Treball de Fi de Grau

GRAU D'ENGINYERIA INFORMÀTICA

Facultat de Matemàtiques i Informàtica
Universitat de Barcelona

Yieldy, a cross platform device monitoring and internal communication application for small sized enterprises

Xin Liu

Director: Prof. Raúl Rocas Cánovas
Realitzat a: Departament de
Matemàtiques i Informàtica

Barcelona, 2 de september de 2020

Acknowledgement

I wish to express my sincere gratitude to my supervisor, Prof. Raúl Rocas Cánovas, for his enthusiasm, patience, insightful comments, helpful information, practical advice and unceasing ideas that have helped me tremendously at all times in my research of this project. His immense knowledge, profound experience and professional expertise in cybersecurity have enabled me to complete this project successfully. Without his support and guidance, this project would not have been possible. I could not have imagined having a better supervisor in my study.

Abstract

The following project, called "Yieldy," is a cross-platform application that runs on Android, iOS, and web to monitor small-sized businesses' devices and allow users to communicate by posting messages, creating, and editing tasks.

It is worth mentioning that before this project, there is a previous one called "WyServer" that has almost the same task. For several reasons, the first project, "WyServer" that I designed and implemented before this final degree project, did not run as expected since it missed many requirements. To conclude, this project is developed under the deprecated "WyServer" by adding new requirements and redesigning its structure.

The application is mainly designed for companies that have a large number of devices running. "Yieldy" helps the company to manage all the devices (computers or servers) connected. It provides the possibility to monitor the device's status, such as memory usage, CPU usage, socket, and process summary... Besides, the mobile version has a real-time notification system to let users know when a task or a message is published in a company's system (the group that allows users to communicate within them).

Resum

El següent projecte anomenat "Yieldy" és una aplicació multiplataforma que funciona amb Android, iOS i web per a monitoritzar dispositius de petites empreses i també permet als usuaris comunicar-se entre ells publicant missatges, creant i editant tasques.

Val la pena esmentar que abans d’aquest projecte, n’hi havia un d’anomenat “WyServer” que té gairebé la mateixa tasca. Per diverses raons, el primer projecte “WyServer” que vaig dissenyar i implementar abans d’aquest projecte final de carrera., no va funcionar com s'esperava, ja que el projecte va desaprofitar molts requisits. Per concloure, aquest projecte es desenvolupa sota el antic project "WyServer" afegint nous requisits i redissenyant l'estructura de l'aplicació.

L’aplicació està dissenyada principalment per a empreses que tinguin en funcionament un gran nombre de dispositius. “Yieldy” ajuda l’empresa a gestionar tots els dispositius (ordinadors i servidors) connectats. Ofereix la possibilitat de monitoritzar l'estat del dispositiu, com per exemple l'ús de memòria, l'ús de la CPU, socket i el resum del procés , etc. A més, la versió mòbil disposa d’un sistema de notificació en temps real per permetre als usuaris saber quan es publica una tasca o un missatge en el sistema d’una empresa (el grup que permet als usuaris comunicar-se entre d’ells).

Resumen

El siguiente proyecto llamado "Yieldy" es una aplicación multiplataforma que funciona con Android, iOS y web para monitorizar dispositivos de pequeñas empresas y también permite a los usuarios comunicarse entre ellos publicando mensajes, creando y editando tareas.

Vale la pena mencionar que antes de este proyecto, había uno llamado "WyServer" que tiene casi la misma tarea. Por diversas razones, el primer proyecto "WyServer" que diseñé e implementé antes de este proyecto final de carrera, no funcionó como se esperaba, ya que el proyecto desperdició muchos requisitos. Para concluir, este proyecto se desarrolla bajo el antiguo proyecto "WyServer" añadiendo nuevos requisitos y rediseñando la estructura de la aplicación.

La aplicación está diseñada principalmente para empresas que tengan en funcionamiento un gran número de dispositivos. "Yieldy" ayuda a la empresa a gestionar todos los dispositivos (ordenadores y servidores) conectados. Ofrece la posibilidad de supervisar el estado del dispositivo, tales como el uso de memoria, el uso de la CPU, socket y el resumen del proceso. Además, la versión móvil dispone de un sistema de notificación en tiempo real para permitir a los usuarios saber cuándo se publica una tarea o un mensaje en el sistema de una empresa (el grupo que permite a los usuarios comunicarse entre ellos).

Index

1.Introduction

1.1 Motivation and context

Nowadays, almost every company uses their computer systems that have a larger number of computers connected, and we can consider it a "cluster" of computers. The cluster might set up a network between several office computers and a printer and scanner, either wired or wireless. An internet connection is essential too, so we can say that everything in the office will connect.

It is considered necessary to have an agile tool that allows users to monitor the system inside the office and outside, besides to prompt attention to resolving the problems that might occur. Mismanagement of vulnerabilities may affect the company and progress of a project, thus causing a decrease in the company's efficiency.

Some medium-sized or large-sized businesses have more than one office; therefore, every office has its devices connected within the local network. A company must have a tool that provides some functionalities like monitoring systems for every single computer to detect a computer's abnormal status so that when some vulnerability or abnormal status were detected, all the users will be aware of it. Despite that, there are also many people who have many devices running in different places and want to monitor their devices.

There are indeed many tools that could perform some similar tasks. The main focus will not be a competition for them but is assumed as a challenge to learn, research, and deepen knowledge of the cross-platform environment in order to create a convenient, intuitive and straightforward tool, and also to get into the cybersecurity field by research and incorporate different cybersecurity software to this tool.

1.2 Project Description

"Yieldy" stems from an interest in the cross-platform environment and research for the improvement of the existing monitoring system of servers and devices. A company needs to have a chance to monitor the device's status, such as CPU, memory, disk usage of the company, and communicate within them to identify a possible threat and create an incidence to resolve it to protect sensitive data on a device or server. It is required to have a web browser or mobile depending on the user's platform, and also a stable connection to the webserver is needed. The access is done through a web browser, tablet, or mobile phone.

The access of the web platform is immediate since there is no need for a previous installation. Contrarily, the installation is required for the mobile version, but it is more convenient for the user to receive a notification than the web version.

The platform allows users to create a group, connect devices to the system to monitor it, assign employees to the system, post messages to the system, create a task and assign employees to work on it.

This platform will focus on that all the information is "linked" between it to be navigated in a simple, pleasant way, which is useful when looking for a piece of factual information.

2. Objectives and Methodology

2.1. General objectives

This platform's main objective is to provide the necessary help in a simple, intuitive, and fast way to specific users, such as small and medium-sized enterprises and users who have many devices running in many different places and want to monitor it. In addition to improving productivity, efficiency, and response times to the different hardware or software problems of their devices presented in their daily development tasks. Moreover, to provide a communication tool within them.

2.2. Specific objectives

This project aims to investigate, acquire, and gain knowledge in the web, mobile, and cybersecurity fields to develop a tool that allows managing incidents and scan devices to detect abnormal status to be displayed for the corresponding user. To specify: the project will have to complete the following requirements:

- Allow the superuser (Company owner) to create different subgroups named system, and assign normal users to this group.
- Allow the superuser to create different types of normal users (admins and technicians), then assign them to a system.
- The web version has to develop as a SPA (single page application).
- Users can connect several devices in a system that allows the user to view its status.
- The user assigned to the system can publish messages and create tasks mentioning a specific device.

- The superuser and the administrator of the system can assign technicians to a particular task to let them work on it.
- The user interface has to be simple and intuitive.
- The page that shows the device's status has to be clear and concluding.

2.3. Methodology

This project followed the incremental model, in which the fundamental requirement is divided into various builds. Multiple development cycles take place here, making the life cycle a "multi-waterfall" cycle. Cycles are divided up into smaller, more easily managed modules. An incremental model is a type of software development model like the V-model, Agile model, etc.

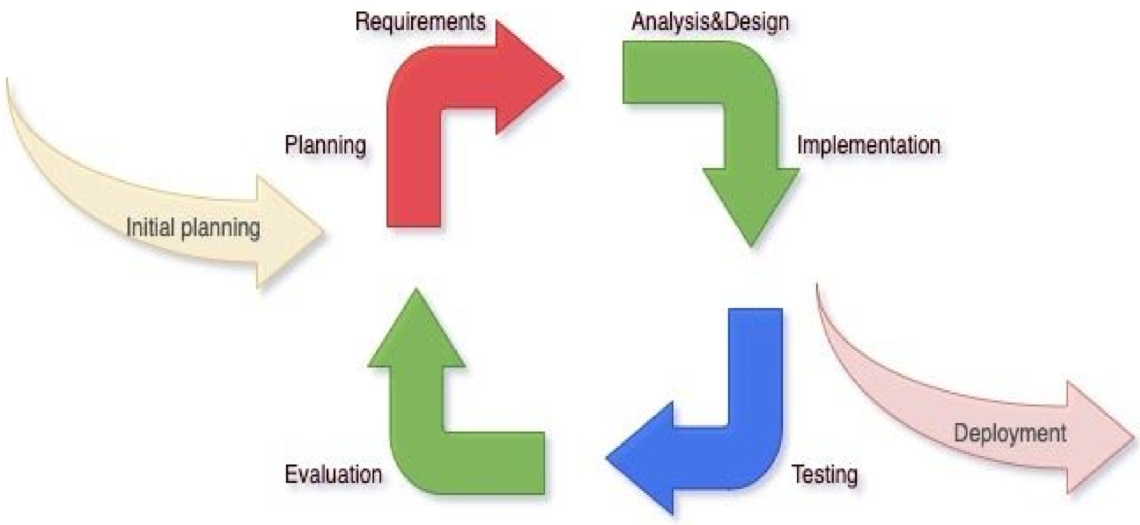


Figure 1: Incremental build model

In this model, each module passes through the requirements, design, implementation, and testing phases. A working version of the software is produced during the first module, each subsequent release of the module adds function to the previous release. The process continues until the complete system is achieved.

3. Planification

3.1. Project Phase

The project development was planned in 4 primary phases. The first phase is the pre-project phase, focusing on developing an application "WyServer" for acquiring basic knowledge of different techniques that I will use to create the final project, "Yieldy." As I mentioned before, this project "WyServer" has deprecated, and it was converted as the prototype for the final project.

Phase 2 consisted of a redesign and restructured of the prototype. To improve it, a backend that interacts with MongoDB is required. Because interacting directly from mobile to the MongoDB server has some limits and makes an android application more redundant. The implementation of the mobile frontend was also included in this phase.

The main object of phase 3 is to develop the web version of this project, after evaluating the last mobile version's performance and adding new functionality. It is necessary to redesign the backend logic that interacts with MongoDB and, consequently, redesign the mobile app using React Native to adopt new changes.

The last phase consists of evaluating the application made before, to develop the final version of the project. In this closing phase, I have added the ELK stack to the project to complete the application as a real monitoring tool.

3.2. Breakdown of milestones and tasks:

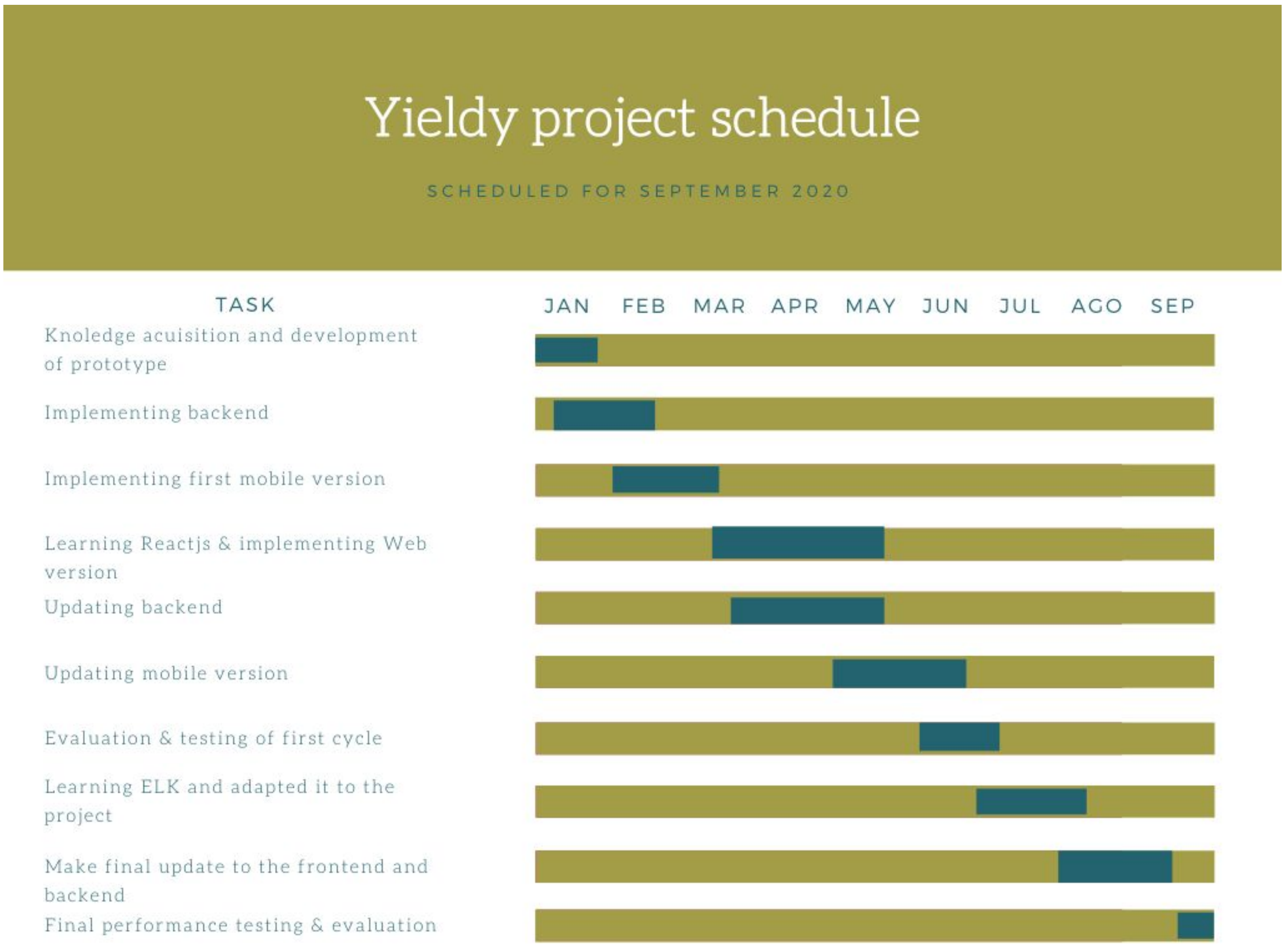


Figure 2: Project schedule diagram

3.3. Requested Analysis

At the beginning of this project, the main functionality requested was to have a mobile application to monitor different devices' status and notify the user when some abnormal status was detected.

Nevertheless, after implementing the prototype and doing research, the definitive application will have the following types of users to adopt new requirement integrated after meetings with the tutor:

- **Company owner**: the superuser of the application, registered with email, password, and company's information.
- **Administrator**: created by the company owner, assigned by the company owner to a system (the admin can be assigned to multiple systems).
- **Technicians**: created by the company owner, assigned to a system by the company owner (the technician can be assigned to multiple systems). Has permission to monitor the status of each device in the system.

All the users can publish messages in the system that he has permission to access and receive a notification from the same system while using the mobile version when an incident is produced, like creation and notification of a task, a publication of a message in the system.

3.3.1 Functionalities

After the first meeting with the project tutor, we defined the main functionalities that the web application should have. Later on, these functionalities were tuned and redefined, and some others came up as the project progressed.

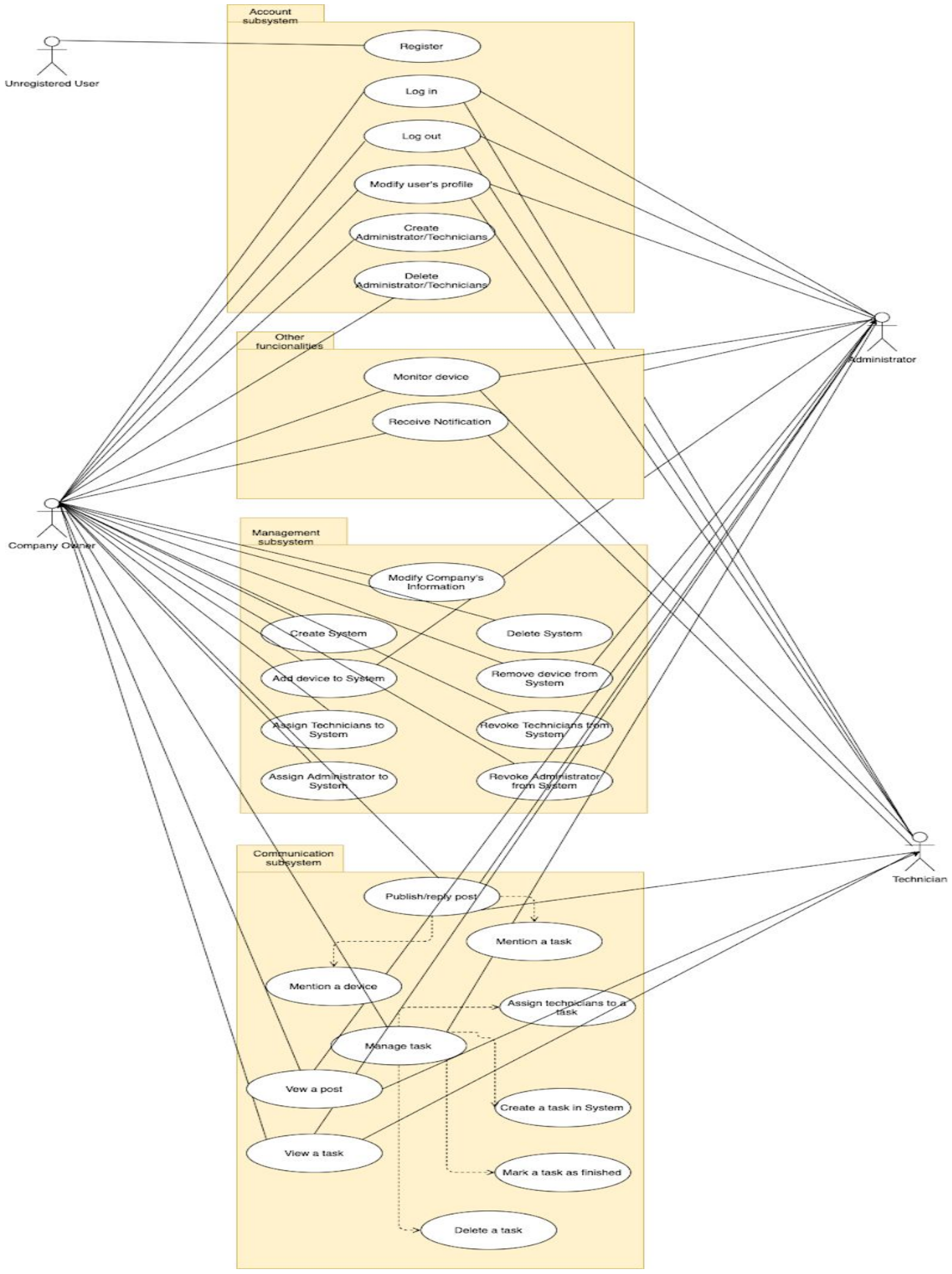


Figure 3: Project's use cases diagram

3.3.1.1. Company owner Interface

The company owner is the superuser of a company, registered with a valid email and password. The company owner can perform all the action that we listed below (note that the actions like creation of system, creation and assigning users have a limitation per company due to the server’s limitation in the development stage, which can be adjusted later in the deployment process):

- Action in account subsystem: register an account, log in, logout, creation, and deletion of a normal user (admin or technician) and modify the user’s profile.
- Action in Management subsystem: modify the company’s information, creation, and deletion of a system, add and remove a device to/from the system, assign and revoke a regular user to/from the system in the company.
- Action in Communication subsystem: view, publish and reply a message (mentioning a device or a task or without) to a system in the company, and manage all the tasks in the system by assigning and removing technicians, visualization, creation of the task, and also modify the task.
- Action in Other functionalities: monitor devices of all the systems in the company, receive a notification while using the mobile version when another user has published a message, modified a task.

