

Aalto University
School of Science
Degree Programme in ICT innovation

Relja Paunovic

Contracting Service For Industrial Internet

Master's Thesis
Espoo, June 18, 2011

DRAFT! — September 12, 2017 — DRAFT!

Supervisor: Professor Petri Vuorimaa, Aalto University
Advisor: Instructor?

Aalto University
 School of Science
 Degree Programme in ICT innovation

ABSTRACT OF
 MASTER'S THESIS

| | | | |
|---|---|---------------|---------|
| Author: | Relja Paunovic | | |
| Title: | Contracting Service For Industrial Internet | | |
| Date: | June 18, 2011 | Pages: | vi + 51 |
| Major: | Hypermedia | Code: | T-110 |
| Supervisor: | Professor Petri Vuorimaa | | |
| Advisor: | Instructor? | | |
| <p>A dissertation or thesis is a document submitted in support of candidature for a degree or professional qualification presenting the author's research and findings. In some countries/universities, the word thesis or a cognate is used as part of a bachelor's or master's course, while dissertation is normally applied to a doctorate, whilst, in others, the reverse is true.</p> <p>!FIXME Abstract text goes here (and this is an example how to use fixme). FIXME! Fixme is a command that helps you identify parts of your thesis that still require some work. When compiled in the custom mydraft mode, text parts tagged with fixmes are shown in bold and with fixme tags around them. When compiled in normal mode, the fixme-tagged text is shown normally (without special formatting). The draft mode also causes the "Draft" text to appear on the front page, alongside with the document compilation date. The custom mydraft mode is selected by the mydraft option given for the package aalto-thesis, near the top of the thesis-example.tex file.</p> <p>The thesis example file (thesis-example.tex), all the chapter content files (1introduction.tex and so on), and the Aalto style file (aalto-thesis.sty) are commented with explanations on how the Aalto thesis works. The files also contain some examples on how to customize various details of the thesis layout, and of course the example text works as an example in itself. Please read the comments and the example text; that should get you well on your way!</p> | | | |
| Keywords: | ocean, sea, marine, ocean mammal, marine mammal, whales, cetaceans, dolphins, porpoises | | |
| Language: | English | | |

Acknowledgements

Will write later

Espoo, June 18, 2011

Relja Paunovic

Abbreviations and Acronyms

| | |
|----------|---|
| IOT | Internet Of Things |
| IIOT | Industrial Internet of Things |
| W3C | World Wide Web Consortium |
| OMA | Open Mobile Alliance |
| ETSI | European Telecommunications Standards Institute |
| M2M | Machine To Machine |
| GSCM2MTF | Global Standards Collaboration Machine-to-Machine Task Force (GSCM2MTF) |
| 3GPP | 3rd Generation Partnership Project |
| REST | Representational State Transfer |
| SCL | RESTful Service Capability Layer |
| LWM2M | Light Weight Machine-to-Machine |
| CoAP | Constrained Application Protocol |
| DoS | Denial of Service |
| FQDN | Fully Qualified Domain Name |
| CES | Customer Edge Switching |
| W3C | World-wide-web Consortium |
| WOT | Web Of Things |
| JSON-LD | JavaScript Object Notation for Linked Data |

Contents

| | |
|---|-----------|
| Abbreviations and Acronyms | iv |
| 1 Introduction | 1 |
| 1.1 Research Question | 2 |
| 1.2 Outline | 2 |
| 2 Literature Review | 4 |
| 2.1 Internet of Things | 4 |
| 2.1.1 Industrial Internet of Things | 5 |
| 2.1.2 Open Issues | 5 |
| 2.1.2.1 Standardization | 6 |
| 2.1.2.2 Security and privacy | 7 |
| 2.2 Notable standards | 7 |
| 2.2.1 Light Weight Machine to Machine (LWM2M) | 8 |
| 2.2.2 World-wide-web Consortium (W3C) - Web Of Things (WOT) | 9 |
| 2.3 Policy Based Communications for 5G Mobile with Customer Edge Switching | 12 |
| 3 Requirements | 14 |
| 3.1 Context | 14 |
| 3.1.1 Smart Crane | 14 |
| 3.1.1.1 Maintenance | 15 |
| 3.1.1.2 Users of the Crane | 17 |
| 3.1.1.3 KoneCranes | 17 |
| 3.1.2 Smart Traffic | 18 |
| 3.1.3 Connected Goods | 18 |
| 3.2 Contracting Service Requirements | 19 |
| 3.2.1 General Requirements | 19 |
| 3.2.1.1 Security | 20 |
| 3.2.1.2 User Experience | 20 |

| | | |
|----------|--|-----------|
| 3.2.1.3 | Functional Requirements | 21 |
| 3.2.2 | Administrator Requirements | 21 |
| 3.2.3 | Manufacturer Requirements | 22 |
| 3.2.4 | Customer Requirements | 23 |
| 4 | Prototype Implementation | 26 |
| 4.1 | Technologies Used | 26 |
| 4.1.1 | Django Framework | 26 |
| 4.1.2 | Bootstrap | 28 |
| 4.2 | Overview | 28 |
| 4.2.1 | Account Types | 28 |
| 4.2.2 | Device Management | 29 |
| 4.2.3 | Policy Management | 31 |
| 4.3 | Manufacturer | 32 |
| 4.4 | Customer | 35 |
| 4.5 | User | 38 |
| 5 | Evaluation and Discussion | 39 |
| 5.1 | Evaluation of General Requirements | 39 |
| 5.1.1 | Security | 39 |
| 5.1.2 | User Experience | 41 |
| 5.1.3 | Functional Requirements | 43 |
| 5.2 | Evaluation of Administrator Requirements | 43 |
| 5.3 | Evaluation of Manufacturer Requirements | 43 |
| 5.4 | Evaluation of Customer Requirements | 45 |
| 6 | Conclusion and Future Work | 48 |
| A | Appendix A | 51 |

Chapter 1

Introduction

In the past few decades, vision of a world where sensors and actuators are installed everywhere is becoming real. Today, we have wearable health monitors with direct contact with a physician, smart pet feeders choosing perfect food for your pet and smart toothbrush that gamifies mundane task of brushing teeth. These are just a few examples from the sea of different smart solutions and more are emerging every day creating an Internet Of Things (IOT). Parallel to the development of IOT for mass use, IOT is being developed for industrial purposes known as Industrial Internet Of Things (IIOT). For example, smart indoor cranes packed with many features, including sensing of obstructions and material wear off, and reporting of its location. These features are used by different actors in the ecosystem, such as maintenance companies being interested in usage statistics or worker supervisor in a factory monitoring how the workers are operating the crane. Since these ecosystems can be very large, robust security architectures are required to battle classical weaknesses of the Internet.

As a result of rapid development in the area, substantial vertical fragmentation has emerged, where large corporations have created their own proprietary systems. Since the development of these systems has been mostly independent from each other, they differ greatly and are incompatible with each other. As an attempt to solve this fragmentation, many standardization agencies have proposed their solutions. Most notable solutions are developed by Open Mobile Alliance (OMA) and World Wide Web Consortium (W3C) which will be described thoroughly in chapter 2. Although, none of the standardization attempts have yet been widely accepted and implemented.

This thesis proposes a solution that will allow easy management of access to devices in a secure and robust way. It is a web platform where manufacturers can assign machines and devices to customers that have bought them. Customers can then declare who has access to those devices by simply

adding or removing policies in a centralized database. This thesis is based on a Light-weight Machine-to-Machine (LWM2M) standard proposed by OMA and Policy Based Communication proposed by [8] which will be thoroughly described in chapter 2.

1.1 Research Question

As mentioned above the main focus of this thesis is to design a technical solution for management and organization of millions of devices in a secure way. Thus, a problem that this thesis is aiming to solve is:

”How to design a solution for easy and secure access management to large number of devices?”

In order to answer this question, several sub questions needs to be addressed first:

1. What are the requirements of such a solution?
2. What are necessary components of such a solution?
3. Are the requirements met?

To answer these questions, firstly, relevant research needs to be evaluated. Secondly, use cases of these devices needs to be examined in order to understand the requirements. Thirdly, number of components to meet the requirements needs to be minimized. Finally, links between requirements and actual components needs to be justified.

1.2 Outline

In this chapter I have given background information about IOT and problems related with it in order to explain where this thesis fits. This chapter also gives a hint of use cases and how the solution is going to look like followed by a concise description of which problem I am addressing.

Firstly, in chapter 2 I will describe relevant research. It will explain in more detail what is IOT and IIOT, and what are the differences in my context, followed by description of open questions (and proposed solutions) regarding standardization and security in section 2.1.2. In the remainder of chapter 2 I will give a more detailed description of most promising standardization attempts while highlighting the differences and justifying the choice

used for this thesis. In the end of the chapter, I will describe work Policy Based Communications work [8] which is a basis for this thesis.

Secondly, chapter 3 will give an in depth description of context in which IIOT devices are used through three different use cases, highlighting the similarities and differences between them. After the reader is familiarized with the context, I will list requirements drawn from them.

Thirdly, chapter 4 will give a detailed description of prototype implementation I have constructed based on knowledge gathered in chapter 2 and requirements gathered in 3. This chapter will describe technologies used, while giving a justification of why they were chosen for this implementation. In the following sections I will describe all necessary components by describing how to use the solution. This description will be given for all different account types which represent different users of the solution.

Fourthly, chapter 5 shows how the requirements are met in implementation. It will assess every requirement and corresponding component from chapter 4.

Finally, chapter 6 concludes the thesis by giving an answer to the defined research question, summarizes the main findings and gives direction for future work on this topic, highlighting the flaws of this solution and what needs to be done in order to improve it.

Chapter 2

Literature Review

In the following section, I will give overview about Internet of Things (IOT) and Industrial Internet of Things (IIOT), highlighting differences between them. Further, I will describe open issues in IIOT followed by standardization attempts and related work regarding security. At the end of this chapter, I will give a more detailed description of most notable standards, Light Weight Machine-to-Machine (LWM2M) and Web of Things at the end of this chapter followed by Policy Based Communications, which is the basis of this work.

2.1 Internet of Things

Internet of things (IOT) is a relatively new concept gaining momentum since the start of this century. Although, first examples of IOT date back as early as 1983 with an automated inventory system. IOT can be seen as an extension to Internet, with physical devices (such as sensors and actuators) communicating with each other and with humans creating an enormous network.

