

**Diary Thesis at Huawei Technologies Oy (Finland) Co. Ltd**

Andrey Krendzel

Bachelor's Thesis  
Degree Programme in BITE  
2022



<b>Author(s)</b> Andrey Krendzel	
<b>Degree programme</b> Business IT	
<b>Report/thesis title</b> Diary Thesis at Huawei Technologies Oy (Finland) Co. Ltd.	<b>Number of pages and appendix pages</b> <b>88</b>
<p>In this thesis the author explores the tasks he completely daily during his internship at Huawei. He also describes the issues faced on a day-to-day basis, and how he solved them. This thesis comprises a total of 40 workdays, and 9 weekly observations.</p> <p>In weekly observations, the author explores theoretical topics related to the daily practical tasks. Such as web-development related concepts, IT security concepts, Java and JavaScript concepts.</p> <p>Of course, naturally the thesis also has an introduction where the author describes the company he works for, the skills he needs currently and will need further in his position, and the professional sources he is going to be using. The author has a framework chapter where he explains the development objectives as well as the analysis of his current work.</p>	
<b>Keywords</b> Spring Boot, Fullstack, C, LiteOS, keybox	

## Table of contents

1	Introduction .....	5
2	Framework .....	9
2.1	Analysis of your current work .....	9
2.2	Evaluation of my current skills and stage of professional development .....	11
2.3	Interest groups at work.....	13
2.4	Interaction skills at work .....	14
3	Diary entries .....	15
3.1	How I organize and order my objectives and tasks.....	15
3.2	Observation week 1 .....	15
	Tuesday 1 <sup>st</sup> February 2022.....	15
	Wednesday 2 <sup>nd</sup> February 2022.....	16
	Thursday 3 <sup>rd</sup> February 2022 .....	17
	Friday 4 <sup>th</sup> February 2022 .....	19
	Weekly observations week 1 (calendar week 5) .....	20
3.3	Observation week 2 .....	23
	Monday 7 <sup>th</sup> February, 2022. ....	23
	Tuesday 8 <sup>th</sup> February, 2022. ....	24
	Wednesday 9 <sup>th</sup> February, 2022. ....	26
	Thursday 10 <sup>th</sup> February, 2022. ....	27
	Friday 11 <sup>th</sup> February, 2022. ....	28
	Weekly observations week 2 (calendar week 6) .....	29
3.4	Observation week 3 .....	31
	Monday 14 <sup>th</sup> February, 2022. ....	31
	Tuesday 15 <sup>th</sup> February, 2022 .....	32
	Wednesday 16 <sup>th</sup> February, 2022. ....	33
	Thursday February 17 <sup>th</sup> , 2022. ....	34
	Friday February 18 <sup>th</sup> , 2022. ....	35
	Weekly observations week 3 (calendar week 7) .....	36
3.5	Observation week 4 .....	38
	Monday February 21 <sup>st</sup> , 2022.....	38
	Tuesday February 22 <sup>nd</sup> , 2022.....	39
	Wednesday February 23 <sup>rd</sup> , 2022. ....	40
	Thursday February 24 <sup>th</sup> , 2022. ....	41
	Friday February 25 <sup>th</sup> , 2022. ....	42
	Weekly observations week 4 (Calendar week 8) .....	44
3.6	Observation week 5 .....	45
	Monday February 28 <sup>th</sup> , 2022. ....	45

Tuesday March 1st, 2022.....	48
Wednesday March 2nd, 2022.....	50
Thursday March 3 <sup>rd</sup> , 2022. ....	51
Friday March 4 <sup>th</sup> , 2022. ....	52
Weekly observations week 5 (Calendar week 9) .....	53
3.7 Observation week 6 .....	55
Monday March 7 <sup>th</sup> , 2022.....	55
Tuesday March 8 <sup>th</sup> , 2022.....	56
Wednesday March 9 <sup>th</sup> , 2022.....	57
Thursday March 10 <sup>th</sup> , 2022.....	58
Friday March 11th, 2022. ....	59
Weekly observations week 6 (Calendar week 10) .....	60
3.8 Observation week 7 .....	62
Monday March 14th, 2022.....	62
Tuesday March 15th, 2022.....	63
Wednesday March 16 <sup>th</sup> , 2022.....	65
Thursday March 17 <sup>th</sup> , 2022.....	67
Friday March 18 <sup>th</sup> , 2022.....	69
Weekly observations week 7 (Calendar week 11) .....	71
3.9 Observation week 8 .....	72
Monday March 21 <sup>st</sup> , 2022.....	72
Tuesday March 22 <sup>nd</sup> , 2022. ....	73
Wednesday March 23 <sup>rd</sup> , 2022.....	74
Thursday March 24 <sup>th</sup> , 2022.....	75
Friday March 25 <sup>th</sup> , 2022.....	76
Weekly observations week 8 (Calendar week 12) .....	79
3.10 Observation week 9 .....	81
Monday March 28 <sup>th</sup> , 2022.....	81
4 Discussion and conclusions.....	82
References .....	85

# 1 Introduction

During the Autumn of 2021 I had an opportunity to intern for Huawei, a Chinese company that is the leading provider of ICT infrastructure and smart devices all around the world. During my internship I have learned to put my study skills in practice, as well as to learn about the work-life environment in a well-established company.

Naturally, after having this experience, I thought it would be good to reflect further on the working process in this company. Luckily for me, the Diary Thesis provides me with just that: an opportunity to reflect better on my day-to-day as an intern. After suggesting this idea to my supervisor and the line manager, my contract was extended for 2 more months to write this thesis.

My internship is taking place in Helsinki System Security Lab, established in 2012. It is a research unit and as such the practices here are slightly different from what we're taught about software development companies.

Also, because the level of the employees in here is considerably higher than mine, most being seniors and with higher education (Master's, PhD), my tasks and tempo was lighter than theirs. However, I thought it was good that I could see how more experienced developers & researchers work and try to learn for my future work positions.

## **Observation period for thesis (start and end dates)**

01.02.2022 – 31.03.2022

## **Central content and sources, professional literature**

A considerable part of my learning will involve using Linux and C/C++. My supervisor has provided me with two books to deepen into the knowledge of those two fields. One, C++ Primer (Lippman & Lajoie, 1998), is considered to be a foundational book in C++ by other colleagues as well. The other book, Linux Application Development (Johnson & Troan, 2005), was also very interesting and informative.

I have also used Tutorialspoint (2022) to learn more about C concepts such as memory allocation, pointers, enums and structs. I will use it further to learn more about C sockets interface. For Java I will use online resources such as Baeldung (2022). There are several references to different resources from Baeldung along this thesis.

For JavaScript projects, that are not strictly necessary but that I think improve the project slightly, I will use Full Stack Open from Helsingin Yliopisto (2021), Mozilla Developer Network Webdocs (2005-

2022). For React code, I will use the official React documentation (2022) and React Native (2022) documentation.

### **Skills needed in my work duties**

I need some basic skills, such as understanding the current state of project development by reading some papers and talking to the coordinators, as well as designing the project architecture based on current state and established objectives. To better understand the current state and architecture, I will need to draw some diagrams in Draw.io.

From coding skills, I will need some basic C programming and library knowledge (file processing, sockets, etc). I will need some Java programming knowledge, specifically when it comes to server programming (Spring Boot). To fully program the server, I will also need basic SQL knowledge (a MySQL database should be attached to the server).

I will need basic knowledge of LiteOS (components, file structure, building process). I will need Javascript and Javascript frameworks knowledge: React, React Native, NodeJS. I will also need to know how to document my progress in reports, and how to research information: basic researching skills.

### **The company: basic information**

Company: Huawei Technologies Oy (Finland) Co. Ltd.

Industry: Information and communications technology (ICT) infrastructure and smart devices.

Size: Big – 100 000+ employees.

Annual turnover: 891.4 CNY billion.

### **The company: description and work environment**

Huawei (2022) is the leading global provider of information and communications technology (ICT) infrastructure and smart devices. The company has approximately 197 000 employees. It is heavily invested in R&D and had 100 000+ patents by the end of 2020.

The company has published 590+ journal and conference papers in high-impact channels (Huawei, 2022). It has members on the board or on the executive committee in 3GPP, ETSI, IETF, IIC, IEEE SA, the Linux Foundation, CCSA, All, TM Forum, WFA, WWRF, CNCF, OpenInfra (formerly OpenStack), LFN, LFDL, IFAA, GP, CUVA, VRIF, and BBF (Huawei, 2020).

Huawei contributes to innovative technologies: they create resources to help implement industry projects including 5G, AI, industrial Internet, video, broad Internet of Vehicles (IoV), and intelligent computing.

It also contributes to more than 200 standards organizations on an ongoing basis, including 3GPP, ITU, IEEE, IETF, ETSI. They are working with industry partners to develop and release international standards like ITU-T H.266 and MPEG-5 EVC, build up ecosystems, and improve user experience (Huawei, 2020).

Huawei continues to promote global open-source projects and they are an active contributor to leading open-source communities. Huawei is continuing to increase its contributions to leading global open source communities like Linux, Apache, Kubernetes, CNCF, OpenInfra, OCI, ONAP, OPNFV, Akraino, Acumos, Hadoop, and Linaro. Huawei is among the top 10 code contributors in these communities worldwide and remains the No. 1 code contributor in Asia Pacific (Huawei, 2022).

The company I work for also contributes to standards in fields like AI, consumer technology, and smart vehicles. By working with these new and innovative industries, they create channels for international conversations on standards, helping industries go digital.

Personally in our department, Hesinki System Security Lab, the work was focused on security of the newer operating systems developed by Huawei: LiteOS and HarmonyOS, as well as the operating systems themselves. Our department was established in 2012 and is an R&D department. I got a chance to work with LiteOS first-hand and see the demos for different components.

### **Key professional concepts**

**Sockets Interface:** The Berkeley Socket API was designed as a gateway to multiple protocols. It is much easier than inventing a new interface for every new protocol. Linux uses the socket API for many protocols, including TCP/IP. Sockets are implemented through the file abstraction. They are created through `socket()` system call, which returns a file descriptor. Once the socket has been properly initialized, that file descriptor may be used for `read()` and `write()` requests, like any other file descriptor. When a process is finished with a socket, it should be `close()`ed to free the resources associated with it (Johnson & Troan, 2005).

**RESTful service:** Representational state transfer (REST) is a software architectural style that was created to guide the design and development of the architecture for the World Wide Web. REST defines a set of constraints for how the architecture of an Internet-scale distributed hypermedia system,

such as the Web, should behave. The REST architectural style emphasizes the scalability of interactions between components, uniform interfaces, independent deployment of components, and the creation of a layered architecture to facilitate caching components to reduce user-perceived latency, enforce security, and encapsulate legacy systems (Fielding 2000).



## 2 Framework

### 2.1 Analysis of your current work

Here I explain some of the tasks that I do during my work, as well as their detailed description.

*Understanding (at least partially) how keybox provisioning/attestation works* on a bigger scale (my work is just a small part of it). Specifically, to read a 30-page paper on how the project is organized, take notes. Draw some schemes in my notebook. Draw better schemes in draw.io. Ask the project coordinator questions about things I don't understand.

*Designing the architecture* of the server, and connection between server and LiteOS device. The server should be in Java Spring Boot that would have REST endpoints. It should be able to generate keyboxes, offer them for downloading, show details in JSON format both for individual keyboxes and lists of keyboxes.

It should also be able to fetch keyboxes based on their used status, device, id and quantity. Server should manage quantity and status of keyboxes based on their use. It should take modifications to the keyboxes, should be able to receive requests to generate new keyboxes, or to delete existing ones.

Server should be integrated with SQL database (MySQL or PostGRE) where the keyboxes are stored. It will be integrated with a React front-end that can be used for administering the keyboxes through requests, as well as a React Native mobile app to be able to control the server from a mobile device.

Server will be integrated with a C client that should run on a Linux based operating system (LiteOS). Because I do not have the knowledge to activate the networking functionality in LiteOS, a third intermediary program will be written and maintained that connects together the server, C client, and LiteOS. The program will be called `host_pipe`.

LiteOS is a Linux-based OS that can run C code, but without networking functionality it cannot use the sockets interface to send REST requests to the server. However, it can transmit information using a UART port interface, and a colleague at work has already experimented with this approach. I will ask him for tutoring to be able to transmit the keyboxes to the LiteOS device without using sockets.

This will be done through a third program (`host_pipe.cpp`), written in C++ and which receives the keybox through sockets interface from the server, saves it locally, transports it to LiteOS using UART port (also using sockets interface), and then confirms, if necessary, the keyboxes' use by sending a POST request to the server, also using sockets.

*Coding in different languages* (Java, C, C++, Javascript, basic SQL statements). Server: Java, Front-end: Javascript, LiteOS operations: C, Hostpipe (intermediary app): C++.

*Connecting it all together in a fullstack fashion.* It will be slightly challenging, Database – Server – Front-end, Database – Server – Hostpipe – LiteOS. But in a sense, it is very similar to most fullstack projects.

*Documenting* and making specifications of how my software works. Sequence diagrams and flowcharts were and will be done using draw.io for better visualization. Documentation is done in short reports. Specification is written in a longer report-style document. Specification will focus mostly on the server, as that is the part most relevant to the project

### **Skills needed in my tasks**

As explained in chapter 1 (p. 2), I will need a combination of basic skills such as understanding the current state of project development, and coding skills such as basic C programming, Java programming.

Here would be those skills summarized in a list:

- Understanding the current state of project development by reading a paper provided by project supervisor.
- Designing the project architecture based on current state and established objectives
- Drawing a diagram of the project architecture in Draw.io.
- Basic C programming and library knowledge (file processing, sockets, etc).
- Java programming knowledge, server programming knowledge (Spring Boot).
- Basic SQL knowledge (a MySQL database was used for the server side).
- Basic knowledge of LiteOS (components, file structure, building process).
- Javascript and javascript frameworks knowledge: React, React Native, NodeJs.
- Drawing sequence diagrams.
- Documenting my progress in reports.
- Researching skills.

### **Skills I have obtained so far**

So far, from technical skills, I have learned a lot about C, specially involving networking functionality. I have learned about different data types as, coming mostly from Javascript background, I am used to using strings, not bytes or integers.

Also, I have learned about signed/unsigned data types, pointers, memory allocation and such. I have learned about using Java Spring Boot in work-based projects. Previously I have known how to create basic server connected to Thymeleaf template engine. Now I have learned how to develop a more complex server not linked to a template engine.

In the future I expect to learn more about the soft skills required to work in a big company, as well as more about C++, C, React and React Native.

### **Information I need to understand**

I need to learn more C. C is an old language and using high-level frameworks such as Spring Boot or Javascript, solves many of the issues which in C you have to solve yourself. Integer/byte manipulation were specifically hard issues for me.

I need more information on how the keyboxes are attested and further used on the device, as well as how they are generated. However, that is security information, and I cannot really request much information about it nor write about it. Also, I lack a lot of cybersecurity skills and concepts to understand the big picture fully.

## **2.2 Evaluation of my current skills and stage of professional development**

I consider myself a beginning-stage actor. I need instructions, either from literature materials, or from mentors, especially when it comes to C., I would like to have also more instructions on how the server should work (not how I think it would) from the project coordinator. I thought that I learned at this job at a very fast pace, so, even though I was stressed and found it challenging many days, I really enjoyed the growth.

The North Central Regional Educational Laboratory has developed a research-based professional development framework that promotes ongoing professional development (University of Oklahoma 2019).

The stages are as follows:

- *Building a Knowledge Base.* Activities in this phase might include goal setting, assessing needs, participating in interactive workshops and forming a study group.
- *Observing Models and Examples.* In this phase, one might participate in activities such as school and classroom visitations, peer observation, using instructional artifacts, co-planning and listening to or watching audio and video examples.

- *Reflecting on Your Practice.* Activities in this phase might include the use of journals or teacher-authored cases for collegial discussion and reflection.
- *Changing Your Practice.* Activities might include action research, peer-coaching, support groups and curriculum development.
- *Gaining and Sharing Expertise.* Activities in this phase might include team planning, mentoring or partnering with a colleague and participating in a network.

Based on these descriptions, and the nature of this thesis, I am obviously in the reflecting & changing your practice stages. However, as the same Oklahoma University (2019) page says, "in practice, the five phases overlap, repeat and often occur simultaneously".

Therefore, I am also observing examples, both from other people and instructional literature, and setting new goals as a part of the building a knowledge base stage. I am journaling (or better said, reporting) my progress with weekly reports and with this thesis. I am submitting the reports to project coordinator and my supervisor (or tutor) for feedback, as well as the line manager sometimes.

In my reports, I try to estimate also what my future goals are based on what I have achieved. I have now started to do similar thing with my thesis, except feedback is provided by the thesis advisor. When it comes to changing my practice, I do so by researching information on the topics I'm working on and requesting feedback from peers and tutors.

Unfortunately, I am not in the gaining and sharing expertise phase as I am a beginning-stage actor, while most colleagues are in the experienced specialist category. Obviously, by researching and reading instructional material, I am also building my knowledge base.

### **Focus on the future. Personal and professional development goals.**

I think I could focus on connecting more with my environment in the next few months: find out what others are working on, the bigger scheme, and find if I can contribute somehow. Also, it would be interesting to see what headquarters' plans are, what direction specifically they are going forward to.

However, as I discussed with one senior worker before, the "main action" is happening in China, and our research center is slightly disconnected. The more senior workers of course are aware of the main action and more connected than me, but as an intern I probably shouldn't be, I don't have that much knowledge or experience.

Also, if I focus on more of what my colleagues are working on, that would be focusing on my weaknesses in a sense, as C, compilers, and OS development is not my profile (I want, and studied to be a full stack developer, and my study path is not ICT infrastructures).

On the other hand, I could focus on my strengths and become better at something I am already moderately good. That is, at developing the Java server (back-end side) or the React applications (front-end side).

Which aspect I should focus on depends, I guess, on the philosophy I chose to adapt. Some say you should capitalize on your strengths; some say you should get out of your comfort zone and improve your weaknesses.

### **2.3 Interest groups at work**

From internal interest groups, I would highlight my supervisors: personal tutor/supervisor, project supervisor, line manager. They have the most direct influence over my work. From external interest groups, there is the headquarters that reviews our research papers and the “products” documented in them, and the networking or consumer business group, part of the headquarters. They would be in a sense our customers.

In Figure 1, you can see these interest groups represented. I would be within the internal, employee’s category, of course. Therefore, other employees and the line manager would be the people of most influence. I do not have direct relationship to the external interest groups, also shown in Figure 1, so those hold less influence on my work.

Abstracting from a quick overview of the Huawei website, the following can be gathered: Huawei is owned by employees. It is a private company wholly owned by its employees. Through the Union of Huawei Investment & Holding Co., Ltd., we implement an Employee Shareholding Scheme involving 121,269 employees. Only Huawei employees are eligible to participate. No government agency or outside organization holds shares in Huawei (Huawei 2022).

Huawei works with a broad range of stakeholders including partners, industry organizations, open-source communities, standards organizations, universities, and research institutes all over the world (Huawei 2022).

It is furthermore an active member of more than 600 industry organizations, including standards organizations, industry alliances, open-source communities, and academic associations. Examples of such organizations are 3GPP, ETSI, IETF, IIC, IEEE SA, the Linux Foundation, CCSA, All, TM Forum, WFA, WWRF, CNCF, OpenInfra (formerly OpenStack), LFN, LFDL, IFAA, GP, CUVA, VRIF, and BBF (Huawei 2022).

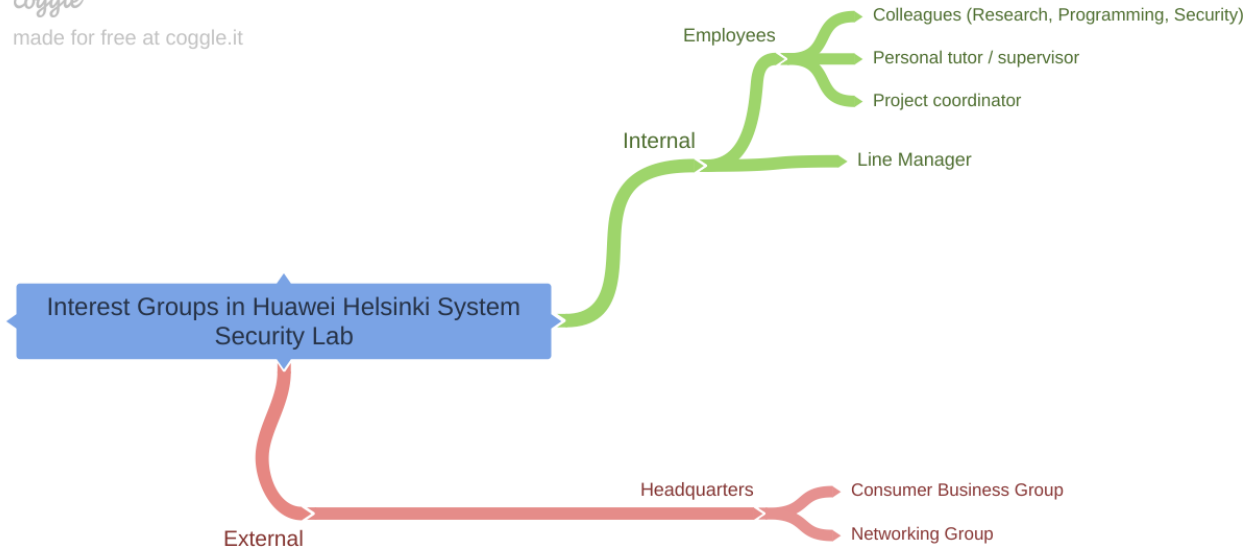


Figure 1. Chart (Coggle) of the Internal/External interest groups. The external interest group can be expanded much further, depending on how broad you want the scope to be. I would be positioned within the Internal, Employee's category.