3.3.1.2. Administrator Interface

An administrator is created and assigned by the company owner to the system. The administrator has the part of the permission to the application:

- Action in account subsystem: login and logout to/from the application and modify his profile.
- Action in Management subsystem: add and remove a device to/from the system which the user has permission to access (assigned by the company owner).
- Action in Communication subsystem: view, publish and reply a message (mentioning a device or a task or without) to a system that the company owner has assigned the user to, and manage all the tasks in the system (visualization, creation, deletion and modification of it).
- Action in Other functionalities: monitor devices of the system that the admin has permission to access, receive a notification while using the mobile version when another user has published a message or has modified a task within the system.

3.3.1.3. Technicians Interface

A technician is created by the company owner, also assigned by the company owner. The technician has the part of the permission to the application:

- Action in account subsystem: login and logout to/from the application and modify his profile.
- Action in Communication subsystem: view, publish, or reply a message (mentioning a device or a task or without) to a system that the company owner has assigned the technician and view the tasks within the system.
- Action in Other functionalities: monitor devices of the system that he has permission to access, receive a notification while using the mobile version when another user has published a message or has modified a task within the system.

3.3.2 Non-functional requests

3.3.2.1. Platform compatibility

For a cross-platform application, the most important non-functional requirement is to create a software application to work with more than one hardware platform or operating system.

3.3.2.2. Scalability

The application must have many possibilities to be extended in the future, and the application must be scalable, both of the server-side application and frontend. Adding more software or script that helps to detect the abnormal status of devices is also required in that case.

4. Technology Stack

The application has been developed under a combination of MERN stack and ELK stack. The MERN stack is used for the development of frontend and backend, and it is formed by the stack of MongoDB, Express JS, React Native/React.js, and Node.js. The ELK stack is used for being able to monitor the device's status and process.

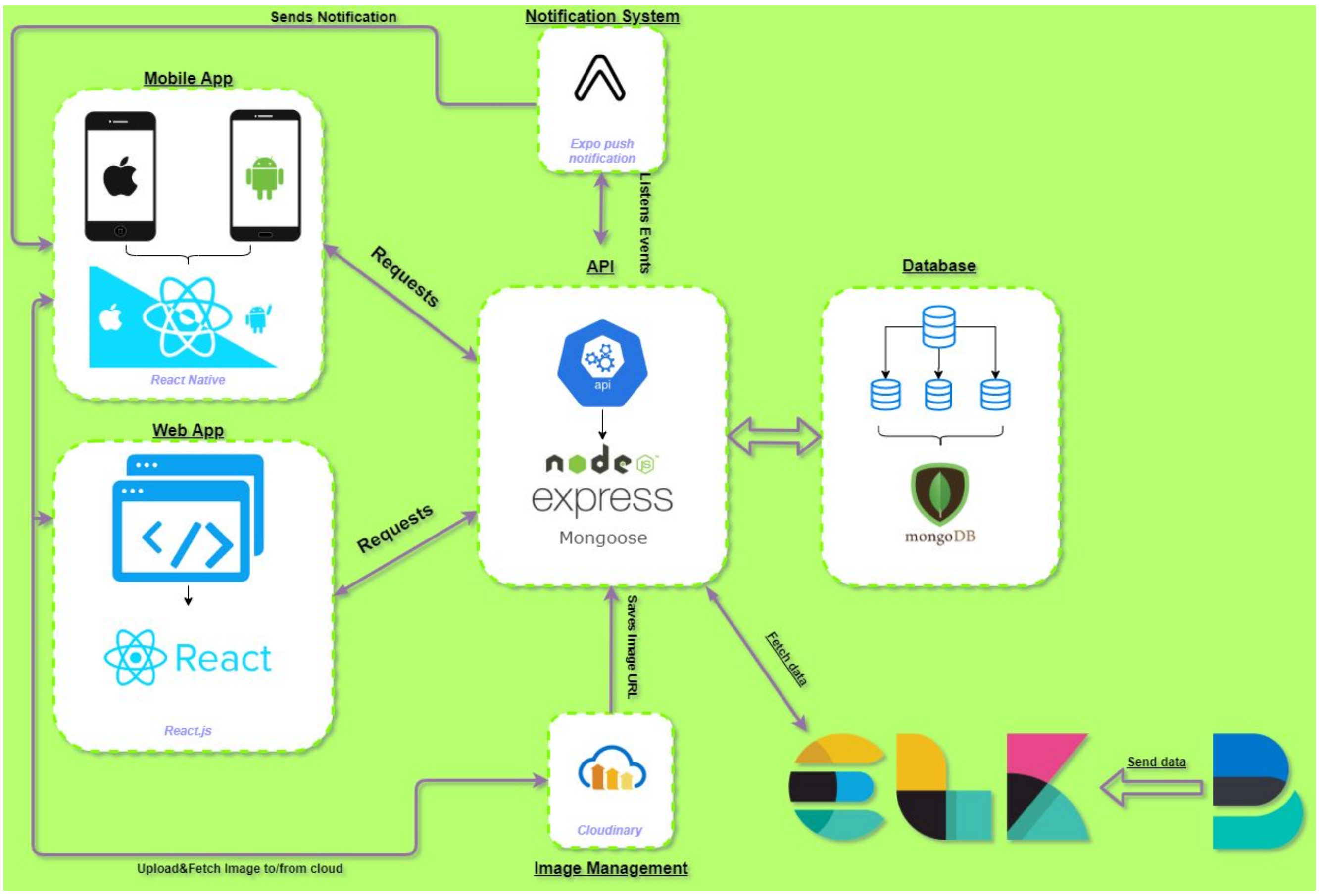


Figure 4: Application technology stack diagram

4.1. MongoDB

Figure 5: MongoDB Atlas

MongoDB is an open-source database management system (DBMS) that uses a document-oriented database model that supports various forms of data. It is one of the numerous nonrelational database technologies which arose in the mid-2000s under the NoSQL banner for use in big data applications and other processing jobs involving data that does not fit well in a rigid relational model. Instead of using tables and rows as in relational databases, MongoDB architecture is made up of collections and documents.

MongoDB does not require predefined schemas, and it stores any data. That gives us the flexibility to create any number of fields in a document, making it easier to scale compared to relational databases. It fits nicely in this project because the database was planned to be fed with different sources (Nessus, Metasploit, or indexing Elasticsearch), so I used the MongoAtlas (MongoDB hosted cloud service) as the database for this project.

4.2. Express.JS

Figure 6: Express.JS and node.JS

Express.js is a Node js web application server framework, which is specifically designed for building web applications. In order to create a REST API to help organize my web and mobile applications into an MVC architecture on the server-side. I have to use a database like MongoDB with Mongoose (a node.js library that allows me to connect backends to MongoDB) to develop a backend for my application. Express.js helps me manage everything, from routes to handling requests and views.

4.3. ReactJs

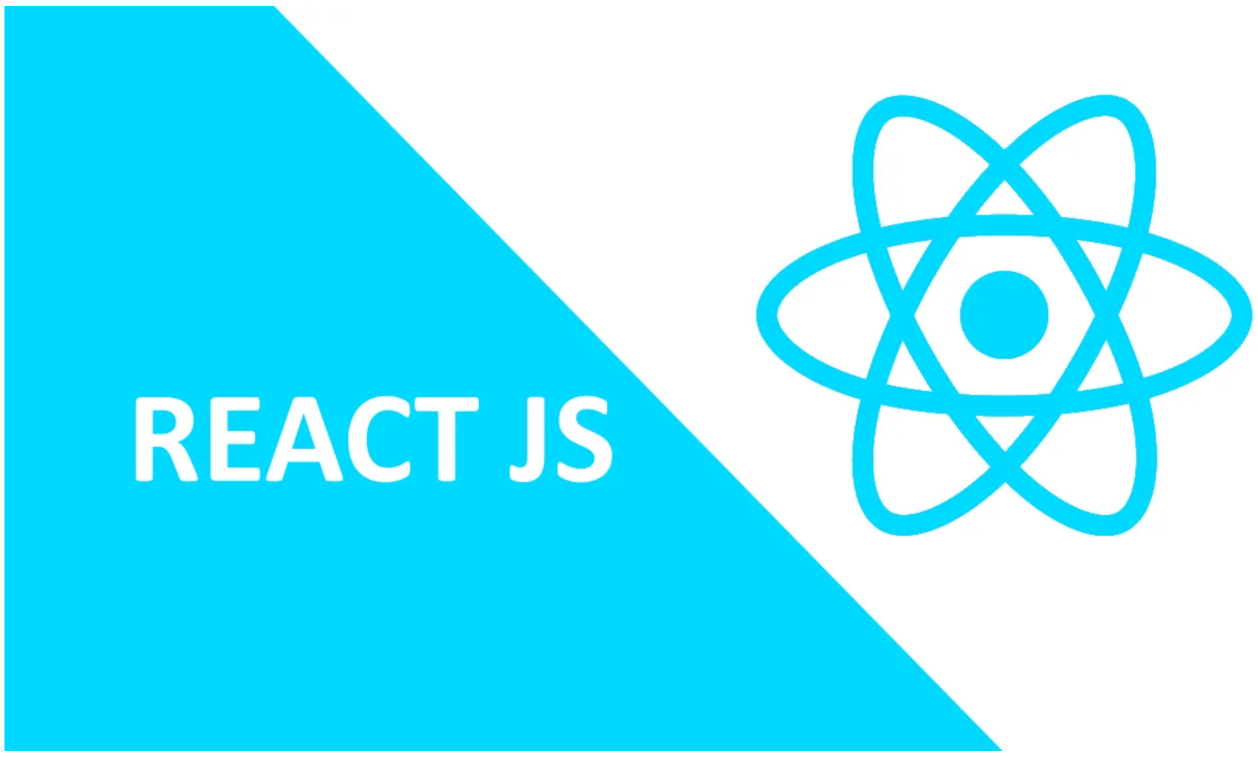


Figure 7: ReactJS

React provides a series of clear advantages over the traditional way of making a website; its development facilities, together with performance, flexibility, and code organization, make it one of the best options.

One of the main reasons for this to be possible is the use of the virtual DOM. React can generate the DOM dynamically, makes the changes in a copy in memory, and then compares it with the current version of the DOM. In this way, avoid rendering the whole page every time there are changes; this change is applied to the component that has been updated, fast, and straightforward. That leads to better user experience, as well as impressive performance and fluidity.

4.4. React Native

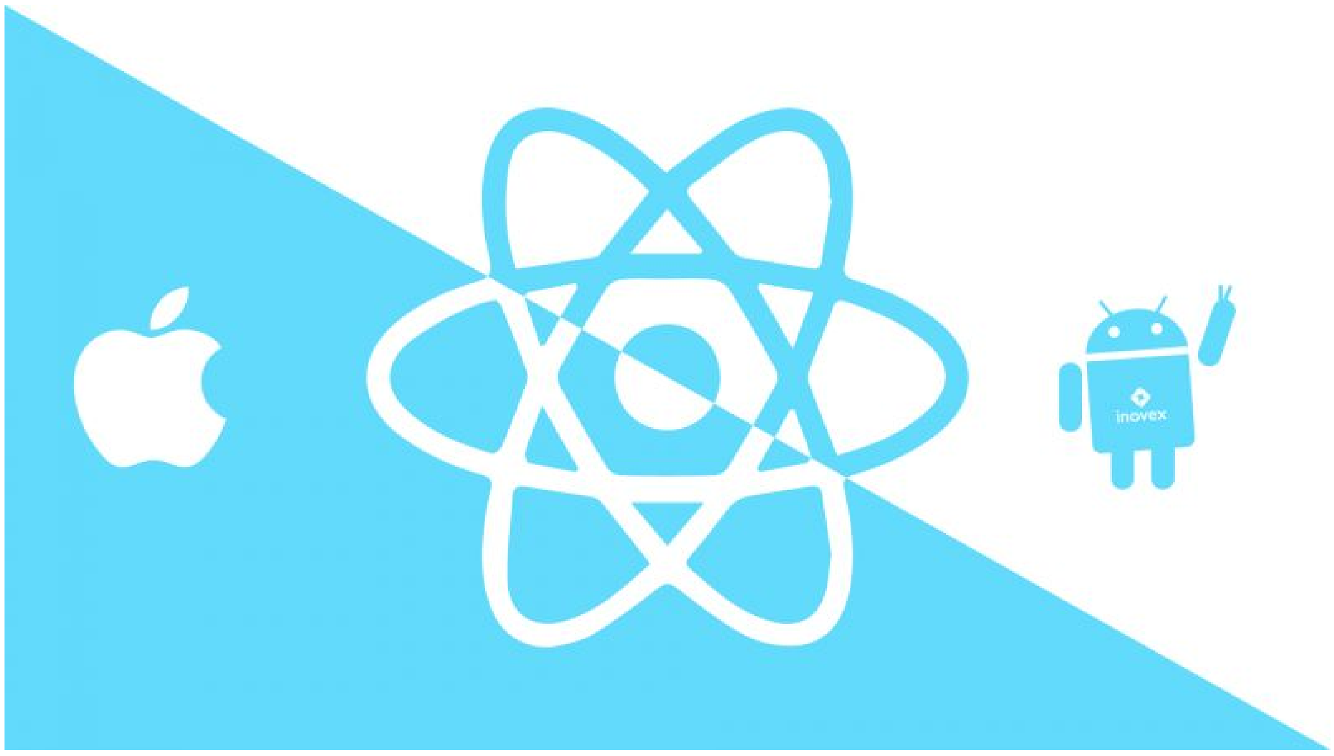


Figure 8: React Native

React Native is a cross-platform (iOS and Android) native application programming framework that is based on JavaScript and ReactJS.

One of the advantages of this framework is the performance because it gets connected to the native components for both of the operating systems and generates code to the native APIs. Another advantage is that I can reuse part of the code from this version and adapt to the implementation of the web version.

4.4.1 Expo

Figure 9: Expo

This project uses Expo to develop the React Native application. Expo is a framework for rapidly developing React Native applications. Expo provides a layer on top of the React Native APIs to make it easier to use and manage. It also provides tools that make it easy to start and test React Native applications. Lastly, it provides services that are usually only available when you install a third-party React Native component and UI components. All are made available through the Expo SDK. It also provides many pre-built libraries such as UI components and push-notification systems, so it is straightforward to use.

4.5. Node.JS

Node.js is an open-source, cross-platform, JavaScript runtime environment that executes JavaScript code outside of a browser. Node.js lets developers use JavaScript to write command-line tools and for server-side, scripting—running scripts server-side to produce dynamic web page content before the page is sent to the user's web browser. Consequently, Node.js represents a "JavaScript everywhere" paradigm, unifying web application development around a single programming language, rather than different languages for server- and client-side scripts.

4.6. ELK stack

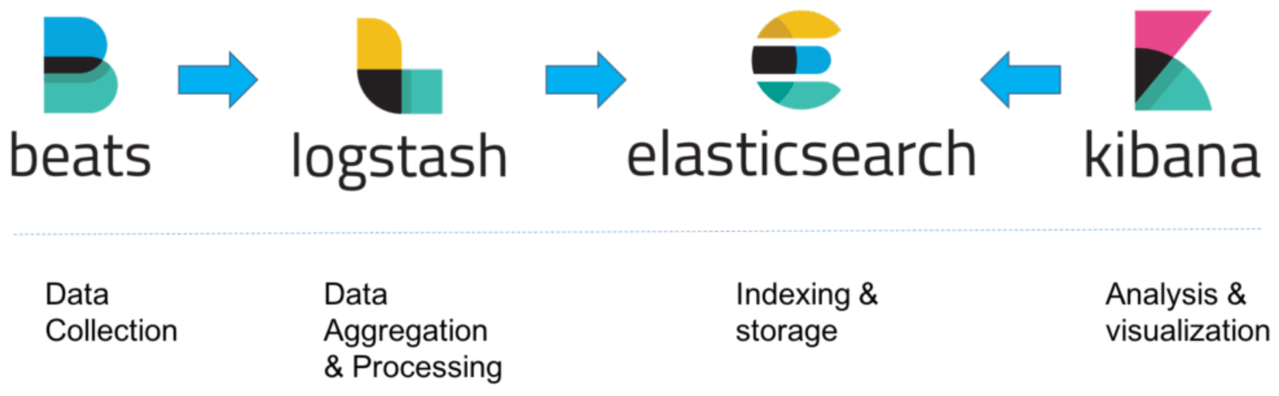


Figure 10: The ELK stack

ELK Stack (or Elastic Stack) is a set of open-source tools developed by Elasticsearch that allows you to collect data from any source and in any format to search, analyze, and visualize data in real-time.

The following component forms it:

- Elasticsearch: a distributed search engine built on Apache Lucene. It allows indexing and retrieving JSON documents in different formats. It is based on Java and provides a RESTFUL access interface.
- Logstash: it is an engine that allows collecting data from different sources, normalizing and distributing it. It was initially optimized for log files, although it can read data from many types of sources.
- Kibana: the tool that allows the visualization and exploration of large amounts of data in real-time. Through the graphical representation, it allows consulting complex data sets quickly.
- Beats: these are data loaders that are installed on servers and function as agents that send operational information such as system logs and processes running on the system to Elasticsearch or Logstash.

