

# ACKNOWLEDGEMENTS

First of all, I would like to express my gratitude to God for giving me the strength and the resistance to do this work.

My sincere thanks go to my academic supervisor **Dr. Hayet Tlijani** for offering her invaluable guidance, comments, and suggestions throughout the completion of the project.

Besides my advisor, I would like to thank my professional supervisor **Mr.Nabil Khlifi** for support, enthusiasm, and immense knowledge. This project would not have been completed without his enormous contribution. A special thanks also to **Mr.Nizar Khlifi** co-founder and CEO of Linksoft Consulting for helping to realize this project within his company.

Last but not least, I would like to thank the staff members of the ESIP Gafsa school for their generous attitude and friendly behavior. To all relatives, friends, and others who in one way or another shared their support, either morally, financially or physically.

*Thank You!*

# CONTENTS

<b>Acknowledgements</b>	<b>i</b>
<b>General Introduction</b>	<b>9</b>
<b>1 Project Presentation</b>	<b>11</b>
1.1 Overview: . . . . .	11
1.2 The host company . . . . .	11
1.2.1 Description: . . . . .	11
1.2.2 Activities: . . . . .	11
1.3 Project Context: . . . . .	12
1.3.1 The Current Situation: . . . . .	12
1.3.2 Critical examination of the existing: . . . . .	12
1.3.3 Problematic: . . . . .	12
1.3.4 Proposed solution: . . . . .	13
1.4 Methodology of work: . . . . .	14
1.4.1 Agile Methodology . . . . .	14
1.4.2 Scrum: . . . . .	14
1.5 Modeling Language: . . . . .	16
1.6 Work partitioning: . . . . .	16
1.7 Conclusion . . . . .	17
<b>2 Internet of Things Ecosystem</b>	<b>18</b>
2.1 Introduction . . . . .	18
2.1.1 Sprint Planning: . . . . .	18

---

2.2	Sprint Implementation: . . . . .	20
2.2.1	Data Storage . . . . .	21
2.2.2	Data Transmission . . . . .	23
2.2.3	Hosting server . . . . .	26
2.2.4	IoT ecosystem architecture . . . . .	26
2.3	Sprint Review . . . . .	28
2.3.1	Burndown chart: . . . . .	29
2.4	Conclusion . . . . .	30
<b>3</b>	<b>Machine Learning for Predictive Maintenance</b>	<b>31</b>
3.1	Introduction . . . . .	31
3.2	Scope . . . . .	31
3.3	Background . . . . .	32
3.3.1	Scheduled maintenance . . . . .	32
3.3.2	Time series data . . . . .	32
3.4	Sprint Planning . . . . .	33
3.5	Machine learning for predictive maintenance . . . . .	33
3.6	Artificial Neural Networks . . . . .	33
3.6.1	Recurrent Neural Networks . . . . .	34
3.7	Task 1: Data Generation . . . . .	35
3.8	Task 2: Model implementation . . . . .	36
3.8.1	Tools and Libraries . . . . .	36
3.9	Task 3: Model evaluation . . . . .	37
3.9.1	Evaluation Metrics . . . . .	37
3.9.2	Results . . . . .	38
3.10	Conclusion . . . . .	41
<b>4</b>	<b>Web Application</b>	<b>42</b>
4.1	Introduction . . . . .	42
4.2	Sprint Planning: . . . . .	42
4.3	Sprint Implementation: . . . . .	44
4.3.1	The planning calendar . . . . .	44
4.3.2	The 3D Maps . . . . .	47
4.3.3	Sensors view: . . . . .	52
4.3.4	Work Integration . . . . .	53
4.4	Sprint review . . . . .	54
4.4.1	The planning calendar review . . . . .	54
4.4.2	3D Map review . . . . .	55
4.4.3	Sensors View Review . . . . .	58
4.4.4	Work integration review . . . . .	59

---

4.5 Conclusion . . . . .	60
<b>Conclusion and Perspective</b>	<b>61</b>

# LIST OF FIGURES

1.1	Smart Application Architecture . . . . .	13
1.2	Scrum general process [1] . . . . .	14
2.1	Iot ecosystem global use case diagram . . . . .	20
2.2	Google cloud platform: project creation. . . . .	21
2.3	Google cloud platform: MySQL instance creation. . . . .	22
2.4	Mysql prompt through cloud shell. . . . .	22
2.5	Database instance connectivity. . . . .	23
2.6	MQTT Sequence Diagram. . . . .	25
2.7	Google cloud platform container registry. . . . .	26
2.8	Iot ecosystem class diagram. . . . .	27
2.9	IoT ecosystem deployment diagram. . . . .	27
2.10	Google monitoring: Read/write operations on Virtualplant-DB. . . . .	28
2.11	Google monitoring: Read/write operations on Virtualplant-DB. . . . .	29
2.12	Burn down chart plot: Work progress VS. Time. . . . .	30
3.1	LSTM Architecture . . . . .	34
3.2	Dataset Visualization . . . . .	35
3.3	Training and validation loss over epochs . . . . .	39
3.4	ROC curve plot. . . . .	40
4.1	Work Order: class diagram. . . . .	45
4.2	Web Application Development Architecture. . . . .	46
4.3	SAPUI General Architecture. . . . .	47
4.4	Factory physical architecture Hierarchy. . . . .	48

4.5	Plane creation class diagram. . . . .	50
4.6	Web Application : adding area to the scene. . . . .	51
4.7	Activity diagram Dataflow. . . . .	53
4.8	WEB Solution Component diagram. . . . .	54
4.9	Web Application: planning calendar and navigation page. . . . .	55
4.10	WEB application: Factory Hierarchical tree. . . . .	56
4.11	3d Map: adding area. . . . .	57
4.12	3d Model Equipment. . . . .	58
4.13	Web application: Sensors View. . . . .	59
4.14	Web application: Work Integration. . . . .	60

# LIST OF TABLES

1.1	Project Sprint Backlog . . . . .	15
2.1	Internet Of Things: Sprint Backlog. . . . .	19
2.2	EMQ X image Configuration. . . . .	25
3.1	Predictive Maintenance: Sprint Backlog . . . . .	33
3.2	Hyperparamters of our neural networks models . . . . .	37
3.3	Confusion Matrix description . . . . .	37
3.4	Confusion Matrix for LSTM Model . . . . .	39
3.5	Confusion Matrix for Logistic regression Model . . . . .	39
4.1	Web Application: sprint backlog. . . . .	43

# LIST OF ABBREVIATIONS

<b>AI</b>	Artificial Intelligence
<b>ANN</b>	Artificial Neural Networks
<b>DL</b>	Deep Learning
<b>IOT</b>	Internet Of Things
<b>LSTM</b>	Long Short Term Memory
<b>ML</b>	Machine Learning
<b>RNN</b>	Recurrent Neural Networks
<b>RUL</b>	remaining useful life
<b>SDM</b>	Software Development Methodology
<b>UML</b>	Unified Modeling Language
<b>VM</b>	Virtual Machine



# GENERAL INTRODUCTION

In the early days of the Industrial Revolution, machines were not too complex and that meant fewer breakdowns. As we entered into the second and third wave of the Industrial Revolution, with the assembly line and rapid automation through Programmable Logic Controllers (PLCs), the scenario had changed. There was less manual labor and more automation through complex machinery. To remain competitive, factories started measuring and closely tracking various performance metrics including production output, overall equipment effectiveness, personnel productivity, etc.

Maintenance which was seen as an activity to be undertaken only when there was a breakdown, became much more important. In an effort to predict impending failures and mitigate downtime in their manufacturing facilities, maintenance professionals have combined many techniques, like machine learning, internet of things and digital twins.

In fact, as a part of the internship program related to my graduation as a software engineer, the company Linksoft consulting assigned me for the study, design, and development of an application called "Virtual plant" that aims to present a predictive maintenance system. This dissertation is organized into four chapters:

In the first section, we give an overview of the project where we present the context of the project as well as the methodology of the work.

The second section covers our IoT ecosystem where we will stimulate the uses of smart sensors and microcontrollers to determine equipment changes and cloud computing solutions that provide a pathway for that data to travel to its destination.

The third section is about machine learning and its relation to predictive maintenance based solutions that enable the system to generate, analyze data and make sense of it,

then we will implement a predictive model based on a deep learning approach to predict machine failures.

In the last section, we will put all the work together in a web application based on 3D maps that visualize predictions as well as real-time data obtained from our equipment.

Finally, the aim of this project is to determine the importance of machine learning and preventive maintenance through prediction so we will close with a general conclusion and perspectives.

# CHAPTER

## 1

# PROJECT PRESENTATION

## 1.1 Overview:

In this chapter, we will give a brief overview of the project by first presenting the company in which the internship takes place in, then, we continue with the presentation of the subject explaining its different aspects as well as the methodology adopted for its realization. Later, we will expand with a product backlog where we will specify our project requirements.

## 1.2 The host company

### 1.2.1 Description:

Linksoft is a newborn small-sized company specialized in IT consulting and more specifically in the SAP ERP systems. Founded by an SAP expert (Nabil Khlifi) in "Plastic Omnium" and managed by a software engineer (Nizar Khlifi). This company is located in "Sidi bouzid-Tunisia".

### 1.2.2 Activities:

During the last years, Linksoft consulting performed some interesting tasks, their main goals was about trading and investing in several IT projects.

The ambition of the company is to create its own development team so that it can depend on them to realize their project instead of selling them. And actually, this project is a key player in this context.

### 1.3 Project Context:

As it was mentioned Linksoft consulting work in projects trading and for that reason, it was on the lookout for the future plans of Plastic Omnium which is represented in a predictive maintenance system.

In order to convince the company that the idea is worth investing in, Linksoft consulting has thought about a proof of concept where we will determine whether the software can be adapted in the real environment, what development technologies should be applied in, and whether the software is likely to be used by its intended users.

For this context, in the next sections, we will present the main business problem and the different techniques and solutions used to solve it.

#### 1.3.1 The Current Situation:

To help keep equipment up and running, the company relies on a preventive maintenance system that plans and schedules maintenance on assets before an actual breakdown happens.

Manufacturer's recommendations are used to help determine the type of inspections and maintenance needed and how often they should be performed.

#### 1.3.2 Critical examination of the existing:

Unfortunately, the preventive approach applied by the company isn't that efficient, it can indeed reduce unnecessary maintenance and inspections but, on the other hand, he could fail in predicting unplanned downtime as well as wasting the remaining useful life (RUL) of equipment parts by changing it before it gets defected.

In addition, the system's display wasn't practical in most of the work scenarios.

#### 1.3.3 Problematic:

According to Aberdeen's research [2], the average cost per hour of equipment downtime across all businesses is 260k dollars.

Besides cost, downtime can also have an impact on the productivity of employees, business revenue, and in some cases, it may even affect the reputation and the credibility of the company.

All these confusions lead to the necessity to improve and accelerate the proactive maintenance process.

### 1.3.4 Proposed solution:

In order to overcome the aforementioned issues, the company seek to consider using the predictive maintenance approach through it process.

For this reason, we have thought about a system that's able to predict failure within a given time window with the help of sensors and microcontrollers, while it keeps an efficient and simplified display.

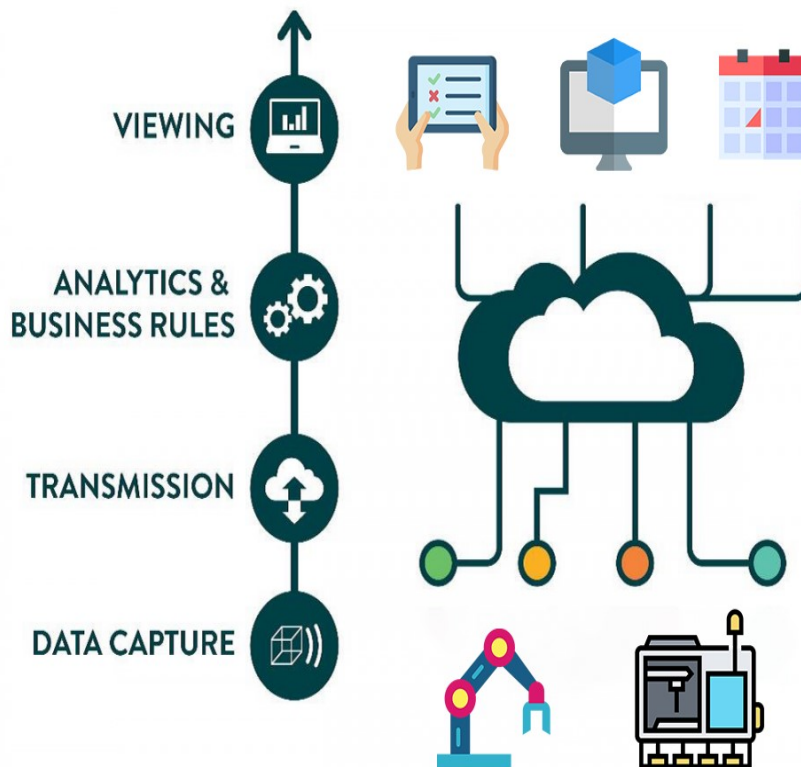


Figure 1.1: Smart Application Architecture

Figure 1.1 shows the overall application architecture which is based on the use of smart sensors to capture information, make sense of it, and identify any areas that need attention. This application requires sensors to measure things such as temperature, vibrations, and energy consumption. We can read these measurements through Raspberry Pi software and connect with front end applications that provide actionable insights. Then, we implemented an ML model using data collected over time, the goal is to find patterns that can help predict and ultimately prevent failures.

Finally, a web application that will enable users to be notified when predictions are made.

## 1.4 Methodology of work:

As we have mentioned earlier, our project is composed of a number of complicated individual parts, each part requires it's one technology and environment to be developed.

In order to reduce the complexity of the development process, we chose to use a Software Development Methodology[3] to plan, create, and control our project.

### 1.4.1 Agile Methodology

At an early stage of development, the requirements of the system were uncertain as well as the development priorities. Thus, to be capable of handling this problem, we have adopted the agile methodology[4], which is a set of engineering best practices that help to quickly respond to changing demand.

### 1.4.2 Scrum:

One of the most widely used agile processes is scrum, which is a lightweight process framework that follows an incremental and iterative development approach. The illustration in Figure 1.2 shows the basic Scrum steps to build a specific project.

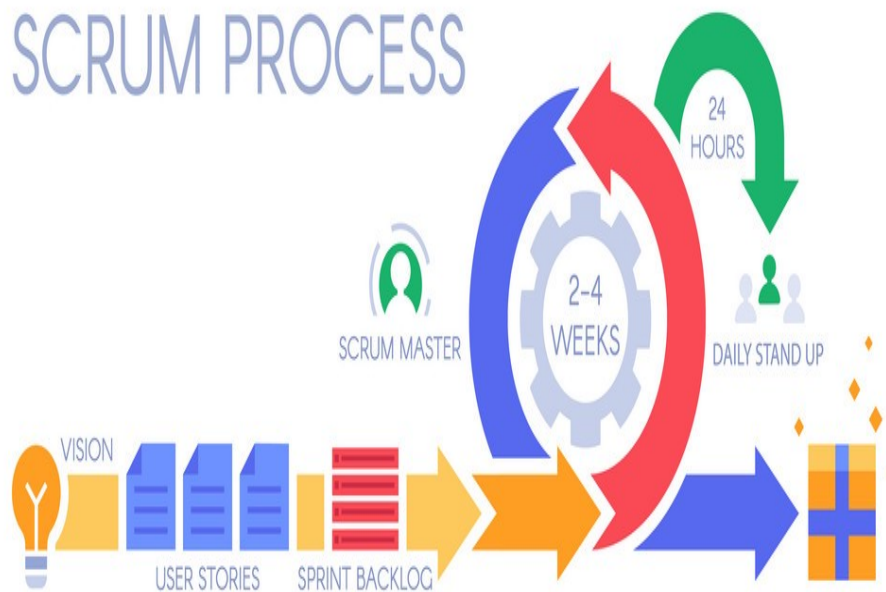


Figure 1.2: Scrum general process [1]

**Roles Definition:** Referring to the scrum approach we can define the different roles adopted in our project which are defined as the following:

- **Product Owner:** *M.Nabil Khlifi.* founder of "Linksoft consulting", understand the customer, explain and prioritize the work.
- **The development team:** *Bilel Khlifi.* design, implement, and verify the different parts of the system, and *Dr. Hayat Tlijani.* help in documenting the project.
- **Scrum Master:** This role was provided by *Mr. Nizar Khlifi* who was able to ensure good communication between the different members of the team. As well as Tracking the progress of the sprints.
- **Product Backlog:** The Product Backlog is a prioritized list of all product requirements, including new features, functions, technologies, enhancements, and bugs. The following Table 1.1 presents the overall project sprint backlog.

