



**Politecnico
di Torino**

Politecnico di Torino

Master's Degree in Computer Engineering

A.a. 2021/2022

Graduation session of October 2022

Design and development of a cloud-based application for smart agriculture

Master's Degree Thesis

Supervisors:

Prof. Claudio Casetti

Prof. Aurelio Somà

Candidate:

Pablo Andres Vejar Gomez

Summary

The agriculture sector has existed for thousands of years, however in the last period it has been innovating with solutions that improve this practice thanks to the advancement of technology and the digitization of machines in all industrial sectors. In this context, a thesis work is proposed in collaboration with the company WiRail SRL, whose objective is to design and develop functionalities within a cloud-based web application, which improve and help users to manage agricultural production through real-time monitoring of their machines. This work is part of a larger project aimed at developing control unit devices that monitor and measure parameters of diverse machines of the agricultural and construction industry, thus providing a whole service of status-data recollection and data processing.

The monitoring system is composed of three parts, a REST API server, a data receiver server and a front-end, based on technologies such as Angular, Spring Boot, Docker, MongoDB, among others. Thus, using this architecture it is shown how the new developments focused on improving the user experience and adding value to the use of the application were carried out. In particular, it is illustrated how the concept of activities together with the concept of Smart farming can significantly improve the management of agricultural machines, integrating them with algorithms and visual monitoring tools. Furthermore, the validation of design decisions using global standards of application development is shown. Finally, possible future continuations

of the project are discussed, with adaptations for other platforms and the implementation of additional functionalities.

Acknowledgements

First of all, I would like to thank the entire WiRail team for their support and availability during these months of work and development of the thesis, specially my tutor Angelo. A big thanks to my supervisors, Prof. Aurelio Somà and Prof. Claudio Casetti, who were available from the first day to the end.

Thanks to all my friends from Torino, who made my day to day life in this new stage of my life much nicer and easier, because when you find a group of people who make you feel like you are at home, everything becomes better. You guys make being 11.800 kilometers away from home worth it.

Thanks to my friends in Chile, lifelong friends, friends who were fundamental in my university period and friends who became closer just before leaving, thanks for all the support despite the distance and for all the love that you continue to show me.

A special thanks to my friend Catalina, thank you for always being by my side and accompanying me in every process from a distance, you have been a fundamental support in this experience.

Finally, I want to thank my family, because it is thanks to them that I am in this place today, after almost 7 years of supporting and helping me in everything I have needed to develop my professional and personal future, your love and support has been fundamental in my life.

Table of Contents

List of Figures	IX
1 Introduction	1
1.1 State of the art of the agricultural sector	2
1.2 Solutions in the market	3
1.3 Goal of thesis	9
2 Architecture of System	10
2.1 Control units	11
2.2 Receiver	16
2.3 REST API	17
2.4 Client-side	19
2.4.1 Maps	20
2.4.2 Graphics	21
2.4.3 Reports	23
2.4.4 Documents	24
2.4.5 Config	24
2.5 Database	24
2.6 Software Technologies	26
2.6.1 Docker	26
2.6.2 MongoDB	28

2.6.3	Angular	30
2.6.4	Spring Boot	32
2.6.5	Kibana and ElasticSearch	34
2.6.6	Digital Ocean	35
2.7	Programming Languages	35
2.7.1	Java	36
2.7.2	Typescript	36
2.8	Performance	37
3	Preventive Software Maintenance	40
3.1	Initial situation	40
3.2	User stories analysis	45
4	Smart Farming	48
4.1	Project approach	48
4.2	User Stories for new features	50
4.3	Activities as center of management	52
4.3.1	Previous Activity Management	54
4.3.2	Current Activities	55
4.3.3	Planned Activities	59
4.3.4	Completed Activities	61
4.4	Geofencing	63
4.4.1	Fields and Work Areas	63
4.4.2	List of Fields	65
4.4.3	Integration with Activities	66
4.5	Ray Casting Algorithm	69
4.6	Farm Implements	71
4.7	Limitations of Project	74

5	User Experience	76
5.1	Guidelines for improving UX	76
5.2	User-experience and Design Improvements	79
5.2.1	Guides	79
5.2.2	Statistics for Machines	80
5.2.3	Details on Map	81
5.2.4	Main Layer Transformation	83
5.3	Results of graphic updates	88
6	Conclusions	89
6.1	Conclusion	89
6.2	Further Work	91
	Bibliography	92

List of Figures

1.1	Europe Smart Farming Market Growth.	3
1.2	DOT Mobile for construction	6
1.3	DOT Mobile GPS	6
1.4	WAY web page	8
2.1	Diagram of system architecture	11
2.2	WiRail control unit - Vineyard Tractors type	12
2.3	WiRail control unit - Precision type	13
2.4	Control unit already installed in machine	15
2.5	Microservices Architecture pattern	18
2.6	SPA lifecycle vs Traditional Page lifecycle	19
2.7	WiRail Map homepage	21
2.8	WiRail Graphics page	22
2.9	WiRail Report GPSB3N310	23
2.10	WiRail Report Precision	23
2.11	Software containers structure	27
2.12	Receiver DockerFile	28
2.13	REST API Server DockerFile	28
2.14	Client-side DockerFile	28
2.15	MongoDB architecture	29
2.16	MongoDB Replica Set	30

2.17	Angular 8 Architecture	32
2.18	CPU usage of frontend and REST API	37
2.19	Memory usage of frontend and REST API	37
2.20	Bandwidth of frontend and REST API	38
2.21	CPU usage of receiver	38
2.22	Memory usage of receiver	38
2.23	Load (1 5 15) of receiver	39
2.24	Database storage usage	39
3.1	Graphics page error	42
3.2	Empty Documents saved correctly	43
3.3	Validation code introduced for Documents	44
3.4	Validation added to "Save Document" function	45
4.1	Manual activity configuration	54
4.2	Current Activities	56
4.3	New Activity	57
4.4	Activity Details	58
4.5	Activity Position when "Show map" is active	59
4.6	Delete planned activity	60
4.7	Completed activities list	62
4.8	Completed activities details	62
4.9	Geofence representation	63
4.10	New field	64
4.11	Create new field dialog	65
4.12	Elements in list	65
4.13	Delete field	66
4.14	Field interaction with control unit	67
4.15	Field position reported in activity tab	68
4.16	Implemented code to check if point is inside field	70

4.17	Final Path on data with algorithm [beta]	71
4.18	Farm implements section	73
4.19	Farm implements reports	74
5.1	Web site guides	80
5.2	Control Unit guides	80
5.3	Statistics for specific machine	81
5.4	Old machine summary on map	82
5.5	New machine summary on map	83
5.6	Old menu bar	84
5.7	New menu bar	85
5.8	Old list layout	86
5.9	New list layout	86
5.10	Old borders on elements	87
5.11	New borders on elements	87

Chapter 1

Introduction

The agricultural industry has had a strong impact on the history of mankind and has been growing strongly over the years. Agriculture plays a vital role in the economic development of countries, which is why many economic sectors have set themselves the goal of modernizing the industry and taking advantage of current technologies.

With the introduction of IoT to the industries, a revolution has been created such that, technologies and new techniques have managed to be integrated in various industrial sectors, with the aim of improving the production and their distribution. This is where the concept of Industry 4.0 is born, which corresponds to the fourth revolution of the industry that is here to stay and change the way we do things.

The increase of knowledge in computer engineering and the integration with other disciplines, has transformed IoT and data analysis systems into an essential feature, resulting into a more optimized and sustainable way of producing and to allow a better decision making. The value created by these tools is even greater when the data is applied with other techniques such as ERP, supply chain, among others.

By combining the concept of Industry 4.0 to agriculture is born what is

known today as Smart Farming. It corresponds to the use of IoT within farm devices, which allows building databases that can be shared and used by multiple entities. This means that financial and time costs caused by human error can be reduced. Today there are already sensors within the agricultural sector to monitor soil nutrition, temperature, moisture, parameters associated with the tools used, among others. This allows the farmer to work on what really matters and provides benefits.

1.1 State of the art of the agricultural sector

The agricultural production of the European Union is dominated by livestock products, which correspond to vegetables, fruits, dairy products, wine, grains; being the most regularly exported products. In spite of this, the EU has progressively become a bigger importer of agricultural products, due to the globalization of the world economy and the low costs present in low-income countries. [1]

In this context, the EU is trying to impose its dominance in agriculture by including more technologies, aiming at the sustainability of the industry, i.e. using fewer resources while reducing the impact on the environment. For this reason, EU member states have begun to prioritize the modernization of European farms, introducing the concept of Smart Farming as the industry's standard of action. Experts predict that the introduction of this new concept will grow the market rapidly in a short period of time, reaching values of €6.6 billion by 2023.[2]

The aim in investing in smart agriculture is to make the EU the leader in technological knowledge and environmentally friendly farming, creating a model that can be copied by the rest of the world. In this context, a funding program called The Life GAIA Sense created by an important Greek association, is aimed at investing in this new technology, giving farmers

across Europe the opportunity to install sensors in their fields and on their machines. These sensors can be used to monitor many field conditions and machine status in order to reduce environmental impact.

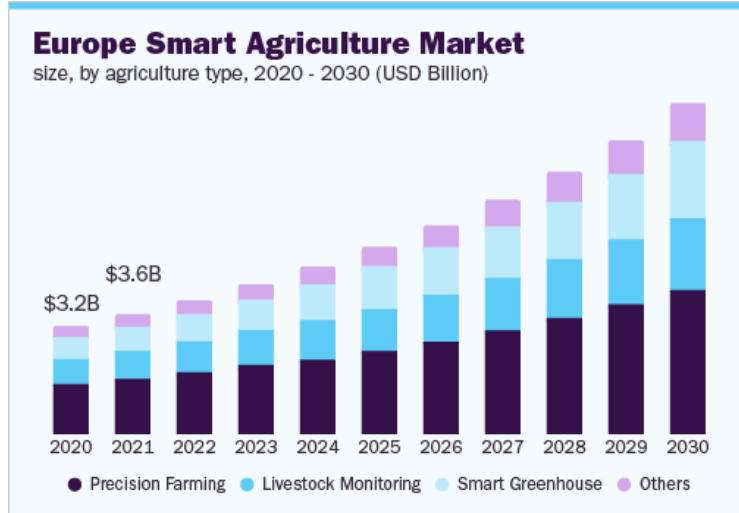


Figure 1.1: Europe Smart Farming Market Growth.

Source: www.grandviewresearch.com

1.2 Solutions in the market

The potential growth of Industry 4.0 in agriculture has allowed many agricultural companies to change the way they manage production, opening up to the integration of new technologies to reduce financial costs and environmental impacts.

For this reason, several projects are born to provide solutions for the application of Smart Farming within agricultural enterprises, offering various opportunities through new technologies. The type of technology of interest for this thesis corresponds to the monitoring, further analysis and control of the activity of agricultural implements, such as tractors, plowing machines, harvesting machines, among others.

Existing monitoring tools play a fundamental role in agricultural work, which is to collect data of interest in real time and thus create a complete database for the analysis of machinery farming behaviors. These tools, integrated and applied with software engineering, produce user-friendly products to deliver the Smart Farming experience and provide the opportunity to improve the outcomes of enterprises and small farmers.

For the correct operation and access of technologies of this type to the market, it is important to highlight the following conditions for competition:

- Produce monitoring tools at low economic cost;
- Work on a Wireless Sensor Network;
- Simplicity, robustness and confidence in the quality of the product;
- Possibility of geo-localization of equipment;
- Have a high compatibility with existing agricultural machinery;
- Low power requirements;

The following are some of the existing alternatives in the market that offer the above mentioned services.

Dot Mobile (Punto Mobile)

DOTMobile is a telemetry platform for the monitoring and management of vehicles. Currently, they offer automotive services, for agriculture, construction, nautical and work in general. They are present in US and Europe with a coverage in more than 100 countries and with 3 real companies: DotMobile Inc, Punto Mobile Srl and I.T.Comm Srl.

The technology provided by this company offers solutions for real-time location and monitoring, in order to reduce management costs and optimize

work. This is achieved through the implementation of on-board sensors inside its devices, such as CAN BUS, Analog/Digital Input, Mileage, etc. Moreover, they allow to obtain a tracking of the historical data on cartography, allowing the users to visualize the routes made and the position of their machines on a map.

At the moment of analyzing the digital platform, it can be observed that the application has the following characteristics:

- Graphical visualization of a map with the position of the machines next to the terrain where they perform activities;
- Visualization of machine performance analyses;
- Visualization of the machine paths;
- Possibility to configure the security of your machines;
- Scheduling or predicting maintenance;

From the analysis, it can be concluded that, like many competitors in the sector, they use Google Maps to record the position of their machines. They also use similar strategies to record the routes, i.e. taking the position recorded by the GPS and joining the points.[3]

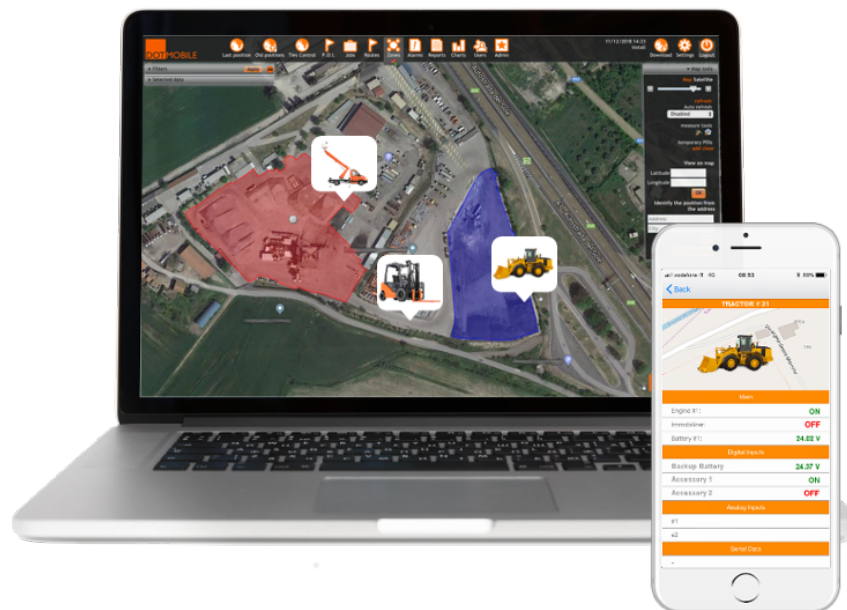


Figure 1.2: DOT Mobile for construction

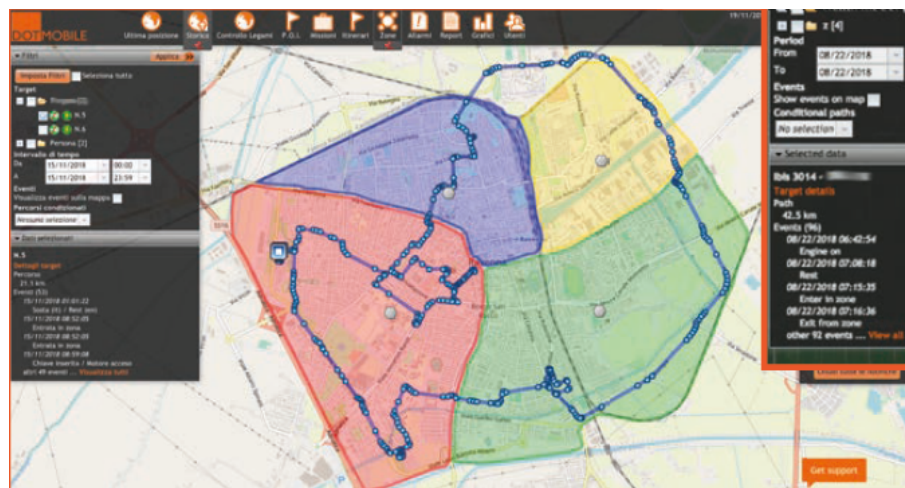


Figure 1.3: DOT Mobile GPS

W.A.Y Srl. - Where Are You?

WAY Srl is a company that was born as a technology partner for IoT solutions applied to localization and advanced management of vehicles, people and goods in motion. Thus, one of its work areas corresponds to the application of industry 4.0 in agriculture, proposing to the market a kit called Kit WAY4FARM.[4]

As many companies do in this sector, WAY offers monitoring services, giving way to a better management and optimization of resources to agricultural companies. The WAY4FARM Kit solution of this company consists of the following:

- A device to be installed on the agricultural machine to enable remote monitoring and command management functionalities;
- A CANBUS interface programmed and assembled for the purpose of reading and interpreting the telemetry data diagnosed by the machine;
- Installation services of the devices directly on the machines at the customer's site;
- Access to a web platform to take advantage of the services offered for a period of 5 years;
- SIM and telephone communication for data transfer to cloud platform
- Software support and maintenance services for installed devices

The device is equipped with a SIM for bidirectional data transmission between the cloud platform and the agricultural machine. After the collection of diagnostic data, users can access the website to take advantage of all functionalities made available by the company. These functions include the following:

- Show in detail the status of digital or analog sensors that detect that a vehicle is operating;
- Follow a vehicle on the map as it travels, view the route and all activities done during a day. Possibility of highlight stopping points with time duration;
- Display the current position of all machines on the map or of a specific fleet;
- Check the status of activities of each area by accessing real-time and historical data with details of sensors for each of the positions surveyed;

After analyzing the company's website, it can be seen that they also use google technology to give function to their maps. However, the user experience at the level of graphical interaction with the application does not provide great value. Figure 1.4 shows one of the pages of the application, where the work areas are managed together with the machines.

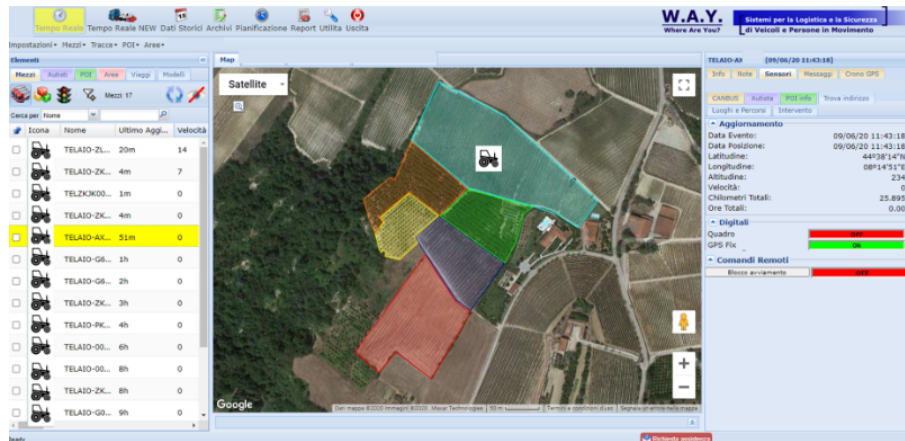


Figure 1.4: WAY web page

1.3 Goal of thesis

The objective of this thesis is to carry out an integrated development and design of new software functionalities to an existing system in a company that impulses Smart Farming, applying software development techniques considering the user requirements and user experience issues. This work aims to create new tools as well as update the existing ones, that are able to meet the new requirements of customers and the agricultural market by improving the management of agricultural production.