4.7. Cloudinary

Cloudinary is an end-to-end image- and video-management solution for websites and mobile apps, covering everything from image and video uploads, storage, manipulations, optimizations to delivery. It was used for saving the user’s profile image and providing the link to the frontend.

5. Database Design

5.1. MongoDB

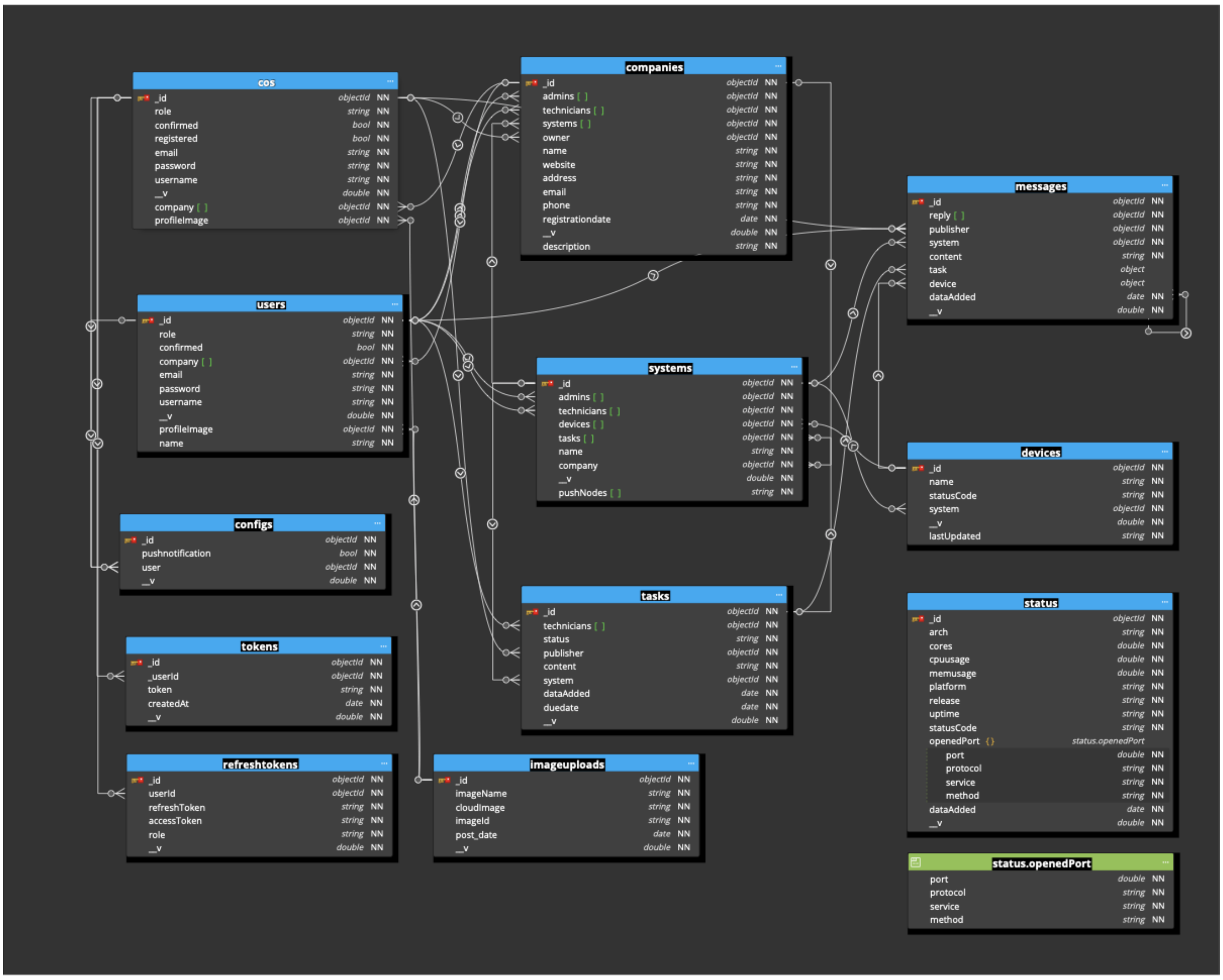


Figure 11: Database Model diagram

Instead of storing data in the form of a traditional two-dimensional row-column structure like most SQL-like databases, MongoDB saves data in the form of documents and collections. In this project, the database was initially designed with 13 collections but lately has turned into 11 collections.

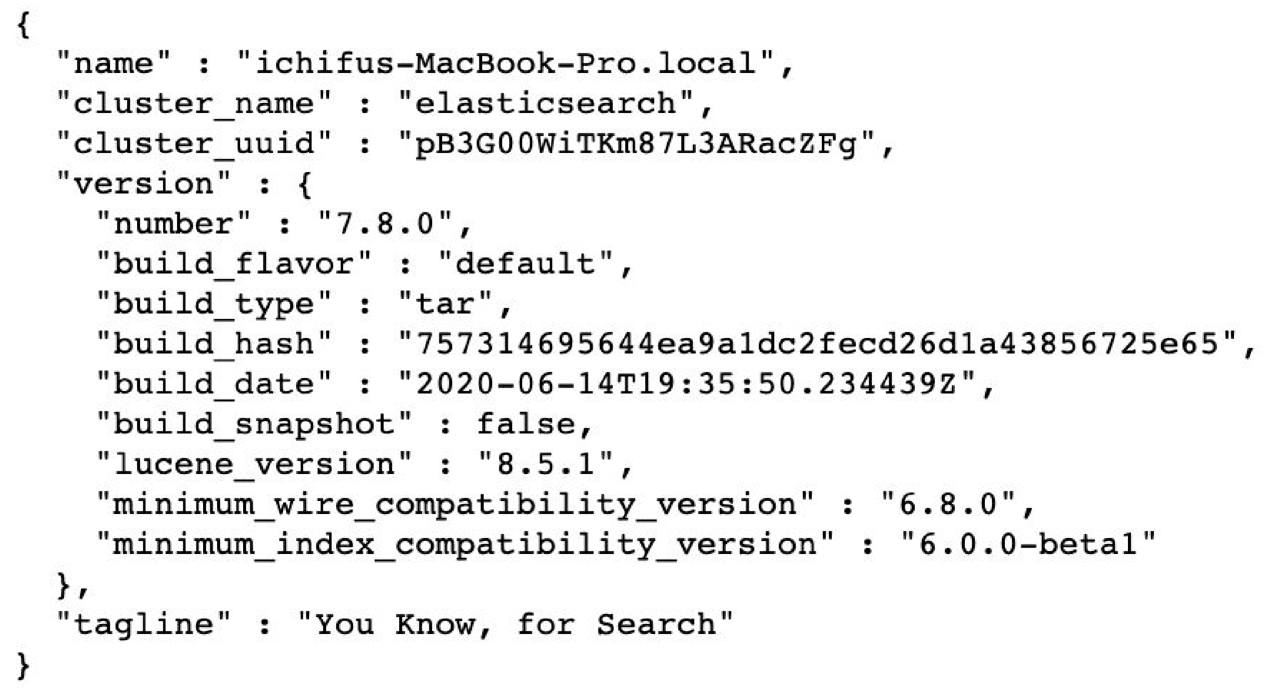
1. **COS:** designed for saving the company owner's information. It differs from the collection **USERS** because it only saves the superuser of this system.
2. **USERS:** designed for saving normal users like administrators or technicians. It has a role field like the COS collection to distinguish the different types of users.
3. **CONFIGS:** stores the superuser's (company owner) and the normal user's (administrator and technician) configuration. It was designed to save the setting variable like enable or disable the push notification.
4. **TOKENS:** stores the confirmation email's activate token. It does have a lifetime of about 7,2 minutes, so after this lifetime, the activation token will be deleted from the database.
5. **REFRESHTOKENS:** stores the refresh tokens of current active users. Allow users to fetch a new access token with their refresh tokens.
6. **COMPANIES:** stores all the companies' information and their relation with the company members (company owner and other normal users) by saving multiple lists of its objectID.
7. **SYSTEMS:** stores all the systems and their relation with the system members (assignees).
8. **TASKS:** stores all the tasks' information.
9. **MESSAGES:** stores all the messages' information.
10. **DEVICES:** stores all the devices' information.
11. **IMAGEUPLOADS:** stores the image link hosted by the Cloudinary that was initially uploaded by the user.

STATUS AND **STATUS.OPENEDPORT** : both are deprecated during the implementation and redesign of this project. Instead of using MongoDB to save this information, the project uses Elasticsearch to receive, digest then establish the connection to the backend in order to send the device’s status.

5.2. Elasticsearch

During the first cycle of the research and implementation of this project, the application uses a personalized node.js script to upload the device's basic status (such as CPU usage, memory usage, uptime…) to MongoDB. After evaluating the application at the end of the first cycle, that option did not run as expected due to node.js script's performance limitation.

The solution to this problem is that instead of using a personalized script, use the ELK stack to receive, normalize, and store the information to Elasticsearch. So the second cycle of development's main task is to configure the ELK stack and adapt it to the project. This project uses the 7.8.0 version of the ELK stack.



```
{
  "name" : "ichifus-MacBook-Pro.local",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "pB3G00WiTKm87L3ARacZFg",
  "version" : {
    "number" : "7.8.0",
    "build_flavor" : "default",
    "build_type" : "tar",
    "build_hash" : "757314695644ea9a1dc2fecd26d1a43856725e65",
    "build_date" : "2020-06-14T19:35:50.234439Z",
    "build_snapshot" : false,
    "lucene_version" : "8.5.1",
    "minimum_wire_compatibility_version" : "6.8.0",
    "minimum_index_compatibility_version" : "6.0.0-beta1"
  },
  "tagline" : "You Know, for Search"
}
```

Figure 12: Elasticsearch

5.2.1. Logstash and Beats

The project uses a combination of two types of beats: Metricbeat and Auditbeat. With Metricbeat and Auditbeat, users can send the device's system status, such as memory usage, to the Logstash. In the case of Auditbeat, users can also send system process logs to the Logstash.

Logstash provides the possibility to receive the data from Beats before sending it to Elasticsearch. It can normalize the data by defining the rules in the configuration file before sending them to Elasticsearch. After normalizing the data following each configuration file's rule, the logs will be sent to Elasticsearch. Then the application can fetch the data from the server-side (backend) by querying the Elasticsearch server.

6. Implementation

This web version and the backend were deployed to the internet after finishing the first cycle of development. However, after introducing the ELK stack, the deployed version was deprecated because it is difficult to find a free server to host the Elasticsearch service. The final version of this project is not deployed, but it is essential to mention it here before we dive into the project's implementation.

6.1. User system

This application offers access to different types (roles) of users: **company owner**, **administrator**, and **technician**. Using react and react-native to develop the front-end client, the application shares the workflow in these two different frameworks.

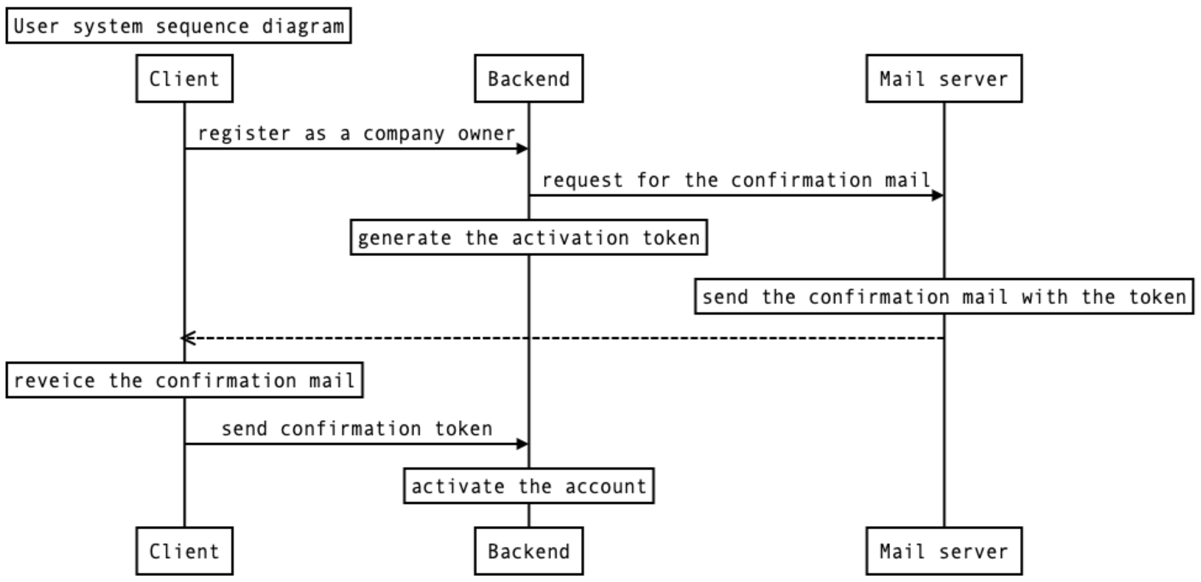

User system sequence diagram
Client
Backend
Mail server
register as a company owner
request for the confirmation mail
generate the activation token
send the confirmation mail with the token
reveice the confirmation mail
send confirmation token
activate the account
Client
Backend
Mail server

Figure 13: Company owner registration sequence diagram

To register as a company owner, the user has to provide an email that will use this email address to activate the account. It was designed as a superuser that can create a company with this account and create new normal users such as administrators and technicians. The mail server is developed using the library **Nodemailer** of node.js. The backend handles it, so when the backend receives the registration request, the **Nodemailer** will send the confirmation mail with the activation token after the user receives that mail and confirms it. The user will be registered to the database and will be able to log to the system.

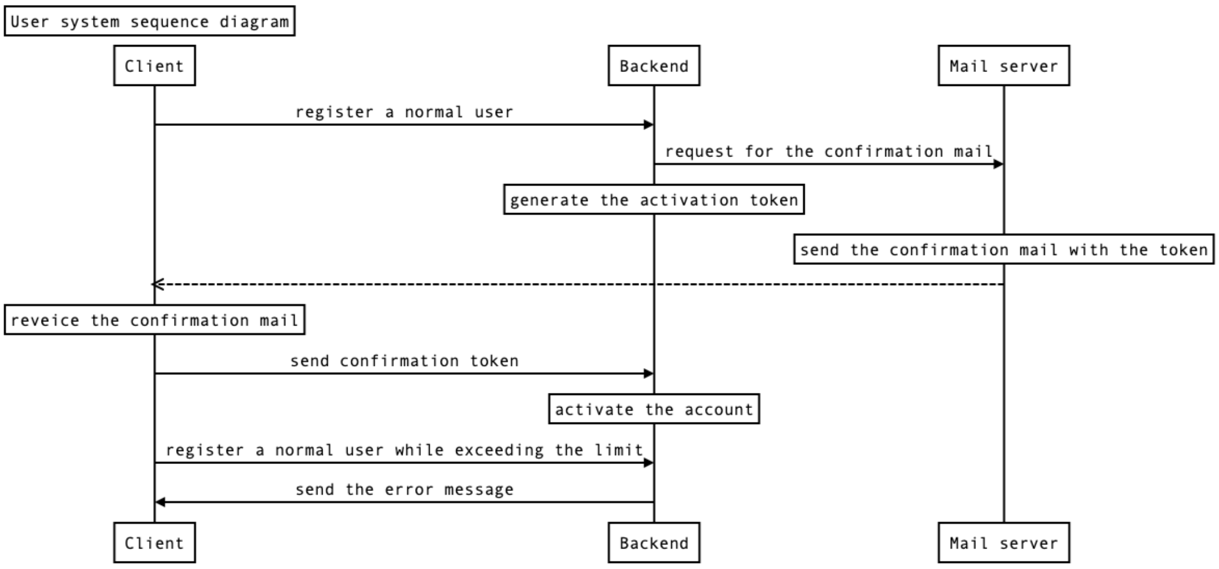

User system sequence diagram
Client
Backend
Mail server
register a normal user
request for the confirmation mail
generate the activation token
send the confirmation mail with the token
reveice the confirmation mail
send confirmation token
activate the account
register a normal user while exceeding the limit
send the error message
Client
Backend
Mail server

Figure 14: Normal user registration sequence diagram

The registration of a normal user shares the same workflow with the registration of a company owner. The only difference is that the normal user is created by the company owner instead of registered by himself. To create a normal user, the company owner has to log into the system. It does have a limitation on the number of normal users for each company. So when exceeding the limit, the backend will not create the account. The normal user's ID and password that he will use for login are generated automatically by the server and will not be able to change in the future. The normal user will receive the confirmation mail with the user's Id and password. After confirming the email, the user will be able to log in to the system.

	Company	User	Device	System	Task	Message
Create	Company owner	Company Owner	Company Owner, Administrator	Company Owner	Company Owner, Administrator	Company Owner, Administrator, Technician
Read	Company Owner, Administrator, Technician	Company Owner, Administrator, Technician	Company Owner, Administrator, Technician	Company Owner, Administrator, Technician	Company Owner, Administrator, Technician	Company Owner, Administrator, Technician
Update	Company Owner	Company Owner, Administrator, Technician	Company Owner, Administrator, Technician	Company Owner	Company Owner, Administrator	-
Delete	-	Company Owner	Company Owner, Administrator	Company Owner,	-	-

Figure 15: Project CRUD table

Once logged in, users can access the different contents of the platform according to the user's role. Being a company owner, the user will have total access to the application: from its fields to create, list, and delete users and systems, add and remove the device to/from the system and also has full control of all incidents (message and task) generated by the different registered users or devices in different systems of the company.

Administrators and technicians are considered normal users**,** and the company owner must previously register it and assign them to the system. The administrator also has full control of all incidents, but only for the system in which the administrator is assigned.

For being a technician, users will be able to view all devices' status, post messages, and view the task created by the system administrator or company owner in the system that the technician has been assigned.

After the registration succeeds and the user is logged in to the system, the user will be able to change his profile, such as username and profile image (the profile image can only be changed in the web version due to the limitation of the Expo framework).

6.2. Authentication system

The authentication system works pretty similarly to the registration system. Users will send different requests to log into the system depending on the user account's role (company owner or normal user).