## 2.4 Interaction skills at work

I have regularly tutor-intern interaction with my tutor and with the project supervisor. I ask them questions and try to get some instructions on what to do, as well as some feedback. I have formal interaction with colleagues working on similar projects (asking for help and sharing code through Git systems). I sometimes have informal interaction with colleagues.

I have no interaction with headquarters or our "customers". That is, no interaction with external interest groups.

### Challenges to these interaction skills

Sometimes it's hard to get clear instructions on what to do as I am an intern and do not have the necessary skills or knowledge to fully participate in the ongoing projects. With my colleagues there are no challenges, except they also are working on different projects from mine.

With external groups, such as the headquarters that approve the new technology or papers I have no interaction, the project coordinator and line manager have them. They are also the ones that have interactions with the Customer Business Group.

### 3 Diary entries

#### 3.1 How I organize and order my objectives and tasks

In the beginning of the day, I outline the general picture of what I have done and am going to do. This is useful in the weeks where I start in the middle of an ongoing task. I also number Tasks, and the Task Outcomes correspond, in numbering, to the tasks (i.e., task 1 has task outcome 1).

#### 3.2 Observation week 1

##### Tuesday 1<sup>st</sup> February 2022

I currently have a Spring Boot Java server with classes, including the Keybox or KeyboxDB class. I currently have a basic front-end to interact with the server. I haven't started coding the client in C yet. I want to introduce more info about the keyboxes in the MySQL database (and server), and also figure out how these keyboxes are added to the server (now I have an upload form).

Objectives:

- Introduce more attributes to the Keybox Class.
- Insert these attributes in the MySQL database where the keyboxes are stored.
- Explore/ask about methods of generating or securely transferring keyboxes to the server, as, the current MultiPartFile POST upload is very insecure.

Tasks:

1. Go to the Model folder of my server, edit the KeyboxDB class.
2. Go to MySQL database via MySQL workbench, insert necessary fields.
3. Go to the Repository folder of my server, open KeyboxDBRepository class, a JPA repository of KeyboxDB objects, and add some methods there.
4. Request the code that is involved in keybox generation, and packaging, from another expert. Read them, take notes.
5. Go to Service folder and see what can be done about the storage service.

Task outcomes:

1. Opened KeyboxDB class. Added quantity (int), changed used from Boolean to int, added "orderindex" parameter (int), probably unnecessary. It basically gives KeyboxDB objects a number in the order they were uploaded, or, created (index and order were reserved keywords). Updated the constructor as well as the getters and setters.
2. Went to MySQL Workbench, connected to "testdb", selected table "files", clicked Alter Table. Added quantity (int), used (int), orderindex (int). Removed isUsed (BIT(1)). I forgot why it was BIT(1), but something about how JPA treats Boolean values.

3. Added methods `findByUsed` and `findByDevice`, necessary after talking to the project coordinator about how the keyboxes will be provisioned.
4. Thanks to my tutor/personal supervisor, he talked to another colleague that has been working in Java with keybox generation, he has allowed me access to his Git repo.
5. Opened `StorageService`. Changed the `store()` method to include quantity and used status.

All tasks were completed. Objectives were accomplished.

Skill development: My tech skills were improved by modifying the Java classes, discovering how to use MySQL workbench, reading the code in the git repo that was shared by another programmer (my colleague). My interpersonal skills were improved as I have asked for some tutoring from my tutor.

### **Wednesday 2<sup>nd</sup> February 2022**

With new attributes added, I need the upload form to support them, at least temporarily until I have not found another way to upload the keyboxes to the server. The current front-end is unprofessional. I need a React one to administer the keyboxes.

Objectives:

- Update the `MultiFormData` to handle new keybox attributes, meanwhile I'm working on internal keybox generation on the server.
- Create a basic React front-end to administer the keyboxes
- Make sure GET commands from server that return JSON content return also the newly added attributes yesterday.

Tasks:

1. Go to Controller folder, check `MultiFileUpload` (/upload endpoint) in the `KeyboxController` class.
2. Follow this tutorial by Baeldung (2022) on how to set-up React Front-end for Java Spring Boot server.
3. Go to the Keybox controller and check GET `/files/` and GET `/files/individual/{id}`, that they return the newly added attributes (I plan on renaming `/files/` to `/keyboxes/` later).
4. Go to `ResponseFile` class and see how the Response JSON is returned back from the endpoints.

Tasks outcomes:



1. Checked the /upload endpoint, made sure new attributes are requested as parameters to the upload. Made sure the storageService.store uses new attributes.
2. Followed the tutorial and set-up a basic Front-end, after some trial and error. It uses fetch to send a GET requests to /files, after which it sets the state with the fetch response. Then map() function is used on the state to map each object of the JSON to a <td></td> entry, within a table that is in the return statement, as JSX.  
  
Router too was easy to set, but I only have the keybox list page for now, and the buttons also remain to be coded.
3. Endpoints /files/ and /files/individual/{id} both used ResponseFile to return JSON to the browser.
4. Opened ResponseFile class and added the attributes there, as well as modified the constructor and the getters and setters.

Tasks and objectives were accomplished.

Skill development: My tech skills were improved considerably by following the Baeldung (2022) tutorial. I learned more about programming the file storage service from there. Also more about returning JSON to the browser/client using ResponseFile.

### **Thursday 3<sup>rd</sup> February 2022**

Decided to improve the React front-end further.

Objectives:

- Add buttons (and start working on the functionality) to
  - Generate keybox, front-end should send GET request to the server, to the /generate endpoint.
  - Delete keybox, should send a DELETE request to the server
  - Edit keybox, should open Edit page in Router.
- Edit page in router: should be able to fetch existing keybox data and show it. Then should have the functionality to edit the data. And finally, to save the data and send it as PUT request to the server.

Tasks:

1. Edit the React .js files to add the buttons, as per the Baeldung (2022) instructions.
2. Add function for sending GET request to /generate.

3. Add function for sending DELETE request to individual keybox, also the same function should renew the list of keyboxes / table data of keyboxes, removing the deleted element
4. Add EditKeybox.js page to router.
5. Add function for opening EditKeybox when Edit button is clicked.

#### Task outcomes:

1. Previously, to generate `<td></td>` entrances to the table of keyboxes, `map()` function was used on the keyboxes stored in state. The variable is `keyboxes: []`, within the constructor of the React component. Baeldung (2022) used old class-based React example, hence why there was a constructor.

Now, when applying this `map()`, I added another `<td>`, which included a `<ButtonGroup>` with Edit and Delete buttons. This way, for each keybox item in the table, there would be two buttons.

Also added a "Generate Keybox" button to the top of the container, above the table and above the heading.

2. Added an async `generate()` function, I decided that it should use `fetch` to send a POST request, after all, creating a new keybox on the server changes the state of the server.

The `generate` function awaits this POST request, and then sends another `fetch` request to get the list of all keyboxes in JSON. Then updates the React state. This way the table gets refreshed with the updated keyboxes.

3. Added an async `remove(id)` function, which sends DELETE requests to `/keyboxes/{id}`. The appropriate endpoint will have to be created on the server. After the request is sent, per the Baeldung (2022) guide, I use the `.filter()` on the state, to remove the deleted item locally from the table.

4. Created `EditKeybox.js`, added it to the router in `App.js` in such a way: `<Route path='/keyboxes/individual/:id' component={KeyboxEdit}/>`

5. Modified the Edit button with the following parameters: `tag={Link} to={"/keyboxes/individual/" + keybox.id}`. It does the trick with the router, to open the edit page, but I need to explore more what `tag={Link}` means.

Tasks were achieved. Objective 1 was fully achieved, objective 2 was partially achieved, moved to the next day.

Skill development: my tech skills with Javascript functions were considerably improved. Also, I started thinking more in terms of front-end to back-end integration: if some button does some function in the front-end, what request it should send to the back-end, and what should the back-end do with this request.

## Friday 4<sup>th</sup> February 2022

Edit further the front-end React. Today is a more relaxed day, so I will attend Friday coffee to discover more what's going on in the office.

### Objectives:

- Create the Edit page in router: should be able to fetch existing keybox data and show it. Then should have the functionality to edit the data. And finally, to save the data and send it as PUT request to the server.
- Attend Friday coffee (14:00-15:00) to get an overall idea of what is happening in the office

### Tasks:

1. Within KeyboxEdit.js page, add a state in the constructor, "item". This is where the current keybox will be loaded.
2. In componentDidMount(), fetch the keybox by id, id being provided in URL params. Save the fetched JSON to item.
3. Create Inputs of type text for all the keybox attributes. Place them inside a form.
4. Populate the Inputs with the attributes from the item.
5. Add handleChange() and handleSubmit() to the form.

### Tasks outcomes:

1. Added the constructor and the state.
2. Added await fetch to the componentDidMount(). The id of the keybox to be fetched is gotten using this.props.match.params.id. Then I use setState with the fetched keybox.
3. FormGroup with Inputs was created. Each input has onChange={this.handleChange}. handleChange is bound to the constructor (necessary in old React).
4. Each Input has value={item.attribute} parameter.
5. onSubmit={handleSubmit} was also added. handleSubmit uses fetch to submit a PUT request to the server, however, because the code was used from Baeldung (2022), it is a bit messy, and contains a choice between PUT and POST requests. handleChange works as intended,

changing the attribute of the item in the state to the value in the Input element. It is also bound to the constructor.

Tasks were achieved. However, PUT and DELETE requests have to be further coded in the Keybox-Controller on the server, they are currently giving errors.

Both objectives were also completed, I both worked on my technical skills and attended the Friday coffee to listen what others have to say.

Skill development: Javascript skills have developed considerably by learning more about the input component, as well as DOM components. Analyzed how the server should be modified considering the new Javascript code additions. People skills were improved by attending the Friday coffee.

### **Weekly observations week 1 (calendar week 5)**

My skills have developed considerably this week. From the technical aspect, I have developed my Javascript skills, using EcmaScript functions. I have developed class based React skills by following the Baeldung (2022) tutorial and expanding on the code using my knowledge from Server Programming and Front-end programming classes.

I have developed my interpersonal skills by asking feedback on the server from the project coordinator, asking for help from my tutor regarding security, keybox encryption and such. I have developed my observational skills and interpersonal skills further by coming to the Friday coffee and listening about what my colleagues were talking. I did little contribution myself as I was a bit intimidated, being an intern surrounded by senior-level workers.

I had to get informed more on security and encryption algorithms, how they are used in Java classes. I'll revisit them next week and analyze the theory in more detail. I also wanted to clarify from my project coordinator how my server development fits into the bigger picture of the whole project. I will re-read the article he gave me before and make a diagram for better understanding, for myself.

I need to clarify more about how DELETE/PUT functions affect the server, I'll do so next week. I had to research further how existing KeyboxDB objects, within the KeyboxDB repository and on the MySQL database, can be edited through PUT requests, and, specifically, through a front-end edit page.

When it comes to problems, as I currently see it, I lack the understanding of the bigger picture in how my work fits the project. Talked to the project coordinator. Re-read the article. Will make a diagram next week to simplify the understanding.

I lack some understanding of basic security concepts when it comes to keybox encryption. Talked to my tutor. Got access to some Java files where encryption happens so I could read and analyze them. I will read more about X.509 certificates, ASN.1 encryption and BouncyCastle by myself.

As I have mentioned previously, I have used the Baeldung (2022) resource as a baseline on “good ways to act”, which was based on Spring Boot and its integration with React through a plugin they developed, which was added to the pom.xml file.

When it comes to C and Linux application development, I have not used any of my main cited books (Johnson & Troan, 2005 or Lippman & Lajoie, 1998) yet as the main task for which I’ve been hired is for developing the server. However, I am already starting to think how, instead of using a web-based front-end to fetch the information from the server (and POST/PUT), a C client could be used for such purpose.

I have used an internal thirty-page paper/article prior to starting this thesis to understand what this project is about, and will use it further to draw a diagram now, and maybe, with the use of that diagram I can see an alternative view on my coding/architecture. I cannot cite this article in my thesis, however, due to confidentiality issues (it’s not available publicly either).

Since my project is in a sense a full-stack project, I’ve considered that I need to read more about MVC architecture. I have already used concepts of MVC in my daily reporting (i.e. model in Tuesday’s report). According to Wikipedia (2021), an MVC architecture consists of these components

- *Model*: The central component of the pattern. It is the application's dynamic data structure, independent of the user interface. It directly manages the data, logic and rules of the application.
- *View*: Any representation of information such as a chart, diagram or table. Multiple views of the same information are possible, such as a bar chart for management and a tabular view for accountants.
- *Controller*: Accepts input and converts it to commands for the model or view.

In my case I have the `KeyboxController.java` class, which accepts HTTP requests to the endpoints, and then does the pertinent commands in other classes, such as `KeyboxDB` repository, `StorageService`, or the `ResponseFile` classes.

The views would be JSX pages produced in React, where user can interact with the elements that interact with the controller. The model would be all the internal Java classes, including the class on which keybox objects itself are based (`KeyboxDB`), the `KeyboxDB` repository as well as the MySQL database accessed through JPA/Hibernate, that the end-user knows nothing about.

A specific theme would be the development of a REST server using Spring Boot that would cater all the requests from both the front-end, meant to be used as an admin API, and from the C client located on host machine that is supposed to act as a bridge between the LiteOS (Linux based OS where the keyboxes will be provisioned). I decided to research more on RESTful services to see if it somehow relates to my project.

*Representational state transfer (REST)* is a software architectural style that was created to guide the design and development of the architecture for the World Wide Web. REST defines a set of constraints for how the architecture of an Internet-scale distributed hypermedia system, such as the Web, should behave (Wikipedia, 2022).

Oracle (2010) mentions that RESTful applications should have the following principles:

1. *Resource identification through URI.*
2. *Uniform interface:* Resources are manipulated using a fixed set of four create, read, update, delete operations: PUT, GET, POST, and DELETE. PUT creates a new resource, which can be then deleted by using DELETE. GET retrieves the current state of a resource in some representation. POST transfers a new state onto a resource.
3. *Self-descriptive messages:* Resources are decoupled from their representation so that their content can be accessed in a variety of formats, such as HTML, XML, plain text, PDF, JPEG, JSON, and others.
4. *Stateful interactions through hyperlinks:* Every interaction with a resource is stateless; that is, request messages are self-contained. Stateful interactions are based on the concept of explicit state transfer.

I think my project follows all these principles quite well, based on the daily descriptions I have provided.

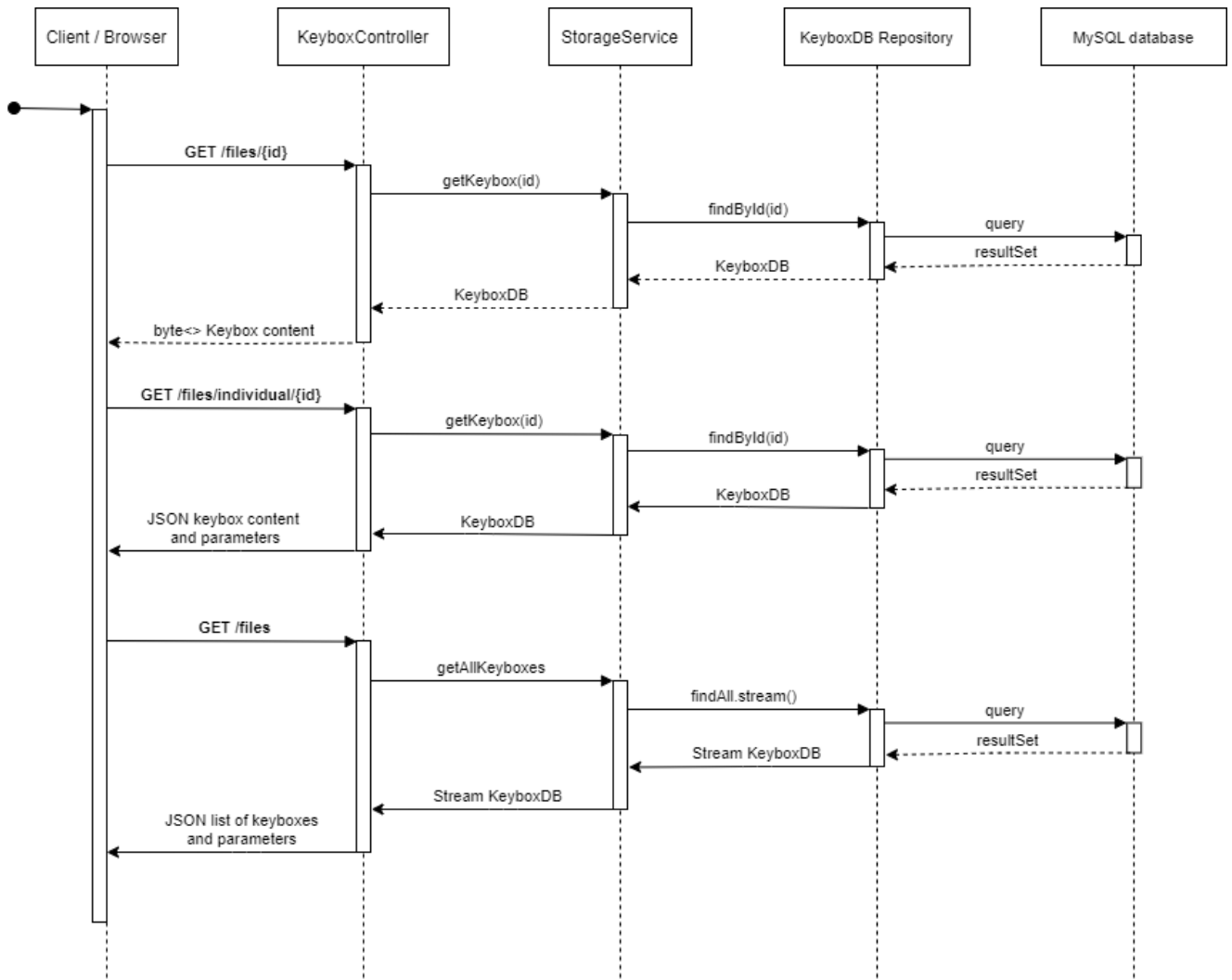


Figure 2. Sequence diagram of the current components in the project. See explanation above.

I have added above a sequence diagram (Figure 2) of the current REST endpoints I am using in my project. You can see how the requests, for example GET /files/{id}, get processed by the different Java components, and what they return (in first example, a byte array with the keybox content).

### 3.3 Observation week 2

**Monday 7<sup>th</sup> February, 2022.**

Currently the PUT and DELETE endpoints are not working properly. Make them work properly in the following days.

Objectives:

- Code PUT and DELETE requests to the server so that delete and edit buttons from Front-end work.

Tasks:

1. Go to Controller > KeyboxController. Implement PUT and DELETE requests there

2. Go to Repository > KeyboxDB Repository. Add methods there to fit the new PUT and DELETE Endpoints

Task outcomes:

1. Went to the controller. Added @PutMapping. I decided that since the editing is gonna be on a keybox of certain id, the id could be in the @PathVariable.

After many errors and consecutive debugging, I decided that what would happen it that the program would load an existing keybox from the KeyboxDB repository and replace it with a new one, that would be created with the data in the Input elements on the EditKeybox page.

I added .getById(id) method to the KeyboxDB repository to get the existing keybox using id from @PathVariable. I added @RequestBody KeyboxDB to the PUT method to request a new keybox with which the old would be replaced.

Then it was only a matter of using getters and setters, in such way existingFile.setData(newFile.getData()); for all the attributes.

For DELETE function, I also added a @PathVariable to get the id of the keybox, and then I just had to create a method deleteById(id); in the KeyboxDB repository.

2. Created methods .getById(id) and .deleteById(id)

All tasks and objectives were accomplished.

Skill development: Java skills have developed considerably. Debugging skills (and patience) has increased.

**Tuesday 8<sup>th</sup> February, 2022.**

I decided to have a more theoretical day, so I can understand better the current project, summarize it using a figure, and study some security concepts.

Objectives:

- Get to know the bigger picture of the project by re-reading the article and drawing a diagram in draw.io
- Familiarize with security concepts: BouncyCastle, ASN.1, X.509, AES, Hashes

Tasks:

1. Re-read the article and draw a diagram



2. Read Wikipedia articles on these concepts, document, and follow Wikipedia sources to wherever they might lead.

Task outcomes:

1. Today I decided to dedicate the day to theory. It took me a while to read the article and create the diagram, almost the entire day.

I have put the diagram here (Figure 3) with blurred out keywords in case of confidentiality issues. Essentially the PKI server would provision the device with keyboxes. The device has a MCU and SE components. These components have sub-components, like SE has JavaCard for example.

