -putkisto	6
3.3.1	Jatkuva integraatio	7
3.3.2	Jatkuva julkaisu ja toimitus	8
4	Pilvipalvelut.....	9
5	Ohjelmistojen päivittäminen	11
6	MVP malli	12
7	Toimintaympäristö	13
7.1	mScales punnituspalvelu	13
7.2	mScales verkkopalvelu	15
7.3	Vaa'at ja oheislaitteet.....	15
7.4	IoT gatewayt.....	16
7.5	Nykytila.....	16
8	Toteutus	18
8.1	Tiedon keruu ja suunnittelu.....	18
8.2	Ohjelmistoversioiden kehittäminen ja toimittaminen.....	19
8.2.1	Versionhallinta	19
8.2.2	CI/CD -putkisto	19
8.3	Ohjelmistoversioiden hallinta ja jakaminen	21
8.3.1	Ohjelmistoversioiden hallinta	21
8.3.2	Ohejlmistoversiotietojen jakaminen.....	22
8.3.3	Ohjelmistoversioiden jakaminen	23
8.4	Ohjelmistoversioiden tiedustelu ja lataaminen	23
8.4.1	Päivittäjä.....	24
8.4.2	Tilanvalvoja	25
8.5	Kokonaiskuva	26

9 Yhteenveto	27
Lähteet	28

Käsitteet

API	Lyhenne sanoista Application Programming Interface eli ohjelmointirajapinta. Rajapintojen avulla ohjelmat voivat keskustella keskenään.
Gateway	Fyysinen IoT -laite tai sovellus, joka liittää pilvipalvelun kontrollereihin, sensoreihin tai muihin laitteisiin.
IoT	Lyhenteellä IoT viitataan termiin Internet of Things eli suomeksi esineiden internet. Esineiden internetillä tarkoitetaan interneettiin liitettyjä esineitä
Lähdekoodi	Tarkoitetaan tavallisessa tekstimuodossa kirjoitettua kuvausta ohjelmistosta, joka toteutetaan jollain ohjelmistokielellä.
MVP -malli	Lyhenne sanoista Minimum Viable Product eli pienin julkaisukelpoinen tuote. Tämä on toimintakonsepti, jolla yritykset pyrkivät selvittämään mahdollisimman nopeasti uuden tuotteen markkinakelpoisuuden
OTA	Lyhenne sanoista Over The Air, jota käytetään ohjelmistopäivitysten yhteydessä ja tarkoitetaan verkon yli tapahtuvasta ohjelmistopäivityksestä.
Pilvipalvelu	Pilvipalvelu tai verkkopalvelu on internetin välityksellä toimiva ohjelmistopalvelu
URL	Lyhenne sanoista Uniform Resource Locator, joka kertoo tietyn tiedon paikka. Käytetään usein osoittamaan verkkosivuja.

1 Johdanto

Viimeiset vuosikymmenet ovat olleet rajua digitalisoitumisen ajanjaksoa, minkä seurauksena useat arjen rutiinit helpottuvat. Digitalisaatio näkyy tänä päivänä ihan kaikkialla, niin myös teollisuudessa.

Lahti Precision Oy on lahtelainen teollisuuden punnitus-, annostus- ja palveluratkaisuiden valmistaja sekä toimittaja. Alallaan yritys on johtava ja edelläkävijä digitaalisten palveluiden tarjoaja. Lahti Precisionin digitaalisen punnituspalvelun mScalesin avulla asiakas pystyy liittämään vaakansa liiketoimintajärjestelmäänsä. mScales-punnituspalvelu automatisoi punnitusprosessin ja punnitukset hallitaan reaaliaikaisesti. Kaikki punnituksiin liittyvät tiedot ovat sidosryhmien käytettävissä. (Lahti Precision Oy, 2020a.)

Asiakkaiden vaa'at ja oheislaitteet ovat liitettävissä mScales palveluun hyödyntäen IoT:ta (Internet of Things), joka on suuressa roolissa tässä opinnäytetyössä. mScalsein myynnin, sekä asiakkaiden IoT laitteiden määrän kasvaminen tuo tullessaan haasteita laitteiden hallittavuuden kanssa.

Opinnäytetyön tavoitteena on toteuttaa MVP malli järjestelmästä, jonka avulla voidaan hallita toimeksiantajan hajautettuja IoT laitteita verkkopalvelusta käsin. Pääpainona työssä on IoT laitteiden ohjelmistoversioiden jakelun automatisointi verkkopalvelun kautta. Kehitettävällä järjestelmällä saavutetaan helpompi ja nopeampi tapa hallita kaikkia laitteita kootusti. Työn tuloksena säästetään ihmistyötunteja laitteiden hallinnassa, sekä minimoidaan inhimillisten virheiden tuomaa riskiä.

Opinnäytetyön raportissa esitellään toteutettavaa järjestelmää, sen osia ja toimintaperiaatteita yleisellä tasolla, mutta tarkkoihin teknologioihin, protokolliin tai työkaluihin ei puututa toimeksiantajan toiveesta. Raportista käy ilmi, mitä kaikkea vastaava järjestelmä voi sisältää.

2 Esineiden internet

2.1 Yleistä

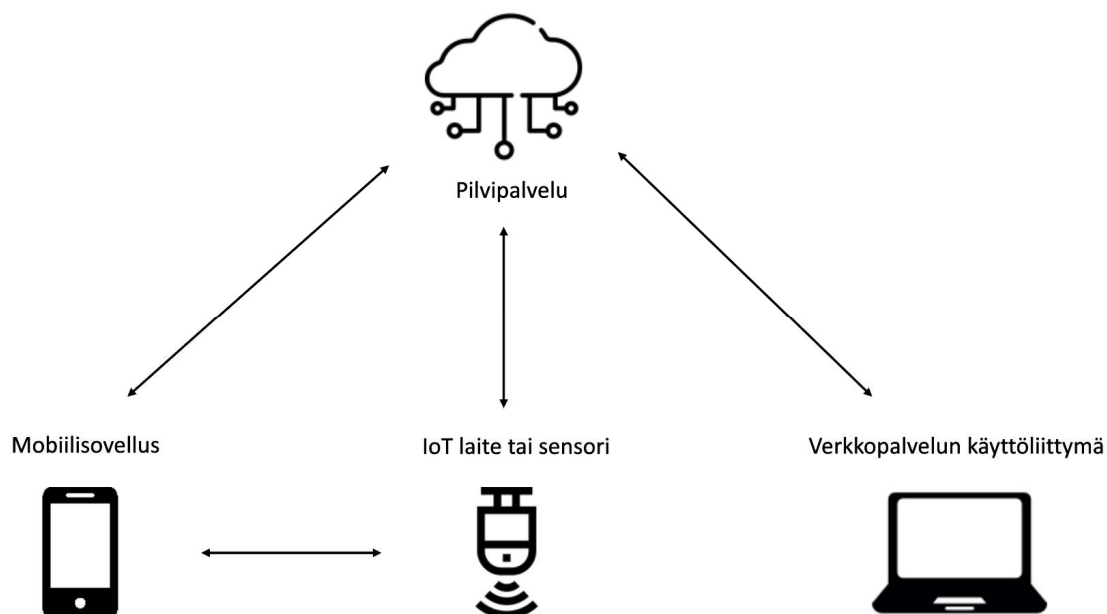
Internet of Things (IoT), eli suomeksi esineiden internet on termi, joka on yleistynyt viimeisen vuosikymmenen aikana eksponentiaalisesti. Esineiden internetillä tarkoitetaan verkkoa, johon on liitetty fyysisiä "esineitä", joilla on sensoreita, ohjelmistoja tai muita teknologioita, joilla voidaan yhdistää laitteita toisiinsa ja siirtää dataa internetin ylitse. Datalla tarkoitetaan tietoa, mitä saadaan esimerkeiksi sensoreilta, kuten lämpötila-arvoja. Yksittäinen laite voi joko tuottaa tai vastaanottaa dataa tai sekä että. Jaettujen tietojen perusteella laitteet voivat suorittaa toimenpiteitä itsenäisesti ilman ihmisen antamia käskyjä. (Oracle & Empirica Finland Oy.)

Perinteisillä kuluttajalle suunnatuilla IoT laitteilla pyritään helpottamaan arjen rutiineja kahvinkeitosta autolla ajamiseen. IoT mahdollistaa säästämään rahaa, aikaa ja vaivaa. Useat kuluttajalle suunnatut IoT laitteet ovat ohjattavissa tai hallittavissa mobiilisovelluksilla.

2.2 IoT laite

Yksittäisellä IoT laitteella voidaan tarkoittaa lähes mitä vaan fyysistä laitetta, joka voidaan yhdistää internettiin. IoT laitteeksi voidaan kutsua esimerkiksi hehkulamppua, jota pystytään hallitsemaan mobiilisovelluksella internetin ylitse tai vaikka itsestään ajava autoa. (Ranger, S. 2020.)

Kuvalla 1. havainnollistetaan mahdollista yksinkertaista IoT ekosysteemiä. Kuvassa vasemmalla alhaalla on älypuhelin, jolla voidaan hallita tai tarkastella IoT laitetta mobiilisovelluksen kautta. Keskellä alhaalla oleva IoT laite kerää dataa ja lähettää verkon yli keskeiselle palvelimelle eli pilvipalvelulle, joka on kuvassa keskellä ylhäällä. Oikealla alhaalla on tietokone, jolla voidaan hallita tai tarkastella IoT laitetta verkkopalvelusta. Isommat ekosysteemit voivat sisältää useita IoT laitteita tai sensoreita, jotka voivat olla yhteydessä toisiinsa ja suorittaa toimenpiteitä saamallaan tiedoilla. IoT ekosysteemit voivat olla lähes täysin riippumattomia ihmisten interaktiosta eli IoT laitteet voivat keskustella suoraan toistensa kanssa, ja jaetun tiedon perusteella aloittaa toimenpiteitä.



Kuva 1. Yksinkertainen IoT ekosysteemi

Internettiin liitetyjä IoT laitteita on Statistan verkkosivulla ennustettu vuonna 2020 olleen 8.74 miljardia kappaletta ja sen oletetaan kolminkertaistuvan vuoteen 2030 mennessä. Luvuista voidaan todeta, että ihan pienestä ekosysteemistä ei siis ole kyse. (Holst, A. 2021.)

IoT laitteet suunnitellaan yleensä toimittamaan vain yhtä tiettyä tehtävää, joten yksittäinen laite voi olla hyvinkin yksinkertainen ja edullinen. Koska kustannustehokkuus on kaupallisissa tuotteissa merkittävässä roolissa, on IoT -laitteet usein hyvin rajallisia. Tämä jättää yleensä vähän tilaa vankoilta suojausmekanismeille sekä tietosuojalle. (Trend Micro, 2020.)

