FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Web Application for Supervising and Managing an Automatic Cross-Docking System

Francisco Rego Moreira da Silva Costa

FOR JURY EVALUATION

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Orientador Interno: Prof. Hélio Mendonça

Co-orientador: Prof. Manuel Silva

Orientador Externo: Eng. Miguel Ângelo

Orientador Externo: Eng. João Oliveira

June 29, 2021

Abstract

We live in a world where technology becomes more and more important in our routine, making us dependent on our machines or automated processes. Over the years, and after revolutions in the industrial world, new concepts have emerged that have shown themselves capable of improving people's lives, increasing the efficiency of processes and reducing the associated time and costs. The main concept, better known as the Internet of Things (IoT), shows itself capable of establishing a viable communication between the various objects of the Internet, such as sensors and actuators in the real world. Concepts such as cross-docking, whose goal is to make the storage of products null or practically non-existent (with an intelligent flow between the arrival and departure of products), is becoming increasingly popular and companies want to integrate it into their routine operations. Therefore, this thesis presents the development of a Web Application that manages and monitors processes related to the Cross-Docking concept, which is part of the CrossLog project: consists in the realization of a physical and software system for automatic mixed palletizing in Cross-Docking logistics centers for demand-driven responsive value chains. Although the concept is recent, solutions have been studied in order to meet the requirements imposed by all parties involved in the project, The effort to develop a SCADA system capable of monitoring and controlling processes is reduced by an intuitive Web Application that communicates with all modules and allows users to interact with them, resolving anomalies and avoiding increased costs and time wasted. Simulation and real tests, as far as possible, were performed to evaluate how the application would behave in a real-world scenario. Results have shown to be concise and accurate, due to fast communication between modules and correct implementation.

Keywords: Cross-Docking, IoT, Web Application, SCADA, MQTT, Node-Red, Vuejs

Resumo

Atualmente, vivemos num mundo onde a tecnologia se torna cada vez mais importante na nossa rotina, tornando-nos dependentes das máquinas e processos automatizados. Ao longo dos anos, e após revoluções no mundo industrial, foram surgindo novos conceitos que se mostraram capazes de melhorar a vida das pessoas, aumentando a eficiência dos processos e reduzindo o tempo e custos associados. O conceito principal, mais conhecido como a Internet das Coisas (IoT), mostra-se capaz de estabelecer uma comunicação viável entre os vários objectos da Internet, tais como sensores e actuadores no mundo real. Conceitos como o *Cross-Docking*, cujo objectivo é tornar o armazenamento de produtos nulo ou praticamente inexistente (com um fluxo inteligente entre a chegada e a partida de produtos), torna-se cada vez mais popular e as empresas pretendem integrá-lo nas suas operações de rotina. Por conseguinte, esta dissertação apresenta o desenvolvimento de uma Aplicação *Web* que gere e monitoriza processos relacionados com o conceito *Cross-Docking*, que faz parte do projecto CrossLog: consiste na realização de um sistema físico e de software de paletização mista automática em centros logísticos *Cross-Docking*. Embora o conceito seja recente, foram estudadas soluções de maneira a cumprir os requisitos impostos por todas as partes intervenientes no projeto, O esforço para desenvolver um sistema SCADA capaz de monitorizar e controlar processos é reduzido por uma Aplicação *Web* intuitiva que comunica com todos os módulos e permite aos utilizadores interagir com eles, resolvendo anomalias e evitando o aumento dos custos e o desperdício de tempo. Simulação e testes reais, na medida do possível, foram realizados para avaliar como a aplicação se comportaria num cenário do mundo real. Os resultados demonstraram ser concisos e precisos, devido à rápida comunicação entre módulos e à correta implementação.

Keywords: *Cross-Docking*, IoT, Aplicação *Web*, SCADA, MQTT, Node-Red, Vuejs

Acknowledgements

First of all, I would like to thank my internal advisors, Prof Hélio Mendonça and Prof Manuel Silva, for all their availability throughout the development of this project, guiding me whenever there were doubts and obstacles, inciting my pride and the will to achieve increasingly satisfactory results. I would also like to thank Miguel Ângelo and JPM, for making available to me all the resources that were necessary during the last 5 months. They always made me feel at home and for that I am deeply grateful and satisfied. I have had the pleasure of working with very capable people, who have advised and taught me how to get around problems and obtain satisfactory solutions, even though we are currently experiencing an atypical global situation, COVID-19. I would also like to thank my family for the support and conditions given and my friends, especially Vasco Gradim, who accompanied me in this important journey of the last 5 years. Finally, to express my gratitude to the Faculty of Engineering of the University of Porto (FEUP) that prepared me for the world of work and to overcome obstacles that appear in life, which shape me and make me be what I am today.

Francisco Costa

Contents

List of Figures

List of Tables

Abbreviations

AGV	Automated Guided Vehicle
ERP	Enterprise Resource Plannning
GB	Gigabyte
HTML	HyperText Markup Language
IIoT	Industrial Internet of Things
IoT	Internet of Things
JPG	Joint Photographic Experts Group (JPEG)
KPI	Key Performance Indicator
MES	Manufacturing Execution System
MQTT	Message Queuing Telemetry Transport
OPC-UA	OPC Unified Architecture
PLC	Programmable Logic Controller
PNG	Portable Network Graphics
RFID	Radio Frequency Identification
RGB	Red Green Blue
RGBA	Red Green Blue Alpha
SCADA	Supervisory Control and Data Acquisition
SSGI	Sistema de Supervisão e Gestão Integrada
XML	Extensible Markup Language

Chapter 1

Introduction

1.1 Context

As part of the final dissertation for the conclusion of the master's degree in Electrical Engineering by the Faculty of Engineering of the University of Porto (FEUP), I chose to carry out this research in a business environment. After some online interviews, I was selected to be part of the JPM group, the company with which I preferred to develop the current project. JPM is a distinguished company as SME Leader, specialized in the assembly of industrial conveyors, industrial equipment, supervision and control software and automation and robotic projects. Speed and productivity of a supply chain has become an important factor of growth for organisations. *Cross-docking* is just one strategy that can be implemented to help achieve a competitive advantage. Implemented appropriately and in the right conditions, this concept can provide significant improvements in efficiency and handling times. The research that I will develop focuses on the creation of a Web Application that aims to manage and monitor a process related to the *Cross-docking* concept, which is part of the *CrossLog* project [28]: consists in the realization of a physical and software system for automatic mixed palletizing in cross-docking logistics centers for demand-driven responsive value chains. More details about this project are covered in the Literature Review and adjacent titles.

1.2 Objectives

Although the dissertation is an integral element of the course, the work in company enabled me to apply concepts addressed over the past five years, in addition to contact with the business environment (initially, the work was carried out remotely due to current pandemic circumstances, however, as soon as possible, it was preferable to move to the headquarters of JPM or their office in Parque da Ciência e da Tecnologia da Universidade do Porto (UPTEC). In addition to what has already been stated, the realization of this project brought me a broader knowledge about:

- Programming languages aimed at developing Web applications, both back and front-end.

- Secure authentication.
- Communication between modules.
- PLC Programming.
- Order prioritization.
- SVG manipulation.
- Writing of minutes, progress reports and final document (according to pre-defined structure).

1.3 Motivation

The subject in question arouses a lot of interest in me, since currently the information available on the concept of *cross-docking* is reduced, because it is something recent and not yet very explored, but at the same time very interesting. If the project in which I am inserted is developed as outlined, Industry 4.0 will suffer a positive change, creating an intelligent, robotized and effective system, decreasing costs in the medium/long term and increasing levels of effectiveness of the different actors in the process. I intend to be part of this new path that's being outlined.

> *"There is no alternative to digital transformation. Visionary companies will carve out new strategic options for themselves — those that don't adapt, will fail."* Jeff Bezos, Amazon

1.4 Problem Definition

The *CrossLog* [28] project aims to study, develop and implement an automatic cross-docking system, capable of managing the flow of products quickly and safely. The control of the objects will be carried out using a set of modules:

- Industrial conveyors.
- Machine vision systems (using robotic units).
- Routing and communication systems between modules.
- Management and controlling system.

Currently, the processes present in distribution centers are highly dependent on the human hand. With the implementation of this system, they will be automated, guaranteeing a new milestone in the logistics sector, where unpredictability is taken as an obstacle. The project should result in an innovative concept aligned with the principles and concepts of Industry 4.0, whose main ideas and objectives are:

- Presentation of an alternative to manual cross-docking systems.
- Ability to create a system that is capable of monitoring and supervising each activity in the factory.
- Implementation of a three-dimensional identification, weighing and measuring station.
- Development of an accumulator station with optimized positioning.
- Conception of a multi-format gripper capable of handling various products.
- Implementation of a three-dimensional mosaic generation software.
- Constitution of a packing and palletizing station, allowing to guarantee the effectiveness and speed in these processes.
- Concept of a control and monitoring software (project on which this dissertation is focused).

It should be noted that all these concepts should first be tested in the laboratory and only then move on to a physical and operational environment, which will allow results about the reliability, effectiveness and usability of the proposed solution [13]. Regarding the Integrated Management and Supervision Software (SSGI), the main research topic of this dissertation, it is necessary to take into account some obstacles that may arise throughout the project and some minimum requirements that must be presented and met, so that this integral part of the whole project is responsive, fast, effective and secure.

1.5 Document Structure

This document is divided into 5 chapters:

- **Chapter 1 - Introduction -** Presentation of the proposed theme and main concepts, clarification of the general objectives and definition of the problem.
- **Chapter 2 - Literature review -** Study and theoretical review of the concepts and technologies to be implemented in the construction of the application.
- **Chapter 3 - Proposed solution -** Presentation of the proposed solution and project objectives, as well as system architectures and explanations.
- **Chapter 4 - Implementation -** Implementation strategies and tools used, visualization of results obtained.
- **Chapter 5 - Conclusions and Future Work -** Final reflection about the solution presented in the dissertation, possible alternatives, and future work.

Chapter 2

Literature review

2.1 Industry 4.0

Since the beginning of industrialization, the advance of available technologies and resources has promoted the development of industrial activities, leading to a revolution in this sector. It can be considered that, until today, there have been four great revolutions in the world of industry, which have led to changes in the structure of the company, changing people's way of life and work [24]:

- The first one started at the end of the 18th century, revolutionizing the mechanization of processes. It marked the transition from manual to mechanical work, using water and steam power for such purposes.
- The second occurred almost a century later, and was based on the use of electricity for mass production.
- The third happened in the 20th century, with the implementation of digitalization. To generate an automatic production, resources from electronics and computers were used.
- The fourth and last revolution, better known as Industry 4.0, has emerged in Western countries and will be discussed with further detail over this project chapter.

Figure 2.1 shows the four phases in chronological order and an icon representing each of them.