Nowadays, with the continuous decrease in cost of computational devices, we are able to produce powerful devices with communication abilities for a very low price. With the increase in number of devices that needs to be connected (some estimates show up to 75 billion connected devices by year 2025¹), issues regarding scalability, security, heterogeneity of devices, etc. have emerged.

These issues are holding back the progress of IOT because it forces companies to create their own proprietary systems to fit their needs, which is only an option for large companies with financial capabilities to do so. This leads

¹<https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/>

to large number of systems designed for similar purposes but incompatible with each other (due to use of different protocols or architectures). Temporary solution for this problem is to introduce a middle-ware as proposed by [2] that acts like a bridge between two systems, but this has scalability issues and with the rise of number of connected devices will soon be unacceptable. In order to tackle these issues, several standards have been proposed, most notably smartM2M, oneM2M and most recently LightweightM2M (LWM2M), which will be explained in 2.1.2.1. None of these have yet become *de facto* standard.

In the remainder of this section, differences between Industrial Internet of Things (IIOT) and IOT will be introduced followed by open issues of IOT and IIOT in 2.1.2.

2.1.1 Industrial Internet of Things

In IOT, a rough distinction is made between consumer and industrial IOT according to [2]. Consumer IOT applications are aimed to make everyday life easier by saving time and money, such as smart locks, smart homes and wearable hearth monitors. On the other hand, industrial IOT (such as production, automation and intelligent computation systems) focuses on how smart machines, data analytics and networked sensors can improve services in business-to-business domain [12]. As an example, predictive maintenance can generate savings up to 12% over scheduled repairs, leading to a 30% reduction in maintenance cost and a 70% cut in downtime from equipment breakdowns according to Accenture². This usually implies extensive machine-to-machine (M2M) communication compared to consumer IOT, where in most cases real time guarantees are not required.

Generally, IIOT has stricter requirements regarding delay, security and general robustness compared to consumer IOT. This is because failures in these devices can have consequences on safety of people and environment. For example, in a factory setting, pressure sensor installed on an indoor crane can cause serious damage by failing to communicate about an obstruction.

2.1.2 Open Issues

As previously mentioned, there are many issues surrounding IOT, most notably, lack of standards and security. These issues will be explained in the remainder of the section.

²<https://www.accenture.com/us-en/insight-industrial-internet-of-things>

2.1.2.1 Standardization

According to Global Standards Collaboration Machine-to-Machine Task Force (GSCM2MTF) there are more than 140 organizations involved in the M2M standardization process worldwide. This is a huge vertical fragmentation of IOT market, and it is a result of long history in Industrial use, starting from seventies with process control systems that continued to be used in to-days process automation systems. As already mentioned, these systems are proprietary and are incompatible with each other.

In an attempt to resolve this fragmentation issue three notable initiatives stand out. SmartM2M is an initiative led by European Telecommunications Standards Institute (ETSI), it is based on RESTful Service Capability Layer (SCL) [1] which is available through open interfaces. Resource tree residing on SCL along with procedures for handling them is standardized following Representational State Transfer (REST) principles allowing technology agnostic way of accessing them. SmartM2M also defines security framework including authentication, M2M service bootstrap, key agreement and establishment, and M2M service connection procedures, based on a key hierarchy of the M2M node[6]. Unfortunately, it may have issues with scalability as pointed out by [7], thus, making it unsuitable for IOT and IIOT.

A follow up project was formed in order to resolve issues with SmartM2M, this time with a broader partnership. It is an international project started by seven telecom standards organizations: Association of Radio Industries and Businesses (ARIB) and Telecommunication Technology Committee (TTC), Japan; the Alliance for Telecommunications Industry Solutions (ATIS) and Telecommunications Industry Association (TIA), United States; the China Communications Standards Association (CCSA), China; the European Telecommunications Standards Institute (ETSI), Europe; and the Telecommunications Technology Association (TTA), Korea. This project is based on RESTfull design, same as SmartM2M, resource naming conventions same as SmartM2M and is grounded on horizontal service layer principle. Although, it relaxes the scalability constraints by using hierarchical organization of different actors in the system. With this improvement with respect to SmartM2M, it is a serious contender to become IOT standard, being already adopted by various companies according to [13]. Along with these improvements, it defines a security architecture in three layers: security functions, security environment abstraction and security environment.

Recently, a new standard has emerged from Open Mobile Alliance (OMA) targeting constrained devices named Light Weight Machine-to-Machine (LWM2M). It defines a fast deployable client-server specification while minimizing memory consumption and network overhead making it very appealing to IOT and

IIOT devices. It provides device management and security work-flow for IOT applications in a very light weight manner. Recent results from [14] show that memory footprint overheads on a client side protocol stack are no more than 6-9%.

Another standard has recently been announced by World-wide-web Consortium (W3C) named Web of Things (WOT). This standards aims to provide a middleware (presented as an abstraction layer) to connect different IOT platforms and protocols. Detailed description of LWM2M and WOT will be given in section 2.2 since they represent two best candidates for this work.

2.1.2.2 Security and privacy

Devices in IOT generate, process and exchange vast amount of data that is safety-critical and/or private and they are subject to various attacks. Therefore, it is crucial to assure integrity of the devices code and data from malicious modifications [4]. In IIOT, following two requirements are crucial for security according to [16]. Availability is most important requirement, because it can lead to loss in productivity and consequently loss of revenue. This is particularly affected by denial of service (DoS) attacks and preventing any system failure that may result in physical damage or harm to humans, particularly affected by sabotage.

There are many security architectures for embedded IOT devices, although, majority of them are too complex for low-end devices. Solutions for low-end devices usually rely on physical (hardware) isolation of security-critical code and data from other software on same device. Examples of such architectures are SMART [5], SPM [15], SANCTUS [11] and TrustLite [9] but they all have major flaws. SMART does not allow code changes after deployment. SPM and SANCTUS have hardware assisted task isolation but they are non-interruptible, which violates real-time guarantees. TrustLite requires all software components to be loaded at boot time, which reduces flexibility. TyTAN [3] is the only work that provides secure loading of tasks at runtime, secure inter-process communication, local and remote attestation and real-time guarantees.

2.2 Notable standards

Following section will describe two most notable standards which were final candidates for this thesis. These standards come from Open Mobile Alliance

and World-wide-web Consortium which are two biggest standardization bodies in mobile communications and Web respectively.

2.2.1 Light Weight Machine to Machine (LWM2M)

OMA has approved first version (V1.0) of LWM2M standard in February 2017, since it is a very recent specification not many implementations exist, although, number has been steadily increasing since then. Most notable implementation by [14] focuses on client side architecture (residing on IOT devices).

LWM2M provides a light and secure communication interface with compact data model to enable device management and service for IOT (and IIOT) devices. It is a client-server architecture named OMA LWM2M Enabler. Enabler consists of LWM2M Server, which is a central point that takes care of the devices, assigning security keys, registering device capabilities and more. Another part of Enabler is LWM2M Client, which resides on a device and provides necessary information to the Server. Furthermore, it uses Constrained Application Protocol (COAP), instead of HTTP, which has a greatly reduced communication overhead making it ideal for constrained devices (or devices benefiting from efficient communication, as the case in IIOT) along with UDP/SMS transport bindings. Architecture of LWM2M is shown on Figure 2.1.

The LWM2M Enabler defines the application layer communication protocol between Server and Client. It is separated into four logical interfaces, namely, Bootstrap, Device Discovery and Registration, Device Management and Service Enablement, and Information Reporting. Diagram showing interfaces and corresponding messages is illustrated in 2.2.

1. Bootstrap: allows LWM2M Bootstrap server to manage keying, access control and to configure device for communication with the Server.
2. Device Discovery and Registration: allows Server to discover devices and register their capabilities, e.g., which objects and how to access them does a device have.
3. Device Management and Service Enablement: allows LWM2M Server to manage devices and provide M2M service by sending operations to devices and getting corresponding responses from them.
4. Information Reporting: This is a core interface in LWM2M, which allows reporting of resource information. It can be triggered periodically, on events or on request.

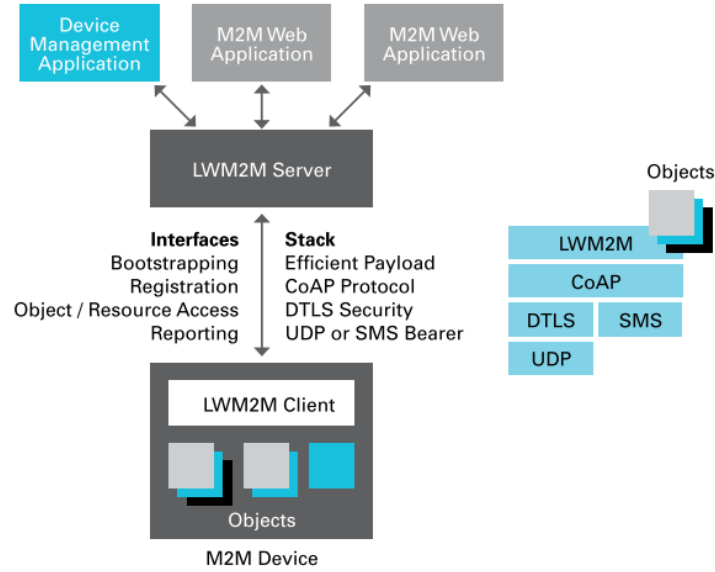


Figure 2.1: LWM2M Architecture

Communication model of LWM2M is based on CoAP methods similar to HTTP verbs GET, POST, PUT, DELETE to manipulate *resources* on devices. Compared to HTTP, CoAP starts with only four bytes of overhead in binary encoded message and is easily translatable to HTTP. Unlike HTTP, CoAP messages are exchanged asynchronously between CoAP end-points over a datagram-oriented transport, in this case UDP.

In LWM2M Enabler, each individually addressable piece of information is called a Resource and groups of Resources are logically organized into Objects. For example, predefined object *Location* contains all resources needed for locating the device. Full list of existing objects can be found at OMA registry ³.