IoT gateway

IoT gateway on fyysinen IoT -laite tai sovellus, joka liittyy pilvipalvelun kontrollereihin, sensoreihin tai muihin laitteisiin. Kaikki pilvestä kentälle tuleva ja kentältä pilveen menevä data kulkee gateway:n kautta. Gateway:tä käytetään myös prosessoimaan sensoreilta saatavaa raakadataa, jolloin internetin ylitse kulkeva data olisi valmiiksi suodatettua. Raakadatalta taas tarkoitetaan antureilta saatavaa tietoa, jota ei ole vielä käsitelty ollenkaan. Raakadatan suodattaminen tai käsittely vähentää huomattavasti verkkoliikennettä laitteiden välillä, ja nopeuttaa laitteiden välisten viestien vasteaikaa, sekä jakaa pilvipalveluelimen kuormitusta. Teollisuudessa gateway laitteet ovat keskeisessä roolissa luotaessa teollista internet ekosysteemiä. (Lavery, T. 2017.)

2.3 Teollisuuden Internet

Esineiden internettiä on alettu hyödyntää myös teollisuudessa, ja tästä tulee myös nimitys teollisuuden internet. Teollisuuden internetissä pyritään hyödyntämään antureita, dataa, tekoälyä ja analytiikkaa mittaamaan, automatisoimaan ja optimoimaan teollisuuden prosesseja. (axiomtek.)

Teolliseen internetiin suunnatut laitteet saattavat erota suuresti kuluttajille suunnatuista laitteista. Teollisuuteen suunnatuilta laitteilta vaaditaan yleensä enemmän kestävyyttä, sillä ne suunnitellaan äärimmäisiin olosuhteisiin. Laitteita saatetaan sijoittaa paikkoihin, jossa on erittäin kylmä, kuuma, kostea tai olosuhteet voivat vaihdella rajusti.

Teollisuuden internetin isoimpia käyttökohteita on tuotannon koneiden ja laitteiden ennakoitava ylläpito. IoT laitteilta saadaan reaaliaikaista dataa näiden koneiden tai laitteiden kunnosta ja mahdollisista vioista. Näin ollen koneita voidaan huoltaa tai korjata ennen mahdollista suurempaa ongelmaa tai vikaa. Tätä voidaan myös hyödyntää asiakkaille myytyjen tuotteiden kunnan valvonnassa. (Posey, B. 2020.)

2.4 Tulevaisuus

Esineiden liittäminen internetiin tuo lähes rajattomat mahdollisuudet melkein joka sektorille. Teollisen internetin aikakausi on vasta alussa, ja tiedonsiirtoteknologioiden kehittyminen sekä tekoälyn hyväksikäyttäminen lisäävät käyttökohteita.

Valtavat mahdollisuudet tuovat mukanaan myös suuret uhat. Datan, kuten henkilökohtaisten tietojen tallentaminen tai liikuttaminen internetissä altistaa niiden pääsyn väärin käsiin. Tämän datan vuotaminen saattaa aiheuttaa tuhoisia seurauksia. IoT laitteiden väärinkäyttö ja niiden hyväksikäyttö erilaisissa verkkohyökkäyksissä on otettava huomioon, kun kehitetään IoT laitteita tai kokonaisuuksia. (Symanovich, S, 2019.)

3 DevOps

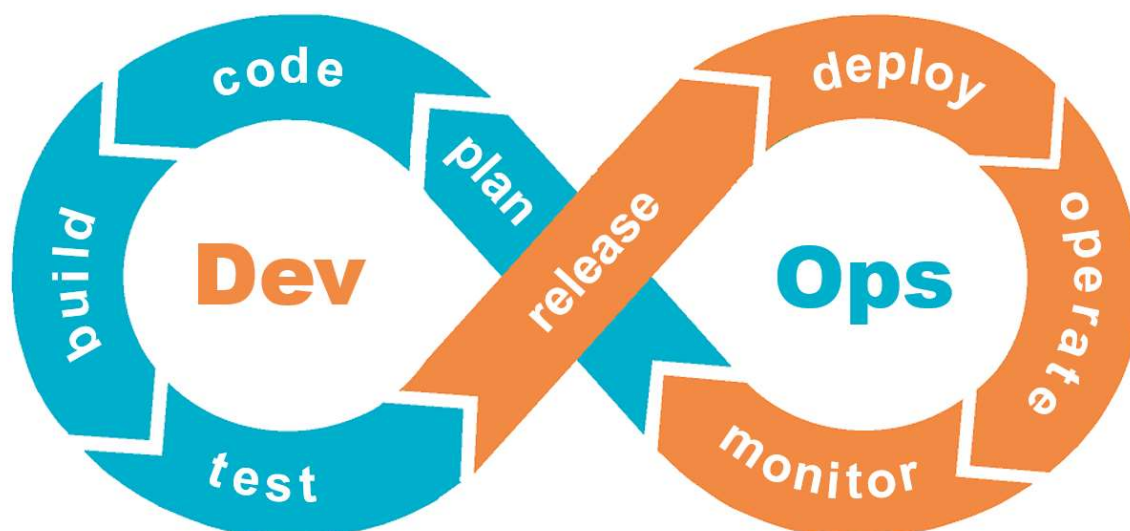
3.1 Yleistä

Termi DevOps tulee englannin kielen sanoista "development" ja "operations", jotka voidaan vapaasti suomentaa ohjelmistokehitykseksi ja palveluntarjonnaksi. Näitä kahta on pidetty pitkään erillisinä "siiloina", joiden tavoitteet ovat keskenään ristiriidassa. Palveluntarjonnassa arvostetaan vakautta ja toiminnallisuuden varmuutta, mikä voi hidastaa ohjelmistojen julkaisua tai toimitusta. Ohjelmistokehitys sektorilla arvostetaan nopeutta ja innovatiivisuutta, jossa pyritään kehittämään koko ajan uusia ominaisuuksia. DevOps on siis toimintamalli, jolla pyritään yhdistämään nämä kaksi erillistä toimintoa yhdeksi saumattomaksi kokonaisuudeksi, sekä automatisoimaan tämä toimintaketju. (Mazyar, M. 2019 & Abildskov, J. 2020.)

DevOpsilla saavutetaan parempi vasteaika ohjelmiston tuottamisesta tuotantoon laadusta tinkimättä. Kuva 2. esittää koko DevOps silmukan työnkulkua ja sen yksittäisiä työvaiheita, joita ovat:

- Suunnittelu (plan), mikä sisältää ohjelmiston jatkuvaa suunnittelua.
- Koodaaminen (code), eli lähdekoodin kirjoittamista ja sen integroimista.
- Kokoaminen (build), missä lähdekoodista kootaan toimiva sovellus.
- Testaaminen (test), missä kootun sovelluksen toiminnallisuutta testataan automatisoiduilla testeillä.
- Julkaiseminen (release), missä testit läpäissyt ohjelmisto julkaistaan.
- Käyttöönotto (deploy), eli versioitu sovellus viedään tuotantoon.
- Käyttö (operate), eli uutta sovellusta käytetään tuotantoympäristössä.
- Tarkkailu (monitor), tuotantoon vietyä versiota tarkkaillaan mahdollisten vikojen takia.

Kuvan silmukkamainen rakenne havainnollistaa ohjelmistokehityksen jatkuvaa kiertokulkua.



Kuva 2. DevOps silmukka ja sen yksittäiset vaiheet

Toimintamalliin linkitetään yleensä kolme periaatetta, joiden avulla DevOps silmukkaa toteutetaan. Nämä kolme tapaa ovat: ketterä kehitys, jatkuva integraatio, sekä jatkuva julkaisu tai toimitus. Tässä luvussa käydään läpi mitä edellä mainitut periaatteet pitävät sisällään.

3.2 Ketterä kehitys

Ketterä kehittäminen on periaate, jota hyödynnetään DevOps kokonaisuudessa. Termi käytetään ohjelmistokehityksessä ja se tarkoittaa sitä, ettei ohjelmistoa yritetä saada kerralla valmiiksi, vaan sitä voidaan kehittää sen koko elinkaaren ajan. (Järvenpää, J. 2020.)

Peruseriaatteena ketterässä kehityksessä on asiakkaiden tarpeiden tyydyttämistä julkaisemalla uusia ohjelmistoversioita säännöllisin väliajoin. Ominaisuuksia lisätään ohjelmistoon vähitellen ja tiedossa olevia vikoja voidaan korjata samalla. Vanhan vesiputousmallin korvaajaksi tullutta ketterää kehitystä voidaan hallinnoida esimerkiksi Scrum tai Kanban toimintamalleilla. (Järvenpää, J. 2020.)

3.3 CI/CD -putkisto

DevOpsia automatisoidaan niin sanotulla CI/CD -putkistolla (eng. pipeline), jolla tarkoitetaan toimintasarjaa, mikä suoritetaan julkaistavalle ohjelmistolle. Nimi koostuu kahdesta osasta: jatkuva integraatio (CI eng. continuous integration) ja jatkuva julkaisu tai toimitus (CD eng. continuous deployment tai continuous delivery) ja näiden prosessien läpiviemistä kuvataan "putkistolla". (Anastasov, M. 2019.)

Putkiso koostuu useammasta pienemmästä putken palasta, jotka kuvaavat putkiston yksittäisiä vaiheita. Osa putken paloista kuuluu jatkuvaan integraatioon ja osa jatkuvaan julkaisuun tai toimitukseen. Tämän toimintosarjan suorittamiseksi on kehitetty useita työkaluja, joista tunnetuimpia ovat: Jenkins, CircleCI, TeamCity ja Bamboo.

3.3.1 Jatkuva integraatio

Jatkuvalla integraatiolla tarkoitetaan ohjelmiston kehittämistä ja sen liittämistä osaksi vanhaan olemassa olevaan ohjelmistoon. Uusia liitoksia voidaan liittää olemassa olevaan ohjelmistoon useita kertoja päivässä.

Jatkuvan integraation tavoitteena on luoda yhtenäinen ja automatisoitu tapa rakentaa, kasata ja testata ohjelmistoja. Integraatioiden liittäminen vanhaan ohjelmistoon on nopeaa ja vaivatonta. Jatkuvan integraation kulmakiviä ovat versionhallinta sekä automatisoidut testit. (Sacolick, I. 2020.) Kuvassa 3. on esimerkki CI/CD -putkiston toteutuksesta ja sen vaiheista. Kuvasta voidaan havaita putkiston työnkulkua ja rakennetta. Putkisto koostuu useasta pienemmästä putken palasta, jotka ovat toisistaan riippumattomia toimintovaiheita. Ideana putkistossa on, että seuraavaan putken vaiheeseen mennään vasta, kun edellinen kohta on läpäisty. Jos jokin putken vaiheista epäonnistuu, aloitetaan putkisto alusta. Kuvan putkiston kolme ensimmäistä palasta ovat jatkuvaa integraatiota.

Versionhallinta

Ohjelmistojen versionhallinnalla tarkoitetaan palvelua, johon voidaan tallentaa lähdekoodia. Ohjelmistot versioidaan yksilöllisin numeroin ja palvelu pitää tallessa myös vanhat versiot, jotta niihin voidaan palata tarpeen vaatiessa. Lähdekoodin jakaminen on palvelun avulla helppoa ja se mahdollistaa ohjelmiston yhtäaikaisen kehittämisen. Versionhallintaan on saatavilla useita työkaluja. Näistä tunnetuimpia ovat Git, CVS ja SVN. (Helsingin Yliopisto.)