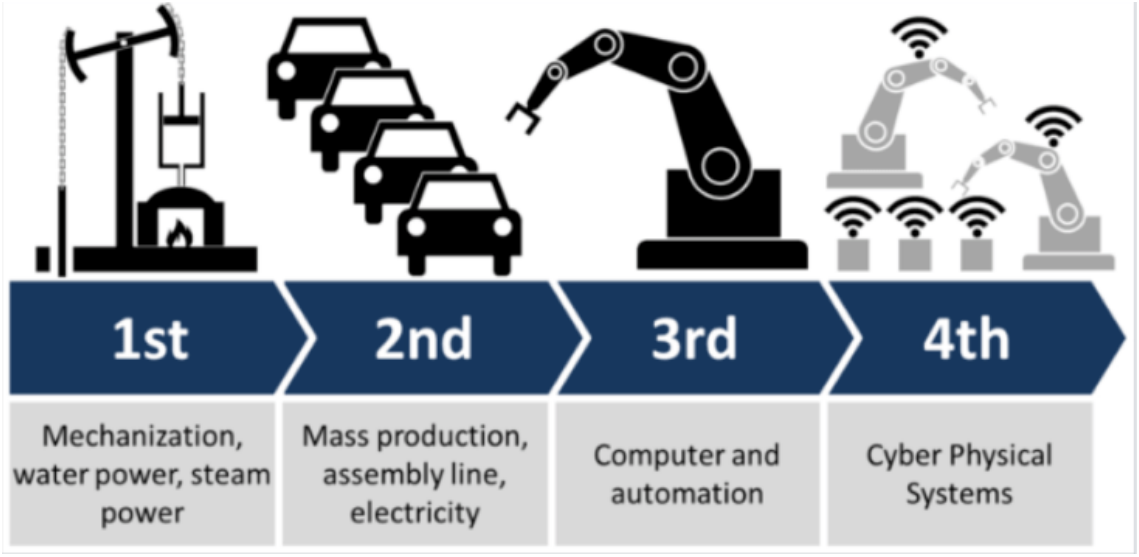

Figure 2.1: Different phases of the industrial revolution in chronological order [22]

Industry 4.0 dictates the end of the traditional industry as we once knew it, and encompasses concepts such as The Internet of Things (IoT), Enterprise Resource Planning (ERP), Big Data and Cloud Computing [18]. The concept of Industry 4.0 is still under development, but it is already possible to imagine the benefits inherent to its use [32]:

- Production increase due to process optimization and production speed increase.
- Improved working conditions, due to increased safety and reduced stress.
- Product and service customization, according to customer demand.
- Flexibility and adaptation to various types of technological markets.

Having reviewed the positive points about this innovative concept, it also presents some problems, the most worrying being the industrial espionage and the possibility of computer attacks on the systems in question. Another issue, this one a little more ethical, is related to the malicious use of artificial intelligence, whose use is recurrent in industries.

2.2 Internet of Things

The Internet of Things, better known as IoT, is the concept used to designate connectivity between various types of everyday objects sensitive to the Internet. The advantage is that all the participants are connected via *wireles* and easily accessed and controlled remotely, allowing a data flow at any time [16]. Being a cyberphysical system (system composed of computer elements with the purpose of controlling physical entities), IoT devices are equipped with embedded software and computer power through sensors, actuators and processors. The sensors and actuators are devices that enable interaction with the physical environment. While the sensors provide information about the current state of the device, an actuator performs actions to affect the environment in

some way. The combination of these two elements allows objects to be constantly available to interact with people, allowing production. The IoT paradigm opens the door to new innovations between *things* and human beings, allowing the realization of intelligent cities, infrastructures and services to improve the quality of life and use of resources. It is possible to find this concept in several areas such as health, sports, education, entertainment, social life, transportation, etc. Figure 2.2 represents the impact of IoT on various segments, and the percentage of global share of these type of projects.

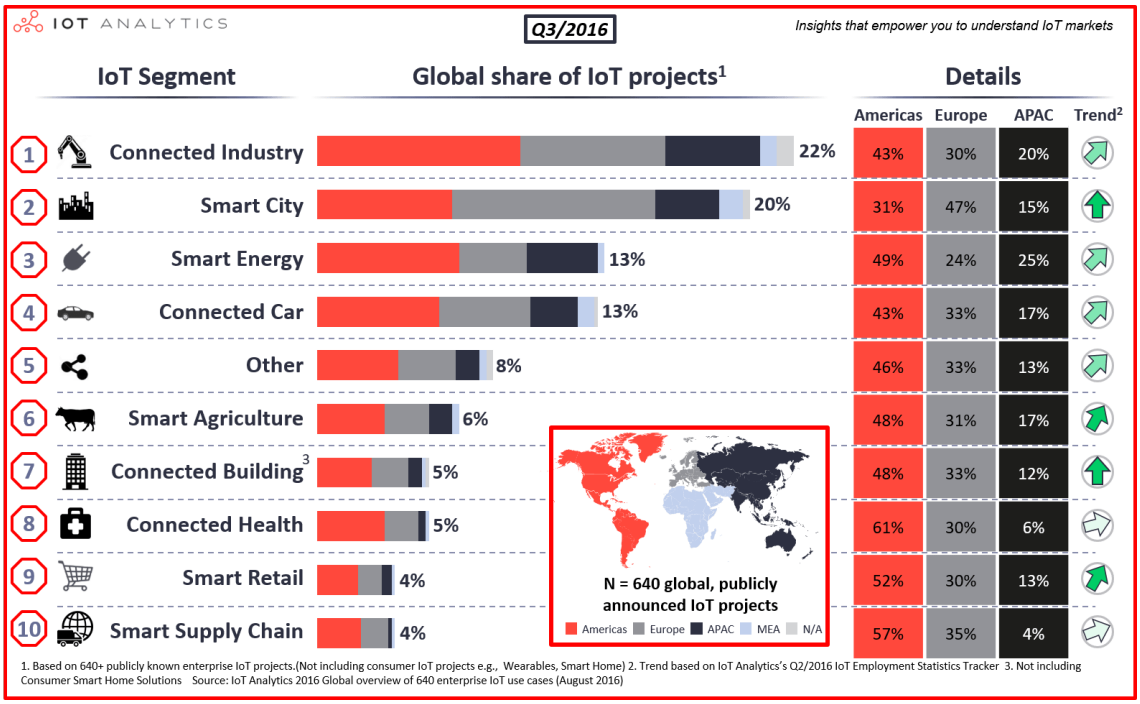

Figure 2.2: Internet of Things and its impact on various areas [26]

It is also possible to mention that most IoT projects focus on reducing costs to about 54%, according to studies. The other main purposes were related to the increase in revenues (about 35%) and increasing the overall security of processes (24%) [16].

2.2.1 Internet of Things architecture

The core workflow of IoT can be described and shown in Figure 2.3:

- Sensing, identification, and communication - information comes from sensors, depending on their type. This is then transmitted to other objects.
- Trigger an action - Information is received and functions are invoked according to the objectives previously defined by the user.
- Feedback - The system is able to provide monitoring of the process variables, as well as the result of the executed actions.

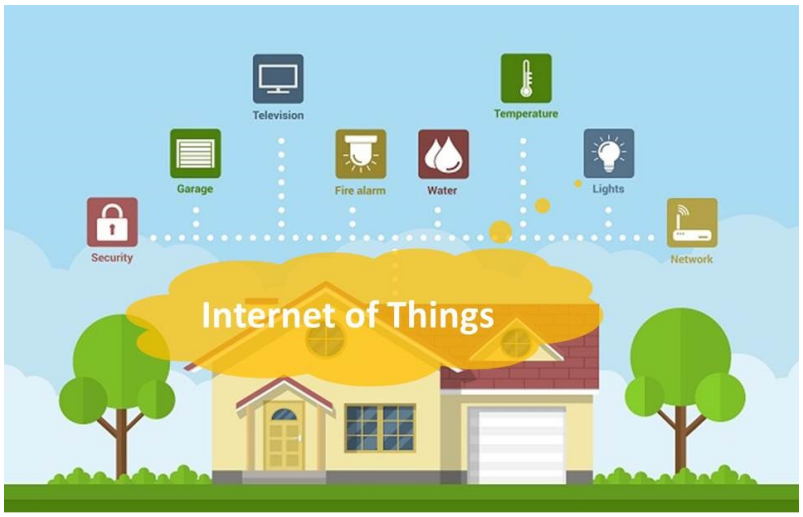

Figure 2.3: IoT concept application in a smart house [28]

Some adversities and points to be aware of when using this concept are recognised, such as:

- Object identification - due to the large number of objects belonging to an IoT system and the need for each of them to have a unique identifying key, an efficient management and naming system is needed so that variables can be easily accessed and modified.

- Standardisation - Enabling globalisation of technologies used and thereby granting better interoperability for all devices.

- Information privacy - An IoT system, by using different types of object identification (RFID, barcode, QRCode), needs security measures against unauthorised access.

- Physical security of objects - IoT systems are composed of several devices, located in a certain area. Thus, their physical integration is necessary to prevent the physical action of intruders.

- Confidentiality and data encryption - The information available in the system should not be accessible to all users. Consequently, it is important that the data read from the sensors is encrypted, as well as possible passwords and other data.

- Spectrum - The devices need an exclusive channel to transmit data over the network. However, the spectrum available to allocate these channels is restricted. Therefore, it is necessary to develop an efficient allocation mechanism to allow a huge number of devices to communicate over a wireless network [29].

- Sustainable use - It is important to consider the energy consumed by the system, taking into account the increase in devices and objects. Therefore, it is necessary to adopt measures to make the network as energetically efficient as possible.

2.3 Industrial Internet of Things

This concept is an intrinsic subsystem to IoT and it is used in industrial applications, promoting communication between the plant floor and system management. The fact that machines are able to perform specific tasks with a smaller margin of error when compared to human hands, such as data acquisition, has generated a change in the industry paradigm, making IIoT almost mandatory in the giants of technology [5] [3]. It has the function of interconnecting all industrial components, including machinery, control systems and information and management systems, to provide a fast and effective response throughout the entire life cycle of a product. The generation of large amounts of data enables collection and analysis with the aim of improving activities and consequently processes, and thus make the system more sustainable, faster and less expensive. It is also possible, through this same analysis, to reduce the number of component failures and optimize industrial operations [10].

2.4 Just in Time manufacturing

This concept, also known as *Just-in-Time production* or Toyota Production System (TPS) is a method used primarily to reduce production system times as well as supplier-customer chain response times. It was initially developed in Japan. Following the World War II, this country lacked the cash to finance big-batch, large inventory production methods used by other developed countries. They also had a high rate of unemployment and lack of natural resources.

Therefore, to overcome this crisis, they had to make their processes more effective. They built small factories, which concentrated on quickly turning small amounts of raw materials into small amounts of physical products. Processing smaller batches allowed the manufacturers to reduce financial risk, while slowing generating sustainable levels of working capital [19]. The system that they used came to be known as *Just in Time* manufacturing, popularized in Western media as the Toyota Production System. Later, the concept was globalized, and large companies adopted this method, making their processes simpler, with constant supervision and communication between all modules of the company, ensuring a healthy industrial flow. The adoption of the JIT method requires discipline, structuring and process explanation. Some procedures that are included in this concept are [19]:

- Housekeeping - physical organization and discipline.
- Elimination of defects.
- Setup reduction and flexible changeover approaches.
- Balanced flow - actively managing flow by limiting batch sizes.
- Control by visibility - using visual tools to improve communication (supervision and management systems).

- Streamlining the movement of materials
- Constant product tracking and analysis of processes.

When implemented in the best way, the amount of advantages associated with this method is high, impacting the productivity of companies, risk management and cost of operations. The main benefits are the reduction of inventory, labor costs, space required to perform an operation, throughput time and standard hours. Associated to these points, it is also possible to increase the production, the quality of the products and the number of shipments. In general, companies that implement JIT production methods benefit from this reduction in the production cycle, increase in the speed of products to market and reduction in the cost of operations. However, for smaller organizations, to achieve success, it is important to find suppliers at a reduced distance, or who can supply in a way with limited advance notice. Sometimes, minimum order policies can pose a risk to smaller manufacturers who might order smaller quantities of materials [19]. Cross-docking is a Just in Time strategy for distribution centers. It aims to reduce inventory levels and distribution deadlines, creating a continuous flow of products between suppliers and customers.