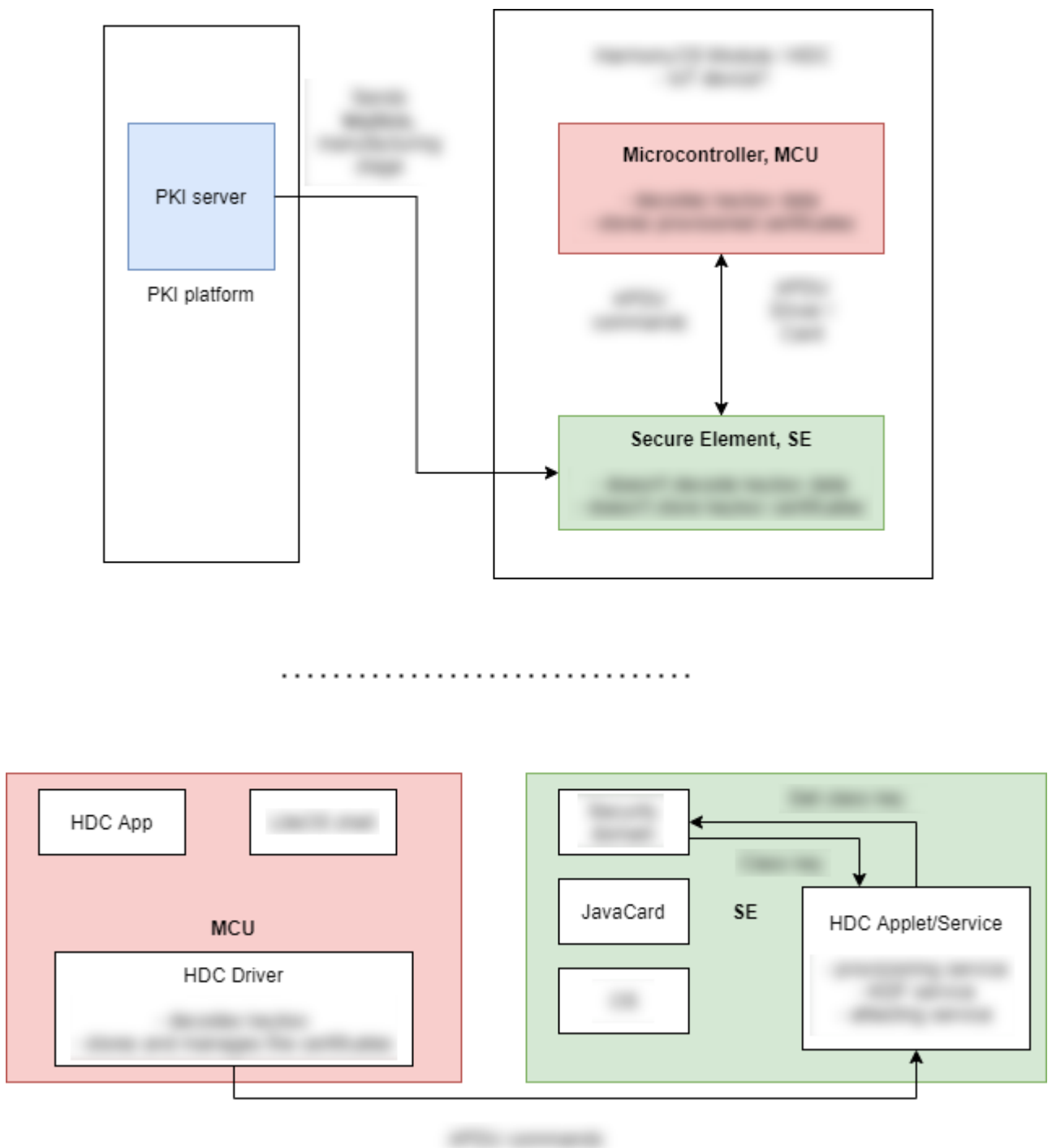


Figure 3. Diagram of the article. Shows different components within the IoT device and its connection to the PKI server. See explanation above.

2. I researched basic concepts. I will document them in the weekly observations. Found out that X.509 is a certificate standard, based on ASN.1, another standard. I thought it was the other way around. Also found out that BouncyCastle is a collection of APIs', that supports different certificates.

My tutor has explained me the basics of BouncyCastle, although I haven't grasped it fully so more research was needed. I also found an ASN.1 sample certificate generator, that has helped me to understand better what these certificates contain.

Skill development: I improved my research and reading comprehension skills by reading the article and the theory on security concepts.

### **Wednesday 9<sup>th</sup> February, 2022.**

I got some code from a colleague to generate keyboxes on the server using Java. Unfortunately, the Java files do not use Maven dependency management. I have to convert them, and then create a service within my application that uses the keybox generator code.

Objectives:

- Incorporate the keybox generator files provided on previous week into my server.

Tasks:

1. Convert the files from Gradle to Maven, put them in a folder in my server architecture.
2. Create a GenerateKeybox service in my application. That service would be part of the Model aspect in MVC as the customer will not know anything about it.

Task outcomes:

1. This task was what I would describe as a pain in the bottom rear end. Very trivial task but gave me a lot of errors so I had to do it through trial and error. I read Baeldung (2022) tutorial as well as several StackOverFlow discussions, and in the end it was something involving mvn package command. In the end, the .jar compiled without errors and the server was running with the keybox files in it.
2. Luckily this was way easier than I thought. I found that using the keybox classes you could just generate a byte array, then using another class, Util, create the keybox filename. Then it was

only a matter of creating a new KeyboxDB object with the name, byte array as the data, and some sample values for the other attributes.

Then you would save the new object into the repository, after which it was automatically added using JPA to the MySQL database. That's the functioning of the GenerateKeybox service.

I consider the tasks and objectives to be accomplished.

Skill development: research skills have improved through reading tens of StackOverFlow articles. I also learned more about Gradle/Maven dependency managers.

## Thursday 10<sup>th</sup> February, 2022.

Now that the generator code is on the server, make sure it can be accessed using the front-end.

Objectives:

- Incorporate generate function into the front-end.

Tasks:

1. The generate keybox button on the front-end already sends a POST /generate request. Now the request should return something. Update the controller.

Task outcomes:

1. Edited the KeyboxController class. Made /generate endpoint run the GenerateKeybox service, then return a ResponseMessage with a message and keybox id. ResponseMessage is a class in my application, like ResponseFile, that returns JSON to the browser. It uses the ResponseEntity class that is already within the Spring Boot library.

In Figure 4 & 5 you can see how pressing the generate button, the keybox is added automatically to the MySQL database, and shown in the view (it gets updated automatically).

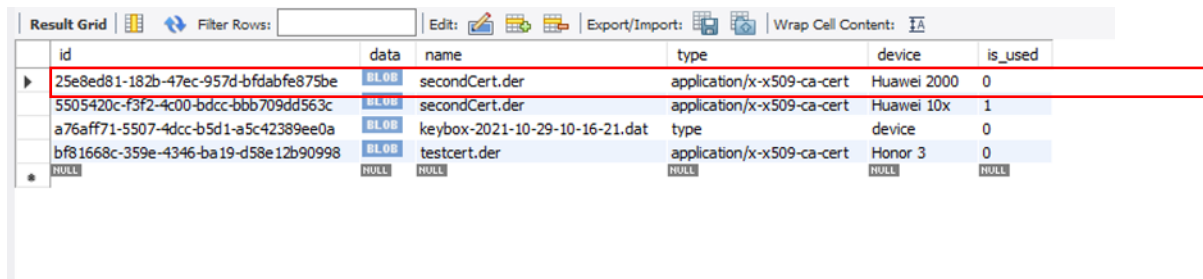
Home Nav Item 2 Nav Item 3

Generate Keybox

Files

Name	Download .DER	Device	Used	Actions
secondCert.der	<a href="http://localhost:8080/files/25e8ed81-182b-47ec-957d-bfdabfe875be">http://localhost:8080/files/25e8ed81-182b-47ec-957d-bfdabfe875be</a>	Huawei 2000	false	Edit Delete Mark as Used
secondCert.der	<a href="http://localhost:8080/files/5505420c-f3f2-4c00-bdcc-bbb709d4563c">http://localhost:8080/files/5505420c-f3f2-4c00-bdcc-bbb709d4563c</a>	Huawei 10x	true	Edit Delete Mark as Used
keybox-2021-10-29-10-16-21.dat	<a href="http://localhost:8080/files/a76aff71-5507-4dcc-b5d1-a5c42389ee0a">http://localhost:8080/files/a76aff71-5507-4dcc-b5d1-a5c42389ee0a</a>	device	false	Edit Delete Mark as Used
testcert.der	<a href="http://localhost:8080/files/bf81668c-359e-4346-ba19-d58e12b90998">http://localhost:8080/files/bf81668c-359e-4346-ba19-d58e12b90998</a>	Honor 3	false	Edit Delete Mark as Used

Figure 4. Pressing the generate keybox button adds new keybox to the database and updates the front-end website view.



id	data	name	type	device	is_used
25e8ed81-182b-47ec-957d-bfdabfe875be	secondCert.der	secondCert.der	application/x-x509-ca-cert	Huawei 2000	0
5505420c-f3f2-4c00-bdcc-bbb709dd563c	secondCert.der	secondCert.der	application/x-x509-ca-cert	Huawei 10x	1
a76aff71-5507-4dcc-b5d1-a5c42389ee0a	keybox-2021-10-29-10-16-21.dat	keybox-2021-10-29-10-16-21.dat	type	device	0
bf81668c-359e-4346-ba19-d58e12b90998	testcert.der	testcert.der	application/x-x509-ca-cert	Honor 3	0

Figure 5. Keybox is added to the database. Figure shows MySQL workbench, with the database I am working with opened and showing the rows and columns.

I consider my tasks and objectives to be accomplished.

**Friday 11<sup>th</sup> February, 2022.**

Next week I want to start developing the client in C, so I decided to study C socks interface today.

Objectives:

- Do a simple task in C provided by the tutor to grasp the basics.
- Study the sock interface.

Tasks:

1. The task provided was coding an application that opens files, writes, closes them, and reads the file content.
2. I will study the sock interface by using the Linux Application Development book by Johnson Troan (2005).

Task outcomes:

1. I used several sources to learn how to do the required functions, including Tutorialspoint (2021), Programiz (2022) and Geeksforgeeks (2021).

I have used the libraries `stdio.h`, `stdlib.h`, `string.h`. I have tried different functions (C has apparently many of them), `fopen`, `fread`, `fwrite`, `fgetc`, `fgets`, `fscanf` and of course `fclose` to close the file and prevent memory leaks.

2. The socks interface seemed fairly easy to use. I only needed to use it on client side, as the server was not in C. On client side, there are just a couple functions: `socket()`, `connect()`, and

once the connection has been established, `write()` and `read()`. Finally, when communication is no longer necessary, `close()` is executed.

Meanwhile on the server side, what happens after the client has executed the `socket()` command is, the server (after also creating the `socket()`) goes through `bind()`ing, and then `listen()`ing to the client connections. After client sends the `connect()` request, the server executed the `accept()` command. After that `read()` and `write()` functions can be executed. That is the basics that I have understood from Johnson & Troan (2005).

The `socket()` command creates a new uninitialized socket. It is tied to a particular protocol, but it is not connected to anything. As it is not connected, it cannot be read or written to (Johnson & Troan 2005). This function returns a negative value on error, and 0 or positive value on success. First parameter specifies domain, second type, and third the protocol.

Then the `connect()` command specifies the socket, the address, and length of the address. Length is just `sizeof`, but when it comes to the address, you specify it as `(struct sockaddr *) &address`. This is where complexity of C becomes apparent (at least for me). Structs and pointers are really not that well studied in the programming classes at Haaga-helia, so it brings some confusion.

Both `read()` and `write()` functions are similar,  
*`write(sockfd, buff, sizeof(buff));`*  
*`read(sockfd, buff, sizeof(buff));`*

They require the socket, the buffer, and `sizeof` buffer (Geeksforgeeks 2022).

I consider the tasks and objectives accomplished.

Skills developed: from reading the theory on C language functions, I have improved my CC skills, specifically when it comes to the functions involved in sockets interface.

### **Weekly observations week 2 (calendar week 6)**

I was interested on how JPA/Hibernate works in the sense how the changes in my Java code affect the associated MySQL database. The Java Persistence API (JPA) is a specification that defines how to persist data in Java applications (Baeldung 2022).

It is part of a more general Object-Relational Mapping layer, which is the process of converting Java objects to database tables (Baeldung 2022). Hibernate is one of the most popular Java ORM frameworks in use today.

JPA processes classes as “entities” that are added to the database via `@Entity` annotation. Furthermore, you can use `@Table(name="STUDENT")` to further name the SQL table where these entities are stored. Similarly, `@Column` can be used for different attributes of the class (Baeldung, 2019).

Also I decided to investigate on the encryption algorithms. BouncyCastle is a collection of APIs used in cryptography. It includes APIs for both the Java and the C# programming languages (Wikipedia, ). Original version supports basic X.509 certificate generation, which I believe our keybox certificates are based on. The Bouncy Castle architecture consists of two main components that support the base cryptographic capabilities. These are known as the 'light-weight' API, and the Java Cryptography Extension (JCE) provider. Further components built upon the JCE provider support additional functionality, such as PGP support, S/MIME, etc (Wikipedia 2021).

That prompted me to research more on X.509. X.509 is an International Telecommunication Union (ITU) standard defining the format of public key certificates. X.509 certificates are used in many Internet protocols, including TLS/SSL.

An X.509 certificate binds an identity to a public key using a digital signature. A certificate contains an identity (a hostname, or an organization, or an individual) and a public key (RSA, DSA, ECDSA, ed25519, etc.), and is either signed by a certificate authority or is self-signed. When a certificate is signed by a trusted certificate authority, or validated by other means, someone holding that certificate can use the public key it contains to establish secure communications with another party, or validate documents digitally signed by the corresponding private key. I believe there was a part in the data security course which described public keys that use digital signatures.

X.509 also defines certificate revocation lists, which are a means to distribute information about certificates that have been deemed invalid by a signing authority, as well as a certification path validation algorithm, which allows for certificates to be signed by intermediate CA certificates, which are, in turn, signed by other certificates, eventually reaching a trust anchor.

X.509 is based on ASN.1, another ITU-T standard (Wikipedia 2022). So, it seems that X.509 is based on ASN.1, not the other way around as it was my hypothesis during the week. When it comes to the ASN.1 standard itself, it is a standard that defines a formalism for the specification of abstract data types.

The notation provides a certain number of pre-defined basic types such as:

- integers (INTEGER),
- booleans (BOOLEAN),
- character strings (IA5String, UniversalString...),
- bit strings (BIT STRING), (ITU 2022).

This structure can be analyzed in the ASN1 playground resource which I have mentioned previously (OSS Nokalva 2021-2022). Prior to ASN.1, information to be conveyed in communication protocols was typically specified by ascribing meanings to particular bits and bytes in protocol messages, much as programmers, before the advent of high-level languages, had to deal with the bits and bytes of storage layout.

With ASN.1, the protocol designer can view and describe the relevant information and its structure at a high level and need not be unduly concerned with how it is represented while in transit. Compilers can provide run-time code to convert an instance of user or protocol information to bits on the line (Stedman, 1990).

### **3.4 Observation week 3**

**Monday 14<sup>th</sup> February, 2022.**

Using the new knowledge of socks interface, see if I can send some HTTP requests to the server (to the rest endpoints from week 1).

Objectives:

- Find how HTTP requests can be sent using the socks interface.
- Send an example request to the server

Tasks:

1. Get a simple example code from some website that sends HTTP requests (I guess there are plenty of examples written for this purpose).
2. Analyse the code with the newly learned information about the socket interface.
3. Plug in values of my own Java server that is running on localhost to the C client, and see if it works

Task outcomes:

1. I found a simple code on StackOverFlow (2014). Somehow, I cannot find the post anymore, but it looks very similar to the one I cited (StackOverFlow 2014). Besides `stdlib` and `stdio.h`, it uses different libraries, such as `unistd.h`, for the `read`, `write` and `close` functions, `string.h` for `memcpy` and `memset`, `sys-socket.h` for `socket()` and `connect()` functions, `netinet/in.h` for the structs involved `sockadd_in` and `sockaddr`. Finally, it uses `netdb.h` for `hostent` and `gethostbyname`.
2. The code gets the hostname from argument 1 `argv[1]`, it uses `atoi` to convert the port number from string, presented as argument 2, to an integer. Finally it reserves 1024 bits to `message_fmt` variable, which contains the GET request itself.

The GET request is structured as such:

```
strcpy (message_fmt, "GET ");  
strcat(message_fmt, argv[3]);  
strcat(message_fmt, " HTTP/1.0\r\n\r\n");
```

Where argv[3] is the resource we are trying to get. Then it uses the familiar socket() and connect() functions, and then it has a do while loop where it sends the message, the message being a combination of message\_fmt mentioned above, the host, and the port.

Then, there is a do while loop where it reads the response. Then the socket is closed, and response is printed on the command line.

3. I tried the code introducing my own values, and changing GET to POST, and it worked.

Skill development: learned more about C reading the theory behind the sockets interface, also connected the theory I've read before to the practical code I've seen now.

## **Tuesday 15<sup>th</sup> February, 2022**

I want to connect the server sending a keybox, and the C client, fetching the keybox.

Objectives:

- Make the server respond with an unused keybox when receiving a GET request to the /unused endpoint
- Plan how the C client would fetch such an unused keybox

Tasks:

1. So far, the C client can send a request to /generate a new keybox on the server. It would be more useful however that the client would download an unused keybox for a device, as on practice the keyboxes would be moved to the server and not generated there, but the client would have to fetch them. For this a new /unused endpoint is needed to be coded in the controller.

Task outcomes:

1. This was a bit challenging, I found that JPA supports such methods as .findByParameter(), so I could set it up to .findByUsed(0). However, there are multiple unused keyboxes, so that command would return a stream of the KeyboxDB objects.

I would further have to use .findAny() method on the stream to make the application pick a random keybox out of the stream, and for some reason I have to also make these methods go through StorageService (and not directly to KeyboxDB repository) because otherwise it will give me an error.



Finally, in the controller, I had to also put `.orElse` method and create a new `KeyboxDB` with it, in case `.findAny()` doesn't return any values (i.e. there are no unused keyboxes). This is so because otherwise the return of that "method chain" would be `Optional KeyboxDB`, and that gives me an error.

So, in short, the `KeyboxController` has this line to return an unused keybox:

```
KeyboxDB unusedKeybox = storageService.getUnusedKeyboxes().findAny().orElse(new KeyboxDB());
```

Where in `storageService` the `.getUnusedKeyboxes()` method returned `keyboxDBRepository.findByUsed(0).stream()`;

Finally, I had to determine what to return back to the client/browser. I decided that if `unusedKeybox.getData() == null`, it means new `KeyboxDB` was created in the `orElse` methods, means there were no available keyboxes. In such case the server would return a response message that "No keyboxes are available", with a HTTP status of 404.

Else it would send back a `ResponseFile` with the keybox in it, and change the used status to 1 from 0.

Task 1 was completed successfully. Objective 2 was not achieved, although honestly, from the beginning I thought I would not have enough time to achieve it. Moved objective 2 to the next day.

Skill development: A lot of coding-based thinking was developed today to find out how the `/unused` endpoint should work.

### **Wednesday 16<sup>th</sup> February, 2022.**

Work more on the C client that should fetch the keyboxes.

Objectives:

- Plan and do initial coding of C client that fetches an unused keybox.

Tasks:

1. Start coding and figure out on the go how to solve the obstacles that are appearing.

Task outcomes:

1. Using the previous C client code, I sent a GET request to the /keyboxes/unused, now the response consists either of Keyboxes' JSON or a message that no unused keyboxes are left. C doesn't have an inbuilt JSON parser, and using the JSON parser libraries was too time consuming for me.

However, the response is still received with sockets, and it contains both headers and body. And body contains the id of the keybox. Instead of JSON parser, I decided to use C string functionality. In the download url, there is the word files/ followed by id. I decided to use strstr to find the "files/" keyword, find the length of the id substring, then extract it into char[] using a while loop.

C works with char arrays instead of strings.

If the length of the substring is negative, means there is no substring found, means the server didn't return any keyboxes, meaning there are no unused keyboxes. If the substring is found and extracted, send another GET request to /files/{id} endpoint, which returns the keybox content.

fopen() in the C client, write the content to the file, fclose. That way I could download the keybox locally using C and save it.

I consider the objectives and tasks accomplished.

### **Thursday February 17<sup>th</sup>, 2022.**

On practice, the keybox has 3 used statuses: 0, unused, 1, used, 2, confirmed. If the quantity of keyboxes is for example 100 for one device type, you want to mark the keybox as used as soon as first keybox is loaded, but once they all end, you want to confirm them. For now, in my C code, I will make it send a request to confirm the keybox as soon as first one is downloaded. On practice however, it would have to do the confirmation once all the keyboxes are loaded onto the device.

Objectives:

- Add functionality to the C client confirm the downloaded keyboxes.

Tasks:

1. Make the C code send a confirm request
2. /confirm endpoint needs to be added to the server controller.

Task outcomes:

1. I added a third request to the C client to the path files/confirm/ to confirm the keyboxes.
2. Added /confirm/{id}, so upon calling that endpoint you can confirm the device by id. The method in the controller calls StorageService, findById() method, then changes the used status of the keybox to 2.