The work will be validated by testing with real users who have real needs, and provide feedback as the new tools are released.

Chapter 2

Architecture of System

In this chapter the overall architecture of the actual system will be presented, together with a brief presentation of the technologies that compose it.

All design and architecture decisions were made by previous engineers that had the opportunity to work on this project. These decisions were taken considering different factors such as a possible growth of the system due to an increment in connections, changes in requirements by stakeholders, ease to modifications to code, among others.

A graphic representation of the system and its components is presented in figure 2.1.

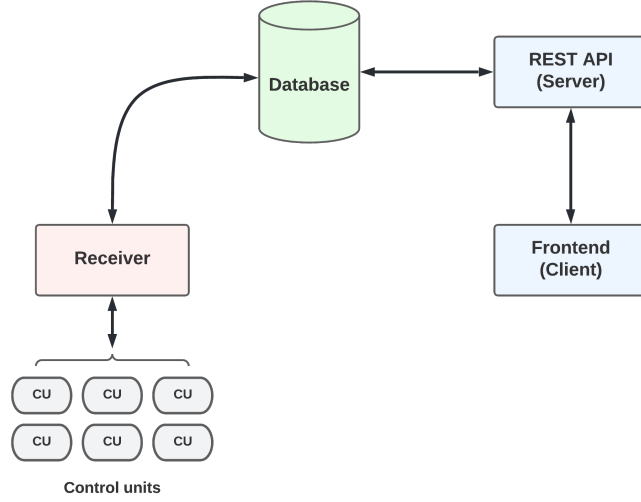


Figure 2.1: Diagram of system architecture

2.1 Control units

The structure of this monitoring system has several pillars and basements, nevertheless one of the most important parts of this project are the control units. They allow the connection between the receiver and the machine by using a TCP connection (Transmission Control Protocol) through the modem inside the control unit. Consequently, all data transmission is sent to a domain instead of a specific IP address, since in case of changes in the address of the system server, all functionalities can keep working without the necessity of updating the firmware of the control unit.

Agricultural machinery and platforms vehicles are very diverse and fulfill different goals in their industry, therefore control units cannot be the same for all machines. Every control unit is assembled and configured depending on its final purpose, which means that electronic components inside the devices can also vary. For the matter of this project, there are 10 different types listed as follows:

- Baler: Mostly used for hay baler farming machines
- Vineyard Tractors: Used in generic type of tractor of a specific company
- Construction Machines: Used for electric self-lifting scaffolding of a specific company
- GPSB3M4: Used for GPS monitoring of a specific company's machines with focusing on preventive maintenance of 4 specific parameters
- GPSB3N2: Used for monitoring of construction machines, such as hydraulic systems for lifting with the addition of two machine type-specific parameters: height and tilt of the telescopic boom
- GPSB3N3: Used for monitoring of construction machines, such as cranes
- GPSB10N5S3: Used for monitoring of construction machines, such as cranes with the addition of parameters taken directly from the car's engine control unit from the CANbus line
- Precision: Used for electric self-lifting scaffolding of specific machine models with a specific position-precision request within a range of less than one meter from the machine

Figures 2.2 and 2.3 show two different types of control units that are applied in different farming machines.

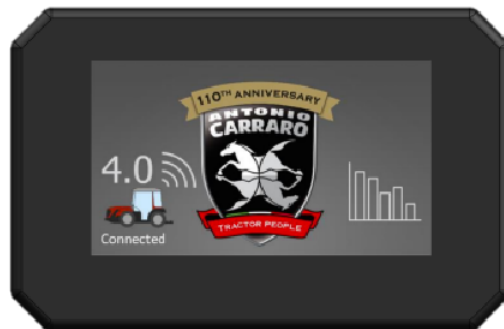


Figure 2.2: WiRail control unit - Vineyard Tractors type

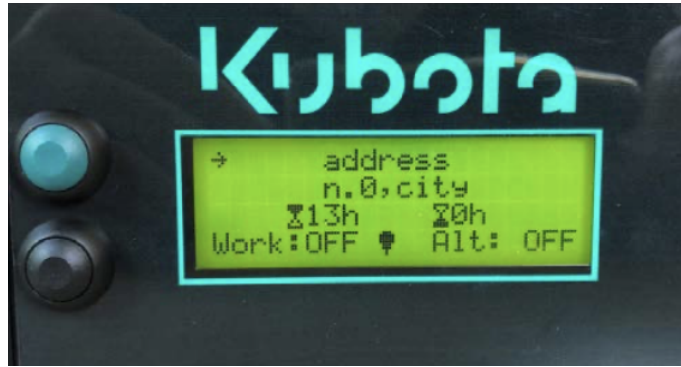


Figure 2.3: WiRail control unit - Precision type

At the beginning of a connection, every control unit or asset receives a command that declares the configuration that defines the behaviour with which it should operate. This command corresponds to a transmission string, where every sub-string is separated by a comma and ended by a carriage return followed by a lined feed, i.e. $\backslash r \backslash n$. Once the connection has been established and the control unit has received its configuration through a command, it starts to collect the specified data, save it on its own sd and then send them to database through the receiver connection.

The process of transmission of strings occurs in an interval of time that is predefined by the clients or by the administrator of the company, it usually goes from every 2 minutes to every 8 minutes and it follows the next pattern:

1. A new communication socket is opened and the control unit sends its parameters such as serial number.
2. The receiver recognises the other part of the connection and sends the command that will define the behaviour of the asset.
3. The transmission of strings starts following the defined interval.

As previously mentioned, all control units are different and have different parameters that they can measure. Hence, strings sent from the asset can be different in quantity of measured parameters and type of the represented

data, whether they are an integer or a float point number, a Boolean or a string. Nevertheless, these strings can be classified in two types: header string and node string.

A header string is used as the first package sent after the connection for the receiver to recognise the asset. It contains the following standard parameters:

- Hub ID: A unique serial number that identifies the control unit. It is defined by fabric and used as an identifier in the system.
- Value: A specific value for the machine's time in sending a header string
- AssetV: The value in milliseconds of the period that the asset sends a string

On the other hand, a node string contains all the information about the state of the machine and its parameters. As an example, a Precision type control unit has the following parameters:

- *Presenza Uomo*: This boolean parameter is used to identify whether a person is present inside the vehicle.
- Hand-break: Corresponds to a boolean parameter that shows if the hand-brake of the machine is on or off.
- Alternator: The proper operation of the alternator with preventive alerts in case of failure.
- Latitude: Represents the latitude coordinate of the GPS inside the control unit.
- Longitude: Represents the longitude coordinate of the GPS inside the control unit.

- Point Of Interest (POI): Represents the precise GPS coordinates to schedule the geographical point of start or end of a specific processing.
- Speed: Integer value that shows the speed of the vehicle where the aset is installed.
- Acknowledged (Ack): Corresponds to a boolean parameter that can be altered by the user, depending on whether he or she recognises the activity of the running control unit.

In order to not overflow the communication channels, there is a maximum limit of 30 strings per minute; when a limit violation occurs, the receiver closes the connection and sets the control unit to an alarm state, sending a notification to the client.



Figure 2.4: Control unit already installed in machine

2.2 Receiver

The receiver is modelled as a Client-Server model, which is a distributed architecture that partitions tasks between two subjects, one being the server or provider of resources, and the other being the client or service requester. This complex structure is mainly in charge of handling the multiple connections of control units, using the TCP method through the use of sockets to collect data these assets send constantly; with a maximum capacity of 500 connections simultaneously.

As mentioned in the previous chapter, control units are classified depending on their final purposes, therefore inside the receiver all types are modelled as different services with functionalities that change in base of the requirements. Consequently, the complexity of this structure increments based on new developments and requirements that the market or final clients could have.

Currently, the main functions that the receiver implements are the following:

- Connection manager: As already mentioned, the receiver is in charge of managing the connections of the machines and recognizing them to iterate over the data they send. The receiver handles also in a case where a control unit is not recognized or does not exist at all.
- Connection notifications: Every time a recognised machine connects, if so desired by the client, a connection notification is sent to those in charge of the control unit.
- Parameter management: The receiver is capable of modifying parameters belonging to the machines if the commands so indicate.
- Fault notifications: Every time a machine presents incorrect or unusual parameters, which could imply a problem with the machine, it takes

care of notifying clients and interested parties about the status of the asset.

- Data correctness verification: Since some control units send a lot of data in a short time, it is possible that some measurements are not exactly accurate. For this reason, a verification of the parameters received by the control units is performed inside the receiver and managed depending on the case.

The technologies implemented in the receiver and its structure allow this component to be scalable in the amount of functionalities it can execute. This gives way for future works, where functionalities can be added to make this component more versatile, complete and optimal.

2.3 REST API (Server)

The server is modeled with a microservices architecture, which corresponds to a typology of software development that works as a set of small services that run independently and autonomously, providing a completeness of the application at the time of joining them. Each microservice has a unique database and performs specific functions, yet they can communicate with each other through the use of APIS.[5]

This type of architecture has remarkable advantages compared to other types of architectures. The following is a list of the main advantages:

- The use of a microservices architecture avoids overloading the application, since each service is independent and can operate autonomously;
- In the event of failure of one of the services, the entire application is prevented from crashing, since the other microservices can continue to operate;

- This type of architecture allows flexibility when adding new services in case of application growth, since it is not necessary to modify the entire infrastructure of the project to add a new service;
- In case of maintenance, costs are more economical and efforts more simple, since it is not necessary to intervene on the entire structure;

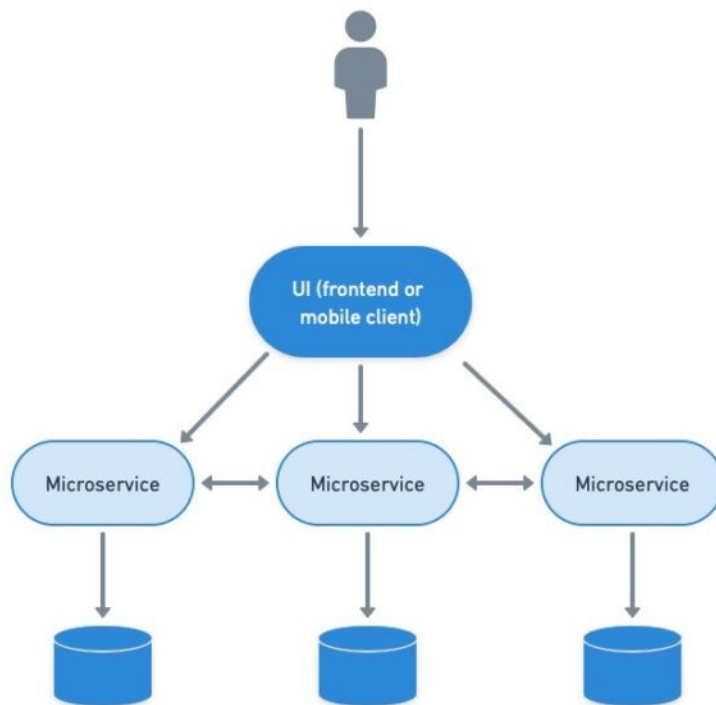


Figure 2.5: Microservices Architecture pattern

The REST API in this project is mainly in charge of controlling data access permissions and responding to requests made from the client-side. In addition, inside the server are stored some elements that are used as common resources of consumption of the client, such as user guides, images of the machines, documents for mails, among others.

2.4 Client-side (Frontend)

The client-side or frontend corresponds to the interactive part of the project that links all the components to make them visible and available to final users. This component is based on Angular -technology that will be explained later-, which means that the web application is developed using the concept of dynamic Single-Page Applications (SPAs).

A SPA is a web application that handles user interaction in such a way that it dynamically rewrites the entire contents of the current web page, showing the user the new data from the server. This contrasts with the default method that web browsers have of loading entire new pages. Thus, it allows the application to be more efficient by increasing the transition speed, therefore eliminating page refreshes and making the website feel more like a native application.[6]

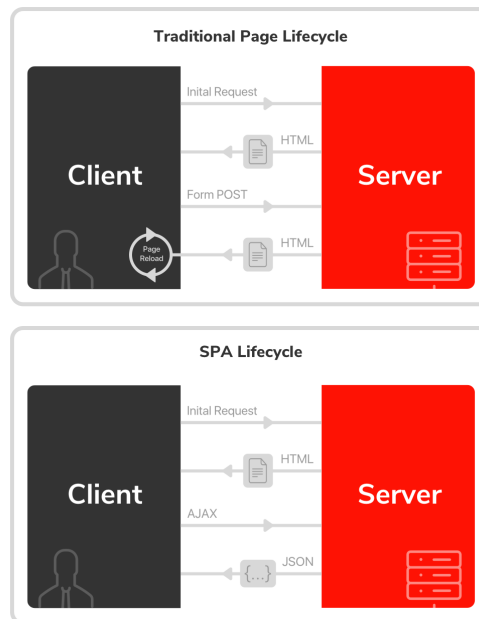


Figure 2.6: SPA lifecycle vs Traditional Page lifecycle

An nginx server has been used in this project as a support to issue the

REST API calls. In addition, a famous software called Gzip has been used to reduce the dimensions of the transferred pages, which means an increase in the loading speed of the pages.

The application is composed of several pages that provide the necessary functionalities for the administration of the control units and the visualization of the data collected by them. In the following sub-chapters the main pages and their features are presented.

2.4.1 Map

This page corresponds to the homepage of the application because it shows one of the most important information for the final users, the geographical location of their machines. On this page all the machines associated to the user's account can be displayed. There are two display modes available for the type of machines under study. The first one is "Last Position" which shows the last GPS position retrieved by the control unit, and the second mode called "Date Range" shows the route made by a machine between the two selected deadlines.

The map service used is that of Google, due to its immediate integration with Angular. In addition, this API offers the ability to customize the maps, choosing the type of layers to use and the option to hide information that may be unnecessary for end users, such as bars, pharmacies, supermarkets, among others, rendering more visible streets or field boundaries.

Among the graphical decisions oriented to the user experience that were made are the following:

- Each control unit is represented by a circle of a random color, differentiating it from the others;
- The representative circles are clickable, showing the detail of the last connection and giving the possibility to go to other pages of the application

to see more detailed information;

- If the "Date range" mode is selected, the trip display allows you to differentiate the starting point and the end point using two different icons;
- A search bar is provided so that users can insert a specific machine and not see all existing information on the map;

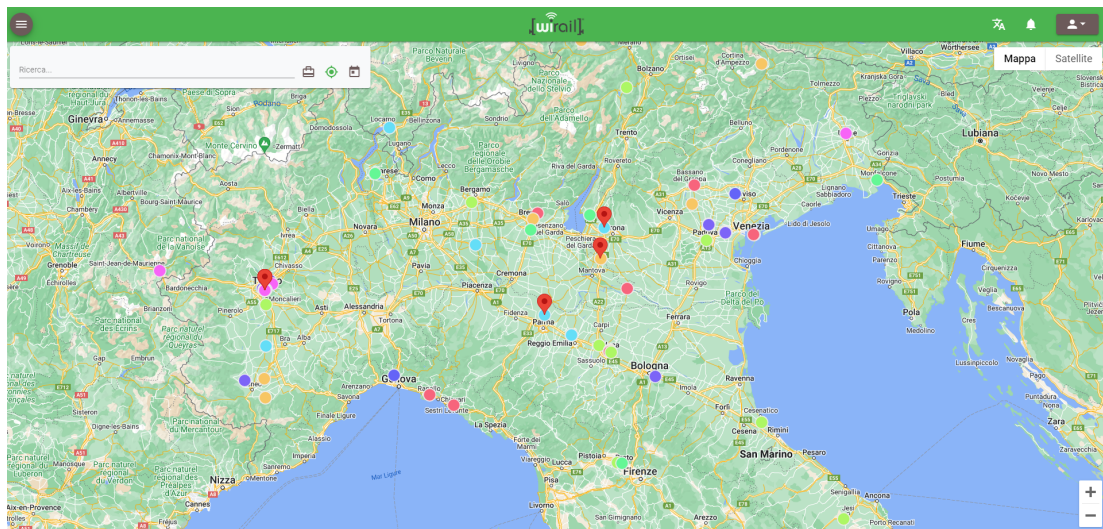


Figure 2.7: WiRail Map homepage

2.4.2 Graphics

The charts section is available only for some types of machines, so those users whose accounts have one or more machines of the specific type associated with them, can access and view data on these charts. This functionality receives a date range and an identifier that is sent through the search engine to look for the specific control unit. The graphed data is as follows:

- Speed versus time
- Instantaneous consumption versus time

- Motor loading versus time
- Engine RPM versus time

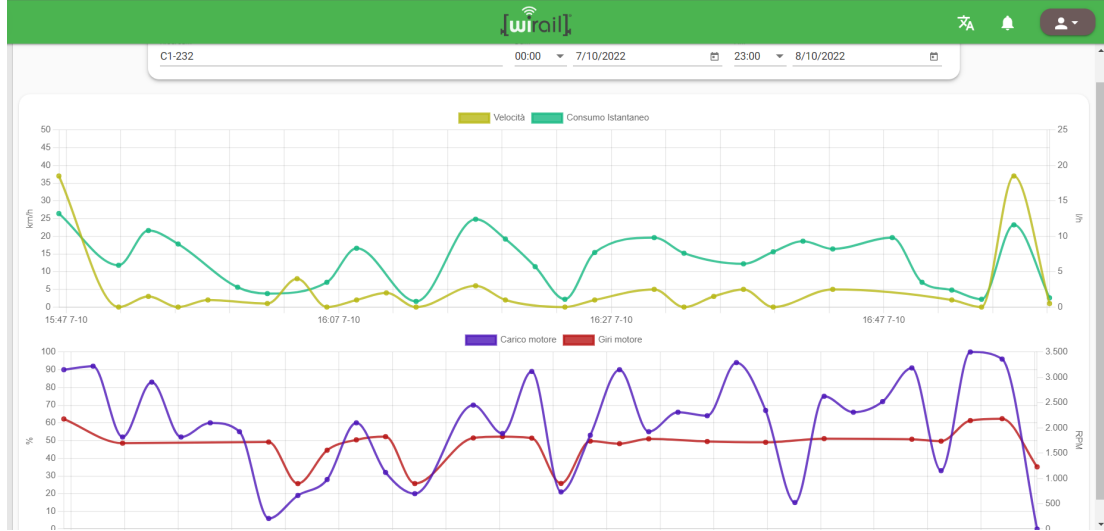


Figure 2.8: WiRail Graphics page

Since the amount of data collected by the server is high, at the time of graphing it has been decided to filter and normalize the data that are very similar to each other, otherwise the page would take a long time to load and create the graphs. In this way, if a user requests graphs with a range of one month, the data displayed will not be for every second, but for every hour.