2.2.2 World-wide-web Consortium (W3C) - Web Of Things (WOT)

Since there are many IOT platforms already existing, it is becoming a big problem for developers creating applications that span them. The solution W3C proposes aims to enable worldwide discovery and interoperability by exposing these platforms through the Web with a new class of webservers that

³<http://www.openmobilealliance.org/wp/OMNA/LwM2M/LwM2MRegistry.html>

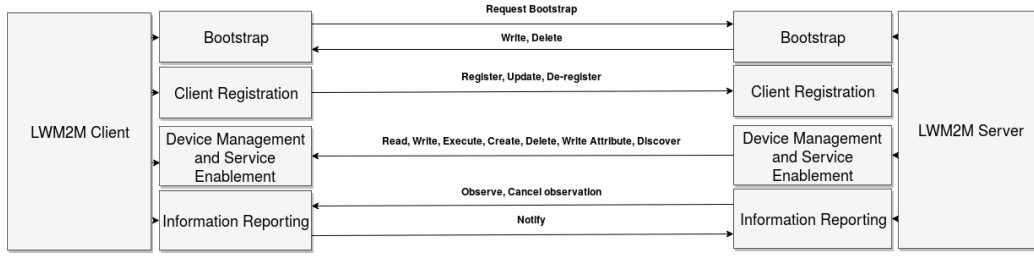


Figure 2.2: LWM2M Interfaces

support an open framework for the WOT as pictured on 2.3. In other words, WOT aims to interconnect existing IOT platforms and complement available standards, to reduce cost and risk, and to boost market opportunities.

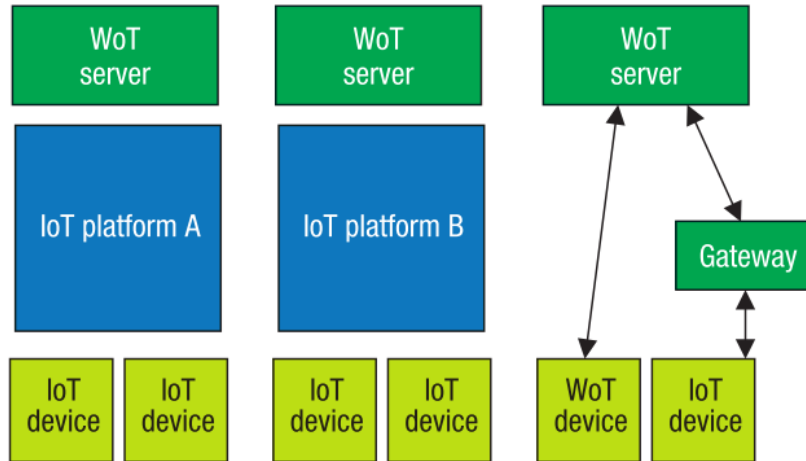


Figure 2.3: WOT framework that bridges the multiple existing IOT platforms

In W3C WOT all things (devices) are modeled as events, properties and actions, while also possessing metadata describing location of the device, serial number and similar. Firstly, events represent notifications from device to server, such as "battery low" or "the door just opened". Secondly, properties represent information that can be checked through server, such as the temperature or the state of the light switch (on or off). Finally, actions represent commands to coming from server to the device, such as "move the crane from position x to position y" or "turn off". In order to retrieve metadata for things, WOT uses JavaScript Object Notation for Linked Data (JSON-LD).

Analogous to the role played by the Internet as an abstraction layer for networks and networking technologies, WOT describes an abstraction layer over heterogeneous IOT standards, communication patterns, protocols and data formats. This abstraction layer is pictured on figure 2.4, where applications interact with software objects for things (devices, either virtual or physical). Webservers providing this abstraction can be placed on the edge of the network (or in the fog or cloud) to provide efficiency.

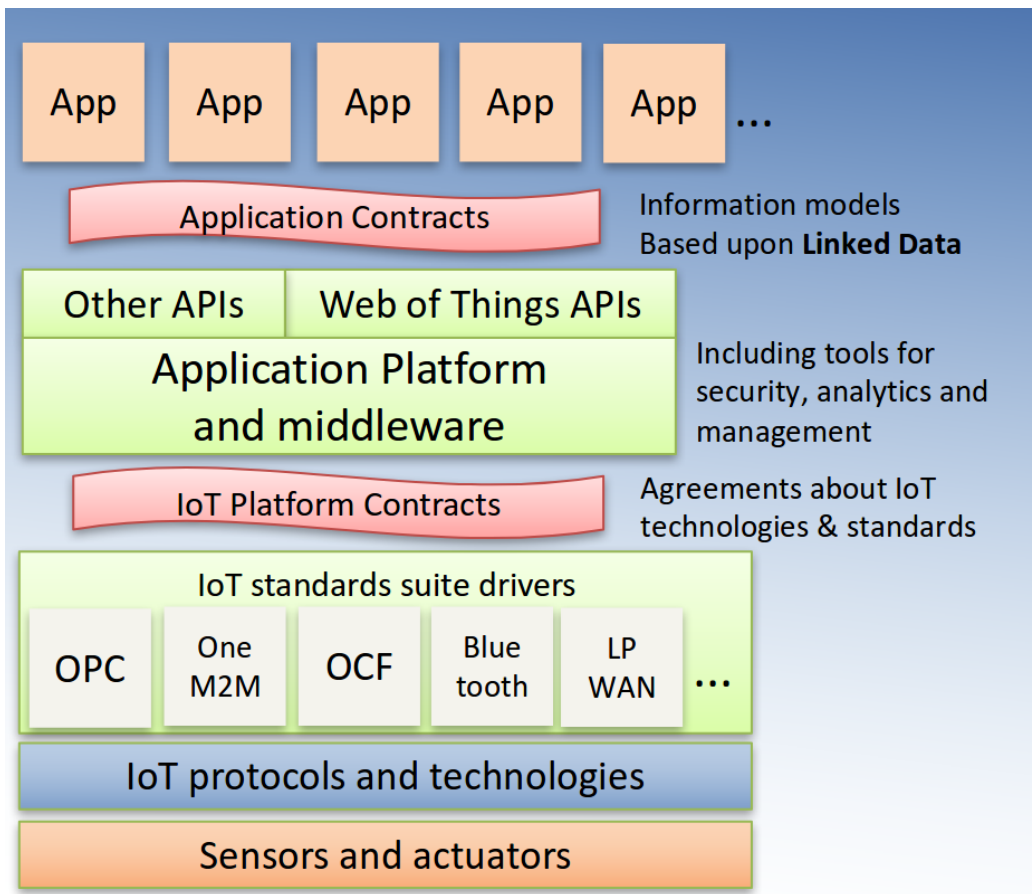


Figure 2.4: WOT abstraction layer

Regarding security, WOT is building upon existing security standards such as IOT security foundation, and IOT platforms such as oneM2M (described in section 2.1.2.1), claiming to provide end to end security.

W3C WOT is a new standard, with the working group formed in the beginning of 2017. Since it is a new standard, their work has mostly been abstract, without any real implementations. It is relatively heavy weight

compared to LWM2M standard, and focuses on providing a bridge between different platforms unlike LWM2M, that proposes a new solution not necessarily compatible with other existing solutions. Because the work on WOT is in a very early phase and reasons described above I have chosen to make use of LWM2M standard for this thesis.

2.3 Policy Based Communications for 5G Mobile with Customer Edge Switching

Particularly interesting work by [8] proposes a policy based communication with built in security, while also addressing classical weaknesses of Internet, namely, address spoofing, unwanted traffic and DoS attacks. It is based on a principle that before establishing communication between two hosts (or networks) they need to negotiate interests, and only if they are matching communication is established. These interests are described with a policy.

This work proposes to replace Network Address Translator (NAT) from the edge of the network with their own Customer Edge Switch (CES) node. This node will act exactly like NAT if the sender, who is behind a CES node, is interacting with a receiver using legacy ip. On the other hand, if the situation is reversed CES node will act as a *realm gateway*. Only if both actors are behind CES node it will act as a cooperative firewall negotiating interests of sender and receiver. Because of this, CES can be applied one network at a time making it suitable for IIOT purposes, where vertical fragmentation is a big issue.

Furthermore, CES allows efficient communication by dropping unwanted traffic at the edge and in that way reducing amount of traffic that passes through network. Also, CES makes use of Domain Name Servers (DNS) to find receivers faster using Fully Qualified Domain Names (FQDN) and MS-ISDN numbers.

In essence, policy database holds all policies and can be accessed through API. Before communication is established, policies of both actors are checked and if they match communication is allowed. Example is shown on 2.5.

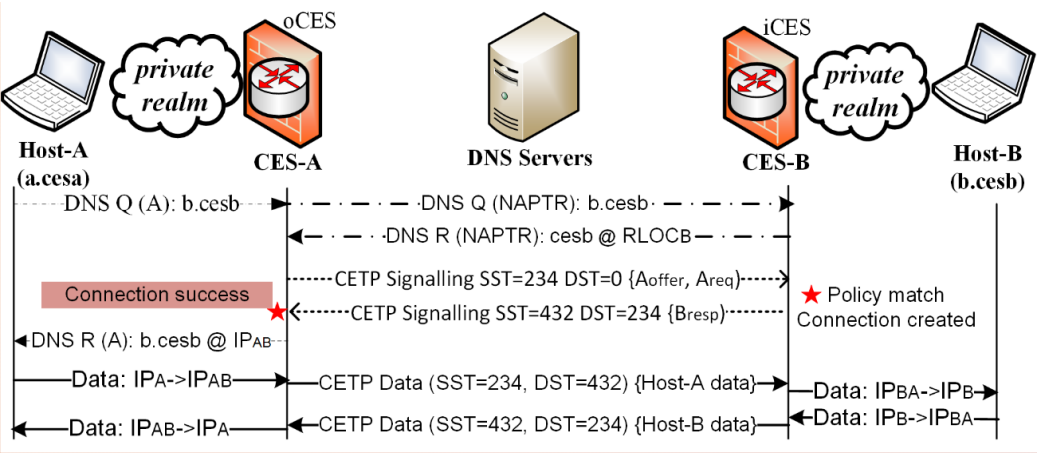


Figure 2.5: CES communication example

Chapter 3

Requirements

To design a solution, environment around the problem needs to be understood, specifically, stakeholders involved and what needs they have. This chapter explains several industrial use cases with aim to familiarize the reader with environment Contracting Service is tailored for, followed by concrete requirements of this solution. Since there are countless number of different use cases, not everything can be covered, especially because new applications are continuously emerging. Therefore, this solution needs to be flexible to cover most of the existing and future use cases.