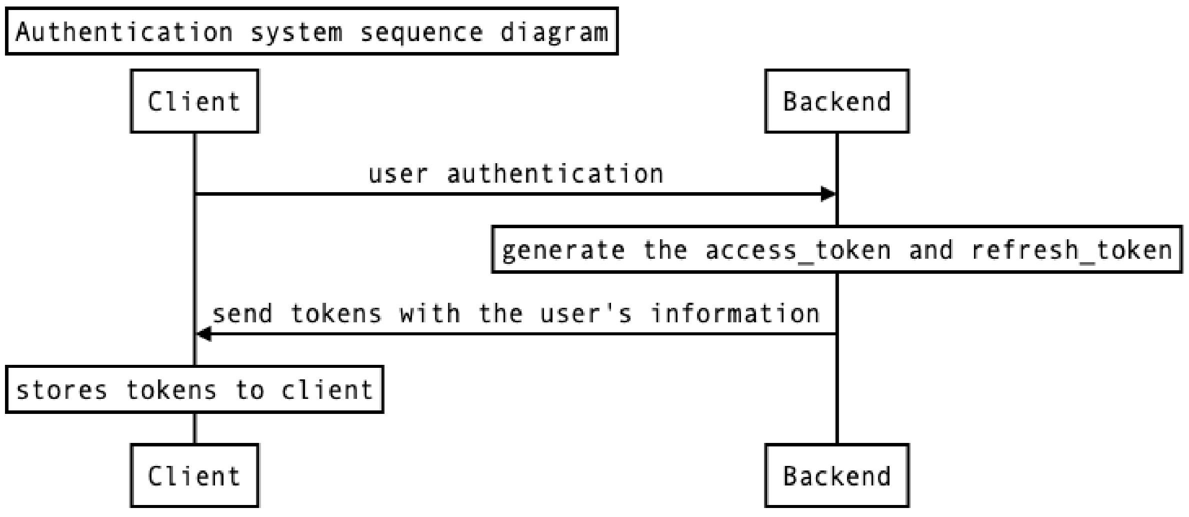

Authentication system sequence diagram
Client
Backend
user authentication
generate the access_token and refresh_token
send tokens with the user's information
stores tokens to client
Client
Backend

Figure 16: Authentication sequence diagram

Depending on the platform and the account's role, the user selects the login form (login as a company owner or normal user) that suits the user's role. After sending the valid user's credential information to the backend, the backend will send back the access token and the refresh token. The backend will also store the tokens with the user's information. Every user can have multiple tokens (depending on the platform that the user logged in) stored in the backend. The access token has 10 minutes of lifetime; after that duration, the different clients will use the refresh token and the user's information to request the backend to get back a new access token. The front end uses the access token to make all other requests to the backend to fetch the information that requires the user to be connected.

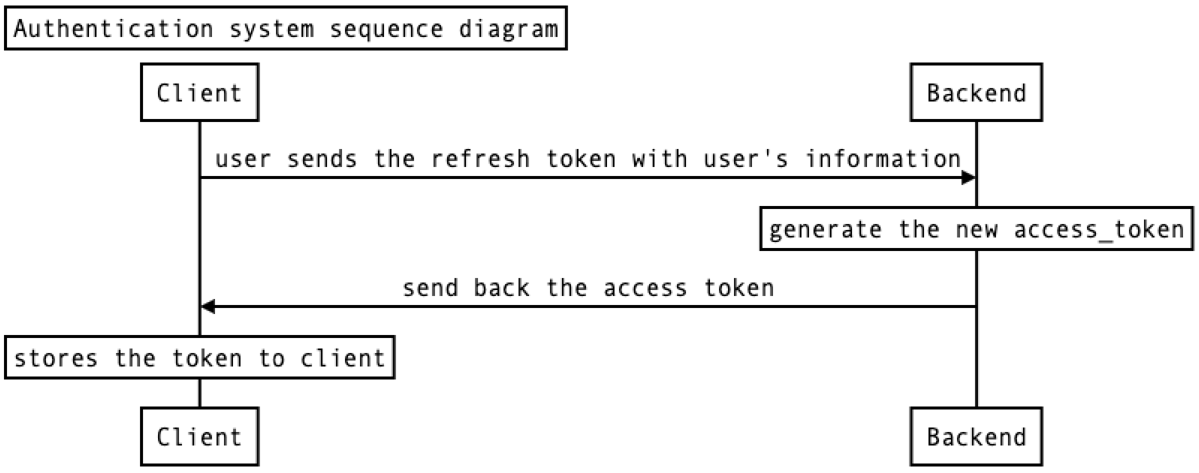

Authentication system sequence diagram
Client
Backend
user sends the refresh token with user's information
generate the new access_token
send back the access token
stores the token to client
Client
Backend

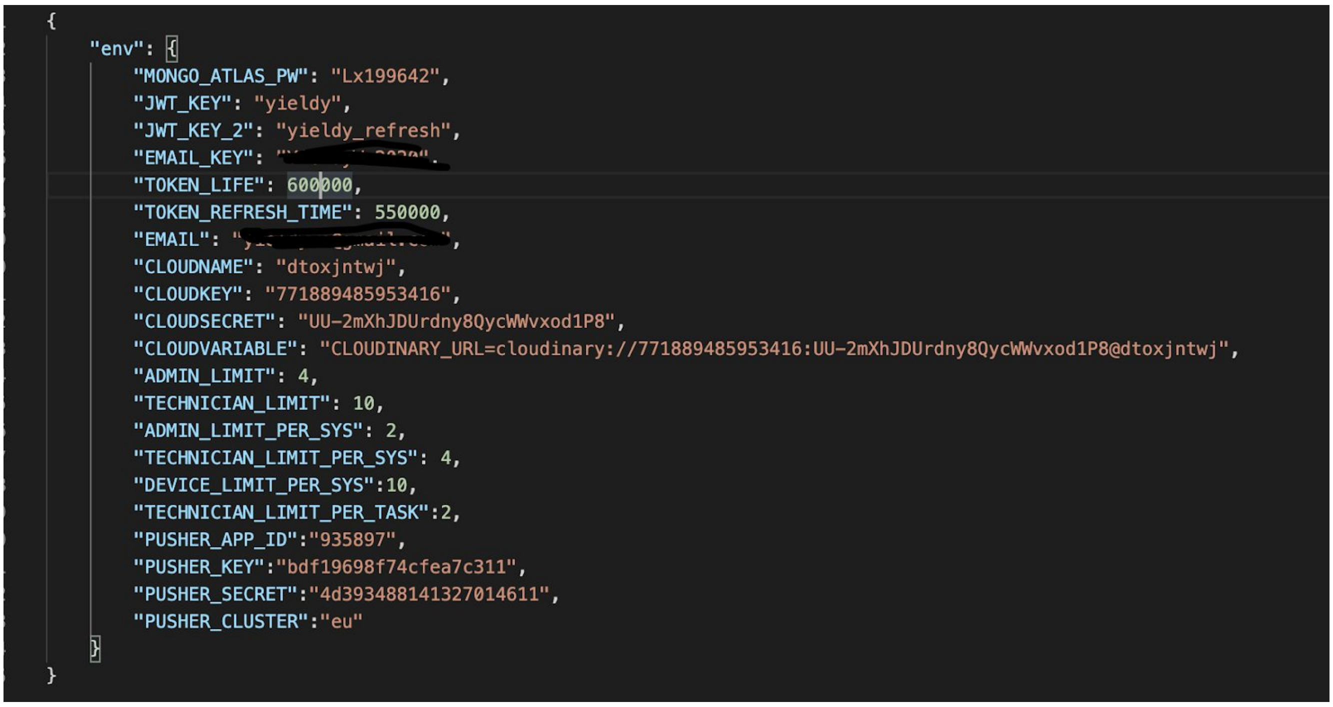
Figure 17: Refresh access token sequence diagram

After the user logs out from the client, the client will also send a request to the backend to delete stored tokens.

6.3. Work group system

The application is designed to monitor the company's devices for medium-small sized enterprises and allow them to communicate within them. With this requirement, the application has designed two levels of the workgroup for the user:

Company: the highest level of the workgroup.The company owner is obliged to create a company after the first login to the application. The company owner has to provide the company's information to create the company. After that, the company owner will be able to create normal users , systems and assign users to the system. At the development stage, there is a limitation on the number of users and systems created in each company; it was designed like this for the easy extension in the future, such as making different paid plans for the company owner to change the limitation. For now, users can create four administrators, ten technicians, and two systems in each company. It is controlled by modifying the backend's environmental variable.



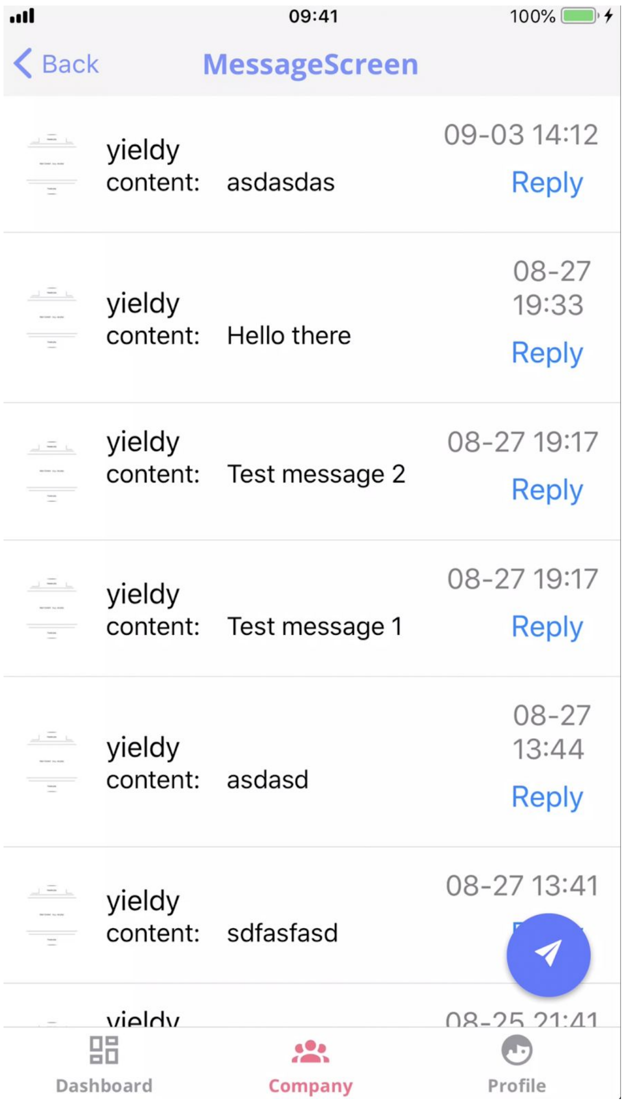
```
{
    "env": {
        "MONGO_ATLAS_PW": "Lx199642",
        "JWT_KEY": "yieldy",
        "JWT_KEY_2": "yieldy_refresh",
        "EMAIL_KEY": [redacted],
        "TOKEN_LIFE": 600000,
        "TOKEN_REFRESH_TIME": 550000,
        "EMAIL": [redacted],
        "CLOUDNAME": "dtoxjntwj",
        "CLOUDKEY": "771889485953416",
        "CLOUDSECRET": "UU-2mXhJDUrdny8QycWWvxod1P8",
        "CLOUDVARIABLE": "CLOUDINARY_URL=cloudinary://771889485953416:UU-2mXhJDUrdny8QycWWvxod1P8@dtoxjntwj",
        "ADMIN_LIMIT": 4,
        "TECHNICIAN_LIMIT": 10,
        "ADMIN_LIMIT_PER_SYS": 2,
        "TECHNICIAN_LIMIT_PER_SYS": 4,
        "DEVICE_LIMIT_PER_SYS":10,
        "TECHNICIAN_LIMIT_PER_TASK":2,
        "PUSHER_APP_ID":"935897",
        "PUSHER_KEY":"bdf19698f74cfea7c311",
        "PUSHER_SECRET":"4d393488141327014611",
        "PUSHER_CLUSTER":"eu"
    }
}
```

Figure 18: Backend environmental configuration

System: the second level of the application workgroup. A company can create at most two systems according to the current environmental configuration. Each system can have two administrators and four technicians assigned, and a limitation of ten devices per system to be created. In this level, each administrator and technician assigned to this system will be able to publish posts, view tasks, and monitor a device. The administrator assigned to the system also has the permission to create devices and manage tasks (the creation, assign a technician, and change its status) within the system. There is an exception that the company owner has permission to access all the systems' content in the company.

6.4. Communication system

The application provides two functionalities of communication in the system level: the message board (or the post-board) and the task board.


09:41 100%

Back MessageScreen

yieldy
content: asdasdas
09-03 14:12
Reply

yieldy
content: Hello there
08-27 19:33
Reply

yieldy
content: Test message 2
08-27 19:17
Reply

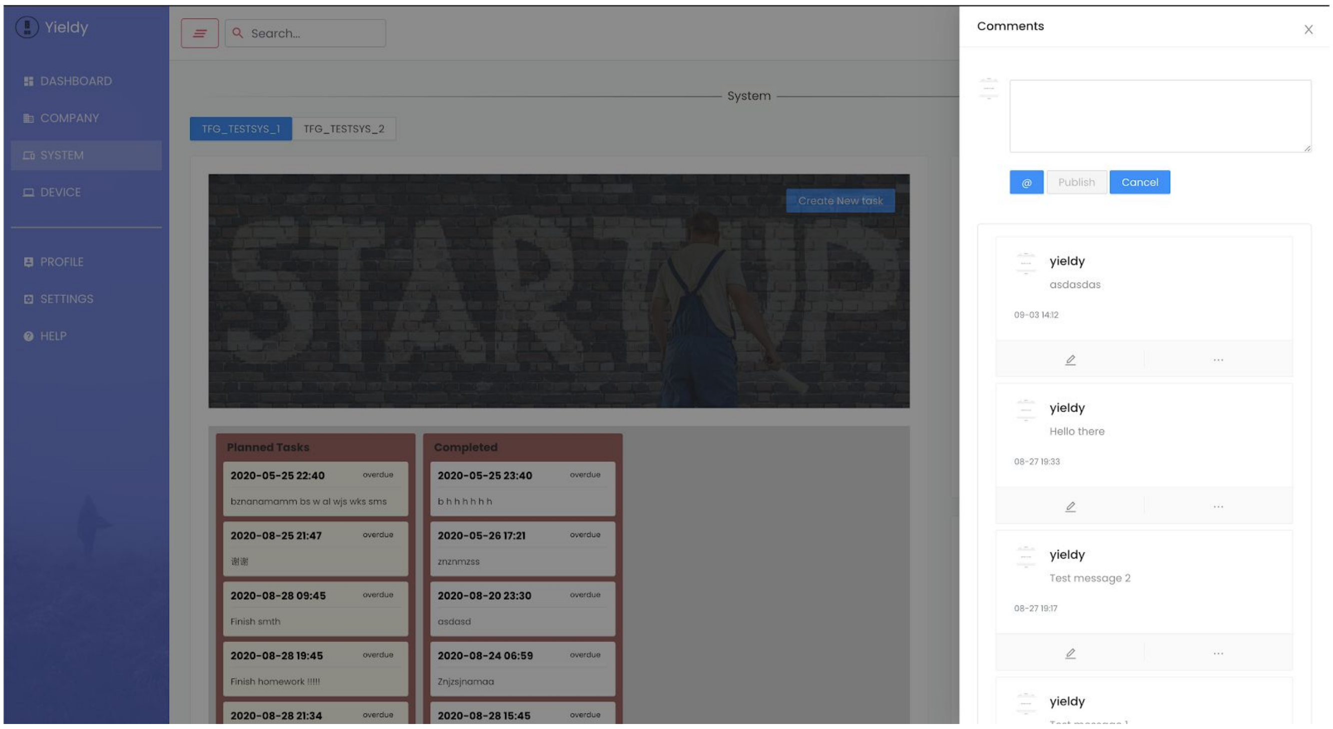
yieldy
content: Test message 1
08-27 19:17
Reply

yieldy
content: asdasd
08-27 13:44
Reply

yieldy
content: sdfasfasd
08-27 13:41

yieldy
08-25 21:41

Figure 19: Message board on the mobile version



Planned Tasks

2020-05-25 22:40 overdue
bznanamamm bs w al wjs wks sms

2020-08-25 21:47 overdue
谢谢

2020-08-28 09:45 overdue
Finish smth

2020-08-28 19:45 overdue
Finish homework !!!!!

2020-08-28 21:34 overdue

Completed

2020-05-25 23:40 overdue
b h h h h h h

2020-05-26 17:21 overdue
znznmzss

2020-08-20 23:30 overdue
asdasd

2020-08-24 06:59 overdue
Znjzsjnamaa

2020-08-28 15:45 overdue

yieldy
asdasdas
09-03 14:12

yieldy
Hello there
08-27 19:33

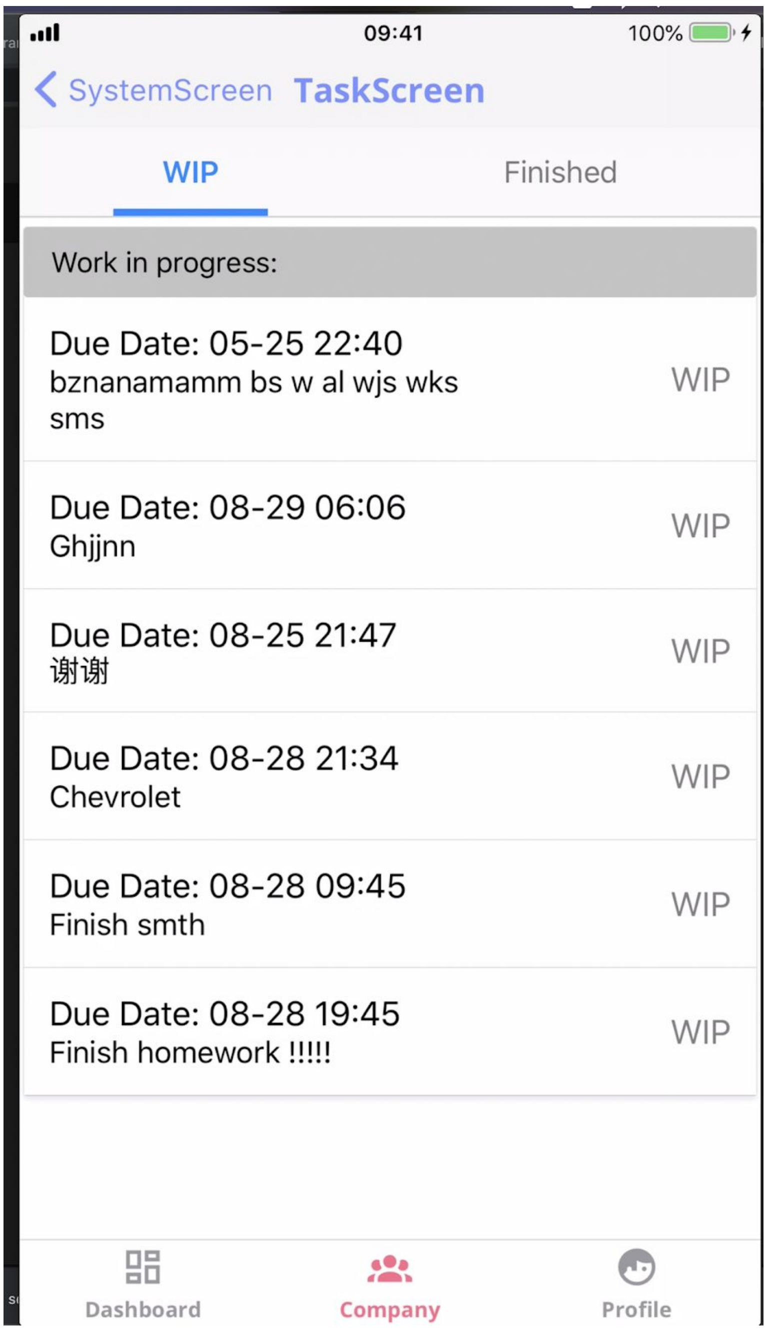
yieldy
Test message 2
08-27 19:17

yieldy

Figure 20: Message board on the web version

Besides the basic action: view, publish, and reply to a post, the application also has implemented the functionality of mentioning a device or a task. That allows the user to communicate more efficiently and in detail. The mention function is made by adding a relation between TASKS collection, MESSAGES collection, and the Devices collection in the database.