In the case that a user does not specify anything in the search, the application will show the data of all the machines that have registered activities within the selected date range. This does not make much sense in terms of user experience, since it does not add value to the user not being able to visualize separately how their machines behave and thus make comparisons that can impact the management of agriculture.

These charts are developed with the Chart.js library, which is fully compatible with Angular and receives constant updates. This is an important starting point, since in order to update and realize new functionalities, it is necessary to use libraries that are up to date with the latest technologies.

2.4.3 Reports

Reports are one of the main tools of the application to illustrate the data collected by the control units. Currently, the data visible on each machine depends on the typology of the control unit, since as mentioned in previous paragraphs, each asset takes data depending on the assembly and customer requirements. Figures 2.9 and 2.10 show a graphical display of data from two different machines.

Data	Tipologia PLC	V. PLC	Codice Anzila	Area Full Left Sx	Area Full Right Dx	Stab. Ant. Sx a Terra	Stab. Ant. Dx a Terra	Stab. Post. Sx a Terra	Stab. Post. Dx a Terra	Configurazione trasporto	Angolo rotazione	Inclinazione braccio	Ore impianto	Maintenance Hour Left
25-01-2022 09:44	****	****	***	✗	✗	✗	✗	✗	✗	✗	0	0	0	0
25-01-2022 09:52	TCAN	2822	***	✗	✗	✗	✗	✗	✗	✗	0	0	0	0
25-01-2022 10:00	TCAN	2822	***	✓	✓	✓	✓	✓	✓	✗	-25.14	-11.6	12	22232
25-01-2022 10:08	TCAN	2822	***	✓	✓	✓	✓	✓	✓	✗	-26.98	13.69	12	22232
25-01-2022 10:17	TCAN	2822	***	✓	✓	✓	✓	✓	✓	✗	-43.1	-10.2	12	22232
25-01-2022 10:25	TCAN	2822	***	✓	✓	✓	✓	✓	✓	✗	-43.62	17.8	13	22232
25-01-2022 10:41	TCAN	2822	***	✗	✗	✗	✗	✗	✗	✓	0.1	-29.5	13	22232
25-01-2022 10:42	TCAN	2822	***	✗	✗	✗	✗	✗	✗	✓	0.1	-29.5	13	22232
25-01-2022 10:42	TCAN	2822	***	✗	✗	✗	✗	✗	✗	✓	0.1	-29.5	13	22232

Figure 2.9: WiRail Report GPSB3N310

Data	m1	m2	m3	m4	Presenza uomo	Fungo	Alternatore	Allarme alternatore
21-01-2022 07:41	10	20	50	100	✓	✗	✗	✗
21-01-2022 07:41	10	20	50	100	✓	✗	✗	✗
21-01-2022 07:41	9	19	49	99	✓	✗	✗	✗
21-01-2022 07:41	9	19	49	99	✓	✗	✗	✗
21-01-2022 07:42	9	19	49	99	✗	✗	✗	✗
21-01-2022 07:42	9	19	49	99	✓	✗	✗	✗
21-01-2022 07:42	9	19	49	99	✗	✗	✗	✗

Figure 2.10: WiRail Report Precision

Data visualization is organized by "jobs", this means that every time a machine is turned on, an activity entity is automatically created and associated to the data collected by the machine on the corresponding date,

having also the possibility to create it automatically. Then, inside the reports page the user can visualize all the data organized by the activity performed.

2.4.4 Documents

The documents page has all the bureaucratic management for the uploading of documents related to control unit procurement contracts. This page allows users to fill in their administrative data to generate all the necessary documents for signing to the contract for the use of the service. In addition, users belonging to the administrative part can modify the information entered and view historical documents.

2.4.5 Config

Within the configuration page, users are given the ability to view the full list of their control units along with details related to recent activity. Finally, users can view two tabs, one allows them to create a scheduled job instead of being created automatically by the control unit and another tab to review the maintenance status and the hours remaining for maintenance.

The functionalities available on this page depend on the role that the user has in his account, for example, an administrator of the company has access to the entire list of control units, to the list of all associated companies and to all created users, but a user who has contracted the services to control his tractor can only see the details of his machine.

2.5 Database

For any IoT system to work properly, database is essential. Therefore, having a well structured and secure database according to the necessities of the system can differentiate a good product from a precarious one.

Database on this project is organised as a non-relational database, as a consequence of the properties that define it, such as the different types of control units that can exist. Thus, it gives a possibility of an ease in producing new modifications or adding new entities.

Specifically, the database is organised in documents representing entities that are defined according the requirements of the project. Here is a list of all existing documents:

- Assets: All information related to control unit is inserted here.
- Companies: Manages information about companies that are present in the project.
- Docs: Corresponds to a backup of all bureaucratic procedures associated to the sell of a control unit.
- Jobs: Saves information about location and data of the activity executed by a control unit.
- Push: Manages information about the notification system of the project.
- Sensor-Data: All parameters gathered by the control units are inserted here.
- User-Tokens: Saves the status of the registration of a user.
- Users: Manages personal information of users affiliated to the program.

This project depends on database, whether it is to save all data collected from control units or to show important information to users. The incredibly big amount of data that a system like this collects, leads to the use of denormalized data in their database. This ensures a very high performance in read/write queries due to the minimization of the number of physical

tables that need to be accessed in order to retrieve the data [1], leading into less access to disk locations.

The database program chosen is MongoDB, since it is a source-available cross-platform document-oriented database and it is classified as a NoSql database, hence, all data is denormalized. More details about the technology are explained in chapter 2.6.1.[7]

2.6 Software Technologies

An IoT system of this caliber needs to be well integrated in all of its parts, in consequence all chosen technologies that build this project are worldwide known and used for even bigger platforms, such as Microsoft Office, PayPal, Gmail, among others.

The following chapters show the technologies that compose the system together with a general description of their roles.

2.6.1 Docker

Docker is an open-source project that automates the process of deployment of an application by using the concept of software containers. Thus, giving an extra layer of abstraction and automation for having virtual applications in multiple operating systems.

A software container is a standard unit that through packaging all the dependencies and the code itself, allows the running of an application in a quick and reliable way in different computing environments[8]. A visual representation of how a container is composed is available in figure 2.5.

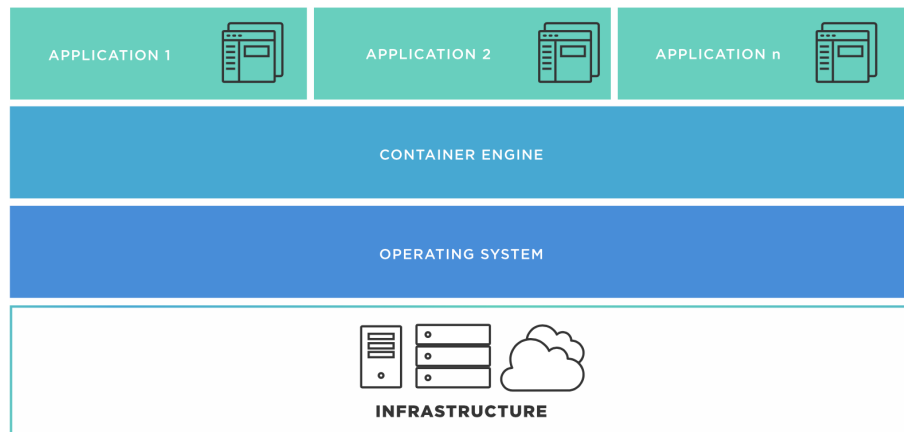


Figure 2.11: Software containers structure

Docker has been a revolution in the world of software development, since it has allowed to replace the use of virtual machines that can emulate any operating system over the native one of the server. However, the use of virtual machines means a great cost in terms of data storage and build speed. On the other hand, technologies that use containers, such as Docker, deliver good results when it comes to build speed, since they include only high-level software and are easy to modify and rebuild. Furthermore, they offer simplicity when developing an ecosystem, since there are varieties of public containers already developed for direct use in databases, messaging, among others.[9]

In order to create the containers, Docker defines a new concept called Images. These images are basically instructions or templates on how to create a container. This means that all the necessary instructions to carry out the building are present within this document. Specifically, it is made up of steps or layers that follow a logical order to, from a base image, create a real container by adding the necessary pieces.

Inside this project, Docker has been used for building all parts of the architecture, i.e, the receiver, the REST API server and the client-side application.

DockerFile

As mentioned in the previous paragraph, Docker is present in all three main components of the project. For this reason, to create the 3 corresponding images, a tool of this technology called DockerFile is used. DockerFiles are text documents that contains all the commands a user could call on the command line to assemble an image [10].

All Dockerfiles used for the receiver, API server and client-side app are available in figures 2.6, 2.7 and 2.8 respectively.

```
1 FROM openjdk:11
2 ARG DEPENDENCY=target/dependency
3 EXPOSE 8647
4 COPY ${DEPENDENCY}/BOOT-INF/lib /app/lib
5 COPY ${DEPENDENCY}/META-INF /app/META-INF
6 COPY ${DEPENDENCY}/BOOT-INF/classes /app
7 ENTRYPOINT ["java", "-cp", "app:app/lib/*", "it.wirail.receiverail.ReceiverrailApplication"]
8
```

Figure 2.12: Receiver DockerFile

```
1 # la versione slim non è compatibile con gli excel (report automatici), sono 100MB in più
2 FROM openjdk:11
3 ARG DEPENDENCY=target/dependency
4 EXPOSE 8080
5 COPY ${DEPENDENCY}/BOOT-INF/lib /app/lib
6 COPY ${DEPENDENCY}/META-INF /app/META-INF
7 COPY ${DEPENDENCY}/BOOT-INF/classes /app
8 ENTRYPOINT ["java", "-cp", "app:app/lib/*", "it.wirail.serverrail.ServerrailApplication"]
9
```

Figure 2.13: REST API Server DockerFile

```
1 FROM nginx:alpine
2 EXPOSE 80 443
3 COPY nginx.conf /etc/nginx/nginx.conf
4 COPY dist/clientrail /usr/share/nginx/html
5
```

Figure 2.14: Client-side DockerFile

2.6.2 MongoDB

As presented in previous paragraphs, the technology in charge of managing the database of this project is MongoDB. This program has an architecture

composed of 3 parts: Routers, Config Servers and Shards.

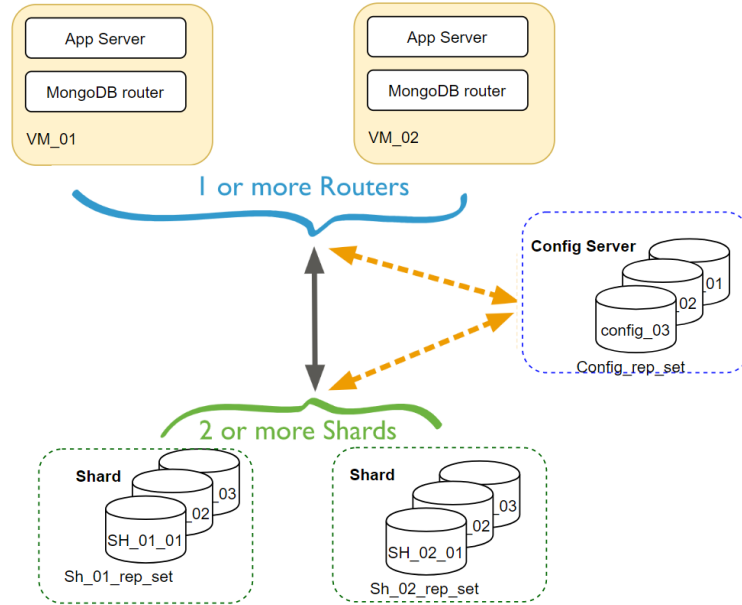


Figure 2.15: MongoDB architecture

Routers, also called *mongos*, are in charge of instantiating routes for the input queries, in addition to writing the operations to the shards within the clusters. On the other hand, Config Servers store in memory the metadata of a shared cluster, which represents the state and organization of all components and data within the cluster. Finally, Shards correspond to the data containers. From version 3.6 of MongoDB, the shards must be deployed as a replica set, in order to ensure redundancy and high availability.

For the purposes of this project, the paid service of the MongoDB company called MongoDB Atlas is used. Corresponds to a cloud service that manages databases using mongo. The structure offered by this service corresponds to a Cluster with only one Shard, a Replica Set that consists of a primary node and two secondary ones, as can be seen in figure 2.10 [11].

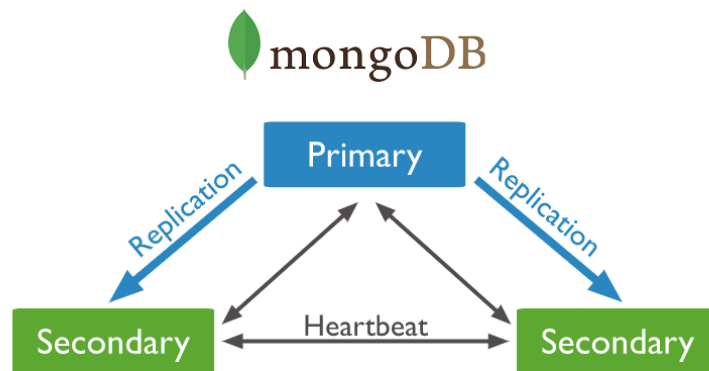


Figure 2.16: MongoDB Replica Set

The architecture previously presented allows MongoDB to offer the following characteristics:

- Scalability: It is always possible to add new shards in order to increment size of database.
- Connectivity: Thanks to developed drivers connection is almost immediate.
- Schema flexibility: There are no limitations on how to build documents, hence they can contain any type of data.
- Data redundancy: As a consequence of replica sets, data cannot be lost.

2.6.3 Angular

Angular is a specialized framework for the development of web and mobile applications mainly for companies that require a robust and progressive system. This technology was developed and is currently maintained by Google, which is why many large companies in the world use it, such as Delta, Santander banks, Paypal, Overleaf, among others. Despite the difficulty to

implement it within a system, once overcome, it can lead to very complete products.

There are many aspects in favor of using this technology, since it is created by a serious company and present all over the world, it can be expected that any problems related to the framework will be solved in a relatively immediate way. Moreover, this ensures that the updates will last for a long time, which means that a Long Term Support is guaranteed. Another advantage is the integration with the Material Design language, which is also developed by Google and is friendly to almost any user. The same applies for the components developed by Google, which allows a quick and immediate integration into the systems to be developed.

In the context of the project, one of the advantages described above is exploited, which corresponds to the integration of one of the products developed by Google: Google Maps. The role of this product within the system is essential, since it uses the geopositioning of the machines to monitor various factors that ultimately give added value to customers.

Undoubtedly, Angular corresponds to a complete framework for developing a serious and robust application, which has a high learning cost but with very rewarding performance rewards.

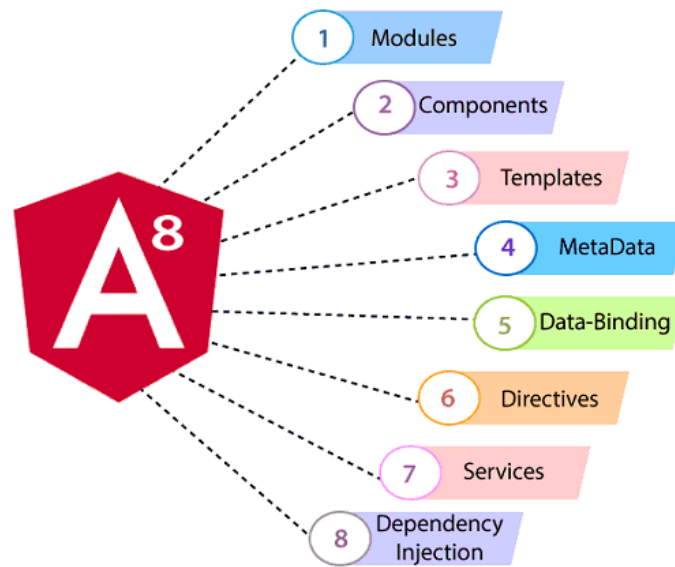


Figure 2.17: Angular 8 Architecture

2.6.4 Spring Boot

Spring Boot is an open-source micro framework created by Pivotal. This technology provides a platform for Java developers to start developing a self-configurable, production-grade Spring application.

There are other Spring frameworks available in the market, these offer dependency injection functionality that allows objects to define their own dependencies, which once ready the Spring container will inject into them. This is useful when developing applications consisting of isolated components ideal for microservices or distributed network applications. However, these frameworks require significant amounts of time and knowledge to configure, setup and deploy applications. This is why Spring Boot comes into play, as it mitigates all these efforts and reduces them thanks to 3 important capabilities:

Autoconfiguration

Autoconfiguration means that when an application is initialized, it already has pre-configured dependencies that do not need to be configured manually. This is a consequence of the fact that Spring Boot comes with built-in autoconfiguration capabilities that will generate the packages depending on the imposed settings. One of the main advantages of this capability is the high speed of spring-base application development and the reduction of human errors.

Opinionated approach

This capability corresponds to the use of an opinionated approach to add initial dependencies based on the needs of the project. Therefore, following its own judgment, Spring Boot decides for us which packages to install and which values to use. The needs of the project can be defined during the initialization process, where the process of choosing between multiple starter dependencies called spring starters must be carried out.

Standalone applications

Finally, this technology helps developers to create applications that simply run, allowing the creation of software that do not need an external web server, by combining with a server such as Tomcat or Netty in the initial phase [12].