3.1 Context

This section describes three representative industrial use cases in order to help reader understand the range of applications. All use cases have a common core even though they seem very different from each other as it can be seen from the following text.

3.1.1 Smart Crane

KoneCranes have donated a smart crane to Aalto Industrial Campus as a tool for research. This crane is an indoor crane whose head (which carries the load) is mounted on rails allowing it to reach every position inside, crane is pictured on figure 3.1. Following information was gathered through interviews with researchers and with KoneCranes and it will be used as a representative example for the rest of this work.

This crane has many sensors attached to make it automated and smart as possible. These sensors bring many features, namely **load weighting** (which can also be used to detect whether the crane is stuck somewhere), **remote**



Figure 3.1: KoneCranes Indoor Crane (model K16052)

monitoring of the position of the crane, **signaling** when some error or warning has occurred, **live video feed** from camera attached above the hook (at the moment, not used for automation but could be supported in the future, for example, image processing to detect obstructions) and more.

There are multiple stakeholders in this ecosystem and all of them require some of the data that Crane generates, although not all information should be available to them, only the bare minimum that is required by their business. These stakeholders are KoneCranes, maintenance service (part of the KoneCranes) and users of the crane. Currently, KoneCranes has access to all the data Crane generates, but in order to increase security, this should not be the case.

3.1.1.1 Maintenance

Maintenance service makes use of usage data (alarms, usage parameters) to monitor and predict maintenance needs of the cranes. They also use the data together with the customer, in order to review their maintenance spend of

the assets, study patterns to reveal relationships between variables and more. Following are some of the examples of use cases.

One part of the crane that needs to be changed most often are the breaks. Breaks have limited number of uses and this number is approximately known (there is a regulation in place that requires brakes to be changed regularly). In order to determine whether the breaks needs changing or repair, maintenance service requires information about their usage. This information is provided by a pressure sensor installed on the breaks. How much pressure is needed to make the crane head stop is directly proportional to weight the crane is carrying which can then be used to calculate the wear of the breaks.

Large and expensive machines like this crane are under warranties and in order to determine whether the warranty is valid, maintenance service needs to know if the crane has been used in a proper way. Therefore, they need information coming from different sensors installed on the crane. These sensors include:

1. Pressure sensor on the crane head: determining whether the weight carried is in the recommended limits.
2. Temperature sensor on rails: when the crane head moves on the rails, friction generates heat and heat levels needs to be in a recommended limits in order for warranty to be valid.
3. Pressure sensor on rails: determining whether they have been properly oiled.
4. Humidity sensor: which determines whether the crane is kept in an environment suitable for it.
5. Speed sensor: which determines whether the speed of crane head does not exceed recommended limits.

Sensors listed are just several examples, and they can provide enough information for maintenance companies to determine whether the crane is being properly used. Crane sensors can also detect some irregularities and issue a warning (or error) directly to maintenance companies so that the problem can be addressed as fast as possible.

Information like this should be disclosed to the maintenance service, while restricting the access to data that can be used to infer the processes inside the factory. Examples include, position of the crane in a certain point in time, or live feed from the camera mounted on the head.

3.1.1.2 Users of the Crane

Inside a factory there are different actors with different privileges. These actors can be roughly grouped in two categories: workers and managers. Worker operating the crane should have restricted access to data the crane operates. Information that he needs is tied to the current operation of the crane and some of the sensors that give that information are following. Position sensor, which tells him where the head of the crane is in space, and pressure sensor measuring the weight of the load. On the other hand, worker should not have information about the conditions of the brakes or temperature of the rails since it is of no use to him.

Managers responsible for multiple workers and machines should have access to all information related to operations of these machines. In that way, they can monitor their workers remotely, ensuring that everything is operating smoothly.

In the future, machine to machine communication will be utilized much more, for example, in a setting with two Cranes operating (automated, not by human) in the same room they should be able to signal to each other with their planned path in order to avoid collisions and to optimize the process. In this way, need for a centralized control is eliminated (or at least minimized).

3.1.1.3 KoneCranes

KoneCranes needs access to some of the data that Crane generates, in order to make product improvements, react to possible reliability problems and get better specification for new product generations, and adjust warranties. They analyze all types of data that Crane generates including:

1. Manufacturing data: such as component lists and manufacturing dates.
2. Usage data: including number of hours in operation, weights carried, mileage of the crane head and more
3. Sensor data: such as vibrations and temperature
4. Maintenance data: examples include maintenance task history, ordered materials and more.

As mentioned, at the moment KoneCranes has access to all the data Crane generates and their customers are aware of that. For privacy reasons, and in environment with multiple manufacturers, data access should be restricted to only what is needed for a specific role.

3.1.2 Smart Traffic

Recently, smart traffic is emerging as promising trend with the idea to automate transportation of goods and people. Big corporations, such as Google, Tesla, Mercedes and Uber have already joined the race with their models but many technological, legal and business obstacles are holding back its deployment. Integration of these vehicles into regular, non-automated traffic is a big challenge since it needs to take into account human error and correct it if possible.

One interesting example are smart convoys, driver-less trucks transporting goods in a convoy. For this to be secure, communication between trucks needs to be in real-time so that all, for example, obstructions noticed by sensors on truck in front are conveyed to the rest in time to react (slowing down or avoiding obstruction). It is also very important that only authorized people have access to data convoy produces, because if something gets tampered with, human lives are at stake along with structural damages.

For these security reasons not all stakeholders should have access to all the data but only to what is necessary for their operations. Stakeholders involved are: manufacturers; maintenance company; users; and third parties. Firstly, manufacturers should not have access to data that discloses anything that is confidential for truck users, for example, exact location of the convoy in any time because this information can be used to get insight into operations of users. Secondly, manufacturers might not want to give out all information to their users, either because of confidentiality or because it might discredit them. Thirdly, maintenance companies should have only information about the state of the parts of truck. How many times have the breaks been used, level of motor oil, gas and similar, which they need in order to schedule maintenance control. Finally, third parties could be any company, or public, that benefits from data these trucks produce and fit into business models of manufacturers or users. For example, traffic control application that provides information about how many convoys are on a particular road so that if the number is too high, traffic jams are expected.

3.1.3 Connected Goods

The servitisation of physical goods will be of strategic importance for the manufacturing industry, where instead of selling parts and machines it will be possible to sell engine hours, kilometers and similar. The vision here is that goods will remember how they were made and produce data throughout whole cycle of their usage, even giving insights into customer satisfaction with those goods. In this case, privacy is a big obstacle, because it is hard

to assure customer that data you collected in their home or workspace is not going to be used by anybody they do not want to.

Consider a scenario where all goods in your apartment from carton of milk to air-conditioning are equipped with sensors. Milk carton might possess a heat sensor, which alerts when milk is being kept in a warm place for too long, labeling it as spoiled and ordering fresh one from a local marketplace. Same data can be used for statistical purposes by a third party, to determine, for example, how much milk is being wasted by a nation and use it to adjust size of milk cartons or predict peaks in milk consumption. With some more complex goods, such as air-conditioning, predictive maintenance can be realized with sensors that tell if a particular part is wearing off and alert maintenance service specified by user or manufacturer of that machine. Subset of the data generated by air-conditioning can be used by health organizations to determine whether people are living in unhealthy environments, for example, by checking the ratio between how many times air filters have been changed over hours of usage.

Scenarios described in previous paragraph are just few out of many possible ones and it is impossible to tell, which ones will be implemented. Although, we have to prepare for the future by designing a flexible system that can withstand rapid changes in the ecosystem.

3.2 Contracting Service Requirements

At this point, the environment Contracting Service is tailored for has been explained. It is clear from previous section that this solution requires different accounts for all actors involved according to their role. These roles are, roughly, be divided into three groups: Administrators, Manufacturers and Customers.

Remainder of this section is organized into four subsections. First subsection will give general requirements for this platform, spanning requirements regarding security, user interface design rules and functional requirements required for all roles. Following subsections will describe requirements for each role in the system, administrator, manufacturer and customer respectively.

3.2.1 General Requirements

Following text will describe general requirements for this solution, drawn from the context explained in section 3.1. It will address requirements related to security, user experience and functional requirements not tied to any specific role.

3.2.1.1 Security

This work describes a platform that is used manage access to machines and devices. These machines are usually very expensive, and produce data which could reveal trade secrets of their users if unauthorized people get access to them. Moreover, accessing these machines without permission can lead to serious property damage and injuries to people working with them. Therefore, making this platform secure is of great importance. Following list will describe requirements that must hold in order for the platform to be usable:

1. ***Only authenticated users can access platform:*** This means that freely creating accounts should not be allowed. Only verified manufacturers and customers should have access to any of it.
2. ***Platform needs be to robust against common cyberattacks:*** This requirement is hardest to fulfill since the list of cyberattacks is constantly expanding. Although, providing security for the common ones is possible, while constantly adapting to newly emerging ones. List of common cyberattacks include cross-site request forgery, cross-site scripting, clickjacking, SQL injection and (distributed) denial of service attacks.
3. ***Only authorized people can manipulate machines or devices:*** Meaning that as a manufacturer, you can only manipulate machines and devices that you have added to platform. And, as a customer, you can only manipulate machines and devices that are assigned to you.
4. ***Users personal information can only be changed by them:*** This requirement protects the identity of a user, while also protecting their privacy.

3.2.1.2 User Experience

Every user interface need to be designed in an intuitive way in order to reduce the amount of effort required to learn how to use it. Well designed interface positively affects the adoption rate which is highly important. Nielsen et. al. [10] has proposed ten rules of thumb required for good design, which are widely accepted as good design rules. These rules are listed and briefly described bellow:

1. ***Visibility of system status:*** Users should be informed about what is going on in the system, through easily understandable feedback.
2. ***Match between system and the real world:*** Use real world terms, instead of system oriented terms.

3. ***User control and freedom***: Support undo and redo of all actions in the system.
4. ***Consistency and standards***: Follow conventions of the platform.
5. ***Error prevention***: Make it hard for users to make mistakes.
6. ***Recognition rather than recall***: Reduce users memory load by making objects, actions, and options visible.
7. ***Flexibility and efficiency of use***: Allow users to tailor frequent actions.
8. ***Aesthetic and minimalist design***: Dialogues should not contain information that is irrelevant or rarely needed.
9. ***Help users recognize, diagnose, and recover from errors***: Error messages should be in plain language, problem should be clearly indicated and solution constructively suggested.
10. ***Help and documentation***: Provide an easy way to learn how to use the platform.