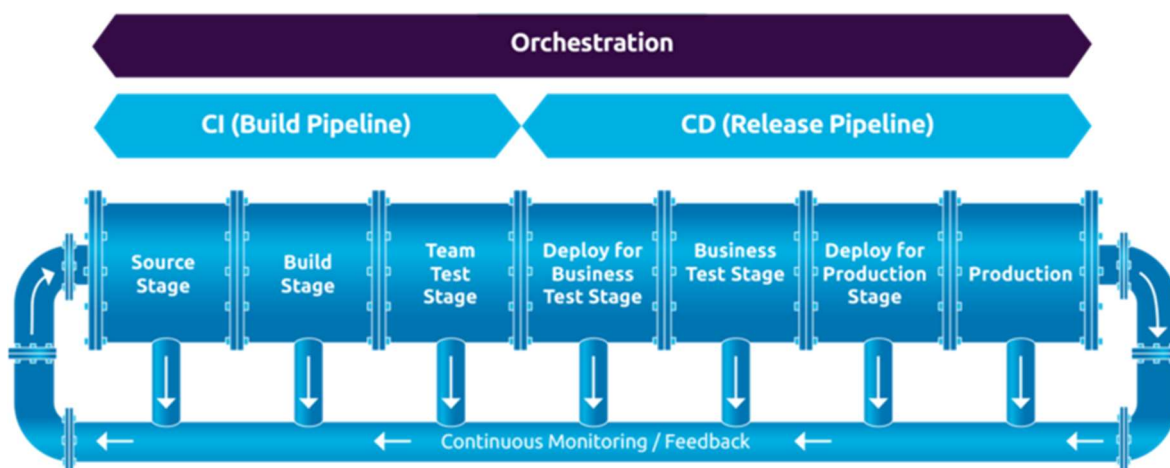
Automaattitestit

Vaikka ohjelmistoja ei pyritä saamaan kerralla valmiiksi ja uusien versioiden integrointi on suhteellisen helppoa, pyritään ohjelmista tekemään silti toimintavarmoja. Tämän takia on tärkeätä testata ohjelmistot ennen niiden julkaisua. Jatkuvan integraation kannalta testien automatisointi nopeuttaa prosessia huomattavasti, sekä tekee siitä turvallisempaa. Automaattiset testit testaavat ja vertaavat sovelluksen antamia tuloksia oletettuihin tuloksiin. Automaattiset testit voivat testata esimerkiksi ohjelmiston yksittäisiä funktioita, ohjelmistojen komponenttien keskinäistä toimivuutta tai ohjelmiston yleistä toiminnallisuutta. (Vertics.)

Testien automatisointiin on myös tarjolla lukuisa joukko ohjelmistokehyksiä (eng. framework). Näistä suosituimpia: Robot Framework, WebDriverIO ja Citrus. (Murugan, B. 2020.)

3.3.2 Jatkuva julkaisu ja toimitus

Jatkuvaa integraatiota seuraa yleensä jatkuva julkaisu tai jatkuva toimitus. Jatkuvan julkaisun ja toimituksen ero on, että jatkuvassa julkaisussa sovellusta ei viedä tuotantoon asti automaattisesti, vaan se vaatii ihmisen hyväksynnän. Jatkuvassa toimituksessa tämä askel on automatisoitu ja siinä ei tarvita ihmisen hyväksyntää. Alla oleva kuva 3 havainnollistaa CI/CD -putkiston toiminnan. Putkiston viimeiset neljä osaa kuuluu jatkuvaan julkaisuun tai toimitukseen.



Kuva 3. CI/CD -putkisto

4 Pilvipalvelut

Pilvipalvelu tai verkkopalvelu on melko uusi käsite ja yleistynyt viimeisen kymmenen vuoden aikana huomattavasti. Pilvipalvelulla tarkoitetaan ohjelmistopalveluita ja tietokoneresursseja, jotka ovat käytettävissä internetin välityksellä. (Laaksonen, A. 2015)

Puhuttaessa pilvipalvelusta tarkoittaa se sitä, ettei palvelu ole omalla tietokoneella tai edes yrityksen palvelimella. Pilvipalvelua isännöidään palveluntarjoajan palvelimella ja se on asiakkaan käytössä yleensä internetin välityksellä. Hyvin tyypillinen pilvipalvelun esimerkki on sähköposti, jossa viestit tallennetaan palvelun tarjoajan palvelimille. Pilvipalvelun tiedot ovat siis tallessa, vaikka omat laitteet hukkuvat tai rikkoutuvat. Pilvipalvelumallin avulla tiedot ovat helposti jaettavissa ja niitä voi käyttää kaikki, joille käyttöoikeuksia on jaettu. (Kangasniemi, H & Lintulahti, M 2017.)

Pilvipalveluiden jakelumallit voidaan jakaa ympäristön saatavuuden mukaan. Jakelumalleja ovat: julkinen pilvi, yhteisöpilvi, yksityinen pilvi ja hybridi. Julkisella pilvellä tarkoitetaan palvelua, joka on useiden käyttäjien jakama ympäristö, jota voi käyttää kuka vaan. Yhteisöpilvellä käyttö voidaan rajata tietyn yhteisön sisälle. Yksityisellä pilvellä tarkoitetaan yhdelle asiakasorganisaatiolle rajattua ympäristöä, joka voidaan pystyttää organisaation omalle palvelimelle. Hybridipalvelut yhdistävät asiakkaan omaa infraa tai omassa infrassa pyöriviä ohjelmistoja julkisen pilven palveluihin. (Laaksonen, A. 2015)

Pilvipalvelumallit voidaan luokitella kolmeen pääluokkaan: IaaS (Infrastructure as a Service), PaaS (Platform as a Service) ja SaaS (Software as a Service). Kuvassa 4. on esimerkkejä mahdollisista pilvipalveluista luokittain. Kuvaan on havainnollistettu esimerkkilaitteita pilven ulkopuolelle, joilla pilvipalveluita voidaan käyttää.

IaaS -malli

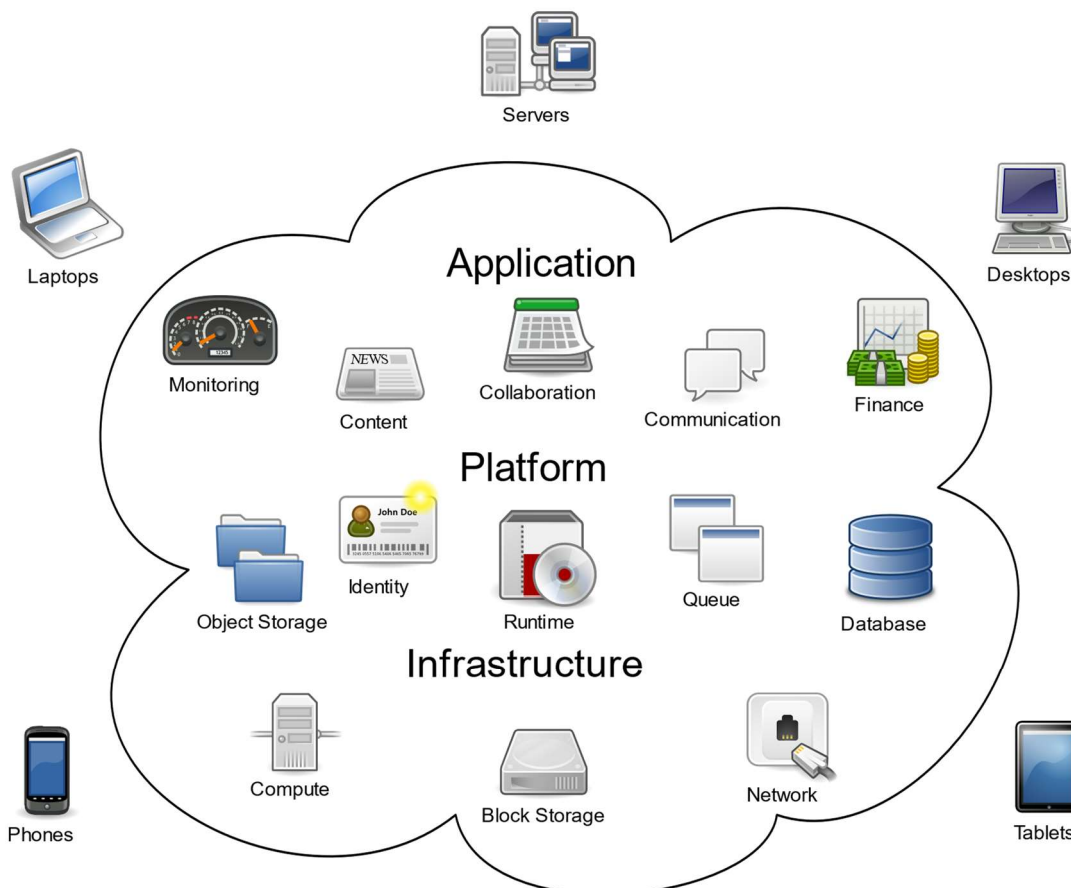
IaaS mallissa palveluntarjoaja tarjoaa infrastruktuuria palveluna eli käytännössä virtuaalipalvelimia ja tallennustilaa verkon välityksellä. Tässä mallissa palveluntarjoajan vastuulla ovat vain alustat ja käyttökapasiteetin tarjoaminen. Malli vaatii käyttäjältään paljon osaamista, ja usein asiakkaana on organisaatio, jolla on oma IT-osasto. (Eronen, H. 2017.) IaaS -malli on kuvassa 4 pilven alin kerros.

PaaS -malli

PaaS mallissa palveluntarjoaja tarjoaa sovellusalustaa palveluna, jotka on yleensä suunnattu ohjelmistokehittäjille. Palveluita voidaan liittää osaksi asiakkaan omaa ohjelmistoa esimerkiksi API-rajapinnan kautta. (Eronen, H. 2017.) PaaS -malli on kuvassa pilven keskellä.

SaaS -malli

SaaS taas tarkoittaa valmiin ohjelmiston tarjoamista palveluna verkossa. Tämä on näistä kolmesta yleisin palvelumalli. Mallissa palvelun tarjoaja huolehtii kokonaisvaltaisesti ohjelmistostaan ja loppukäyttäjä voi olla esimerkiksi yksityinen henkilö. Esimerkkinä SaaS palvelusta on sähköposti tai vastaava verkkosovellus. (Eronen, H. 2017.) SaaS -malli on kuvassa pilven ylin kerros.



Kuva 4. Pilvipalvelumalli kaavio (Johnston, S 2009)

5 Ohjelmistojen päivittäminen

Nykyisin suurimpaan osaan ohjelmistoista on tarjolla uusia päivityksiä säännöllisin väliajoin. Ohjelmistopäivityksille voi olla monia eri syitä, mutta usein niihin liittyy tunnistetut ohjelmistovirheet eli bugit, suojautumisen ei toivotuilta hyökkäyksiltä tai lisäominaisuuksien liittäminen vanhaan ohjelmistoon. Kaikella tällä pyritään hyödyttämään loppukäyttäjää ja siksi on tärkeää, että ohjelmistoja päivitetään säännöllisin väliajoin. (Temboo, 2018.)