Table 1.1: Project Sprint Backlog

User Story	Story points	Priority	Sprint
As an IT technician, I want the system to collect data so that I can use them in the upcoming tasks.	8	High	Iot Sprint
As an IT technician, I want the system to be scalable so that it can handle all the equipment in the factory.	13	Meduim	Iot Sprint
As an IT technician, I want the system to be easy to deploy so that I can reduce the time of setup and configuration.	5	Low	Iot Sprint
As a manager, I want to build a predictive model So that I can predict future machine failures.	20	High	AI Sprint
As a maintenance manager, I want to consult the scheduled maintenance tasks, So that I can easily assign and describe tasks.	8	Meduim	WEB Sprint
As an IT technician, I want to represent the entire factory, So that the maintenance technician could make updates and amends, without tampering with the physical asset itself.	20	High	WEB Sprint
As a maintenance technician, I want to visualize the equipment status in real-time, So that I can perform service tasks at multiple sites simultaneously.	5	Meduim	WEB Sprint
As a manager, I want the application to be compatible with the existing system, So that I can easily switch between them.	2	Low	WEB Sprint

## 1.5 Modeling Language:

To help visualize our system architecture and understand their different components, we chose to use the Unified Modeling Language (UML) [5].

During this phase, we will use five different diagrams:

- **Use case diagrams:** To capture the different requirements of our system.
- **Class diagrams:** To Analyse and design, a static view of our system.
- **Sequence diagram:** To describe interactions among the different elements of the system.
- **Activity diagram:** To captures the dynamic behavior of the system by showing message flow from one activity to another.
- **Deployment diagram:** To describe the physical aspects of an object-oriented system.

## 1.6 Work partitioning:

Unlike classical models, scrum has its proper approach of development which in turn can affect how documentation could be performed. For this reason, we have added this part to explain the steps we took to write this document.

Since each part in this project has its one business rules, architecture, and development environment we have divided the work into a time-boxed repeatable work cycle known in the scrum as sprints. Each sprint is composed of :

- **Sprint planning:** Where we define the sprint backlog using a set of product backlog items.
- **Sprint implementation:** Where we design the system architecture using different UML diagrams and present the technologies and tools used to develop our system.
- **Sprint review:** Where we represent what was done during the sprint depending on screenshots and charts.



## 1.7 Conclusion

After defining our requirements and the methodology of our work as well as the project context, we will investigate in the next chapter our first sprint where we will introduce the ecosystem used to collect and transmit equipment data.

## CHAPTER

## 2

# INTERNET OF THINGS ECOSYSTEM

## 2.1 Introduction

Having multiple different sensors that monitor different metrics can be a key to have a better understanding of our processes and preventing early failures. Thus, in this chapter, we present how we have applied sensors and how we have connected them to different parts of our system.

First, we start with a sprint planning in which we select some related product backlog items, then we identify the necessary tasks to complete each user story. After that, we conducted a sprint implementation where we talk about different technologies and environments used in order to perform those tasks. several UML[5] diagrams are introduced together with the implementation to describe the structure and behavior of the system.

Finally, a sprint review where we will inspect our increment and discuss what we have accomplished.

### 2.1.1 Sprint Planning:

In this section, we will define what can be delivered in the upcoming sprint and how that work will be achieved through.

#### 1. Sprint backlog:

Table 2.1: Internet Of Things: Sprint Backlog.

Id_US	User Story	Task_ID	Task	Estimation/day
1	As an IT technician, I want the system to collect data so that I can use them in the upcoming tasks..	1.1	Database creation.	2
		1.2	Database connection.	2
		1.3	Data insertion.	3
2	As an IT technician, I want the system to be scalable so that it can handle all the equipment in the factory.	2.1	Protocol observation.	4
		2.2	Client adoption.	3
		2.3	Broker selection.	3
3	As an IT technician, I want the system to be easy to deploy so that I can reduce the time of setup and configuration.	3.1	VM instance creation.	1
		3.2	Container installation.	3

## 2. Use Case diagram:

To acquire a better understanding of the expected features in this sprint, we have created a use case diagram that describes the functional requirements of the system from the end user's perspective. The use case diagram is shown in Figure 2.1, our diagram is based on 2 actors, the "System equipment" which is serve to generate the necessary data, while the "IT-technician" is then able to collect or handle the factory equipment data and also he want to reduce the deployment time.

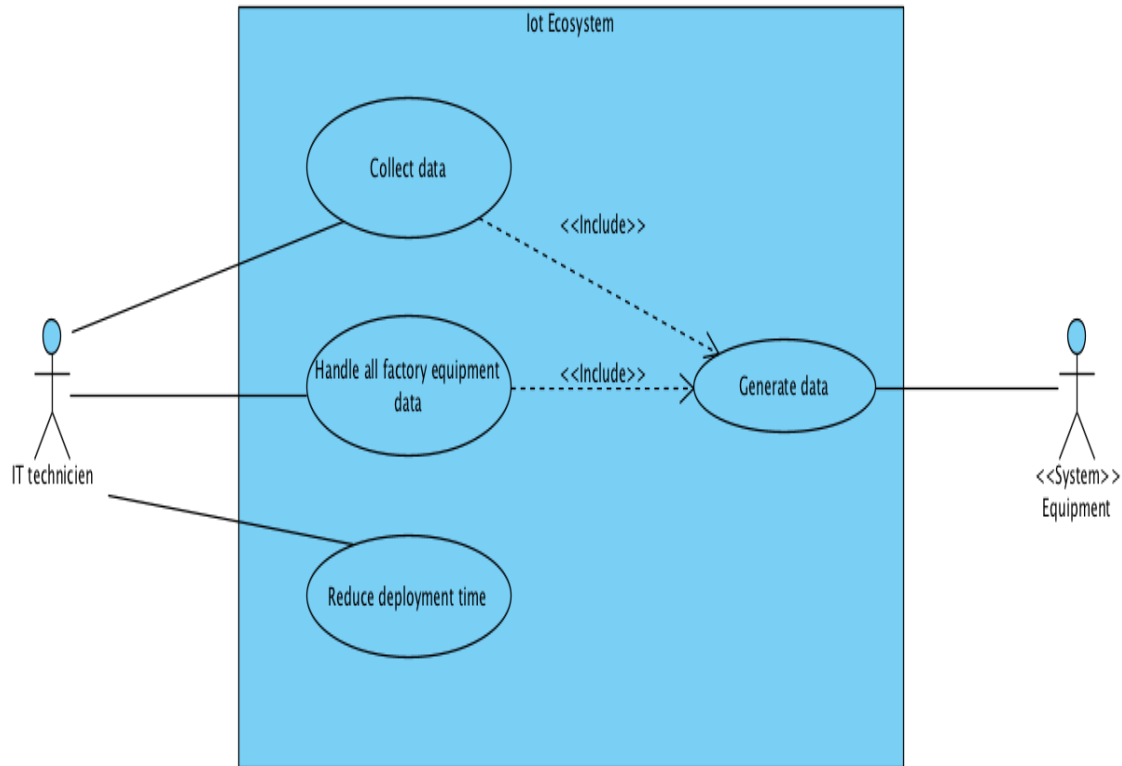


Figure 2.1: Iot ecosystem global use case diagram

## 2.2 Sprint Implementation:

As we have stated earlier, smart sensors will be used in this application to gather data, then several analytic will be performed to find hidden patterns.

However, gathering this data requires a considerable amount of time as well as accessing real equipment and environments. As those conditions cannot be met, we have implemented an algorithm using Thonny IDE and flask based on the python programming language to generate our synthetic data.

Since using real hardware will make no sense in this scenario, we have only used Raspbian OS, the official Raspberry Pi operating system.

Based on Debian bluster, the system was fully compatible with being run on a real

Pi, we can write and test code as well as installing packages and connecting to various devices.

### 2.2.1 Data Storage

We're in a digital economy where data is more valuable than ever, processing and collecting data become an essential task for almost any business. For this reason, storing data was a trivial task for our customers.

In the next steps, we will represent the different tasks that we have performed including the creation and connection of our database, as well as storing the collected data.

A- **Database creation:** To be capable of accessing data from any device in any location, we have chosen to adopt a cloud storage solution. Therefore, we preferred to use cloud SQL, the fully managed relational database service from Google.

Google Cloud Platform offers hundreds of cloud-based features and tools, but before we can access a single one, we have to create a project. We started by creating our main project which is untitled as “Virtual plant” as can be seen in Figure 2.2.

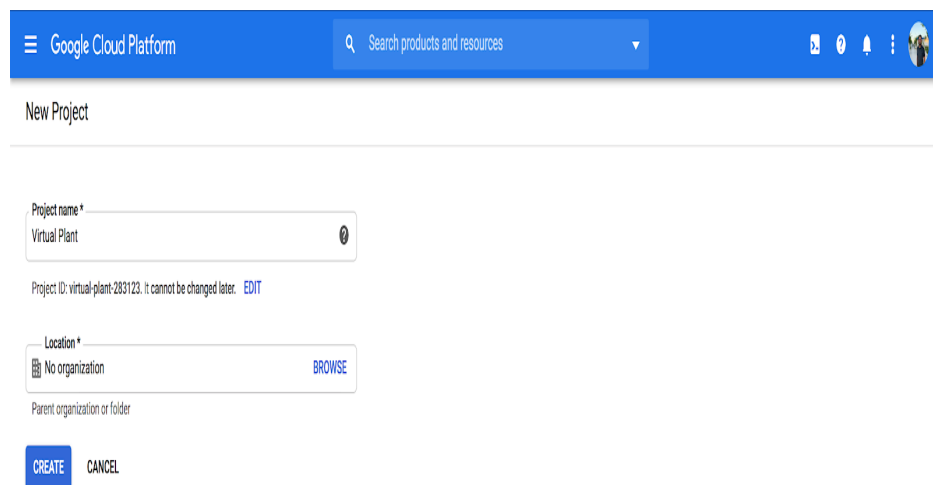


Figure 2.2: Google cloud platform: project creation.

Cloud SQL offers a variety of instance types, in our case we have favored MySQL instances that offer high performance and availability. Figure 2.3 shows the interface in which we have added the instance and set the root password.

Google Cloud Platform Virtual Plant

SQL Create a MySQL instance

Instance info

Instance ID  
Choice is permanent. Use lowercase letters, numbers, and hyphens. Start with a letter.  
virtualplant-db

Root password  
Set a password for the root user. [Learn more](#)  
[password field] Generate

☐ No password

Location  
For better performance, keep your data close to the services that need it.

Region  
Choice is permanent  
us-central1 (Iowa)

Zone  
Can be changed at any time  
Any

Database version  
MySQL 5.7

[Show configuration options](#)

Create Cancel

Figure 2.3: Google cloud platform: MySQL instance creation.

After that, through Cloud Shell Terminal and using our user and password (see Figure 2.4), we were able to grant access to a fully functional MySQL prompt.

```

Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to our-mediator-279716.
Use "gcloud config set project [PROJECT ID]" to change to a different project.
khalifi bilel@cloudshell:~ (our-mediator-279716)$ gcloud sql connect virtualplant-db --user=root --quiet
Whitelisting your IP for incoming connection for 5 minutes...done.
Connecting to database with SQL user [root].Enter password:

```

Figure 2.4: Mysql prompt through cloud shell.

Later using regular SQL statements, and through this prompt, we will be able to create our database and table.

- B- **Database connection:** To be able to interact with the database using external applications, it was necessary first to configure access to our Cloud SQL instance, by adding our raspbian os's public IP address as an authorized network (see Figure 2.5).

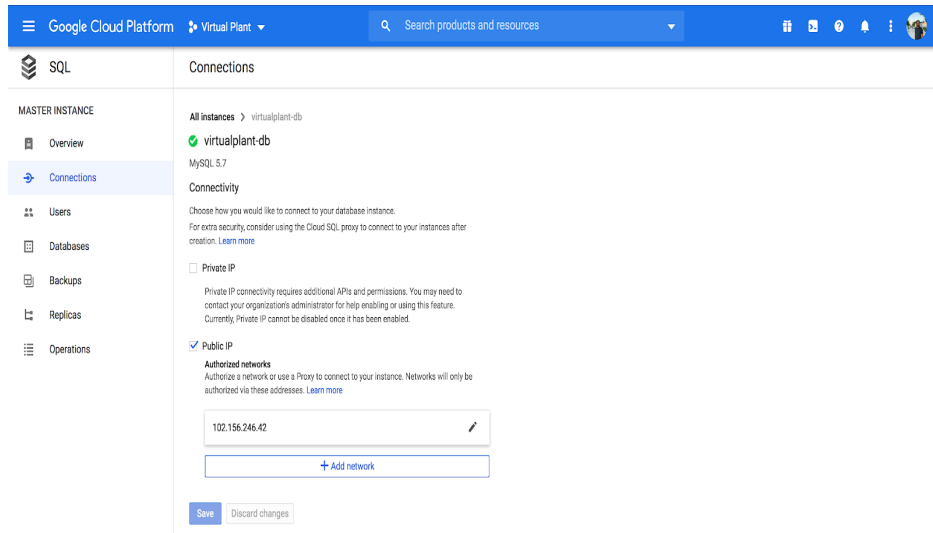


Figure 2.5: Database instance connectivity.

Next, to assure the connectivity between our database and our python application we have installed MySQL connector for python.

**C- Data storage:** As we have mentioned at the beginning of this chapter, we have developed an application that is capable of generating sensors data. In order to save those data to the newly created database, we have imported the connector module first, then we fed our application with the required information to connect our instance. This information includes:

- Host (MySQL instance public IP address).
- User
- Password
- Database name

Later, we have associated our generated data to a cursor object that in turn persists data to the cloud.

## 2.2.2 Data Transmission

Besides storing the data, connecting data to our final application was a trivial task to accomplish our prediction mission.

In fact, many existing solutions provide these functionalities like Leonardo from SAP and Google Cloud IoT.

Unfortunately, those platforms are too expensive that may conflict with the needs of our customers. In addition to cost-effectiveness, scalability, real-time response, and low server load were mandatory requirements for our clients.

For that reason, we have implemented a solution in which we will try to meet our customer expectations as well as realizing our proof of concept. In the next stage, we will walk through the different tasks that need to be done to achieve our goal.

### A- **Protocol observation:**

When we talk about transmission and communication of data, the first questions that come to head is how network devices should format, transmit, and process information. And in response to this, we have chosen MQTT[6], one of the most commonly used protocols in IoT projects.

Practically speaking, MQTT was built to be a low-overhead protocol that strongly considered bandwidth and CPU limitations. It was designed with the ability to run in an embedded environment where it would reliably and effectively provide an avenue for communication.

As from a fundamental view, MQTT is a publish/subscribe protocol, it allows clients to connect as a publisher, subscriber, or both. You connect to a broker that handles all the message passing.

B- **Client adoption:** To follow MQTT protocol rules, it was required in the first-hand to obtain a client that will allow us to send our data. We chose for this task the Eclipse Paho project that provides an open-source client implementation for MQTT which suits our requirement.

Since Paho affords a client library for python, we have continued with extending our python application to be able to send data to a remote broker together with the capability of data storage and generation.

We create a topic for each metric that allows us to publish the obtained data. To do so, we connected to the broker in which the client will use the required broker IP address and port.

### C- **Broker selection:**

In the Second-hand and always with MQTT protocol rules, it was also required to set up a broker that enables us to publish our data. For this context, we chose EMQX, a massively scalable and highly available open-source MQTT Message Broker.

Our selection was made based on different requirements:

- Support MQTT over WebSockets, which ensures communication with our web application.
- Scalable and portable, which agrees with our customer needs.
- Support containerization, which reduces deployment complexities.



The following Table 2.2 describes the configuration used in the broker.

Table 2.2: EMQ X image Configuration.