I consider the objectives and tasks accomplished.

Skill development: I learned to think more globally regarding client and server, not separately, but think at the same time what I want to do client-wise and server-wise.

### **Friday February 18th, 2022.**

On practice, the server will contain X quantity of same keyboxes for a certain device type. It will transfer these keyboxes to all the devices of that type, then confirm that the transfer has been done. So, an endpoint is needed to fetch the keyboxes depending on the device on which the C client is running (the client will know the device on practice).

Objectives:

- Change the functioning of the program so that keyboxes can be fetched by device and confirmed by device, instead of ID. Also do so that confirmation only occurs when all the keyboxes are transferred to the pertinent device type.

Tasks:

1. Create endpoint, /keyboxes/device/{device}.
2. Create a confirm endpoint /keyboxes/confirm/{device} that confirms the keyboxes based on the device type.

Task outcomes:

1. In the KeyboxController class, I created an endpoint /keyboxes/device/{device} for GET requests. The class uses method findByDevice() that I defined in the StorageService. The method returns a KeyboxDB object. It then runs the .getQuantity() method on the keybox object.

If the quantity is less or equal than 0, it sets the used status to 0, saves the KeyboxDB object, and returns a message that there are no unused keyboxes left. All the keyboxes have been transferred to the devices of that type, and so the transfer has been confirmed.

Else, it checks if used status is 0. That means it's the first keybox that is being transferred. It sets the used status to 1 to signal that the provisioning is in process. Whether the used status is 0 or 1, it returns the keybox.

2. `/keyboxes/confirm/{device}` reduces the keybox quantity by one. My application logic is now that the client calls first GET request to `/keyboxes/device/{device}`, gets the keybox, then calls second GET request to `/keyboxes/confirm/{device}`, confirming that 1 keybox has been transferred. And once all the keyboxes have been transferred (quantity = 0), the status changes to 2 which is that the provisioning has been completed/successful.

I consider my objectives and task for today accomplished.

### Weekly observations week 3 (calendar week 7)

This week I decided to focus on analyzing more the C code I am currently using to download and confirm the keyboxes. The code now sends two requests: one GET to get the keybox by device, one POST to confirm that the keybox has been procured on the device and reduce the quantity on the server by 1. After the first GET request, the keybox gets saved to a file. It's interesting to analyse how it works, i.e. the `recv()`

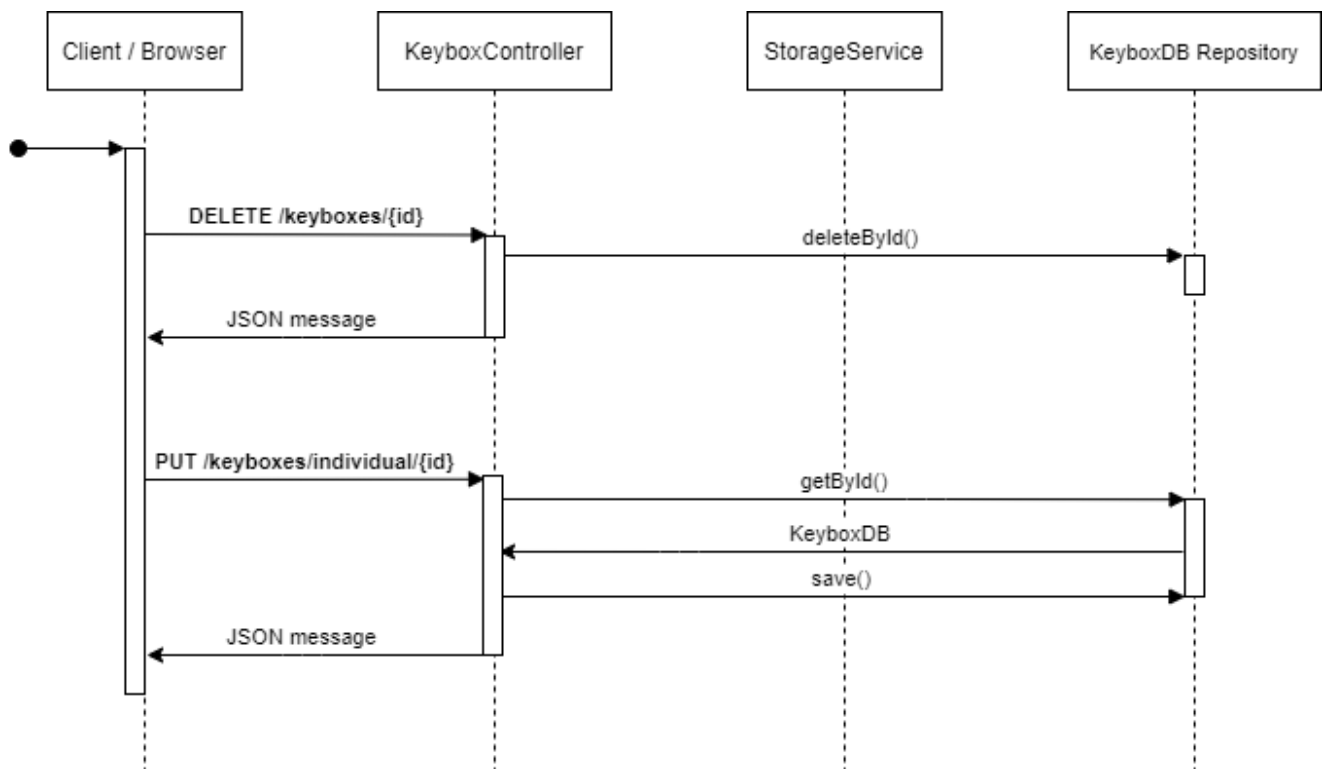


Figure 6. PUT/DELETE endpoints in the diagram. The endpoints used in the previous diagram are not listed here (`GET files/{id}`, `GET files/`, `GET files/individual/{id}`). I have also renamed `/files` endpoints to `/keyboxes`.

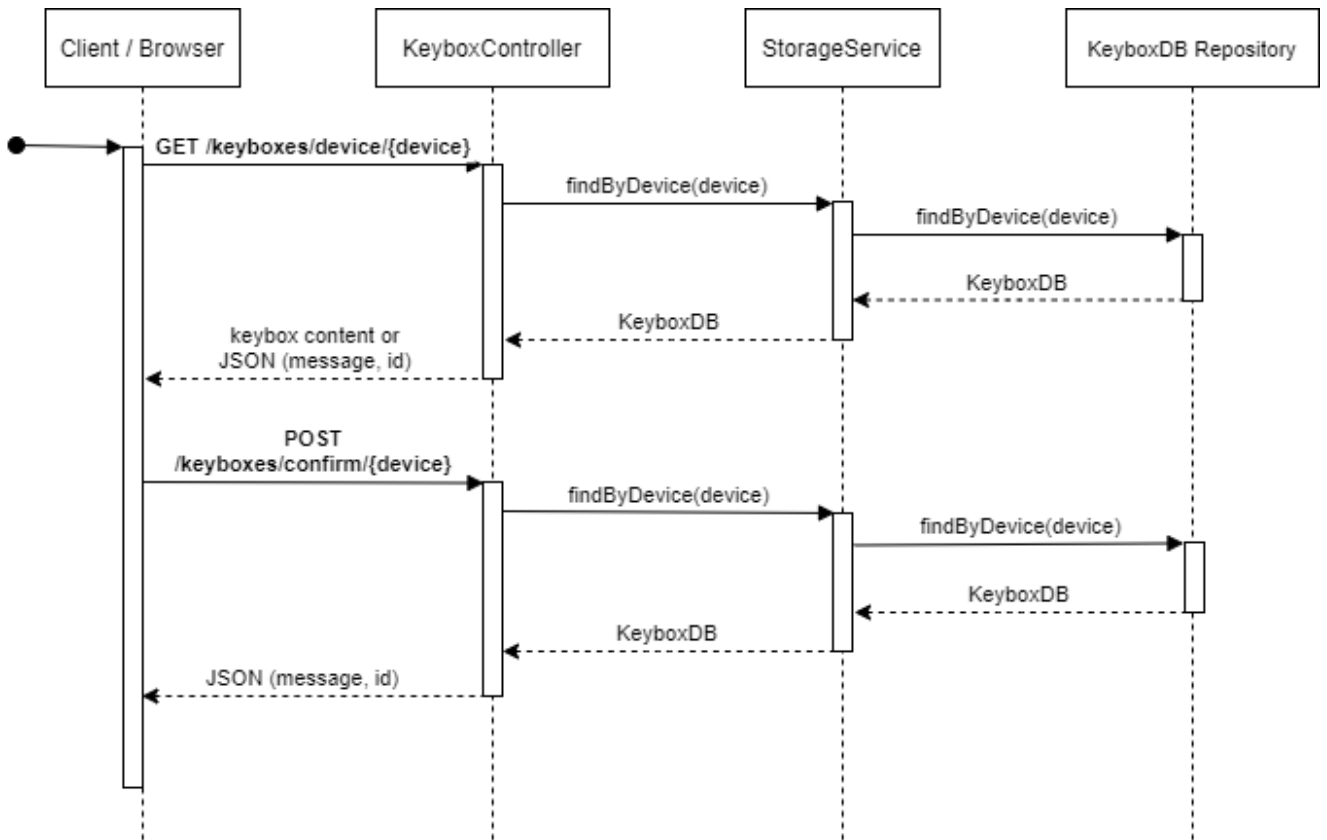


Figure 7. The endpoints used by C client, get keybox by device type, and confirm keybox by device type.

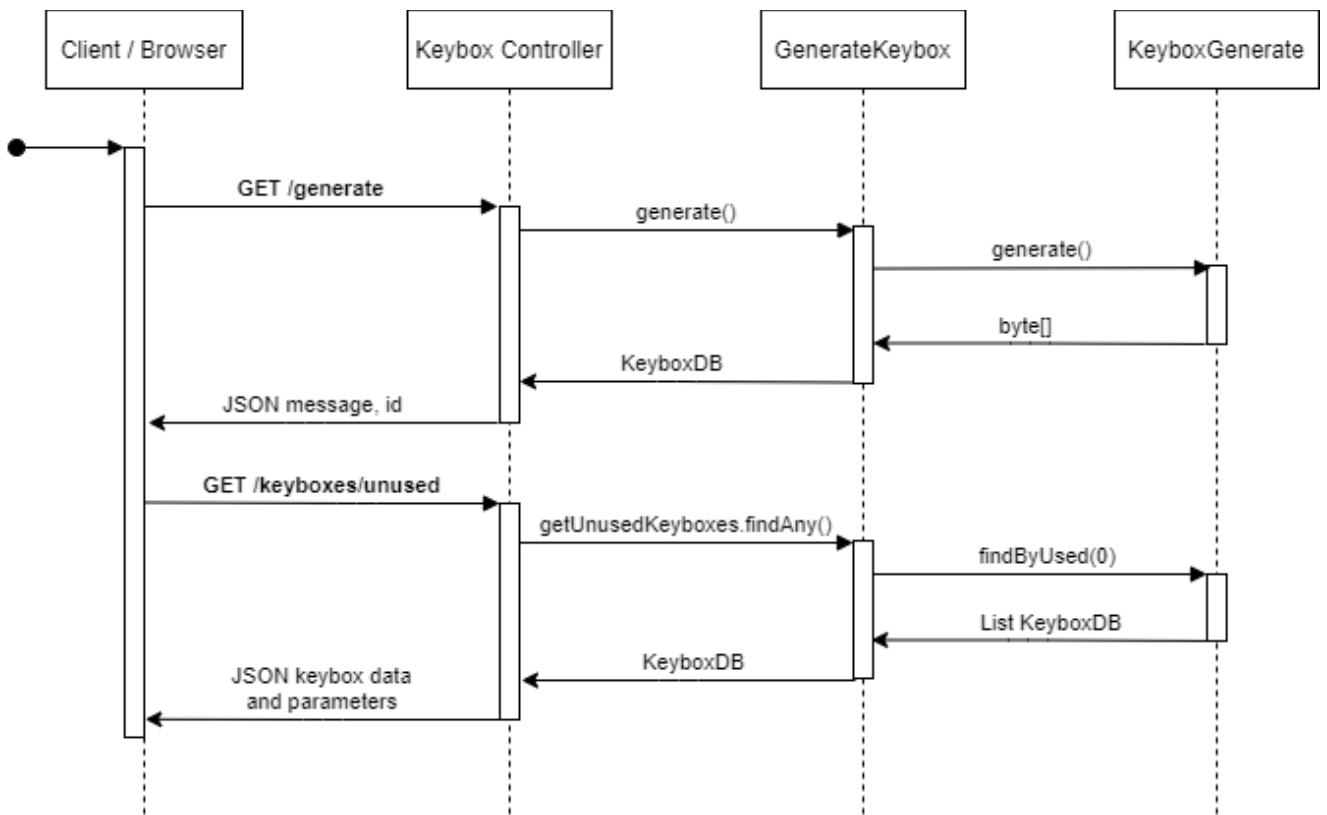


Figure 8. Other endpoints coded this week, /generate keybox and get unused keybox.

I have the diagrams for the new requests that can be made to the server (in addition to those of the previous diagram), and the responses that the server will send. For example, Figure 6 contains

PUT/DELETE endpoints. Figure 7 contains the endpoint that C client would use (which are different from the basic ones that the web-browser would use). Figure 8 contains other endpoints, such as those used to generate new keyboxes, or for getting an unused keybox.

### 3.5 Observation week 4

#### Monday February 21<sup>st</sup>, 2022.

Create a new code for the C client. It should first ask for the device name. Although on practice, the client on the OS would know the device name, for the purpose of my experimentation, I will ask for the device name.

Objectives:

- Change the functioning of the C client so that two GET endpoints get called, /keyboxes/device/{device}, and /keyboxes/confirm/{device}
- Confirm that everything works as intended

Tasks:

1. Modify the C client code
2. Then it should call first endpoint, download the keybox.
3. After 1 keybox has been downloaded, confirm should be sent to the server, reducing the quantity by 1.
4. The front-end should reflect the change in quantity.

Task outcomes:

1. Modified the existing C code slightly with the new URLs, tested, everything seems to work. I modified the endpoint for confirm to POST, although its internal functioning is the same.

I did it as per the advice of one of my colleagues who said GET requests shouldn't change the state of the server. I hope this is what he meant. Although I am not sure if I should also change the first GET request: it gets the keybox, in a sense, but it also changes the used status on the server.

2. Keybox is downloaded and saved locally. You can see the success message printed to the command prompt in Figure 9.
3. Works. You can see in Figure 9 that confirm request has been sent and the success message.
4. Works, except you have to refresh the front-end.

```
$ ./a.exe
Input device name: device8
device8

GET /keyboxes/device/device8 HTTP/1.1

Keybox saved locally.

POST /keyboxes/confirm/device8 HTTP/1.1

Keybox has been confirmed. Keybox counter -1.
```

Figure 9. Working of the C client locally. Prints two requests (GET and POST) and the success messages. See the explanation above.

I consider the objectives and tasks accomplished.

Skill development: Improved C skills by coding two HTTP requests using socks interface in the C program.

### **Tuesday February 22<sup>nd</sup>, 2022.**

I decided to continue with C development, and try to run some C code on the LiteOS device itself. For that first I need to download the LiteOS image from the repositories, and then build it correctly.

Objectives:

- Download LiteOS from Github, build an image.

Tasks:

1. Find a repository with working LiteOS code. Since most LiteOS versions are on private repositories, I don't have so much hope for public repositories. I have asked from another colleague to give me access to a private repository that he was using.
2. Compile it on Linux.
3. Compile it on Windows.

Task outcomes:

1. I have tried several public source codes from <https://github.com/LiteOS>, [https://code.open-source.huaweicloud.com/openharmony/kernel\\_liteos\\_a](https://code.open-source.huaweicloud.com/openharmony/kernel_liteos_a), <https://gitee.com/LiteOS/LiteOS>.
2. There were compilation errors when using the networking and filesystem components, using both Linux and LiteOS studio on Windows (Figure 10). I used the Gitlab private repository.

- I managed to compile it on Windows with Ramfs component to store the keyboxes, but networking functionality was still not working. I used the originally planned idea of using UART port via Qemu and then using another file on host called `hostpipe_cpp`, to push the keyboxes to the LiteOS.

Most of the functionality for the UART was already coded by my colleague, so I had to just plug in my C client code inside the demo.

I consider the tasks and objectives completed.

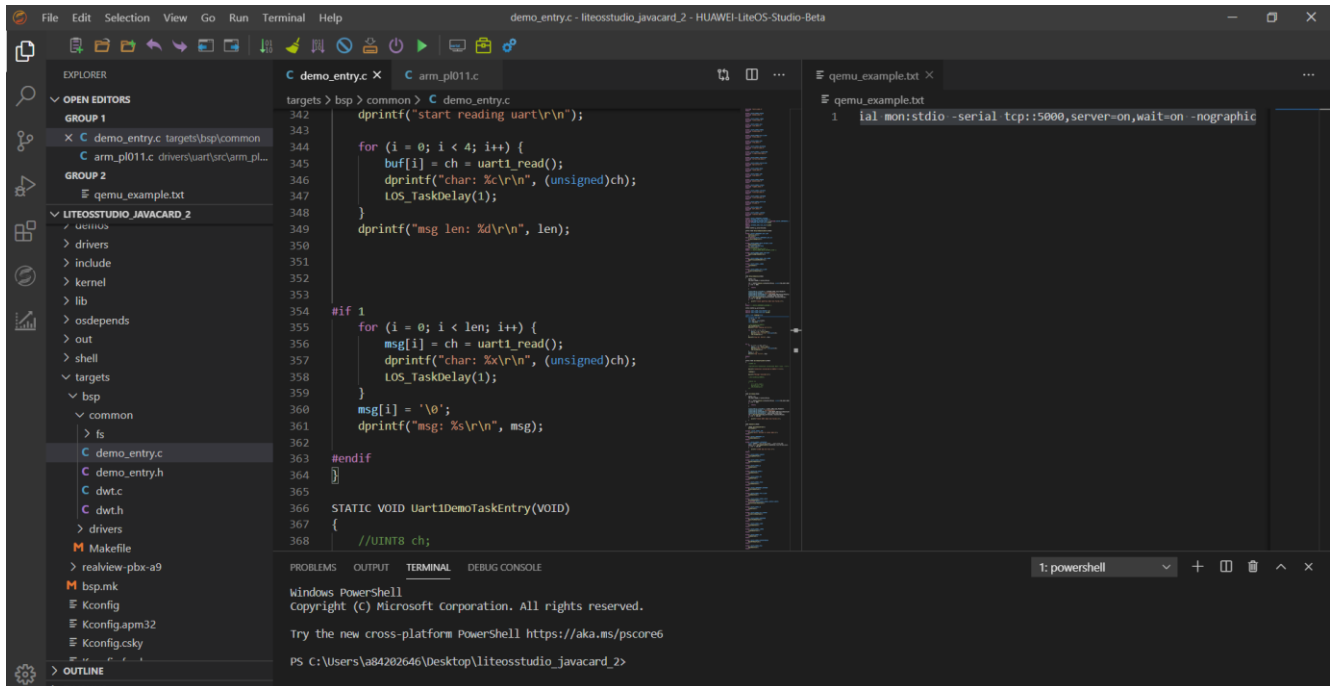


Figure 10. LiteOS studio simplifies life a lot when you're using Windows for development. Figure shows the LiteOS studio, which looks a lot like VS Studio.

Skill development: Improved my C skills by analyzing what errors the GNU make compiler threw and trying to fix them.

### Wednesday February 23<sup>rd</sup>, 2022.

Because using the network components in LiteOS studio gives an error, the keyboxes cannot be fetched directly to the LiteOS device (in emulation). I.e. the sockets code cannot be placed directly on the device, an intermediate app is needed, as I commented in chapter 1. This intermediate app is `hostpipe_cpp`.

Objectives:

- Edit `hostpipe_cpp` code in Visual Studio.



### Tasks:

1. Use `hostpipe_cpp` with my C code to download the keybox and save it locally.
2. Open it as `char[]`, send the length of the `char[]` to LiteOS, and then the `char[]` itself.

### Task outcomes:

1. I have put the code to download and save the keyboxes to `hostpipe.cpp`, using mostly C code, but also using some CPP when convenient. There were several compilation errors but I managed to solve them.
2. I've used the sockets to send the length and later the keybox itself to the LiteOS.  

```
printf("Sending size of keybox content to the device \n\n");  
sockResult = send(socketapdu , num_char , 4, 0 );  
printf("Sending the actual keybox to the device\n\n");  
sockResult = send(socketapdu , (char*)buffer2, length , 0 );
```
3. Unfortunately, this code keeps hanging on practice when I try to send the length of the keybox to the LiteOS device emulated on Qemu, and I don't know the reason for this bug.

### Thursday February 24<sup>th</sup>, 2022.

I will plug some code into the LiteOS device, and `hostpipe`, and try to make them both communicate (as LiteOS – server communication directly is not possible).

### Objectives:

- Try to plug my demo code in targets > `demo_entry.c` on LiteOS, to read from `hostpipe`.

### Tasks:

1. Build LiteOS image, run it through Qemu, parallelly run the previously compiled `hostpipe` on localhost.
2. See if the transmission, via sockets interface, is going as intended.