In the context of the project, this technology was one of the important factors for the initial development of the receiver and server at code level, since being a relatively small development team, it made the work easier in terms of time and human efforts. However, this advantage is not the only reason to use Spring Boot within the system, in addition to the above, it has a facility to integrate with databases and modules that are of great help to

complete all possible axes of a server. The main modules made available are:

- Spring Boot Starter Web: This submodule allows the construction of a web application with little configuration compared to other alternatives;
- Spring Boot Starter Security: Corresponds to the module in charge of security. It is characterized by reinforcing all the restrictive policies of the created endpoints, thus creating a configuration that concentrates everything in a single endpoint;
- Spring Boot Starter MongoDB: Spring provides drivers already created and complete for the integration of databases such as Mongo within the applications built, again requiring configuration. It also takes care of the mapping of java objects, allowing database operations to be only function calls. However, this module works only for simple operations, in the case of wanting to make more complete calls, it must be integrated with other tools such as aggregations;
- Spring Boot Starter Validation: Allows the automatic management of input validation operations within the REST API server;
- Spring Retry: Provides the ability to automatically re-invoke a failed operation, allowing the application to continue running despite errors that may be transitive or momentary;
- Spring Boot Starter Test: Specialized module to create necessary tests within the created application;

2.6.5 Kibana and ElasticSearch

The importance of logs in a company that manages tons of data becomes paramount. Whether it is to detect common mistakes made by users, developers, bugs in code or just for security purposes. For this reason, the system

has its own logging technique that connects to Elasticsearch and Kibana, allowing the developing team to do specific queries to check at logs and find errors.

2.6.6 Digital Ocean

Digital Ocean is an American provider of private virtual servers used to deploy the project into the cloud. This technology uses the concept of virtual machines (VMs), specifically, Linux-based VMs that run on the Digital Ocean hardware.

Currently, this project is using 3 instances of virtual servers:

- **Maia:** This server has the purpose of containing stage developments of the 3 main components: receiver, REST API server and frontend.
- **Elk:** The Elk server contains mainly the frontend and the REST API Server.
- **Theo:** Its purpose its deploying the receiver with all the necessary parameters to accomplish its purpose.

2.7 Programming Languages

Within the present project there are two main programming languages, whose choice was strongly linked to the technologies described above due to compatibility. The programming languages used are Java and Typescript. The former is used for the components associated with the receiver and the REST API, and the latter for the frontend integration with Angular. The following is a brief description of the mentioned languages and their uses in today's software development world.

2.7.1 Java

Java is a high-level, class-based, object-oriented programming language that is designed to have as few implementation dependencies as possible [13]. It is a general-purpose programming language, which allows to build applications for various platforms without the need to recompile them, such as televisions, tablets, cell phones, computers, among others. This language adheres to the WORA requirements, which stands for Write Once Run Anywhere.

The syntax of this language is very similar to that of C and C++, but has less low-level qualities than both. However, it is highly used for developments that provide dynamic capabilities that are usually not available in low-level languages. For the purposes of this project, this programming language has been used to develop the receiver and the Rest API, due to its compatibility with Spring Boot.

2.7.2 Typescript

TypeScript is a free and open source programming language developed and maintained by Microsoft [14]. It corresponds to a somewhat stricter evolution of Javascript, which adds strict scripting to the language. Being an evolution of Javascript, programs written with Javascripts can be valid within a Typescript program.

This language can be used to create both client-side and server-side applications, however for the purposes of this project it has been used only to develop the frontend side of the system.

The advantage of using TypeScript, is that you can define files dedicated to the variable type information of existing javascript libraries, as C++ header files do when describing the structure of objects. This allows easier recognition of all functions and values defined in the files.

2.8 Performance

To better understand the capabilities of the current infrastructure to support the project, it is important to evaluate and analyze the performance of the architecture with the current number of users, control units and data traffic.

Currently, the client-side or frontend together with the REST API server are hosted on a machine with 4 GB RAM, 2 vCPU and 80 GB disk memory. This server is physically located in Frankfurt. The features of this server make sense as the amount of users and, therefore, API requests is increasing, due to the growing market.



Figure 2.18: CPU usage of frontend and REST API

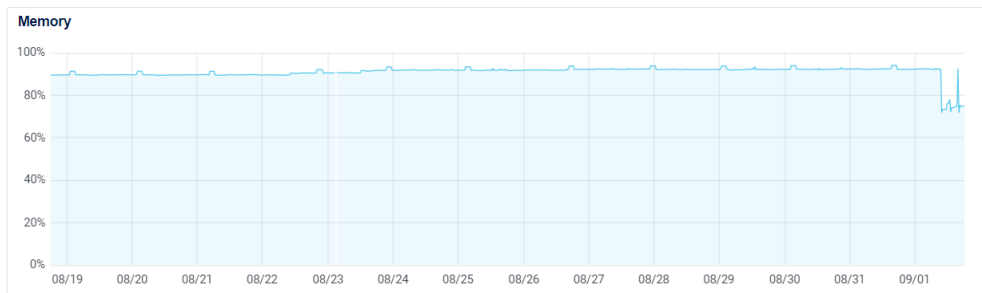


Figure 2.19: Memory usage of frontend and REST API

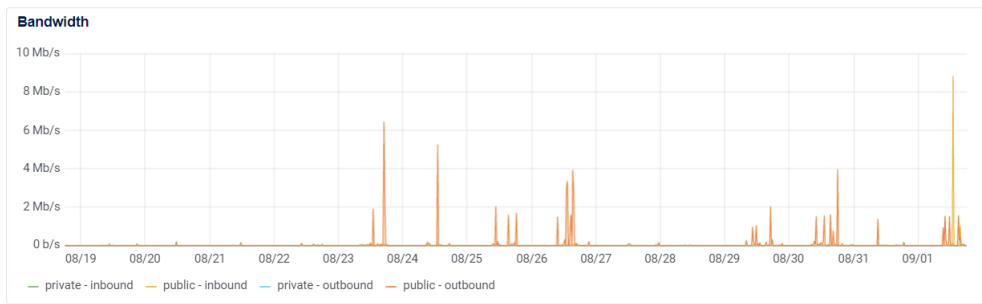


Figure 2.20: Bandwidth of frontend and REST API

On the other hand, the receiver is hosted by a machine with 2 GB of RAM, 2vCPU and 60 GB of disk memory.

The company has more than 600 registered control units, however the maximum number of simultaneous connections never exceeds 200. For this reason, the graphs shown correspond to 170 simultaneous connections for the receiver and about 20 - 30 user connections to the site per day.

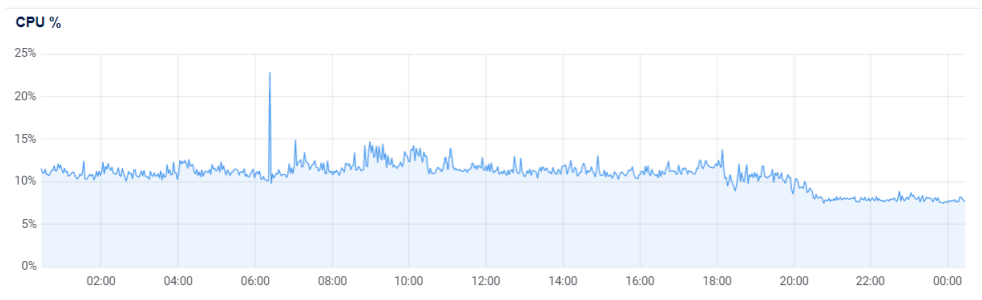


Figure 2.21: CPU usage of receiver

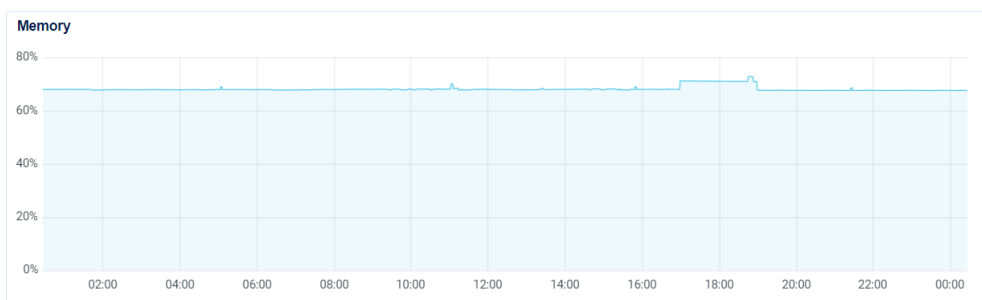


Figure 2.22: Memory usage of receiver

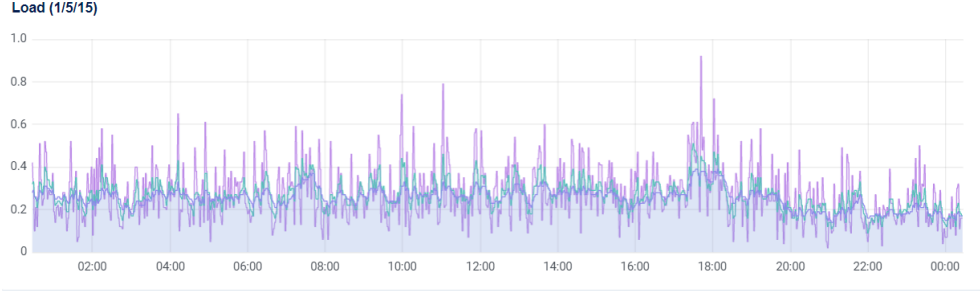


Figure 2.23: Load (1|5|15) of receiver

The database on the other hand, is hosted in Azure through MongoDB Atlas. The configuration of this machine is similar to that of the receiver with 2 GB of RAM, 2 vCPUs and a disk memory usage of 10 GB. At the moment, the memory used by the database is approximately 3.1 GB as can be seen in the graph of figure 2.24. In this case, the machine is underutilized, since on average and at most only 4% of the CPU is used.



Figure 2.24: Database storage usage

From the graphs we can see that the server and client-side is at a regular performance, since the CPU does not take values higher than 50%, however, the memory usage remains constant at a percentage close to 90%. This means that an upgrade should be performed in the near future to meet the demands of the requests.

Finally, the receiver is in a process of growth due to the increasing number of existing control units, however, for the purposes of the current architecture, the only parameter that is at risk would be Memory and Load, since Load is very close to being 1, so the queue is filling up and there is a risk of exceeding the threshold that slows down the services.

Chapter 3

Preventive Software Maintenance

In this chapter all the work done prior to the development of the new functionalities will be presented.

Software maintenance is the process of modifying and updating the software according to the customer needs or changes inside the system where the program is based. This process has several objectives such as correcting bugs or issues, to improve overall performance or to upgrade the user experience while using the software.

3.1 Initial situation

One of the initial challenges on this thesis was to study the existing application and understand the distribution and function of the components present, with the objective of performing a panoramic analysis on the operation and interaction with the user. Once the analysis was done, it was concluded that the application had a regular functioning with the following problems:

- Data integrity issues: There are problems with data integrity in the sense that the variation of one or more characters can result in application crashes or delivery of incorrect results, as for example in the documents section, where there can be two entities with the same ID but with different upper and lower case letters.
- Outdated components: Within some components that depended on external libraries, problems were found with functionality that had been deprecated due to updates and were still in use, which can lead to malfunction and loss of data.
- Verification Problems: There were some forms that could be sent empty, which generated confusion on users and administrators. Moreover, there were not verification of data types, so if a field was supposed to be a number the user could write a phrase with alphabetic characters. There was not enough verification to allow sending requests with parameters that were mandatory but empty, such as in the documents section.
- Malfunctioning API endpoints: Some endpoints in charge of delivering information to the client-side to visualize some views were not working correctly. This happened because some of the internal structures of the sensor data were modified without updating the services in charge of returning them, so it generated an internal error in the application that prevented to visualize, for example, the graphs.

After the generated analysis, most of the bugs and errors were individually identified through manual integration testing in order to mitigate them. Once identified, the code of the parts involved was modified to correct the operation. As an example, we take two specific cases, the first one corresponds to the error shown in figure 3.1, whose reason was an error in the API service due to the fact that the method was not updated while the data structure to which it queried changed. The second one, corresponds to the possibility of sending "forms" of invalid documents to the database, and that the server processes them to be accepted even though they are empty or erroneous. For this specific case, the client-side component was modified to perform verification using Angular technology. In figures 3.3 and 3.4 is present the modified code that solved this issue.

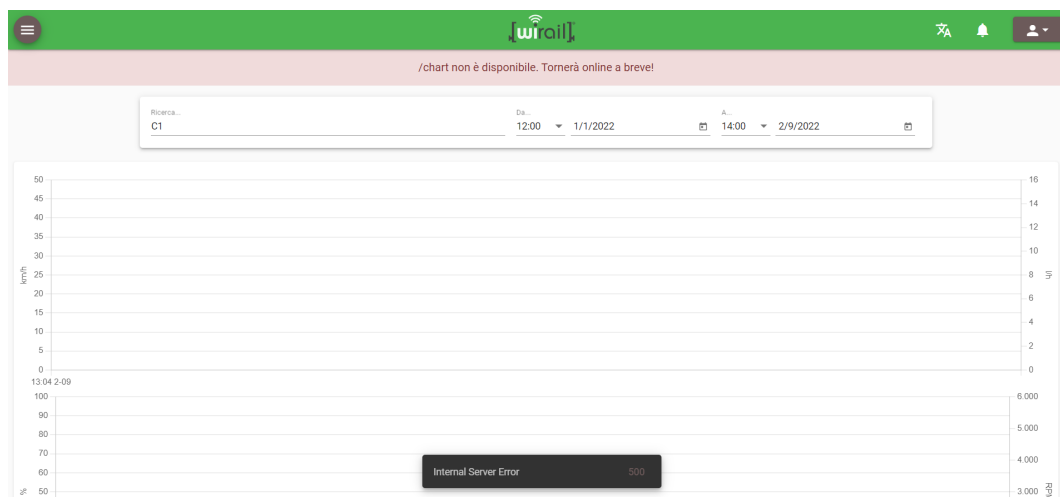


Figure 3.1: Graphics page error

It was possible to obtain a good starting point for the new interfaces after the identified problems were solved, since this allowed to attack every weak point of the project and repair them in order to get a more solid basement.

The screenshot displays a web application interface with two main sections. The left section, titled '1 Compila il form', contains a form with several input fields: 'S/N Centralina', 'Marca (*)' (with 'WiRail' entered), 'Modello (*)', and a row with 'Telaio (*)', 'Seriale', and 'Targa'. Below these are fields for 'Ragione Sociale (*)', 'Partita IVA (*)', 'Codice Fiscale', 'Sede Legale (*)', 'Città (*)', 'CAP (*)', and 'Provincia'. The right section, titled '2 Scarica e Firma', lists download links for 'Autocertificazione', 'Caratteristiche tecniche abilitanti', and 'Scheda cliente'. A green 'Salva' button is present, and a red box highlights a confirmation message 'Form salvato correttamente!'. Below this, a red box highlights a header row in a table with a pencil icon and a user icon. The table lists four documents with IDs A4-00001 through A4-00004, each with a document icon, a user icon, and a delete 'X' button.

Icona	Nome Documento	Id Documento	Operazione
		A4-00001	X
		A4-00002	X
		A4-00003	X
		A4-00004	X

Figure 3.2: Empty Documents saved correctly

```
docFormInit() {
  this.docForm = this.formBuilder.group({
    id: ['', Validators.required],
    ragioneSociale: ['', Validators.required],
    pIva: ['', Validators.required],
    cFiscale: '',
    sedeLegale: ['', Validators.required],
    codiceDestinatario: '',
    city: ['', Validators.required],
    cap: ['', Validators.required],
    prov: ['', Validators.required],
    legaleRappresentante: ['', Validators.required],
    cellLR: '',
    mailLR: ['', [Validators.email]],
    legaleAmministrativo: '',
    cellLA: '',
    mailLA: ['', [Validators.email]],
    referenteTecnico: ['', Validators.required],
    cellRT: ['', Validators.required],
    mailRT: ['', [Validators.required, Validators.email]],
    marca: [{
      value: !isMultimarca(this.authService.currentUserValue) ?
        this.authService.currentUserValue.companyID : '',
      disabled: !isWiRail(this.authService.currentUserValue) &&
        !isMultimarca(this.authService.currentUserValue)
    }, Validators.required],
    serviceCompany: this.authService.currentUserValue.serviceCompany,
    modello: ['', Validators.required],
    telaio: ['', Validators.required],
    seriale: '',
    targa: '',
    emailAccount: ['', [Validators.required, Validators.email]],
    emailReport: ['', [Validators.required, Validators.email]],
    dataAttivazione: ['', Validators.required],
    dataCompilazione: [formatDate(new Date(), 'dd/MM/YYYY', this.locale)],
    imei: ''
  });
}
```

Figure 3.3: Validation code introduced for Documents

```

this.subs.sink = this.saveForm$.pipe(throttleTime(600)).subscribe(() => {
  this.saveFormSucceeded = false;
  this.saveFormMsg = '';
  this.saveFormMsgProgress = true;

  if (this.docForm.valid) {
    this.docService.create(this.schedaCliente,
      this.autocertificazione,
      this.schedaCollaudo,
      this.ddt,
      this.docForm.getRawValue()).subscribe(() => {
        this.saveFormMsg = 'Form salvato correttamente!';
        this.saveFormSucceeded = true;
        this.update.emit();
      }, error => {
        this.saveFormMsg = 'Errore nel salvataggio. Riprovare!';
      }, () => this.saveFormMsgProgress = false);
  } else {
    /*TODO: Use $localize */
    var message: string = this.locale === "it" ? "Form incompleto, compilare tutti i campi per procedere al salvataggio." :
      "Incomplete form, fill in all fields to save." ;
    var error: string = this.locale === "it" ? "Documenti non salvati":
      "Document compilation error" ;

    for (var i in this.docForm.controls) {
      this.docForm.controls[i].markAsTouched();
    }
    this.saveFormMsg = 'Errore nel salvataggio. Riprovare!';
    this.errorDialog.open(DialogErrorComponent, {
      data: {error: error, message: message, color:"warn"},
      minWidth: '25vw'
    })
    this.saveFormSucceeded = false;
  }
})

```

Figure 3.4: Validation added to "Save Document" function

3.2 User stories analysis

To better understand the initial situation of the application, an analysis of the user stories -a tool used in software engineering to describe a functionality from the eyes of a final users- was performed. A user story is composed as follows: "As a **role**, I want to **action** do that **benefit**". Then, this same method will be used at the end to describe the newly added functionality.