OTA (over-the-air) päivityksillä tarkoitetaan laitteiden päivittämistä langattomasti internetin välityksellä. OTA päivitykset helpottavat hajautettujen verkkoon kytkettyjen laitteiden päivittämistä ja mahdollistavat massapäivitykset. Hyvänä esimerkkinä OTA päivityksistä voidaan pitää älypuhelimien käyttöjärjestelmän tai siihen asennettujen sovellusten päivittämistä. Päivityksiä ladataan sovelluksiin jatkuvasti ilman langallista yhteyttä. (Mixon, E. 2019.)

IoT ekosysteemissä perinteinen manuaalinen ohjelmistojen jakelu toimii, kun laitteita on vielä kohtuullisen vähän. Tällöin ei ole välttämättä kannattavaa lähteä kehittämään OTA järjestelmiä. Kun ekosysteemissä olevien laitteiden määrä kasvaa, on OTA lähes välttämätön osa laitteiden kestäväen kehityksen kannalta.

OTA järjestelmän arkkitehtuuri perustuu kahteen erilliseen kokonaisuuteen. Ensimmäiseksi asiakaslaite on itse vastuussa laitteen omien ohjelmistoversioiden ylläpitämisestä ja päivittämisestä. Toisena palvelinohjelmisto on vastuussa siitä, että se pystyy jakamaan asiakaslaitteiden ohjelmistoversioita. (Temboo, 2018.)

6 MVP malli

Tämän opinnäytetyön tavoitteena on toteuttaa MVP mallin mukainen järjestelmä ohjelmistopäivitys järjestelmästä. MVP on lyhenne, joka tulee englannin kielen sanoista Minimum Viable Product, ja suomennetaan pienin julkaisukelpoinen tuote. Vaikka nimi viittaa tuotteeseen, ei se aina pidä paikkansa. MVP on enemmänkin konsepti, jolla yritykset pyrkivät selvittämään mahdollisimman nopeasti uuden tuotteen markkinakelpoisuuden tai tässä tapauksessa käyttökelpoisuuden. Tässä ajatusmallissa olennaista on luoda tuote, joka voidaan myydä, mutta ei vastaa toimittajan visiota lopullisesta tuotteesta. Näistä tuotteista pyritään keräämään palautetta, ja jatkokehittämään tuotetta lähemmäs lopullista visiota. (Pendolin, H. 2018.)

Mallin tarkoituksena ei siis ole saada loppuun hiottua valmista tuotetta käyttöön, vaan ketterän kehityksen ajatusmallin mukainen tuote, joka jättää varaa vielä jatkokehitykselle. Ohjelmistokehitysympäristössä mallin mukaisella tuotteella saadaan järjestelmän hyödyt aikaisessa vaiheessa käyttöön, ja sitä voidaan jatkokehittää myöhemmissä vaiheissa vastaamaan yrityksen tai asiakkaiden tarpeita. Mallin avulla tunnistetaan tuotteen kannattavuus, ja asiakas näkee vastaako tuote heidän tarpeitaan melko aikaisessa vaiheessa. Tuotetta on helppo lähteä viemään uuteen suuntaan ja projektin edetessä voidaan kehittää uusia ominaisuuksia. Ideana on optimoida kehitystyötä karsimalla yli- ja alikehittäminen. MVP malli soveltuu hyvin uusien innovaatioiden kehittämiseen, jossa pyritään myymään tuote loppuasiakkaalle tai sijoittajille. (Hurja, 2018.)

7 Toimintaympäristö

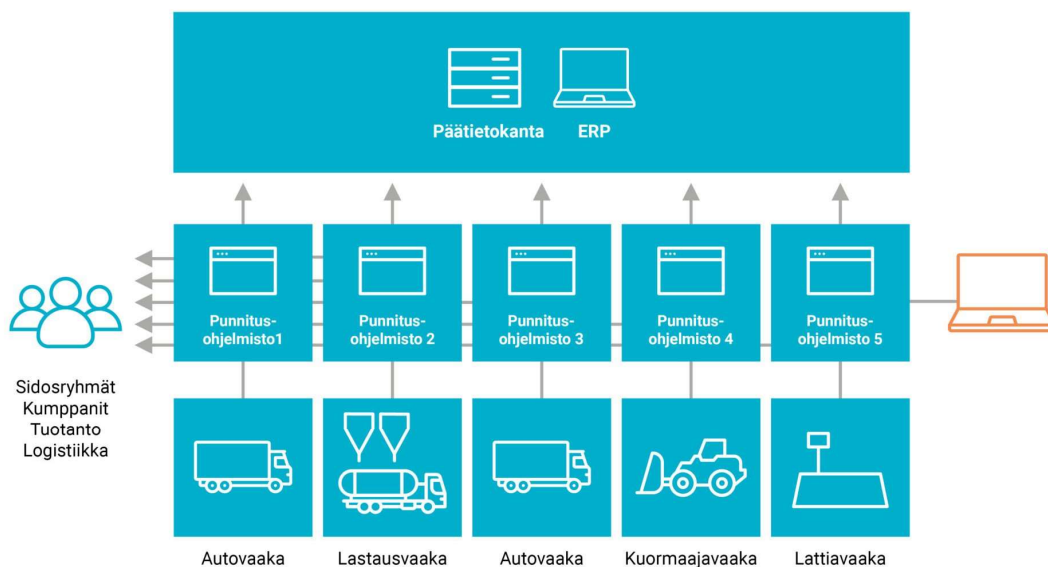
7.1 mScales punnituspalvelu

mScales on Lahti Precision Oy:n kehittämä digitaalinen punnituspalvelu, jolla pystytään liittämään vaa'at ja liiketoimintajärjestelmät yhdeksi saumattomaksi kokonaisuudeksi. Punnituspalvelun lähtökohta oli liittää ajoneuvovaa'at järjestelmään, mutta palvelu on laajentunut myös muihin vaakoihin. Punnituspalvelu automatisoi punnitusprosessin ja punnituksia hallitaan reaaliaikaisesti. Kuva 5 esittää perinteistä punnitusjärjestelmää, joka sisältää useita toisistaan erillään olevia laitekokonaisuuksia, jota on vaikea hallita, ja näiden ylläpitäminen on kallista. Kuvassa 6 nähdään digitaalisen punnituspalvelun malli.

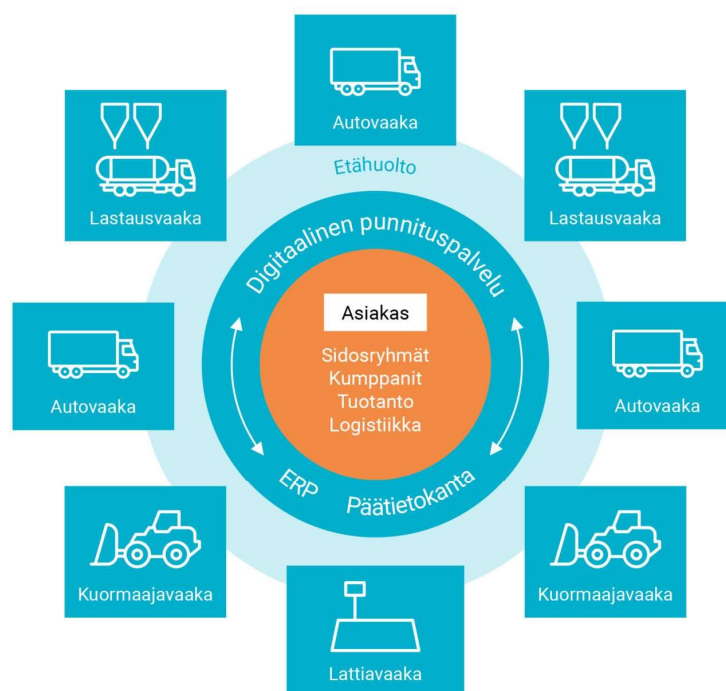
Punnituspalvelun avulla voidaan tehostaa liiketoimintaa automatisoimalla punnitusprosessin eri vaiheita. Punnituksia voidaan operoida helposti, vaikka älypuhelimella ajoneuvon hytistä, eikä ylimääräisiä punnitusoperaattoreita tarvita.

Punnituksia voidaan hallita, seurata ja kontrolloida verkkopalvelussa, jossa ne ovat näkyvissä vain niillä sidosryhmillä, joille se on tarkoitettu. Lisäksi mScales voidaan liittää osaksi asiakkaan liiketoimintajärjestelmää asiaksrajapinnan kautta.

Punnitustapahtumat ovat tehokkaampia ja kustannuksiltaan selvästi perinteistä tapaa edullisempia. Palvelu tuo toimintavarmuutta punnitusprosessiin ja tekee siitä ketterämpää. Palvelun käyttö on myös turvallista, sillä se käyttää aina internetin ylitse tapahtuvassa viestinnässä salattua viestintäyhteyttä.



Kuva 5. Perinteinen punnitusjärjestelmä (Lahti Precision Oy 2020c)



Kuva 6. Digitaalinen punnituspalvelu (Lahti Precision Oy 2020c)

Punnituspalvelu perustuu kolmesta elementistä: verkkopalvelusta, kentälaitteista ja niitä yhdistävästä gateway-laitteesta. Pelkkään vaa'an ja verkkopalvelun viestintään ei välttämättä tarvita gateway-laitetta, mutta tällöin vaa'an tulee käyttää Lahti Precisionin

viestintästandardia. Edellä mainittujen kolmen elementin toimintaperiaatteita käydään alla läpi.

7.2 mScales verkkopalvelu

Verkkopalvelu tallentaa ja prosessoi punnituksia reaaliaikaisesti ja sillä voidaan hallita tilauksia ja sopimuksia. Palvelun kautta voidaan luoda punnituskoodoja, joiden avulla kuljetusyritys voi suorittaa punnitusoperointeja.

Kaikista tapahtumista jää jälki järjestelmään, ja niitä voidaan tutkia jälkikäteen, ja niistä voidaan luoda raportteja. Järjestelmä pitää myös kirjaa tapahtumista koituvista varastosaldon muutoksista, jotta varastojen kirjanpito pysyy ajantasalla.

Verkkopalvelu koostuu pääosin kahdesta käyttäjärajapinnasta. Toinen on verkkosovellus, jolla asiakas voi hallita ja tarkastella punnitustapahtumiaan. Palvelu vaatii rekisteröitymisen, millä varmennetaan myös, että oikeat tiedot näkyvät vain oikeille ihmisille.

Toinen on kuljetusyritykselle luotu mobiilikäyttöliittymä, josta voidaan suorittaa yksittäisiä punnitusoperointeja. Tämä ei vaadi rekisteröitymistä, mutta se vaatii punnituskoodin, jonka kuljetusyrityksen tilaaja on antanut punnitusoperointia varten.

Verkkopalvelu voidaan liittää asiakkaan toiminnanohjausjärjestelmiin asiakasrajapinnan kautta. Asiakasrajapinnasta käsin voidaan hakea, muokata ja lisätä lähes tulkoon kaikkea, mitä verkkokäyttöliittymästäkin voidaan operoida.