### Task outcomes:

1. On LiteOS I've used while loops to read the length and then the keybox itself. The while commands have `uart1_read()`; function in them which reads one char at a time. I then add that char to an array of chars `keyboxLength[i]`.
2. Unfortunately, as I've mentioned before, this code keeps hanging on practice when I try to send the length of the keybox to the LiteOS device emulated on Qemu, and I don't know the reason for this bug.

3. Either way, if the LiteOS device does read the length, I use atoi to parse the char[] to an int u, and create another char[] of the length u to take the content of the keybox.
4. This char array is called bufWrite[u] and has this code.
5. 

```
bufWrite[o] = ch;
dprintf("0x%02x", (int)bufWrite[o]);
```

The writing happens while  $o < u$ , with o starting at 0 and increasing by 1 each while cycle.

I consider my objectives and tasks accomplished for today. Objective 4 is partially accomplished as the transmission gets stuck continuously due to some unknown bug. Tomorrow I will try to save the keybox to LiteOS device using the file system component and try to make it so that everything works seamlessly.

Skill developments: as you can see I have manipulated some C code today, using loops, uart1\_read() and ramfs functions. I am improving in the C language.

### **Friday February 25<sup>th</sup>, 2022.**

Continuing with my objective of transferring the keybox data to LiteOS device, then saving the keybox there.

Objectives:

- Activate the ramfs on LiteOS, save the keybox on LiteOS device emulated on Qemu
- If it doesn't work, ask tutor to look at my code

Tasks:

1. Activate the filesystem
2. On LiteOS use the following code:

```
int bufLen;
char bufRead[DEMO_BUF_LEN];
bufLen = u;
printf("Saving the keybox to .dat file.... \n");
write_file("/ramfs/keybox.txt", bufWrite, bufLen);
memset_s(bufRead, DEMO_BUF_LEN, 0, DEMO_BUF_LEN);
read_file("/ramfs/keybox.txt", bufRead, bufLen);
printf("%s\n", bufRead);
```



```
dump mem around SP:0x8e7e0
0x8e7a0 :0000000e aaa00000 0000000f aaa00000
0x8e7b0 :00000000 40000000 0008e7c0 0004bc64
0x8e7c0 :00020bd8 00000000 00000000 00000001
0x8e7d0 :40000000 00015144 0000f400 20000133
0x8e7e0 :00000004 000767bc 00020bd8 00000001
0x8e7f0 :00000000 00000002 0008e814 00012380
0x8e800 :0a0a0a0a 00000004 00000000 0004cf08
0x8e810 :0b0b0b0b 00028198 fff717e7 00000040
dsound: Could not stop playback buffer
dsound: Reason: The buffer memory has been lost and must be restored
```

Figure 14. Buffer memory has been lost.

Figure 14 shows the error that the emulated LiteOS device is throwing.

Unfortunately, despite the ramfs system working fine, the transmission is not working and there's some memory dump on the LiteOS side.

4. Asked tutor, he made a way simpler code for using the sockets interface. I'll analyse how the code works tomorrow, and try to see if I can accomplish my goals using that code.

Skill development: A lot of trial and error using C to transmit the keybox. I felt like my mind was clouded today. Hopefully next week I can make some progress.

#### **Weekly observations week 4 (Calendar week 8)**

The sequence diagram below (Figure 15) is how everything is supposed to work. Hostpipe CPP should communicate with the Java server using sockets, and transmit information using `uart1_read()` command to LiteOS. Unfortunately, on practice it doesn't seem to be working the same way.

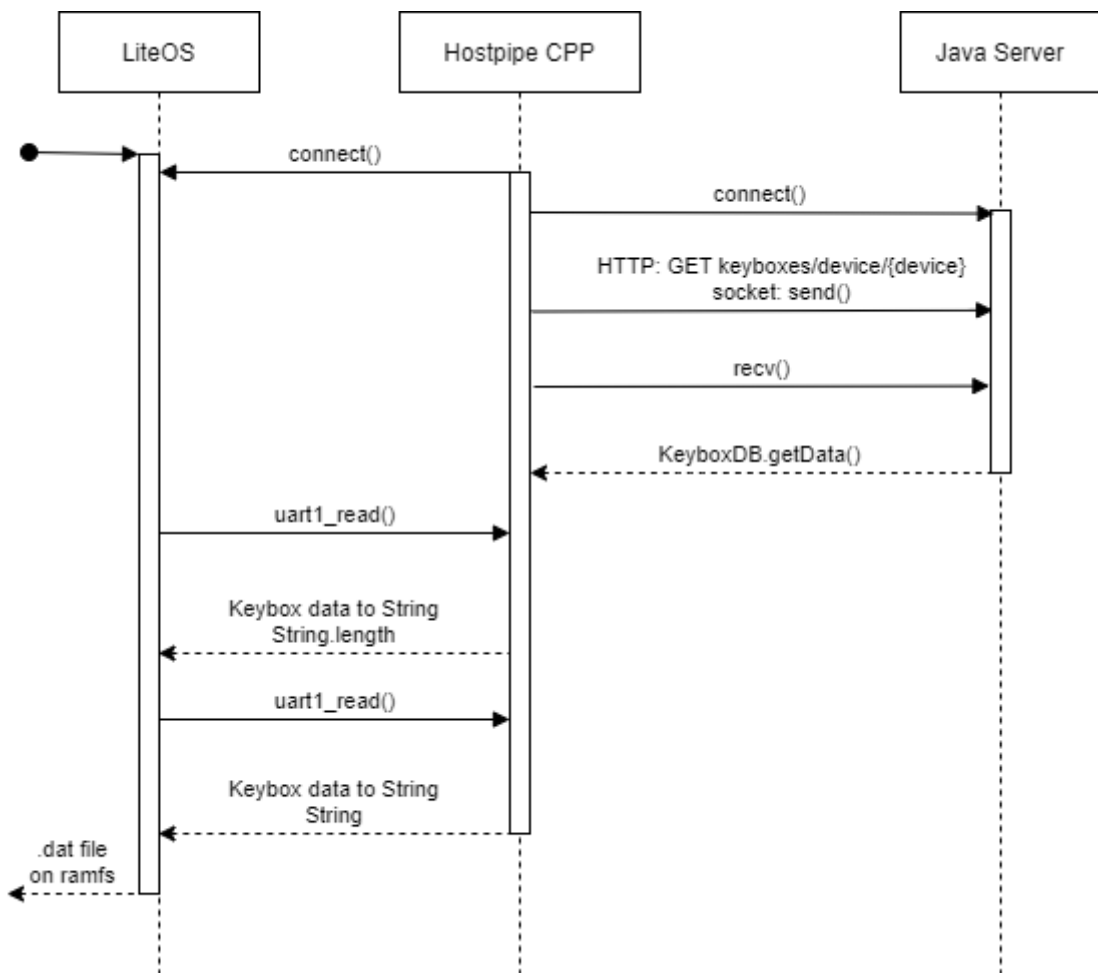


Figure 15. Sequence diagram of the LiteOS to Hostpipe to Java Server connection.

I asked my colleague about the errors but he said he never saw errors like that. Next week I'm going to talk to my tutor and see if I can solve this problem. If I can't, I am going to improve the front-end to the server, with React Native.

### 3.6 Observation week 5

**Monday February 28<sup>th</sup>, 2022.**

My tutor gave me a simpler C code for the transmissions. If I understand how it works, maybe I can solve the previous bug.

Objectives:

- Analyse the code tutor gave me and make a simple drawing of how it works using draw.io
- Use the code to transmit an example length of char[], and then the char[] itself to the LiteOS device.

Tasks:

1. Draw in draw.io

2. Make a test use of the code.

Task outcomes:

1. My tutor coded some simple functions in the `hostpipe_cpp` to send messages using sockets once the `connect()` function has been executed. The `connect()` function already worked well from my previous code so there was no reason to really change it.

```
int sendmsg(int sock, const char *buf, int len)
{
    int ret;

    ret = send(sock, (char*)&len, sizeof(len), 0);
    printf("ret1: %d\n", ret);
    // send msg
    ret = send(sock, buf, len, 0);
    printf("ret2: %d\n", ret);
    return 0;
}
```

On LiteOS there is now two *for* loops, that read first the length of the message, and then the content itself, similarly to what I had in the code previously.

```
for (i = 0; i < 4; i++) {
    buf[i] = ch = uart1_read();
    dprintf("char: %c\n", (unsigned)ch);
    LOS_TaskDelay(1);
}
dprintf("msg len: %d\n", len);
```

`#if 1`

```
for (i = 0; i < len; i++) {
    msg[i] = ch = uart1_read();
    dprintf("char: %x\n", (unsigned)ch);
    LOS_TaskDelay(1);
}
msg[i] = '\0';
dprintf("msg: %s\n", msg);
```

`#endif`

- I made a simple scheme in draw.io of how the code works (Figure 16), it's not very technical (or good looking) but it helps me understand the basics. Basically it shows the `sendstr()` and `sendmsg()` functions that my tutor coded in the `hostpipe`, and the `uart1_read()` function that tries to read these sent messages on LiteOS (`demo_entry.c` is a file in the LiteOS build).

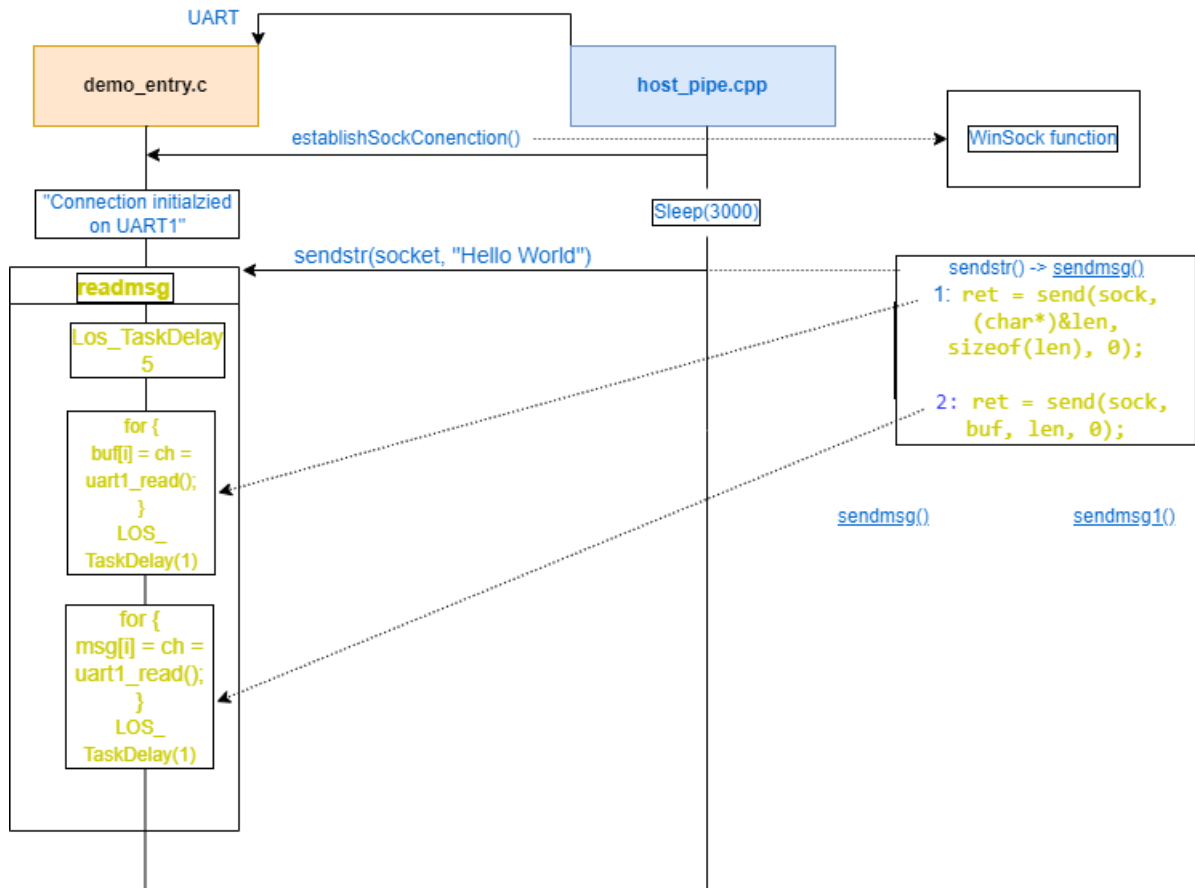


Figure 16. Simple socket communication between LiteOS and Hostpipe. Visually it's not very pleasant, but it serves its purpose. See the explanation above.

- I tried to send a simple "Hello world" message to the emulated LiteOS device. See the results in Figure 17.

```
sendmsg(socket, "Hello World", 11);
```

The LiteOS should first read the length then the message and then print them.

```

*****Hello Huawei LiteOS*****
liteOS Kernel Version : 5.1.0
Processor : Cortex-A9 * 4
Run Mode : SMP
GIC Rev : GICv1
Build time : Jan 10 2022 10:04:22
*****

main core booting up...
psAppInit
releasing 3 secondary cores
cpu 3 entering scheduler
cpu 0 entering scheduler
cpu 1 entering scheduler
cpu 2 entering scheduler
app init!
connection initialized on UART1!!!!

Huawei LiteOS # start reading uart
ERR] uart1 handler: b
ERR] uart1 handler: 0
ERR] char: 0
uart1 handler: 0

src/host_pipe.cpp:144:30: warning: unused variable 'losRecvBufLen' [-Wunused-variable]
uint32_t losTotRecv = 0, losRecvBufLen = DEFAULT_BUFLEN;
src/host_pipe.cpp:145:10: warning: unused variable 'losRecvBuf' [-Wunused-variable]
char losRecvBuf[DEFAULT_BUFLEN];
g++ -std=c++17 -Wall -Wextra -g -U _STRICT_ANSI_ -Iinclude -o output\host_pipe.exe src\host_pipe.o -L"C:\Program Files (x86)\Windows Kits\10\Lib\10.0.22000.0\um\x86" -lWS2_32 -lIib
Executing 'all' complete!

C:\Users\A84202646\Desktop\liteos_javacard_pipe_2>cd output
C:\Users\A84202646\Desktop\liteos_javacard_pipe_2\output>host_pipe.cpp 5000 localhost
'host_pipe.cpp' is not recognized as an internal or external command,
operable program or batch file.
C:\Users\A84202646\Desktop\liteos_javacard_pipe_2\output>host_pipe.exe 5000 localhost
getaddrinfo failed with error: 10109
ret1: -1
ret2: -1
^C
C:\Users\A84202646\Desktop\liteos_javacard_pipe_2\output>
C:\Users\A84202646\Desktop\liteos_javacard_pipe_2\output>host_pipe.exe localhost 5000
LOS <-> SERVICE connection established
ret1: 4
ret2: 11

```

Figure 17. Transmission is working but can't make sense of LiteOS console. LiteOS console on the left, hostpipe file on the right.

Unfortunately, the reality was a bit disappointing, I was told that the messages are sent in binary. I will analyse what this all means tomorrow.

## Tuesday March 1st, 2022.

Trying to make sense of yesterdays errors.

### Objectives:

- Analyse yesterday's socket transmission, what it means.
- See if I can make LiteOS emulation print the size of content somehow.
- See if I can make LiteOS emulation print the content itself, without getting stuck.

### Tasks:

1. Analyse yesterday's code results
2. Adjust yesterday's code results

### Task outcomes:

1. Yesterday, LiteOS printed what Figure 18 shows in the console. The error it was printing is not really an error (Figure 19).

```

*****Hello Huawei LiteOS*****
liteOS Kernel Version : 5.1.0
Processor : Cortex-A9 * 4
Run Mode : SMP
GIC Rev : GICv1
Build time : Jan 10 2022 10:04:22
*****

main core booting up...
psAppInit
releasing 3 secondary cores
cpu 3 entering scheduler
cpu 0 entering scheduler
cpu 1 entering scheduler
cpu 2 entering scheduler
app init!
connection initialized on UART1!!!!

Huawei LiteOS # start reading uart
ERR] uart1 handler: b
ERR] uart1 handler: 0
ERR] char: 0
uart1 handler: 0

src/host_pipe.cpp:144:30: warning: unused variable 'losRecvBufLen' [-Wunused-variable]
uint32_t losTotRecv = 0, losRecvBufLen = DEFAULT_BUFLEN;
src/host_pipe.cpp:145:10: warning: unused variable 'losRecvBuf' [-Wunused-variable]
char losRecvBuf[DEFAULT_BUFLEN];
g++ -std=c++17 -Wall -Wextra -g -U _STRICT_ANSI_ -Iinclude -o output\host_pipe.exe src\host_pipe.o -L"C:\Program Files (x86)\Windows Kits\10\Lib\10.0.22000.0\um\x86" -lWS2_32 -lIib
Executing 'all' complete!

C:\Users\A84202646\Desktop\liteos_javacard_pipe_2>cd output
C:\Users\A84202646\Desktop\liteos_javacard_pipe_2\output>host_pipe.cpp 5000 localhost
'host_pipe.cpp' is not recognized as an internal or external command,
operable program or batch file.
C:\Users\A84202646\Desktop\liteos_javacard_pipe_2\output>host_pipe.exe 5000 localhost
getaddrinfo failed with error: 10109
ret1: -1
ret2: -1
^C
C:\Users\A84202646\Desktop\liteos_javacard_pipe_2\output>
C:\Users\A84202646\Desktop\liteos_javacard_pipe_2\output>host_pipe.exe localhost 5000
LOS <-> SERVICE connection established
ret1: 4
ret2: 11

```



Figure 18. What was printed yesterday in the LiteOS console. Left emulated LiteOS, right the host-pipe.

```
STATIC VOID Uart1Handler(VOID)
{
    UINT8 ch = UARTREG(UART1_REG_BASE, UART_DR);
    PRINT_ERR("uart1 handler: %x\n\r", (unsigned)ch);
    (VOID)LOS_QueueWriteCopy(g_uart1Queue, &ch, sizeof(UINT8));
}
```

Figure 19. [ERR] is not really an error. The Uart1Handler uses the PRINT\_ERR function for some reason. Also, it prints %x not %c, which might be the reason why it is printing the message in binary.

2. I repeated executing both the host\_pipe and LiteOS code, and unfortunately the dreaded memory dump is still there (Figure 20), even with this simplified code.

```
irq_stack 2 0x39980 0x40 0x0
irq_stack 1 0x399c0 0x40 0x0
irq_stack 0 0x39a00 0x40 0x0
exc_stack 3 0x41b40 0x1000 0x0
exc_stack 2 0x42b40 0x1000 0x0
exc_stack 1 0x43b40 0x1000 0x48
exc_stack 0 0x44b40 0x1000 0x0

dump mem around R11:589812
0x8ffb4 :00013854 200001d3 0004662c 0008ffd4
0x8ffc4 :000117fc 00000000 00045b40 0008ffe4
0x8ffd4 :0001186c 0008ffec 00045b40 0008fff4
0x8ffe4 :600001d3 0c0c0c0c 0008ffff 0008ffff
0x8fff4 :600001d3 00090014 00013928 0009001c
0x90004 :600001d3 000465e8 20000193 00090024
0x90014 :00090024 00090034 20000193 00090034
0x90024 :000174c8 20000193 00045b40 00090044

dump mem around SP:0x8ffe8
0x8ffa8 :0000000c 200001d3 0008ffc4 00013954
0x8ffb8 :200001d3 0004662c 0008ffd4 000117fc
0x8ffc8 :00000000 00045b40 0008ffe4 0001186c
0x8ffd8 :0008ffec 00045b40 0008fff4 600001d3
0x8ffe8 :0c0c0c0c 0008ffff 0008ffff 600001d3
0x8fff8 :00090014 00013928 0009001c 600001d3
0x90008 :000465e8 20000193 00090024 00090024
0x90018 :00090034 20000193 00090034 000174c8
```

Figure 20. The memory dump.

However, this time the console did print more on the screen (Figure 21), and now I can analyze it using binary converter.

```
Huawei LiteOS # start reading uart
[ERR] uart1 handler: b
[ERR] uart1 handler: 0
[ERR] uart1 handler: 0
char: 0
[ERR] char:
uart1 handler: 0
[ERR] uart1 handler: 48
cpu0 is running.
cpu1 is in exc.
cpu2 is running.
cpu3 is running.
```

Figure 21. What the console printed.

I couldn't make sense of anything except that 48 converts from HEX to "H", which could be the first letter to the "Hello World" message.

Gave up for the time being, decided to develop further the interface to the server. Technically speaking, I don't have to do the actual server to LiteOS transmission, but it still would have been interesting to put it all together.

Skill development: increased my C comprehension skills by analyzing LiteOS internal C functions. Did a lot of searching in the LiteOS code but still couldn't make sense of what's going on.

### **Wednesday March 2nd, 2022.**

Since I gave up on programming the LiteOS tool, I decided to improve the front-end to access the server. I was given the freedom to do that if I wanted. I decided to make a mobile front-end using React Native.

Objectives:

- Before I start the React native front-end determine what the C code will do if there are no keyboxes left for a device, or if the device is not in the system (i.e. requesting keyboxes for a device but they are not in the DB yet).

Tasks:

1. In case I get the C code working, I need to anticipate some errors, such as what will C client do when there are no keyboxes, or none left.