Within this project there are 5 different roles: System Administrator, WiRail Administrator, Company Administrator, Service Company user and Client Company user. In this context, user cases were made for the last 3 user types, since administrative roles have access to all functionalities. Below are 3 tables with the user cases, a column to check if it was implemented and another one to define if the operation is correct or incorrect due to failures

or bugs.

Table 3.1: Company Administrator user stories

User story	Implemented	Correctly Functioning
As a company administrator I want to check all the control units	Yes	Yes
As a company administrator I want to know the position of my control units	Yes	Yes
As a company administrator I want to check all documents related to control units	Yes	Yes
As a company administrator I want to see the position of my control units	Yes	Yes
As a company administrator I want to see graphed information gathered by sensors so that I can check all machines' performance	Yes	No

Table 3.2: Service Company user stories

User story	Implemented	Correctly Functioning
As a service company user I want to complete all bureaucratic documentation	Yes	No
As a service company user I want to know the position of my client's control units	Yes	Yes
As a service company user I want to create accounts for my clients	Yes	Yes

Although not all user stories are presented in the three tables above, it can be seen that there are indeed problems with the operation of some of the functionalities, causing them to not be able to enjoy the full potential of

Table 3.3: Client Company user stories

User story	Implemented	Correctly Functioning
As a client company user I want to program an activity for my machine	Yes	No
As a client company user I want to download the reports of my machine's activity	Yes	Yes
As a client company user I want to receive a monthly report mail of my machine's activity	Yes	Yes
As a client company user I want to lock my control unit	Yes	Yes

a platform of this kind and causing the user to have a bad experience when using the application.

Chapter 4

Smart Farming

The following chapter will make an outline of the whole process of design and development of the project, taking into consideration all the steps performed that allowed to reach the final result and product. In particular, it is intended to show all the software development carried out with the technologies described above together with all its stages, i.e. the initial requirements analysis, the limits of the approach during the development of the tools and the successive operational modality.

The main focus of the work carried out always corresponds to the application of tools that facilitate the application of smart farming in the company, delivering new value propositions to the final customers that improve the management and control of their machines.

4.1 Project approach

The initial approach to a new project tends to be impulsive and full of ideas that sometimes exceed its limits, especially in areas of software engineering where there is a wide range of development possibilities. For this reason, it is important to establish a line of organization of the work to be done,

establishing clear and achievable objectives considering the own resources and limits, putting into practice the multiple strategies that exist for the development and design of information technologies.

In this line, it was decided to opt for existing strategies that guide and define the development processes, such as agile methodologies. These methodologies allow the development of new functionalities to be more flexible, accepting changes along the way depending on the needs of the users and the team itself. At the same time, the new tools created are released consistently, with the possibility of improvements and response to possible changes in a fast way. However, due to the characteristics of the team and the company, some applications of the strategy had to be adapted.

Among the principles applied to carry out the project, the following were considered:

- Software always in operation before comprehensive documentation
- Collaboration with the customer rather than contractual negotiations
- Responding to change rather than sticking to the plan.

Following these principles, the project was carried out by holding brief meetings at the beginning of each week to establish the weekly objectives, which could be subject to change depending on the client's needs. Thus, tools such as Trello were used to organize the activities to be developed, producing a Backlog of things to do, describing the objective of each activity and its priority; in addition, the Issues section available in Github were used to register every time an error was found or a change was required from the users, defining priorities per task and responsibilities.

4.2 User Stories for new features

In order to better organize the further development and to document the new features, it was decided to perform an analysis of the new user requirements through User Stories as was done in the previous chapter. Thus, the stories were divided by type of user to obtain the following definition:

Table 4.1: Client Company and Service Company user stories

User story
As a client/service company user I want to check the current activities of my machines
As a client/service company user I want to check the planned activities of my machines
As a client/service company user I want to check the completed activities of my machines
As a client/service company user I want to see details of my machine's current activity
As a client/service company user I want to write notes on the creation of my activities
As a client/service company user I want to delete a planned activity
As a client/service company user I want to see reports of a completed activity
As a client/service company user I want to confirm a current activity so that I can acknowledge the job
As a client/service company user I want to confirm a completed activity so that I can acknowledge the job
As a client/service company user I want to draw a field on the map of my work area
As a client/service company user I want to associate my working area to my activity
As a client/service company user I want to search activities based on parameters of my machines
As a client/service company user I want to identify my field with a name
As a client/service company user I want to browse my fields and see details and location

Table 4.2: Company Administrator user stories

User story
As a company administrator user I want to browse all current activities of all machines under my company
As a company administrator user I want to browse all planned activities of all machines under my company
As a company administrator user I want to browse all completed activities of all machines under my company
As a company administrator user I want to browse all fields created by service users and clients
As a company administrator user I want to delete a field of one of my service users or clients
As a company administrator user I want to see technical details of the machines under my company
As a company administrator user I want to see reports of my machines' implements

As can be seen in tables 4.1, 4.2 and 4.3, the user stories are developed in 3 main elements: activities, fields and farm implements. Likewise, these stories are useful to guide the line of work and to achieve a better organization of the time available by concentrating on these 3 elements and developing functionalities that meet the user's expectations, always considering variables such as security, good programming and code writing practices, application integrity and performance.

Although user stories superficially define an end goal on the part of the users, they do not describe all the additional work that must be done to meet these requirements. For example, when modifying an entity used to represent activities, all new services must be modified and created to meet the end goal.

Table 4.3: WiRail Administrators user stories

User story
As a WiRail administrator user I want to add a current activity for all machines under all companies
As a WiRail administrator user I want to browse all current activities of all machines under all companies
As a WiRail administrator user I want to plan an activity for all machines under all companies
As a WiRail administrator user I want to browse all planned activities of all machines under all companies
As a WiRail administrator user I want to browse all completed activities of all machines under all companies
As a WiRail administrator user I want to confirm activities for all machines under all companies
As a WiRail administrator user I want to browse all fields created
As a WiRail administrator user I want to delete a field
As a WiRail administrator user I want to see technical details of the machines under all companies
As a WiRail administrator user I want to add farm implements to machines of all companies
As a WiRail administrator user I want to specify measure parameters for implements
As a WiRail administrator user I want to see reports of implements already on fields

4.3 Activities as center of management

The activities, or in its original language "*commesse*", correspond to the system unit that manages all the jobs performed by the control units. The presence of these entities is absolutely primordial, since they allow the connection between the control unit and the web site to work, in order to record all the data collected during the activity's duration. This is because the first action that a control unit performs is to search for an activity to connect to, and therefore, without the existence of a "*commessa*" it would

not be possible to save the data in the database. In addition, these allow to establish certain characteristics to the jobs, such as the job's description, registering the associated responsible, the place where the activity occurs, the date when it is carried out, among others, and it also allows the association with all the parameters that are collected by the site.

Currently, there are two types of activity, the first one is the automatic activity and the second one is the manual activity. More details about both types are presented below.

- **Automatic Activities:** The automatic activities correspond to those jobs in which the user does not make a record of the activity through the service prior to the connection or switching on of the control unit. As mentioned in the previous paragraph, it is mandatory that there is a current activity for the correct operation and connection of the control unit with the server, this is because at the connection time with the server, it is consulted if there is an activity with a start date less and end date greater than the current time. If this does not happen, an activity is automatically created, whose duration is from the moment of connection of the device until 23:39 of the same day. Thus, each data that the control unit registers within this data range is associated with the automatically created job.
- **Manual Activities:** Manual activities are all those registered through the web site, prior to switching on the control unit and connecting to the network. These can be entered thanks to the provision of a "New activity" functionality within the application. Consequently, at the moment of connection, if it is turned on within the range specified in the activity, the control unit associates it directly by displaying the information on the screen and collecting the data within the job.

4.3.1 Previous Activity Management

In the version prior to the intervention of the application, the management of the activities or "commessa" took place inside the configuration tab of the control unit. In this place, under the asset details there was a tab called "Commessa corrente" to visualize the current job or, if one was not present, the empty fields to enter an activity manually. As can be seen in figure 4.1, this screen has two different functionalities. The first of these functionalities corresponds to a green button with a "+" sign, which cancels all the fields presented in the tab and replaces the current activity -if present-. For this reason, a user after entering his manual activity, would press this button thinking that it was used to save, thus canceling all the data entered. The second functionality corresponds to a "Reset to previous job", represented by a simple arrow that, in case a previous activity exists, resets all the form fields with the activity details.

Figure 4.1: Manual activity configuration

The previously described functionalities caused confusion within users of all types, including the administrator users belonging to the company, since

it had a strange and unfriendly behavior with the user. Furthermore, there were problems when saving changes in the activity, as there were bugs and errors in the code related to the activities. Among these confusions, there was also the fact that if a current activity was entered for a time after the current time, the system did not show it and there was no way to modify it, which generated the feeling that "the application does not work well". The same thing happened if the end time was changed to an hour earlier, there was no way to verify if the change was actually made or neither modify it.

The above described and the analysis made at the beginning, led to the conclusion that greater importance should be given to the management of activities, since they are the basis that connects all the structures present in the system, whether it is in reports, in the configuration of the control units, in the graphs and in the maps. For this reason, it was not sustainable to have a non-functional and non user-friendly service to manage this issue. Thus, it was decided to reorganize the whole structure and to create a system for the management of the activities, which would add value to the users and facilitate the organization of their work. The following sub-chapters present in detail the development of what is now called "Activities", which corresponds to a new option within the main menu and includes a series of tools to manage the work of the final users.

4.3.2 Current Activities

The first section of the "Activities" tab corresponds to "Current". Within this option are all those activities that occur within a period that includes the current time, i.e. if we represent it with a mathematical language we can define the following: $Start\ Date\ Activity \leq Actual\ Hour \leq End\ Date\ Activity$.

A new view that continues with the general design of the application was developed, making available a screen divided in two, where on the left side there are three main elements: a search engine, a list of activities and a

button to create a new activity. Regarding the search engine, it is a tool that finds the activities from an input entered by the user, which can correspond to any characteristic related to the job, whether it is the identifier, the company, the number plate of the control unit, among others. Then, the list of activities is designed in such a way that it is pleasing to the eye and user friendly, thus presenting a clickable rectangle with rounded edges for each activity, where the machine's number plate, the identification of the activity, start date, end date and a button to acknowledge the activity are displayed. This last information can be updated either from the site or directly from the control panel, the information will always be displayed in real time.

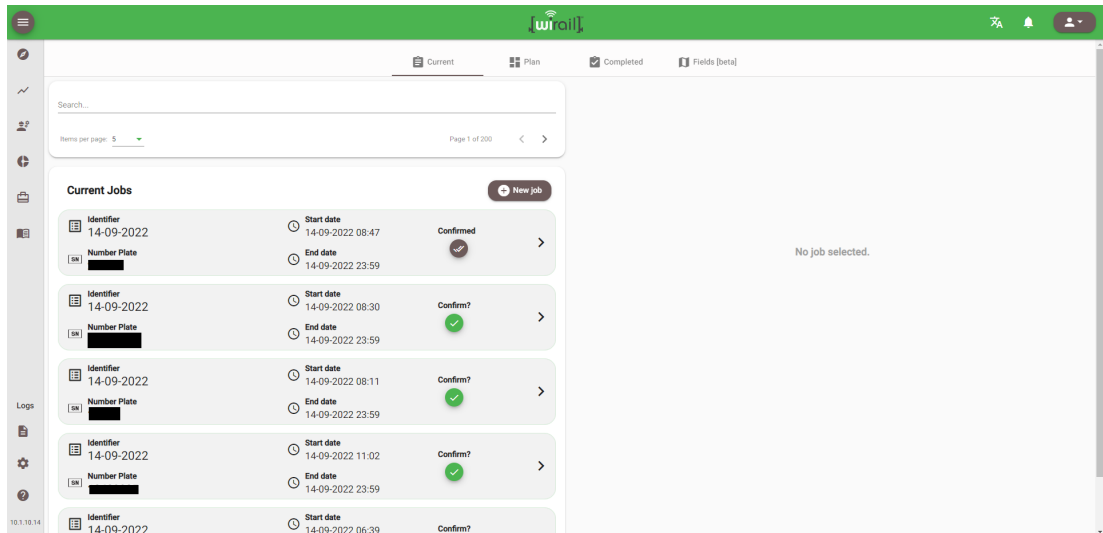


Figure 4.2: Current Activities

The "new job" button offers the possibility to create a new activity with a friendly and easy to use design. Hence, a modal opens covering half of the screen, where the user must select a control unit within the list in order to complete the rest of the data: identification, start date, end date, city, address, notes and speed if it corresponds to a specific type of control unit. If the start date is later than the current time, the activity would not correspond to a current one, so it is redirected to the planned activities tab, which will

be discussed in the next sub-chapter, but if it does correspond to a current activity, the newly created activity is saved and displayed immediately.

The screenshot shows the 'Crea Attività' (Create Activity) form in the [wi]rail application. The form is overlaid on a dashboard with tabs for 'Current', 'Plan', 'Completed', and 'Fields [beta]'. The form contains the following sections:

- Select Asset:** A dropdown menu with a green checkmark.
- Dettagli:** A section with a green dot icon and a tractor image. It contains the following information:
 - Client Company: Test Client Company
 - Service Company: Test Service Company
 - Total Hours: 39 hrs.
 - Phone Number: phone
- Form Fields:**
 - Identifier: A text input field.
 - Speed: A numeric input field with a green checkmark and a '0' value.
 - ☐ Usare campi: A checkbox.
 - City: A text input field.
 - Address: A text input field.
 - Start time: A time input field with '16:00'.
 - Start date: A date input field with '9/14/2022'.
 - End time: A time input field with '00:00'.
 - End date: A date input field with '9/15/2022'.

Figure 4.3: New Activity

When selecting a specific activity, the right part of the screen will be activated showing a card element with all the information separated by section offering also the possibility to edit the current activity. As seen in the figures, the first section offers a panoramic detail of the machine where the control unit is installed, i.e. plate number, client company, service company, the total working hours of the machine, the telephone number of the person in charge and if available, a picture of the machine. Moreover, after the details, the activity section is available, where the user can see more or less the same variables as when creating a new job, such as identification, dates,

address and a section of notes. In addition, the hours of planned work for the activity are presented together with a checkbox to show the current position of the job in case it corresponds to a manually entered activity, whereas in the case of an automatic one, the last position delivered by the machine is taken. Finally, if a change has been made, a button will be activated to update the activity and its components.

The screenshot shows the 'Current job' details page in the Wirail application. The page is divided into several sections:

- Header:** Wirail logo, navigation icons, and a user profile icon.
- Plan:** Tabs for 'Plan', 'Completed', and 'Fields [beta]'.
- Current job:** A section with a briefcase icon and a green status dot.
- Details:** A section with a tractor icon and the following information:
 - Number Plate: 16908614
 - Client Company: Client Company Test
 - Service Company: Service Company Test
 - Total Hours: 34 hrs.
 - Phone Number: phone
- Activity:** A section with a calendar icon and a clock icon. It shows 'Planned Activity Hours: 15 hrs.' and a 'Show map' checkbox.
- Form Fields:**
 - Identifier: 14-09-2022
 - City: city
 - Address: address
 - Start time: 08:00
 - Start date: 9/14/2022
 - End time: 23:00
 - End date: 9/14/2022
 - Notes: (empty text area)
- Update:** A large button at the bottom to update the activity.

Figure 4.4: Activity Details

allow users to adapt more quickly to new changes.

As in the previous section, the screen is divided in two, where the left side has a list of all planned activities, a button to create new activities and a search engine. These elements have the same characteristics as those present in the Current tab. On the other hand, on the right side there is the same card with the details of the machine, the activity and the position where it will be carried out. It gives the possibility to edit the job with all its parameters, and it adds an extra functionality that corresponds to the deletion of the planned activity in case of cancellation of the working activity.

Commessa painificata

Dettagli

Targa: C1-534
Azienda Cliente: _____
Service: _____
Ore Totali: 0 hrs.
Numero di telefono: phone

Attività

Ore Attività Previste: 11 hrs.

☒ Mostra mappa

Identificativo
test-pablo

Città
▲ Torino

Indirizzo
Via Marco Polo 24

Ora inizio
11:00

Data inizio
▼ 21/9/2022

Ora fine
22:00

Data fine
▼ 21/9/2022

Note

Posizione di attività

Figure 4.6: Delete planned activity

4.3.4 Completed Activities

Finally, the third section of the "Activities" tab corresponds to "Completed", which as the name indicates, contains all the activities already completed within a period of time. Within the "Completed Activity" category are all those activities that fulfill the following relation: *End Date Activity < Actual Hour*. However, in order to request the information, a date limit is requested within the search so as not to overload the server, so that all completed activities are delivered within the defined range.

This new tool allows users to keep a history of all activities performed in order to access relevant information such as the job site, machine details at the time of the activity and direct redirection to the associated reports, continuing with the same graphical style of the other sections.

As in the previous two sections, a two-part design is implemented, where on the left side there is a list of all the activities that meet the search criteria entered in the search engine, and on the right side the complete detail of the activity along with the possibility to view the associated reports. In addition, the user is allowed to confirm the activity even if it is already finished, as some customers perform this procedure after the jobs have been completed.











Ricerca...		Seleziona un range di date 1/7/2022 – 8/7/2022	
Elementi per pagina: 5		Pagina 1 di 200	
Commesse Completate			
Ore previste di lavoro: 69 ore 24 minuti			
Identificativo 05-07-2022 Targa 	Data inizio 05-07-2022 08:23 Data fine 05-07-2022 23:59	Confermata 	>
Identificativo 07-07-2022 Targa 	Data inizio 07-07-2022 08:11 Data fine 07-07-2022 23:59	Non Confermata Confermare? 	>
Identificativo 01-07-2022 Targa 	Data inizio 01-07-2022 15:20 Data fine 01-07-2022 23:59	Non Confermata Confermare? 	>
Identificativo 04-07-2022 Targa 	Data inizio 04-07-2022 07:58 Data fine 04-07-2022 23:59	Non Confermata Confermare? 	>

Figure 4.7: Completed activities list

Commissa completata


Dettagli

Targa: 
 Azienda Cliente: Client Company Test
 Service: Service Company Test
 Ore Totali: 112 hrs.
 Numero di telefono: phone

Attività

Ore Attività Previste: 15.6 hrs.