7.3 Vaa'at ja oheislaitteet

Punnitusprosessin ydin on itse punnitseminen, josta Lahti Precisionilla on pitkä ja ammattitaitoinen kokemus. Yritys on toimittanut vaakoja jo yli sata vuotta. Vaikka yritys toimittaa omia vaakoja, ei mScales-palvelu edellytä käyttämään heidän toimittamiaan vaakoja, vaan lähtökohtana on se, että palvelu voidaan liittää mihin tahansa vaakaan.

mScales palveluun voidaan liittää muitakin oheislaitteita kuten portteja, liikennevaloja tai kameroita ja niitä voidaan ohjata kätevästi verkkopalvelusta käsin. Kuvassa 7 on Lahti Precisionin ajoneuvovaaka, joka on upotettu maahan. Kuvan esimerkki vaa'an voi liittää mScales punnituspalveluun.



Kuva 7. Lahti Precisionin upotettu autovaaka (Lahti Precision Oy 2020b)

7.4 IoT gatewayt

Jotta kentältä saatava data olisi reaaliaikaisesti kaikkien sidosryhmien käytettävissä, viestivät kenttälaitteet verkkopalvelun kanssa internetin ylitse. Osa vaaosta sekä kaikki oheislaitteet vaativat viestintään tai ohjeukseen IoT gateway laitteen, joka välittää käskyjä verkkopalvelulle.

Gateway -laitteet tarvitsevat internet-yhteyden, jotta yhteys mScales-verkkopalveluun voidaan toteuttaa. Gateway sisältää jossain tapauksissa myös varakäyttömahdollisuuden, jos internetyhteys on heikko tai se katkeaa.

7.5 Nykytila

Yrityksen tuotekehitys on asiakaslähtöistä, joten asiakkaan toivomia ominaisuuksia pyritään lisäämään ja täydentämään nykyistä ohjelmistoa mahdollisuuksien mukaan. Kehitystyö on yrityksessä intensiivistä ja uusia päivityksiä ja parannuksia kehitetään jatkuvasti usean ohjelmistokehittäjän toimesta. Koska paranneltu sovellus halutaan myös asiakkaan käyttöön mahdollisimman nopeasti, tehdään ohjelmistopäivityksiä säännöllisesti muutaman viikon välein.

Nykytilanne gateway laitteiden päivityksessä on pitänyt sisällään paljon manuaalista työtä ja se on sitonut yhden henkilön resursseja tähän työhön, joten sitä on haluttu lähteä kehittämään automatisoidumpaan suuntaan. Toimeksiantaja haluaa kehittää laitteiden hallintaa siten, että gateway -laitteiden ohjelmistoversioita voidaan päivittää automatisoidusti ja keskitetysti heidän verkkopalvelustaan.

Tällä hetkellä laitteet on päivitetty yksi kerrallaan. Päivityksessä ohjelmisto on täytynyt koota, testata, sekä pakata toimitusta varten. Kohdelaitteella ajetaan valmiiksi tehty scripti eli lyhyt ohjelma, joka tekee ohjelmiston vaihdoksen, mikä kestää muutamia sekunteja. Kokonaisuudessaan yhden laitteen päivittäminen saattaa kestää muutamia minuutteja, kun lasketaan mukaan yhteyden ottaminen kohdelaitteeseen, uuden ohjelmiston lataaminen ja paikallinen asennus. Kun laitteita on useita, vie tämä prosessi paljon aikaa ja inhimilliset riskit tekevät tästä epäluotettavaa. Esimerkiksi yksittäinen laite saattaa vahingossa jäädä päivittämättä, mikä on suuri riski.

Työn tavoitteena on siis automatisoida edellä kuvastettua prosessia niin, että manuaalinen työ jäisi mahdollisimman pieneksi. Työ toteutetaan MVP mallin mukaisesti, joten työ ei kata kaikkea automatisointia ja lisäkehityskohteita jää myöhemmin kehitettäväksi. Tavoitteena on rakentaa hyvä alusta, joka kattaa minimalistisesti koko prosessin, ja sen pohjalta on hyvä lähteä tekemään jatkokehitystä.

Työllä pyritään saavuttamaan kustannustehokkuutta päivitysprosessiin vähentämällä ihmistyötuonteja ja priorisoimaan ohjelmistokehittäjien resurssit kehitystyöhön. Järjestelmän automatisointi toisi toimintavarmuutta prosessiin, ja saavutettaisiin nopeampi vasteaika ohjelmistojen jakamisella tuotantoon.

8 Toteutus

8.1 Tiedon keruu ja suunnittelu

Toimeksiantajalla oli muutamia vaatimuksia järjestelmän arkkitehtuuriin, kuten verkkopalvelun ja IoT laitteiden viestiketjuun liittyen ja tiettyjä yksittäisiä protokollia, mitä tuli käyttää toteutuksessa, muutoin voitiin suunnitella ja toteuttaa järjestelmää vapaasti. Yrityksellä oli myös jo ennestään käytössä tiettyjä työkaluja tai alustoja, joita pystyi ja kannatti käyttää hyväksi tässä toteutuksessa.

Työn toteutus alkoi aiheeseen tutustumisella. Selvitettäviä asioita oli muun muassa, miten vastaavia järjestelmiä oli rakennettu, mitä valmiita alustoja oli tarjolla ja mitä nämä alustat tarjosivat. Vastaavien palveluiden tuote-esitteet, ohjelmistokehitykseen suunnatut foorumit, sekä artikkelit aiheeseen liittyen antoivat kattavan tiedon järjestelmän rakenteesta. Näistä lähteistä kootun tiedon perusteella saatiin käsitys, mitä järjestelmän kehittäminen vaatisi. Lisäksi täytyi kartoittaa nykytilaa toimeksiantajan olemassa olevista järjestelmistä ja mitä niihin täytyi kehittää, jotta haluttu lopputulos saavutettaisiin.

Toimeksiantajalla oli aiemmin kehitetty vastaavaa järjestelmää, mitä ei koskaan oltu otettu käyttöön. Tämä järjestelmä ei myöskään toiminut aivan sellaisenaan. Peruselementit tästä kyllä löytyi, kuten palveluiden rajapinnat ja minimaalinen arkkitehtuurinen toteutus järjestelmästä. Vuosien ohjelmistokehityksen muutokset tekivät tästä kuitenkin käyttökelvottoman nykyiseen ympäristöön, joten sitä täytyi lähteä kehittämään vastaamaan nykyistä ympäristöä, sekä kehittämään haluttuja uusia ominaisuuksia. Tämä oli kuitenkin hyvä pohja, jonka päälle oli helppo lähteä kehittämään uutta järjestelmää. Lisäksi yrityksen verkkopalvelun toimintaympäristöstä saatiin paljon ideoita.

Hankittujen tietojen pohjalta käytiin yhdessä toimeksiantajan kanssa läpi, miten kokonaisuus toteutettaisiin. Uuden järjestelmän tuli tukea vanhaa arkkitehtuurimallia ja olla yhtenäinen muun järjestelmän kanssa. Ideoinnin lopputuloksena syntyi idea järjestelmästä, jonka rakenne koostui kolmesta erillisestä kokonaisuudesta. Ensimmäinen osa koostui sovellusversioiden kehityksestä, kokoamisesta ja jakelusta, joka toteutettiin CI/CD -putkistoa hyväksikäyttäen. Toinen osa koostui verkkopalvelun hyödyntämisestä laitekohtaisten versiotietojen lisäämisessä, tallentamisessa, hakemisessa, muokkaamisessa ja poistamisessa. Verkkopalvelun tuli myös toimia asiakasohjelmistojen välittäjänä. Kolmas osa koostui IoT gateway -laitteiden päivittäjäohjelmasta, joka tiedustelee uusimpia versiotietoja verkkopalvelulta ja lataa sekä asentaa sille määritetyn asiakasohjelmistoversion automaattisesti.

8.2 Ohjelmistoversioiden kehittäminen ja toimittaminen

Ensimmäinen osa ohjelmistoversioiden automaattisen päivityksen kannalta oli valmiin asiakasohjelman toimittaminen keskeiseen paikkaan, josta se voidaan hakea ja jakaa gateway laitteille. Ennen asiakasohjelmistoversioiden säilömistä ne täytyi kasata valmiiksi ja pakata, jotta uuden version käyttöönotto oli turvallista, nopeaa ja vaivatonta. Tämä osio ei pitänyt sisällään paljoa ohjelmoimista, vaan koostui suurimmaksi osaksi työkalujen käytöstä ja niiden konfiguroinnista. Vaikka osa työkaluista oli jo valmiiksi käytössä, täytyi suurin osa asentaa ja konfiguroida paikalliseen ympäristöön. Tämä siksi, että toteutettavaa järjestelmää voitiin kokeilla ilman huolta siitä, että tuotantoympäristö häiriintyisi.

Koska järjestelmä toteutettiin MVP mallin mukaan, jäi prosessiin vielä jatkokehitykselle tekemistä. Tässä aluvuossa käydään läpi, mitä vaiheita valmiin asiakasohjelman vieminen arkistoon vaatii ja mitä tämän työn aikana kehitettiin.

8.2.1 Versionhallinta

Jotta asiakasohjelman kasaaminen oli mahdollista, täytyi lähdekoodin olla saatavilla. Lähdekoodit tallennettiin versionhallintaan, josta jokainen ohjelmistokehittäjä voi niitä hakea ja päivittää. Versionhallinnan avulla asiakasohjelmistoille määriteltiin uniikit versionumerot, jonka perusteella voitiin yksilöidä eri versioita arkistoon. Automatisoinnin kannalta myös CI/CD -putkisto työkalulla oli oltava pääsy lähdekoodiin.

Ohjelmiston lähdekoodien tallentamisessa oli ennestään käytetty versionhallintaa, joten tämä osuus ei vaatinut kehittämistä. Tästä oli helppo lähteä jatkamaan työn toteutusta eteenpäin.

8.2.2 CI/CD -putkisto

Gatewayn asiakasohjelmiston osalta ei CI/CD -putkistoa ollut otettu käyttöön, mutta yrityksen verkkopalvelun ohjelmistossa oli se jo käytössä, joten työkalun valinta oli helppoa. Työkaluun täytyi määrittellä gatewayn asiakasohjelmalle oma projektihakemisto, jossa projektin konfiguraatioita voi muuttaa. Työkaluun oli saatavilla iso lista lisäosia, jotka olivat välttämättömiä asiakasohjelman kokoamista varten.