Task outcomes:

1. I decided that if the quantity of keyboxes is 0 or below, the server is going to give a 404 error, and if the keybox is not found in /keyboxes/device/{device}, the Spring boot automatically throws a 500 error. So all I needed to do was make the C code interpret the status codes.
2. I decided not to use error messages because parsing JSON with C is a pain in the lower part of the body, it's much easier to use status codes.
3. I have a function ReadHttpStatus in my code which reads the return from send() function.
4. This is the full code of the function:

```
while(bytes_received = recv(sock, ptr, 1, 0)){  
    if(bytes_received==-1){
```

```

        perror("ReadHttpStatus");
        exit(1);
    }

    if((ptr[-1]=='\r') && (*ptr=='\n' )) break;
    ptr++;
}
*ptr=0;
ptr=buff+1;

    sscanf(ptr,"%*s %d ", &status);

```

5. So, it gets the status from reading the bytes received. Either way, I put two if clauses, and it seems to have done the trick.

```

if (status == 404){
    printf("\nNo keyboxes available for this device.");
    exit(404);
} else if (status == 500){
    printf("This device is not in the system.");
    exit(500);
}

```

I consider the tasks and objectives accomplished.

**Thursday March 3<sup>rd</sup>, 2022.**

Try to code that React native front-end that I was planning to code.

Objectives:

- Start React project from scratch and plan how I can re-create the front-end for mobile devices.

Tasks:

1. My original React Front-end contains a table. Using a table for a mobile device would make the design look ugly. Find another way
2. Add fetch function from React to React native to fetch the keyboxes.

Task outcomes:

1. I finally was able to switch back to full-stack development, or well at least the front-end part, which I prefer much more than C.

2. I started a new React Native project with expo, and put my main application code inside a `<SafeAreaView>` to account for the top bar on mobile phones. I also created a style for Android because Android for some reason doesn't recognize the `SafeAreaView`. I added the style to the view using `style={styles.AndroidSafeArea}`.
3. The style itself was as follows:

```
AndroidSafeArea: {
  flex: 1,
  backgroundColor: "white",
  paddingTop: Platform.OS === "android" ? StatusBar.currentHeight : 0,
},
```
4. I decided that I'm going to use a `FlatList`, fetch the keyboxes and show them there.
5. I re-used the fetch function to load the keyboxes from uri + `"/keyboxes"`, created a new function called `loadKeyboxes()`

I consider the tasks and objectives accomplished.

Skill development: improved my React/js skills by reading the documentation on `FlatList`, styles, and by creating new working functions in the JavaScript language.

### **Friday March 4<sup>th</sup>, 2022.**

Keep improving iteratively the React Native front-end.

Objectives:

- Make the fetch function actually work and save the keyboxes to the state.
- Figure out the `renderItem` parameter in the `FlatList`.

Tasks:

1. Change the URI for the fetch function so that it works on the mobile device.
2. Create a `renderItem` function to be used within the `FlatList`.
3. Create an `<Item>` component to be used within the the `renderItem`.

Task outcomes:

1. Updated the URI to `const uri = `http://${manifest.debuggerHost.split(':')[0].shift()}:8080``; so that it works on mobile devices. The counterpoint is that it doesn't work locally now in the browser. When the load function fetches the keyboxes, it saves them to the `keyboxes` state. I can use that state directly in the `FlatList` `data={keyboxes}`.

2. The renderItem function accepts item as props, and passes them to the <Item> component. It was created, and added to the renderItem parameter within the FlatList.
3. The Item component is going to have 3 <Text> elements, with different styles, to show the most important information about the keybox item, in this way `<Text style={{styles.textDevice, textColor}}>Device: {item.device}</Text>`

The Item component is also going to have three buttons, 2 linked to modals, “View details” and “Edit item”, one linked to a function, “Delete item”.

I consider the tasks and objectives accomplished.

Skill development: Improved my React skills by coding the renderItem function and Item component.

### **Weekly observations week 5 (Calendar week 9)**

Decided to research as much about the LiteOS documentation, and the hardware used by LiteOS (Huawei N/A). The documentation is in Chinese, I used Google translate, so there might be some misinterpretations.

I followed their tutorial for emulating Realview PBX A9 on Qemu. The PBX-A9 baseboard is a highly integrated software and hardware development system based on the ARMv7-A architecture. It is supplied self-powered, in an ATX profile enclosure. The Cortex-A9 processor subsystem is located on a custom daughterboard (HBI0183). The daughterboard cannot be removed and used separately from the PBX-A9 baseboard (HBI0182) (Arm Limited or its affiliates, 1995-2022).

The documentation also says it provides general instructions for STM32. Upon researching what STM32 is, I found that they are a family of 32-bit Arm Cortex MCUs. They are “designed to offer new degrees of freedom to MCU users”. I do not know exactly what that statement means.

The STM32 chips are grouped into related series that are based around the same 32-bit ARM processor core, such as the Cortex-M33F, Cortex-M7F, Cortex-M4F, Cortex-M3, Cortex-M0+, or Cortex-M0. Internally, each microcontroller consists of the processor core, static RAM, flash memory, debugging interface, and various peripherals (STMicroelectronics, 2022; Wikipedia, 2022).

In my LiteOS studio project, the Qemu simulation uses a Cortex-A9 process. I also tried the other simulation option qemu-virt-a53 which uses Cortex-A53 to see if it compiles with the network functionality of LiteOS. It didn't.

The project is ARM GCC compiled. And so, it requires tools like arm-none-eabi software. Also, it requires GNU Make which is a pretty common tool.

The arm-none-eabi stands for GNU Arm Embedded Toolchain that is an open-source suite of tools for C, C++ and assembly programming. The GNU Arm Embedded Toolchain targets the 32-bit Arm Cortex-A, Arm Cortex-M, and Arm Cortex-R processor families. The GNU Arm Embedded Toolchain includes the GNU Compiler (GCC).

GNU Make is a tool which controls the generation of executables and other non-source files of a program from the program's source files. Make gets its knowledge of how to build your program from a file called the makefile, which lists each of the non-source files and how to compute it from other files. So, in I guess it is just GCC but for a bigger list of files, to simplify the compilation process.

Once compiled, you have an .elf image. Upon researching it stands for Executable and Linkable Format, the standard binary file format for Unix and Unix-like systems on x86 processors by the 86open project (Wikipedia, 2022). So, I assume it's more of a .exe for Unix than a .iso.

Either way, if you use burn function of LiteOS, you use the .elf, if you run qemu from command prompt, you use .bin. `qemu "C:\Program Files\qemu\qemu-system-arm.exe" -machine "realview-pbx-a9" -smp 4 -m 512M -kernel "c:/Users/a84202646/Desktop/liteosstudio_javacard_2/out/realview-pbx-a9/Huawei_LiteOS.bin" -serial mon:stdio -serial tcp::5000,server=on,wait=on -nographic`

Qemu itself is a system-emulation and virtualization software, that in my eyes looks similar to Virtual-Box or Docker. I wonder why it is preferred to those two.

From the Github documentation (Fuqiang 2020). I have also found that the base kernel of Huawei LiteOS includes a non-trimmable very small kernel and other modules that can be trimmed. Tiny kernels contain task management, memory management, interrupt management, exception management, and system clocks. The modules that can be tailored include semaphores, mutex locks, queue management, event management, software timers, etc. Huawei LiteOS supports UP (single-core) and SMP (multi-core) modes, that is, supports running on a single-core or multi-core environment.

It integrates the full set of IoT interconnection protocol stacks of LwM2M, CoAP, mbedTLS, and LwIP. It is an open-source OS used for developing the IoT community. However, on practice I have seen that close-source variants of the code contain less compilation errors.

I have also checked during my compilation journey of LiteOS a different kernel from OpenHarmony. OpenHarmony LiteOS Cortex-A brings small-sized, low-power, and high-performance experience and builds a unified and open ecosystem for developers. In addition, it provides rich kernel mechanisms,

more comprehensive Portable Operating System Interface (POSIX), and a unified driver framework, Hardware Driver Foundation (HDF), which offers unified access for device developers and friendly development experience for application developers (OpenHarmony 2021).

### 3.7 Observation week 6

**Monday March 7<sup>th</sup>, 2022.**

Continue with fetching the keyboxes and showing them in the React Native front-end.

Objectives:

- Improve the loadKeyboxes() function so that it shows keyboxes in the list in an ordered fashion.

Tasks:

1. Create new field for KeyboxDB object in Java, where it shows the data in which the keybox was created.
2. Use the Javascript.sort() function to sort the keyboxes upon fetching them, then set the state with the updated keyboxes.

Task outcomes:

1. Went to the KeyboxDB Model in Java server, added dateCreated field of the type LocalDateTime. Went to ResponseFile class (class that gets used to show JSON of the keybox to the browser), updated it also.
2. Updated the KeyboxController where necessary.
3. Went to App.js in React, added a sort function

```
function sortResults(data, prop, asc) {
  data.sort(function(a, b) {
    if (asc) {
      return (a[prop] > b[prop]) ? 1 : ((a[prop] < b[prop]) ? -1 : 0);
    } else {
      return (b[prop] > a[prop]) ? 1 : ((b[prop] < a[prop]) ? -1 : 0);
    }
  });

  return data;
}
```

Tasks and objectives have been partially accomplished, I have to put it all together now in the `loadKeyboxes()` function.

Skill development: remembered a bit more about Java, concretely editing the Model of KeyboxDB, and the ResponseFile class. Figured how to use the basic `sort()` function in Javascript.

## Tuesday March 8<sup>th</sup>, 2022.

Keep improving the load keyboxes function to load them from the server. I want the list of keyboxes to be ordered. I also want to be able to delete these keyboxes.

Objectives:

- Integrate that newly created sort function into `loadKeyboxes()` function.
- Create a `handleDelete()` function for when the delete button is pressed in an `<Item>`.

Tasks:

1. Modify `loadKeyboxes()` so that it uses the `sortResults()` function that is just an extension of the Javascript `sort()` function.
2. `handleDelete()` will be used in the `<Button>`'s `onPress` attribute. It will get the id of the item as props, then call an async function `removeKeybox` that sends a DELETE request to the server.

Task outcomes:

1. I've integrated the `sortResults()` into the `loadKeyboxes()` function as follows:

```
function loadKeyboxes(){
  fetch(uri + '/keyboxes')
  .then(response => response.json())
  .then(data => {setKeyboxes(
    sortResults(data, 'dateCreated', 1));
  });
}
```

2. Changed the button `onPress` to `handleDelete(id)`, `handleDelete` calls `removeKeybox(id)`, which  
1) calls `fetch` with DELETE to the server 2) `filter()`s the removed item from the state and sets the state to the new filtered list.

```
async function removeKeybox(id) {
  await fetch(uri + `/keyboxes/${id}`, {
```



```

    method: 'DELETE',
    headers: {
      'Accept': 'application/json',
      'Content-Type': 'application/json'
    }
  }).then(() => {
    let updatedKeyboxes = keyboxes.filter(i => i.id !== id);
    setKeyboxes(updatedKeyboxes);
  });
}

```

Skill development: improved my understanding of the Javascript sort() function, improved my understanding of async/await functions, as well as the fetch function.

### Wednesday March 9<sup>th</sup>, 2022.

I decided to improve the keybox list of React Native front-end visually. Previous days I have focused more on the technical aspects of loading the keyboxes.

Objectives:

- Style the list.
- Make it so that when an item is selected it is highlighted.
- Add a load keyboxes button that always stays on top of the list, to call the loadKeyboxes() function.

Tasks:

1. I didn't find any guidelines for fancy design, so I decided to make a simple design.
2. Selecting an item when pressed is good design-wise, but it's also necessary for later when new keyboxes are generated, to select automatically the newly generated keybox.
3. I decided not to use useEffect hook because it was giving me error of constantly reloading the DOM. So instead I'll put a button to call the function.

Task outcomes:

1. Created some styles in the StyleSheet.create component, where I previously had the AndroidSafeArea. You can see the styles applied and how they look on Android in Figure 22.
2. For the selection, I used the ? Javascript conditionals, like this:  
`const backgroundColor = item.id === selectedId ? "deepskyblue" : "lightskyblue";` I passed props to the <Item> component within the renderItem component, including backgroundColor,

textColor, and *onPress*= which calls the *setSelectedId(item.id)*. *setSelectedId* just changes the *selectedId* state.

3. Added a button to `<SafeAreaView>` above `<FlatList>`, *onPress* it calls the *loadKeyboxes()* function.

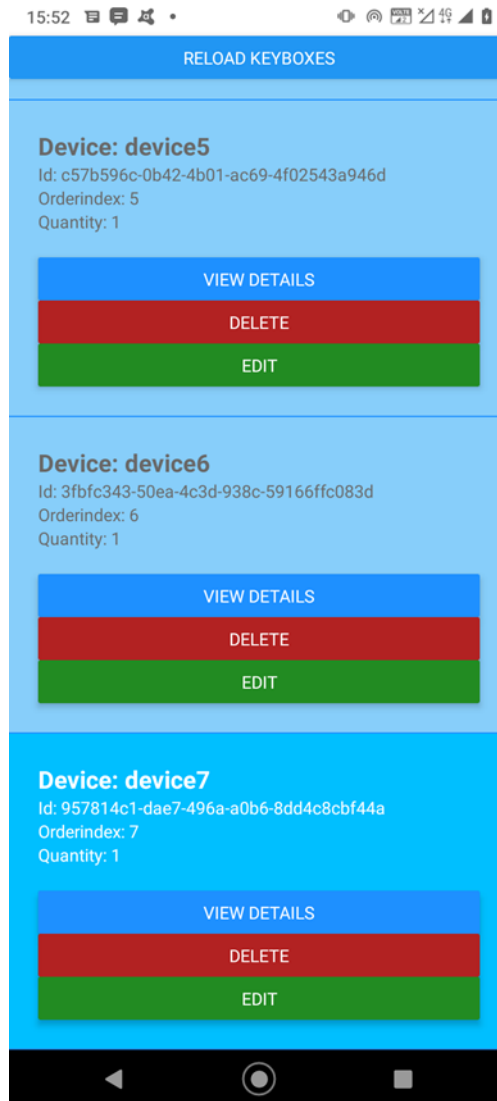


Figure 22. Current React Native interface under Android, it highlights the selected item.

Skill development: Improved my knowledge of some JS concepts like the conditional “?”. Improved my CSS styling skills by creating the stylesheet for the FlatList, Item and Button components.

**Thursday March 10<sup>th</sup>, 2022.**

Just like I created a generate keybox function in the React front-end, do the same for the mobile front-end. The generate function sends a POST request to the /generate endpoint.

Objectives:

- Create a generate keybox button that shows at top of the list of keyboxes.
- Create a generate function that sends POST request to server, then fetches the list of keyboxes again, updating the view. It should also scroll to the end of the list.

#### Tasks:

1. Add list header to the list, place the generate button there.
2. Create async generateKeybox(), with two await fetches first POST generate, second GET all the keyboxes.
3. Once the generateKeybox() function runs it should scroll to the end of the list, where the newly generated keybox is added.

#### Task outcomes:

1. Added a header to the FlatList using `ListHeaderComponent={listHeader}`, also added `ref={list}`, necessary for later scrolling to the bottom of the list in the generateKeybox() function.

2. Coded the generateKeybox() function.

3. For scrolling towards the end of the list

```
const list = useRef(null);
setTimeout(() => {
  list.current.scrollToEnd({ animated: true });
}, 500);
```

4. The `list.current.scrollToEnd` scrolls it to the end, but I need to wait for the fetching of the updated list of keyboxes to take place, that is why the timeout was necessary.

Skill development: developed my Javascript skills slightly, by using the `setTimeOut()`, `useRef()` functions. Also coded the `generateKeybox()` function.

#### **Friday March 11th, 2022.**

Decided to do some minor changes, such as update the React Native code, update the React front-end code, and attend the Friday coffee to get updated on the current state of things.

#### Objectives:

- Update the code so that the keybox that's been generated is selected automatically in the list.
- Update the original browser front-end to automatically update when the new keybox is generated in the mobile app.
- Attend the Friday coffee.

## Tasks:

1. This is a bit problematic; I have to change the selectedId state to the id of the newly generated keybox but I am not sure at which point to change it.
2. Update the front-end so that it shows the newly generated keybox. This should be easy using just the useEffect hook.

## Task outcomes:

1. I tried to set selectId when the fetch function runs, with no luck, I've been trying to set it when generate button is pressed. Finally, I decided to go with the useEffect() function, as I figured it would run when the keyboxes state would be updated, and it gets the id of the last keybox (so length of keybox array -1). This is the code:

```
useEffect(() => {  
  setSelectedId(keyboxes[keyboxes.length-1].id);  
}, [keyboxes]);
```

2. I also decided to add useEffect to the React/browser front-end using this function. Except I remembered that the browser front-end used function based react, so I had to use the componentDidMount() function. It seems to have done the trick.

```
componentDidUpdate() {  
  fetch('/keyboxes')  
    .then(response => response.json())  
    .then(data =>  
      this.setState({ keyboxes: data })  
    );  
}
```

Skill development: made some connections between class-based React and function-based React. I.e. the componentDidMount() and componentDidUpdate() functions in class based react, and useEffect() function in function-based react.

## Weekly observations week 6 (Calendar week 10)

I decided to read more about React hooks, such as useRef, useEffect, useCallback. Hooks let you split one component into smaller functions based on what pieces are related (such as setting up a subscription or fetching data), rather than forcing a split based on lifecycle methods, how you would do with componentDidMount() or componentDidUpdate() in class-based React (React 2022).

`useEffect`, you're telling React to run your "effect" function after flushing changes to the DOM. Effects are declared inside the component, so they have access to its props and state. By default, React runs the effects after every render — including the first render. So, both on mount and on update.

`useRef(initialValue)` is a built-in React hook that accepts one argument as the initial value and returns a reference (aka ref). A reference is an object having a special property `current`. Updating a reference doesn't trigger a component re-rendering.

```
const reference = useRef(initialValue);
const value = reference.current;
reference.current = newValue;
```

Different function objects sharing the same code are often created inside React components:

```
function MyComponent() {
  // handleClick is re-created on each render
  const handleClick = () => {
    console.log('Clicked!');
  };
  // ...
}
```

`handleClick` is a different function object on every rendering of `MyComponent`. But sometimes you need to retain a single function between the re-renderings. `useCallback` is a good hook in this case.

```
const handleClick = useCallback(() => {
  console.log('Clicked!');
}, []);
```

React also has other hooks, such as `useContext`, `useMemo`, `useReducer`, `useImperativeHandle`, `useLayoutEffect` and `useDebugValue`. I haven't seen the other ones used so much, except `useMemo` and `useReducer`.

The React `useMemo` Hook returns a memoized value. Memoization is caching a value so that it does not need to be recalculated. The `useMemo` Hook only runs when one of its dependencies updates.

`useReducer` is an alternative to `useState`. Accepts a reducer of type `(state, action) => newState`, and returns the current state paired with a `dispatch` method. It is closely related to `Redux`. An example they show is this, which is similar to the `Redux` logic.

```
const initialState = {count: 0};
```

```
function reducer(state, action) {  
  switch (action.type) {  
    case 'increment':  
      return {count: state.count + 1};  
    case 'decrement':  
      return {count: state.count - 1};  
    default:  
      throw new Error();  
  }  
}
```

```
function Counter() {  
  const [state, dispatch] = useReducer(reducer, initialState);  
  return (  
    <>  
    Count: {state.count}  
    <button onClick={() => dispatch({type: 'decrement'})}>-</button>  
    <button onClick={() => dispatch({type: 'increment'})}>+</button>  
    </>  
  );  
}
```

(React 2022).

### 3.8 Observation week 7

#### Monday March 14th, 2022.

Work further on React Native front-end appearance by creating modals. For some reason I feel this is exciting because it would use a mobile layout. Being able to create modals on mobile is probably essential to any mobile app, as well as navigation/router, which I might later add as well.

Objectives:

- Create a mobile (React Native) modal to show further details about keyboxes.

Tasks:

1. Attempt to create a modal.

Task outcomes:

1. It was a bit confusing at first, but nothing too complicated. Basically the `<Modal>` component would always be present, and it would have a “visible=” property, linked to a state. The state would be changed from false to true and then the Modal would become visible.
2. For some reason I have to put the `<Modal>` component inside another const though, I used `ViewDetailsModal` const.
3. The button to “View details” is in the `<Item>` component, each Item representing an item from the Flatlist (and therefore one keybox).
4. Now here’s something I could not understand. Within the Item component, you have the item prop, passed by `renderItem`. Each item prop is an object from the FlatList. So, within the Item component, you could write `<Text>{item.name}</Text>`, and it would show the name of a specific item.
5. If, however, I try to pass the item prop to the Modal, it passes the whole array of items.
6. I’ve been trying to figure for some time why it is like it is, but didn’t find it out. I found a workaround which I will implement tomorrow.
7. I also created another Modal for editing the keybox details, under the `ViewDetailsModal` const.

I consider the tasks and objectives accomplished.

Skill development: A little bit of a mess today with how React handles props. However, I learned more about it. I hope I can figure it fully out – how React hooks and props function.

**Tuesday March 15th, 2022.**

Keep working on the modal. Style it a little bit.

Objectives:

- Pass the properties of single item to the Modal, i.e. quantity, url, used status. So that later, when you click on “View details” for any item within the FlatList, you can see these properties.
- Style the modal.