☒ Mostra mappa

Identificativo
 05-07-2022

Città city	Indirizzo address
Ora inizio 08:00	Data inizio 5/7/2022
Ora fine 23:00	Data fine 5/7/2022

Note


 Posizione di attività

Figure 4.8: Completed activities details

4.4 Geofencing

As the name says, Geofencing is a technology based on establishing a geofence, which basically corresponds to a virtual fence on a real map. This can be described as an invisible perimeter that delimits a real geographic area in any possible way and shape. Then, this area is exploited to take advantage of services that use geolocation through GPS, such as businesses, markets, restaurants, municipalities, the government and in our case, precision and telemetry projects. From what the market currently offers, there is a great need to implement functionalities such as these, as they are gradually becoming the standard for telemetry companies.



Figure 4.9: Geofence representation

4.4.1 Fields and Work Areas

The integration of geofencing in the project is done through a new tool made available in the "Acitivity" tab. It allows the final users to design their own fields or work areas on a map runned by google. This tool called "drawing" was modified and adapted to the application to have a harmonious design and operation, also more functionalities were added through "event listeners", that is actions that are triggered by the occurrence of a specific event. Specifically, event listeners were added at the moment of clicking on

the map and at the moment of finishing a representative polygon inside the map.

To accomplish this goal, a new button called "New Field" was made available that activates the functionality to draw on the map and displays a couple of key elements to tell the user what to do. As shown in Figure 4.11, a small GIF is provided to indicate how to use the tool and a banner to indicate that the "draw your new field" mode is active. When the new "draw" mode is activated, the user can start drawing on the map by creating points, so that the system takes the longitude and latitude of each one and saves them in an array when closing the figure, that is when the last point matches the first point created.

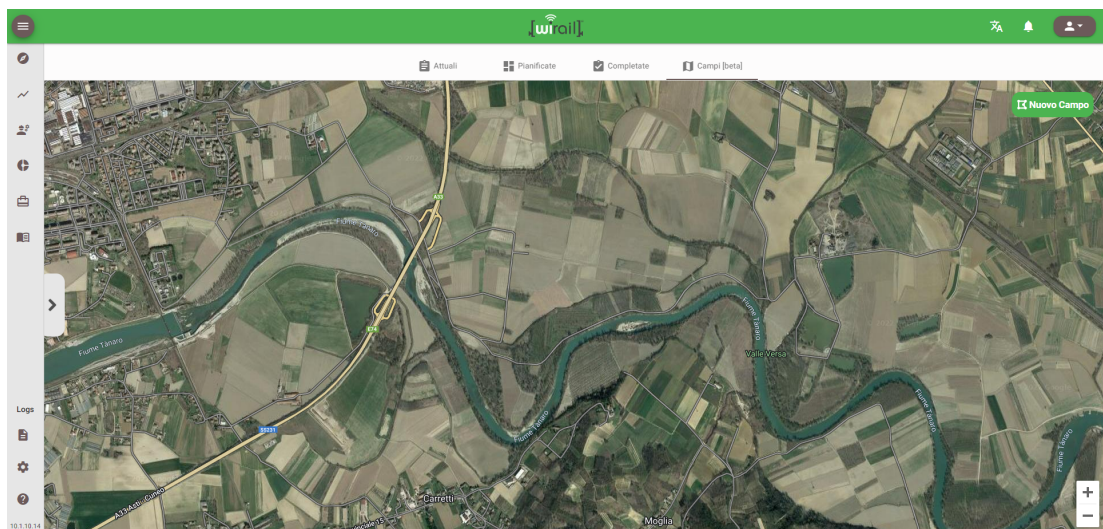


Figure 4.10: New field

Then, a new dialog or pop-up window automatically opens to enter the field identifier or name, which is unique for each client, along with a short description or notes about the field characteristics. Notes are not mandatory but highly recommended to better identify and describe the work to be done in the field and make it available on future reports.



Figure 4.11: Create new field dialog

4.4.2 List of Fields

As written in the user stories of previous chapters, it is important to keep a record of all the fields created by the users, since each client may have different amounts of lands where work is done. Consequently, a list of all the fields created by the clients was made available, because they're a natural way to summarize content and optimize scannability when reading.

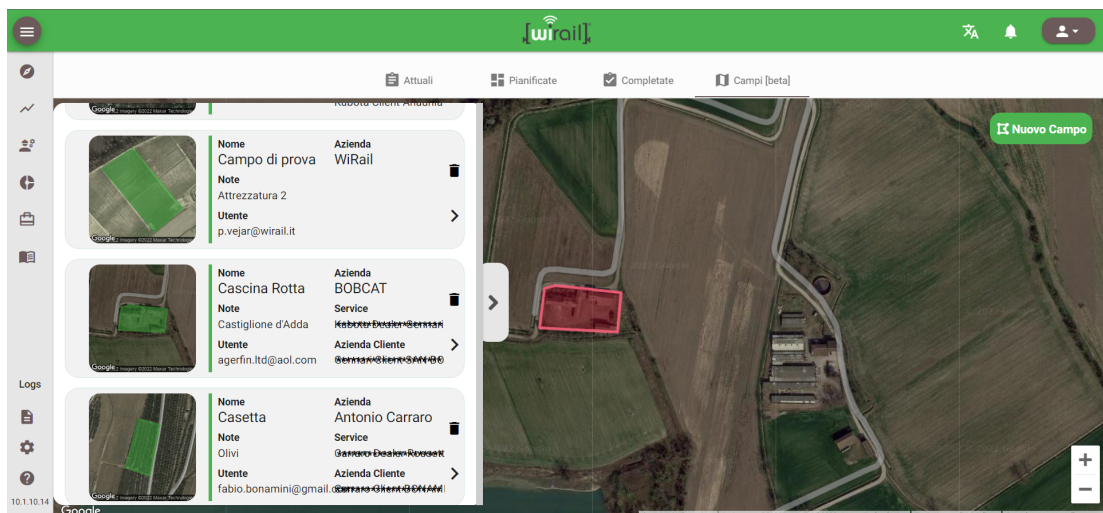
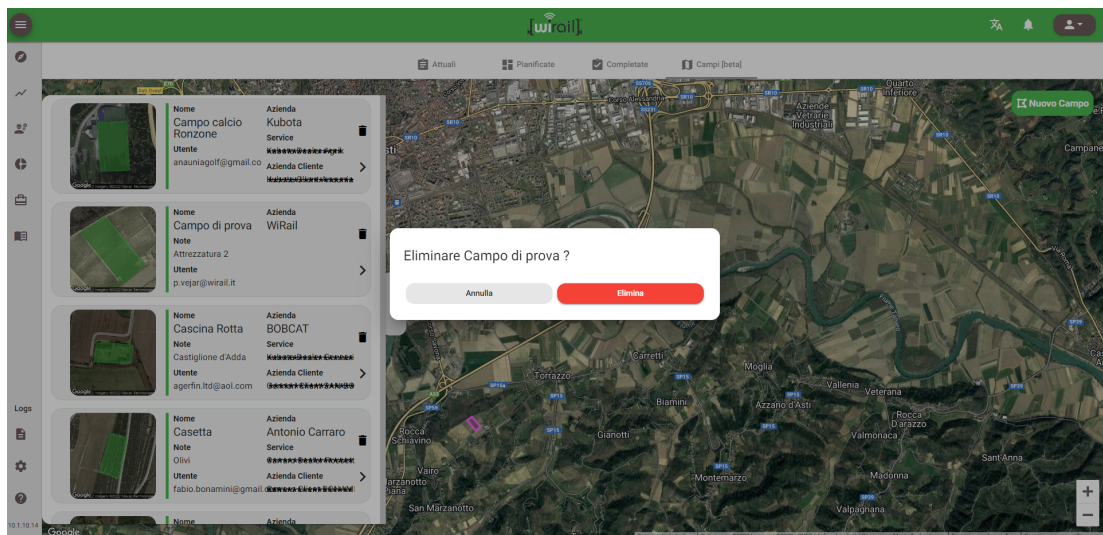


Figure 4.12: Elements in list

The list can be displayed automatically after creating a new field, or simply by using the gray bar with an arrow on the left side of the map. As can be seen in figure 4.12, each item in the list has the main information of the created entity, i.e. the field name, the notes entered and the user who created it. In addition, only for WiRail users, the name of the company of the field, the service and the corresponding client company are shown, in order to identify each field in a faster and easier way. Furthermore, there are two new available functionalities within each card in the list, the first one is to delete the fields in case of errors or termination of use of the field, and the second one is the redirection in the map of the field when clicking on it.

**Figure 4.13:** Delete field

4.4.3 Integration with Activities

The integration of fields into the monitoring system changes the dynamics with which data collection and activities are handled. This is due to the fact that, as mentioned in previous paragraphs, the activities have an address and a city to identify the place of work, however, with the new modifications

it is possible to associate a field directly to the work. This means that within the information of an activity, instead of showing the address and city to the operator, the name of the field and the corresponding number are shown within the list of fields created by that client. Consequently, as shown in Figure 4.14, the control unit shows the customer the current job together with the field code and the name of the field to be addressed.



Figure 4.14: Field interaction with control unit



Figure 4.15: Field position reported in activity tab

Regarding the dynamics of the data collection of each activity, it opens a lot of doors to improve the accuracy of the GPS antennas integrated in the control units, thus taking advantage of the potential of the IoT tools under development. For the moment, an algorithm has been developed to identify each time a point sent by the control unit, whose current activity is associated with a field already existing in the cast, is within the limits of the terrain. Thus, if a point sent is outside the limits, it is not recorded in the database, since it corresponds to an error of accuracy of the antenna, on the other hand, if a point is sent from inside the field, then it is taken, evidenced on the map and recorded within the activity.

4.5 Ray Casting Algorithm

As presented in the previous sub-chapter, the introduction of geofencing and its integration with the activities made it necessary to take measures to improve the geopositioning service of the entire system. In this sense, it was thought to take advantage of the new implementation of the fields to delimit the zones where the control units actually perform the data collection. This is of great importance, since many times GPS antennas are not accurate due to errors in the connection with the satellites or the lack of them, which can lead to errors that mean kilometers of inaccuracy and show incorrect data to the final users. This in turn could generate a feeling of uncertainty and distrust in the service delivered by the company, thus harming the productivity and experience of the final users.

As a consequence of what was presented in the preceding paragraph, an algorithm must be implemented to detect whether a certain pair of geographic coordinates is located within a zone on the map delimited by a polygon. Thus, it was decided to implement the Ray Casting Algorithm, also known as crossing number algorithm, which corresponds to an algorithm recognized in 1962 [15]. This algorithm is based on a simple geometrical observation: if a point moves infinitely along a ray to a defined point and crosses one of the boundaries of a polygon, possibly many times, it means that this ray is alternately entering and leaving from the outside to the inside of the figure. Thus, it is defined that after two border crossings the point moving along the ray goes to the outside of the figure, i.e., a point is outside a polygon if its corresponding ray intersects the figure an even number of times, while if this number is odd, the point is inside the polygon.


```
public boolean isPointInPolygon(List<GeoJsonPoint> points, GeoJsonPoint pointToVerify) {  
    int i;  
    int j;  
    boolean result = false;  
    for (i = 0, j = points.size() - 1; i < points.size(); j = i++) {  
        if ((points.get(i).getY() > pointToVerify.getY()) != (points.get(j).getY() > pointToVerify.getY()) &&  
            (pointToVerify.getX() < (points.get(j).getX() - points.get(i).getX() ) *  
            ( pointToVerify.getY() - points.get(i).getY() ) /  
            (points.get(j).getY() -points.get(i).getY() ) + points.get(i).getX() ) ) {  
            result = !result;  
        }  
    }  
    return result;  
}
```

Figure 4.16: Implemented code to check if point is inside field

In the case of the present application, the google API related to the implementation of maps and the delimitation of the zones, has a tool to determine if a certain coordinate is within a polygonal zone formed by more than 3 points, however, this functionality is only available at a graphical level (frontend) and not at a logical or server level. This generates a problem, since the coordinates are collected in real time by the receiver server and must be verified at that time and thus, avoid storing in memory data that are incorrect and inaccurate. For this reason, it was necessary to implement a solution at the server level that verifies in real time if a point corresponds or not to an integer data.

In this context, a simplified version of the algorithm was implemented using a semi-infinite projection of a horizontal ray for each point being evaluated. Due to the amount of information received by the server, it was decided to implement this alternative on the test server to evaluate the behavior in real time, so this functionality is not yet available to final users.

The implemented algorithm performs a check of all the times that this semi-infinite ray crosses some edge of the figure using a for cycle and validations which determine if the point is inside or not. The implemented code can be seen in the figure 4.16.

On the test server, a provisional view was created to visualize the data collected using the algorithm, resulting in what is shown in Figure 4.17.

As can be seen, all the points that the control unit records are within the field boundaries. However, because the mode for recording the trajectory on the map is linear, some lines cross the edges of the polygon without showing a realistic trajectory. For this reason, these changes are still under development and testing, in order to achieve a better and more accurate monitoring system.



Figure 4.17: Final Path on data with algorithm [beta]

4.6 Farm Implements

Another of the axes of this project is to work for an important aspect of the agricultural activity, in this case the farm implements. This machinery plays a fundamental role for the realization of the production activities, carrying

out different phases of an agricultural production such as harvesting, pluming, shredding, etc.

Given the importance of the role of the implements, it was proposed to modify the control units in order to monitor them and provide data to the clients to optimize their functionalities always through the implementation of smart farming. To start with the initial phase of this mini-project, the system had to be updated at all levels: receiver, server and client-side. Thus, the services at receiver level were modified, creating new entities in order to receive data from these machines and process them, to later show them to the client through the reports mentioned at the beginning of this thesis.

First of all, this new type of control unit allows the customer to insert data such as minimum and maximum RPM of the machinery and the bearing diameter, if it exists within the type of implement. This data can be modified both from the site and from the control unit and is updated in real time. In addition, company administrators must insert the corresponding machine codes with a maximum of 4 different machines per control unit. For this, it was made available in the configuration section of the control unit, a new tab to insert the corresponding data and observe the type of work performed by the implement, as can be seen in the image 4.18.

Commissa corrente

Attrezzature

Targa Attrezzatura

Identificativo
ATTR-001

Minimum RPM
250

Maximum RPM
700

Diametro (mm)
0

Nessun tipo di lavoro

Targa Attrezzatura

Identificativo
ATTR-002

Minimum RPM
250

Maximum RPM
700

Diametro (mm)
0

Nessun tipo di lavoro

Targa Attrezzatura

Identificativo
ATTR-003

Minimum RPM
250

Maximum RPM
700

Diametro (mm)
0

Nessun tipo di lavoro

+ Nuova attrezzatura: 3

Figure 4.18: Farm implements section

Once the farm implement information is validated, the control unit can access the network, so the user must select the type of work from the device screen to update it in the corresponding section of the site, thus starting the data collection. In this way, the information related to the machinery is also updated in the corresponding activity, being added to the machine details with a new section "Farm implements" and in the reports of the control panel if it has collected the data correctly. Figure 4.19 shows the information collected by the implement in the application reports.

Data	Identificativo attrezzatura	Ore Lavoro Attrezzatura	Ore Sessione Attrezzatura	Giro Minuto	Metri Sessione	Metri Totali	Porta Aperta	Allarme RPM Alto	Allarme RPM Basso
04-10-2022 10:19		---	---	---	---	---			
04-10-2022 10:50	ATTR-003	0	0	446	0	180			
04-10-2022 10:54	ATTR-003	0	0	446	0	180			
04-10-2022 10:58	ATTR-003	0	0	446	0	180			
04-10-2022 11:02	ATTR-003	0	0	446	0	180			
04-10-2022 11:06	ATTR-003	0	0	446	0	180			
04-10-2022 11:10	ATTR-003	0	0	446	0	180			
04-10-2022 11:14	ATTR-003	0	0	446	0	180			

Figure 4.19: Farm implements reports

This new tool is being evaluated directly by the final clients, since it corresponds to a new project and to a direct intervention on the control devices. It is therefore necessary to wait for their feedback in order to proceed with the development of this part of the project and wait for the data collection to effectively measure the impact of the introduction of farm implement monitoring on the management of customer activity.

4.7 Limitations of Project

In every project there are limitations that in one way or another prevent or hinder the progress of a line of work, which is why the previous organization is important and plays a key role in the development. For this reason, as explained at the beginning of this chapter, it was decided to use agile strategies to be prepared for any unpredictable change in the course of the project, whether due to changes in the requirements of the final users, human resources and timing problems, etc.

In the context of this project, the initial limitations corresponded to the difficulty of understanding the code and the structure of the system, due to the fact that there was not enough documentation of the application and its

details. In addition, there was no IT representative to guide me and indicate the steps to follow for each task to be performed. However, in the end this was a positive aspect for my learning, the development of my skills and my personal development within the work team.

As the stages of the project progressed, new limitations began to be created that sometimes prevented the natural progression of the development, such as the fact that the code was not being maintained regularly and had not been modified for a number of months before I started my intervention. For this reason, there were many libraries and frameworks that did not have versions adapted to the current requirements of the application, as for example the library in charge of illustrating the graphics. Also, some of the components had a considerable amount of bugs and erroneous behaviors due to the lack of maintenance and updates.

Finally, one of the strongest limitations in the course of the project was the lack of time and the difficulty of knowing how to manage it well, since it is a large and complex application that requires constant maintenance and updating. Thus, the fact of having worked on this project for only 5 months meant a limitation for the possible improvements and solutions that were proposed at the beginning of the thesis activity. For this same reason, some of the functionalities are still in beta versions as of today because they are being tested by the final users, waiting to receive comments and feedback so that the company can continue advancing in the development of this project.

Chapter 5

User Experience

In this chapter all graphic updates will be presented as a separate chapter together with the validation of the previous chapter's developments, since they were developed with the main objective of improving the user experience of the application. These decisions will also be validated following the Nelson Norman Group Design for UX guidelines.

User experience is a concept that describes how a user experiences and interacts with a certain product or service. It is composed by three main concepts that come from the person in study: efficiency, utility and ease of use. Having a good user experience is important for all kind of designers, developers and creators, since they can mark negatively the use of a product, ruining a whole system. For this reason, it was proposed as an additional objective of the thesis to update the current application to a more user-friendly one, focusing on design and usability of the whole system.

5.1 Guidelines for improving UX

Within the work of this thesis it was considered as a transversal objective to improve the user experience, since, it is useless to add new functionalities

if the user is not able to use them in a correct way. For this reason, as a personal work guide the usability heuristics of the user's interaction with the systems of the Nelson Norman group was used, which corresponds to a leading user-experience research association that has offered its consulting services to large companies such as Google, National Geographic, Visa, Sony, among others [16].