Options	Default	Mapped	Description
EMQX_NAME	container name	none	Emqx node short name
EMQX_WAIT_TIME	5	none	Wait time in sec before timeout.
EMQX_LISTENER_TCP_EXTERNAL	1883	listener.tcp.external	MQTT TCP port
EMQX_LISTENER_WS_EXTERNAL	8083	listener.ws.external	HTTP and WebSocket port
EMQX_MQTT_MAX_PACKET_SIZE	64KB	mqtt.max_packet_size	Max Packet Size Allowed

The sequence diagram presented in Figure 4.6 explains how our client (IoT application) publishes a topic to the broker, then our second client subscribes to these topics to get data.

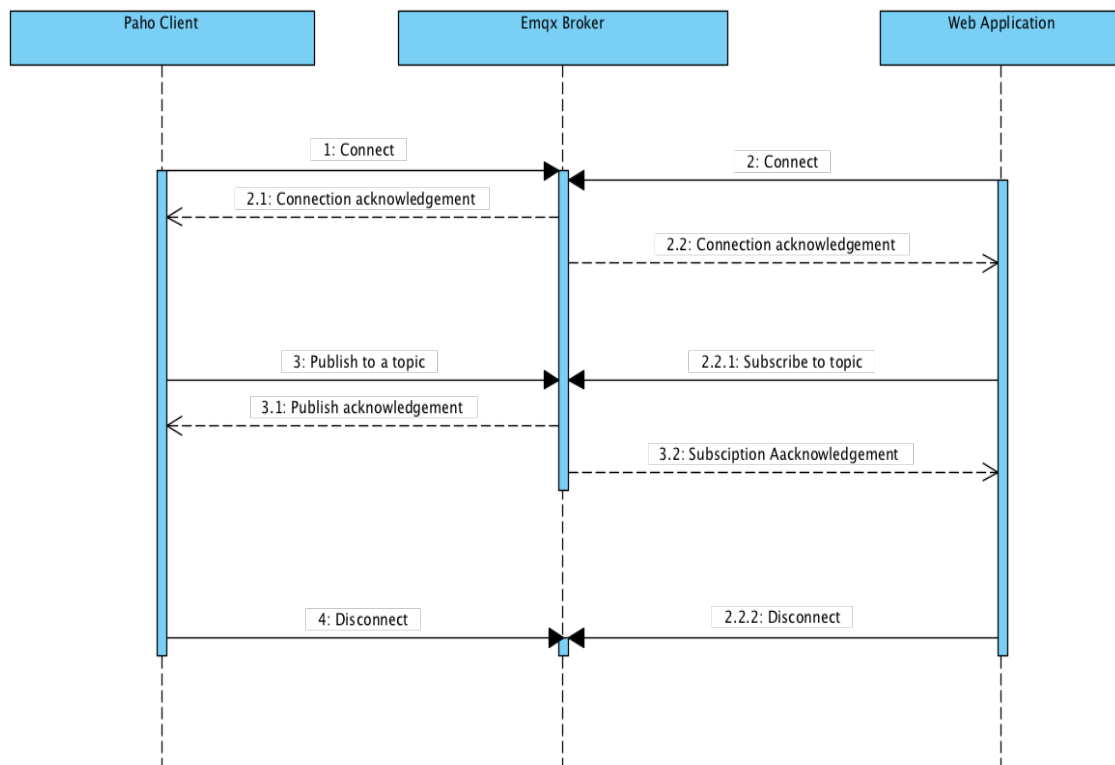


Figure 2.6: MQTT Sequence Diagram.

After selecting the client and the broker, we will introduce in the next section, the deployment of the broker considering the compatibility and the portability point.

### 2.2.3 Hosting server

Using a broker involves installing the software on an operating system, therefore, we have operated a compute engine on Google Cloud Platform that allows us to create a Virtual Machine (VM) instance.

Nevertheless, installing directly the broker on the VM instance will require a lot of configuration, libraries, and dependencies, for that reason, we have considered using a containerized application to solve this issue.

A container is essentially a fully packaged and portable computing environment. Everything an application needs to run – its binaries, libraries, configuration files, and dependencies – is encapsulated and isolated in a bundle called container image.

Compute Engine instances support a declarative method for launching applications using containers. When creating the VM instance, we have provided the EMQX docker image using the container registry (see Figure 2.7).

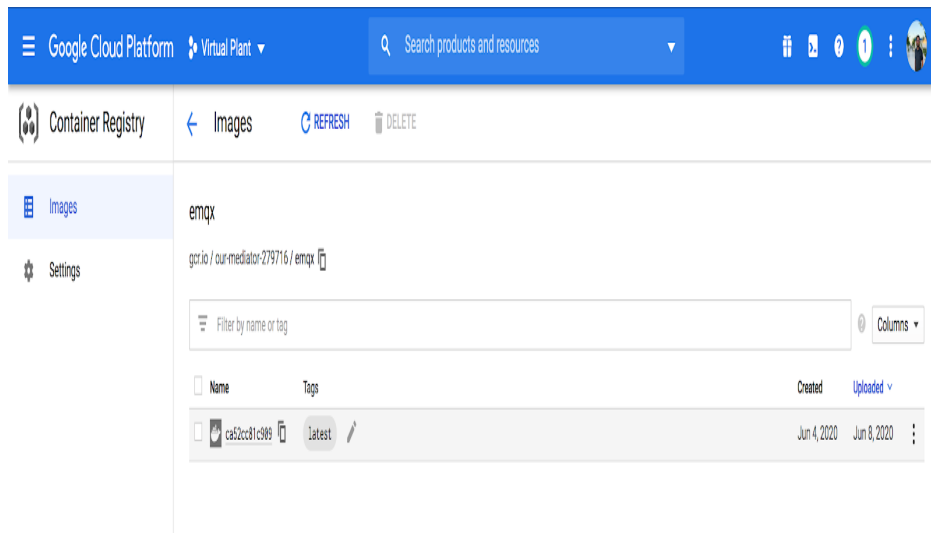


Figure 2.7: Google cloud platform container registry.

After that, the Compute Engine will supply an up-to-date Container-Optimized OS image with Docker installed and launch the container when the VM starts up.

### 2.2.4 IoT ecosystem architecture

The following class diagram presented in Figure 2.8 can briefly describe better the static view of our application.

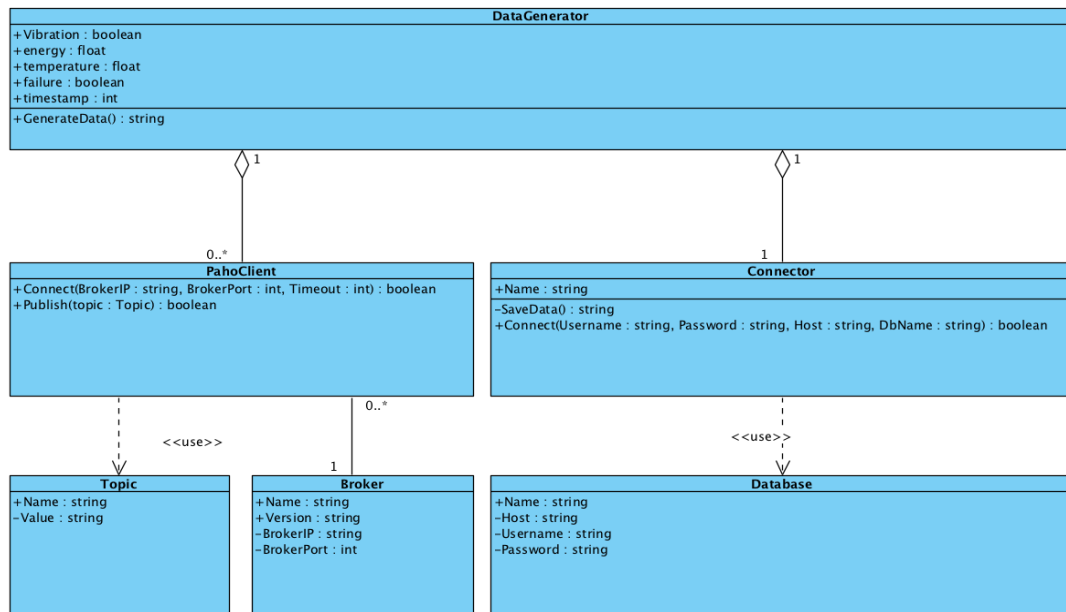


Figure 2.8: Iot ecosystem class diagram.

Furthermore, to better visualize our system we have used a deployment diagram (see Figure 2.9) that shows the execution architecture, including nodes such as hardware or software execution environments, and the middle-ware connecting them.

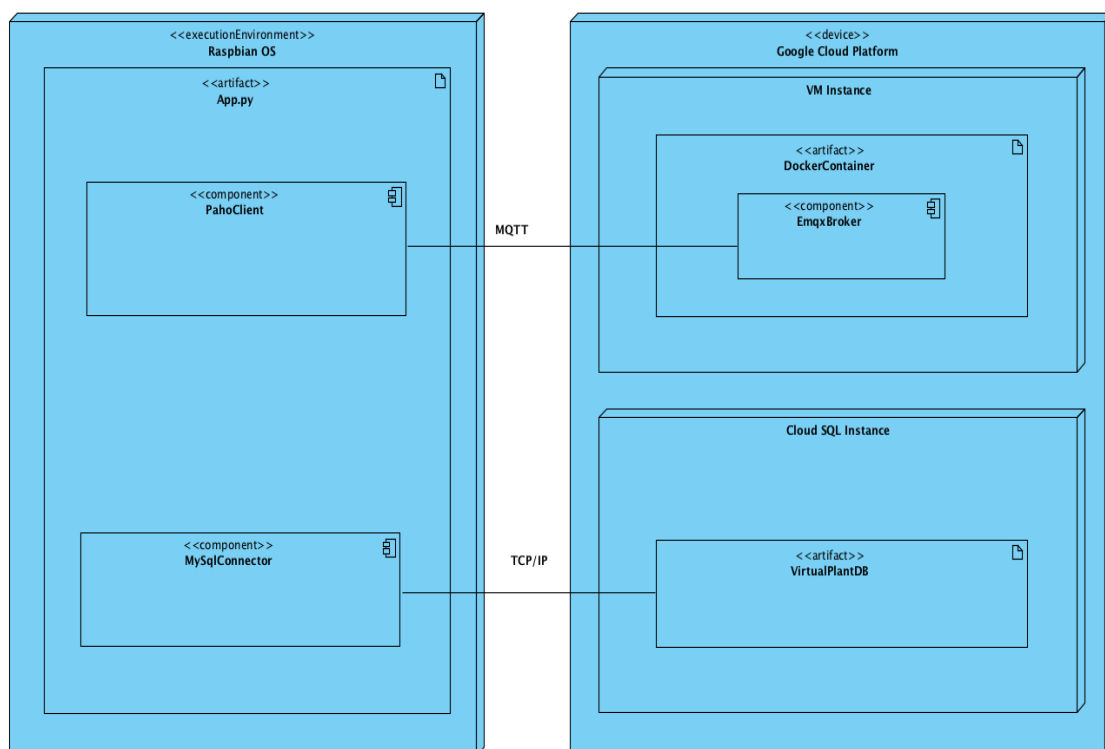


Figure 2.9: IoT ecosystem deployment diagram.

## 2.3 Sprint Review

*"I think it's very important to have a feedback loop, where you're constantly thinking about what you've done and how you could be doing it better". [Elon mask]*

As we have seen, the goal of this sprint was to satisfy three different user stories among 8 tasks and in a period of three weeks.

Primary, it was required to create a database to store our data. The Figure 2.10 shows how our database have received data and saved it for later tasks.

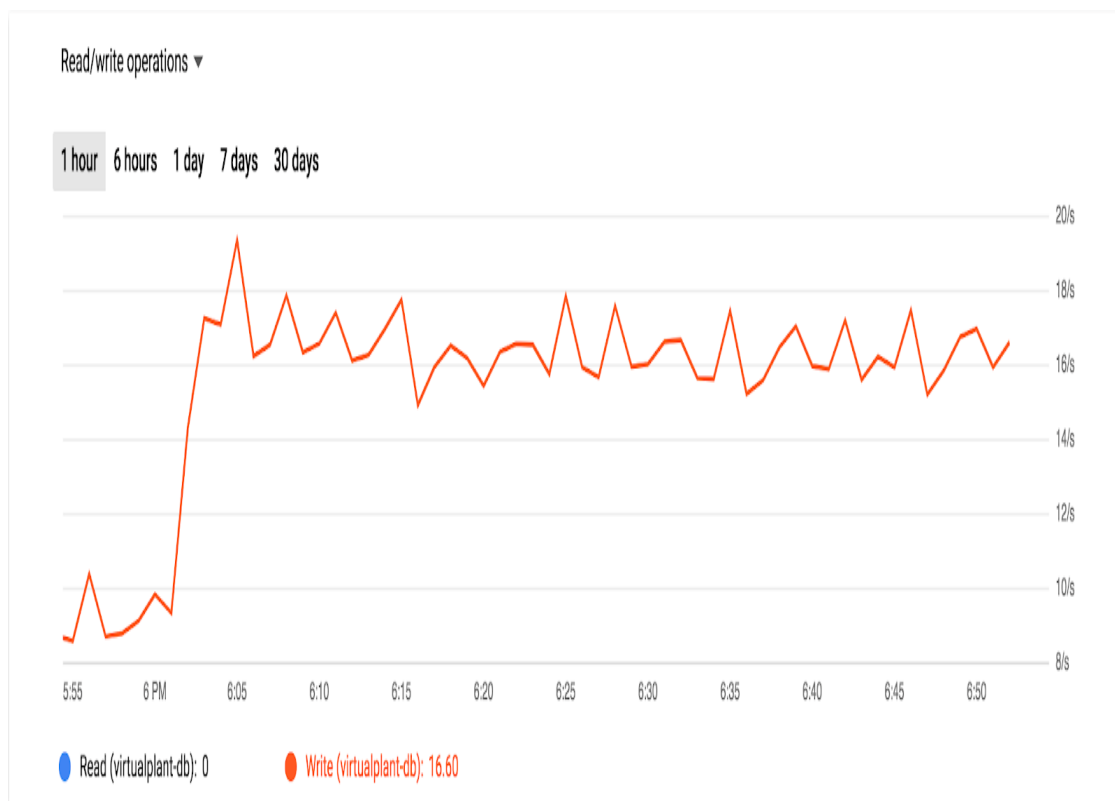


Figure 2.10: Google monitoring: Read/write operations on Virtualplant-DB.

Secondly, it was required to transmit data over our systems using MQTT and containers. The figure 2.11 below shows how the data transmitted over our VM instance.



Figure 2.11: Google monitoring: Read/write operations on Virtualplant-DB.

Later in the production environment, the VM instance will be abandoned and replaced by the server of the company, where we will move our container and import our database table.

### 2.3.1 Burndown chart:

The following Figure 2.12 shows our work advancement against time. At the beginning (first week) we didn't find any struggle in the creation and connection of our database, same for the client and broker selection phase. Since we are adopting the MQTT protocol for a complex task, it needs a feasibility study which in turn required a considerable amount of time more than it was estimated earlier.