Tasks:

1. I figured out I can do this with using an object for a state instead of a single property like visible.

2. This will be done using simple `<Text>`, `<Visible>` and `<View>` elements. They will have styles like `fontWeight`, `textAlign`, `padding` and `margin`.

#### Task outcomes:

1. I had before a state for changing the visibility of the modal, `modalVisible` and `setModalVisible`, which was linked to the `visible=` property of the modal. I changed it to an object, `useState({visible: false, id: "", name: "", url: "...})`. So, now it contains the property `visible` along with properties of the item we want to show in the item.

I think it's far from optimal and looks ugly style-wise, but it works. When the button View Details is pressed, `visible` status along with item properties get passed to the state.

I've also updated the name of the state from `modalVisible` to `detailsVisible`, to contrast it with the other modal, `editVisible`. I am not sure why I cannot just use the props and have to technically save the props to state, which I'm sure violates many React guidelines.

2. Having done that I can now have `<Text>{item.property}</Text>` elements within the modal. It also has a `<Pressable>` element, which sets the `visible` to `false`, therefore closing the modal. Figure 23 has the current code to the modal. I will show how the modal looks in the upcoming days, when the styling is finished. Styling will be finished tomorrow, I was busy figuring how to pass the item prop/single item to the Modal today for another day.

I consider the tasks and objectives accomplished, except for the styling objective, which is a minor thing.



```

const ViewDetailsModal = () => {
  return (
    <Modal
      animationType="slide"
      transparent={true}
      visible={detailsVisible.visible}
      onRequestClose={() => {
        setDetailsVisible(!detailsVisible);
      }}
    >
      <View style={styles.centeredView}>
        <View style={styles.modalView}>
          <Text><Text style={{ fontWeight: 'bold' }}>Id:</Text> {detailsVisible.id}</Text>
          <Text><Text style={{ fontWeight: 'bold' }}>Name:</Text> {detailsVisible.name}</Text>
          <Text><Text style={{ fontWeight: 'bold' }}>URL:</Text> {detailsVisible.url}</Text>
          <Text><Text style={{ fontWeight: 'bold' }}>Device:</Text> {detailsVisible.device}</Text>
          <Text><Text style={{ fontWeight: 'bold' }}>Used:</Text> {detailsVisible.used}</Text>
          <Text><Text style={{ fontWeight: 'bold' }}>Quantity:</Text> {detailsVisible.quantity}</Text>
          <Text><Text style={{ fontWeight: 'bold' }}>Date created:</Text> {detailsVisible.dateCreated}</Text>
          <Pressable
            style={[styles.button, styles.buttonClose]}
            onPress={() => setDetailsVisible({visible: !detailsVisible.visible})}
          >
            <Text style={styles.textStyle}>Hide Modal</Text>
          </Pressable>
        </View>
      </View>
    </Modal>
  );
};

```

Figure 23. The Modal code. Not great, not terrible.

### Wednesday March 16<sup>th</sup>, 2022.

Style the modal so I can show how the code in Figure 23 looks on practice. Apply same style to the second, EditDetails modal.

#### Objectives:

- Quickly style everything.
- Apply the same style to the EditDetails modal.

#### Tasks:

1. Just edit the StyleSheet.create with some styles.
2. Duplicate the code for EditDetails modal, except this time instead of <Text> use <Input> to change the existing values. Fetching function will have to be used probably, with one GET request to fetch the existing values, and one POST request to post the updated value.

#### End of the day.

#### Task outcomes:

1. Created styles like these:

```

modalView: {
  margin: 20,
  backgroundColor: "white",

```

```
borderRadius: 20,  
padding: 35,  
alignItems: "center",  
shadowColor: "#000",  
shadowOffset: {  
  width: 0,  
  height: 2  
},  
shadowOpacity: 0.25,  
shadowRadius: 4,  
elevation: 5  
},  
button: {  
  borderRadius: 20,  
  padding: 10,  
  elevation: 2  
},
```

You can see how the styled modal looks in Figure 24.

2. I've created a similar Modal with similar style for editing the keybox. However, I decided that for the actual editing keybox I decided to re-use the React component from the browser front-end. Recall that it was coded in function based React, so the coding is slightly different.

And so, I added another component to my /components/ directory, EditKeybox.js, and used it within the Modal as `<EditKeybox />`. I decided that I only need to pass the item.id as props to EditKeybox so that it can fetch the keybox in question by id. The rest of the properties it can get from the rest function. So, `<EditKeybox id={editVisible.id}/>`. I will look more into the editing functionality itself tomorrow.

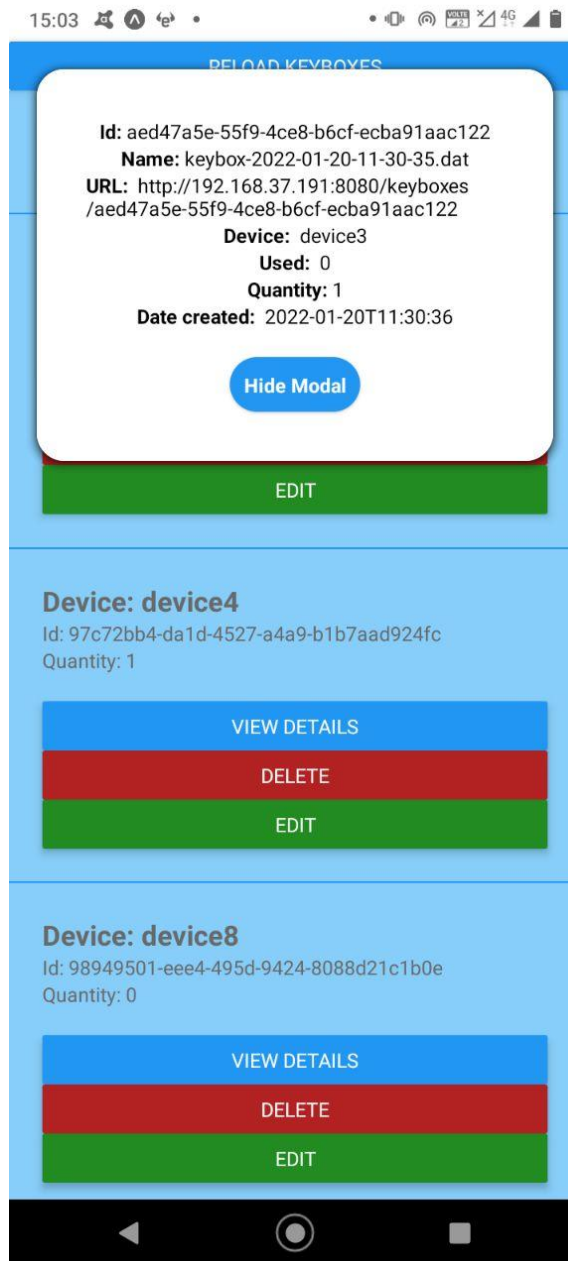


Figure 24. The modal style. It works for every item, no matter how many in the list. The Modal always show the details of the item where “View Details” is clicked, no errors.

Skill development: learned more about React Native through coding the modal.

**Thursday March 17<sup>th</sup>, 2022.**

I decided to incorporate the edit keybox functionality to the React Native front-end, just like it works in the regular web front-end. This includes creating a form specifically for the React Native front-end.

Objectives:

- Adapt the browser front-end code to edit keyboxes to React Native.
- Create the form itself and see how it looks in the modal

Tasks:

1. Analyze the browser front-end code and see if it can be re-used
2. For the form design, as I mentioned previously, just use `<TextInput>` and `<Text>` instead of just text. For some reason according to React Native practices it's better to use `<TextInput>` instead of just the `<Input>`

## End of the day.

### Task outcomes:

1. I analyzed the code and it was giving many compile errors trying to translate it to React Native, I decided to start it from scratch. I found you can use some hooks to make the forms. I found that instead of creating `handleChange` functions for each field, you can use a function like this. I've never seen it being used before:

```
const onChangeField = useCallback(  
  name => text => {  
    setValue(name, text);  
  },  
  []  
);
```

This `useCallback` hook should probably be explored in weekly observations. Furthermore, you can just register the different fields of the form in this way:

```
useEffect(() => {  
  
  register('name');  
  register('device');  
  register('used');  
  register('quantity');  
  register('content');  
  
}, [register]);
```

All I need to do to complete the `EditKeybox` component is make it fetch a keybox having the id passed to it as props from the `App.js`, and then account for the fact what does the form do when there is no keybox to edit for some reason (i.e. error handling).

2. Design was done using `React.styles`. You can see how the `Edit keybox modal` looks in Figure 25.

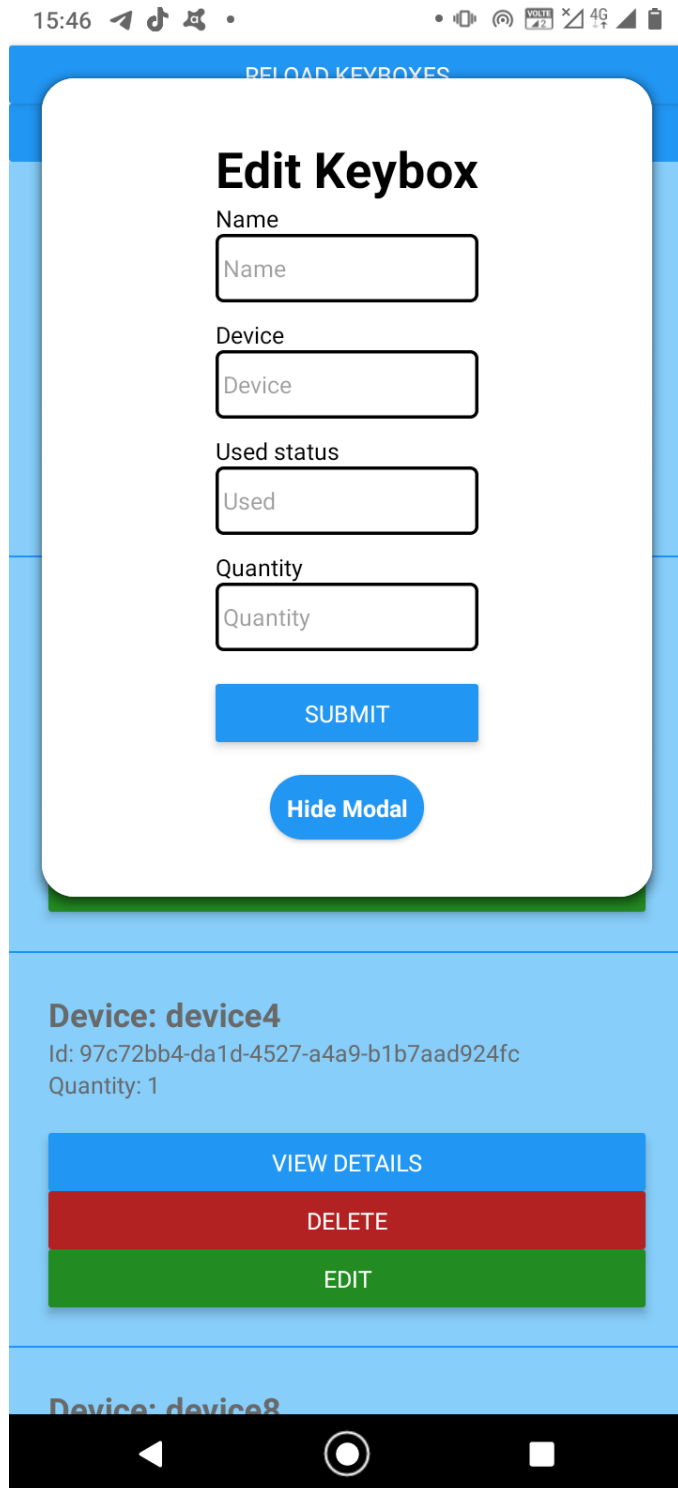


Figure 25. How the current EditKeybox modal looks.

Skill development: learned more about React Native through creating a form for editing the keybox within a modal, and using the `useCallback()` hook.

**Friday March 18<sup>th</sup>, 2022.**

To edit the keybox I need to first fetch it and load the parameters of the keybox within the form.

Objectives:

- Fetch keybox by id prop
- Account for the fact that if no keybox is fetched, an empty item should be shown with empty values inside the <TextInput>'s.

#### Tasks:

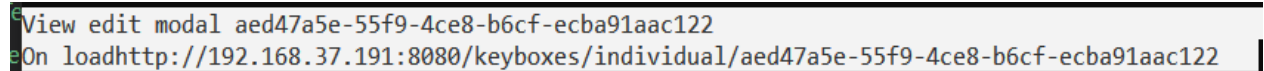
1. Make the fetch function, plug it into useEffect.
2. Create state for the keybox, with emptyItem as the default state.
3. Test that the values of the keybox are shown in the <TextInput> element.

#### Task outcomes:

1. The fetch function seems to interpret the id prop correctly. I've placed the following code in a useEffect() hook:

```
useEffect(() => {
  console.log("On load" + uri + '/keyboxes/individual/' + id);
  fetch(uri + '/keyboxes/individual/' + id)
    .then(response => response.json())
    .then(data => {setKeybox(data);
  console.log(data);
});
}, []);
```

As the console shows, the URL for fetching is the correct (Figure 26). It also should set the keybox information to the state.



```
View edit modal aed47a5e-55f9-4ce8-b6cf-ecba91aac122
On loadhttp://192.168.37.191:8080/keyboxes/individual/aed47a5e-55f9-4ce8-b6cf-ecba91aac122
```

Figure 26. Id is interpreted correctly, URL is correct.

However, it doesn't show the loaded properties of the keybox in the TextInput. Debugged it. For some reason it fetches the right keybox, but does so 4 times. Another Javascript mystery. Going to analyze it next week.

2. Anyway, for the state where the keybox is stored, I created `const [keybox, setKeybox] = useState(emptyItem)`; With emptyItem being defined as follows:

```
const emptyItem = {
  name: "",
  device: "",
  used: 0,
  quantity: 0
};
```

So, even if 4 keyboxes get loaded instead of 1 as is this case, the React Native app still won't crash. It will just show the emptyItem.

I consider my tasks and objectives accomplished, but now I have more puzzles to solve.

Skill development: improved my debugging skills by using console.log, discovered some more interesting things about React (or not so much interesting as “puzzling”), like why it doesn't plug the fetched parameters in the form.

### Weekly observations week 7 (Calendar week 11)

Decided to research more about React Native development practices, as I am basically learning by trial and error now. React Native and React components are similar, except React Native focusses more on Views.

In Android development, you write views in Kotlin or Java; in iOS development, you use Swift or Objective-C. With React Native, you can invoke these views with JavaScript using React components. At runtime, React Native creates the corresponding Android and iOS views for those components.

Main core components of the React Native framework, which translate to the corresponding views, are:

Table 1. React Native core components and comparison to view and web elements (Meta Platforms 2022).

REACT NATIVE UI COMPONENT	ANDROID VIEW	IOS VIEW	WEB ANALOG
<View>	<ViewGroup>	<UIView>	A non-scrolling <div>
<Text>	<TextView>	<UITextView>	<p>
<Image>	<ImageView>	<UIImageView>	<img>

<code>&lt;ScrollView&gt;</code>	<code>&lt;ScrollView&gt;</code>	<code>&lt;UIScrollView&gt;</code>	<code>&lt;div&gt;</code>
<code>&lt;TextInput&gt;</code>	<code>&lt;EditText&gt;</code>	<code>&lt;UITextField&gt;</code>	<code>&lt;input type="text"&gt;</code>

One view that is not listed there, and that I have used extensively, is the `ListView`. From the component standpoint, there are two components: `FlatList` or `SectionList`.

The `FlatList` component displays a scrolling list of changing, but similarly structured, data. `FlatList` works well for long lists of data, where the number of items might change over time. Unlike the more generic `ScrollView`, the `FlatList` only renders elements that are currently showing on the screen, not all the elements at once.

The `FlatList` component requires two props: `data` and `renderItem`. `data` is the source of information for the list. `renderItem` takes one item from the source and returns a formatted component to render. If you want to render a set of data broken into logical sections, maybe with section headers, similar to `UITableViews` on iOS, then a `SectionList` is the way to go (Meta Platforms 2022).

React Native also provides access to native modules. The `NativeModule` system exposes instances of Java/Objective-C/C++ (native) classes to JavaScript (JS) as JS objects, thereby allowing you to execute arbitrary native code from within JS.

### 3.9 Observation week 8

**Monday March 21<sup>st</sup>, 2022.**

I want to figure out how to fetch the keybox and load the parameters into the form inputs, so I can edit them and later save them. I believe that is because the fetch loads more than one keybox, but the two things could be not connected at all.

Objectives:

- Figure out why fetch loads four keyboxes of the same id, instead of just one.
- If I cannot figure that, figure a workaround so that one keybox gets opened for editing.

Tasks:

1. This could be because the `useEffect` is called 4 times. Analyse the code.
2. I thought about a work around, if it's an array of objects, I could just use the 0 index to get 1 keybox.

Task outcomes:

1. One thing I noticed is that the Edit modal is opened 6-7 times, and so the URL is fetched 5-6



times. However, even so, if each fetch sets the state a new, it should only set the state to one keybox. As in, the `setKeybox` function within the fetch should not concatenate the multiple responses together.

It might have more to do with the fact that I have not used the `value=` property in the `TextInput` rather than the URL fetching several times. However, now another error has occurred, and it is that when I use the `state.name` (`{keybox.name}`), for example, it says the value is undefined.

2. My workaround of using `keybox[0].name` doesn't seem to work. Interestingly enough it does print the keybox name in the `useEffect()` hook. I think I've found a workaround, I'll test it tomorrow. I consider my tasks and objectives not accomplished.

Skill development: did a lot of exploring and trial and error, have to have patience because I don't know where the bug is yet.

## Tuesday March 22<sup>nd</sup>, 2022.

I get an error of value is undefined. I need to solve it. Then continue with my task to show the keybox parameters in the form.

### Objectives:

- Use an if clause to solve the issue of the form inputs loading before the fetch has the chance to finish fetching.
- Update the keybox form to show information about the actual keybox.

### Tasks:

1. Code if clause
2. Update form components
3. Fix any errors

### Task outcomes:

1. My workaround was to use an if clause to see if `keybox !== undefined`. If such, return the form, if not, return loading. It did work. Now within the `TextInput`'s I can use the `defaultValue` property filling it with `keybox.name`, `keybox.data`, etc (the state for `EditKeybox` is `keybox`, `setKeybox`).
2. I updated the form to work as intended. `defaultValue` and `value` properties only accept string (I don't know the difference between the two), so I had to convert `used` and `quantity` to string. I will have to convert them back during the form submission.
3. I used the `toString()` function and obviously it gives an error because otherwise life would be too easy. Apparently solving it was as easy as just creating variables with `let` using `keybox.quantity.toString()` then passing the variables to the `TextInput`. You can see how the `TextInputs` look in Figure 25.

Now I realized that if I use `value` it won't let me change the actual quantity, so I need to use `defaultValue`. Now I need it to use the `useCallback` hook to post the updated keybox back to the server. I'll do that tomorrow.

I consider the tasks and objectives accomplished.

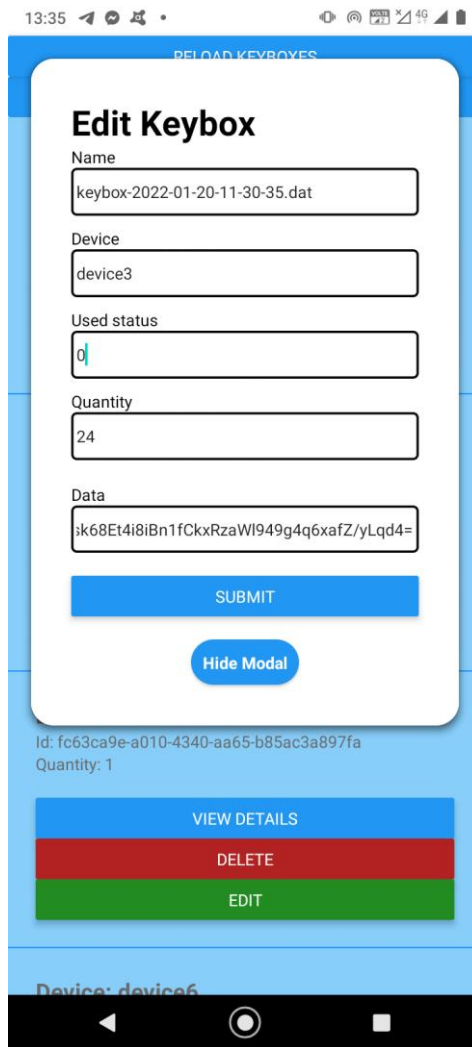


Figure 25. How the current Edit modal looks.

**Wednesday March 23<sup>rd</sup>, 2022.**

Now that the keybox information is displayed in the form, and can be edited, I need to figure a way to PUT the modified keybox back to the server. Currently the form can print the updated details to the console. Also, before the PUT request with the updated information can be sent, I need the form to print all the details to the console, not only the updated ones. I think that the whole form has to be re-made as my form uses useCallback hook which only reflects the updated values. For some reason, in React Native you can't use regular forms, you have to use dependencies.