When replying to a message that already mentioned a device or a task, the user will not be able to mention another device and task and the system forces the user to mention the same device and task.



Work in progress:

Due Date: 05-25 22:40
bznanamamm bs w al wjs wks sms
WIP

Due Date: 08-29 06:06
Ghjjnn
WIP

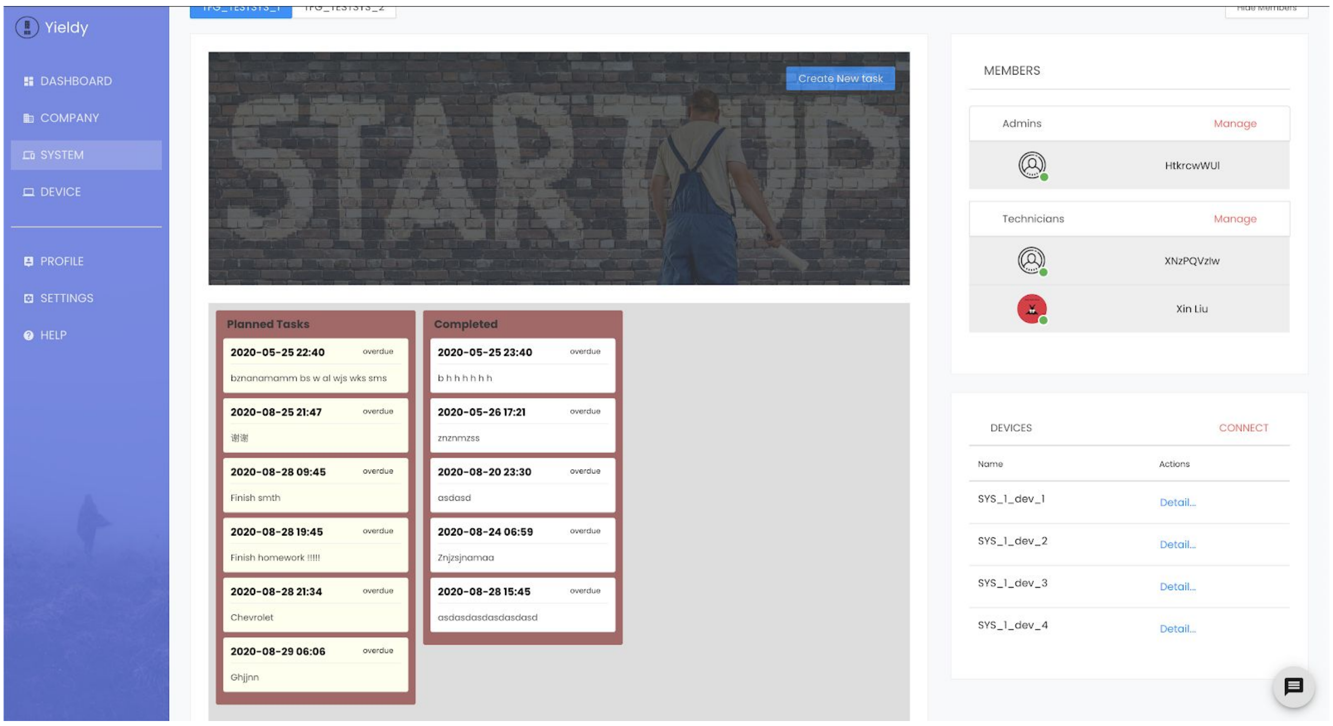
Due Date: 08-25 21:47
谢谢
WIP

Due Date: 08-28 21:34
Chevrolet
WIP

Due Date: 08-28 09:45
Finish smth
WIP

Due Date: 08-28 19:45
Finish homework !!!!!
WIP

Figure 21: Task board on the mobile version



Planned Tasks

2020-05-25 22:40	overdue
bznanamamm bs w al wjs wks sms	
2020-08-25 21:47	overdue
谢谢	
2020-08-28 09:45	overdue
Finish smth	
2020-08-28 19:45	overdue
Finish homework !!!!!	
2020-08-28 21:34	overdue
Chevrolet	
2020-08-29 06:06	overdue
Ghjjnn	

Completed

2020-05-25 23:40	overdue
b h h h h h h	
2020-05-26 17:21	overdue
znznmzss	
2020-08-20 23:30	overdue
asdasd	
2020-08-24 06:59	overdue
Znjzsjnamaa	
2020-08-28 15:45	overdue
asdasdasdasdasdasd	

Admins: HtkrcwWUl

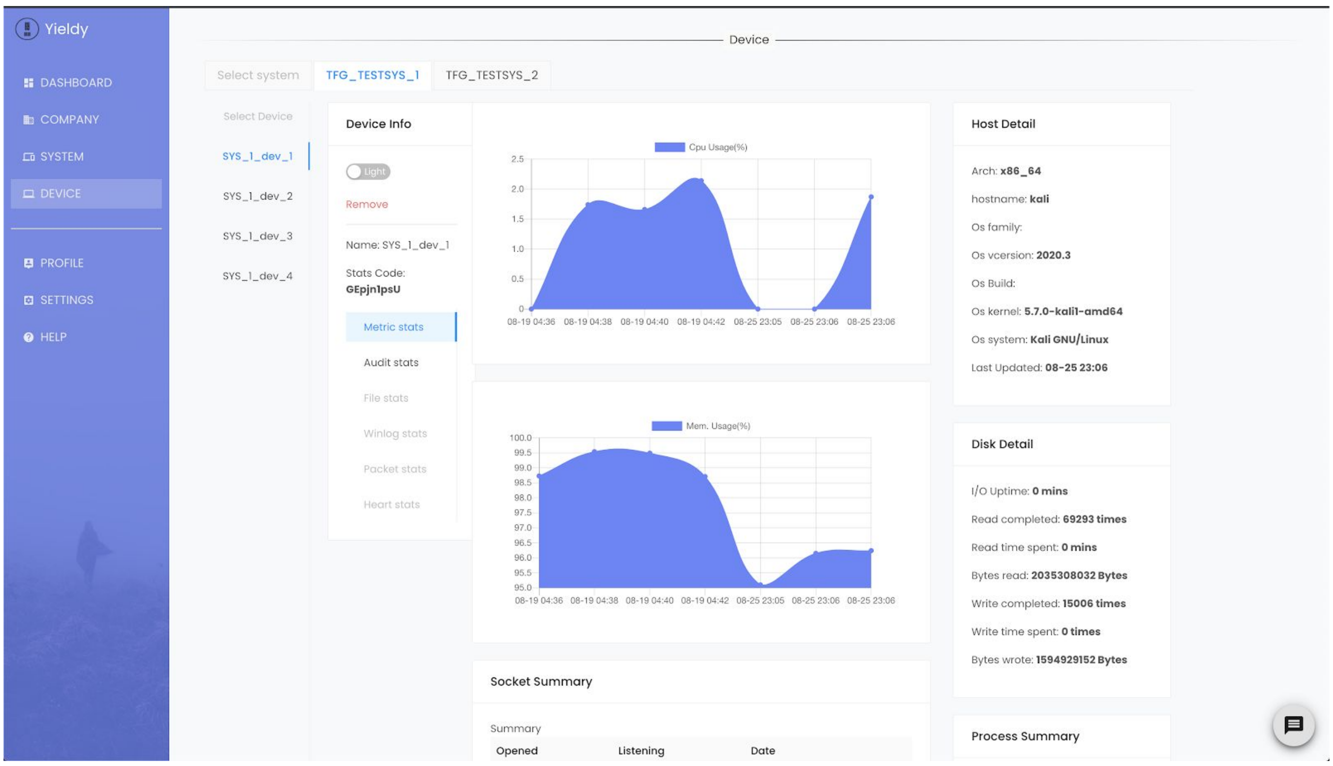
Technicians: XNzPQVzlw, Xin Liu

Name	Actions
SYS_1_dev_1	Detail...
SYS_1_dev_2	Detail...
SYS_1_dev_3	Detail...
SYS_1_dev_4	Detail...

Figure 22: Task board on the web version

Administrators (that have permission to the system) and the company owner can create the task; the task is created with the content of the task and the due date. After the task is created, the administrator and the company owner will be able to modify its status, such as the assignment of technicians to the task (the limit of technicians per task is 2) and change its status (Finished or Work in progress). The technician can only view the task of the system that he has permission to access. The web application also provides a drag and drop functionality that allows the user to modify the task status more efficiently. It is not available for the mobile version due to the limitation of the Expo framework.

6.5. Device system



Device Info

Name: SYS_1_dev_1

Stats Code: **GEpjn1psU**

Host Detail

Arch: **x86_64**

hostname: **kali**

Os family:

Os vcersion: **2020.3**

Os Build:

Os kernel: **5.7.0-kali1-amd64**

Os system: **Kali GNU/Linux**

Last Updated: **08-25 23:06**

Disk Detail

I/O Uptime: **0 mins**

Read completed: **69293 times**

Read time spent: **0 mins**

Bytes read: **2035308032 Bytes**

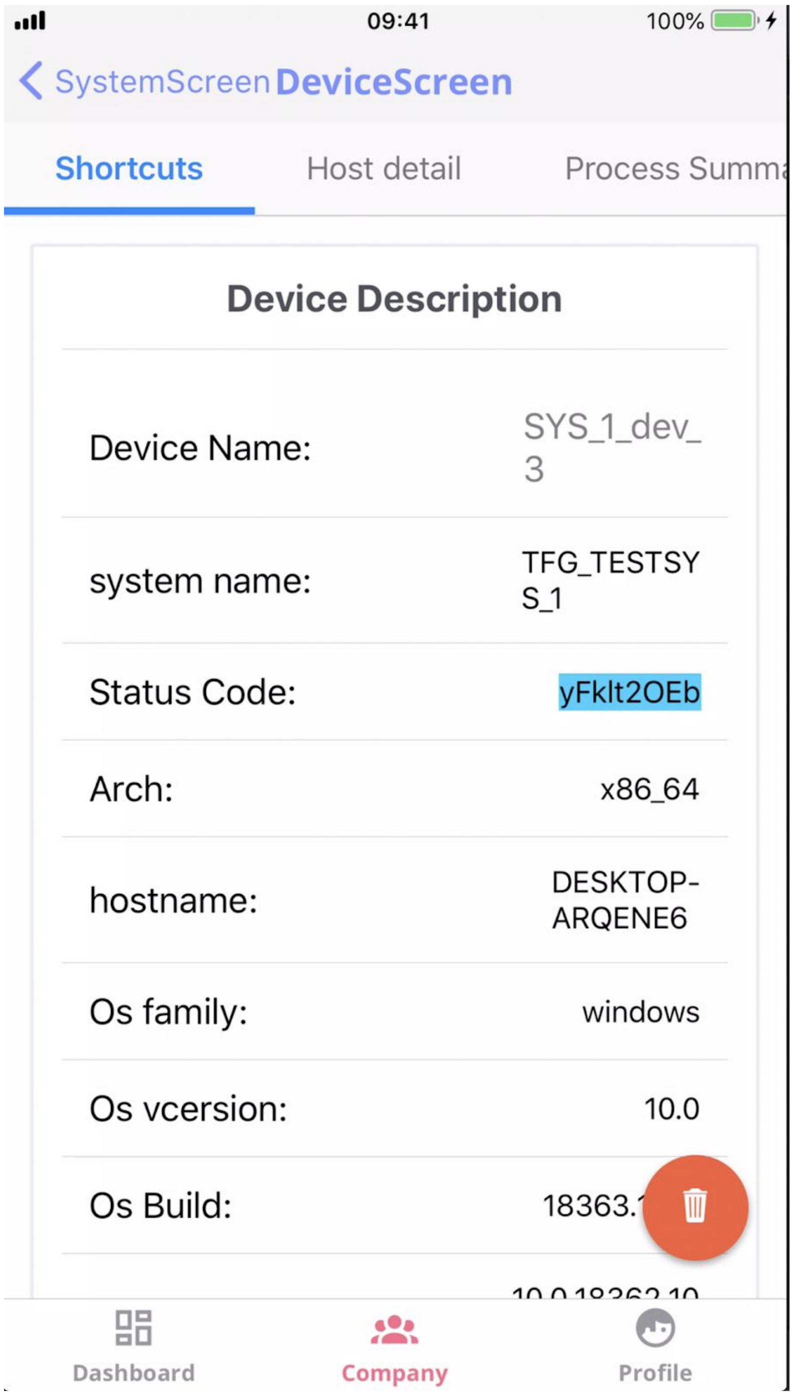
Write completed: **15006 times**

Write time spent: **0 times**

Bytes wrote: **1594929152 Bytes**

Process Summary

Figure 23: Device info page on the web version


09:41 100%

SystemScreen DeviceScreen

Shortcuts Host detail Process Summa

Device Description

Device Name:	SYS_1_dev_3
system name:	TFG_TESTSYS_1
Status Code:	yFkIt2OEb
Arch:	x86_64
hostname:	DESKTOP-ARQENE6
Os family:	windows
Os vcersion:	10.0
Os Build:	18363.1
	10.0.18363.10

Figure 24: Device info page on the mobile version

The device system uses the combination of MongoDB and the ELK stacks. On the one hand, when a device is created by a user, the server saves the device's information, such as the device id, the device's name, the device's relation to the system, and the statuscode generated by the server on MongoDB. The statuscode is nothing but a unique ID for users to use on the ELK stack, more concretely, to use it on the beat's configuration file. To send the device's status to the server and monitor it, the user has to download the beat and its configuration file provided by the application.

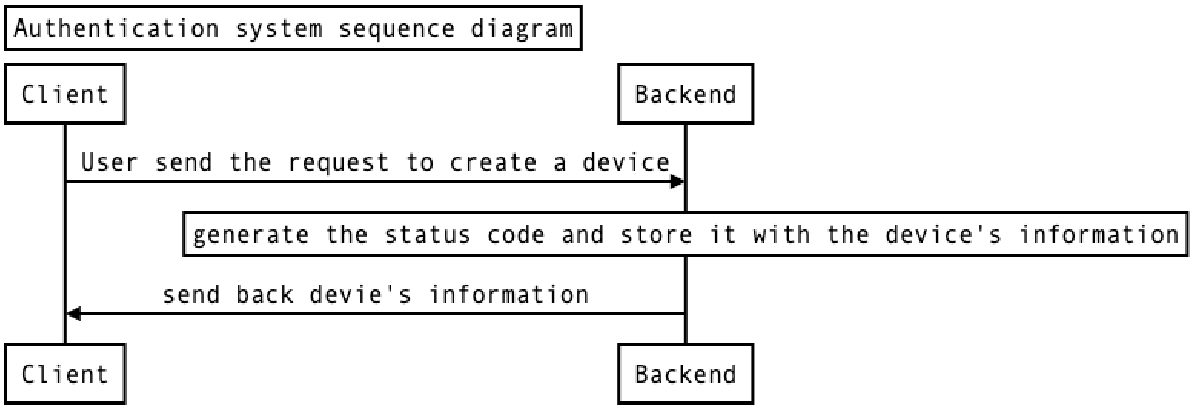
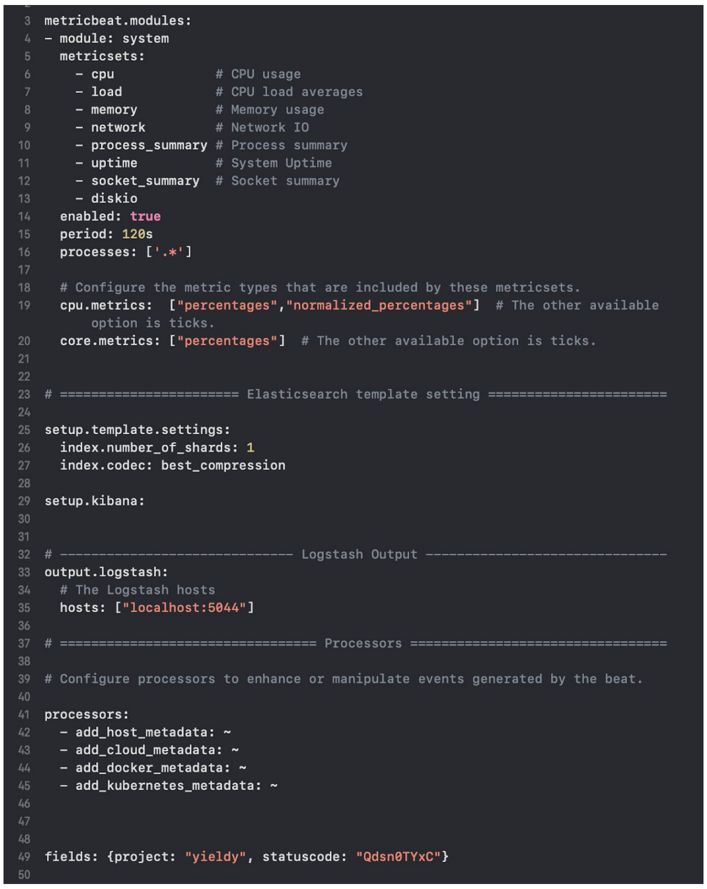

Authentication system sequence diagram
Client
Backend
User send the request to create a device
generate the status code and store it with the device's information
send back devie's information
Client
Backend