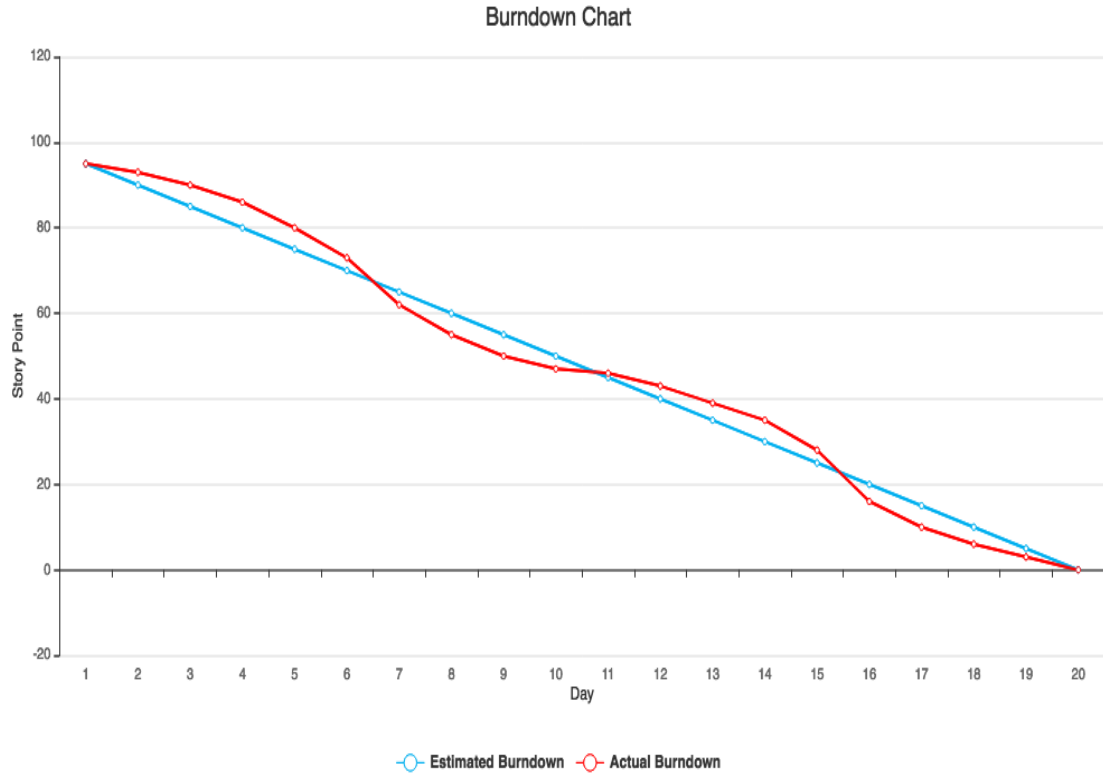


Figure 2.12: Burn down chart plot: Work progress VS. Time.

We can interpret from the obtained results that our conducted sprint succeeded to cover all the necessary parts in order to achieve our POC. In addition, it offers reusable functionalities that can be adopted in a real-world project.

## 2.4 Conclusion

In this chapter, we have presented the foundation of our POC by introducing our IoT ecosystem and its different parts including remotes dashboards, networks, gateways, and data storage.

In the next chapter, we will dive more into details and precede with our analytic ecosystem and explain how we have implemented machine learning techniques to uncover the hidden patterns.

## CHAPTER

## 3

# MACHINE LEARNING FOR PREDICTIVE MAINTENANCE

### 3.1 Introduction

In the last few years, the industry starts investigating the potential of data that has been stored for a long time but hardly exploited. Advances in data science (or machine learning) turned very sophisticated problems into feasible tasks that can bring useful added value to almost every industrial domain. Nowadays, learning models from data in order to perform tasks such as predictions or information extraction has become very popular and therefore, machine learning algorithms are becoming indispensable tools in decision making. In this context, this thesis summarizes our efforts to exploit a deep learning model for problems arising in the plastic industry and more precisely, in predictive maintenance.

### 3.2 Scope

Plastic Omnium is one of the most popular industries that serves billions of clients each year. A key parameter for the efficiency of this industry is the availability of the plastic machines, which is the percentage of time a plastic machine is in good condition and capable to fabricate the necessary equipment . The availability is usually reduced by

unexpected failures, whose occurrences render the plastic machine unable to perform an operation and therefore, sometimes leading to severe delays causing financial damages to industry and reduce the customers satisfaction. It is the goal of predictive maintenance to prevent such situations by making predictions about potential failures that could affect the normal plastic machines operation.

### 3.3 Background

#### 3.3.1 Scheduled maintenance

Scheduled maintenance is the traditional way of maintaining equipment and consists of a set of maintenance procedures defined at plastic machines design time and that must be planned in advance and performed periodically. However, whenever an unexpected failure occurs between two scheduled maintenance slots, the equipment becomes unavailable until the necessary maintenance actions are performed. These unexpected failures can be a costly burden to the equipment owners because during the downtime they may not be able to provide the expected services to their clients. On the other hand, the goal of predictive maintenance is to prevent such unexpected equipment failures by continuously observing the status of the equipment and raising alerts well in advance. The time between the alert and the failure can be used by the experts to plan and perform the maintenance and to avoid any operational disturbance. In the case of aviation, where the status of the equipment can be reflected by data related to the operation of the plastic machine, the challenge is to analyze such data, to translate high level predictive maintenance objectives into data science tasks (or data mining routines) and to achieve the best possible results.

Given the increasing collections of available data generated by monitoring processes (e.g. sensors and event logs), the goal is to predict upcoming critical events or system failures.

#### 3.3.2 Time series data

Time series is one of the most common data types in many real world applications (e.g., finance, meteorology, medicine, sensor networks) that constantly draws the attention of the research community, and has been very well studied over the past decades. The research and development is being performed by employing modern distributed computing frameworks, suitable for the big-data demands of the project due to the huge amount of information generated by various companies components.



## 3.4 Sprint Planning

Table 3.1: Predictive Maintenance: Sprint Backlog

Id_US	User Story	Task_ID	Task	Estimation/day
1	As a manager, I want to build a predictive model, So that I can predict future machine failures.	1.1	Data Generation.	10
		1.2	Model Implementation.	15
		1.3	Model Evaluation.	5

## 3.5 Machine learning for predictive maintenance

In the recent years, the progress in the field of Artificial Intelligence (AI) with the emerging of the Machine Learning (ML) approach has led to the improvement of the state of the art in many areas of computer vision, reinforcement learning, Robotics, and been recently adopted in the predictive maintenance domain. This mentioned approach offers promising solutions and an ability to deal with problems that are largely difficult to solve by using traditional Machine Learning methods.

Predictive maintenance and machine learning have developed a very strong connection. However, it is not always easy or straightforward to perform effective predictive maintenance for several reasons (Lack of annotated data, Huge amounts of data, etc.). Therefore using modern techniques based on artificial neural networks could be feasible for solving these problems.

## 3.6 Artificial Neural Networks

ANN is a computational tool inspired by the network of neurons in biological nervous system. its a classifier modeled after how the human brain works, which is different from how one usually writes computer code.

ANN presents a network consisting of arrays of artificial neurons linked together with different weights of connection. The states of the neurons as well as the weights of connections among them evolve according to certain learning rules.

Practically speaking, neural networks are nonlinear statistical modeling tools which can be used to find the relationship between input and output or to find patterns in a vast database.

ANN has been adopted in various areas of statistical model development, adaptive control system, pattern recognition, data mining, and decision making under uncertainty. There are several types of network architectures that can be used depending on the type of problem and the purpose intended. Different design patterns and architectures improved to deal with specific case problems.

Generally, it exists a different variety of neural network architectures such as Recurrent neural networks which become an active research area in the DL approach.

### 3.6.1 Recurrent Neural Networks

Recurrent neural networks (RNN) can be defined a sequential model of neural networks, which have the property of reusing information already given. One of their main assumptions is that the current information has a dependency on previous data. The ultimate goal of this neural architectures is recognizing the data's sequential characteristics and applying patterns to predict the next likely scenario. RNN's has been widely used in speech recognition and NLP. One pf the popular algorithm based on RNN's architectures is the Long Short Term Memory (LSTM).

- **LSTM:** this types of RNN offers a special architecture allows them to represent long term dependencies. Moreover, they are specifically designed to remember information patterns and information over long periods of time. Therefore, this model is useful in our case for the prediction the machine failure trained within an important period of time.

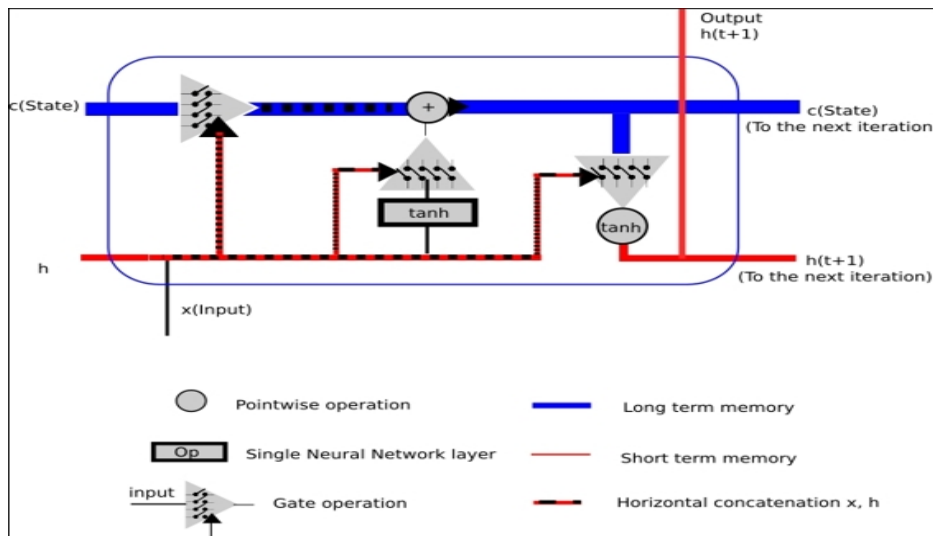


Figure 3.1: LSTM Architecture

Figure 3.1 shows the general architecture of LSTM algorithm, When the data flows inside the LSTM cell, the first step consists in deciding what information is necessary to keep and what is not. The second step involves the input gate and allows the LSTM to establish which of the new information to store timestep, and the values coming from the previous one. Finally, the last step allows the generation of the LSTM output predictions.

### 3.7 Task 1: Data Generation

Since we don't yet have an access to a flexible data set related to the omnium company, our experiment was essentially based on synthetic dataset, which is a repository of data that is generated programmatically. So, it is not collected by any real-life survey or experiment.

Our initial dataset was created using a script based on python language and SQLite[7], we designed our module to automatically collect 500k instances and their corresponding metadata. The program focused on collecting the necessary information from the corresponding sensors. Since we are collecting data instances related to machine failure. Our choice was essentially based collecting data over a period of 1 year. Figure 3.2 shows an overview of our obtained dataset.

MachineID	Energy	Vibration	Temperature	Timestamp	Failure
1	15.9	1	53.85	1.546E+09	0
1	17.3	1	50	1.546E+09	0
1	26.79	1	57.31	1.546E+09	1

Figure 3.2: Dataset Visualization

The metadata of our data sets are :

- **ID:** Primary key.
- **Energy:** The energy consumption for a machine.
- **Vibration:** The vibration condition.
- **Temperature:** The degree of temperature of a machine.
- **Timestamp:** Reference particular moments in time related to machine failure.
- **Failure:** The condition of a machine which takes 1 as failure and 0 as working.

For the training of our deep learning model and due to the GPU limitation and computation resources, we limited the number of instances to 100k, while we used the strategy of train-test-validation of 80% train, 10% test and 10% validation.

## 3.8 Task 2: Model implementation

### 3.8.1 Tools and Libraries

Following [8] who demonstrated that using recurrent neural network LSTM in predictive maintenance can outperform other conducted approaches. In first hand, we aimed to use this model as a main algorithms, while in second hand, we performed an analysis study to test our approach using other machine learning algorithms such as: logistic regression[9], decision tree, random forest.

In order to obtain the results presented in this thesis we performed our experiments using available tools and libraries. The ones that were during the implementation of our predictive model are:

- Google collab. Since training our model is expensive in term of resources, we ended up by using the Google Colaboratory tool, which is a free research tool with a Tesla K80 GPU and 12G RAM.
- Keras.[10] Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow.
- Matplotlib.[11] A python plotting library that was used to generate all the figures that present analysis results.
- Scikit-Learn.[12] A python library for Machine Learning which includes a plethora of both state of the art and recent algorithms.

To accomplish this, we goes through different steps to implement our predictive model.

1. **Data ingestion:** We used data ingestion task on our datasets which presents a process by which data is moved from one or more sources to a destination where it can be stored and further analyzed.
2. **Data preprocessing:** Since the time series data is of varying length, we cannot directly build a model on this dataset. Moreover, data need to be transform into an understandable format before feeding it to our model. Thus, we Pad the shorter sequences with zeros to make the length of all the series equal, while we used a scaler to perform the data transformation.
3. **Building model:** It is important to adjust the most suitable hyper-parameters when training neural networks to increase their performances and avoid some problems like over-fitting. We adapted our LSTM model for a binary classification. To do so, we go through a model selection process in order to find the best model configuration. As for our main LSTM classifier, we adopted the hyper-parameters

of 100 hidden layers-based model, with a dropout a dropout equal to 0.1. we used Adam as the optimization algorithm. The batch size is fixed to 200 due to GPU limitation, other Hyperparamters are shown in Figure 3.2. As for testing strategy (all models), We split our dataset into 80% for training and 20% for testing and validation respectively. During each epoch, our LSTM was trained on the training set and evaluated on the validation set.

Table 3.2: Hyperparamters of our neural networks models

Hyperparamaters	Values
Dropout	0.2
Window size	5
Nb. of epochs	100
Batch size	200
Activation function	Sigmoid
Optimizer	Adam

Since we are training our model to perform a binary classification, at the end of the architecture is necessary to use a dense layer to generate as output, only one number which in this context is represented by prediction given in input a sequence of sensors measurements. In addition, we used a Sigmoid activation function to generate the desired prediction. The final outputs can be seen as following.

*Example. "The machine will fail at (1.546E+09) with (21.364) failure"*

Where "1.546E+09" presents and the timestamp and "21.364" the percentage of failure.

## 3.9 Task 3: Model evaluation

### 3.9.1 Evaluation Metrics

for our binary classifier model, we first used the confusion matrix that is able to evaluate the optimal solution during the classification training [13] which is shown in Table 3.3.

Table 3.3: Confusion Matrix description

	Predicted Positive	Predicted Negative
Actual Positive	TP	FN
Actual Negative	FP	TN

We details each term below:

- **TP**: the number of true positives that represents the correctly predicted positive values.

- **FN**: the number of false negatives which represents the incorrectly predicted labels that are actually positive and predicted as negative.
- **FP**: the number of false positives which refers to the values that the model identifies as positive and are actually negative.
- **TN**: the number of true negatives that refers to the correctly predicted negative values.

So, we use this confusion matrix to compute our evaluation metrics:

- **Precision**: the number of correctly predicted positive labels divided by the total number of actual labels classified by the system as positive [14].

$$Precision = \frac{TP}{TP + FP} \quad (3.1)$$

- **Recall**: the fraction of correctly classified positive labels to the total number of positive examples in the data [14].

$$Recall = \frac{TP}{TP + FN} \quad (3.2)$$

- **Roc Curve**: Receiver operating characteristic curve shows the true positive rates against the false positive rate at various cut points. It also demonstrates a trade-off between sensitivity (recall and specificity or the true negative rate).
- **Accuracy**: Which presents the quintessential classification metric and means that shows how many true results are predicted correctly. It can be defined as the ratio of the correctly predicted labels to the total number of samples.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \quad (3.3)$$

### 3.9.2 Results

- **Loss and Accuracy**: After training our LSTM model only 5 epochs we scored an accuracy value of 98% for both training and validation set, while we got a loss values of 0.09 for training and 0.098 for validation as can be shown in Figure 3.3. However, logistic regression model was able to score a higher score of 1% point margin from LSTM model.