3.2.1.3 Functional Requirements

All roles in the system, except administrator, have three requirements in common. These requirements are following: users should be able to view their personal information and change it if needed; they should be able to manipulate single devices or all devices attached to a machine in one action, depending on their needs; and a navigation bar needs to be present at all times so that users of the platform can navigate through it.

3.2.2 Administrator Requirements

Administrator of the platform has the simplest role in the system, although he has big responsibility. He is a central, impartial authority that makes sure that the system is running smoothly. Apart from regular responsibilities of a system administrator, he is *required to create accounts for manufacturers* and manage them.

This requirements comes from the fact that, in order to qualify as a manufacturer, you need to have a verified manufacturing company. In order to fully understand the need for this requirement, consider a scenario where everybody is allowed to create an account and pose as a manufacturer. This scenario introduces security concerns, such as a party acting as a manufacturer and assigning fictional devices to a customer, thus affecting customers view of the platform.

3.2.3 Manufacturer Requirements

Manufacturers of smart industrial machines have a central role in the whole ecosystem, they are the providers of the service of selling or lending machines. Therefore, their requirements are of great importance. They have all necessary information about machines they are manufacturing, thus, they should be the ones adding the machines to the system and assigning them to the customers. Following list will explain all of their requirements thoroughly:

1. **Create customer accounts:** Allowing anybody to create a customer account may create many empty accounts that are only taking space in the database. Furthermore, there is no use for them if they do not possess any machines. Therefore, when customers buy or lend a machine *for the first time*, their accounts need to be created. Depending on the needs of the customer, opening multiple accounts should be possible.
2. **Managing templates:** Almost every machine that is being produced is not one of a kind, it is made following a model. Manufacturers offer many different models to their customers, such as a smart crane pictured on figure 3.1 with model number K16052. Therefore, manufacturers need a way to define templates describing different models of their machines. These templates represent a simplified "digital twin" of a particular machine describing devices that are attached to it. This requirement consists of three smaller requirements:
 - (a) **Add templates:** Way to introduce new templates needs to be available.
 - (b) **Delete templates:** Deleting faulty or outdated templates needs to be available.
 - (c) **View templates:** Overview of all templates needs to be presented to manufacturer.
3. **Add machine through a template:** When the necessary templates are added to the system, manufacturers should be able to instantiate them to create a machine and add it to the system. Adding machines following a template reduces error and work needed when inputting data for all devices for a machine.
4. **Add custom machines:** As mentioned in Managing templates, almost every machine is made using a template. Although, some machines are custom made on a request of the customer. These machines

do not require a template because they are not being massively produced. Therefore, a way to add custom machines is required.

5. ***Remove and modify machines***: For any machine, regardless of how it was inputted into system, there needs to be a way to remove them or to modify their contents.
6. ***Manage machines and devices***: When all necessary machines are added to the system they need to be assigned to customers. It should only be possible to assign machines to the customers of a specific manufacturer (not the customers of different manufacturers). This requirement consists of three smaller requirements:
 - (a) ***Assign machine or device to a customer***: A way to assign a certain machine or device so it becomes visible to the customer needs to be available to the manufacturer.
 - (b) ***Remove assignment of machine or device***: Reverting the action of assigning a machine or device to customer needs to be provided in order to account for faulty assignments, or if the machine or device was lent for a certain period of time.
 - (c) ***View assignments***: In order to control which machines and devices are assigned to whom, an overview of assignments needs to be provided.

3.2.4 Customer Requirements

Customer requirements are slightly more complicated than manufacturers. Only one account (or a fixed number) of accounts is not necessarily sufficient since the buyer of the machine is rarely the user of the machine. Inside every company there is a hierarchy of responsibilities, where different people are responsible for a subset of machines that the customer company owns. Therefore, it is necessary for customer to be able to create as many accounts as his business requires (this number can be zero). These accounts that a customer creates shall be referred as users in the remainder of this thesis. Following text will list customer requirements, followed by requirements that user shares with customer:

1. ***Create user accounts***: As mentioned above, creation of user accounts is crucial in order to assign responsibilities for certain machines to several actors inside a company. These actors are some type of managers overlooking the machines they are responsible for.

2. ***Manage machines and devices***: When all necessary machines are added to the system by manufacturers and assigned to customers, they need to be assigned to users responsible for them. In certain cases (for example, in a small company), customer retains all responsibility for machines and do not assign them to anyone else. This requirement consists of three smaller requirements:
 - (a) ***Assign machine or device to user***: A way to assign a certain machine or device to the user needs to be available in order for the user to manage access to the machine or device.
 - (b) ***Remove assignment of machine or device***: Reverting the action of assigning a machine or device to the user needs to be provided in order to account for faulty assignments, or if the machine or device needs to be assigned to someone else.
 - (c) ***View assignments***: In order to control what machines or devices are assigned to whom, an overview of assignments needs to be provided.

Requirements listed above are only for the customer. User requirements are similar to customer with the exception of creating additional accounts and assigning machines. Following list describes additional requirements of customers which are also all requirements of users:

1. ***Allow access to machine or device***: Allowing customers to define who can access the data their machines produce is highly important. Customer should be able to define access for the whole machine or parts of the machine by controlling access to particular devices, such as a temperature sensor on crane previously described in section 3.1. Additionally, customer needs to be able to define duration and direction of the communication, explained bellow:
 - ***Duration***: Duration of allowed access is defined by start and end date.
 - ***Direction of communication***: Communication from customer device to the target means that only customer device can send data to the target while rejecting all requests from the target. Opposite direction means that the target can only send requests to the customer device, such request typically give instructions to the device. Allowing flexibility to define direction of communication is highly important to the customer.

2. ***Remove previously allowed access to machine or device:*** A way to revert action of allowing access to a machine or device needs to be provided.
3. ***View who has access to your machines:*** In order for customer to be fully aware of who has access to their machines, and overview needs to be provided.

Chapter 4

Prototype Implementation

For the implementation of contracting service I have used Django framework because it is powerful but also easy to use, and it provides a fixed structure to organize the project. Django framework provides built-in modules for most of the common functionalities with a good security architecture. Along with Django, I have used Bootstrap for styling since it is light-weight and simple.

In this chapter, I will present a brief description of technologies that I used followed by a brief overview of the whole project, showing how different parts fit together before digging into details.

4.1 Technologies Used

In the following text, I will describe two technologies that were used for this implementation in order to give justification of why they are a perfect fit for this project.

4.1.1 Django Framework

Django is a high-level Python Web framework that unlike a library, forces a structure to separate concerns and encourage rapid development and clean, pragmatic design. Structure of Django is influenced by Model-view-controller (MVC), which is a pattern used in software architecture for interactive software or application.

Model maintains the state of the application by holding the data. In web development, that data is stored in a database. Model abstracts the database typically using a Object-relational-mapper (ORM), which maps the data to objects (in case of Django, python objects) and vice versa. Model also takes care of any restrictions on data and takes care of validating, making sure

that correct data is stored.

Concern of the view is to generate user interface. In web applications, this means producing HTML typically using a *templating language*. Data is acquired from the Model (typically through Controller, which will be explained in further text) and mixed with HTML using templating language. View can format the data to suit the requirements of a particular view. In simple words, view controls what user sees.

Controller, in a simplest interpretation, controls which views to use based on user input and data from the models. In a broader interpretation, it handles business logic, using models and what rules they enforce.

Django Framework uses a variation on the traditional MVC pattern called Model-Template-View (MTV). Firstly, role of the Model is the same as in MVC, where Django abstracts the database using OBM, instead of tables you deal with classes and instances of data as objects. Secondly, the role of View in MVC is represented in Django using Templates and View. View in Django has a slightly different meaning than in MVC, it represents which data needs to be presented to a user, not necessarily how the data looks. Views get user input (HTTP request), then they access necessary models followed by processing of data, if necessary, and pass it to Templates. Role of Templates is to create the final HTML presented to a user using the data from Views. Finally, role of the Controller can be viewed as whole framework and URL-routing. URL-routing is achieved in Django with a simple 'urls.py' file, which links URLs to views.

Django Framework provides built-in solutions for most of the common tasks that almost every website has. Since this work is not about Django Framework I will only cite subset of parts used in this prototype:

1. ***User management***: including creating users, authenticating users, creating sessions, managing cookies, managing user groups and more.
2. ***Security mechanisms***: in order to prevent most common forms of cyberattacks such is cross-site request forgery.
3. ***Support for number of email back-ends***.
4. ***Efficient mechanism for creating forms from Models***.
5. ***Creating custom decorators to provide authorization for certain groups of users to a view***.

Most importantly, Django Framework has an enormous user base and excellent documentation making it very comfortable to work with. Having a big user base means that most of the questions you might have are already

answered and easily accessible. These are the reasons I have chosen Django for this solution.

4.1.2 Bootstrap

Bootstrap is an open source toolkit for developing with HTML, CSS, and JS. It is very light weight and perfect for applications that do not require a lot of styling. Bootstrap library is easily included in Django templates using just a `<link>` HTML tag.

Bootstrap library is only one CSS file providing many different classes for styling appealing to human visual cues (for example, red for danger, blue for default and more). It relies on a twelve column system for layout, which, if used correctly, brings responsiveness (for different screen sized) on its own. Bootstrap has a big community of users who share snippets of their code providing different functionalities fulfilling almost any need in modern web development.

4.2 Overview

This solution is a Web platform for managing small devices and controlling who has access to the data they generate. Most challenging part was understanding the users of the platform and their needs. Following section describes account types, which correspond to roles they have, followed by the brief description of main parts of the platform.

4.2.1 Account Types

In the Industrial Internet context, I have identified four different types of accounts, with their separate views of the platform depending on their role. Views of these accounts have a common core although they are suited for different purposes.

Firstly, there needs to be a central authority that distributes accounts to verified manufacturers of industrial machines. In order to receive manufacturer account you need to contact this central authority, which could be a standardization agency but it can be any impartial actor. When that authority verifies that your company is who they claim to be, several accounts can be issued depending on the number of departments that company has or any other criteria depending on the agreement with the manufacturer company. Since the only responsibility this authority has is creating accounts for manufacturers I have used standard Django admin page (and admin account) for

its implementation, stripped down to only basic functionality for managing accounts.