Figure 25: Sequence diagram of the creation of a device

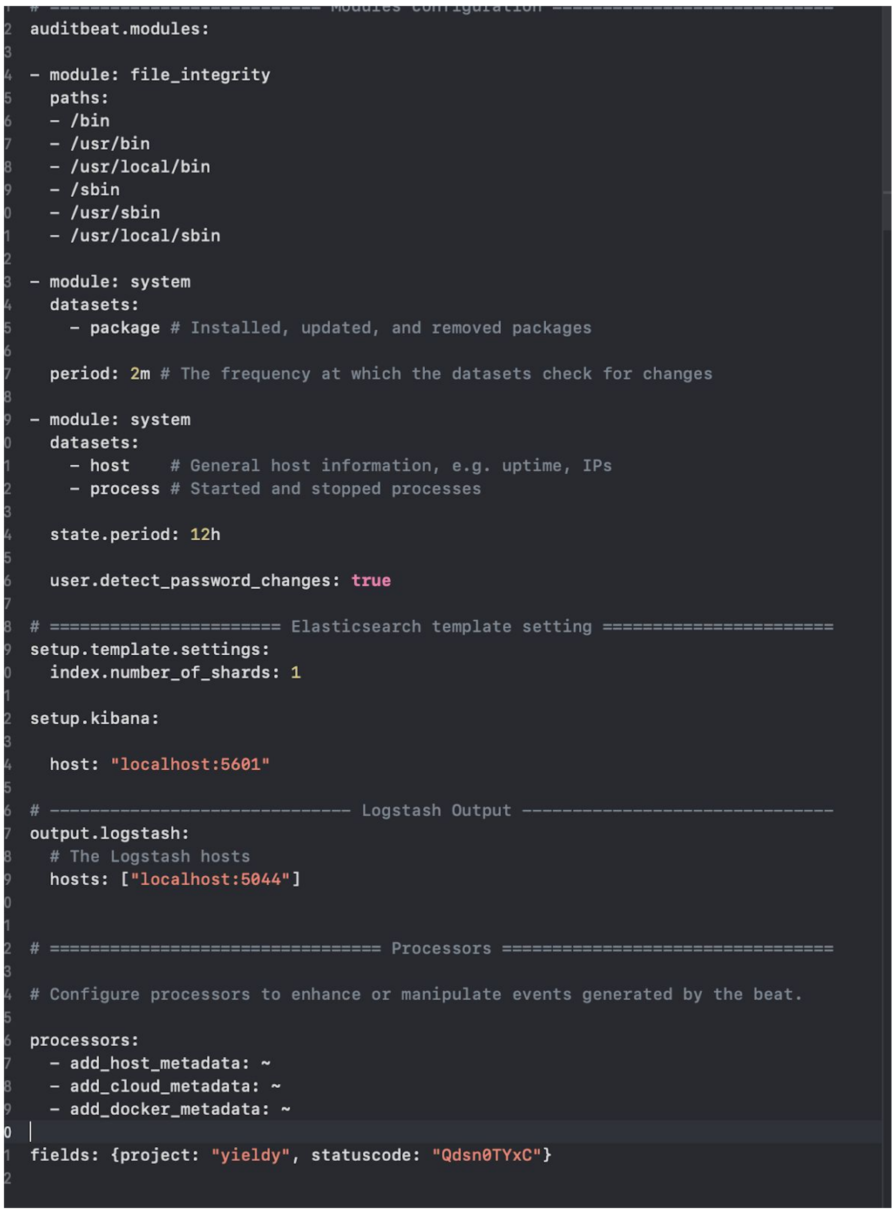
After the user creates the device, and the client displays the statuscode, the user may use it on the beat to send the device's status. To use the statuscode on the beats configuration file, the user can add a ***fields*** property that contains the project's name "yieldy" and the statuscode copied from the client. After the user configures the beat's configuration file and runs it, the process of the Metricbeat will send the device's metric data such as CPU usage, CPU load average, memory usage, process summary, uptime, socket summary, and the disk's IO metrics every 120 seconds. The data is sent to the Logstash. The Logstash server is currently running on port 5044 of the localhost when the project is implemented. However, when the ELK stack is hosted on the server, the user will send the data to a cloud-hosted server; by now, this is only running in the local network.



```
metricbeat.modules:
- module: system
  metricsets:
    - cpu             # CPU usage
    - load            # CPU load averages
    - memory          # Memory usage
    - network         # Network IO
    - process_summary # Process summary
    - uptime          # System Uptime
    - socket_summary  # Socket summary
    - diskio
  enabled: true
  period: 120s
  processes: ['.*']

  # Configure the metric types that are included by these metricsets.
  cpu.metrics:  ["percentages","normalized_percentages"]  # The other available option is ticks.
  core.metrics: ["percentages"]  # The other available option is ticks.


# ======================= Elasticsearch template setting =======================

setup.template.settings:
  index.number_of_shards: 1
  index.codec: best_compression

setup.kibana:


# ------------------------------ Logstash Output -------------------------------
output.logstash:
  # The Logstash hosts
  hosts: ["localhost:5044"]

# ================================= Processors =================================

# Configure processors to enhance or manipulate events generated by the beat.

processors:
  - add_host_metadata: ~
  - add_cloud_metadata: ~
  - add_docker_metadata: ~
  - add_kubernetes_metadata: ~



fields: {project: "yieldy", statuscode: "Qdsn0TYxC"}
```

Figure 26: Example of how to use the statuscode in metricbeats's configuration file

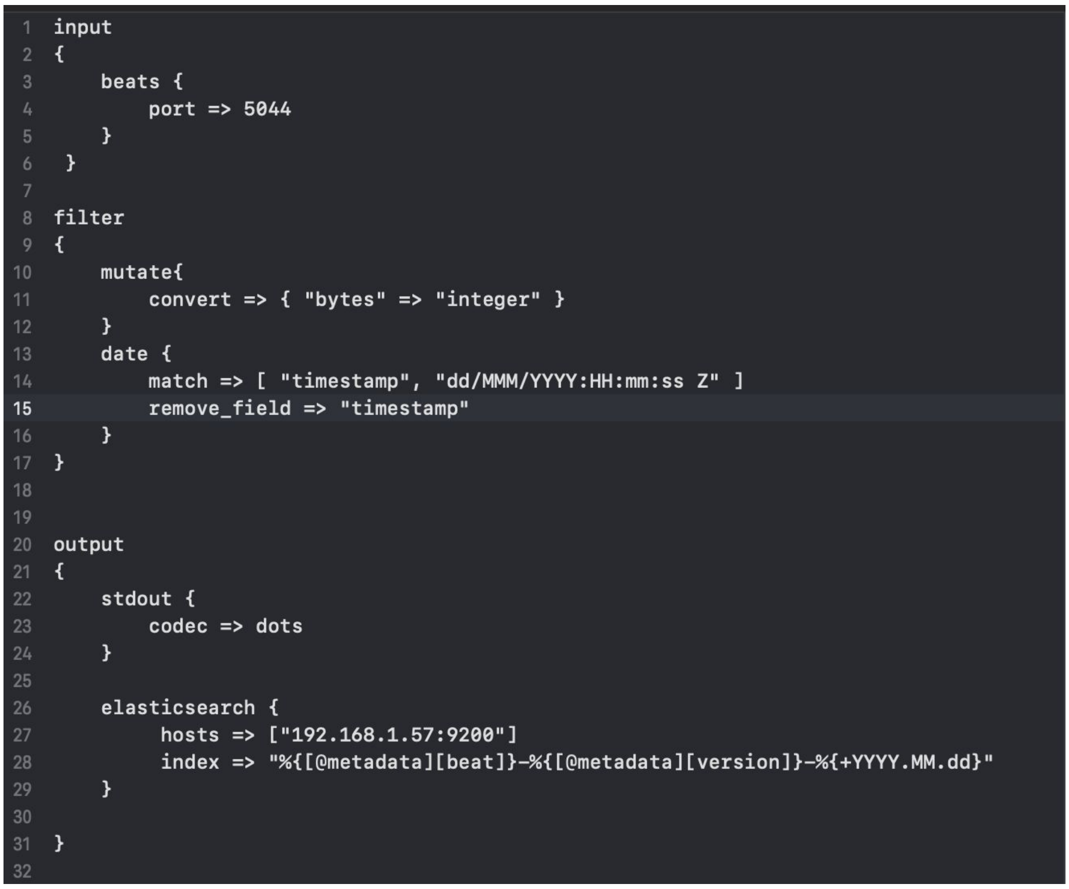
In the case of the Auditbeat, it has almost the same configuration, except the module's configuration. The file_integrity of the path is indicated in the configuration file, depending on the platform. These pieces of information are sent every 2 minutes. Moreover, the log of the host and the process are sent by Auditbeat every 12 hours.



```yaml
# ========================== Modules configuration ==========================
auditbeat.modules:

- module: file_integrity
  paths:
  - /bin
  - /usr/bin
  - /usr/local/bin
  - /sbin
  - /usr/sbin
  - /usr/local/sbin

- module: system
  datasets:
    - package # Installed, updated, and removed packages

  period: 2m # The frequency at which the datasets check for changes

- module: system
  datasets:
    - host    # General host information, e.g. uptime, IPs
    - process # Started and stopped processes

  state.period: 12h

  user.detect_password_changes: true

# ====================== Elasticsearch template setting ======================
setup.template.settings:
  index.number_of_shards: 1

setup.kibana:

  host: "localhost:5601"

# ------------------------------ Logstash Output ------------------------------
output.logstash:
  # The Logstash hosts
  hosts: ["localhost:5044"]


# ================================= Processors =================================

# Configure processors to enhance or manipulate events generated by the beat.

processors:
  - add_host_metadata: ~
  - add_cloud_metadata: ~
  - add_docker_metadata: ~

fields: {project: "yieldy", statuscode: "Qdsn0TYxC"}
```

Figure 27: Example of how to use the statuscode in auditbeat configuration file

After the log file is sent from beat to the Logstash, the Logstash will normalize the data with the predefined rule. The following Metricbeat's configuration file on the Logstash shows that the source data comes from port 5044, and the output is the Elasticsearch and stdout. The Elasticsearch output allows the normalized data to be sent to the Elasticsearch, and the stdout output is used for debugging during the development stage. The filter allows us to define a rule to normalize the data. Here the project uses the mutate filter to convert the field type from bytes to integer and then the date filter to convert the timestamp field.



```
input
{
    beats {
        port => 5044
    }
 }

filter
{
    mutate{
        convert => { "bytes" => "integer" }
    }
    date {
        match => [ "timestamp", "dd/MMM/YYYY:HH:mm:ss Z" ]
        remove_field => "timestamp"
    }
}


output
{
    stdout {
        codec => dots
    }

    elasticsearch {
         hosts => ["192.168.1.57:9200"]
         index => "%{[@metadata][beat]}-%{[@metadata][version]}-%{+YYYY.MM.dd}"
    }

}
```

Figure 28: Logstash's Metricbeat configuration file

6.6. Notification system

The push notification functionality is only available for the mobile version. One primary reason is that the push notification on the web version seems pointless. However, it is vital for mobile applications because of the accessibility and convenience of the mobile platform.

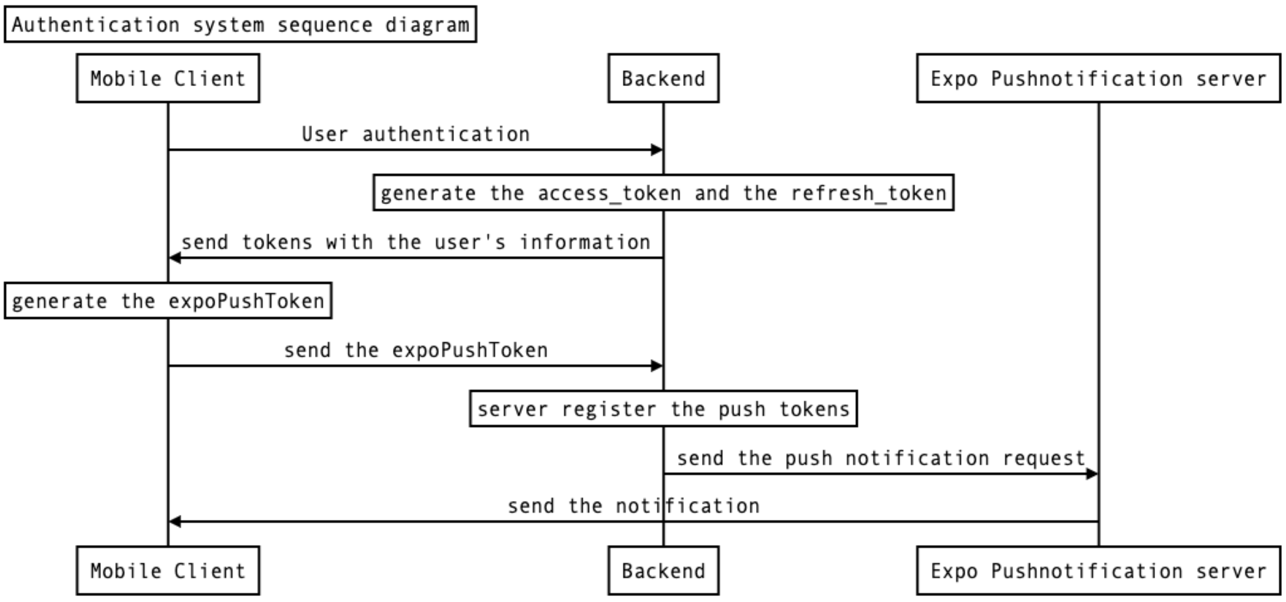

Authentication system sequence diagram
Mobile Client
Backend
Expo Pushnotification server
User authentication
generate the access_token and the refresh_token
send tokens with the user's information
generate the expoPushToken
send the expoPushToken
server register the push tokens
send the push notification request
send the notification
Mobile Client
Backend
Expo Pushnotification server

Figure 29: Notification system's sequence diagram

To receive the notification from the server, the user has to be logged into the mobile application and has the application running in the background. The push token that the expo push notification server uses for sending the notification is generated at the time when the mobile user first logged in to the application. After the token is generated, the mobile client sends the token to the backend. The backend registers the push token; then, the user will receive the push notification when an incident is produced, such as the publication of a message, the modification of the task, and the user's assignment to the system the user has the permission to access.

7. Conclusion and future work

7.1. Conclusion

To conclude, I consider that the final degree project's specific objectives have been covered and have been improved compared to the initial approach. During the implementation of this project, I acquired much knowledge of the process of how to implement a cross-platform application. The project's start was tough because I have to decide what to build and how to build it. It was also challenging because I have to learn multiple different frameworks of frontend and backend. The knowledge that I acquired during the university degree also helps me a lot during this learning phase.

Another challenging section has been the design of the application. From the beginning, it was building adaptable to future changes that become easier to implement and develop thanks to having a clear general structure and its construction from the start.

The final extension (add the ELK stack to the project) that I made at the last development phase of this project gives me many ideas on making this application a more convenient application for users. However, due to lack of time, I could not make all the changes to this project. Nevertheless, it can be considered the first step for me to get into the cybersecurity field.

7.2 Future work

There is a lot of improvement that is possible to make on this project:

1. Improve the UI design: it will be necessary to unify the web platform and mobile platform’s UI design.
2. Improve the push notification system: the user can only receive the notification related to the communication system for now. The improvement would be to connect the device system to the notification system, so when an abnormal status was detected, the related user would receive the notification about it.
3. Improve the ELK stack: it would be crucial for the user to use a real application to connect a device rather than connect it from the command line using the ELK modules. Making a script or an application for automatically configuring and executing the beats' process will improve it.
4. Make different membership plans: the application is designed to have a possibility to create other membership plans for the users by editing the limitation of the creation of systems, users, and devices. By making different membership plans will make this application ready to deploy to the market.

8. References

[1] “Get started with MongoDB.” https://docs.mongodb.com/ Accessed: 2019/10/02

[2] “Material design icons” https://material.io/ Accessed: 2019/10/04

[3]”Mongoose documents” https://mongoosejs.com/docs/guide.html Accessed: 2019/10/15

[4]”Expo vector icons” https://expo.github.io/vector-icons/ Accessed: 2019/11/16

[5]”React Native Documents” https://facebook.github.io/react-native/docs/getting-started Accessed: 2019/11/25

[6]”ReactJS tutorial” http://youtube.com/watch?v=DLX62G4lc44 Accessed: 2020/01/18

[7]”How to Build a Node.js Authentication API with Email Verification, Image Upload and Password Reset Using JWT, Passport.js and Sendgrid” https://medium.com/swlh/how-to-build-a-node-js-authentication-api-with-email-verification-image-upload-and-password-reset-95e35fd46be1 Accessed: 2020/01/25

[8] “Nodemailer documents” https://nodemailer.com/about/ Accessed: 2020/01/27

[9]”Native base UI components” https://nativebase.io/ Accessed: 2020/02/03

[10]’React native elements components” https://react-native-elements.github.io/react-native-elements/docs/getting_started.html Accessed: 2020/02/03

[11]”ReactJS tutorial” https://www.youtube.com/watch?v=fiQh6xUDBBQ&list=PLo5lAe9kQrwrGPjhhzejCt3JENYf5uDNf&index=9 Accessed: 2020/03/04

[12]’ReactJS styled component library” https://github.com/styled-components/styled-components Accessed: 2020/03/12

[13]”Font Awesome” https://fontawesome.com/ Accessed: 2020/03/16

[14]”ReactJS tutorial” https://www.udemy.com/course/react-the-complete-guide-incl-redux/ Accessed: 2020/03/17

[15]”Reactstrap library” https://reactstrap.github.io/ Accessed: 2020/03/31

[16]”Configuring ESLint” https://eslint.org/docs/2.13.1/user-guide/configuring Accessed: 2020/04/05

[17]”CSS Tutorial” https://www.w3schools.com/css/default.asp Accessed: 2020/04/05

[18]”MATERIAL-UI” https://material-ui.com/ Accessed: 2020/04/07

[19]”Ant design” https://ant.design/ Accessed: 2020/05/02

[20]”Beats tutorial” https://logz.io/blog/beats-tutorial/ Accessed: 2020/06/01

[21]”Elasticsearch service documentation” https://www.elastic.co/guide/en/cloud/current/index.html Accessed: 2020/06/19

[22]”Auditbeat Reference [7.8]” https://www.elastic.co/guide/en/beats/auditbeat/7.8/auditbeat-configuration.html Accessed: 2020/07/19

[23]”Logstash Reference [7.9] » Input plugins” https://www.elastic.co/guide/en/logstash/current/input-plugins.html Accessed: 2020/07/21

[24]”Elasticsearch Node.js client [7.x] » Introduction” https://www.elastic.co/guide/en/elasticsearch/client/javascript-api/current/introduction.html Accessed: 2020/07/23

9. Annex

9.1. Web version’s use cases

9.1.1. Account subsystem’s use cases

<u>Register as Company Owner</u>

Actors	Not connected user
Description	Register as company owner
Preconditions	-
Postconditions	User is registered to the system.
Main Course	1. User access to the registration page to register as a company owner by clicking the registration link from the landing page. 2. User enters the required data to the registration form (see EX1). 3. System sends the confirmation email to the user's email address. 4. User clicks the confirmation link from the email to activate the account.
Alternative Course	-
Exceptions	EX1.. The data entered by user is not valid 1. Return to Main Course step 2.