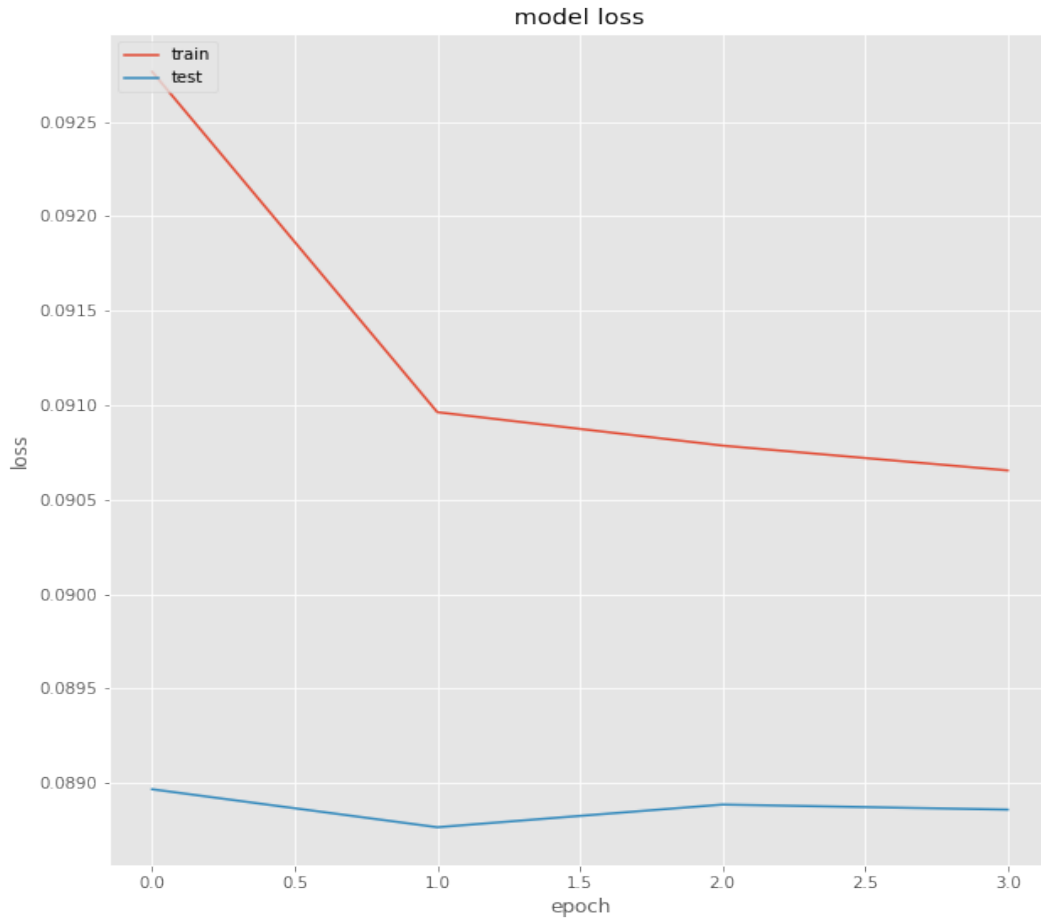


Figure 3.3: Training and validation loss over epochs

- **Confusion matrix:** As mentioned earlier the array from matrix presented in Table 3.4 is as follow , tn, fp, fn, tp.

Table 3.4: Confusion Matrix for LSTM Model

	Predicted Positive	Predicted Negative
Actual Positive	103232	0
Actual Negative	1887	0

Table 3.5: Confusion Matrix for Logistic regression Model

	Predicted Positive	Predicted Negative
Actual Positive	102979	253
Actual Negative	363	1525

Our true negative are 103232, meaning machines that the model consider that aren't in failure state, and our false positive is 0, meaning that LSTM model didn't predict that machine is able to predict failure while it can't in reality.

Our false negative is 1887 and true positive is also 0, meaning that LSTM model didn't correctly predict the machine failure while its in reality in a failure state.

This can be explained that we might have a lot of instances in our data set in which they are in a failure state compared to other instances who are not (balance in vibration, temperature) are similar to both. However, for logistic regression model (Table 3.5), it was no 0 values for FN and TN, meaning that this model enable to perform better predictions values compared to LSTM model.

- **Evaluation under other metrics:** Since the model accuracy is not always the efficient metric for the model evaluation especially when dealing with problems like class imbalance. We used other precise metrics which are widely adopted for time series classification task such as Precision, Recall, F1-score and ROC.

The logistic regression classifier was able to efficiently predict labels with a precision of 85% , while it scored a Recall of approximately 80%. (F1) metric presents the harmonic mean or balance between (P) and (R), our logistic classifier is still scoring an acceptable score of 83%.

According to the Figure 3.4 which contains the ROC curve of our logistic regression, our classifier gives a curve closer to the top-left corner (closer to the value 1) which is 0.99. this score shows that our model performing a better performance and a good measure of separability among all class labels. However, our labels have high correlation with some of the features, and since our target label are clear and easy to predict, models like decision tree and random forest in this case prove to score perfect results of 100% in almost all metrics.

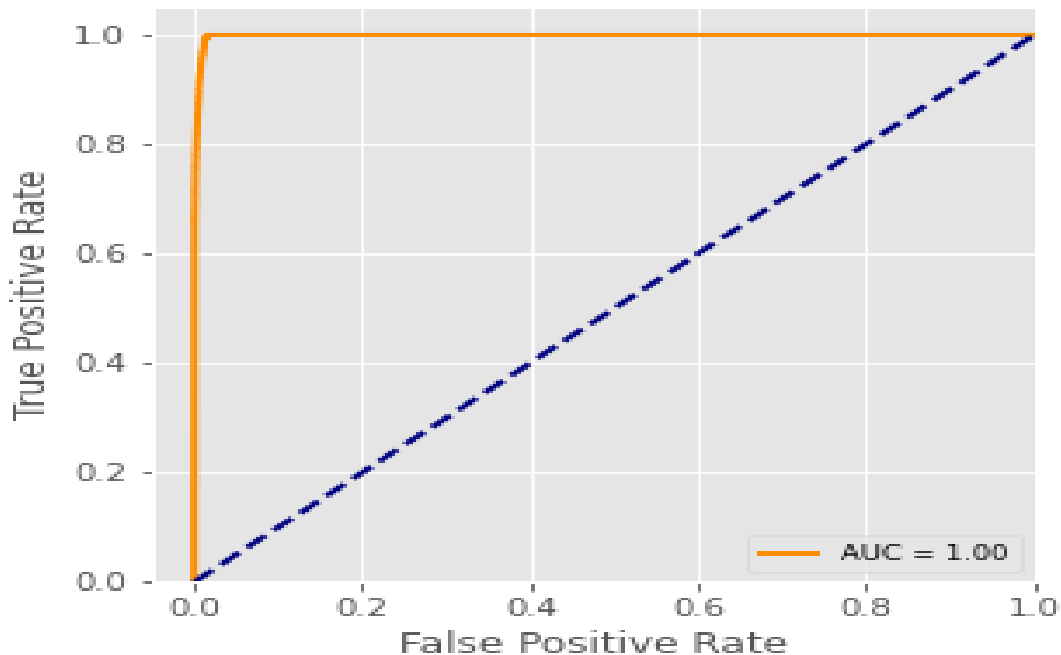


Figure 3.4: ROC curve plot.



## 3.10 Conclusion

In this chapter, we covered the different steps for building our predictive maintenance model. We evaluated our approach under the most known metrics for time series classification. Therefore, We was able to achieve a good prediction results after training our deep models.

## CHAPTER

## 4

# WEB APPLICATION

### 4.1 Introduction

Creating meaningful designs, capturing the attention of customers, and—more importantly—influencing their behavior is something that project managers, developers, and customers will agree that are the most important things to achieve.

For that reason, we have focused on this part in conceiving and implementing an application that delivers an impressive design using top trending user interface technologies, and that interact significantly with the company's business process through the use of SAP ERP different platforms.

Our redaction remains the same as the previous chapters in which we start with a sprint planning defining the different tasks then a sprint implementation describes the development process and a sprint review that exposes our final application.

Since this application interacts directly with the customer, we have considered this sprint the most essential part with ultimate priority.

### 4.2 Sprint Planning:

Table 4.1: Web Application: sprint backlog.

Id_US	User Story	Task_ID	Task	Estimation/ day
1	As a maintenance manager, I want to consult the scheduled maintenance tasks, So that I can easily assign and describe tasks.	1.1	Web service creation.	4
		1.2	Planning calendar implementation.	3
2	As an IT technician, I want to represent the entire factory, So that the maintenance technician could make updates and amends, without tampering with the physical asset itself.	2.1	Hierarchical Tree Creation.	9
		2.2	Plane creation.	3
		2.3	Loading Mechanism Implementation.	5
3	As a maintenance technician, I want to visualize the equipment status in real-time, So that I can perform service tasks at multiple sites simultaneously.	3.1	Paho client integration and implementation	3
		3.1	Sensors view page design	1
4	As a manager, I want the application to be compatible with the existing system, So that I can easily switch between them.	4.1	Integrating with the old app	2

## 4.3 Sprint Implementation:

Good planning without good working is nothing. *[Napoleon Hill]*

### 4.3.1 The planning calendar

Staying up-to-date with operations in the plant is an essential part for maintenance teams. Either it's a recurring task done at regular intervals or a one-time task, having it well scheduled can become a challenge for maintenance managers.

To help get maintenance jobs fall within a suitable time, and for the appropriate technician, we have thought about a planning calendar that aims to facilitate and accelerate the scheduling process.

Actually ,SAP offers a smart calendar through its intuitive web framework SAPUI5 that we have stood on to implement the different components of this project.

Based on Fiori user experience, this calendar could easily and effectively allow managers to make changes to the maintenance schedule.

In order to build our calendar, it was required to perform two major steps, first step consist of the creation of a web service, while the second includes data binding and view implementation.

#### **Creating the web service:**

The planning calendar allows users to see and make various arrangements simultaneously. In terms of maintenance, we can use the calendar to arrange the different orders and show the arrangements of several items related to our machines. Using this calendar and through navigating work orders, a manager can consult order details, history, operations, and more.

The following diagram 4.1 could describe better a work order by showing its structures as well as its attributes and relationships.

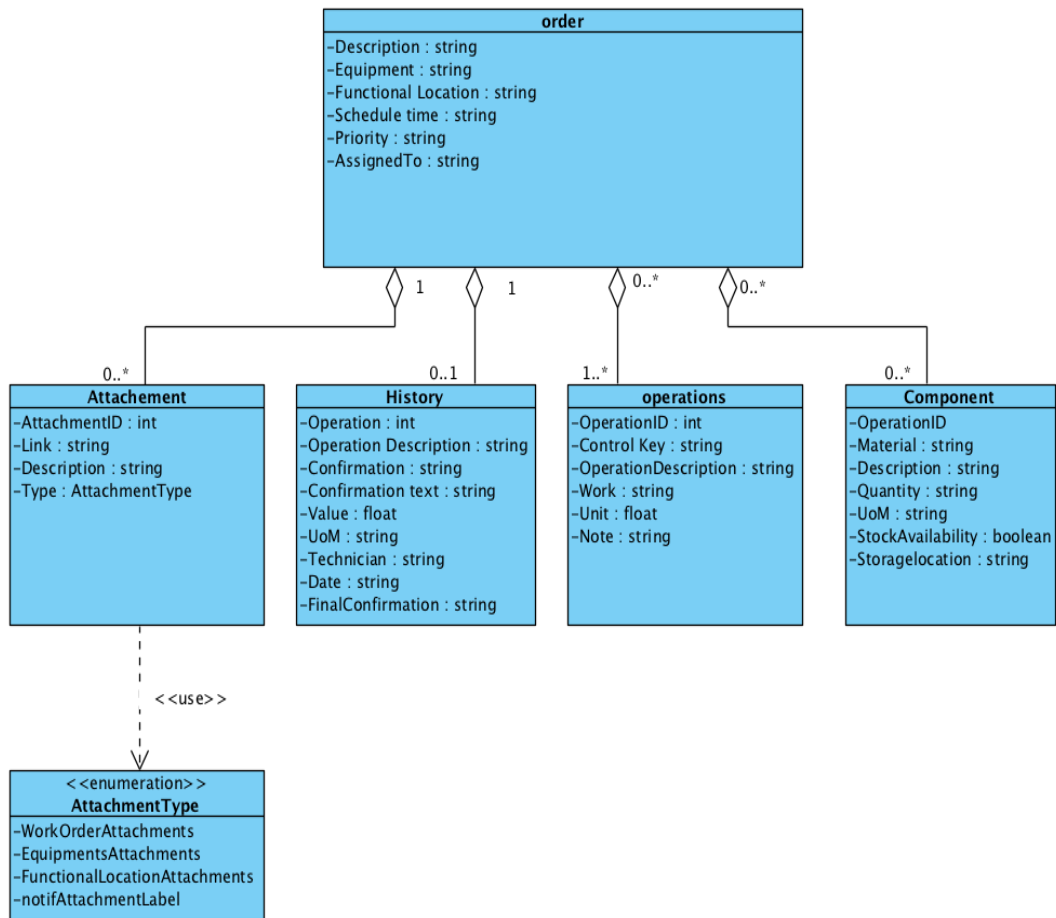


Figure 4.1: Work Order: class diagram.

Using SAP easy access and through the use of transactions, we have connected to the SAP gateway service that will allow data within our SAP system to be accessed by the outside world via OData services.

After that we define our data structure by creating entities and entity sets as well as the associations that link them together. Using another transaction and always with SAP gateway service builder, we have accessed to ABAP Workbench that enabled us through the ABAP programming language to process and transform our data before exposing it to the web services.

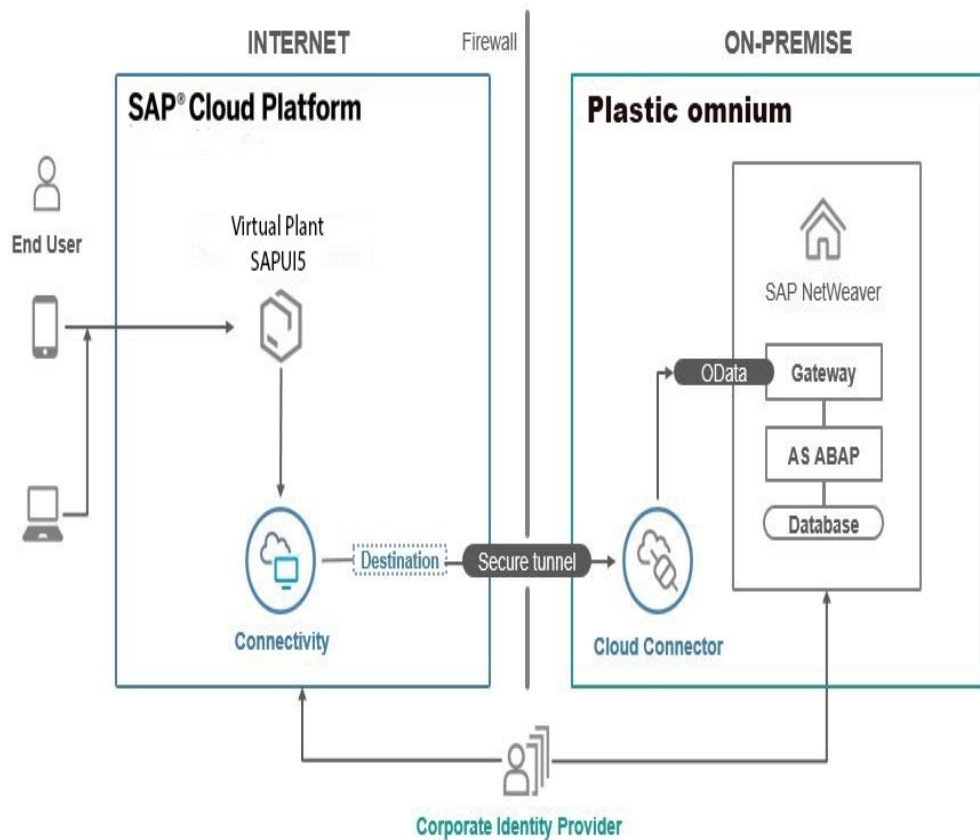


Figure 4.2: Web Application Development Architecture.

The Figure 4.3 explains how our web service interacts with our front end development platform.

### Data binding and user interfaces:

Based on the previously created web service and using different SAPUI5 technologies we will complete the implementation of our solution. SAPUI5 is a client UI technology based on JavaScript, CSS and HTML5. Apps developed with SAPUI5 run in a browser on any device (mobile, tablet or desktop PC).