2.5 Introduction to Cross-Docking

In the age of digitalization, in order to make the processes more and more effective, a paradigm shift arises from a *forecast-lead supply chain* environment to one of *responsive demand-driven supply chain*, to which are associated important factors such as accessibility, customization of the offer and customer experience [25]. Currently, retail (namely food, pharmaceutical) has as main objectives the reduction of costs, speed of response and efficiency of activities associated with a process. With the increase in the number of stores/supermarkets of proximity of smaller dimensions, when compared to traditional hypermarkets, it is necessary an alternative that analyzes oscillations in the customer's consumption patterns, and that allows the absence of a vast warehouse destined for inventory. The management of stock "*Just-in-Time*" is a strategy used to increase levels of efficiency of the company, which determines what should be ordered, stored or sent the same day. However, the increase in the replacement rate requires a greater effort of logistical operations such as identification, palletization and movements. As a response to this problem, the concept of cross-docking, a strategy of minimum cargo handling, arises, with the objective of reducing or ceasing storage between unloads and cargoes of the same. As discussed in the previous chapter, the concept has little information because it is recent, so its automation is not yet possible. For this, technical-scientific advances are still necessary in the subsystems necessary to materialize the system for retail. Figure 2.4 shows the relation between suppliers and customers, and important processes inherent to the application of this concept.

In simple terms, inbound products arrive through transportation such as trucks/trailers, and are allocated to a receiving dock on one side of the *cross dock* terminal. Once the inbound transportation has been docked its products can be moved either directly or indirectly to the outbound destinations; they can be unloaded, sorted and screened to identify their end destinations. After

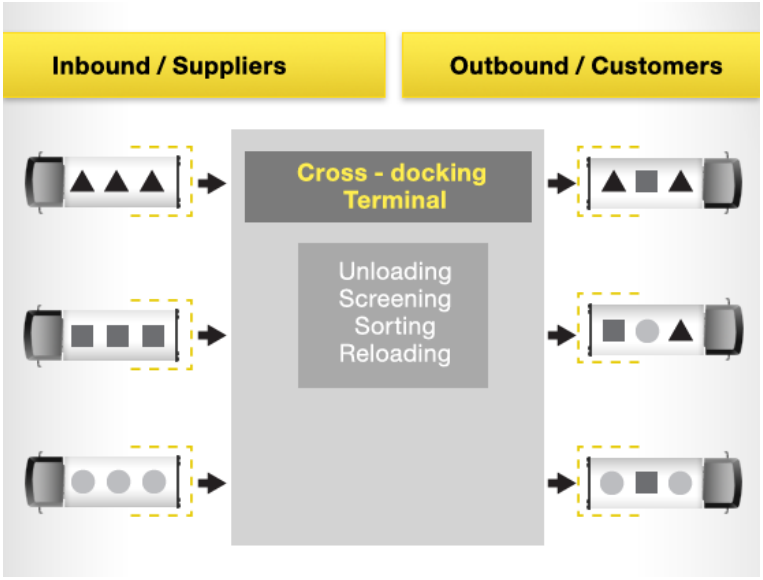

Figure 2.4: Cross-docking main concepts [11]

being sorted, products are moved to the other end of the 'cross dock' terminal via a forklift, conveyor belt, pallet truck or another means of transportation to their destined outbound dock. When the outbound transportation has been loaded, the products can then make their way to customers [11]. The main reasons for implementing this concept are the following:

- Provides a common center for products to be cataloged and combined so that they can be distributed to multiple destinations quickly. This process can be described as *hub and spoke* [11].
- Combine numerous smaller product loads into one method of transport to save on transportation costs.
- Division of large product loads into smaller transport loads in order to create an easier delivery process for the customer. This concept can be described as *deconsolidation arrangements*.

2.5.1 Cross-docking characteristics

Several characteristics can be considered to distinguish between various types of cross-docks (and cross-docking). A common distinction made in the literature is based on the number of touches or stages. In one-touch cross-docking, products are touched only once, as they are received and loaded directly in an outbound truck. This is also called pure cross-docking. In a two-touch or single-stage cross-dock, products are received and staged on the dock until they are loaded for outbound transportation. Usually, the goods are put into zones corresponding to their strip or stack door. In the case of a multiple-touch or two-stage cross-dock, products are received and staged on the dock, then they are reconfigured for shipment and are loaded in outbound trucks. In a typical

configuration, the incoming freight is first put in zones corresponding to the strip doors. The goods are then sorted to the zones corresponding to the stack doors. Another distinction can be made according to when the customer is assigned to the individual products [31]. In pre-distribution cross-docking, the customer is assigned before the shipment leaves the supplier who takes care of preparation (labeling and pricing) and sorting. This allows faster handling at the cross-dock. On the other hand, in post-distribution cross-docking, the allocation of goods to customers is done at the cross-dock. In this section, several characteristics are described that can be used to distinguish between different cross-dock types. They can be divided into three groups: physical characteristics, operational characteristics and characteristics about the flow. In the next sections, these groups will be described in more detail [31].

2.5.2 Physical Characteristics

The physical characteristics of the cross-dock are supposed to be fixed (for a long time). The following physical characteristics are considered [31]:

- Shape: Cross-docks can have a large variety of shapes. The shape can be described by the letter corresponding to the shape (I, L, U, T, H, E, etc).

- Number of dock doors: A cross-dock is also characterized by the number of dock doors it has. In practice, cross-docks range in size from 6 to 8 doors to more than 200 doors. Sometimes, in the literature, the number of dock doors is limited to only 1 or 2. In these cases, the idea is not to model a realistic cross-dock, but to gain some insight by studying a simplified model.

- Internal transportation: The transportation inside the cross-dock can be executed manually (by workers using forklifts) or it can be an automated system (a network of conveyor belts). The available infrastructure will of course be dependent on the type of freight that is handled in the cross-dock. A combination of both transportation modes is also possible.

2.5.3 Operational Characteristics

Some operational decisions can influence how the cross-dock works. These limitations lead to the following features [31]:

- Service mode: determines the degree of freedom in the allocation of inbound and outbound trucks to dock doors. If this service mode is used, mostly one side of the cross-docking terminal is assigned to inbound trucks and the other side to outbound trucks. The second mode is a mixed one, where inbound and outbound trucks can be processed at all doors. These two modes can also be combined. In this combination mode, a subset of doors is operated in exclusive mode while the rest of the doors is operated in mixed mode.

- Pre-emption: if allowed, the loading or unloading of a truck can be interrupted. This truck is then removed from the dock and another truck takes its place. The unfinished truck has to be docked later on to finish the loading or unloading.

2.5.4 Flow characteristics

The characteristics of the flow of goods that have to be processed by a cross-dock can be very different [31]:

- Arrival pattern: The arrival times of the products are determined by the arrival times of the inbound trucks. The arrival pattern can be concentrated at one or more periods if the inbound trucks arrive together at the same time (with a small time margin). For instance, a cross-dock in the LTL industry (transportation of relatively small freight) serving a certain geographical area usually receives load at two periods. Products that have to be transported from inside that area to another area are picked up during the day and all pickup trucks arrive in the evening at the cross-dock. The goods are then sorted during the night and the outbound trucks leave in the morning.

- Product interchangeability: The load handled at a cross-dock is in general not interchangeable. In this case, all products are dedicated to a specific destination or a specific outbound truck (pre-distribution). Information about the destination or the dedicated truck is normally known before the products arrive at the cross-dock. However, it is also possible that interchangeability of products is allowed (post-distribution). In this situation, only the type of products to be loaded on the outbound trucks and the corresponding quantity is known. When the products are interchangeable, usually some value-added activities (labeling for example) need to be performed.

- Temporary storage: In pure cross-docking, the arriving load is directly transported to outbound trucks, so no storage is needed. In practice, however, this is rarely the case. In general, the goods are temporarily stored on the floor of the cross-docking terminal or even in a (small or auxiliary) warehouse. However, it is possible that goods are not allowed to be stored. For instance, if refrigerated products have to be cross-docked in a non-cooled terminal, these products have to be directly moved from a cooled inbound to a cooled outbound truck.

2.6 Design and Development of a Cross-docking Solution

To answer questions like "When should a vehicle reach the cross-dock?", "Which dock should a container be allocated to?", "How to handle dynamics on arriving, processing and facility uncertainty?" or "What should be the sequence of exchanging of pallets between containers?", a cross-docking solution was developed, that solves the problem in three aspects [17], as can be seen in Figure 2.5:

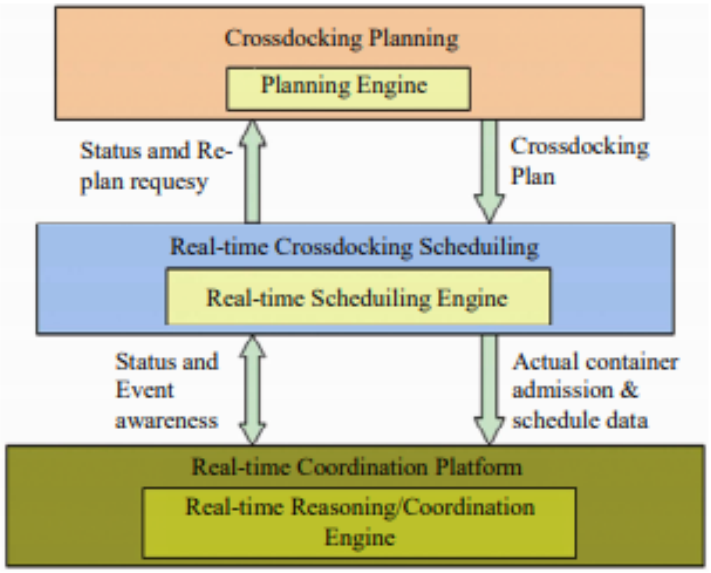

Figure 2.5: Cross-docking solution framework, represented by three layers [17]

- Planning layer - Produces the groups and set of containers based on their exchanging relationships. It also specifies the time for a container to arrive at the dock and the allocation plan to assign to those same containers so that they can complete the cross-docking operations between each other.

- Scheduling layer - Produces more detail operations schedule and task list. It also takes care of the dynamics regarding the late arrival and processing of containers and resource uncertainties in the cross-dock.

- Coordination layer - Coordinates and executes real time operations, mainly to control the resources (such as forklifts) to get tasks done and complete the exchanges of pallets.

2.7 Message Queuing Telemetry Transport

More commonly referred to as MQTT, this is a lightweight messaging protocol for sensors and small mobile devices optimised for TCP/IP networks. With the advance of the Internet of Things, the forms of communication and protocols began to require new forms of actions, which allowed for faster, lighter and more efficient connections. The message exchange is based on the Publisher/Subscriber model, which is found to be simple and lightweight. The principles of this concept are minimising network bandwidth usage, ensuring reliability and some level of guaranteed delivery, enabling communication between machines and for IoT applications. This messaging protocol was developed by IBM and Eurotech in the 1990s. The process consists of sending messages to the broker, capable of understanding and reading multiple devices at the same time. Through the TCP/IP base, the devices read the messages and take advantage of the ones the user wants. Figure

2.6 represents a schematic of how the broker works, being the intermediary for the exchange of messages.