<u>Log in to web</u>

Actors	Not connected user

Description	Log in to web
Preconditions	User has to be registered to the system.
Postconditions	User is logged in to the system.
Main Course	1. User access to the landing page and click the login link (see AC1, AC2). 2. User enters the email and password to the login form (see EX1,EX2,EX3,AC3). 3. User is redirected to the homepage after hitting the login button.
Alternative Course	AC1. User clicks the “Company Owner?” link in order to log in as a company owner. 1. Return to Main Course step 2 AC2. User clicks the ”Normal User?” link in order to log in as Admins or technicians. 1. Return to Main Course step 2 AC3. Logged user is a company owner and it is the first login after activating the account. 1. Web displays the company registration form for the user to create the company first (see EX4,EX5).
Exceptions	EX1. The credential is invalid. 1. Return user to Main Course step 2. EX2. The account is a company owner and has not been activated. 1. System displays the message asking the user to resend the confirmation email. EX3. The account is a normal user and has not been activated. 1. System displays the error and asks the user to confirm the account. EX4. The company registration form is valid. 1. Return user to Main Course step 3. EX5. The company registration form is invalid. 1. Return user to AC3.

Log out from web

Actors	Logged user
Description	Logged user wants to log out from the system.
Preconditions	User has to be registered and logged in to the system.
Postconditions	User will be logged out from the system.
Main Course	1. Logged user clicks the user image popover. 2. User click the “sign out” button. 3. User is logged out and redirected to the landing page.
Alternative Course	-
Exceptions	-

Modify user’s profile

Actors	Logged user
Description	Logged user wants to change his name or profile image.
Preconditions	User has to be registered and logged in to the system.
Postconditions	User’s profile information is updated.
Main Course	1. User clicks the Profile page from the left navigation menu(see AC1, AC2). 2. The new user’s profile is updated.
Alternative Course	AC1. User clicks the “Edit Profile Image” button. 1. User clicks the “Choose file” button on the modal page. 2. User chose the image file to upload. 3. User clicks the “Upload” button. 4. Return to Main Course step 2. AC2. User clicks the “Edit” to edit the username. 1. User enters a new name and clicks the update button (see EX1) 2. Return to Main Course step 2.
Exceptions	EX1. The user is a company owner and the user enters an

	existing user name. 1. System displays the error message and returns the user the Profile page.

Create Administrator/Technicians

Actors	Logged User as Company Owner
Description	Logged company owner wants to create a normal user (administrator/technician) for the company .
Preconditions	User has to be registered as company owner and logged in to the system.
Postconditions	A new administrator/technician is created for the company.
Main Course	1. User clicks the Company page from the left navigation menu. 2. User clicks the “Add User ” button on the Member card. 3. User enters the email to receive confirmation email and selects the type of user he wants to create in the modal form then clicks the “Create” button (see EX1). 4. A new user is created and the logged user is redirected to the “Company” page.
Alternative Course	-
Exceptions	EX1. User reached the limit of the creation of accounts. 1. System shows the error message. 2. Return to Main Course step 1.

Delete Administrator/Technicians

Actors	Logged User as Company Owner
Description	Logged company owner wants to delete a normal user (administrator/technician) from the company.
Preconditions	User has to be registered as company owner and logged in to the system.
Postconditions	An administrator/technician is deleted from the company.

Main Course	1. User clicks the Company page from the left navigation menu. 2. User clicks the user list item that he wants to delete from the Member card. 3. System shows the modal page of user’s profile 4. User click the “Delete User ” button from the modal page. 5. The user is deleted and the logged user is redirected to the “Company” page.
Alternative Course	-
Exceptions	-

9.1.2. Management subsystem’s use cases

Modify company’s information

Actors	Logged User as Company Owner
Description	Logged company owner wants to modify the company's information.
Preconditions	User has to be registered as company owner and logged in to the system.
Postconditions	The company information is updated.
Main Course	1. User clicks the Company page from the left navigation menu. 2. User selects the “Edit” button from the Company Card. 3. System shows the modal page of the company's information form. 4. User edits the information and clicks the “Update” button. 5. Company information is updated.
Alternative Course	-
Exceptions	-

Create system

Actors	Logged User as Company Owner
Description	Logged company owner wants to create a system (company's subgroup) for the company .
Preconditions	User has to be registered as company owner and logged in to the system.
Postconditions	A system is created for the company.
Main Course	1. User clicks the "Company" page from the left navigation menu. 2. User clicks the "Create" button on the System card. 3. User enters the system name in the modal form and clicks the "Create" button (see EX1). 4. A new system (company's subgroup) is created and the page refreshes.
Alternative Course	-
Exceptions	EX1. User reached the limit of the creation of the system(company's subgroup) . 1. System shows the error message. 2. Return to Main Course step 1.

Delete system

Actors	Logged User as Company Owner
Description	Logged company owner wants to delete a system (company's subgroup) from the company .
Preconditions	User has to be registered as company owner and logged in to the system.
Postconditions	A system (company's subgroup) is deleted from the company.
Main Course	1. User clicks the Company page from the left navigation menu. 2. User selects the system (company's subgroup) that he wants to delete from the System card. 3. User clicks the "DELETE" button on the System card. 4. The system is deleted and the logged user is redirected to the "Company" page.

Alternative Course	-
Exceptions	-

Add device to System

Actors	Logged User as Company Owner or Administrator
Description	Logged user wants to add a device to a system (company's subgroup) .
Preconditions	User has to be registered as company owner or administrator and logged in to the system.
Postconditions	A device is added to the system (company's subgroup).
Main Course	1. User clicks the System page from the left navigation menu. 2. User select the system (company's subgroup) by clicking the button that displays the system name (see EX1). 3. User clicks the "Connect" button on the Devices card. 4. User enters the device name in the modal form and clicks the "Create" button (see EX2). 5. A new device is created by the system.
Alternative Course	-
Exceptions	EX1. The user as administrator doesn't have permission to access the system (company's subgroup) . 1. System displays the error page. EX2. User reached the limit of the creation of the device . 1. System shows the error message. 2. Return to Main Course step 1.

Remove device from System

Actors	Logged User as Company Owner or Administrator
Description	Logged user wants to delete a device from a system

	(company’s subgroup) .
Preconditions	User has to be registered as company owner or administrator and logged in to the system.
Postconditions	A device is deleted from the system (company’s subgroup).
Main Course	1. User clicks the Device page from the left navigation menu. 2. User selects the device by selecting the device from the tabs of the page(see EX1). 3. User clicks the “Remove” button from the Device Info card. 4. System displays a popover asking the user to confirm the deletion. 5. User clicks the “confirm” button. 6. System removes the device and refreshes the page.
Alternative Course	-
Exceptions	EX1. The user as administrator doesn’t have permission to access the device’s system (company’s subgroup) . 1. System displays the error page.

<u>Assign technicians to System</u>

Actors	Logged User as Company Owner
Description	Logged user wants to assign a technician to a system (company’s subgroup) .
Preconditions	User has to be registered as company owner and logged in to the system.
Postconditions	A technician is assigned to the system (company’s subgroup).
Main Course	1. User clicks the System page from the left navigation menu. 2. User selects the system (company’s subgroup) by clicking the button that displays the system name. 3. User clicks the “Manage” button on the Technicians list from the Members card. 4. System shows the modal page for the user to edit. 5. User selects the user by clicking the “Add” button from the Available Member Card. (see EX1).

	6. The selected user is assigned to the system (company's subgroup).
Alternative Course	-
Exceptions	EX1. User reached the limit of technicians for the system(company's subgroup). 1. System shows the error message. 2. Return to Main Course step 1.

Revoke technicians from System

Actors	Logged User as Company Owner
Description	Logged user wants to revoke a technician from a system (company's subgroup) .
Preconditions	User has to be registered as company owner and logged in to the system.
Postconditions	A technician is revoked from the system (company's subgroup).
Main Course	1. User clicks the System page from the left navigation menu. 2. User selects the system (company's subgroup) by clicking the button that displays the system name. 3. User clicks the "Manage" button on the Technicians list from the Members card. 4. System shows the modal page for the user to edit. 5. User selects the user by clicking the "Remove" button from the System's Member Card. 6. The selected user is revoked from the system(company's subgroup) .
Alternative Course	-
Exceptions	-

Assign administrators to System

Actors	Logged User as Company Owner
Description	Logged user wants to assign an administrator to a system (company’s subgroup) .
Preconditions	User has to be registered as company owner and logged in to the system.
Postconditions	An administrator is assigned to the system (company’s subgroup).
Main Course	1. User clicks the System page from the left navigation menu. 2. User selects the system (company’s subgroup) by clicking the button that displays the system name. 3. User clicks the “Manage” button on the Admins list from the Members card. 4. System shows the modal page for the user to edit. 5. User selects the user by clicking the “Add” button from the Available Member Card. (see EX1). 6. The selected user is assigned to the system(company’s subgroup) .
Alternative Course	-
Exceptions	EX1. User reached the limit of administrators for the system(company’s subgroup). 1. System shows the error message. 2. Return to Main Course step 1.

Revoke administrators from System

Actors	Logged User as Company Owner
Description	Logged user wants to revoke an administrator from a system (company’s subgroup) .
Preconditions	User has to be registered as company owner and logged in to the system.
Postconditions	An administrator is revoked from the system (company’s subgroup).
Main Course	1. User clicks the System page from the left navigation menu.

	2. User selects the system (company’s subgroup) by clicking the button that displays the system name. 3. User clicks the “Manage” button on the Admins list from the Members card. 4. System shows the modal page for the user to edit. 5. User selects the user by clicking the “Remove” button from the System’s Member Card. 6. The selected user is revoked from the system(company’s subgroup).
Alternative Course	-
Exceptions	-

9.1.3. Communication subsystem’s use cases

<u>Publish/Reply post</u>

Actors	Logged User
Description	Logged user wants to publish a post to a system (company’s subgroup).
Preconditions	User has to be registered and logged in to the system.
Postconditions	A post is published to the system
Main Course	1. User clicks the System page from the left navigation menu. 2. User selects the system (company’s subgroup) by clicking the button that displays the system name (see EX1). 3. User clicks the message FAB (float action button).. 4. System shows the right side drawer for the user to publish posts (see AC1,AC2). 5. User enters the post content and clicks the “Publish” button. 6. A post is published to the system.
Alternative Course	AC1. User wants to publish a new post. 1. User clicks the “New post” button (see EXT1).

	AC2. User wants to reply to an existing post. 1. User clicks the "edit" button to reply to a post. 2. Return to Main Course step 5.
Extensions	EXT1. User wants to mention a device or task. 1. User clicks the "@" button. 2. System shows the modal page for the user to select device and task. 3. User selects device/task to mention and clicks. 4. Return to Main Course step 5.
Exceptions	EX1. The user has a type of administrator or technician and doesn't have permission to access the system (company's subgroup) . 1. System displays the error page.

View a post

Actors	Logged User
Description	Logged user wants to view a post in a system (company's subgroup).
Preconditions	User has to be registered and logged in to the system.
Postconditions	-
Main Course	1. User clicks the System page from the left navigation menu. 2. User selects the system (company's subgroup) by clicking the button that displays the system name (see EX1). 3. User clicks the message FAB (float action button).. 4. System shows the right side drawer for the user to view posts (see AC1).
Alternative Course	AC1. User wants to view the details of a post. 1. User clicks the "detail" button of a post. 2. System shows the post detail in a modal page..
Extensions	-

Exceptions	EX1. The user has a type of administrator or technician and doesn't have permission to access the system (company's subgroup) . 1. System displays the error page.

Manage task

Actors	Logged User as Company Owner or Administrator
Description	Logged user wants to manage a task in a system (company's subgroup).
Preconditions	User has to be registered as company owner or administrators and logged in to the system.
Postconditions	A task's content is updated or created
Main Course	1. User clicks the System page from the left navigation menu. 2. User selects the system (company's subgroup) by clicking the button that displays the system name (see EX1, AC1,AC2,AC3). 3. A task is updated/created to the system (company's subgroup) .
Alternative Course	AC1. User wants to create a new task. 1. User clicks the "Create New task" button (see EXT1). 2. User selects a due date and enters the content. 3. User clicks the "Create" button. 4. Return to Main Course step 3. AC2. User wants to assign/remove a technician to/from an existing task. 1. User clicks the task in the task board. 2. System displays the task information modal page. 3. User clicks the switch button of the user list item. 4. Return to Main Course step 3. AC3. User wants to change the task's status. 1. User clicks the task in the task board. 2. System displays the task information modal page. 3. User clicks the switch button of the change status list item.

	4. Return to Main Course step 3.
Extensions	-
Exceptions	EX1. The user has a type of administrator and doesn't have permission to access the system (company's subgroup) . 1. System displays the error page.

View a task

Actors	Logged User
Description	Logged user wants to view a task in a system (company's subgroup).
Preconditions	User has to be registered and logged in to the system.
Postconditions	-
Main Course	1. User clicks the System page from the left navigation menu. 2. User selects the system (company's subgroup) by clicking the button that displays the system name (see EX1). 3. System displays the task board to the user (see AC1)..
Alternative Course	AC1. User wants to view the task's details. 1. User clicks the task. 2. System displays the task's details on a modal page.
Extensions	-
Exceptions	EX1. The user has a type of administrator or technician and doesn't have permission to access the system (company's subgroup) . 1. System displays the error page.

9.1.4. Other functionalities

Monitor device

Actors	Logged User

Description	Logged user wants to monitor a device in a system (company's subgroup).
Preconditions	User has to be registered and logged in to the system.
Postconditions	-
Main Course	1. User clicks the System page from the left navigation menu. 2. User selects the device by selecting the device from the tabs of the page(see EX1). 3. System displays the device information to the user.
Alternative Course	-
Extensions	-
Exceptions	EX1. The user has a type of administrator or technicians and doesn't have permission to access the system (company's subgroup) . 1. System displays the error page.

9.2. Mobile version's use cases

9.2.1. Account subsystem's use cases

Register as Company Owner

Actors	Not connected user
Description	Register as company owner
Preconditions	-
Postconditions	User is registered to the system.
Main Course	1. User clicks the "Sign up as Company Owner" button from the auth screen. 2. System redirects the user to the registration screen. 3. User enters the required data to the registration screen (see EX1). 4. System sends the confirmation email to the user's email

	address. 5. User clicks the confirmation link from the email to activate the account.
Alternative Course	-
Exceptions	EX1. The data entered by user is not valid 1. Return to Main Course step 3.

Log in to application

Actors	Not connected user
Description	Log in to application
Preconditions	User has to be registered to the system.
Postconditions	User is logged in to the application.
Main Course	1. User accesses the auth screen and clicks the login button (see AC1, AC2). 2. User enters the email and password to the login form (see EX1,EX2,EX3,AC3). 3. User is redirected to the homepage.
Alternative Course	AC1. User clicks the “Company Owner?” button in order to log in as a company owner. 1. Return to Main Course step 2 AC2. User clicks the ”Normal User?” button in order to log in as Admins or technicians. 1. Return to Main Course step 2 AC3. Logged user is a company owner and it is the first login after activating the account. 1. Application displays the company registration form for users to create the company first (see EX4,EX5).
Exceptions	EX1. The credential is invalid. 1. Return user to Main Course step 2. EX2. The account is a company owner and has not been activated. 1. System displays the message asking the user to

	resend the confirmation email. EX3. The account is a normal user and has not been activated. 1. System displays the error and asks the user to activate the account. EX4. The company registration form is valid. 1. Return to Main Course step 3. EX5. The company registration form is invalid. 1. Return to AC3.

Log out from application

Actors	Logged user
Description	Logged user wants to log out from the system.
Preconditions	User has to be registered and logged in to the system.
Postconditions	User will be logged out from the system.
Main Course	1. Logged user clicks the Profile tab. 2. User clicks the “Log out” button. 3. User is logged out and redirected to the auth screen.
Alternative Course	-
Exceptions	-

Modify user’s profile

Actors	Logged user
Description	Logged user wants to change his name.
Preconditions	User has to be registered and logged in to the system.
Postconditions	User’s profile information is updated.
Main Course	1. Logged user clicks the Profile tab. 2. User clicks the “Edit username” button. 3. System shows a dialog for the user to upload his

	username. 4. User enters a new name and clicks the update button (see EX1). 5. System updates the new user's profile.
Alternative Course	-
Exceptions	EX1. The account is a company owner and the user enters an existing user name. 1. System displays the error message and returns the user the Profile page.

Create Administrator/Technicians

Actors	Logged User as Company Owner
Description	Logged company owner wants to create a normal user(administrator/technician) for the company.
Preconditions	User has to be registered as company owner and logged in to the system.
Postconditions	A new administrator/technician is created for the company.
Main Course	1. User selects the "Company" tab from the bottom tab bar. 2. User clicks the user icon FAB (float action button) . 3. System redirects the user to "CompanyMember" screen. 4. User clicks the plus icon FAB (float action button) . 5. User enters the email to receive verification email and selects the type of user he wants to create in the modal form then clicks the "SUBMIT" button (see EX1). 6. A new user is created and the logged user is redirected to the "Company" page.
Alternative Course	-
Exceptions	EX1. User reached the limit of creation of accounts. 1. System shows the error message. 2. Return to Main Course step 1.

Delete Administrator/Technicians

Actors	Logged User as Company Owner
Description	Logged company owner wants to delete a normal user (administrator/technician) from the company.
Preconditions	User has to be registered as company owner and logged in to the system.
Postconditions	An administrator/technician is deleted from the company.
Main Course	1. User selects the “Company” tab from the bottom tab bar. 2. User clicks the user icon FAB (float action button) . 3. System redirects the user to “CompanyMember” screen. 4. User swipes the list item to the right of the user that he wants to delete. 5. User clicks the delete button. 6. The user is deleted and the system shows the success toast message to the user.
Alternative Course	-
Exceptions	-

9.2.2. Management subsystem’s use cases

Modify company’s information

Actors	Logged User as Company Owner
Description	Logged company owner wants to modify the company's information.
Preconditions	User has to be registered as company owner and logged in to the system.
Postconditions	The company information is updated.
Main Course	1. User selects the “Company” tab from the bottom tab bar. 2. User selects the field from the Company Card. 3. System shows the dialog form for the user to update the selected field.