Objectives:

- Make fetch to PUT the form to server. I.e. access the PUT endpoint
- Adjust the form so that it can support this PUT request

Tasks:

1. Re-do the form so that even values that aren't updated are printed to the console.
2. Code the handleSubmit function so that it prints the values to the console and sets them as state.
3. Update the handleSubmit function so that the PUT request is executed.

Task outcomes:

1. I have re-done the form to Formik. Now even if a value is unchanged, it still prints to the console. Same toString() etc methods had to be applied.
2. Added an async function handleSubmit that takes values from the onSubmit property of the Formik component. onSubmit={values => handleSubmit(values)}.
3. Updated the handleSubmit to do the fetch. It prints correct data to the console, but doesn't update the data on the server, and no errors neither on React side nor Java server side. I will work more on it and give an update tomorrow.

```
async function handleSubmit(values) {
  console.log(values);
  setKeybox(values);

  await fetch(uri + '/keyboxes/' + id, {
    method: 'PUT',
    headers: {
      'Accept': 'application/json',
      'Content-Type': 'application/json'
    },
    body: JSON.stringify(keybox),
  });
}
```

I consider my tasks and objectives partially accomplished.

Skill development: I have improved my Javascript coding skills by writing some code for the handleSubmit() function.

**Thursday March 24<sup>th</sup>, 2022.**

I decided to focus today on figuring out why the PUT is not working as expected.

Objectives:

- Check what is wrong with the PUT fetch in JS, diagnose any issues

Tasks:

1. Check everything, Postman, browser front-end, mobile front-end.

Task outcomes:

1. Tried sending a PUT request with Postman. It gives SSL error: *Error: write EPROTO 4090143944:error:100000f7:SSL routines:OPENSSL\_internal:WRONG\_VERSION\_NUMBER:../third\_party/boringssl/src/ssl/tls\_record.cc:242*

Also, Java server seems to be giving an error: Invalid character found in method name. However, if I edit a keybox from the browser front-end, which also sends a PUT request, it seems to work fine.

I added in the React native code `.then` clause to print the response to the console, and `.catch` clause to catch any potential errors. It is still not printing any hint on the console, except that the Android bundle of code has reloaded.

I found a workaround by using `JSON.stringify(values)`, for the values props passed to the `handleSubmit` function, instead of the state. I think the issue is again with the state not updating before running the fetch.

The change is immediately reflected on the browser front-end. However now there is a number of issues

- 1) Sometimes the mobile front-end restarts with the modal frozen.
- 2) Duplicates of keyboxes are created.
- 3) Expo says there is a problem sending log messages.

I believe the duplicates are created because I am using values instead of state, as a result, instead of modifying the existing keybox in state, new one is created. That can be solved by using keybox saved as state but then we need to change the state somehow before the fetch is executed. I am exploring Formik to see `onChangeText=` property can be used for this (or the `handleChange()` function within Formik).

I consider my tasks and objectives partially accomplished.

Skill development: by looking at the books and trying to fix workarounds, my debugging skills have improved, as well as my analytical skills in general.

## **Friday March 25<sup>th</sup>, 2022.**

Objectives:

- Make changes to `handleChange`, `handleSubmit` functions, to make sure editing the keybox works flawlessly, editing the actual keybox and not creating any duplicates.

Tasks:

1. Edit JS code client-wise.
2. Edit Java code server-wise.

Task outcomes:

1. Tried using handle functions to add the changed values to the state, it crashes the app.

```
function handleChangeQuantity(updatedQuantity) {  
    setKeybox({quantity: updatedQuantity});
```

```
}
```

I'll try to re-use the code from the EditKeybox file that I have for the front-end edit, which seems to be working fine, although it is slightly challenging as it was coded in function based React.

As I thought, the `handleChange(event)` function that is used in browser front-end cannot be used as `event` is deprecated. I updated the code to have 2 buttons, thinking that would be a good workaround. It didn't work.

Within the form, `<Button onPress={handleSubmit} title="Submit form" />`, and then within the `<Formik>` component,

```
onSubmit={values =>
  {
    setKeybox(values);
    console.log(values);
  }
}
```

Outside the component, `<Button onPress={() => handleSave()} title="Save keybox" />`

And then the function:

```
function handleSave() {

  console.log(keybox);

  fetch(uri + '/keyboxes/' + id, {
    method: 'PUT',
    headers: {
      'Accept': 'application/json',
      'Content-Type': 'application/json'
    },
    body: JSON.stringify(keybox),
  })
  .then((response) => {
    console.log(response);
  })
  .catch((error) => {
    console.log(error);
  })
}
```

```
});  
  
}
```

However, I also updated the Java server. PUT method was updated from

```
@PutMapping("/keyboxes/{id}")  
public ResponseEntity updateKeybox(@PathVariable String id, @RequestBody KeyboxDB  
dbFile) {  
    KeyboxDB existingFile = keyboxDBRepository.getByld(id);  
  
    existingFile.setData(dbFile.getData());  
    existingFile.setName(dbFile.getName());  
    existingFile.setDevice(dbFile.getDevice());  
    existingFile.setUsed(dbFile.getUsed());  
    existingFile.setQuantity(dbFile.getQuantity());  
    existingFile = keyboxDBRepository.save(dbFile);  
  
    String message = "Success.";  
  
    return ResponseEntity.ok().body(new ResponseMessage(message));  
}
```

The part `existingFile = keyboxDBRepository.save(dbFile);`  
Was updated to `keyboxDBRepository.save(existingFile);`

The code seems to work the same when it comes to the browser front-end.

Tested it on mobile, it works. Now you can edit the information about single keybox. It still gives some warnings and displays 7 modals instead of one, but I am happy that at least the core functionality works.

I will re-style the form again next Monday and see if I can do the whole action with just one button.

I consider my tasks and objectives accomplished.

Skill development: I have analyzed my Java code from weeks 1 and 2 and have revised a little bit of the Java Mapping coding. I have worked with the Formik code a little bit, improving my Javascript skills.

## Weekly observations week 8 (Calendar week 12)

More about PUT HTTP requests / in Java context.

While both PUT and POST can be used to create resources, there are significant differences between them in terms of their intended applications. According to the RFC 2616 standard, the POST method should be used to request the server to accept the enclosed entity as a subordinate of the existing resource identified by the Request-URI. This means the POST method call will create a child resource under a collection of resources.

On the other hand, the PUT method should be used to request the server to store the enclosed entity under the provided Request-URI. If the Request-URI points to an existing resource on the server, the supplied entity will be considered a modified version of the existing resource. Therefore, the PUT method call will either create a new resource or update an existing one.

Another important difference between the methods is that PUT is an idempotent method while POST is not. For instance, calling the PUT method multiple times will either create or update the same resource. On the contrary, multiple POST requests will lead to the creation of the same resource multiple times (Baeldung 2021).

So, in my case I was right to not let PUT method create duplicates of the keyboxes, because that would be more appropriate using the POST method. The example they use in Baeldung (2021) is very similar to my own:

```
@PutMapping("/addresses/{id}")
Address replaceEmployee(@RequestBody Address newAddress, @PathVariable Long id) {

    return repository.findById(id)
        .map(address -> {
            address.setCity(newAddress.getCity());
            address.setPin(newAddress.getPostalCode());
            return repository.save(address);
        })
        .orElseGet(() -> {
            return repository.save(newAddress);
        });
}
```

They get an existing resource from the repository by id, then use setters to set attributes from the request body. My code is similar, except I don't use the map function. And then they save the address resource that they found by ID. My code:

```
@PutMapping("/keyboxes/{id}")
public ResponseEntity updateKeybox(@PathVariable String id, @RequestBody KeyboxDB dbFile)
{
    KeyboxDB existingFile = keyboxDBRepository.getById(id);

    existingFile.setData(dbFile.getData());

    existingFile.setName(dbFile.getName());
    existingFile.setDevice(dbFile.getDevice());
    existingFile.setUsed(dbFile.getUsed());
    existingFile.setQuantity(dbFile.getQuantity());
    keyboxDBRepository.save(existingFile);

    String message = "Success.";

    return ResponseEntity.ok().body(new ResponseMessage(message));
}
```

I was curious about the reasons why you cannot use forms as <Input> elements in React native, or at least why nobody was using them.

From what I understood from React-Hook-Form documentation (Luo 2022), the component re-renders are an issue with forms. Using a dependency such as react-hook-forms or formik avoids this issue. Furthermore, the ability to subscribe to an individual input and form a state update is important without re-rendering the entire form is important for performance issues. From their sales pitch, they are faster than Formik at mounting, but I preferred formik as it does not use the useCallBack() method. Also react-hook-form uses the register() function which is unknown to me.

Formik is a small library that helps simplify React form code and it also takes care of validation and error messages, as well as form submission. For the purpose of my code it worked fine.



### 3.10 Observation week 9

**Monday March 28<sup>th</sup>, 2022.**

For my last day of reporting in this diary thesis, I'll make sure the edit form & modal work perfectly. Then I'll have a basic mobile front-end prototype.

Objectives:

- Style the Formik form as the previous react-native-form
- Make it so that submission of the form automatically PUTs the updated keybox information to the server, no two buttons are necessary.

Tasks:

1. Edit the Stylesheet.create element
2. Change the onSubmit function

Task outcomes:

1. This was an easy and quick to complete task. I've created the style similarly to how I did it in week 8.
2. Even if I add the function to PUT after setState in the onSubmit, it still submits the previous state, not updated. In function based React there used to be a callback option for .setState() function, but it's been deprecated.

Luckily, Kataria S. (2020) found a way to solve this problem using useEffect. `useEffect(() => {callbackFunction()}, [state])`. It worked.

It was also very handy and I will use it in my later React development projects.

Now you can edit the keyboxes, save them on your phone, and it gets reflected immediately on the browser front-end. I also wanted the mobile front-end to close the modal after edit is complete, and reload the list of keyboxes. I will see if I can achieve this today.

I consider the tasks and objectives for this project accomplished.

Skill development: by reading the material written by Kataria S. I learned how to use callback functions for the state updates in function based react, using useEffect. This will come in handy in my future React projects.

## 4 Discussion and conclusions

### How I have developed

I was hired to develop a server that can provision the keyboxes to a LiteOS device. Naturally first thing I did was start to develop the server, in Java Spring Boot, as we were taught in Haaga-Helia. I developed a front-end in React as well to connect it to it, because I have an innate drive to make a full stack system.

My learning was very quick, I believe, in both Java concepts, and C concepts. I also improved my understanding of Javascript and different elements of React, and “learned” React Native (I have not done any RN projects before, but it being very similar to regular React, there was not that much difference).

From personal skills I learned how to ask for feedback, and how to analyse and use other colleague’s code, which was more important than the technical skills – I could join in on already ongoing progress.

I also developed a strong sense of discipline from coming to work every day and just pushing, even when I felt like not doing so, which was very important for me as I used to be a person who would get random bursts of energy which slowly faded over a couple weeks. I can apply this sense of discipline now for my future projects.

*In the beginning of the thesis*, I outlined the following development objectives: focusing on connecting more with my environment in the next few months, finding out what others are working on, the bigger scheme, and if can contribute somehow. Also, seeing what headquarters’ plans are, what direction specifically they are going forward to.

I did such, as I talked extensively with my colleagues. I also attended Friday coffees in which the HQ’s plans were discussed. However, I thought they fall way out of my knowledge and profile.

Also, I outlined that I should focus on more of what my colleagues are working on – which would be focusing on my weaknesses in as sense – such as C, compilers, and OS development, which is not really my profile (I want, and studied to be a full stack developer, and my study path is not ICT infrastructures).

On the other hand, I said in the beginning that I could focus on my strengths and become better at something I am already moderately good at. That is, at developing the Java server (back-end side) or the React applications (front-end side). This is the aspect on which I focused mostly, as you can see. I

have developed React for front-end, React Native, and spent first few weeks working on the Spring Boot server and reading Spring Boot theory.

### **New solutions and methods I have found for my work**

Setting clear objectives for my day was what I found most interesting from the thesis. I previously already did something similar, writing down tasks on paper and ticking them, to give myself a sense of accomplishment. But setting objectives gave my work a bigger sense of urgency, and I also had it clearer what I needed to accomplish during the day by defining the tasks beforehand.

Reporting my progress on a daily basis also gave me a chance to re-analyze what I have done and look at it more from a theoretical perspective. Later, if there was something unclear, I could research it the next day, or when doing the weekly observations.

When you are coding, I feel like it's easy to just go into practical mode, coding and re-writing solutions from StackOverFlow without actually thinking what is behind this code. With this thesis I learned to do otherwise.

### **What I learned during this diary writing process.**

How to put into words what I am planning to do, what I am working on, and the conclusions. It is a bit hard to explain to the reader your point of view, and I am not sure if I managed to do it properly in this thesis. The reader doesn't have the knowledge of my current situation, and is not following my thoughts, neither my progress on an hourly basis. So, I have to explain in limited words what it is I am up to.

### **New issues I have identified and their future value.**

Networking issues with the open-source code of LiteOS. Some unknown bug that stops socket transmission through UART ports between a host pipe and the LiteOS. I reported the bugs to more experienced colleagues in our department, but I don't think it is on our group to fix them, more on the headquarters in China.

### **Possible further development.**

I will personally develop more in the mobile development area, as well as React front-end. Unfortunately, in Huawei they work more on the engineering side, with processors and compilers, so my goals don't match with the organization. Still, I am glad they gave me the freedom to explore different areas of coding.

## **How have you been able to make use of analysing your work?**

I developed more self-awareness. I discovered better how to plan my day and get the goals accomplished instead of leaving them halfway done. I learned to analyze what I am working on the technical side (coding) from theory perspective, instead of just blindly coding.

## References

Baeldung 2022. URL: <https://www.baeldung.com/>. Last accessed 26<sup>th</sup> January 2022.

Baeldung 2022. CRUD Application with React and Spring Boot. URL: <https://www.baeldung.com/spring-boot-react-crud>. Last accessed 31<sup>st</sup> January 2022.

Fielding, R. T. (2000). "Chapter 5: Representational State Transfer (REST)". Architectural Styles and the Design of Network-based Software Architectures (Ph.D.). University of California, Irvine.

Full stack open 2021. URL: <https://fullstackopen.com/en/>. Last accessed 26<sup>th</sup> January 2022.

Huawei 2022. Corporate Information. URL: <https://www.huawei.com/en/corporate-information>. Accessed: 21<sup>st</sup> January 2022.

Huawei 2022. Openness, Collaboration and Shared Success. URL: <https://www.huawei.com/en/corporate-information/openness-collaboration-and-shared-success>. Accessed 20<sup>th</sup> January 2022.

Johnson K. M. & Troan E. W. 2005. Linux Application Development. Second Edition. Addison-Wesley. Massachusetts.

Lippman, S. B. & Lajoie, J. 1998. C++ Primer. Third Edition. Addison-Wesley. Massachusetts.

MDN Web Docs. 2005-2022. URL: <https://developer.mozilla.org/en-US/> . Last accessed 26<sup>th</sup> January 2022.

React – a JavaScript library for building user interfaces. 2022. URL: <https://reactjs.org/>. Last accessed 26<sup>th</sup> January 2022.

React Native – Learn once, write anywhere. 2022. URL: <https://reactnative.dev/>. Last accessed 26<sup>th</sup> January 2022.

Kataria S. LinkedIn. 2020. Provide callback to useState hook like setState. URL: <https://www.linkedin.com/pulse/provide-callback-usestate-hook-like-setstate-saransh-kataria>. Last accessed 28th February 2022.

Tutorialspoint. 2022. C tutorial. URL: [https://www.tutorialspoint.com/cprogramming/c\\_overview.htm](https://www.tutorialspoint.com/cprogramming/c_overview.htm). Accessed 26th January 2022.

University of Oklahoma. 2019. Professional Development Phases. URL: <https://outreach.ou.edu/educational-services/education/edutas/comp-center-landing-page/knowledgebases/english-language-learners/ell-admin-teachers/6-1-2-investigate-different-approaches-professional-learning/professional-development-phases/>. Accessed 25th January 2022.

Helsingin Yliopisto. 2021. Full stack open. URL: <https://fullstackopen.com/en/>. Last accessed 26th January 2022.

Wikipedia. 2021. Model-view-controller. URL: <https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller>. Last accessed 7th February 2022.

Wikipedia. 2022. Representation State Transfer. URL: [https://en.wikipedia.org/wiki/Representational\\_state\\_transfer](https://en.wikipedia.org/wiki/Representational_state_transfer). Last accessed 8th February 2022.

Oracle Corporation and/or its affiliates. 2010. What Are RESTful Web Services? URL: <https://docs.oracle.com/cd/E19798-01/821-1841/gijqy/index.html>. Last accessed 8th February 2022.

OSS Nokalva. 2021-2022. ASN1 Playground. URL: <https://asn1.io/asn1playground/>. Last accessed 10th February 2022.

Tutorialspoint. 2021. C - File I/O. URL: [https://www.tutorialspoint.com/cprogramming/c\\_file\\_io.htm](https://www.tutorialspoint.com/cprogramming/c_file_io.htm). Last accessed 10th February 2022.

Programiz. 2022. C File Handling. URL: <https://www.programiz.com/c-programming/c-file-input-output>. Last accessed 10th February 2022.

Geekforgeeks. 2021. C Program to read contents of Whole File. URL: <https://www.geeksforgeeks.org/c-program-to-read-contents-of-whole-file/>. Last accessed 10th February 2022.

Geeksforgeeks. 2022. TCP Server-Client implementation in C. URL: <https://www.geeksforgeeks.org/tcp-server-client-implementation-in-c/>. Last accessed 10th February 2022.

Stackoverflow. Jeremy Smith. 2014. Simple C example of doing an HTTP POST and consuming the response. URL: <https://stackoverflow.com/questions/22077802/simple-c-example-of-doing-an-http-post-and-consuming-the-response>. Last accessed 10th February 2022.

Baeldung. 2022. Learn JPA & Hibernate. URL: <https://www.baeldung.com/learn-jpa-hibernate>. Last accessed 10th February 2022.

Baeldung. 2019. JPA entities. URL: <https://www.baeldung.com/jpa-entities>. Last accessed 10th February 2022.

Wikipedia. 2021. Bouncy Castle (cryptography). URL: [https://en.wikipedia.org/wiki/Bouncy\\_Castle\\_\(cryptography\)](https://en.wikipedia.org/wiki/Bouncy_Castle_(cryptography)). Last accessed 13th February 2022.

Wikipedia. 2022. X.509. URL: <https://en.wikipedia.org/wiki/X.509>. Last accessed 13th February 2022.

ITU. 2022. Introduction to ASN.1. URL: <https://www.itu.int/en/ITU-T/asn1/Pages/introduction.aspx>. Last accessed 14th February 2022.

Steedman, D. 1990. Abstract Syntax Notation One (ASN.1) - The Tutorial and Reference. URL: <https://www.bgbm.org/TDWG/acc/Documents/asn1gloss.htm>. Last accessed 14th February 2022.

Huawei. N/A. STM32 engineering example. URL: [https://liteos.gitee.io/liteos\\_studio/#/project\\_stm32?id=realview-pbx-a9%e4%bb%bf%e7%9c%9f%e5%b7%a5%e7%a8%8b](https://liteos.gitee.io/liteos_studio/#/project_stm32?id=realview-pbx-a9%e4%bb%bf%e7%9c%9f%e5%b7%a5%e7%a8%8b). Accessed: 04th March 2022.

Arm Limited (or its affiliates). 1995-2022. About the PBX-A9 baseboard. URL: <https://developer.arm.com/documentation/dui0440/b/introduction/about-the-pbx-a9-baseboard>. Accessed 04th March 2022.

STMicroelectronics. 2022. STM32 32-bit Arm Cortex MCUs. URL: <https://www.st.com/en/microcontrollers-microprocessors/stm32-32-bit-arm-cortex-mcus.html>. Accessed 04th March 2022.

Fuqiang, Z. 2020. Huawei LiteOS简介. URL: <https://github.com/LiteOS/LiteOS>. Accessed 04th March 2022.

OpenHarmony. 2021. LiteOS Cortex-A. URL: [https://code.opensource.huaweicloud.com/openharmony/kernel\\_liteos\\_a/home#section1579912573329](https://code.opensource.huaweicloud.com/openharmony/kernel_liteos_a/home#section1579912573329). Accessed 04th March 2022.

Wikipedia. 2022. STM32. URL: <https://en.wikipedia.org/wiki/STM32>. Accessed 04th March 2022.

Wikipedia. 2022. Executable and Linkable Format. URL: [https://en.wikipedia.org/wiki/Executable\\_and\\_Linkable\\_Format](https://en.wikipedia.org/wiki/Executable_and_Linkable_Format). Accessed 04th March 2022.

React. 2022. Hooks API Reference. URL: <https://reactjs.org/docs/hooks-reference.html>. Accessed 04th March 2022.

Meta Platforms. 2022. Core Components and Native Components. URL: <https://reactnative.dev/docs/intro-react-native-components>. Accessed 04th March 2022.

Baeldung. 2021. HTTP PUT vs. POST in REST API. URL: <https://www.baeldung.com/rest-http-put-vs-post>. Accessed 04th March 2022.

Luo, B. 2022. React Hook Form. URL: <https://react-hook-form.com/>. Accessed 09th March 2022.