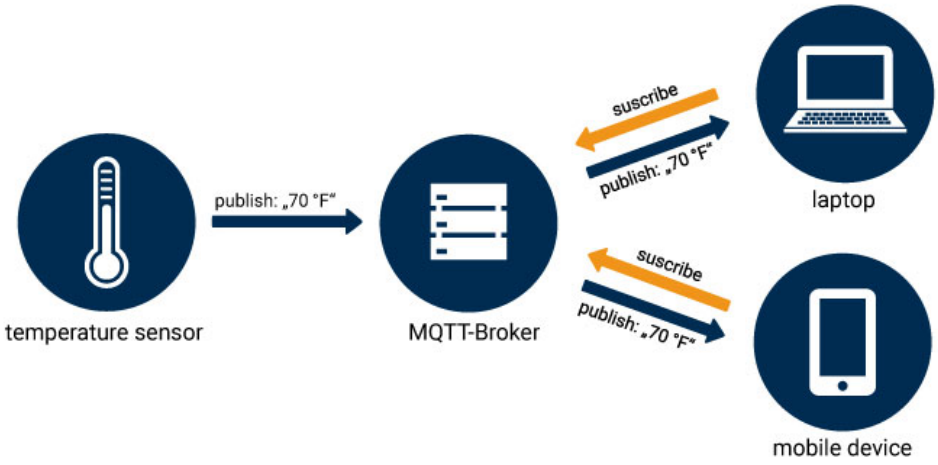

Figure 2.6: MQTT - The Standard for IoT Messaging [8]

2.7.1 MQTT Broker

It is the center of every Publish/Subscribe protocol. Depending on the implementation, a broker can manage up to thousands of simultaneously connected MQTT clients. The broker is responsible for receiving all messages, filtering them, determine who subscribed to each message and sending the message to those subscribed clients. The Broker also holds the session of all persistent clients, including subscriptions and missed messages. Another task is the authentication and authorization of clients. Usually it is extensible, which facilitates custom authentication, authentication and integration with backend systems. In short, the Broker, highly scalable and easily integrated, is the central hub through which every message must be routed [8].

2.7.2 MQTT Topic

The word Topic refers to the String that the broker uses to filter messages for each connected client. The topic consists of one or more topic levels, each one of them being separated by a forward slash (topic level separator). When comparing to a message queue, MQTT topics are very simple. Each client does not have to create the desired topic before publishing or subscribing to it. The broker accepts any valid topic without any prior initialization. Each topic must contain at least one character and the topic String allows spaces. For example, *JPM/temperature* and *MyHome/temperature* are two different topics.

2.7.3 MQTT Client

Any Publisher or Subscriber that connects to the centralized broker over a network is considered to be the client. It's important to note that there are servers and clients in MQTT. Both publishers

and subscribers are called as clients since they connect to the centralized service. Clients can be persistent or transient. Persistent clients maintain a session with the broker while transient clients are not tracked by the broker.

2.8 Node-Red

Node-RED is a tool originally created by IBM to connect harware devices, APIs and online services as part of IoT, based on flow to visual programming. It provides a web browser-based flow editor that can be used to create Javascript functions. Application elements can be saved, imported or exported, for possible re-use. The runtime is built in Node js, and flows are stored in JSON format. Processes are grouped into flows in order to fulfil a certain goal, depending on the intended input and output data. Node-RED allows you to create functionality by linking nodes together using a browser. Its popularity in the IoT area is very high, however this tool can have many other applications, such as reading data from some social networks and web forms. As can be seen in the flow of Figure 2.7, an input node reading the temperature of a house is used, connected to a *function* node (where time and date are added and the temperature is converted from Celsius to Fahrenheit. Some output nodes already existing in this tool allow sending e-mails, as well as generating reports, and publishing *tweets* without programming.

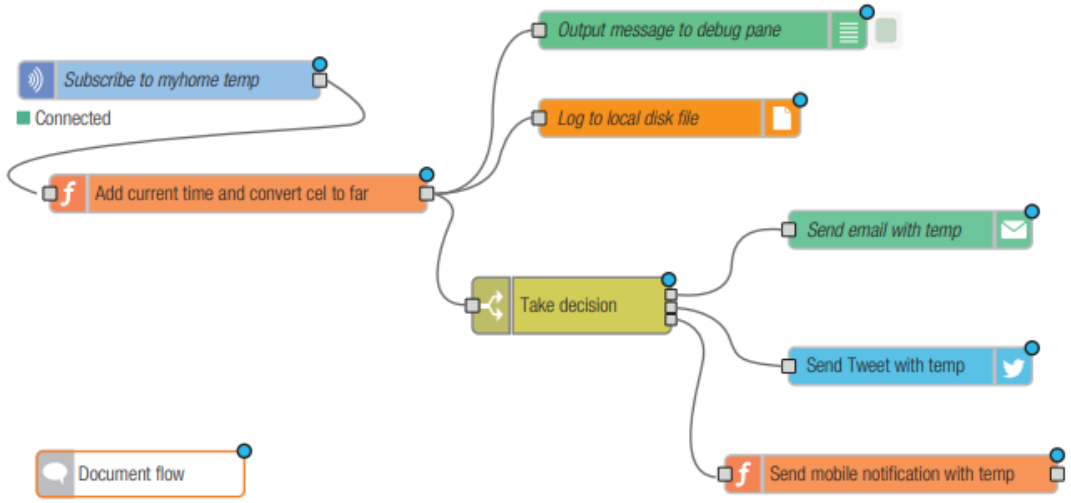

Figure 2.7: Node-RED for building IoT workflows

Besides the ease of connection between nodes using this tool and the possibility of the integration of already existing flows, there are some quite important nodes for the creation of an application that integrates concepts such as the IoT and Industry 4.0. The following enumeration concerns the main nodes used in the course of the development of the *CrossLog* project, and the detailed explanation of why they were used.

- Default Node Red nodes:

  - Inject - can be used to manual trigger a flow by clicking the node's button within the editor, or can be used to automatically trigger flows at regular intervals. The message sent by this Inject node can have its payload and topic properties set. The payload can be set to a variety of different types:
    * Flow or global context property value.
    * String, number, boolean, buffer or object.
    * Timestamp in milliseconds.
  - Debug - can be used to display messages in the Debug sidebar, that provides a structured view of the messages sent, making it easier to manipulate and explore them.
  - Function - this node allows JavaScript code to be run against the messages that are passed through it [8].
  - Switch - allows messages to be routed to different branches of a flow by evaluating a set of rules against each message. It will route to all outputs corresponding to matching rules. But it can also be configured to stop evaluating rules when it finds one that matches (within the scope of the project to be developed, this node is used to find messages containing certain strings).

- Installed Node Red nodes:
  - Message Queuing Telemetry Transport (MQTT):
    * Broker MQTT - element responsible for managing the publications and subscriptions of the MQTT protocol. It works as a mediator between the machines, allowing communication. It is capable of reading multiple devices at the same time, in a safe manner and with high quality levels.
    * Publisher - publishes a message to a topic on a MQTT broker. The destination functions as an MQTT client that publishes messages, writing each record as a message.
    * Subscriber - subscribes to a topic on a MQTT broker to read messages from it.
  - OPC UA:
    * OPC UA Client - communicates with the CODESYS OPC UA Server, where the address and port of the OPC UA server is set as *Endpoint*. It can be defined in the operations whether the action is read or write.
    * OPC UA Item - specifies the *NodeID* of the variable to be accessed. In the specific case of this project, this information is collected from the *UaExpert* software, which in turn reads directly from the programmed PLC.

2.9 Programmable Logic Controllers

Designated by PLC, they are used to control complex industrial processes, easily programmable and flexible to the operator´s requirements. They are also applied in data collection functions of

the control system. They're an upgrade over the old relays and timers previously used to control industrial machinery since PLCs are capable of performing much more complex tasks [30].

PLCs can range from small modular devices with tens of inputs and outputs (I/O), in a housing integral with the processor, to large rack-mounted modular devices with thousands of I/O, and which are often connected to other PLCs and SCADA systems. They can be designed for many arrangements of digital and analog I/O, extended temperature ranges, immunity to electrical noise, and resistance to vibration and impact. Programs to control machine operation are typically stored in battery-backed-up or non-volatile memory. There are numerous advantages to these Programmable Logic Controllers, such as their small physical size, reliability, cost effective for controlling complex systems, less and simple wiring, flexibility, fast response, remote control capability and ease of maintenance. However, only one program at a time can be operated in a compact PLC, and it´s not possible to use software and parts of one PLC brand in another PLC brand. Also, when power restores, PLC automatically starts. It can damage the system, and to avoid it, it´s conceivable to program the output to go to the fail-safe mode. In conclusion, these are the limited disadvantages of the PLC over the old relay system [6].

2.9.1 PLC Programming

When using a PLC, it is important to design and implement concepts depending on the particular use case. The program of a PLC consists of a set of instructions, in textual or graphic form, representing the logic that is to be implemented. There are two types of languages, which in turn are subdivided into other types:

- Textual Language:
  - Instruction list.
  - Structured text.
- Graphical Form:
  - Ladder Diagrams (LD) (i.e. Ladder Logic).
  - Function Block Diagram (FBD).
  - Sequential Function Chart (SFC).

Although all of these PLC programming languages can be used to program a PLC, graphical languages (like ladder logic) are typically preferred to textual languages (like structured text programming).

2.10 Supervisory Control and Data Acquisition

SCADA is a system of software and hardware elements capable of controlling industrial processes locally or at remote locations, monitoring, gathering and processing real-time data, interacting

with devices such as sensors, valves, pumps, motors and Human-Machine Interface (HMI) software and recording events into a log file. SCADA systems are crucial to the industry as they help maintain levels of efficiency and indicate the emergence of problems so they can be quickly mitigated. The basic architecture of a SCADA system starts with PLCs or RTUs. These two units are microcomputers that communicate with a set of objects such as machinery, HMIs, sensors and other devices, and then forward that information to computers with the SCADA software. This system processes, analyzes and visually disposes of the data, helping operators and other workers to make important decisions. For example, a SCADA system quickly notifies an operator if there is an error or malfunction of a component on the factory floor, so that the problem can be resolved and unnecessary costs prevented. Figure 2.8 represents a basic SCADA diagram, illustrating the connection between the physical units, such as sensors, PLCs and display software.

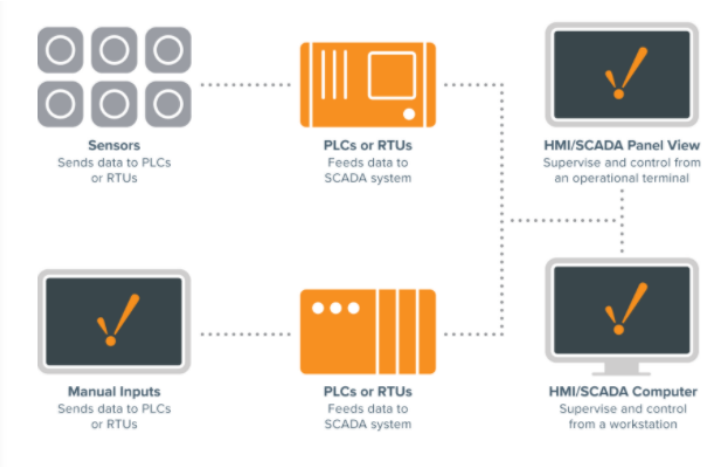

Figure 2.8: A basic SCADA diagram [1]

SCADA systems are used in industry, in order to control and lead to intelligent decision making. SCADA systems are the backbone of many modern industries, including energy, food and beverage, manufacturing, oild and gas, power, recycling, transportation, water, and many others. Also, most modern SCADA designer applications have rapid application development (RAD) capabilities that allow users to design applications relatively easily, even if they don't have extensive knowledge of software development [1]. It is possible to connect SCADA systems to databases, providing great advantages when it comes to integrating MES and ERP systems, allowing a flow of information accessible to the entire company. The insertion of elements in databases provides a continuous analysis of them, reducing errors.

2.11 Manufacturing Execution System

Manufacturing Execution System, or MES, is an information system that has as objective the execution of industrial operations. The main objective of such systems is to achieve and maintain

high levels of performance in a highly competitive environment and flexible to process requirements. For this to happen, the MES has to perform a set of tasks at the factory floor, company organization, transportation network and supply chains [20]:

- Monitoring and enforcing the correct execution of the production process.
- Monitoring and controlling the material used in the production process.
- Gathering information about the production process.
- Providing the tools for the analysis of the data to optimize efficiency.
- Delivering and managing work-instructions.
- Providing the tools to solve problems and optimize procedures.