This group of researchers has proposed 10 heuristics to guide the design and user-experience of applications, corresponding to the following [17]:

1. **Visibility of system status:** The design of the tools should always inform the users about the things that are happening, through feedback in an adequate response time.
2. **Match between system and the real world:** The design should always be written in the language of the users, i.e. words, phrases and concepts familiar to the users should be used to make the information appear in a logical order.
3. **User control and freedom:** Deliver a feeling of control and freedom to the user when committing actions by mistake, thus leaving an emergency exit to reestablish the unwanted action without having to go through a lengthy process.
4. **Consistency and standards:** Users should not have to wonder if different words, situations or actions mean the same thing, so there is a duty to follow industry conventions and not waste the user's time.
5. **Error prevention:** The best designs are those that prevent the occurrence of problems, so it is important to eliminate conditions that could generate errors, check each condition before sending requests to the servers, provide users with a confirmation button each time they

commit an action and above all, display good error messages each time one occurs.

6. **Recognition rather than recall:** The memory demand on users should be minimized by making elements, actions and options more visible. Thus, the user should not have to remember information from one part of the interface to another. For example, field labels or menu items should always be visible and available when needed.
7. **Flexibility and efficiency of use:** Shortcuts can speed up the interaction for the "expert user" so that the design can satisfy both the inexperienced and the expert user. It is important to allow users to customize frequent actions.
8. **Aesthetic and minimalist design:** The interfaces present in the application should not contain information that is not useful or important to the user. Every piece of irrelevant information competes with the relevant information, decreasing its visibility, so it is important to avoid showing data that is not interesting to the final users.
9. **Help users recognize, diagnose, and recover from errors:** The user should be encouraged to suggest solutions to eventual errors, therefore, it is important to have error messages expressed in normal language, without error codes.
10. **Help and documentation:** The best sites are those where no additional explanation is required to use them, however, it is sometimes necessary to provide documentation to better understand how the user can complete their activities.

These heuristics were applied almost entirely in the new tools that were developed throughout the work, thus observing a great change within the

web application that were quickly noticed by the final users, increasing the use of the application and decreasing the number of calls from customers who did not understand how to use some features of the site.

The following chapter will show in detail how these heuristics were applied in the project along with the description of the final objective.

5.2 User-experience and Design Improvements

Among the modifications aimed at improving the user experience within the application, are the addition of two new items in the main menu, modifications within the main map (a tool frequently used by users), changes in the graphic design of some elements and the addition of new control and verification elements.

The new items added in the menu correspond to Guides and Statistics, which fulfill the objectives mentioned in heuristics 1 and 10.

5.2.1 Guides

Within this new page, various types of user guides are made available to users, both for the control units and the tools within the site. This new page is divided and customized by company, since not all companies have the same type of control units and the same access to the different functionalities. For example, some companies do not have access to items such as graphics, as these are tools that can be exploited by companies with a specific type of control unit.

As shown in the image, the guides correspond to an accordion menu with two types of guides, the first one related to the site and the second one to the machines owned by the clients. Thus, within the guides proposed in the first part there are instructions on how to compile the documents, how to use the map, how to organize the activities, how to use the reports, among

others. On the other hand, in the machines section, there are guides on how to install the control unit in the different models of machines owned by the customers.

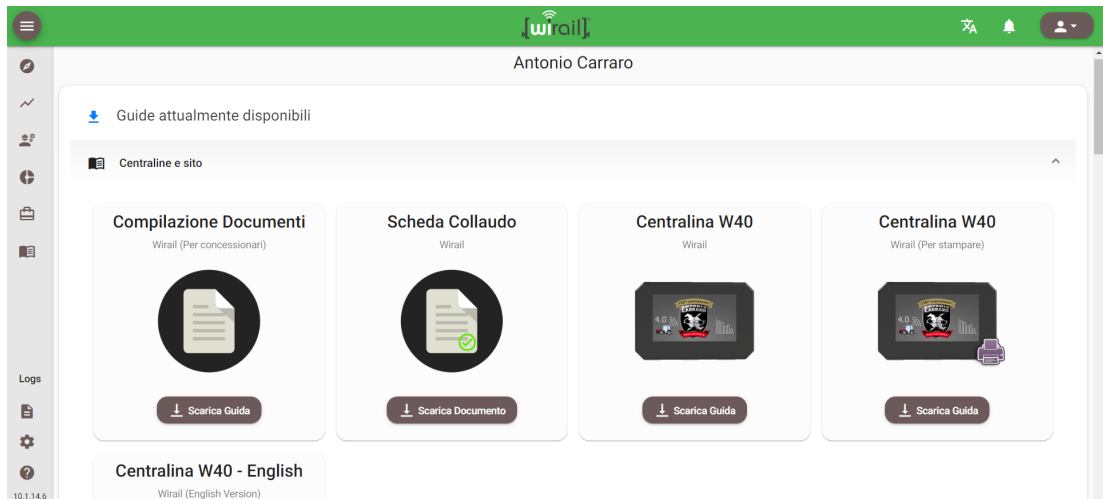


Figure 5.1: Web site guides

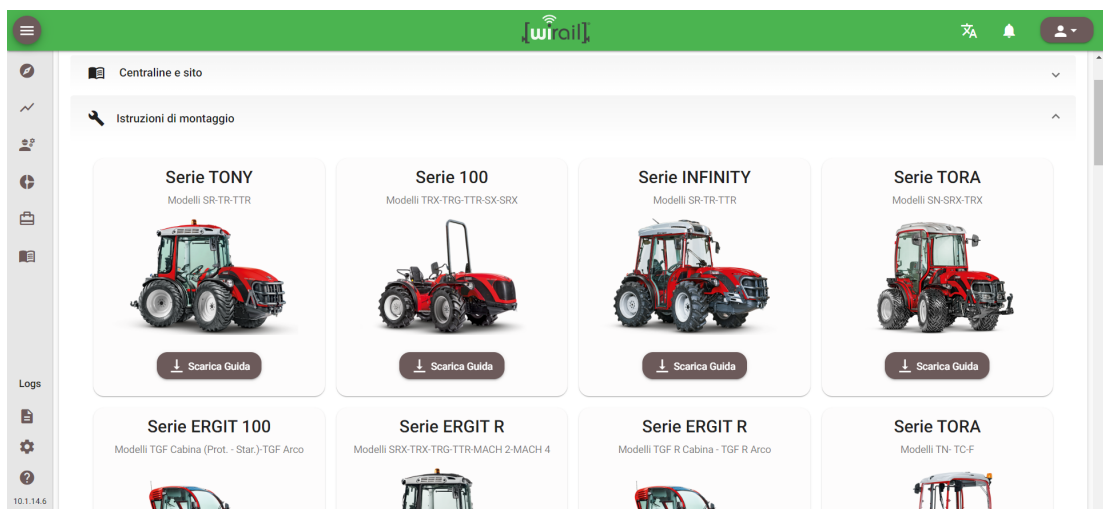


Figure 5.2: Control Unit guides

5.2.2 Statistics for Machines

The statistics page provides users to observe historical status parameters of their machines, choosing a time window along with a frequency that

can be daily, weekly, monthly, semi-annual or annual. In this first stage it was decided to develop only pie charts, but in the future it is planned to implement other types of statistics that may interest users.

As shown in the figure, this page is composed of a search engine, a frequency selector and the information. Among the parameters selected for display are Engine Load and Instantaneous Consumption together with the meaning of the colors represented in the graph, which correspond to levels with different values.

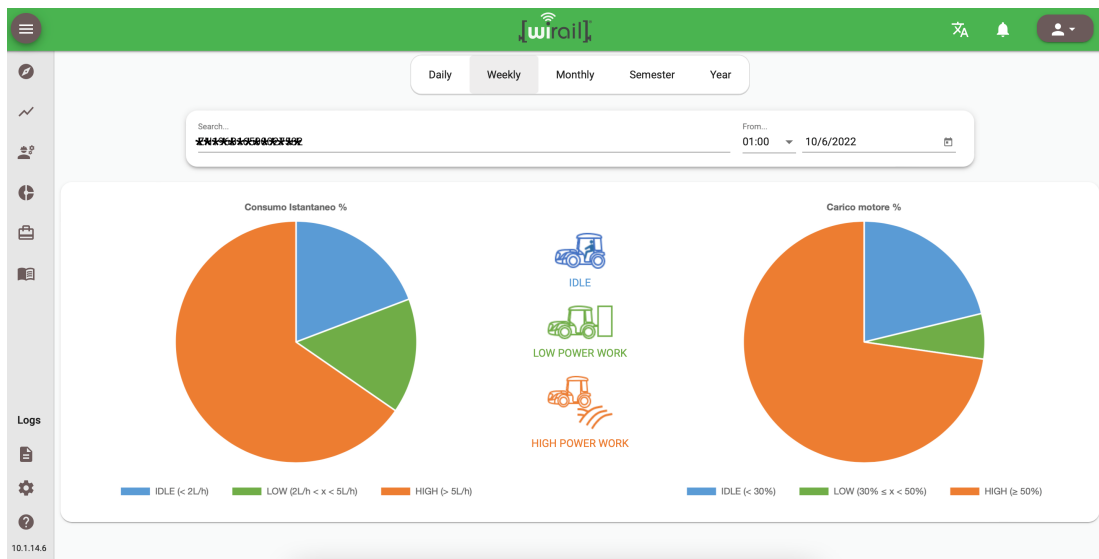


Figure 5.3: Statistics for specific machine

5.2.3 Details on Map

Among the list of changes aimed at improving the user experience, is the change to the map and the visualization of the control units. At the beginning of the project, the application only showed a little information about the machine, such as last update, machine plate number and other data, as shown in figure 5.4. With the new change made, now every time a user clicks on the circle that represents a control unit, they will see the information of the machine where the following data will be shown: plate number, information

about the connection, a picture of the machine, the model of the machine where the control unit is installed, information about the customer and the service that has sold it and finally, a couple of functional buttons that lead to the reports and/or graphics if they are available.

The change in the map arose thanks to direct meetings with customers who told us the need to see details of the machine when viewing the position on the map, so they could more easily identify the machines that have been leased or sold, in order to track the activities and working hours.

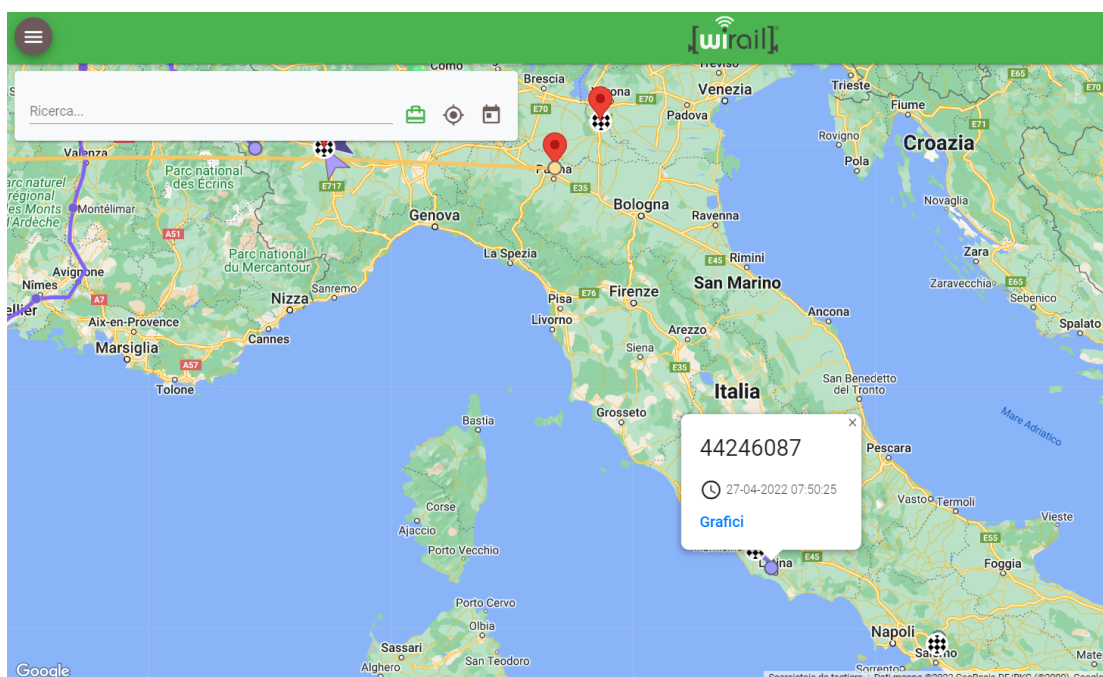


Figure 5.4: Old machine summary on map

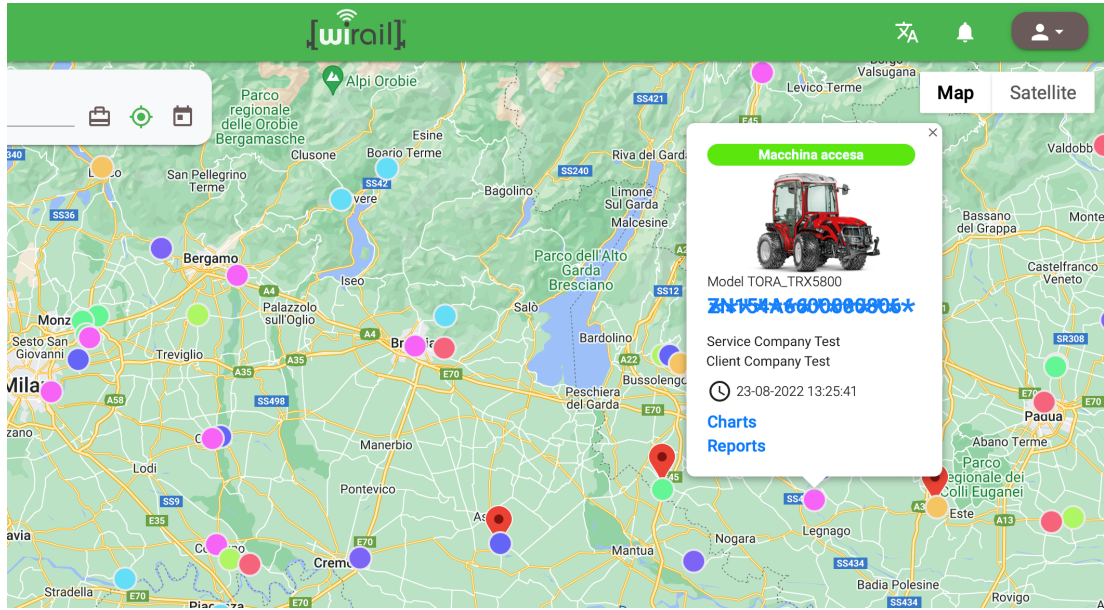


Figure 5.5: New machine summary on map

In figure 5.5 the new graph of the machines on the map together with the available information is shown.

5.2.4 Main Layer Transformation

Among the considerations when developing a current web application, it is important to maintain an updated and modern design, as this provides an experience more attached to what users are used to. It is for this reason that it was decided to make an update of the main layer graphics of the application, giving the user a renewed experience with new ways to interact with the application.

The changes made correspond to the passage of a square graphic to one with rounded edges, to give a sense of modernity to the application. Thus, each element that was inside a square structure has more oval edges as shown in the following images. In addition, a hidden slider menu was changed to a minimized version but always present on the user's screen, so it will be easier for the user to associate the different icons to the functionalities present.

The above changes are illustrated in the following figures.