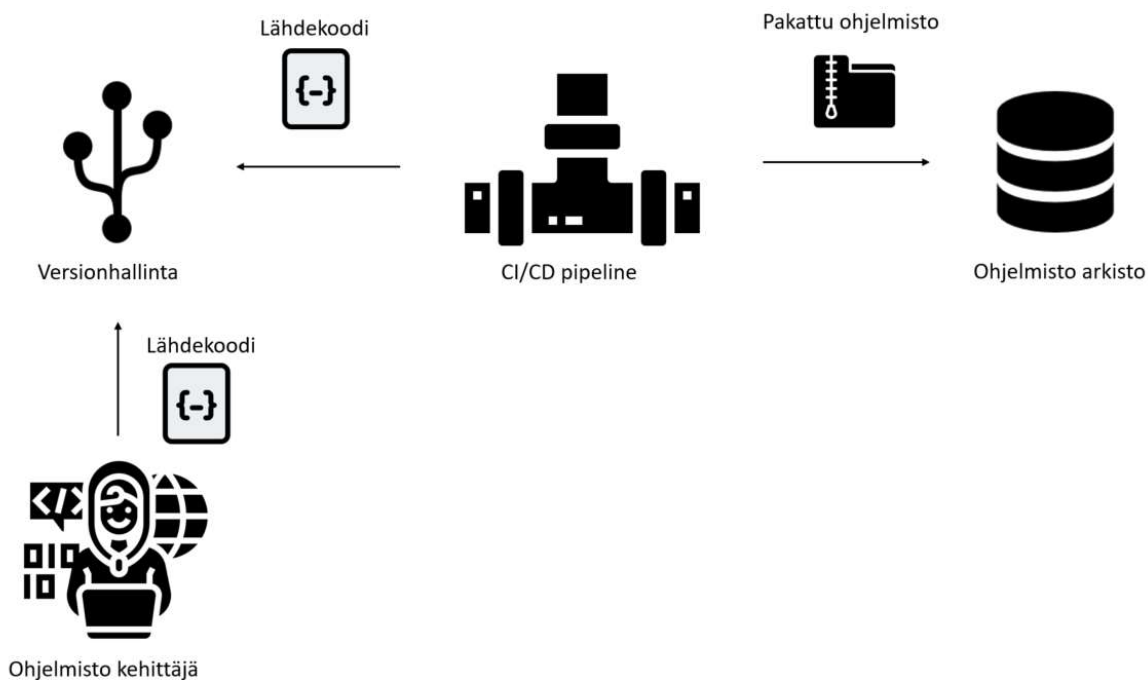
Projektihakemiston luomisen jälkeen työkalun käyttöönotto jatkui tarvittavien resurssien määrittelyillä. Seuraavaksi täytyi määrittellä mistä lähdekoodi haetaan, eli versionhallinnan URL-osoite, josta lähdekoodi voitiin kloonata. Koska lähdekoodiin pääsy oli estetty autentikoinnilla, täytyi CI/CD -putkisto työkalulle luoda valtakirja (eng. credentials), jolla se

autentikoi itsensä versionhallinnalle. Näiden tietojen avulla putkistotyökalu pystyi hakemaan lähdekoodia versionhallinnasta automaattisesti.

Putkistotyökalulle voidaan määritellä ajastin, jonka mukaan se tarkkailee tietyn ajanjakson välein, onko lähdekoodia päivitetty. Jos lähdekoodia oli päivitetty, aloittaa työkalu automaattisesti putkisto prosessin. Koska työ toteutettiin MVP:nä, jätettiin ajastin vielä ottamatta käyttöön, ja prosessin aloitus tapahtuu putkistotyökalun käyttöliittymästä käsin yhden napin painalluksella.

Lähdekoodin hakemisen jälkeiset vaiheet voitiin määritellä toimitettavan asiakasohjelman lähdekoodissa. Ohjelmistopolun juureen kirjoitettiin tiedosto, joka määrittää mitä loppu putkistossa tehdään ja miten. Putkisto työkalu osasi hakea tietyn nimistä tiedostoa ja suorittaa sen osana putkistoa.

Tässä työssä rakennettiin supistettu putkistoprosessi, jota kuva 8 havainnollistaa. Kuvasta käy ilmi prosessin eri toimijat ja nuolilla kuvataan viestiliikenteen suuntaa. Kuvassa keskellä oleva putkistotyökalu haki lähdekoodin versionhallinnasta, johon ohjelmistokehittäjä oli sen päivittänyt. Versionhallinnan URL osoite oli ennalta määritelty työkalun konfiguraatiossa. Kun lähdekoodit oli ladattu, jatkettiin putkistoa lähdekoodista saadun putkistotyökalun tiedoston mukaan. Tiedostoon määritetty prosessi eteni seuraavasti: Ensimmäisenä kasattiin asiakasohjelmisto toimivaksi sovellukseksi, mikä alkoi lataamalla ohjelmiston kannalta välttämättömät kirjastot. Kun kirjastot oli ladattu, voitiin suorittaa tarvittavat ohjelmistokäännökset. Kasaus prosessin jälkeen, valmis asiakasohjelmisto pakattiin tiedostoksi, joka sitten lähetettiin keskeiseen arkistoon, josta se voidaan myöhemmin hakea sille määritellyn versionumeron perusteella. Tämä ohjelmistojen arkistointi työkalu oli verkkopalvelun osalta käytössä, joten täytyi vain luoda uusi kanta gatewayn asiakasohjelmistoille.



Kuva 8. CI/CD -putkisto prosessi

Kaikkea tyypilliseen putkisto prosessiin liittyviä komponentteja ei vielä tässä työssä tehty, koska niiden laajuus olisi riittävän suuri toiselle opinnäytetyölle. Komponenteista esimerkkinä on automaattitestit, jotka on jätetty jatkokehitysvaiheeseen.

8.3 Ohjelmistoversioiden hallinta ja jakaminen

Järjestelmän toinen vaihe liittyi verkkopalvelun ympärille. Asiakasohjelmistoversioiden hallinta tuli tapahtua pilven käyttöliittymästä, joten käyttöliittymän lisäksi verkkopalvelun täytyy pystyä tallentamaan ja prosessoimaan tätä tietoa. Verkkopalvelu osio koostui käyttöliittymästä, kahdesta API rajapinnasta sekä kahdesta prosessista. Toisella prosessilla hallittiin versiotietoja, ja toisella jaettiin näitä asiakasohjelmistoja.

8.3.1 Ohjelmistoversioiden hallinta

Verkkopalveluun oli valmiiksi tehty jo käyttöliittymä laitehallintaan, jossa voidaan lisätä, muokata ja poistaa yksittäisiä laitteita niiden tyyppin mukaan. Käyttöliittymässä voidaan valita listasta yksittäinen laite ja sille voidaan määrittää laitekohtaisia parametreja. Toteutuksessa jokaiselle laitteelle oli luotava kolme parametria, jotka määrittivät laitteen yksilölliset arvot: käytössä olevan asiakasohjelmistoversion, tavoiteversion ja päivitysajankohdan. Nämä parametrit tallennettiin verkkopalvelun tietokantaan halutun laitteen kohdalle. Alla on esimerkkinä kuva 9 miltä yksittäisen laitteen hallintaikkuna voi näyttää. Kuvassa Ylhäällä on laitekohtaisia tietoja, kuten nimi, laitetunnus ja tila. Kuvan keskellä nähdään laitteen

alilaitteet, eli ne laitteet mitä sillä hallinnoidaan, tässä esimerkissä ei ole alilaitteita. Kuvassa alhaalla on laitekohtaiset parametrit, joita on nykyinen asiakasohjelmistoversio, tavoiteversionumero, sekä päivitysajankohta.

Laitteet

IoT-gateway MUOKKAA

Nimi: IoT-gateway
 Toimipiste: RobotLocation2
 Tyyppi: IoT-gateway
 Laitetunnus:
 Lisätiedot:
 Tila: Käytössä
 Luotu: 16.02.2021
 Päivitetty: 05.04.2021

Alilaitteet

NIMI	TOIMIPISTE	LAITETUNNUS	TILA
Ei laitteita			

Parametrit

NIMI	ARVO	LUOTU	PÄIVITETTY	
updateTime	2021-02-26T14:00:20.238Z	18.02.2021	05.04.2021	
installedVersion	1.0.12	18.02.2021	05.04.2021	
targetVersion	1.0.12	18.02.2021	05.04.2021	

Muistiinpanot

Ei muistiinpanoja

SULJE

Kuva 9. Käyttöliittymä

8.3.2 Ohejlmistoversiotietojen jakaminen

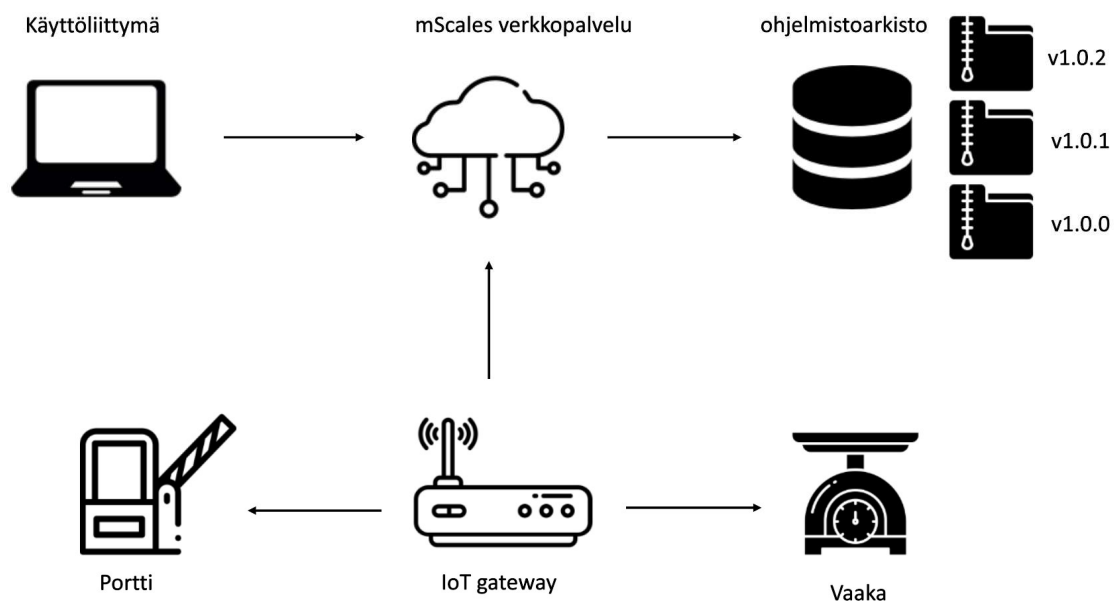
Asiakasohjelmistoversioiden jakelua täytyi pystyä hallitsemaan verkkopalvelun kautta, joten gateway -laitteiden täytyi saada versiotiedot haettua verkkopalvelulta automaattisesti. Myös gateway -laitteen täytyi pystyä ilmoittamaan käytössä oleva asiakasohjelmistoversionsa verkkopalvelulle. Jotta versioiden vaihtelu olisi mutkatonta, molempien osapuolien täytyi olla ajan tasalla laitteen yksilöllisistä parametreista jatkuvasti.

Verkkopalveluun luotiin tätä viestintää varten rajapinta, josta gateway -laitteet voivat hallita omia parametrejaan. Gateway -laitteet pystyivät rajapinnan kautta hakemaan tavoiteltavaa asiakasohjelmistoversiota, sekä päivittämään asennettua asiakasohjelmistoversioita.