There are some key benefits of MES, achievable in short time, to increase efficiency and reduce costs, such the reduction of: manufacturing cycle time, order lead time, direct labor costs, data entry time and work in progress (WIP) inventory. In order to achieve long-term improvements, this system helps increasing customer satisfaction, regulatory compliance, agility and supply chain visibility.

2.12 Manufacturing Execution System vs Enterprise Resource Planning

Both systems are critical for industrial processes. In recent years, the link between Enterprise Resource Planning (ERP) and Manufacturing Execution System (MES) has been established and its integration has proven valuable. The confusion still lies in deciding which processes should be supervised by which entity. MES is the ideal choice for complex production processes with multiple variations and transactions. ERP is generally designed to support homogeneous processes with management information flow [20]. Figure 2.9 illustrates hierarchically the relationship between the MES, the ERP, the PLCs, and the SCADA system.

Figure 2.9: MES vs ERP [20]

2.13 Scalable Vector Graphics

Commonly known by its abbreviation SVG, it is an XML language for describing two-dimensional graphics or drawings in vector form, whether they are static, dynamic or animated. The great advantage of this type of file is that it is an open format and is not owned by any company, having been created by the *World Wide Web Consortium*. SVG is supported by all modern Web browsers natively or through JaVaScript libraries. At the moment, this format type allows for three types of graphic objects:

- Vector geometric shapes - mathematically described lines and curves.
- Images.
- Text.

An example of SVG code, here representing a simple white circle with a black border, is as follows in Listing 2.1:

```
1 <circle cx="100" cy="100" r="50" stroke-width="4" stroke="#000" fill="#fff" />
```

Listing 2.1: Code Example

As can be seen, SVG documents are nothing more than simple plain text files that describe lines, curves, shapes, colors and text. As it is modifiable and understandable, when embedded in an HTML document as an inline SVG, its code can be manipulated via CSS or Javascript. This gives SVG a flexibility and versatility that can't ever be matched by traditional PNG or JPG graphic formats. It uses shapes, numbers and coordinates rather than a pixel grid to render graphics in the browser, which makes it resolution-independent and infinitely scalable. It's possible to combine different shapes, paths and text elements to create all kinds of visuals [12].

Therefore, and applying the use of this type of document format in this project, it is possible to create representations of the shop floor, AGVs and change their state according to the value of variables. For example, a variable called *state* (boolean) can acquire the value 0 or 1. Depending on this value, its colour changes from green to red and vice-versa. It is also possible to change its x and y position by modifying its inherent location characteristics. All this freedom in manipulating SVG properties enables a customisable representation according to the requirements of each application. The following figure shows a simple demonstration of how changing the value of a Boolean variable changes the value of the "fill" attribute, that sets the color inside the object. Color names can be written in RGB, hexadecimal or RGBA values. Figure 2.10 represents a simple example of how SVG properties can change their look.

Figure 2.10: Simple example of SVG property change depending on variable value

2.14 Web Application Authentication

One of the most important aspects when creating a web application is its security and the protection of sensitive information should be one of the top priorities. This becomes especially significant as data grows and data breaches are increasingly common. Attempted illegality and theft of information is becoming more and more common in large companies due to the increased value of information. Therefore, an application needs to be secure through authentication practices in order to increase user trust. These practices prevent hacker access to the administration level and other user roles that an application may have. A company should regularly test its security systems and evolve them.

2.14.1 Gathering Information

The first step in creating a secure application is to recognise what data needs to be protected, where it is stored and how information is transferred between applications that a company may use. Even if one of the applications is secure, if the others are not, the whole security and data protection may be at risk. Start by pointing out areas in each platform that might be at risk and determine which areas of the application can be tested, whether manually or by using a security tool. [27].

2.14.2 Planning and Execution

In this step, once the risks associated with each application are defined, it is necessary to document a testing strategy. For example, if an application has an authentication component, password complexity guidelines should be checked and vulnerability to brute-force attacks, among other security factors. Once you have outlined and distributed the tasks regarding the security of the system, you can move on to the execution phase, where each application will be tested through automatic and manual checks to ensure everything is in order. Manual testing should cover common vulnerabilities, and in most cases testing is done by hacking into the website and identifying areas that need protection [27]. Automated security checks, on the other hand, utilize computer programs to locate more obscure vulnerabilities. A machine can quickly scan through your application to pinpoint risks, but they often lack the ability to spot logical flaws. By combining the two processes, you'll likely be able to discover most, if not all, potential liabilities.

2.14.3 Web Application Authentication Methods

Authentication, as mentioned earlier, is a process of verifying an identity. A unique identification is associated with a user id or username [9]. Traditionally, the combination of a user with a password allows the authentication of a system. If authentication is local, the common flow when implementing it is:

- Username/Email/Mobile register.
- The application stores user credentials in the database.
- The application sends a verification message to validate the registration.
- Post successful registration, the user enters credentials for logging in.
- On successful authentication, the user is allowed access to specific resources.
- The user state is maintained via Sessions or JWT.

Users are recommended to use different passwords for different applications and websites in order to protect their identity. However, in reality, what happens in most cases is the reuse of these passwords, which leads to greater vulnerability in the event of a cyber attack. To reduce the risk, a multi-factor authentication system can be implemented: a user authenticates through two or more factors, as we can see in Table 2.1.

Table 2.1: Authentication factors examples

Factor	Example
Something the user knows	Passwords, PINs, TAN, security questions
Something the user has	USB keys, Software tokens, certificates, email, SMS, phone calls
Something inherent to the user	Biometrics (fingerprints/iris scans, facial recognition, key pattern interval)
Location	Source IP ranges and geolocation

2.14.4 Token-Based Authentication

Token authentication is a protocol that verifies the identity of the user, and returns a unique access token. During the life cycle of the token, the user is granted access to the application or website they requested access to enter, rather than re-entering credentials. Authorised tokens work like a stamped ticket, where the user has access as long as it is valid. Once he logs out, the token becomes invalid. Using a token-based authentication system, visitors will verify credentials just once. In return, they will get a token that allows access for a time period defined by the software developer [23]. The process follows some steps:

1. Request - The user asks for access to a server or protected resource. That could involve a login with a password, or another process.

2. Verification - The server determines that the person should have access. This involves checking the combination of a username and password, or it could involve another process.

3. Tokens - The server communicates with the authentication device, like a ring, key, phone, or similar device. After verification, the server issues a token and passes it to the user.

4. The token sits within the user's browser while work continues.

If the user attempts to visit different parts of the server (for example administration pages), based on the user route priorities, after the token communicates with the server again, access is granted or denied based on the token. Administrators set limits on tokens. They can allow a one-use token that is immediately destroyed when the person logs out, or they can set the token to be self-destructible at the end of a specific time period [23]. Figure 2.11 illustrates how the Client communicates with the Server and how users get a token.

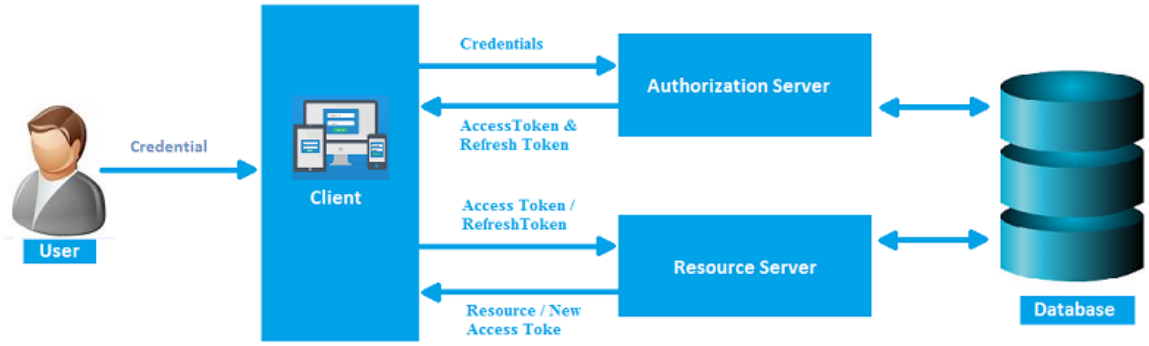

Figure 2.11: Token based authentication for Web APIs [15]

There are many benefits of JSON Web Tokens, like the size (tokens in this code language are tiny, and they can be passed between two entities quite quickly), ease (tokens can be generated from almost anywhere, and they don´t need to be verified on your server) and control (the administrator can specify what someone can access, how long that permission lasts, and what the person can do while logged on). However, there are also some potential disadvantages, such as JWTs rely on a single key, which means that if the key is compromised, the entire system is at risk, these tokens are not simple to understand (if a developer does not have a strong knowledge of cryptographic signature algorithms, they could inadvertently put the system at risk) and the administrator can't manage clients from the server side.

2.15 Manufacturing Dashboard

A manufacturing dashboard is a real-time visual representation of certain processes. They combine graphs, tables, and other visualization techniques to make production KPIs easy to understand. They have the duty to organise the data coming from machines, sensors, devices, and workers into easy-to-read, instantly available breakdowns that the whole operation can reference. The

information must be correctly scaled and grouped so that any user perfectly understands what data is being transmitted and what functions he has (reporting problems, sending messages to other departments, interacting with the shop floor and other subsystems).

2.15.1 Mission Control

The information must be available to get a sense of how production is going on a given day. From there, it should be possible to compare that same day to others within the same week/month/year. Some important information to make available should be:

- Operation conditions (temperature, noise, or other ambient conditions).
- A breakdown of how each operator has performed during a period of time.
- Process variables daily count.
- Line-specific breakdowns, with options for clicking to new dashboards with more detailed information.

2.15.2 Shop Floor Overview

Shop floor overview dashboards provide a bird's eye view of production. They begin with a schematic or a floor plan, and overlay critical product data, and layer information about cell, machine, or plant performance on top. These pages can help tracking materials from the moment they arrive to the moment they leave. It is possible to quickly identify bottlenecks, and track exactly when unplanned downtime slows a certain process. Some important information that should be made available is:

- Whether or not machines are occupied or free.
- How much finished product is ready to go out the door.
- Buffer levels (color coded to make it easier to spot replenishment needs) [14].
- Activity or inactivity of each subsystem.
- Details about certain components, such as their status, speed, position and battery level.

Figure 2.12 represents an example of the factory floor, representing the state of its different modules through colors and the value of the process variables.