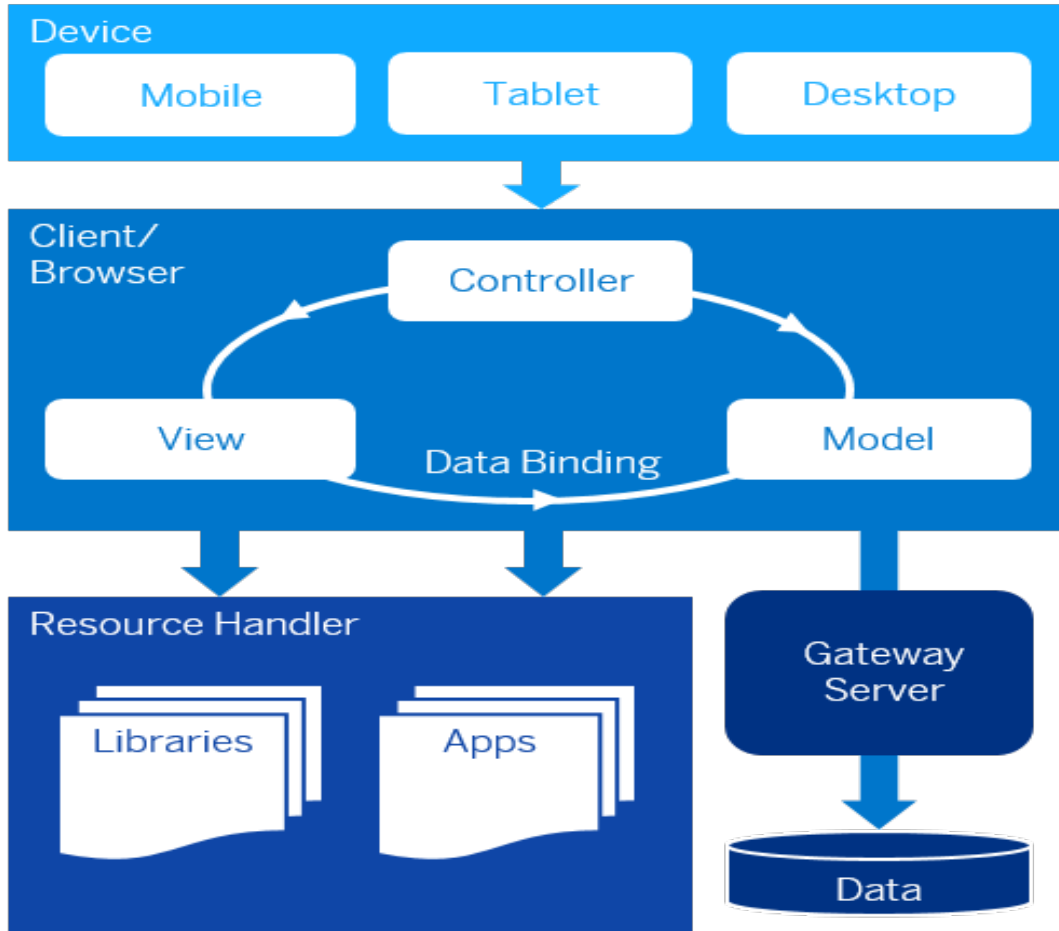


Figure 4.3: SAPUI General Architecture.

Following the MVC architecture used by sapui5, we were able in the first hand to represent our planning calendar and order navigation page and this is depending on XML and JS views.

Using OData and JSON models we interacted and retrieved data from our web service. And In order to allow views and models to interact with each other, we have used controllers.

### 4.3.2 The 3D Maps

Digital Twin has gained significant impetus as a breakthrough technological development that has the potential to transform the landscape of manufacturing today and tomorrow [15]. Digital Twin[16] acting as a mirror of the real world, provides a means of simulating, predicting and optimizing physical manufacturing systems and processes.

Now, Digital Twin has evolved into a broader concept that refers to a virtual representation of manufacturing elements such as personnel, products, assets and process definitions, a living model that continuously updates and changes as the physical counterpart changes to represent status, working conditions, product geometries and resource

states in a synchronous manner [17]. The digital representation provides both the elements and the dynamics of how a physical ‘thing’ operates and lives throughout its life cycle.

Based on this technology and in responding to our customer needs, we have thought to use a digital representation that will represent particular physical assets in the factory. This will subsequently allow technician maintenance to gauge the condition, performance, and history of the physical asset and make updates and amends, without tampering with the physical asset itself.

Our digital representation or as well as called “3D Map” consisted essentially from three different elements:

- A Tree that represents the hierarchy of the element consisting our map.
- A plane that will hold our 3D models.
- A mechanism to create and load 3D models to our map.

### Factory Hierarchical Tree

Having multiple elements constructing a plan makes drawing a 3D map become rather complicated. To simplify this task, We have used a tree that represents the hierarchies of the company in terms of equipment and locations.

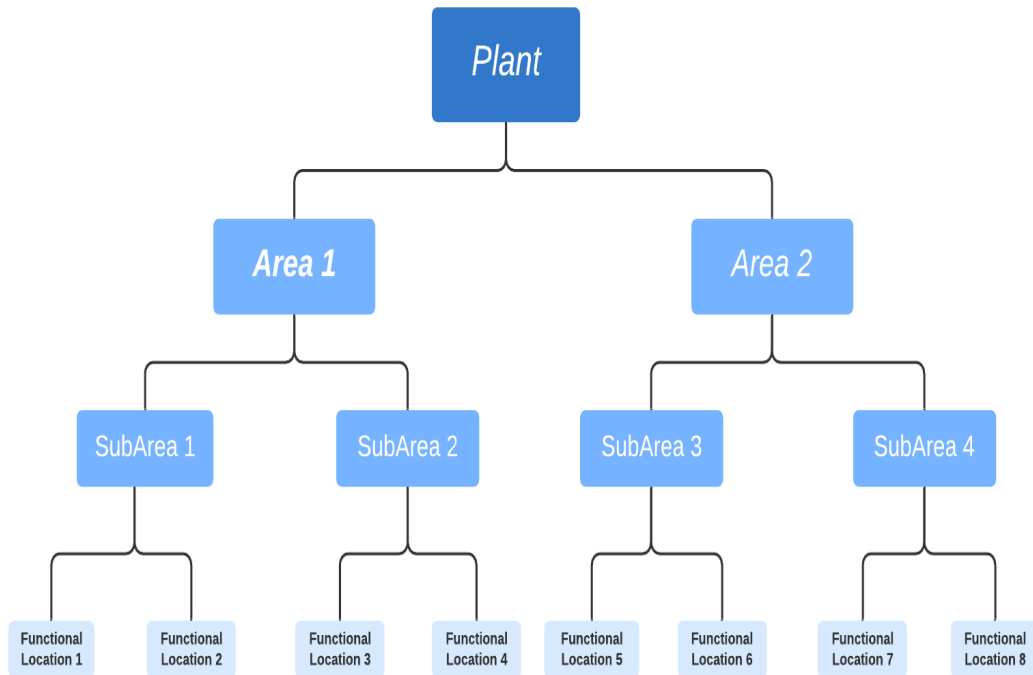


Figure 4.4: Factory physical architecture Hierarchy.



As it's shown in the Figure 4.4 our company hierarchy apply the SAP-PM module organizational structure where we find :

- **Plant:** in which maintenance task are planned.
- **Area:** A maintenance plant can be divided into sub-parts which are known as plant sections. For example, a plant can be divided into production area, store, electrical substation, etc.
- **Sub-Area:** The location where a functional object is physically installed in a plant section. It is used for informative purposes.
- **Functional location:** Represents the place at which a maintenance task is to be performed.

To be able to create this tree, we have conducted the same process used previously in the creation of our web service and link it with our front end component through the use of data binding.

### Plane creation:

In order to create and display our plane, we have used three.js, the famous Javascript library that integrates efficiently with SAPUI5.

Let's give an idea of the structure of our app. A three.js app requires to create a bunch of objects and connect them together. To summarize, the Figure 4.5 represents the different objects that consist our map as well as their relationship. As the class diagram explain, to create our plane we have used different objects including :

- A scene where we place our objects (e.g, geometries, lights and camera.)
- Perspective camera that mimics the way the human eye sees. It is the most common projection mode used for rendering a 3D scene.
- WebGLRenderer that displays and renders scenes using WebGL.
- SpotLight and DirectionalLight used in lighting our scene. This light can cast shadows.
- Boxgeometry to create our plane.

However, additional controls were used to enhance our application, like OrbitControls that allow the camera to orbit around a target and DragControls that provide the drag and drop functionality.

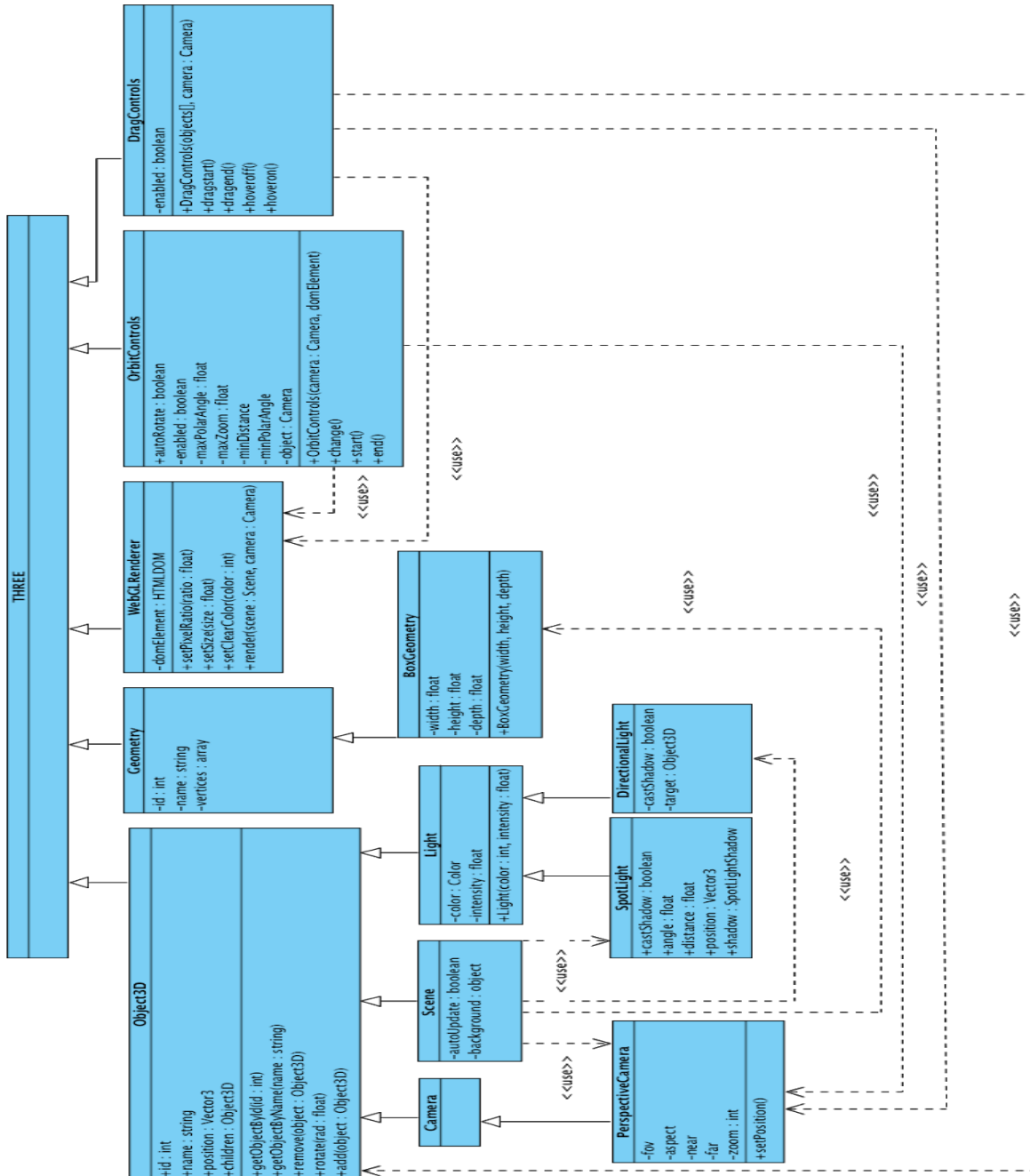


Figure 4.5: Plane creation class diagram.

### 3d Model Loading:

Creating and loading models that will fit the factory design could be a very complicated and time consuming task and since we are developing a proof of concept Thus, it was required from us to deliver a functionality to load the final model on the two corresponding categories. Later in a production environment, we will juste upload our models that fit the factory architecture and then we call our functions for the loading.

To mitigate this, first we constructed our 3D model in three.js using different geometries and texts, then we used our model in creating and adding area and functional location.

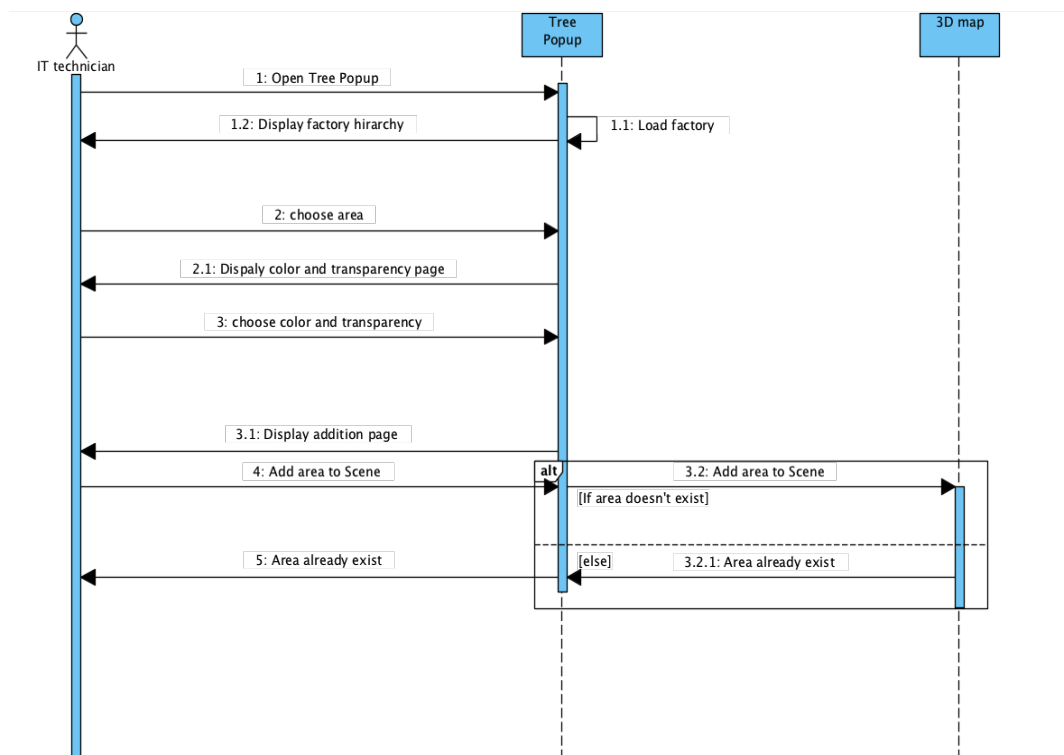


Figure 4.6: Web Application : adding area to the scene.

As the sequence diagram explains, an IT technician will use the tree to navigate and choose the elements (e.g., area, sub area, etc.) to be added to the plane.

Another page that contains an area description, where a user will choose the area to be added , after that another page in the tree will popup and ask for color and transparency. Then, a system will check if the area already exists in the scene, in case it does not exist, the creation of an area is done using a BoxGeometry, after that we assign a color using color palette from sapui5 and MeshBasicMaterial.

Using textSprite from three.js a we added a text containing the area name. Also, adding a subarea will follow the same instruction with a difference in the existing of the

parent area, meaning that we cannot add subarea if its parent does not exist. The second method of adding an element to our map is the use of external 3d models and we have applied this method in loading equipment as well as notifications.

Based on GLTF (GL Transmission Format) which can be defined as a 3D file format that stores 3D model information in JSON format. The use of JSON minimizes both the size of 3D assets and the runtime processing needed to unpack and use those assets. It was adopted for the efficient transmission and loading of 3D scenes and models by applications. Using using GLTFLoader from three.js, we have delivered and loaded our 3D content effectively.

### 4.3.3 Sensors view:

Front End development is no longer limited just to the Browser, Web-based user interfaces are expanding their boundaries to new devices. In IoT context, sensors are devices that can be used to measure a property, such as pressure, vibration, temperature, etc. and respond with feedback. Sensors present the ultimate solution that enables managers to detect and visualize real-world data in an intelligible way.

In order to benefit from the power of Sensors, it was required to connect them with our web application. Again using Paho client together with the flexibility of SAPUI5, we have integrated the Javascript library implementation that enables us to receive messages from the broker using WebSockets.

The following diagram 4.8 illustrates how the communication between the broker and the different clients occurs.