8.3.3 Ohjelmistoversioiden jakaminen

Verkkopalveluun luotiin toinenkin rajapinta, josta gateway -laitteet voivat hakea niille suunnattuja asiakasohjelmistoja. Gateway -laitteet lähettävät pyynnön tähän rajapintaan, missä ilmoitetaan haluttu ohjelmistoversio. Verkkopalvelussa tarkistettiin ensin asiakasohjelmiston olemassaolo, ja sen löytyessä arkistosta versio ladattiin ja tarjottiin pyynnön lähettäneelle gatewaylle. Verkkopalvelu toimii tässä käytännössä välittäjänä, koska ei haluttu sallia gateway -laitteiden viestiliikennettä muualle, kuin mScalesin palvelimelle.

Alla oleva kuva 10 esittää mahdollista ekosysteemiä. Ylhäällä vasemalla käyttöliittymästä hallitaan verkkopalveluun määritettyjä laitekohtaisia parametreja. Keskellä olevaan IoT gateway-laitteeseen on liitetty yksi vaaka ja yksi portti, jota voidaan ohjata gatewayn välityksellä. Gateway tiedustelee laitteelle suunnattua ohjelmistoversiota verkkopalvelulta. Verkkopalvelu yrittää ladata asiakasohjelmistoversiot arkistosta tarpeen vaatiessa.



Kuva 10. Ohjelmistoversioiden hallinta ja jakelu

8.4 Ohjelmistoversioiden tiedustelu ja lataaminen

Järjestelmän viimeinen osio oli gatewaylle kehitetty päivittäjäohjelmisto, joka tiedusteli laitteelle suunnattuja versiotietoja, sekä latsi laitteelle suunnatun version itsenäisesti. Latauksen jälkeen uusi asiakasohjelmaversio asennettiin paikallisesti gatewaylle, ja vanha versio vietiin varmuuskopioksi. Tämä oli ehdottomasti projektin työläin vaihe. Päivittäjäohjelmasta täytyi saada toimintavarma siten, ettei asiakkaiden liiketoiminta

häiriinny. Kaikki mahdolliset virhetilanteet pyrittiin huomioimaan ja luomaan näille tilanteille varasuunnitelma. Päivittäjäohjelma irroitettiin omaksi kokonaisuudeksi varsinaisesta asiakasohjelmasta, jotta se pystyi hallitusti päivittämään uudet asiakasohjelmat ja käynnistämään ne itsenäisesti. Tämän seurauksena kehitettyä päivittäjäohjelmaa ei voitu päivittää muun asiakasohjelmiston kanssa, minkä takia päivittäjäohjelman kanssa tuli olla erittäin tarkkana. Tämän varalle kehiteltiin suunnitelma, miten päivittäjäohjelman päivitykset suoritetaan, jos tarve joskus niin vaatii.

Toteutettu päivittäjäohjelma koostui kahdesta prosessista: päivittäjästä sekä tilanvalvojasta. Molemmat prosessit ajastettiin suoritettavaksi lyhyen ajan välein. Alla avataan molempien prosessien toimintaperiaatteita.

8.4.1 Päivittäjä

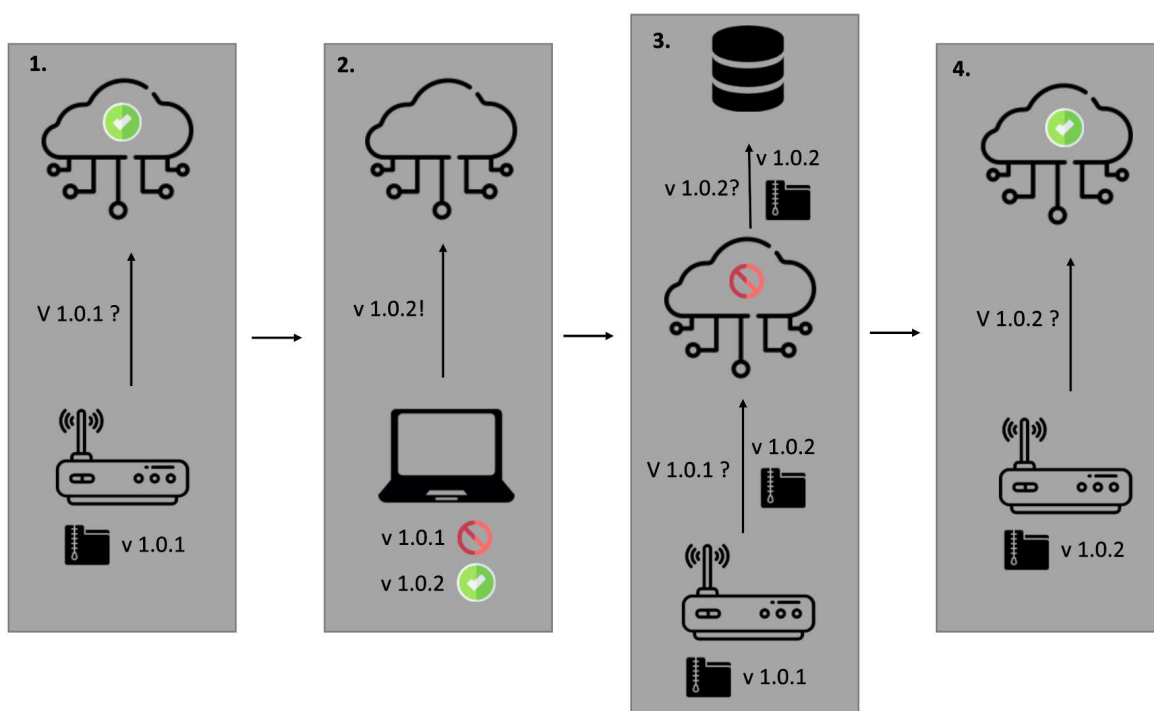
Päivittäjä hakee asennetun versionumeronsa paikallisesta tietokannasta, missä pidetään kirjaa laitekohtaisista parametreista. Jotta voidaan selvittää, onko kyseiselle laitteelle olemassa ohjelmistopäivitystä, lähettää se verkkopalvelun versiotietojen jakelurajapintaan kutsun muutaman minuutin välein. Verkkopalvelu palauttaa sinne määritellyt arvot käytössä olevasta ohjelmistoversiosta, tavoiteohjelmistoversiosta sekä päivitysajankohdasta.

Kun versiotiedot on saatu verkkopalvelulta, tarkastetaan, että verkkopalvelulla on oikea käytössä oleva ohjelmistoversionumero. Jos jostain syystä käytössä olevan ohjelmistoversion lähetys on epäonnistunut ja versionumerot eivät täsmää, koitetaan se päivittää lähettämällä asennettu versiotieto verkkopalvelun rajapinnalle. Kun käytössä olevat ohjelmistoversiot täsmäävät, tarkistetaan, eroaako käytössä oleva versionumero tavoiteversionumerosta. Jos tavoiteversio on muu kuin käytössä oleva versio, aloitetaan päivitysprosessi. Koska prosessi on ajastettu toimimaan kellolla, täytyy sulkea pois mahdollisuus, että asennus tai lataus on vielä käynnissä, ja aloitetaan päällekkäinen latausoperaatio. Sen takia, asetetaan prosessille tila, joka määrittää päivityksen olevan kesken. Jos toinen ajastettu prosessi alkaa latauksen kanssa samanaikaisesti, se tarkistaa onko lataus kesken, ja näin ollessa se lopettaa toimintansa. Näin vain yksi lataus ja asennusprosessi on kerrallaan käynnissä. Päivitysprosessi lähettää kutsun verkkopalvelun versionjakerajapintaan, joka katsoo kyseisen laitteen tavoiteversion ja tarjoilee sen arkistosta, jos versio sieltä löytyy.

Kun uusi ohjelmistoversio on saatu ladattua verkkopalvelulta, ohjelma puretaan pakatusta tiedostosta. Tässä vaiheessa käynnissä oleva ohjelmisto pysäytetään ja aloitetaan ohjelmistojen asennus. Vanha varmuuskopio poistetaan ja nykyinen ohjelmistoversio siirretään uudeksi varmuuskopioksi. Uusi ohjelmistoversio sijoitetaan vanhan tilalle ja

konfiguraatio tiedostot päivitetään uusiin. Ohjelmiston vaihdosta aiheutunut käyttökatkos on noin viiden sekunnin luokkaa.

Alla oleva kuva 11 esittää gatewayn ohjelmistoversion tiedusteluprosessia. Kuvan vasemmassa laidassa, osiossa yksi, gateway tiedustelee laitteelle suunnattua asiakasohjelmistoversiota. Tässä tapauksessa versiotieto on oikea, joten mitään ei tehdä. Osiossa kaksi käyttöliitymästä vaihdetaan versio v1.0.1:stä v1.0.2:teen, joka tallentuu verkkopalvelun tietokantaan. Osiossa kolme gateway kysyelee taas asiakasohjelmistoversiotaan, mikä on nyt muuttunut. Verkkopalvelu hakee oikean ohjelmistoversion arkistosta ja tarjoilee sen gatewaylle. Gateway asentaa uuden v1.0.2 ohjelmistoversion ja asettaa v1.0.1 varmuuskopioksi. Gateway lähettää tiedon verkkopalvelulle, että asennus onnistui ja uusi v1.0.2 on nyt käytössä. Osiossa neljä gateway jatkaa ohjelmistoversionsa kyselyä, joka ei ole muuttunut.



Kuva 11. Gatewayn asiakasohjelmistoversion tiedustelu prosessi

8.4.2 Tilanvalvoja

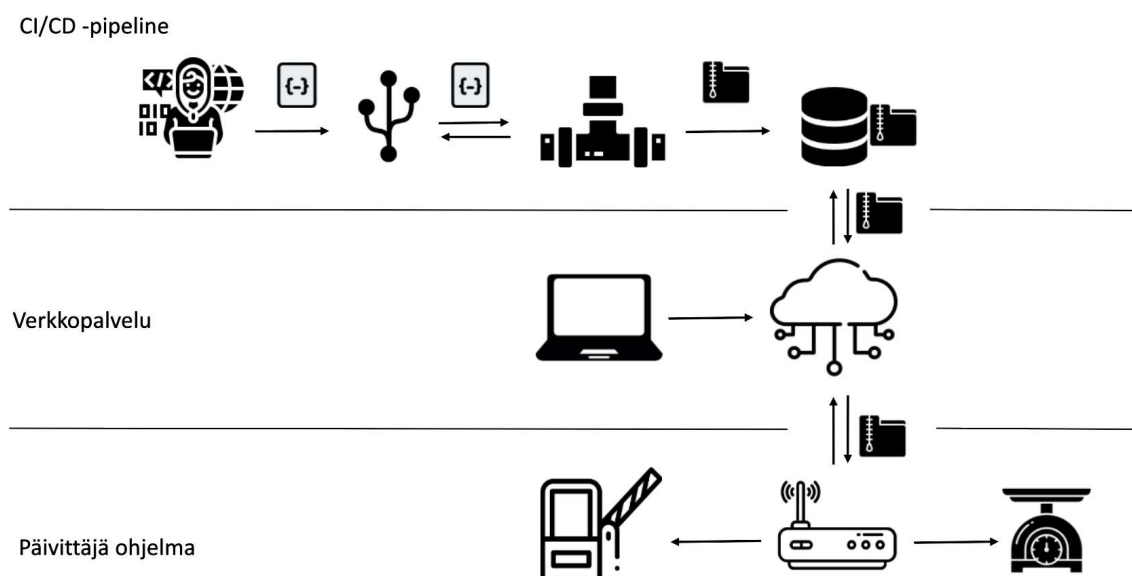
Gatewayn toimintavarmuuden säilyttämiseksi luotiin päivittäjäohjelmaan asiaksohjelmiston tilanseurantaprosessi, joka seuraa käynnissä olevaa asiakasohjelmistoa, ja mahdollistaa niin sanotun "rollback" toiminnon. Rollback toiminnon ideana on säilyttää varmuuskopio toimivasta asiakasohjelmistoversiosta ja ottaa se käyttöön mahdollisessa virhetilanteissa. Jos jakeluun on päässyt viallinen asiaksohjelmistoversio, voidaan palauttaa edellinen toimiva ohjelmistoversio vain hetken käyttökatkoksella, eikä asiakkaan toiminta lakkaa.