2.15.3 Alarm/Event Overview

An alarms and events page is crucial, especially for engineers and operators who want to effectively visualise what is happening on the shop floor, should any mishaps arise. Customised for each type of user, the visualisation of alarms and events facilitates daily decision making, reducing possible fault identification times. Once the problem is identified, users can visualize the

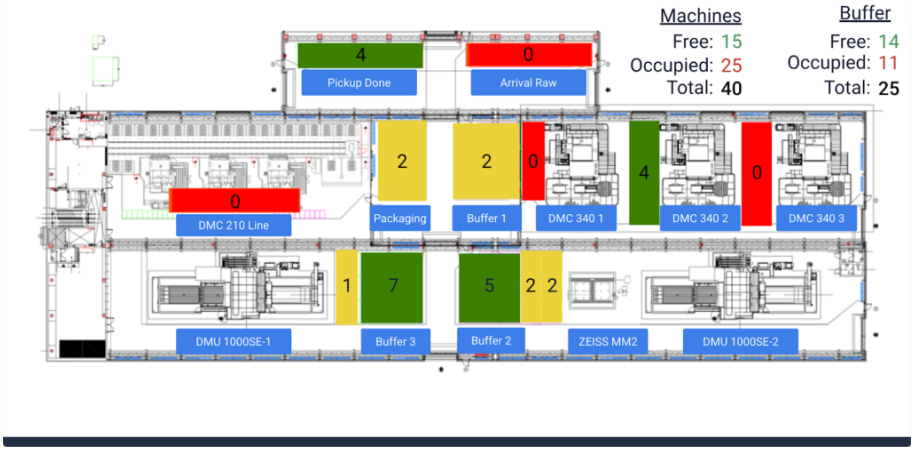

Figure 2.12: Shop floor overview [14]

source of the problem, which might be an oven belt speed malfunction or a problem with an order. Each subsystem must have the possibility to report problems or failures, so that this alarm appears in the table of alarms/events and in the corresponding graphics. The table mentioned above must have:

- Date and time of incidence.
- Type of Alarm/Event.
- Small problem description.
- Link to the resolution of each possible problem.
- Location.
- Indication whether the problem has already been solved or not.

2.16 Summary

The IoT concept harbours with it a new vision where devices not only communicate with each other and with humans, but are also aware of their environment. This interaction encourages better management of resources and increased efficiency of processes. The use of Visual Programming Languages in this area allows the abstraction of low-level concepts and highly heterogeneous architectures that these systems have. Allied to the emergence of the cross-docking concept, a strategy of minimal handling of loads, with little or no storage between unloading and loading, the *CrossLog* project emerged, which proposes to investigate and develop an automated and collaborative system, capable of moving and managing the flow of products within the warehouse in a fast and safe way. This project, consisting of a set of different processes and objectives, integrates the concepts mentioned in this chapter in order to create a new product in the industrial market and consequently change people's life. The creation of a safe and reliable application for management

and process control purposes is fundamental in order to monitor what is happening in real time on the factory floor, receive and provide instructions to other systems that are part of the global process, visualise and change the status of objects and prevent the late detection of possible errors and malfunctions. The impacts of Industry 4.0 are already visible in all sectors, and companies are anticipating and preparing their industry and products for this new market demand. Concepts such as data collection, automation of equipment and robots, and data analysis are all interconnected in a constant transformation of industry, where decisions have a smaller and smaller percentage of human involvement. A combination of systems, IoT and artificial intelligence make Industry 4.0, and this project in particular, more efficient, productive and with less waste.

Chapter 3

Proposed solution

This chapter aims to introduce the project in which the development of this dissertation is inserted, as well as main requirements and ideas for a safe and effective implementation. It is divided into objectives and architecture of the overall system and also main points regarding the Management and Supervision module.

3.1 Objectives

The *Crosslog* project proposes as its main objective, given the need mentioned in the previous chapter, to study, develop and implement an automated cross-docking system capable of moving and managing the flow of products within the warehouse in the fastest and safest way. The manipulation of objects will be carried out using a set of modules (industrial conveyors, highly advanced robotic units and artificial vision systems) that interact with each other and communicate with planning and management software. This project is a revolutionary opportunity, breaking with the current concept of automated distribution centres of today, whose proper functioning is based on static automated warehouses. It also represents a change in cross-docking distribution centres, as current systems are highly dependent on the human hand due to their ability to interact with numerous product variations. The automation of this process will be a new milestone in the logistics sector, where unpredictability has proven to be a major obstacle to evolution. The project should result in an innovative concept and a technological kit (materialised in a set of prototypes, both software and hardware), which allows:

- To present an alternative to manual cross-docking systems in order to respond to the growing demand from this sector.
- To generate an automated and modular system capable of managing and moving products within a logistics centre, throughout their entire route.
- To consolidate the existing market and enter new markets with a high degree of differentiation, by providing a disruptive solution that will give its customers the ability to reduce product storage times and increase the efficiency of distribution centres.

In order to answer these general objectives, a set of specific objectives is also listed, in which the module developed in this dissertation (management and monitoring software) is included:

1. Reception, identification and weighing station:
    - As the products arrive at the logistics center, they are placed manually or automatically on a conveyor, which fits them to the control station and "catalogs" all the products. Depending on the orders for that day (communication with a higher planning system), they are taken to the dynamic accumulator unit or to the warehouse.
2. Dynamic accumulator unit:
    - Products are prepared to be palletized or forwarded to the warehouse. After having already been labeled by the previous unit, they are correctly arranged according to the following operations.
3. Mobile palletizing unit:
    - Composed by mobile robots, which have flexible grippers that enable the safe handling of different types of products.
    - The robot will start palletizing the products in the order they are made available. As soon as a pallet ends, it moves to the next station where palletizing is necessary and the cycle is repeated.
4. Progressive involvement station:
    - Parallel to the final palletization process, each pallet is wrapped in plastic film, ensuring more consistency and stability.
5. Three-dimensional mosaic generation software:
    - Virtual pallet builder, according to the orders received, product specifications and availability of process variables.
6. Planning software:
    - Responsible for calculating the trajectory of each product and sequence of orders.
7. Integrated Supervision and Management System (SSGI) (module developed within this thesis):
    - Responsible for controlling and monitoring products, orders and processes.

Importantly, for each of the outcomes, proof of concept and functionality validation will be done through different prototypes, first in a laboratory environment and later in a relevant and even operational environment. As the completion of this dissertation is earlier than the completion of the *Crosslog* project, the proposed solution of the SSGI module is adapted to the existing conditions and physical objects. Figure 3.1 represents a visual model of the system, and the reproduction of the flow of activities

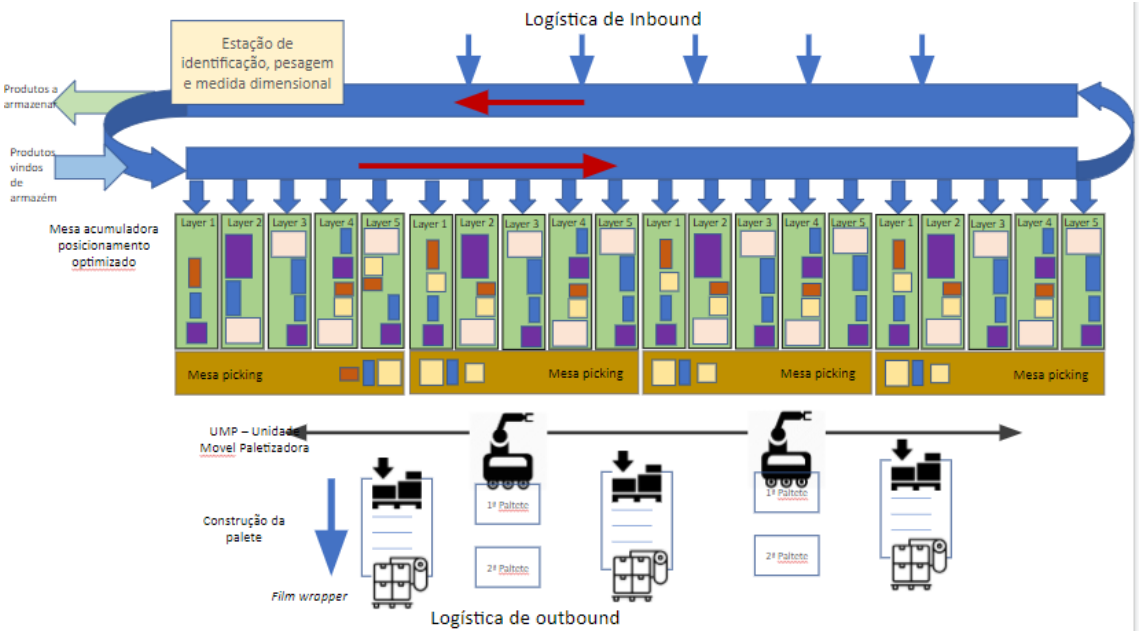

Figure 3.1: Concept diagram [13]

3.2 System Architecture

The architecture of the software to be developed is illustrated in Figure 3.2. The specific module on which this thesis is focused is in color.

Starting from a set of input data coming from Enterprise Resource Planning (ERP) and Warehouse Management System (WMS), namely:

- Outbound needs list - list of products and quantities (with dates and priorities) for the next day.
- List of products in current stock and stock profile (in the short term, considering expected inbounds and outbounds).

The system consolidates the requirements and transmits them to the suppliers, keeping them informed about the expected inbound flows. Based on this information, the system plans the logistical operations for the following day, evaluating capacity and suggesting necessary adjustments, namely in terms of human resources. The solutions obtained seek to optimize the flow and avoid blockages in the accumulator, minimizing logistics movements and the need for possible temporary storage of products [13]. The approved plan, which includes the planned sequence of pallet load formation in the accumulation areas, is subsequently made available to the Integrated Supervision and Management System (SSGI), through the publication of that information on the IIoT platform in a predefined format. Based on this information, the SSGI analyzes and monitors the plan's execution and suggests corrective measures, such as increasing the number of operators when in need, reorganizing the priorities of daily orders, among other activities mentioned

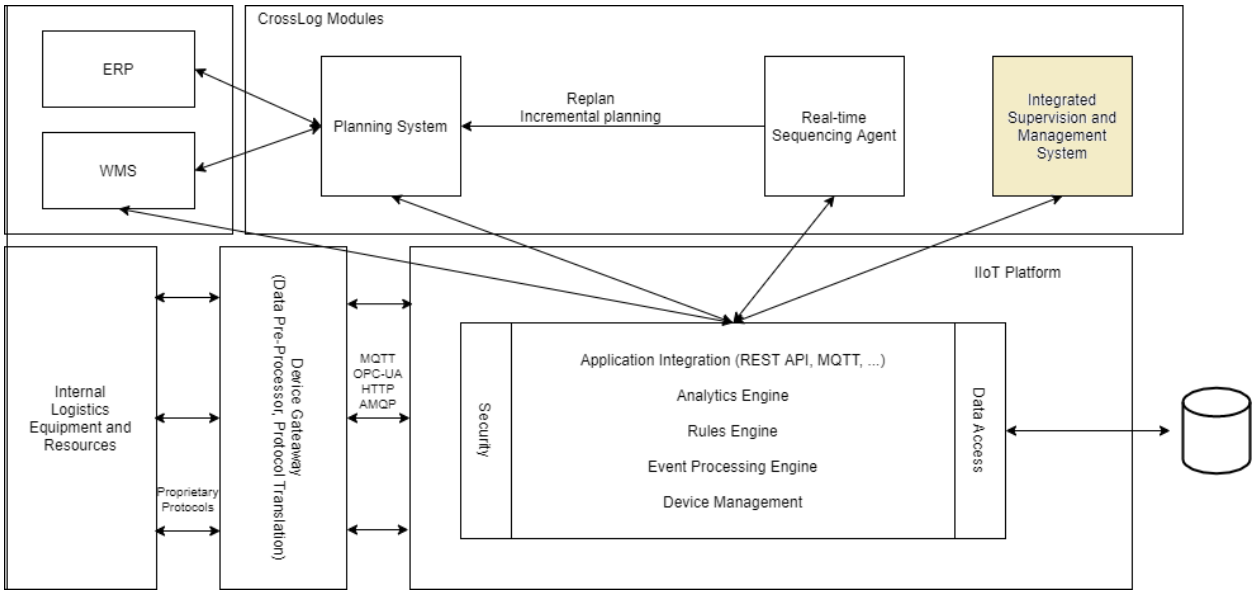

Figure 3.2: Software architecture - global system

in the next chapter. All equipment and internal logistics resources will be Industry 4.0 components that will communicate with each other and with the SSGI through the IIoT platform. The communication between the equipment and logistic resources and the IIoT platform will be done through the OPC Unified Architecture (OPC-UA) protocol, which is the standard recommended by RAMI 4.0 to implement the communication layer. RAMI 4.0 is a model for applications of connectivity solutions for projects adhering to Industry 4.0, enabling a cyber ecosystem of the entire production chain [33]. The products, corresponding to a certain order, are delivered to the Dynamic Accumulator Unit, according to the order defined by the Multiformat Mosaic Generator and Planning System, in order to build the pallet optimally. As the products arrive, an order is issued for one of the mobile palletizing units to collect the stored products and transport and deposit them on a pallet. The different logistical tasks will be assigned to the Mobile Palletizing Units by the system controller, taking into account their effective availability and capacity, if they have different characteristics. For each product, having an origin and a destination, its path (position in the accumulator, palletization order and movement made) is calculated and controlled in real time, ensuring total traceability of the product at every instant. The product passes to the virtual domain where all the information is integrated into a single technological platform, contributing to an intelligent environment capable of making optimized decisions.