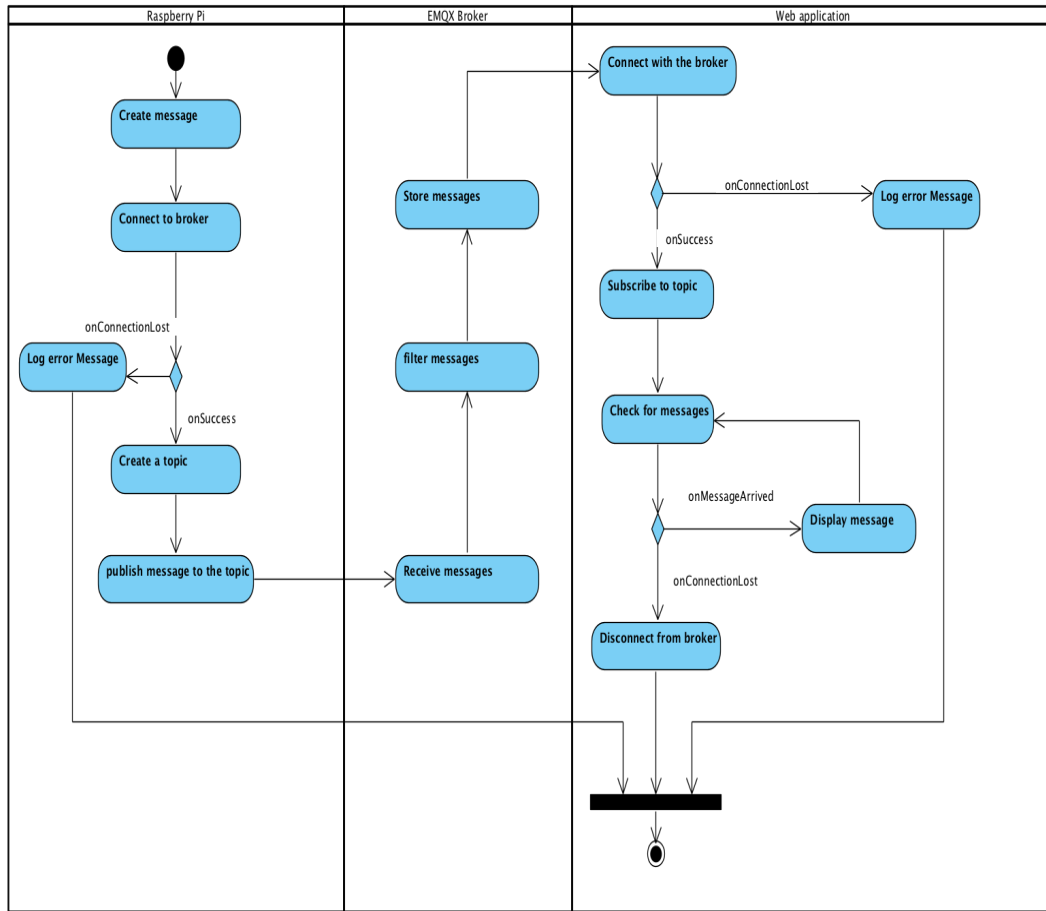


Figure 4.7: Activity diagram Dataflow.

#### 4.3.4 Work Integration

As we have seen during this sprint, our final solution is composed of independently designed applications each one with its own purposes.

To enable these components to work together, it was required to pass by a phase of integration where we will merge and optimize data and workflows between our different applications. To do So, we have updated the libraries of the existing application to fit the version that we have worked with. After that, we moved all related files, configurations, and dependencies to the existing application.

The following diagram 4.8 represents the different components used to model a static implementation view of our system.

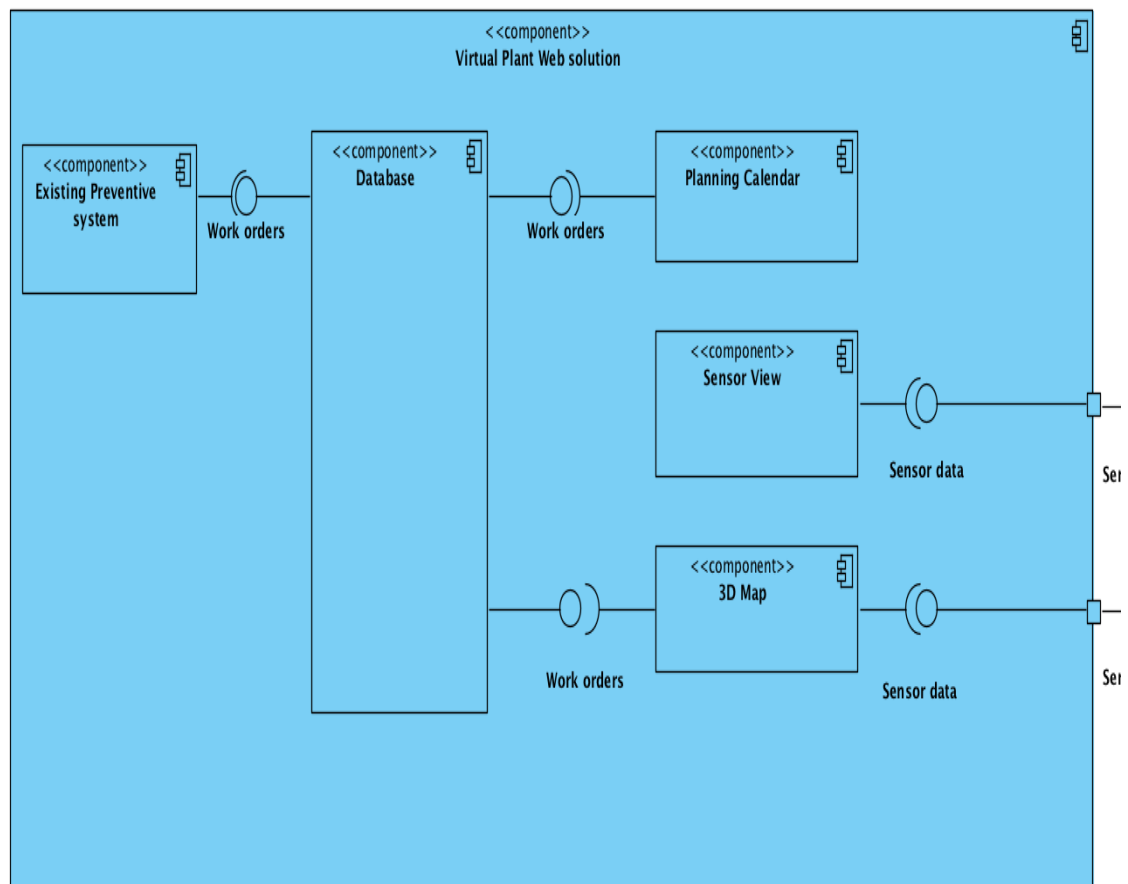


Figure 4.8: WEB Solution Component diagram.

As the diagram explains, our web solution consists of different components including SAP ERP that is used to fetch different work orders. In addition to that, several connections were made through specified ports to obtain sensor data from outside our system.

## 4.4 Sprint review

### 4.4.1 The planning calendar review

As the Figure 4.9 displays, using our calendar and through navigating a web order we can consult order details and operations related to this order as well as history and attachments.

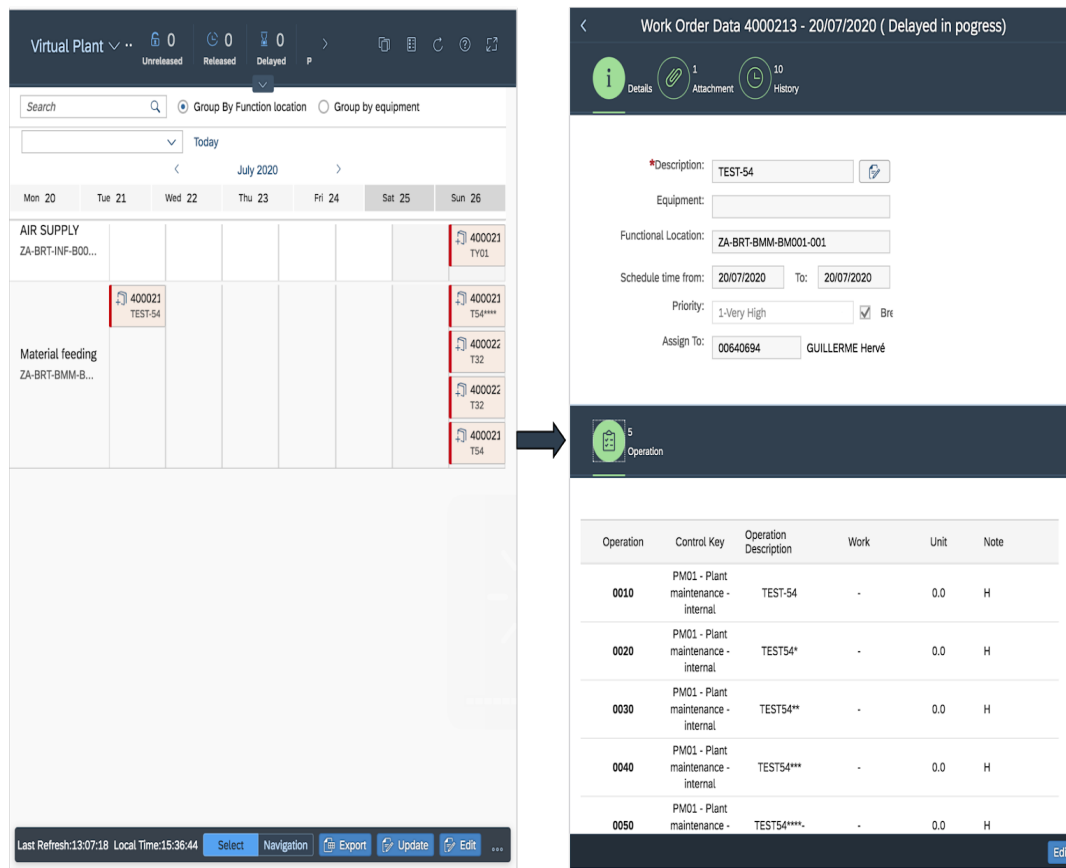


Figure 4.9: Web Application: planning calendar and navigation page.

However, our calendar can be enhanced by adding drag and drop functionality which will facilitate tasks management and improve user experience. Besides, We could also use this calendar in displaying predictive orders. Using different colors and filters would be great functionality to distinguish between orders types.

#### 4.4.2 3D Map review

As it was described in the implementation section, a 3D map necessitates the creation of a tree that will first allow the technician to add elements to our map.

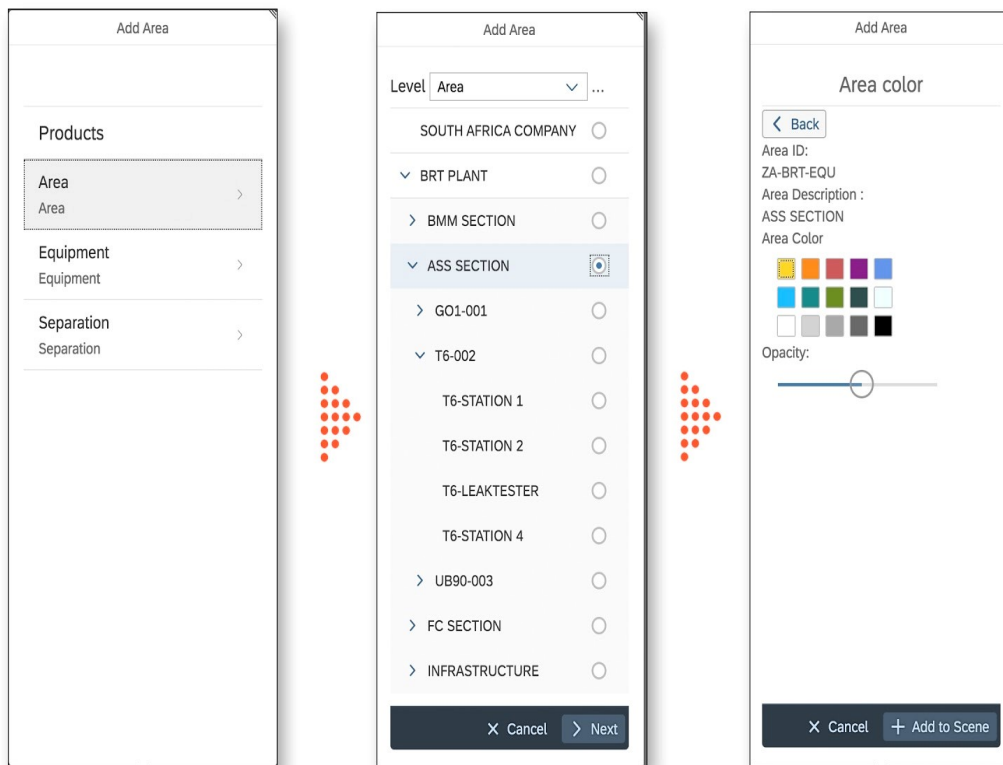


Figure 4.10: WEB application: Factory Hierarchical tree.

As the figure 4.10 displays, using the tree we could navigate through the different elements composing the factory (area, subarea, functional location, and equipment) we could also consult element description as well as assigning a color and transparency degree. After that using the “Add to scene” button we can add an element to our map. In example, the following Figure 4.11 presents our map containing an area.



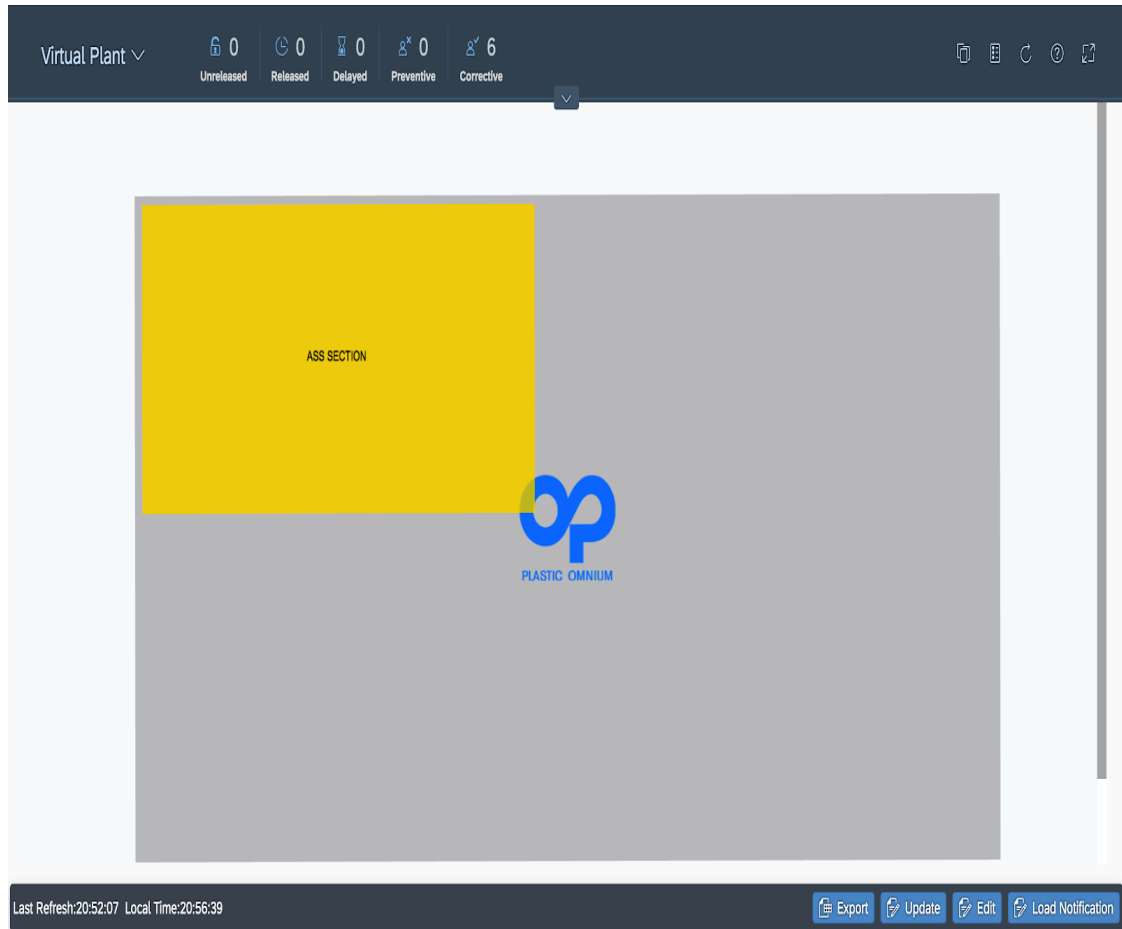


Figure 4.11: 3d Map: adding area.