Secondly, manufacturers account is assigned to companies that produce smart machines. Their responsibility is to add devices (and update their information if necessary) to platform and assign them to machines (detailed description of devices will be given in subsection 4.2.2), which are abstract representation of real machine (such is a smart crane) and devices it possess. When the machine has been bought by the customer, manufacturer creates an account for customer (in case they do not possess one) and assigns it to them, giving them full control over who can communicate with that machine. In a case where a customer already has an account, it is customers responsibility to "subscribe" to manufacturer in order for manufacturer to see their account, this is done through a simple form which will be described in section 4.4. Manufacturer account will be described in detail in section 4.3.

Thirdly, customer account is assigned to owners of the smart machines. Their view is restricted to only devices and machines that they possess and they can manage policies (control access to devices) for those devices only. Along with the possibility of managing policies, customer account can create multiple user accounts (which represents people responsible for single machines or group of them) and assign machines or devices to them. In this way, customer account has a full overview of what policies are in place and who created them while also being able to add or remove faulty ones or general ones (like allowing customers work computer to access all the data or opening their data to a statistical agency). By being able to create user accounts customer can pass on the responsibility and fine tuning of policies for certain machines down the hierarchy of the company.

Finally, user account is assigned to people responsible for a subset of machines that a customer company possess. These accounts are restricted to only managing policies of the machines assigned to them without the possibility to create more accounts or manage devices in the system. Overview of accounts and their views are visualized in figure 4.1.

4.2.2 Device Management

As previously mentioned, manufacturers task is to add devices to the system, filling all necessary information about the device defined by LWM2M standard described in 2.2.1. This information consists of manufacturer name, model number, serial number, public key or identity of the device along with a descriptive name used to refer to it in the system. Public key or identity according to LWM2M standard could be a certificate, a pre-shared key, raw public key or nothing. Since all these, except last one, are similar and exist

| | Login/Logout/Update Account | Account Creation | Manage Devices | Manage Policies | Update Device Info |
|--------------|-----------------------------|------------------|----------------|-----------------|--------------------|
| Manufacturer | ✓ | ✓ | ✓ | | ✓ |
| Customer | ✓ | ✓ | ✓ | ✓ | |
| User | ✓ | | | ✓ | |

Figure 4.1: Account types and their views

in standard only to provide flexibility for the manufacturers, only pre-shared key is supported in this solution, although it can easily be extended.

Further following LWM2M standard, which has a client-server architecture, devices act as a client where my solution acts as a bootstrap server. When a device gets connected to a network, it communicates its IP address (in a form of IPv4 or IPv6) and FQDN or MSISDN to a bootstrap server which saves it for use when managing policies. Example of information implanted on a device is shown on figure 4.2.

| Res. ID | Name | Insta... | Value |
|---------|-------------------------------|----------|------------------------------------|
| 0 | LWM2M Server URI | | coap://bootstrap.example.com:5684/ |
| 1 | Bootstrap Server | | true |
| 2 | Security Mode | | 0 |
| 3 | Public Key or Identity | | [identity string] |
| 4 | Server Public Key or Identity | | [secret key data] |

Figure 4.2: Example device

Server URI represents the URI of my server, which enables a device to locate it. Only after the server has verified identity of the device, using pre-shared key, it can store devices IP address. In an event when a device is moved to another network (changing its IP address), a device communicates a new IP address which replaces the old one and all policies get updated with a new address.

4.2.3 Policy Management

Work described in section 2.3 is very extensive and is made with mobile communications in mind. Although, the idea of policy based communications is perfectly suitable for these purposes since allowing communication between a device and some host can be done in one simple HTTP request to Policy Database. Policy Database holds all policies describing who is allowed to communicate with whom. This database could be distributed or in one central place, although that is outside of the scope of this work, but it has one API that hides the way database is arranged and is the only way database can be accessed.

Only four different API requests are used in this work. These requests are for inserting, deleting, retrieving and updating firewall policies using Http Post for inserting and deleting, Http get for retrieving and Http put for updating. Inserting policies require following information and they are sufficient for the CES node to determine whether the communication should be established, whether the package should be dropped at the edge:

1. ***Target IP address and port***: target represents a host wanting to communicate with a device.
2. ***Source IP address and port***: representing a device, extracted from a database in my system.
3. ***Start and end of validity timestamps***: allowing scheduling of access to devices.
4. ***Direction of communication***: representing, whether only reporting of data from the device to the host is allowed, sending data to a device or both.
5. ***Transport protocol***: in this case CoAP is used, justification is provided in section 2.2.1.
6. ***FQDN or MSISDN***: helps speed up the search for receivers as mentioned in section 2.3 and is used for querying policies.

Direction of communication can be bi-directional or uni-directional (from host to device, or device to host). Bi-directional communication is a regular case, when a host sends a request to a device and gets data or acknowledgment in return. Uni-directional communication is useful in cases where the devices just need to send data (for example, temperature readings) to some external host who is subscribed to them, without a possibility of that host controlling the device (for example, sending request for shutdown). Other

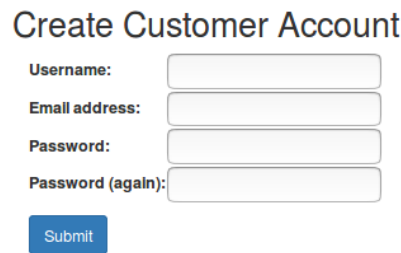
A web form titled "Create Customer Account". It contains four input fields: "Username:", "Email address:", "Password:", and "Password (again):". Each field is a simple rectangular box. Below the fields is a blue button with the text "Submit" in white.

Figure 4.3: Create Customer

way around, uni-directional communication from host to a device could be useful in rare cases where a host is allowed to request from a device to do a task, or send data to some other host. Therefore, all three options are supported in this work.

Deleting and retrieving policies is done using only FQDN or MSISDN (depending on, which one is available for a device), where updating is done the same way as for inserting only different HTTP verb is used. Graphical interface for managing policies will be described in the remainder of this section where all views from different accounts are explained.

4.3 Manufacturer

When manufacturer creates session by logging-in to the platform he is directed to a home page. Home page for manufacturers contains (along with navigation bar, which is always present) only a simple form to create customer account pictured on figure 4.3. This form, along with most of the forms in this solution, is created using Django ModelForm which created a form using database model of a user. When the form is submitted, customer account is made inactive and an automatic email is sent to the provided email address for verification. Automated emails are realized in this prototype using django built-in wrapper for python smtp lib module, using a google email account. Only after the customer follows the link in the email, their account is made active and they are prompted to change their password and provide additional information using update form pictured on figure 4.4. Update account form is available to all types of accounts, to encourage password change in order to promote security.

As previously mentioned, manufacturers task is to add devices to platform, which they can later assign to customers that have bought them. Adding of devices is made available in two ways: through "Manage Devices"

Update Account

First name:

Stephan

Last name:

Fillpov

Email address:

Kone.cranes@konecranes

New Password:

...

New Password (again):

...

Update

Figure 4.4: Update Account

page for custom made machines, or following a previously defined template. For the custom machines, first a machine needs to be created. Machine is an abstract concept that only has a name, which is used to organize devices. For example, AaltoCrane could be a machine and all sensors and actuators that are mounted on that crane are assigned to it. In the process of adding a device to the platform, a machine it belongs to needs to be specified. Note that in order to remove a machine from platform, its name needs to be typed in the text box to prevent accidental removal of machines. Two forms needed to complete this task are pictured on figure 4.5.

Add Machine

Name:

Add

Remove

Add Device

Name:

sensor1.crane1.aalto.fi

Manufacturer:

KoneCranes

ModelNumber:

123456

SerialNumber:

324-qw32-442

PublicKeyOrIdentity:

Machine:

Pond Cleaner

Garbage Bin

Pond Cleaner

Crane

Add

Figure 4.5: Adding machines and devices

Other way a manufacturer adds devices is through templates. Templates define how many devices a particular machine that a manufacturer produces has so that when a new machine is created, template can be instantiated to add a new machine. Creation and overview of templates is pictured on figure 4.6.

Clicking on the "instantiate" button for a particular machine opens a form bellow. This form asks for the name of the machine (which is supposed

Templates

| Template Name | Number of devices | |
|---------------|-------------------|------------------------------|
| Crane | 7 | <button>Instantiate</button> |
| Garbage Bin | 3 | <button>Instantiate</button> |
| Pond Cleaner | 4 | <button>Instantiate</button> |

Add/remove template: Add Template Remove Template

Figure 4.6: Creation and overview of Templates

to be descriptive, for example Aalto Industrial Campus Crane or similar), and information about devices attached to it. Number of devices in a form is equal to the number defined in the template. Form for a machine with three devices is pictured on figure 4.7.

Add machine via template

Machine name:

Device 1:

Device 2:

Device 3:

Add Machine

Figure 4.7: Instantiating a template with three devices

When the customer has an account and all devices are added to the platform and organized to machines, manufacturer needs to assign these devices to customers. Assigning devices to customers can be done on a device level or machine level as pictured on figure 4.8. No matter if the manufacturer is assigning devices or machines (group of devices) the task is done through a simple form and clicking on an appropriate button, either to add or remove a customer. Removing a customer makes sense in case when a machine or device has been rented or a mistake was made while adding. Removing of devices from a platform is also in a same view because it provides a good overview of which devices are attached to which machines.

In the figure 4.8, clicking on a device name navigates to a separate

Assign Devices

Crane

Crane

Aalto

Add Customer

Remove Customer

S3

Aalto

Add Customer

Remove Customer

Delete Device

S4

Aalto

Add Customer

Remove Customer

Delete Device

S5

KoneCranes

Add Customer

Remove Customer

Delete Device

BaseMark

MarkOy

StephanPlat

Garbage Bin

Garbage Bin

Aalto

Add Customer

Remove Customer

G1

Aalto

Add Customer

Remove Customer

Delete Device

G2

Aalto

Add Customer

Remove Customer

Delete Device

G3

Aalto

Add Customer

Remove Customer

Delete Device

Pond Cleaner

P1

Bla

Add Customer

Remove Customer

Delete Device

Figure 4.8: Device Management For Manufacturer

page where details about a device can be inspected, and changed if needed as pictured on figure 4.9. Fields in the form are pre-filled with current information about a device, so that typing mistakes or any small changes can be made easily.