3.3 Integrated Management and Supervision System

This thesis aims to investigate and create a SCADA system that presents features for changing the industrial environment and real-time visualization of data from the IoT platform. This data results from the system's logic controllers (PLC) and the Multiformat Mosaic Generator. The communication should be carried out in OPC Unified Architecture (UA): a machine to machine

communication protocol used in industrial automation. The system must communicate with all modules, to send or receive information. Beyond these points, the system should provide:

- Authentication system for operator/administrator.
- A virtual image of the status of some of the sensors and actuators that make up each of the system modules.
- A virtual image of each subsystem, and its process variables.
- Operation commands.
- An alarm/event log.
- An order list/ Order history:
  - Possibility to change priority and cancel orders.
  - Order status.
  - Percentage of conclusion.
- Each order details:
  - Order ID.
  - Number of pallets associated.
  - Client.
  - Pallet buffer ID.
  - Estimated time/date to finalize order.
  - Actual time/date of order conclusion.
  - Status (Not started, In progress, Complete).
  - Percentage of pallet progress.
  - Pallet 3D visualization.
- Problems report.

3.3.1 Integrated Supervisory and Management System Architecture

The system in question communicates with modules that are important for its proper functioning. As it is part of a project still under development (*CrossLog*), there are modules that cannot be tested in time, and for this reason some alternatives will arise to circumvent existing problems, as will be analyzed in the underlying topics. The following Figure 3.3 illustrates the architecture of the developed application.

Although some connections, in practice, are not possible due to the still developing modules (namely Planning System and Mosaic Generator), the following communications stand out:

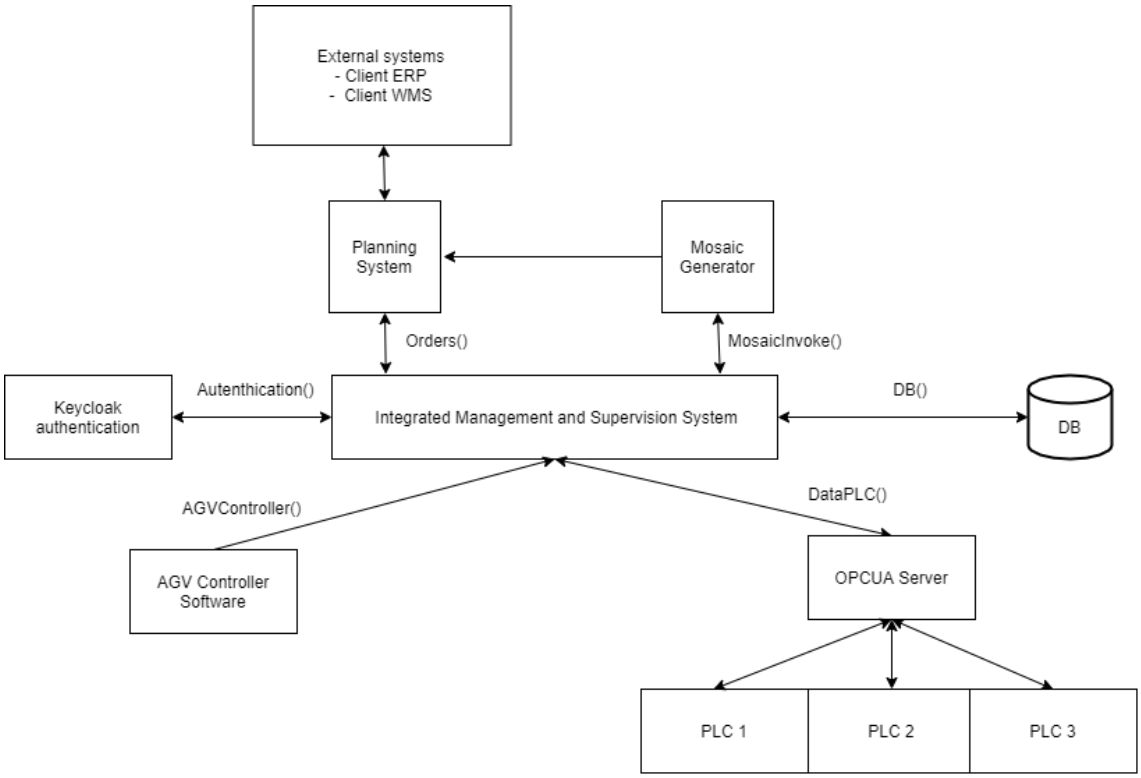

Figure 3.3: Architecture - Integrated Management and Supervision System

- Orders() - The Planning System invokes the function daily to receive the sequence and priority of orders, products and pallets. Sends information to the Planning System if the administrator changes priorities or cancels orders.
- Authentication() - The application, which needs a correct login/password key, communicates with the *Keycloak* tool for verification of authentication data and authorization for certain pages.
- DataPLC() - The application communicates with the UaExpert software, which in turn collects information directly from the PLC. Sends and receives information about the state of the physical systems and process variables.
- DB() - Stores order sequence, order details, products, pallets, alarms and events.
- MosaicInvoke() - Receives a JSON message of the coordinates of the products in the pallet corresponding to a given order, for a 3D pallet view, available to the administrator or operator.
- AGVController() - Sends commands about shop floor positioning, status, battery levels and Gripper positioning.

Chapter 4

Implementation

In this implementation chapter, points regarding how to implement and fulfill the main requirements of the work are reviewed, accompanied by figures and tables for better understanding. It should be noted that there are several ways of implementation, and that the chosen strategy follows some recommendations given by the project manager and work coordinator.

4.1 Implementation Tools

For the development of this project, several tools were used in order to create a sustainable application capable of meeting the requirements initially established. The following table demonstrates in a simple way why each tool was used, both software and hardware. During this Implementation chapter, each of these tools will be discussed in more detail, as well as their importance in the project. Table 4.1 shows the tools used during the development of the project and their respective purpose.

Table 4.1: Project implementation tools

Tool	**Purpose**
Node-Red	Backend of the Web Application
VueJs	Frontend of the Web Application
Keycloak	Web Application Security (Authentication and Authorization)
TiaPortal 15.1	PLC Programming
UaExpert	PLC Client
SIMATIC S7-1500	PLC Server
Figma	Mockup builder
Overleaf	Document writing
SVG Creator	SVG file creator
MongoDB	Database

4.2 Application Mockup

Initially, and in order to visually demonstrate how the information would be organized and how the navigability between the pages of the application would be, a mockup of the system to be developed was created. Figure 4.1 illustrates, respectively, from left to right, the home page, the complete system's factory floor, the palletization page and the alarms and events page. The tool used was Figma, a vector graphics and design prototyping editor. The completion of this task and its presentation to JPM and internal coordinators allowed to receive some feedback and suggestions on how the information could be better organized and easily accessed by users. The colors and fonts used meet the essence of the company and are allied to the attempt to create a user friendly, secure and reliable application.

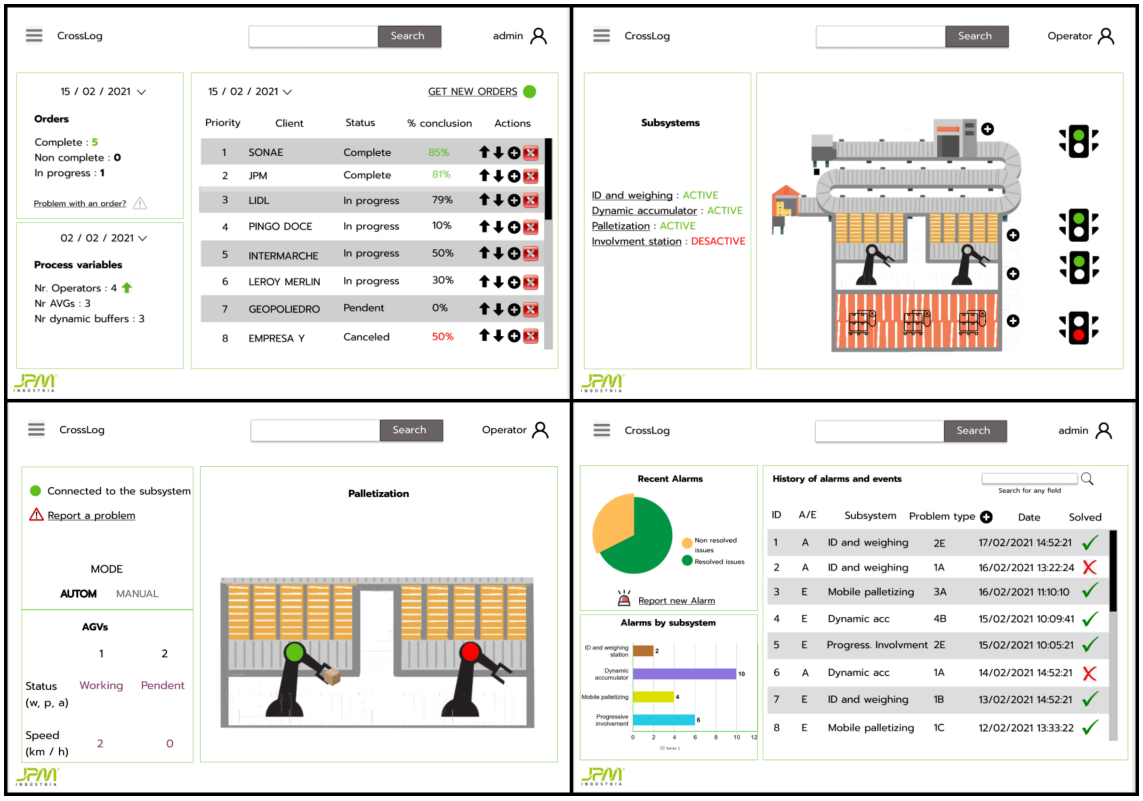

Figure 4.1: Web Application Pages Mockup

The previous figure represents the design of some web pages. The initial idea, and that was later implemented, was for the application to have an authentication system that would redirect the user to the home page. This would contain a menu with some navigation options, as well as a logout button. The functionality of each page will be discussed in more detail in the "Application Development" section.

4.3 Authentication

4.3.1 Frontend

Security in an application is key to giving it high levels of reliability and data protection. In order to protect the frontend of the Integrated Supervision and Management Module application, a tool named *Keycloak* was used. It consists of an open source software to work together with the application in more common authentication and authorization implementations. This technology is a robust solution developed by *Red Hat*, and follows standard security protocols. The main objectives with its use are the authentication of different user types, namely administrator and operator, whose tasks and experience of interaction with the application are different. As it is posible to see in Figure 4.2, the application works as a client, and after inserting the login data, if they are correct, the *Keycloak* tool returns a token, valid for a previously defined time, which allows or not the user to access certain sections of the application.