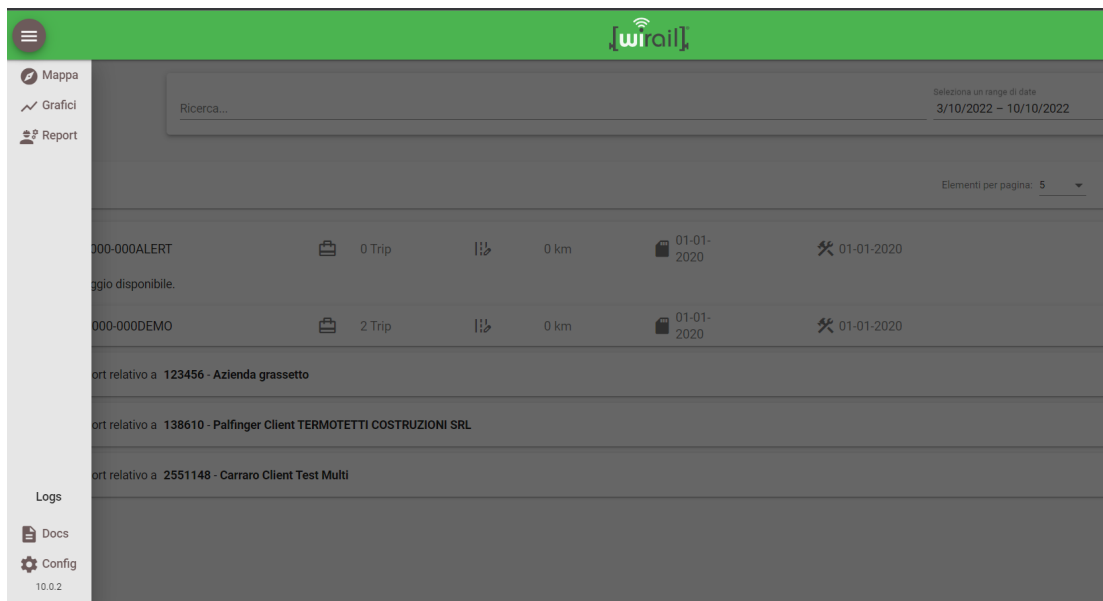


Figure 5.6: Old menu bar

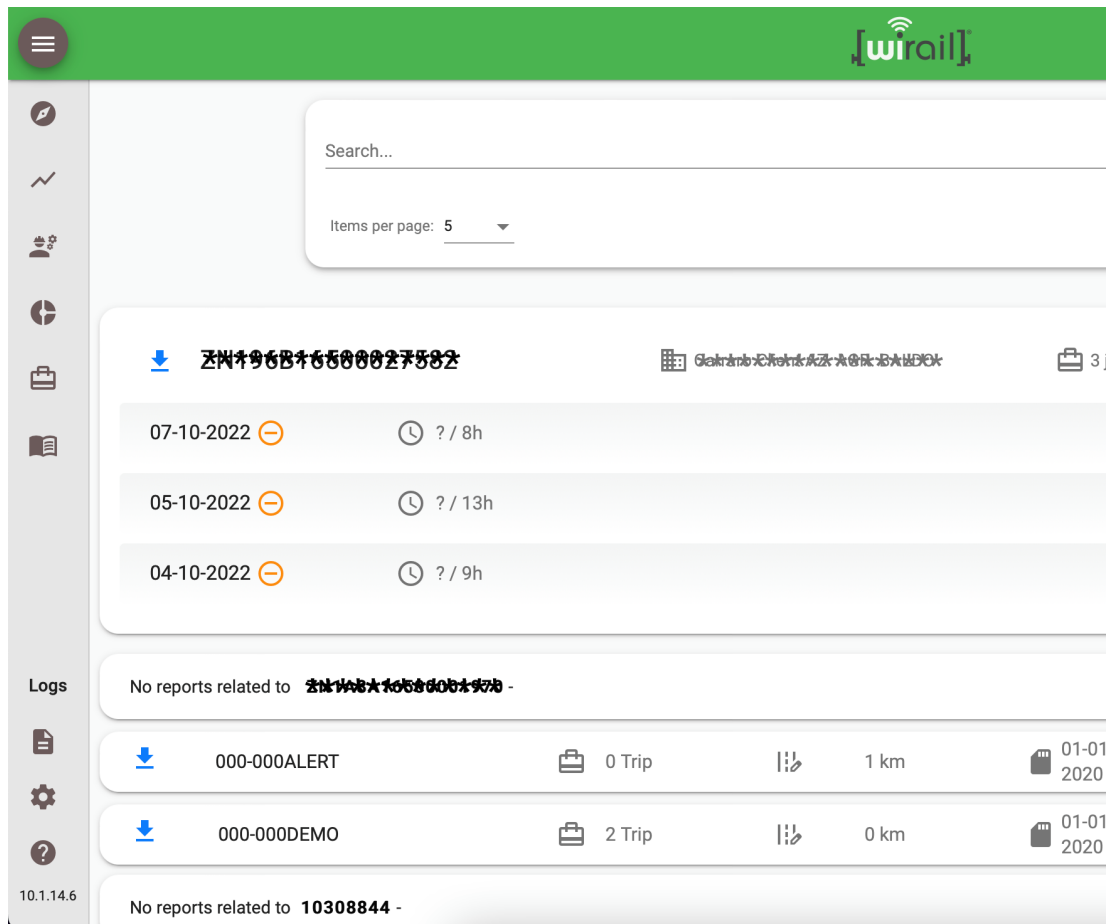


Figure 5.7: New menu bar

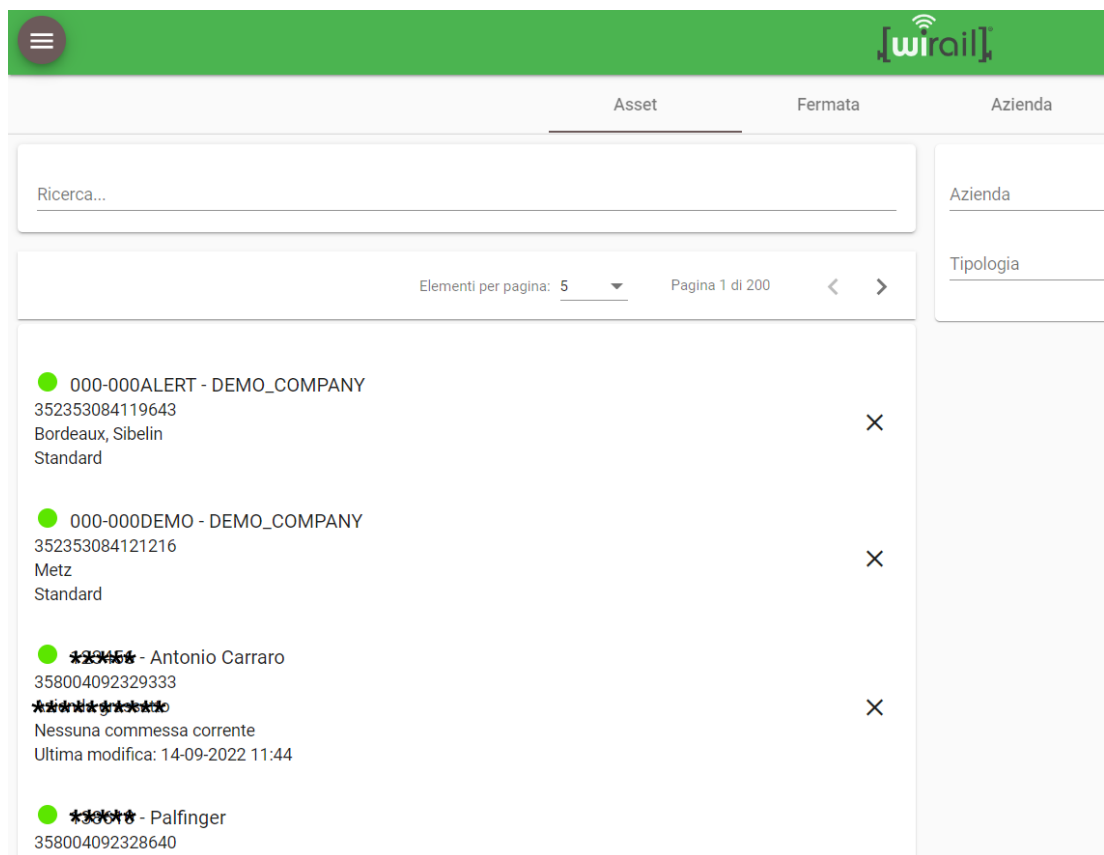


Figure 5.8: Old list layout

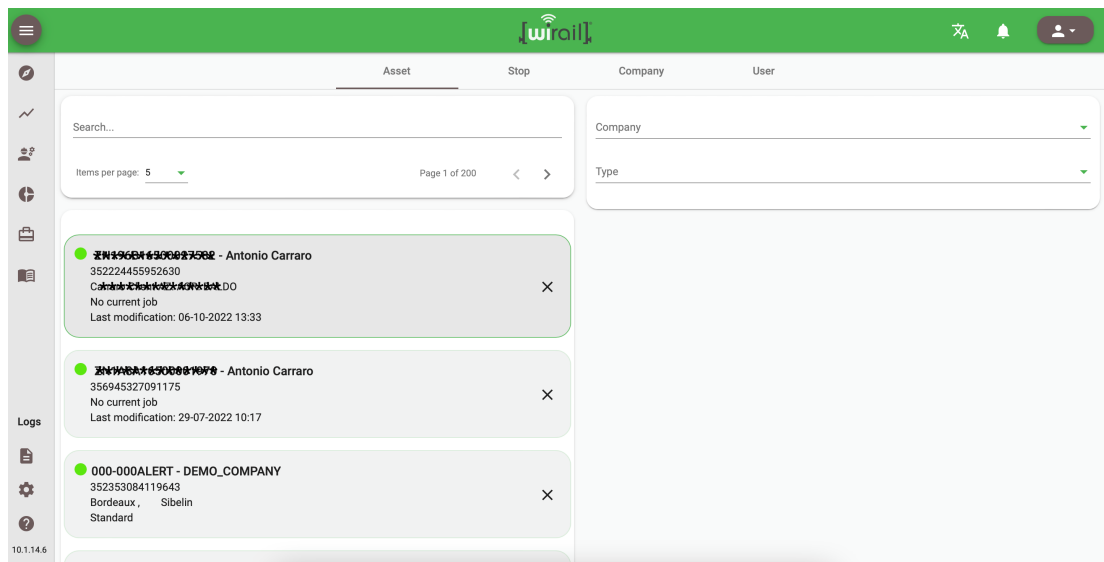


Figure 5.9: New list layout

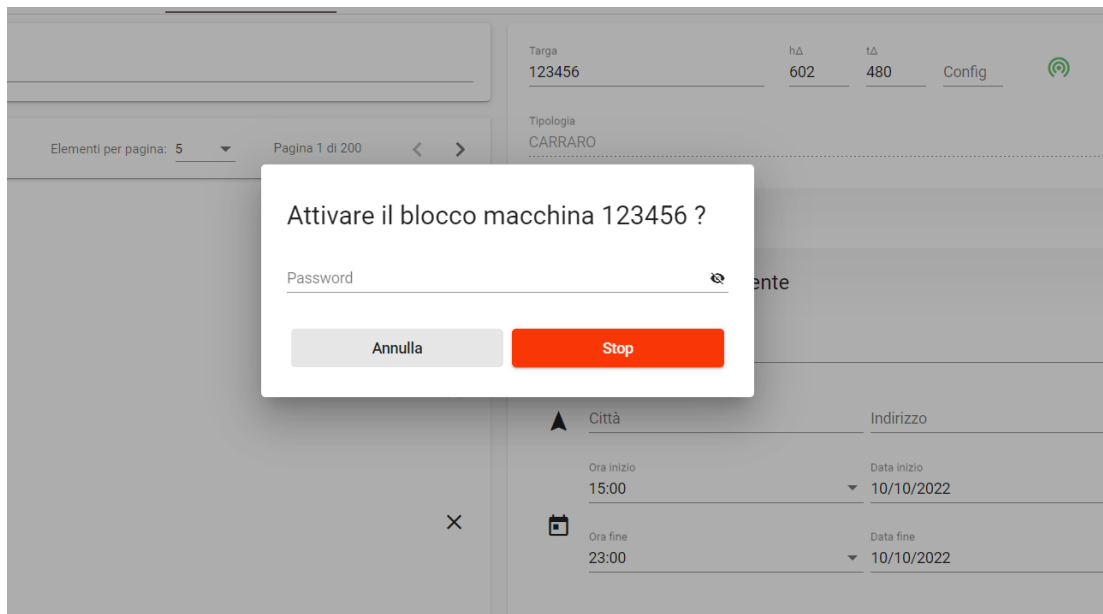


Figure 5.10: Old borders on elements

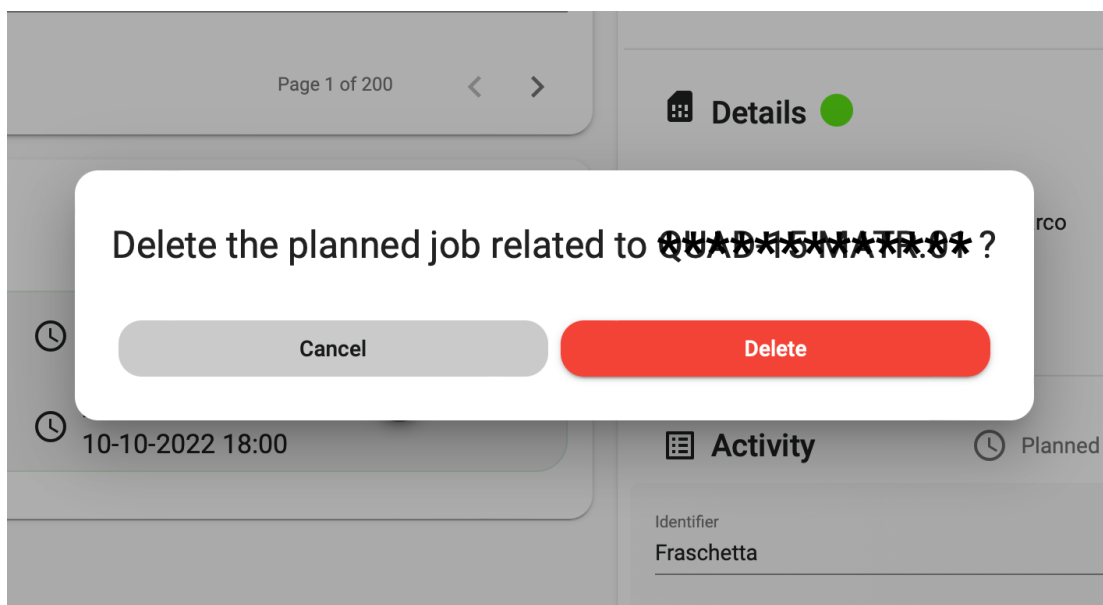


Figure 5.11: New borders on elements

5.3 Results of graphic updates

Although the work done in the paragraphs described above has not been extensive, it has meant a great advance for the project and the application itself, because the users have been able to adopt new ways of controlling their machines and interacting with the system. It has also provided new opportunities to unblock the company's consultation systems, since by making available the installation guides for the control units and the use of the site, users are invited to inform themselves autonomously.

Another important aspect is that these changes motivate users to keep asking for more improvements, which shows that the customer is interested in the system and wants to contribute to its growth.

Chapter 6

Conclusions

This chapter will present the final conclusions of the thesis work that correspond to a summary of work done within this experience, the aspects to be improved by the writer of this thesis and a brief vision of the possible future works to be developed.

6.1 Conclusion

In the work carried out in this thesis, an effort was made to improve the management of agricultural activities through the development of new smart farming tools, based both on the state of the art and on the improvement of the response mode to market and final users' demand that has not yet been fully completed.

Once a general picture of the current situation of smart farming in the industry was made, various solutions present in the agricultural market were analyzed, which gave an overview of the objectives that could be established as a team. Thus, we started with a linear process whose beginning was the analysis of the initial situation of the system and the capabilities that it had to allow an orderly, clear development with specific objectives. This meant

using tools associated with the development and design of globally adapted software, which corresponds to the analysis of the requirements of the final users or stakeholders of an application, thus creating use cases that come to describe how the user interacts with the application and what is obtained as a benefit from this interaction. As described in this thesis, after performing the analysis we proceeded with the first phase of the project, which corresponds to a general maintenance of the system and the updating of the present components, with the objective of establishing a base for the successive steps. For example, the update of the library that manages the interface of the graphs together with the endpoints related to the data collection, or the change of the component in charge of managing the documents that previously had errors and undesired behaviors, which caused a negative experience for the users.

Subsequently, we began with the development of the tools described in this thesis, which have had as main success the delivery of more opportunities to improve the management of agricultural activities of the users, thus improving the interaction that the user has with the application. This was carried out following again the user requirements analysis strategies, carrying out most of the proposed new features. Thus, a new concept "activity as the center of agricultural management" was developed, which allowed orienting the users to a better management. This was proven by the interest shown by the end customers at the time of the release of the new versions and by the increased interaction with the application.

All the work done, shows that one of the main aspects when considering the development of new technologies, is that of user interaction or user experience, an aspect that is constantly addressed in the curriculum of the career, because they are the final users who determine the needs and requirements that the product must satisfy. For this reason, throughout the course of the thesis, the general objective has been to optimize the user

interaction with the application.

6.2 Further work

The growth of market interest in the solution proposed by the company behind this project is evident, the number of control units sold has been increasing over the months. This gives way to the questioning about the growth of the project and its future steps, since it must have a system prepared for the market demand. Thus, as further work we propose a focus on the ability to scale the project in all senses, either in the number of contemporary connections, quantity and quality of functionality, optimizing security and the flow of user interaction with the system and increasing the database space. Specifically, the focus should be on changing the structure of the server in charge of receiving the connections from the control units, in order to increase the number of available connections without affecting the connection flow, avoiding data and time losses. In addition, to ensure a greater openness to the agricultural market, a focus must be placed on improving the usability of the system, adding new functionalities that meet and exceed market standards. This, through improving the current user experience by improving the design of the site and all the elements that compose it. Specifically, we propose an improvement to the main reports page, giving the possibility to visualize the data in an interactive, clear and optimized way in the parameters that users are most interested in, thus adding a new value to the system that will surely allow a growth of the project.

Bibliography

- [1] ERS/USDA. *Agriculture In The European Union*. Beef 2 Live. Aug. 2022. URL: <https://beef2live.com/story-agriculture-european-union-131-108887> (cit. on p. 2).
- [2] TERRATECH. *Smart Agriculture in Europe*. University of Porto (UPorto). Jan. 2022. URL: <https://www.terratechmsc.eu/smart-agriculture-in-europe/> (cit. on p. 2).
- [3] DOTMobile. *Piattaforma Telematica per il Monitoraggio e la Gestione di Mezzi e Attrezzature*. DOTMobile. URL: <https://puntomobile.it/funzionalità/> (cit. on p. 5).
- [4] WAY4FARM. *La soluzione WAY4FARM per Trattori e Macchinari Agricoli*. WAY. 2022. URL: <https://www.waynet.it/industria-4-0/agricoltura-4-0/> (cit. on p. 7).
- [5] Google. *Che cos'è l'architettura dei microservizi?* Google. 2022. URL: <https://cloud.google.com/learn/what-is-microservices-architecture> (cit. on p. 17).
- [6] Wikipedia. *Single-page application*. Wikipedia. 2022. URL: https://en.wikipedia.org/wiki/Single-page_application (cit. on p. 19).
- [7] MongoDB. *About us*. MongoDB. URL: <https://www.mongodb.com/> (cit. on p. 26).

- [8] Docker. *Use containers to Build, Share and Run your applications*. Docker. URL: <https://www.docker.com/resources/what-container/> (cit. on p. 26).
- [9] IBM. *Containers*. IBM. URL: <https://www.ibm.com/cloud/learn/containers> (cit. on p. 27).
- [10] Docker Docs. *Dockerfile reference*. Docker. 2022. URL: <https://docs.docker.com/engine/reference/builder/> (cit. on p. 28).
- [11] MongoDB. *MongoDB Manual*. MongoDB. 2021. URL: <https://www.mongodb.com/docs/manual/core/sharded-cluster-query-router/> (cit. on p. 29).
- [12] Tom King. *What Is Spring Boot?* Fusion Reactor TM. 2020. URL: <https://www.fusion-reactor.com/blog/technical-blogs/what-is-spring-boot/> (cit. on p. 33).
- [13] Wikipedia. *Java (Programming Language)*. Wikipedia. 2022. URL: [https://en.wikipedia.org/wiki/Java_\(programming_language\)](https://en.wikipedia.org/wiki/Java_(programming_language)) (cit. on p. 36).
- [14] Wikipedia. *Typescript*. Wikipedia. 2022. URL: <https://es.wikipedia.org/wiki/TypeScript> (cit. on p. 36).
- [15] Wikipedia. *Point in polygon*. Wikipedia. 2021. URL: https://en.wikipedia.org/wiki/Point_in_polygon (cit. on p. 69).
- [16] Nielsen Norman Group. *Usability and UX Consulting*. Nielsen Norman Group. 2022. URL: <https://www.nngroup.com/consulting/> (cit. on p. 77).
- [17] Jakob Nielsen. *10 Usability Heuristics for User Interface Design*. Nielsen Norman Group. 2020. URL: <https://www.nngroup.com/articles/ten-usability-heuristics/> (cit. on p. 77).