Last element on a Manage Devices page is pictured on figure 4.10. This element only provides an overview of which users are assigned to which devices. It is intended to be used along with element pictured on figure 4.8 in order to check whether any mistakes were made and that everything is how it should be.

4.4 Customer

After logging-in to the platform, customer is directed to a home page where he can create additional user accounts. Creation of user accounts is slightly different from customer accounts as pictured on figure 4.11. In the form, for convenience sake, customer can assign devices to a user while creating the account using a multiple choice select field. In case when the account is created for just one device or a small set of devices, this can greatly reduce the number of clicks and queries to the database to complete a task. As with customer accounts, user accounts also need email verification.

Device Details

Name:

S3

Manufacturer:

ModelNumber:

SerialNumber:

PublicKeyOrIdentity:

1

Machine:

Crane

Update

Garbage Bin

Pond Cleaner

Crane

Figure 4.9: Update Device Information

Device assignment overview

| |
|---------------------|
| Crane |
| S3 -- |
| S4 -- Kone, |
| S5 -- Kone, |
| Garbage Bin |
| G1 -- |
| G2 -- |
| G3 -- BaseMark, |
| Pond Cleaner |
| P1 -- Relja, Aalto, |

Figure 4.10: Device Assignment Overview

One very simple but necessary element pictured on figure 4.12 allows a customer to subscribe to the manufacturer and in that way be visible to manufacturer when assigning devices. Customer only needs to type the name of the manufacturer (that manufacturer communicated to customer in the process of buying the machine) and submit.

On "Device Management" page, customer can assign devices to users. His view is restricted only to devices that are previously assigned to him by the manufacturer, and choice of users is restricted to the ones he created (which represent users he is responsible for). Interface for doing this resembles one pictured in figure 4.8 with two exceptions: customer is not allowed to change details about devices; and user is not allowed to delete devices from the system. In the case when a machine is no longer used (not needed anymore or replaced with a new one), manufacturer needs to be contacted directly in order to remove the machine from platform. This is because customers can not be allowed to add devices, and if they were allowed to remove devices,

Create User Account

Devices:

S4, G2, G3

Username:

S3

Email address:

S4 ✓

Password:

S5

Password (again):

G1

G2 ✓

G3 ✓

P1

Submit

Figure 4.11: User Account Creation

Subscribe to Manufacturer

Manufacturer Name:

Manufacturer Name

Submit

Figure 4.12: Inform Manufacturer that you are a customer

they could mistakenly remove the device without the possibility of adding it back. In industrial case, these machines are expensive, large and robust, therefore, they do not need to be removed frequently, which is why I opted for this solution.

As previously mentioned, only customers and users are allowed to control who can communicate with their devices and they control that through policies. "Manage Policies" page contains two elements, one for adding policies, other for removing them and to give overview of existing ones. Customer view is restricted to only devices that are assigned to him (as in case of managing devices) and he can add policies to Policy database using a group of forms pictured on figure 4.13. Customer only needs to specify IP address (both IPv4 and IPv6 supported) of a host, direction of communication (as explained in 4.2.3), start and end date of policy validity, and click on submit policy. By submitting a policy, customer has allowed a host with specified IP address to interact with a device or all devices that are part of a machine.

Add Policies

Crane

Crane

Target IP

Bi-directional

08/07/2017 - 08/24/2017

Submit Policy

S4

S4

Target IP

Host -> Device

08/07/2017 - 08/23/2017

Submit Policy

S5

S5

Target IP

Communication Direction

08/23/2017

08/23/2017

Apply

Cancel

Garbage Bin

Garbage Bin

Target IP

Communication Direction

G2

G2

Target IP

Communication Direction

G3

G3

Target IP

Communication Direction

Pond Cleaner

P1

P1

Target IP

Communication Direction

Figure 4.13: Policy Management

Second element on a "Manage Policies" page contains overview of all policies related to devices that a customer has. In the same element, customer can remove policies that are mistakenly made or if the circumstances changed since the policy was put in place.

| | | | |
|----|---|--|--------|
| S3 | computer1.home.fi -- 2001:708:150:10::a1c3 -- (08/15/2017 - 08/16/2017) | | Remove |
| | computer2.home.fi -- 2001:708:151:10::a1c3 -- (07/03/2017 - 09/15/2017) | | Remove |
| | | | |
| S4 | computer2.home.fi -- 2001:708:151:10::a1c3 -- (07/03/2017 - 09/15/2017) | | Remove |
| | | | |

Figure 4.14: Overview of policies

4.5 User

Simplest account type is user account. User account can only update information about his account and manage policies for the devices he was assigned to by the customer. "Manage Policies" page for user only contains elements pictured in figures 4.13 and 4.14.

Chapter 5

Evaluation and Discussion

In the requirements chapter, the context for which this thesis is tailored for is described. This context described actors involved and how are they using these machines. From the context, a list of concrete requirements have been drawn and described in the remainder of requirements chapter. Following these requirements, I have implemented a prototype which aims to fulfill these requirements.

In the following sections, I am evaluating the implementation described in chapter 4. Furthermore, I will discuss shortcomings and possible improvements to my solution. This evaluation is done by addressing every requirement described in chapter 3, consequently the current chapter will have similar structure. The general requirements will be addressed, followed by administrator, manufacturer and customer requirements.

5.1 Evaluation of General Requirements

General requirements are not tied to any specific role in the system, they apply to all users. These requirements are grouped into three categories. First group is related to security, second regarding security, In the following section, requirements regarding security, user experience and functional requirements related to all roles in the system are evaluated. Furthermore, shortcomings are discussed and possible alternate solutions are described.

5.1.1 Security

Securing a web platform is not an easy task. As the security measures are evolving to fill security gaps, new methods on how to breach them are also emerging. Thus, securing a web platform is a process and it constantly

needs to be improved. Following list will discuss every security requirement described in section 3.2.1.1 and how the implementation has met them:

1. ***Only authenticated users can access platform:*** As previously mentioned in section 4.1, for this implementation a Django Framework was used. Django provides built-in decorator, named `@login_required`, to restrict access to certain views. To make sure that only authenticated users can access the platform, every view has been secured with this decorator.
2. ***Platform needs be to robust against common cyberattacks:*** One big advantage of using Django Framework are the built-in protection mechanisms against cyberattacks. Protection against most common cyberattacks is discussed in the following list.
 - (a) ***Cross-site scripting (XSS):*** XSS attacks allow a user to inject client side scripts into the browsers of other users. Protection against this is ensured by using django templates, which escape specific characters which are particularly dangerous to HTML. This protection is not completely foolproof since there are cases when it fails. Such examples are well documented in django documentation, and all `<style>` tags used in this implementation are checked against them.
 - (b) ***Cross site request forgery (CSRF):*** CSRF attacks allow a malicious user to execute actions using the credentials of another user without that user's knowledge or consent. CSRF protection works by checking for a secret in each POST request. This ensures that a malicious user can not simply "replay" a form POST to your website and have another logged in user unwittingly submit that form. Since the malicious user does not know the secret, which is stored in a cookie, he can not execute this request. CSRF protection in django is done by simply putting a `% csrf_token %` tag in every form in a template and securing corresponding view with `@csrf_protect` built-in decorator. The `% csrf_token %` creates a hidden input field containing a secret which is then passed to the corresponding view.
 - (c) ***Clickjacking:*** Clickjacking is a type of attack where a malicious site wraps another site in a frame. In that way, an unsuspecting user is tricked into performing unintended actions on the target site. In order to protect against this attack, rendering in a frame has been disabled using django X-Frame-Options middleware.

- (d) **SQL injection:** SQL injection is a type of attack where a malicious user is able to execute arbitrary SQL code on a database. Such attacks are usually prevented by sanitizing any input coming from an user. Using Django Framework, this is prevented by using querysets and in that way an underlying database driver will escape the resulting SQL. Django also provides developers a way to write raw queries which are vulnerable to SQL injection. In order to prevent SQL injection, raw queries are not used.
 - (e) **Denial of service (DOS):** Not much can be done to prevent denial of service attacks at this stage, but rather during deployment of the platform on a server. By not allowing everybody to create accounts on this platform, effect of DOS attack is reduced but it is not eliminated.
3. **Only authorized people can manipulate machines or devices:** Django Framework provides support to write custom decorators, to secure access to views on almost any criteria. For this requirement, three different decorators have been made for each account type, namely manufacturer, customer and user. These decorators have been used for each view that belongs to manufacturer, customer or user accordingly. Furthermore, identity of a user is being checked every time a database is contacted in order to make sure that he has required access rights. In the previous point, protection against cyberattacks are discussed, making sure that user identity was not tampered with.
 4. **Users personal information can only be changed by them:** Similar to the previous point, user identity is determined every time user wants to see or change its information. And the validity of their identity is ensured by protecting against common cyberattacks.

Along with security measures described in the list above, HTTPS protocol is used instead of HTTP. This functionality is enabled by configuring Django settings file to redirect all requests over HTTP to HTTPS and also by configuring a server where this platform is hosted.

5.1.2 User Experience

Requirements related to user experience are more of a guidelines than actual requirements, nonetheless, every well designed user interface is following them. Therefore, it is very important that they are satisfied to a logical extent. Following list will briefly discuss how the implementation described in chapter 4 has followed Nielsen et. al. [10] ten most important heuristics:

1. ***Visibility of system status***: For almost every action that user performs, a message informing user about the success of the action is issued in form of a pop-up window. In case of removing a machine (or device) or a template, this message is not issued since the disappearance of the machine or template from the overview is evident.
2. ***Match between system and the real world***: Terms used in the system correspond to real world terms, few examples include a machine, device, manufacturer and customer.
3. ***User control and freedom***: Actions available in this system are easily reversible, such actions include adding a machine, submitting a policy, and adding a template.
4. ***Consistency and standards***: All terms (such are machine, device, and template) correspond to real world terms and are used consistently throughout the platform.
5. ***Error prevention***: This requirement is very hard to fulfill, since it is hard to predict kind of mistakes people make. Simple measures are taken in order to prevent mistakes since this platform is not intended for computer illiterate people. Preventing accidental removal of machines and templates is assured by forcing an user to type in the name of the machine or template before deleting them.
6. ***Recognition rather than recall***: Almost all actions in the system require user to select options from the list of available ones instead of typing them in, reducing memory load of the user. Exception are the removal of machine or template because of the reasons described in the previous point.
7. ***Flexibility and efficiency of use***: Tailoring of frequent actions is made available through templates, where manufacturers can simplify the way they add machines in the future by defining a template of a particular machine and instantiating them.
8. ***Aesthetic and minimalist design***: All elements described in chapter 4 are necessary to provide desired functionality. Colors and shapes are designed to appeal to human visual cues.
9. ***Help users recognize, diagnose, and recover from errors***: Error messages are presented to a user in plain text, no codes are being used. In order to do so, custom pages for most common http errors are provided. Such errors include “404 not found” and “500 internal server error”.