After adding different shapes and geometries to represent our factory sections, we were asked to develop an algorithm that allows the loading of pre-existing 3D models. The following Figure 4.12 demonstrates a 3D model that represents a piece of equipment together with a cone as a notification.

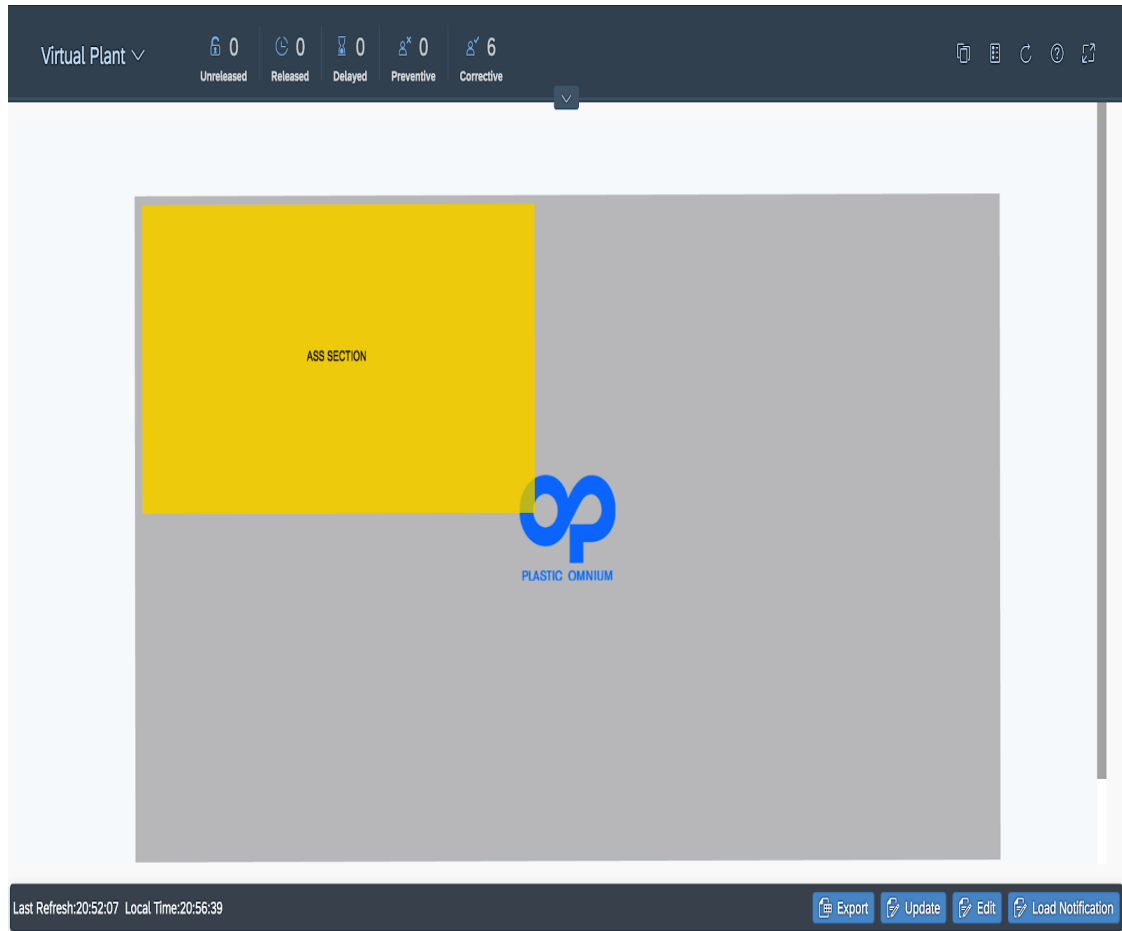


Figure 4.12: 3d Model Equipment.

### 4.4.3 Sensors View Review

As the Figure 4.13 demonstrates, we have arrived to connect temperature, vibration, and power consumption of the material feeding equipment through a simple web interface.

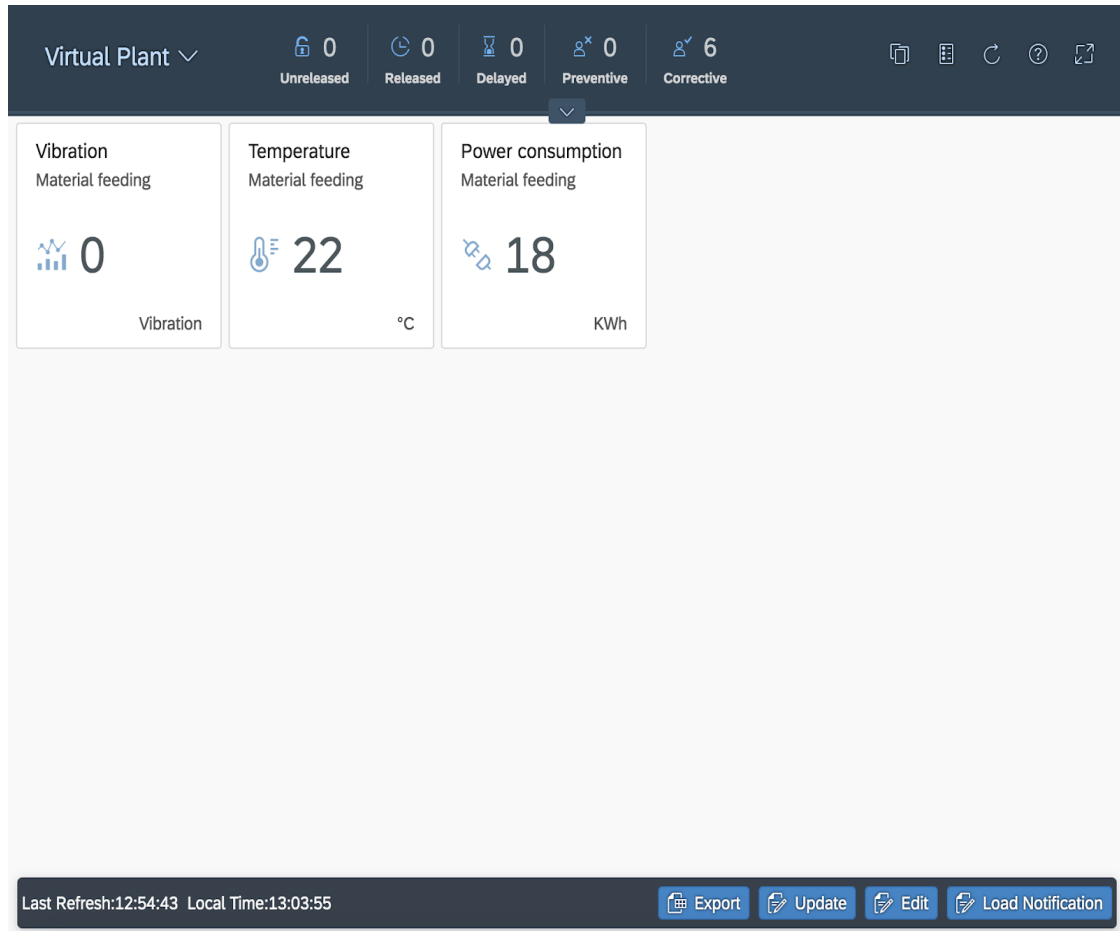


Figure 4.13: Web application: Sensors View.

In a future version of this app, we could record in the background all sensor data and render it as a chart, this will help maintenance technicians to visualize equipment condition history.

#### 4.4.4 Work integration review

Having a fully operational application that contains different components using different technologies was a challenging task.

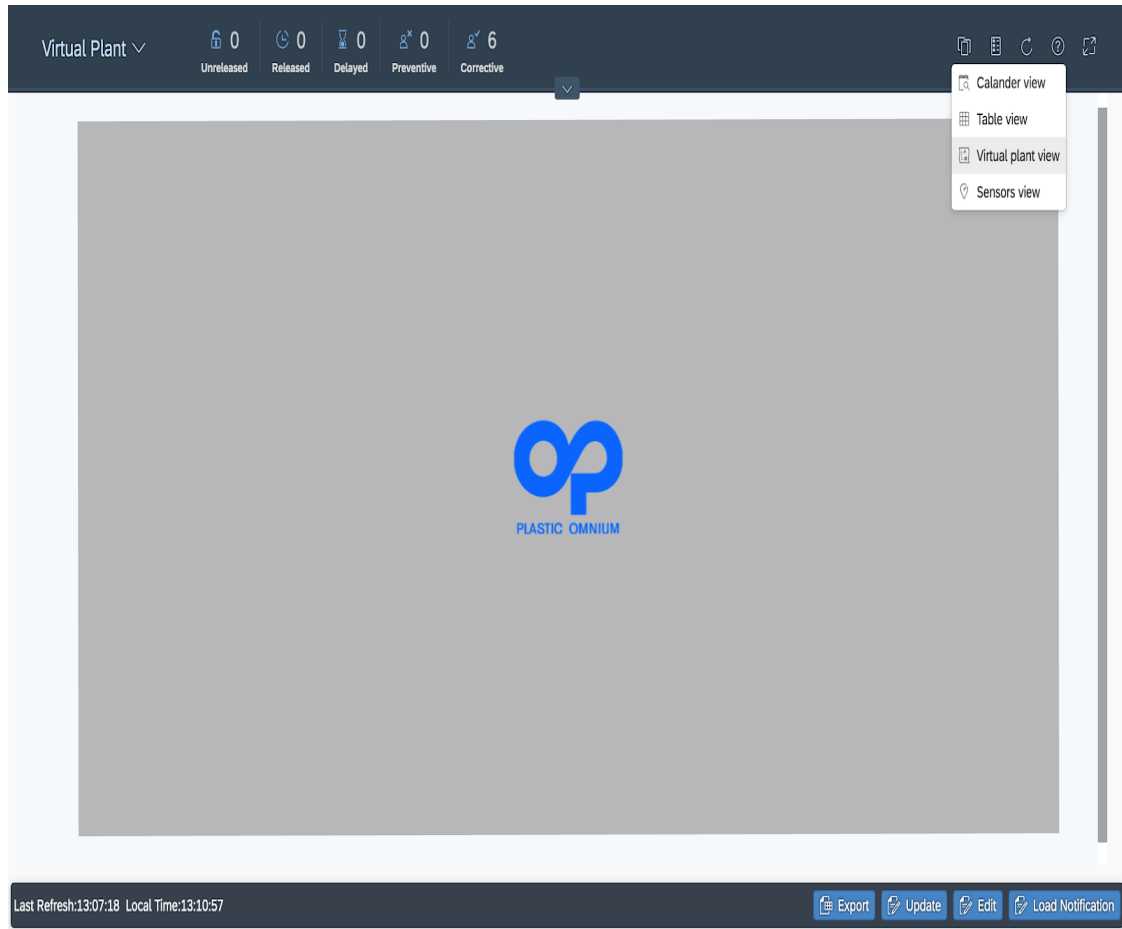


Figure 4.14: Web application: Work Integration.

As it's seen in the Figure 4.14, using a menu we could easily navigate between the different views while maintaining the responsiveness of the page.

## 4.5 Conclusion

Through this last chapter, we enhanced the existing preventive system by implementing a planning calendar that help organize better the work orders. we built a 3D map, to obtain digital representation of our factory. We also connected our sensor data to our application, while we put all the work together by integrating our different application into one single system solution.

## CONCLUSION AND PERSPECTIVE

Manufacturing is undergoing a digital transformation—the use of technology to improve business results—driven by smart technology and connected devices. In a manufacturing organization, digitization can include any function including the back office and supply chain applications, factory automation, data analytic and more. In this context, LINKSOFT consulting open its door and welcomed us to be a part of the digital transformation of Plastic Omnium that will increase the efficiency, productivity and accuracy of this industry.

This experience allowed us to accumulate a large amount of information, it was an opportunity for us to refine our capabilities in terms of cloud computing, artificial intelligence and the internet of things.

We were able in the first place to simulate how a raspberry PI and its different sensors would work, after that using google cloud platform we have run database and virtual machine cloud instances to connect different devices.

Furthermore, we were able to create a DL models that enable to predict machine failures with a good measure of performance (scores above 90% in all metrics.) In addition, this project has led us to discover and manipulate SAP tools which remains a leader in enterprise application software.

This experience also allowed us to deepen and put into practice our theoretical and practical knowledge acquired throughout our university course at ESIP.

As future work, our solution has the advantage of being open and extensible, in this context, we plan to expand our project by adding an augmented reality application that will enable technician to consult equipment conditions using their smart glasses.

# THREATS TO VALIDITY

Internal and external Threats to Validity are concepts that emphasize the trust-worthiness and meaningfulness of the results of a study. Although internal validity is related to how well a study (its structure) is performed, external validity is related to how the findings are applicable to the real world. Several threats might be impactful on the results and validity of our work. The problems that we faced during the project was mostly about the project deadline and data confidentiality.

- **Project deadline:** Due to the pandemic situation and the tasks complexity, we were not able to perfectly arrive to finish deploy our predictive model—Predictive models deployment provides the option to deploy the analytic result to every day decision making process, for automating the decision making process. the predictive model validation and deployment are time consuming activity, which takes months depending on the business scenarios. But overall we have accomplished the rest of all the necessary tasks to perform the prediction process.
- **Data inconsistency:** Since we performed our experiment on a synthetic data, our results remains not consistent, therefore, we aim in the future to use a real world data generated from the Plastic Ominuim company to better prove the trustfulness of our results

# BIBLIOGRAPHY

- [1] V. Marakana. Agile scrum methodology, year = 2020, howpublished=<https://medium.com/@vishal.marakana/agile-scrum-methodology-c1dbd7425dcf>.
- [2] R. Arsenault. Stat of the week: The (rising!) cost of downtime, year = 2016, howpublished=<https://www.aberdeen.com/techpro-essentials/stat-of-the-week-the-rising-cost-of-downtime>.
- [3] David Y. Software development methodologies. *White paper*, 08 2013.
- [4] Gaurav K. and Pradeep B. Impact of agile methodology on software development process. *International Journal of Computer Technology and Electronics Engineering (IJCTEE)*, 2:2249–6343, 08 2012.
- [5] James R., Ivar J., and Grady B. Unified modeling language reference manual, the (2nd edition). 2004.
- [6] Dipa S. and Ashwin M. A survey on mqtt: A protocol of internet of things(iot). 2017.
- [7] Richard D Hipp. SQLite, 2020.
- [8] D. Bruneo and F. De Vita. On the use of lstm networks for predictive maintenance in smart industries. In *2019 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 241–248, 2019.
- [9] Claude S. and Geoffrey I. Webb. Logistic regression. *Encyclopedia of Machine Learning*, 5:631–631, 2010.

- [10] Chollet and François. Keras. <https://keras.io>, 2015.
- [11] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.
- [12] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. VanderPlas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in python. *J. Mach. Learn. Res.*, 12:2825–2830, 2011.
- [13] Mohammad H. and Sulaiman Md. Nasir. A review on evaluation metrics for data classification evaluations. *International Journal of Data Mining Knowledge Management Process*, 5:01–11, 03 2015.
- [14] M. Sokolova and G. Lapalme. A systematic analysis of performance measures for classification tasks. *Inf. Process. Manage.*, 45(4):427–437, 2009.
- [15] G. Lo K.D. Bettenhausen R. Rosen, G. von Wichert. About the importance of autonomy and digital twins for the future of manufacturing. *IFAC-PapersOnLine* 48, 2:567–572, (2015).
- [16] D.D.S. Stargel E.E.H. Glaessgen. The digital twin paradigm for future nasa and us air force vehicles. *in: 53rd AIAA/ASME/ASCE/AHS/ASC structures, Structural Dynamics and Materials Conference 20th AIAA/ASME/AHS Adaptive Structures Conference 14th AIAA*, 2:p. 1818, 2012.
- [17] ISO. Digital twin manufacturing framework (under development). *ISO/AWI 23247*, 2:2249–6343, 2019.