	4. User edits the information and clicks the “Update” button. 5. Company information is updated.
Alternative Course	-
Exceptions	-

<u>Create System</u>

Actors	Logged User as Company Owner
Description	Logged company owner wants to create a system (company’s subgroup) for the company .
Preconditions	User has to be registered as company owner and logged in to the system.
Postconditions	A system is created for the company.
Main Course	1. User selects the “Company” tab from the bottom tab bar. 2. User clicks the “CREATE” button on the System card. 3. User enters the system name in the dialog form and clicks the “SUBMIT” button (see EX1). 4. A new system (company’s subgroup) is created and the logged user is redirected to the “Company” screen.
Alternative Course	-
Exceptions	EX1. User reached the limit of creation of the system (company’s subgroup). 1. System shows the error message. 2. Return to Main Course step 1.

<u>Delete system</u>

Actors	Logged User as Company Owner
Description	Logged company owner wants to delete a system (company’s subgroup) from the company .
Preconditions	User has to be registered as company owner and logged in to the system.

Postconditions	A system (company's subgroup) is deleted from the company.
Main Course	1. User selects the "Company" tab from the bottom tab bar. 2. User selects the system (company's subgroup) that he wants to delete from the System card. 3. System redirects the user to "System" screen 4. User clicks the "Delete System?" button on the System card. 5. The system is deleted and the logged user is redirected to the "Company" screen.
Alternative Course	-
Exceptions	-

<u>Add device to System</u>

Actors	Logged User as Company Owner or Administrator
Description	Logged user wants to add a device to a system (company's subgroup) .
Preconditions	User has to be registered as company owner or administrator and logged in to the system.
Postconditions	A device is added to the system (company's subgroup).
Main Course	1. User selects the "Company" tab from the bottom tab bar. 2. User selects the system (company's subgroup) that he wants to add a device to (see EX1). 3. User is redirected to the selected system screen. 4. User selects the "Connect" button from "Device's List" tab. 5. User enters the device name in the dialog form and clicks the "SUBMIT" button (see EX2). 6. A new device (company's subgroup) is created and the logged user is redirected to the previous screen.
Alternative Course	-

Exceptions	EX1. The user as administrator doesn’t have permission to access the system (company’s subgroup) . 1. System displays the error page and redirects the user to the previous screen. EX2. User reached the limit of creation of the device . 1. System shows the error message. 2. Return to Main Course step 1.

Remove device from System

Actors	Logged User as Company Owner or Administrator
Description	Logged user wants to delete a device from a system (company’s subgroup) .
Preconditions	User has to be registered as company owner or administrator and logged in to the system.
Postconditions	A device is deleted from the system (company’s subgroup).
Main Course	1. User selects the “Company” tab from the bottom tab bar. 2. User selects the system (company’s subgroup) that he wants to remove a device from (see EX1). 3. User is redirected to the selected system screen. 4. User clicks the “More” button of the device that he wants to delete from the “Device’s List” tab. 5. System redirects the user to the device's detail screen. 6. User clicks the delete FAB(float action button). 7. System displays the dialog to ask the user to confirm the deletion. 8. User clicks the “SUBMIT” button. 9. System removes the device and redirects the user to the previous page.
Alternative Course	-
Exceptions	EX1. The user as administrator doesn’t have permission to access the device’s system (company’s subgroup) . 1. System displays the error message and redirects the user to the previous screen..

Assign technicians to System

Actors	Logged User as Company Owner
Description	Logged user wants to assign a technician to a system (company's subgroup) .
Preconditions	User has to be registered as company owner and logged in to the system.
Postconditions	A technician is assigned to the system (company's subgroup).
Main Course	1. User selects the "Company" tab from the bottom tab bar. 2. User selects the system (company's subgroup) that he wants to add a technician to. 3. User is redirected to the selected system screen. 4. User clicks the "EDIT" button from the "Technician's List" tab. 5. System redirects the user to the "EditSystemMemberScreen". 6. User selects the user he wants to add to the system by clicking the "Assign" button of the user (see EX1). 7. The selected user is assigned to the system(company's subgroup) .
Alternative Course	-
Exceptions	EX1. User reached the limit of technicians for the system(company's subgroup). 1. System shows the error message. 2. Return to Main Course step 1.

Revoke technicians from System

Actors	Logged User as Company Owner
Description	Logged user wants to revoke a technician from a system (company's subgroup) .
Preconditions	User has to be registered as company owner and logged in to the system.

Postconditions	A technician is revoked from the system (company's subgroup).
Main Course	1. User selects the "Company" tab from the bottom tab bar. 2. User selects the system (company's subgroup) that he wants to revoke a technician from. 3. User is redirected to the selected system screen. 4. User clicks the "EDIT" button from the "Technician's List" tab. 5. System redirects the user to the "EditSystemMemberScreen". 6. User selects the user he wants to remove from the system by clicking the "Revoke" button of the user. 7. The selected user is revoked from the system (company's subgroup) .
Alternative Course	-
Exceptions	-

<u>Assign administrators to System</u>

Actors	Logged User as Company Owner
Description	Logged user wants to assign an administrator to a system (company's subgroup) .
Preconditions	User has to be registered as company owner and logged in to the system.
Postconditions	An administrator is assigned to the system (company's subgroup).
Main Course	1. User selects the "Company" tab from the bottom tab bar. 2. User selects the system (company's subgroup) that he wants to add an administrator to. 3. User is redirected to the selected system screen. 4. User clicks the "EDIT" button from the "Admin's List" tab. 5. System redirects the user to the

	“EditSystemMemberScreen”. 6. User selects the user he wants to add to the system by clicking the “Assign” button of the user (see EX1). 7. The selected user is assigned to the system (company’s subgroup) .
Alternative Course	-
Exceptions	EX1. User reached the limit of administrators for the system(company’s subgroup). 1. System shows the error message. 2. Return to Main Course step 1.

Revoke administrators from System

Actors	Logged User as Company Owner
Description	Logged user wants to revoke an administrator from a system (company’s subgroup) .
Preconditions	User has to be registered as company owner and logged in to the system.
Postconditions	An administrator is revoked from the system (company’s subgroup).
Main Course	1. User selects the “Company” tab from the bottom tab bar. 2. User selects the system (company’s subgroup) that he wants to remove an administrator from. 3. User is redirected to the selected system screen. 4. User clicks the “EDIT” button from the “Admin’s List” tab. 5. System redirects the user to the “EditSystemMemberScreen”. 6. User selects the user he wants to remove from the system by clicking the “Revoke” button of the user. 7. The selected user is revoked from the system (company’s subgroup) .
Alternative Course	-

Exceptions	-

9.2.3. Communication subsystem's use cases

Publish/Reply post

Actors	Logged User
Description	Logged user wants to publish a post to a system (company's subgroup).
Preconditions	User has to be registered and logged in to the system.
Postconditions	A post is published to the system
Main Course	1. User selects the "Company" tab from the bottom tab bar. 2. User selects the system (company's subgroup) that he wants to publish a message to (see EX1). 3. User is redirected to the selected system screen. 4. User clicks the FAB. 5. User clicks the Message FAB. 6. System redirects the user to the "MessageScreen" (see AC1, AC2). 7. User enters the post content and clicks the "Publish" button. 8. System redirects the user to the previous screen.
Alternative Course	AC1. User wants to publish a new post. 1. User clicks the FAB (see EXT1,EXT2). AC2. User wants to reply to an existing post. 1. User clicks the "reply" button to reply to a post. 2. Return to Main Course step 5.
Extensions	EXT1. User wants to mention a device. 1. User clicks the "@DEVICE" button. 2. System shows the dialog for the user to select the device to mention. 3. User selects the device to mention. 4. Return to Main Course step 7.

	EXT2. User wants to mention a task. 1. User clicks the “@TASK” button. 2. System shows the dialog for the user to select the task to mention. 3. User selects the device to mention. 4. Return to Main Course step 7.
Exceptions	EX1. The user has a type of administrator or technician and doesn’t have permission to access the system (company’s subgroup) . 1. System displays the error message and redirects the user to the previous screen.

View a post

Actors	Logged User
Description	Logged user wants to view a post in a system (company’s subgroup).
Preconditions	User has to be registered and logged in to the system.
Postconditions	-
Main Course	1. User selects the “Company” tab from the bottom tab bar. 2. User selects the system (company’s subgroup) that he wants to view the post from (see EX1). 3. User is redirected to the selected system screen. 4. User clicks the FAB. 5. User clicks the Message FAB. 6. System redirects the user to the “MessageScreen” (see AC1).
Alternative Course	AC1. User wants to view the details of a post. 1. User clicks the “detail” button of a post. 2. System shows the post details in a new screen.
Extensions	-

Exceptions	EX1. The user has a type of administrator or technician and doesn’t have permission to access the system (company’s subgroup) . 1. System displays the error message and redirects the user to the previous screen.

Manage task

Actors	Logged User as Company Owner or Administrator
Description	Logged user wants to manage a task in a system (company’s subgroup).
Preconditions	User has to be registered as company owner or administrators and logged in to the system.
Postconditions	A task's content is updated or created
Main Course	1. User selects the “Company” tab from the bottom tab bar. 2. User selects the system (company’s subgroup) that he wants to manage the task (see EX1). 3. User is redirected to the selected system screen. 4. User clicks the FAB. 5. User clicks the Task FAB. 6. System redirects the user to the “TaskScreen” (see AC1,AC2,AC3,AC4).
Alternative Course	AC1. User wants to create a new task. 1. User clicks the create task FAB . 2. User selects a due date and enters the content. 3. User clicks the “Create” button. 4. Return to Main Course step 3. AC2. User wants to revoke a technician from an existing task. 1. User clicks the task in the task list. 2. System redirects the user to the “TaskDetailScreen”. 3. User clicks the “Revoke” button of each user that he wants to revoke. 4. Return to Main Course step 3. AC3. User wants to assign a technician to an existing task. 1. User clicks the task in the task list.

	2. System redirects the user to the “TaskDetailScreen”. 3. User clicks the “Assign” button. 4. System displays the “Assign Member” dialog for the user to select the user to assign. 5. User clicks the user to assign (see EX2). 6. Return to Main Course step 3. AC4. User wants to change the task's status. 1. User clicks the task in the task list. 2. System redirects the user to the “TaskDetailScreen”. 3. User clicks the switch FAB of the change status of the selected task. 4. Return to Main Course step 3.
Extensions	-
Exceptions	EX1. The user has a type of administrator and doesn’t have permission to access the system (company’s subgroup) . 1. System displays the error message and redirects the user to the previous screen. EX2. User reached the limit of technicians assigned for the task. 1. System shows the error message. 2. Return to AC2 step 2.

<u>View a task</u>

Actors	Logged User
Description	Logged user wants to view a task in a system (company’s subgroup).
Preconditions	User has to be registered and logged in to the system.
Postconditions	-
Main Course	1. User selects the “Company” tab from the bottom tab bar. 2. User selects the system (company’s subgroup) that he wants to view the task from (see EX1). 3. User is redirected to the selected system screen. 4. User clicks the FAB. 5. User clicks the Task FAB.

	6. System redirects the user to the "TaskScreen" (see AC1).
Alternative Course	AC1. User wants to view the task's details. 1. User clicks the task. 2. System redirects the user to the "TaskDetailScreen" of the selected task.
Extensions	-
Exceptions	EX1. The user has a type of administrator or technician and doesn't have permission to access the system (company's subgroup) . 1. System displays the error message and redirects the user to the previous screen.

9.2.4. Other functionalities

<u>Monitor device</u>

Actors	Logged User
Description	Logged user wants to monitor a device in a system (company's subgroup).
Preconditions	User has to be registered and logged in to the system.
Postconditions	-
Main Course	1. User selects the "Company" tab from the bottom tab bar. 1. User selects the system (company's subgroup) where the device is created (see EX1). 2. User is redirected to the selected system screen. 3. User clicks the "More" button of the device that he wants to monitor from the "Device's List" tab. 4. System redirects the user to the device's detail screen.
Alternative Course	-
Extensions	-

Exceptions	EX1. The user has a type of administrator or technicians and doesn’t have permission to access the system (company’s subgroup) . 1. System displays the error message and redirects the user to the previous screen.

<u>Receive Notification</u>

Actors	Logged User
Description	Logged user wants to receive the notification from the server.
Preconditions	User has to be registered and logged in to the system.
Postconditions	-
Main Course	1. User has the application running in the background. 2. System sends the notification to the user when some incidents happen.
Alternative Course	-
Extensions	-
Exceptions	-

9.3. Installation Manual

Prerequisites

1. Node: any 12.x version starting with v12.0.0 or greater

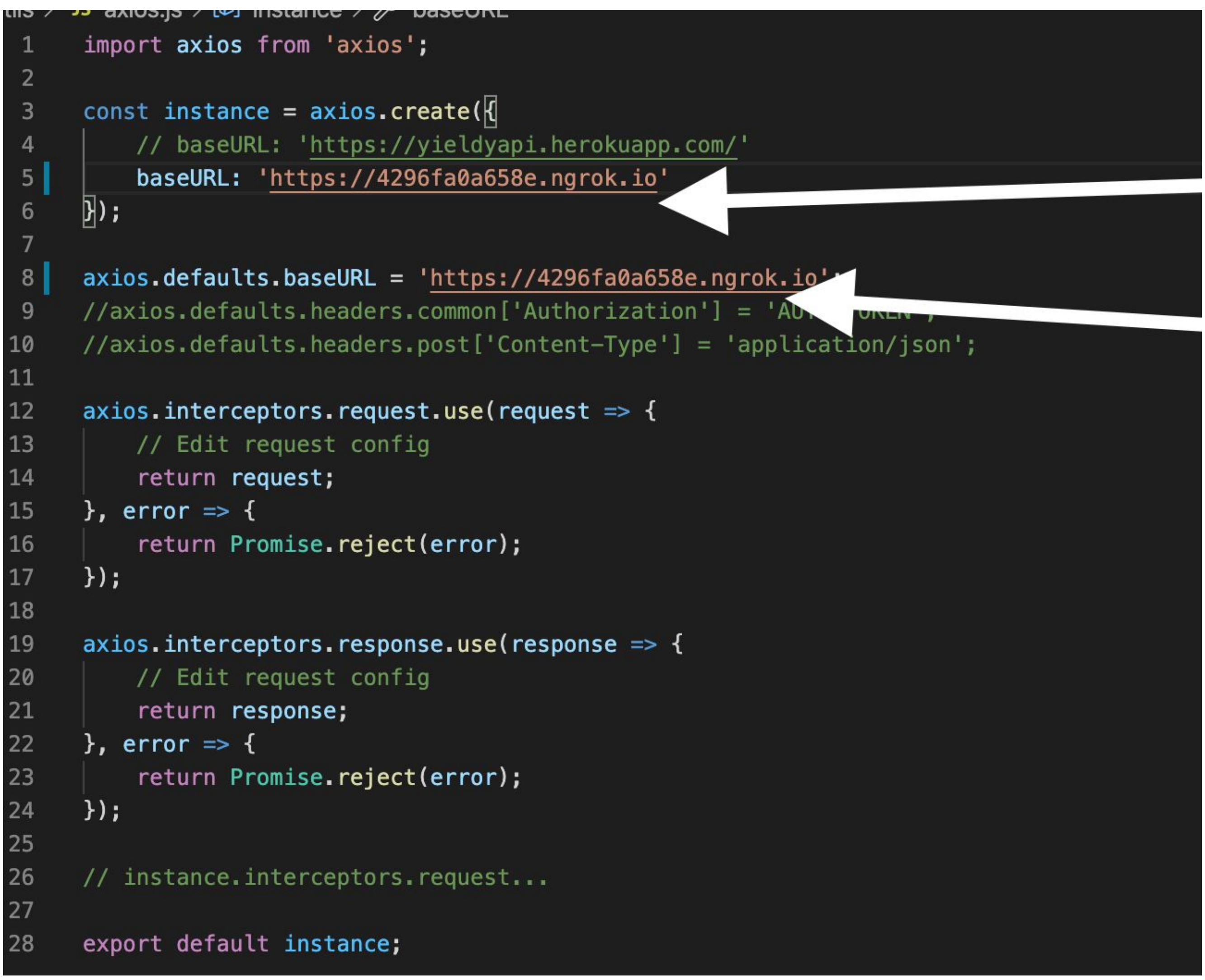
9.3.1. Backend/Web/Mobile

Open the terminal and go into the project root folder

1. `npm install` to install all the dependencies

2. `npm start` to run the application

The execution of the mobile version needs a real backend deployed on the internet, but for now the backend uses the elasticsearch server running on local network. Users can create a local tunnel of this port and then change the default URL variable in `<mobile project folder>/utils/axios.js` to make the mobile version work.



```javascript
import axios from 'axios';

const instance = axios.create({
    // baseURL: 'https://yieldyapi.herokuapp.com/'
    baseURL: 'https://4296fa0a658e.ngrok.io'
});

axios.defaults.baseURL = 'https://4296fa0a658e.ngrok.io';
//axios.defaults.headers.common['Authorization'] = 'AU[illegible]OKEN';
//axios.defaults.headers.post['Content-Type'] = 'application/json';

axios.interceptors.request.use(request => {
    // Edit request config
    return request;
}, error => {
    return Promise.reject(error);
});

axios.interceptors.response.use(response => {
    // Edit request config
    return response;
}, error => {
    return Promise.reject(error);
});

// instance.interceptors.request...

export default instance;
```

9.3.2. ELK

9.3.2.1. Elasticsearch

Open the terminal and go into the elasticsearch root folder

1. `./bin/elasticsearch` to run the elasticsearch server

9.3.2.2. Logstash

Open the terminal and go into the logstash root folder

1. `./bin/logstash -f config/<beats configuration file>` to run the elasticsearch server

Note that users have to check first the configuration file’s output to the current elasticsearch server’s address.

9.3.2.3. Beats

Open the terminal and go into the beat root folder, make sure the `beats.yml` is configured and update the fields project with the desired statuscode and the correct project name.

Mac OS:

1. `sudo chown root <beat’s name>.yml`
2. `sudo ./<beat’s name> -e` to run the beat

Linux OS:

1. `sudo chown root <beat’s name>.yml`
2. `sudo ./<beat’s name> -e` to run the beat

Windows:

1. `.\<beat’s name>.exe -e` to run the beat