10. ***Help and documentation:*** Home page of the platform, briefly describes how to use the platform.

5.1.3 Functional Requirements

List of functional requirements for all roles is a rather short list, although, still important for this platform to be usable. Following list will describe how my implementation has fulfilled these requirements:

1. ***Users should be able to view their personal information and change it if needed:*** This functionality is provided by using a form shown on figure 4.4.
2. ***Manipulating single devices or all devices attached to a machine in one action:*** Forms shown on figure 5 and 5 allow assigning machines and submitting policies for all devices of a machine or single devices in one action, respectively.
3. ***Navigation bar needs to be present:*** Standard navigational bar on the top of the screen, spanning the width of the screen is provided to navigate the website.

5.2 Evaluation of Administrator Requirements

Administrator of the system has a responsibility to overlook the system and make sure that it operates smoothly. Administrator is required to create manufacturer accounts as mentioned in chapter 3. For the implementation of the administrator view, a django built-in administrator page have been used. This page provides many features, such as managing the user groups, and creating and deleting different types of accounts. For this work, I have reduced functionalities of this page to only be able to manage manufacturer accounts. Managing includes creating, deleting, modifying contents of the manufacturer accounts.

5.3 Evaluation of Manufacturer Requirements

As previously described in chapter 4, manufacturers have their own view of the platform. In the following list, manufacturer requirements are discussed, providing justification on how the implementation is meeting them, while pointing out shortcomings and possible alternate solutions.

1. **Create customer accounts:** Fulfilling this requirement was simple. Form pictured on figure 4.3 allows a manufacturer to create customer accounts. Creating multiple accounts with the same username is not possible due to the database restrictions.
2. **Managing templates:** Only manufacturer account has access to the "Manage Templates" page since they are the only ones who add machines or devices. This was ensured using custom made decorators.
 - (a) **Add templates:** Creating a new template is available through form shown in figure 4.6. This is a minimalistic interface which allows the manufacturer to add templates of machines with corresponding number of devices. This solution lacks the sufficient fields to describe the machine. In other words, it is not very expressive, since it only requires information about the name of the machine and number of the devices attached to it. Whereas, real machines are usually comprised of multiple parts and each part has different types of devices attached. It would have been more meaningful to have detailed description of machines, its parts, and types of devices. However, this detailed description of machines require comprehensive study of "digital twins", which is not the core of this work.
 - (b) **Delete templates:** Deleting templates is available through same form as for adding templates. Manufacturer is required to input all information about the template and click on a button "delete". It was implemented this way in order to minimize risk of deleting a template by accident.
 - (c) **View templates:** View of all available templates is also shown on the same figure. For this overview of templates, a table is used.
3. **Add machine through a template:** On a figure 4.6, a table with existing templates is shown. In the third column from the left, a button for each template is provided to instantiate a template. Clicking on a button "instantiate" opens a form to add a machine following that template as shown on figure 4.7.
4. **Add custom machines:** Adding custom machines is made available through "manage devices" page through forms shown on figure 4.5. These forms are on the "manage devices" page because they do not take much space and they are used frequently when managing devices.
5. **Manage machines and devices:** Managing devices for manufacturer is done through "manage devices" page.

- (a) ***Assign machine or device to a customer***: Assigning machines or devices to a customer is done through forms shown on figure 4.8. This implementation gives good overview of devices and to what machine are they attached to, although each assignment requires a separate call to the database. Alternate solution that would allow multiple assignments to be done in one call to the database would not give overview of devices. This solution would list all customers and provide a multiple check list to choose which devices to assign to them. Current solution was chosen because it is simpler to use.
 - (b) ***Remove assignment of machine or device***: Removing assignment is done through the same form as for assigning, just by clicking on "remove customer" instead of "add customer".
 - (c) ***View assignments***: Overview of assignments is shown on figure 4.10. For each device, assigned customers are listed.
6. ***Remove machines or devices***: Removing machines from the system is done through the form shown on figure 4.5. For the same reason as for removing templates, manufacturer is required to input the name of the machine in order to delete it. Removing devices is done through forms shown on figure 4.8 by clicking on "remove customer" button. Figure 4.8 provides an overview of all devices, thus, this position of the button to remove customer is logical.
 7. ***Modify machines***: Modifying machines is done one device at a time. Clicking on a device name in the form shown on figure 4.8 opens a separate page where device information can be changed.

5.4 Evaluation of Customer Requirements

As mentioned in chapter 3, having only one type of account for customer is not sufficient. Thus, in this work two types of customer accounts are created. These accounts are named customer and user accounts. Customer account is created by manufacturer, and user accounts are created by customer in order to assign responsibilities for different machines down the company hierarchy. In other words, user accounts are created for managers inside the customer company who are responsible for certain machines are people working with those machines. Following text will evaluate and discuss customer requirements arranged in two lists, exclusively customer requirements and requirements related to both customer and user, respectively.

1. **Create user accounts:** Creating user accounts is made available to manufacturer through a form shown on figure 4.11. This form has been created using Django ModelForm module, which creates forms directly from the models in the database, alike most of the forms used in this implementation. In this case, that model was User.
2. **Manage machines and devices:** Implementation meets this requirement in a similar way as for the manufacturer. Differences between implementation for manufacturer and customer will be described below:
 - (a) **Assign machine or device to user:** Assigning machines and devices is done through the same form as one shown on figure 4.8, with two exceptions. These exceptions are: links to access device information and change it are disabled, and buttons to remove devices from the system are removed. Alternate solution for manufacturer view of the same functionality applies in this case too.
 - (b) **Remove assignment of machine or device:** In order to remove assignment, customer account performs the same action as for assigning, just by using "remove customer" instead of "add customer" button.
 - (c) **View assignments:** Full overview of assignments for customer is the same as for manufacturer. This overview is shown on figure 4.10

As previously mentioned, above list evaluates and discusses exclusively customer requirements. Following list contains evaluation and discussion of requirements related to both customers and users.

1. **Allow access to machine or device:** Who can access a device is described by a policy as explained in section 2.3. Inserting or removing these policies from a policy database allows or denies a certain host to access a particular device. Forms used for manipulating these policies are shown on figure 4.13. Policies allow high level of customization of the type of access that is allowed, although for this work only required protocol, direction of communication and duration of communication is used.
 - **Duration:** In order to define a period of duration of a policy, slightly modified date range picker ¹ is used as shown on figure

¹<http://www.daterangepicker.com/>

4.13.

- ***Direction of communication:*** All directions of communication are supported. They are defined in select field by choosing appropriate direction (Host -> Device, Device -> Host or Bi-directional).
2. ***View who has access to your machines:*** Overview of policies inside a policy database are shown on figure 4.14.
 3. ***Remove previously allowed access to machine or device:*** Overview of policies shown on figure 4.14 contains a button to remove a policy.

Chapter 6

Conclusion and Future Work

Bibliography

- [1] ALAYA, M. B., BANOUAR, Y., MONTEIL, T., CHASSOT, C., AND DRIRA, K. OM2M : Extensible ETSI-compliant M2M service platform with self-configuration capability. *Procedia - Procedia Computer Science* 32 (2014), 1079–1086.
- [2] BANDYOPADHYAY, D., AND SEN, J. Internet of things: Applications and challenges in technology and standardization. *Wireless Personal Communications* 58, 1 (2011), 49–69.
- [3] BRASSER, F., MAHJOUB, B. E., SADEGHI, A.-R., WACHSMANN, C., AND KOEBERL, P. TyTAN: Tiny Trust Anchor for Tiny Devices.
- [4] CONTROL, E., AND SECURITY, S. Detecting Industrial Control Malware Using Automated PLC Code Analytics.
- [5] DEFRAWY, K. E., PERITO, D., AND TSUDIK, G. SMART : Secure and Minimal Architecture for (Establishing a Dynamic) Root of Trust.
- [6] ETSI. Machine-to-machine communications (m2m); functional architecture, 2013.
- [7] GRIECO, L. A., ALAYA, M. B., MONTEIL, T., DRIRA, K., AND BARI, P. Architecting Information Centric ETSI-M2M systems. 211–214.
- [8] KANTOLA, R., AND BEIJAR, N. Policy Based Communications for 5G Mobile with Customer Edge Switching.
- [9] KOEBERL, P., SCHULZ, S., AND SADEGHI, A.-R. TrustLite : A Security Architecture for Tiny Embedded Devices.
- [10] NIELSEN, J. 10 usability heuristics for user interface design. *Fremont: Nielsen Norman Group.[Consult. 20 maio 2014]. Disponível na Internet* (1995).

- [11] NOORMAN, J., AGTEN, P., DANIELS, W., AND STRACKX, R. Sancus : Low-cost trustworthy extensible networked devices with a zero-software Trusted Computing Base.
- [12] PALATTELLA, M. R., DOHLER, M., GRIECO, A., RIZZO, G., TORSNER, J., ENGEL, T., AND LADID, L. Internet of Things in the 5G Era: Enablers, Architecture, and Business Models. 510–527.
- [13] PARK, H., KIM, H., JOO, H., AND SONG, J. Recent advancements in the Internet-of-Things related standards : A oneM2M perspective. *ICT Express* 2, 3 (2016), 126–129.
- [14] RAO, S., CHENDANDA, D., DESHPANDE, C., AND LAKKUNDI, V. Implementing LWM2M in Constrained IoT Devices. 52–57.
- [15] RAOUL STRACKX, FRANK PIESSENS, B. P. Efficient isolation of trusted subsystems in embedded systems.
- [16] SADEGHI, A., WACHSMANN, C., AND WAIDNER, M. Security and privacy challenges in industrial internet of things. *Proceedings of the 52nd* (2015).

Appendix A

Appendix A