Tästä vaihdoksesta lähetetään ilmoitus verkkopalvelulle, ja kyseinen asiaksohjelmistoversio voidaan vetää pois jakelusta.

Tilanvalvojan toiminta perustuu asiaksohjelmiston tarkkailuun, jossa käynnissä olevaa ohjelmaa tarkkaillaan lyhyin aikaväleihin. Jos mikä tahansa ohjelmiston prosesseista kaatuu, jää siitä jälki, ja riittävän monta kaatumista herättää rollback toiminnon. Tällöin varmuuskopioksi jätetty asiaksohjelmisto otetaan käyttöön ja lähetetään verkkopalvelulle ilmoitus, että varmuuskopio on käytössä. Asiakasohjelmiston vaihdos kestää muutaman sekunnin ja asiakkaan toiminto jatkuu taas normaalisti.

8.5 Kokonaiskuva

Edellä avatut yksittäiset osat muodostavat yhdessä toimivan automatisoidun järjestelmän, joka alkaa asiaksohjelmiston kasauksella ja lopulta päättyy asennettavaksi gateway laitteelle. Manuaalista työtä prosessissa on putkisto prosessin aloitus, laitekohtaisten parametrien päivittäminen sekä asiaksohjelmiston testaaminen. Alla oleva kuva 12 selvittää koko järjestelmän toimintaa. Kuvan nuolilla selvennetään viestiliikenteen suuntaa. Kohdissa joissa on nuolet molempiin suuntiin, alkavat tiedostojen tiedusteluilla, joihin vastauksena tarjoillaan haluttu tiedosto. Kuva on jaettu kolmeen osioon, joilla kuvastetaan kehitetyn järjestelmän jokaista osa-aluetta. Ylhäällä on CI/CD -putkisto, keskellä verkkopalvelun asiaksohjelmistojen hallinta ja alhaalla gatewayn päivittäjäohjelman toiminta.



Kuva 12. Tuotannon IoT laitteiden automaattinen päivittäminen

9 Yhteenveto

Työn tavoitteena oli kehittää MVP mallin mukainen järjestelmä, joka automatisoi hajautettujen asiaksalaitteiden ohjelmistojen päivittämisen. MVP mallin mukainen järjestelmä saatiin toteutettua ja keskeiset tapeet täytettiin. Järjestelmä kattaa automaattisen päivittämisen kannalta kaiken oleellisen, eli hajautetut gateway -laitteet hakevat itsenäisesti niille suunnatut ohjelmistot ja asentavat ne. Ohjelmistoversioita hallinnoidaan mScales verkkopalvelusta ja jokaiselle laitteelle voidaan määrittää tavoite ohjelmistoversio.

Järjestelmä on menossa tarkastettavaksi, jossa sen toiminnallisuutta ja käytettävyyttä testataan. Jos tarkastusvaiheessa ei ilmene ongelmia, lähdetään järjestelmää viemään vaiheittain tuotantoon. Aluksi järjestelmää kokeillaan muutamaaan laitteeseen ja, kun sen todetaan toimivan halutulla tavalla se viedään hiljattain jokaiseen laitteeseen.

Järjestelmä toimii sellaisenaan ja tekee hajautettujen laitteiden päivittämisestä helpompaa. Järjestelmän saavutetuilla hyödyillä voidaan vapauttaa ohjelmistokehittäjien resursseja kehitystyöhön. Työssä tehtiin pieni osa suuresta kokonaisuudesta, jossa minimaaliset vaatimukset täytettiin. Kehitettävää jäi vielä paljon, mutta työssä kehitetyn järjestelmän päälle voidaan rakentaa helpommin uusia ominaisuuksia. Päivitysprosessissa on vielä manuaalista työtä, jota on tarkoitus lähteä automatisoimaan lähitulevaisuudessa. Yksi suurimmista jatkokehitystyön kohteista on automattitestien kehittäminen ja liittäminen osaksi CI/CD -putkistoa. Laitteiden ohjelmistopäivitysten hallintaa on myös tarkoitus lähteä parantamaan sekä lisäämään ominaisuuksia laitteiden päivitysprosessiin.

Lähteet

Abildskov, J. 2020. Mitä on DevOps? Eficode. Viitattu 29.3.2021. Saatavissa <https://www.eficode.com/blog/mita-on-devops>

Anastasov, M. 2019. CI/CD Pipeline: A Gentle Introduction. Viitattu 29.3.2021. Saatavissa <https://semaphoreci.com/blog/cicd-pipeline>

Axiomtek, IIoT Device Management, Viitattu 29.3.2021, Saatavissa <https://www.axiomtek.com/Default.aspx?MenuId=Solutions&FunctionId=SolutionView&Itemid=2035>

Empirica Finland Oy. Mikä on IoT? Viitattu 29.3.2021. Saatavissa: <https://www.empirica.fi/iot/>

Eronen, H. 2017. IaaS, PaaS, SaaS? mikä pilvipalvelu sopii yrityksellesi. Viitattu 29.3.2021. Saatavissa <https://www.planeetta.fi/2016/03/15/iaas-paas-saas-mika-pilvipalvelu-sopii-yrityksellesi/>

Helsingin Yliopisto. Osa 2 - Versionhallinta: Git ja Github. Viitattu 29.3.2021. Saatavissa <https://tkk-lapio.github.io/git/>

Holst, A. 2021. Number of Internet of Things (IoT) connected devices worldwide from 2019 to 2030. Statista. Viitattu 29.3.2021. Saatavissa <https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/>

Hurja, 2018. Viitattu 29.3.2021. Saatavissa <https://www.hurja.fi/blogi/matkalla-ideasta-ohjelmistoksi-prototyyppi-ja-minimum-viable-product-eli-mvp/>

Johnston, S 2009. Pilvipalvelu, Wikipedia, Viitattu 29.3.2021. Saatavissa https://upload.wikimedia.org/wikipedia/commons/b/b5/Cloud_computing.svg

Järvenpää, J. 2020. Ketterä kehittäminen miten ja miksi. Viitattu 29.3.2021. Saatavissa <https://www.vincit.fi/fi/kettera-kehittaminen-miten-ja-miksi/>

Kangasniemi, H & Lintulahti, M 2017. Mikä on pilvipalvelu? Elisa. Viitattu 29.3.2021. Saatavissa <https://elisa.fi/ideat/mika-on-pilvipalvelu/>

Laaksonen, A. 2015. Pilvipalvelut pähkinänkuoressa. Viitattu 29.3.2021. Saatavissa <http://www.pilveen.net/2015/09/pilvipalvelut-pahkinankuoressa.html>

Lahti Precision Oy, 2020a. mScales - Digitalisoi punnitusketjusi yhdellä liitoksella. Viitattu 29.3.2021, saatavissa <https://lahtiprecision.com/mscales/>

Lahti Precision Oy, 2020b. Teräs- ja betonisiltaiset vaa'at, Viitattu 29.3.2021. Saatavissa <https://lahtiprecision.com/products/ajoneuvovaaka/>

Lahti Precision Oy, 2020c. Miten digitalisoitu punnitusketju eroaa perinteisestä tavasta? Viitattu 29.3.2021 Saatavissa <https://www.mscales.com/fi/>

Lavery, T. 2017. IoT gateway. TechTarget. Viitattu 29.3.2021. Saatavissa <https://whatis.techtarget.com/definition/IoT-gateway>

Mazyar, M. 2019. DevOps: The Ultimate Way to Break Down Silos. Viitattu 29.3.2021. Saatavissa <https://devops.com/devops-the-ultimate-way-to-break-down-silos/>

Mixon, E. 2019. OTA (over-the-air) update. TechTarget. Viitattu 21.3.2021. Saatavissa <https://searchmobilecomputing.techtarget.com/definition/OTA-update-over-the-air-update>

Murugan, B. 2020. Top 10 test automation frameworks in 2020. Viitattu 29.3.2021 . Saatavissa <https://dzone.com/articles/top-10-test-automation-frameworks-in-2020>

Oracle. What is IoT? Viitattu 29.3.2021, saatavissa: <https://www.oracle.com/internet-of-things/what-is-iot/>

Pendolin, H. 2018. Mikä on MVP – ja mitä se ei ole. Viitattu 29.3.2021. Saatavissa <https://tuotejohtaminen.fi/mita-tarkoittaa-mvp-ja-mita-ei/>

Posey, B. 2020. industrial internet of things (IIoT). TechTarget. Viitattu 7.3.2021. Saatavissa <https://internetofthingsagenda.techtarget.com/definition/Industrial-Internet-of-Things-IIoT>

Ranger, S. 2020. What is the IoT? Everything you need to know about the Internet of Things right now. Viitattu 29.3.2021. Saatavissa <https://www.zdnet.com/article/what-is-the-internet-of-things-everything-you-need-to-know-about-the-iot-right-now/>

Sacolick, I. 2020. What is CI/CD? Continuous integration and continuous delivery explained. Viitattu 29.3.2021. Saatavissa: <https://www.infoworld.com/article/3271126/what-is-cicd-continuous-integration-and-continuous-delivery-explained.html>

Symanovich, S, 2019. The future of IoT: 10 predictions about the Internet of Things. Norton. Viitattu 29.3.2021. Saatavissa (<https://us.norton.com/internetsecurity-iot-5-predictions-for-the-future-of-iot.html>)

Temboo, 2018. Viitattu 29.3.2021. Saatavissa <https://medium.com/@temboo/how-to-approach-ota-updates-for-iot-d088c217b31c>

Trend Micro, 2020. Smart Yet Flawed: IoT Device Vulnerabilities Explained. Viitattu 29.3.2021. Saatavissa <https://www.trendmicro.com/vinfo/us/security/news/internet-of-things/smart-yet-flawed-iot-device-vulnerabilities-explained>

Vertics. Ohjelmistotestauksen perusteet. Viitattu 29.3.2021, Saatavissa <https://vertics.co/ohjelmistotestauksen-perusteet/>