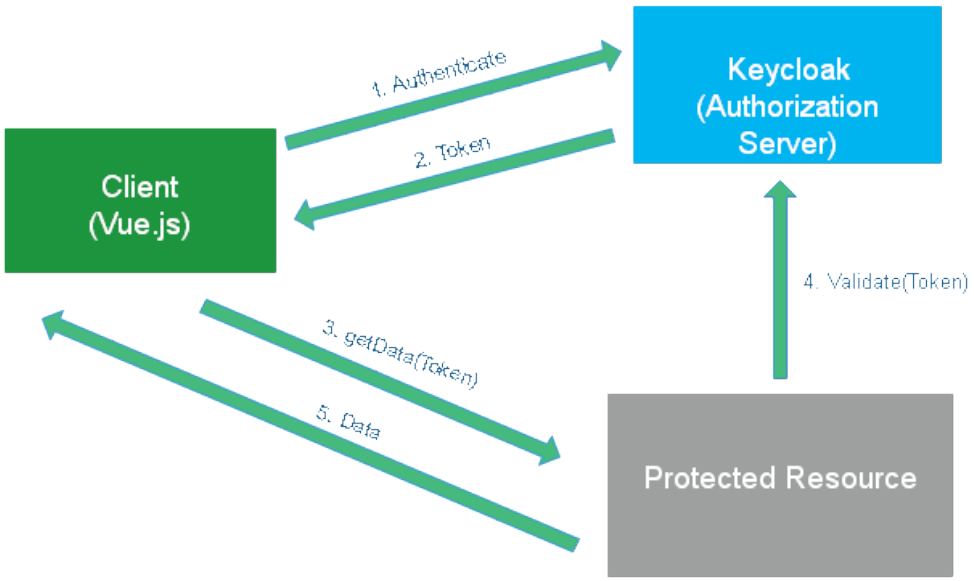

Figure 4.2: Access to Protected Resources using *Keyloak* tool [18]

Regarding the configuration of the *Keyloack* server, first it is necessary to access the administrator's console, in order to:

- Add a Realm and configure a client - In this case, as the client is browser-based JavaScript, it can not be hidden [4].

- Select valid Redirect URIs - Where should it redirect to after a successful login or logout. In this case, and because the application is initiated on port 4000, this field is "http://(machine_ip):4000/*".
- Select root URL - Must be the address where the *Keycloak* tool is running. In this case, "https://(machine_ip):8080/*".
- Choose token characteristics - It is possible to define the idle time (time when the application is not being used) before the session expires, login action timeout and access token lifespan.

All other features of this tool have not been changed, and remain in default mode. Once the server part is set up and working properly, it is then necessary to integrate this tool into the VueJs application. Normally, to perform authentication, a login function is needed. However, there are two options available to authenticate: "login-required" or "check-sso" in the *init* function. Login-required will authenticate the client if the user is logged in to *Keycloak* or display the login page if not. Check-sso will only authenticate the client if the user is already logged-in. If the user is not logged-in the browser will be redirected back to the application and remain unauthenticated [7]. In this case, the first option was used, as it can be seen in Listing 4.1.

```
1     init: {
2     onLoad: 'login_required',
3     },
4     config: {
5     url: 'http://192.168.40.231:8080/auth',
6     clientId: 'vue-teste-app',
7     realm: 'keycloak-crosslog'
8    }
```

Listing 4.1: Init Function settings code

The fields present in the *config* function are the same as those previously defined in the administrator's console. All other parts of the code needed to integrate this tool into the application are attached. It was also necessary to modify the file that configures the page routes, i.e., the navigability settings, so that the login page belonging to the *Keycloak* tool would initially appear, and only after correct authentication would the user be redirected to the Home Page. It is necessary to understand that the code present in this document for the *Keycloak* integration is unique to VueJs applications, which does not preclude its integration into other applications with other programming languages, after the necessary changes have been made.

4.3.2 Backend

By default, the Node-Red editor, used as the backend in the creation of this application, is not secure. This means that anyone who can access the IP address can easily access the editor and deploy changes. This is only suitable if the application is running on a trusted network. However, to enable user authentication on the administrator console and editing page, it is necessary to

uncomment the *adminAuth* property in the settings file, located in the Node-Red installation folder, and insert new users, as it is possible to observe in Listing 4.2.

```
    adminAuth: {
    type: "credentials",
    users: [
        {
            username: "admin",
            password: "$2a$08$zZWtXTja0fB1pzD4s(...)",
            permissions: "*"
        },
        {
            username: "francisco",
            password: "$2k$i8kwoAcPoKsltN27aeFq(...)",
            permissions: "read"
        }
    ]
}
```

Listing 4.2: Enabling and creating Node-Red users

The users property is an array of user objects. This allows to define multiple users, each of whom can have different permissions [21]. This above configuration example defines two users. One called *admin* who has permission to do everything within the editor, the other called *francisco* is given read-only access. Note that the passwords are securely hashed using the bcrypt algorithm. To perform such encryption, a set of commands are required. The documentation needed to perform such steps is described in [21].

4.4 Database

A database must be flexible, easily accessed, and it should meet the specifics of the project at hand. In this case, some important points were taken into account for the correct choice of it. First, it would not be used to store user data and secret information such as passwords, since this functionality is already allocated to the *Keycloak* tool. The information from the Planning System, such as the coordinates of the products on the pallets and also the list of orders with their respective priorities, would theoretically arrive to the application in JSON format. To store the information in such a format and further process, the database would need to be flexible at this point. *MongoDB* was then chosen, a Not only SQL (NoSQL) database that supports creating explicit schemas and it is a non relational database provider.

According to the *CrossLog* project, and as it can be seen in Figure 4.3, tables were created to store information and easily make it available to the application, allowing the visualization of variables and the transmission of information between subsystems belonging to the project.

In order to cross-reference the information and make it more organized, relationships were created between the tables in the database. The use of the tables follows a certain pattern:

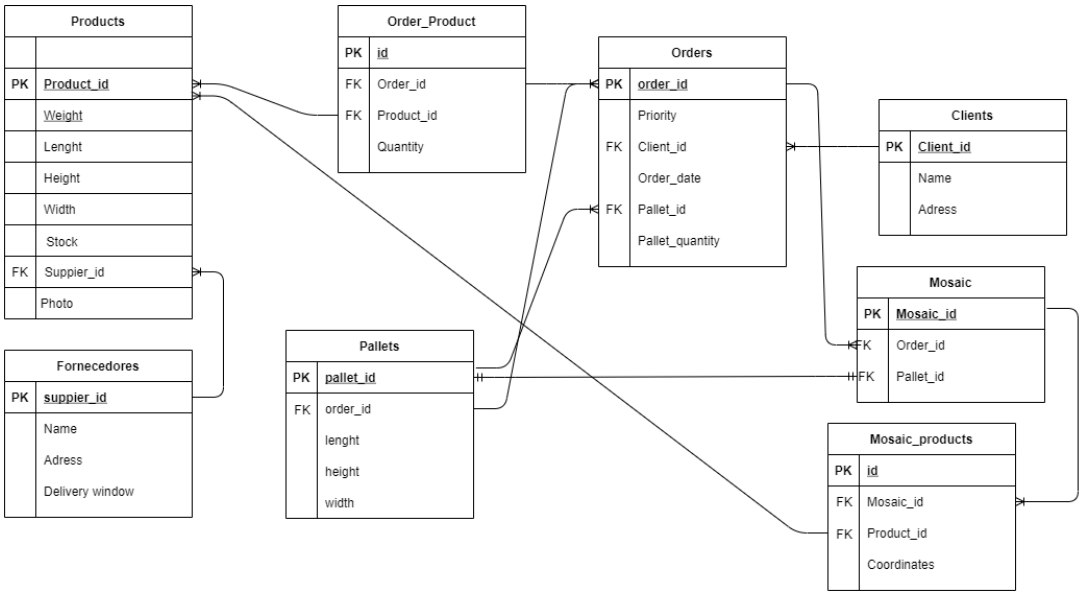

Figure 4.3: Database Relational Model

- Product table, containing information about the characteristics of each one (Weight, Height, Width).
- Supplier table, containing information about each one (Name, Address, Delivery Window).
- Client table, containing information about each one (Name, Address).
- Order Table, containing information from the Planning System, sorted by priority for the respective day. Each order has an *order_id* and a certain quantity of pallets.
- Order_Product table, which establishes the link between the table of Product and Orders. It provides the quantity of each product type present in a given order.
- Pallet table, which contains information about the characteristics of each pallet and to which order it belongs. The pallets used in this *CrossLog* project are estimated to be Europallets (its dimensions are 1200 mm x 800 mm).
- Mosaic table, which indicates to which order and pallet each mosaic corresponds.
- Mosaic_Products, which establishes the link between the Mosiac and Product Table. It gives information about the location (x,y,z coordinates) of each product in the pallet.

4.4.1 Node-Red MongoDB

In order to collect information from the database and make it available in the application, as well as save it for future query and data analysis, a node was used to save and retrieve data in a local MongoDB instance, so it is necessary to have a MongoDB server running. In Figure 4.4, there is

an example of a flow created in the Node Red tool that demonstrates how the alarms and events triggered by the application's users are stored in the database.

Figure 4.4: Example of a Node Red flow containing the MongoDB Node

Initially there is a node that subscribes to the topic *topic/home_page*, and if there is an indication of the existence of an anomaly, this information is collected and transformed into JSON. In the next function node, the time when this occurred is added, as well as the name of the table where the information is to be stored. The code for the function node is attached for consultation in Appendix A. Still regarding this flow, the most important node and the one this subsection is about (MongoDB node), it is necessary to insert the following data:

- Server Host and Port.
- Username and Password (if necessary).
- Operation (find, count, aggregate, insert, update, remove) - in this flow example, the operation used was insert.

4.5 PLC Programming and Integration

Industrial automation in PLC and SCADA systems is an integral part of industry worldwide. Production processes and techniques are increasingly automated and incorporate technology with every cycle. SIEMENS controllers occupy a prominent position in the automation technology park of companies. Given the extent of its use around the world, it was considered important to visualize, in the application, data from a real PLC, and thus simulate the information coming from the factory floor in the scope of this project. The one used is a SIMATIC S7-1500 controller, provided by JPM. Therefore, the following steps were performed:

1. Structured programming and program development on a S7-1500 PLC.
2. Usage of the OPC UA communication protocol.
3. Usage of the UaExpert tool as an OPC UA client.
4. Node-Red communication nodes.

4.5.1 SIEMENS Programming Software

Totally Integrated Automation (TIA) Portal is an engineering tool that allows users to efficiently program and diagnose Siemens automation hardware and software, all within a single software

interface that enables integrated system diagnostics, communication, and other features. As additional versions are announced, newly released hardware and programming functions are added to the wide variety of products and options that the TIA Portal offers. In this case, and under a license acquired by JPM, the project was developed on version 15.1 of the product.

Initially, the PLC used (Siemens S7-1500) was connected to a 24 V DC power supply, and a network cable was used to connect the controller to the computer. The communication protocol used was PROFINET, a protocol designed to exchange data between controllers and devices. To configure the device in the software used, some fields have been taken into account:

- IP adress - also defined in the PLC.
- Subnet mask - separates the IP address into the network and host addresses.
- OPC UA Server adress.
- All other fields did not need to be changed for the device customization.

In order to create a simple program to test the functionality of this tool, a Function Block program was created that, every 2 second (frequency of 0.5 Hz), toggles the value of the *TestInt* variable between the value 0 and 100. Figure 4.5 represents that same program. The variable in question is stored in a database that has been created (*OpcUA_Db*), where the name and type of variable and its initial value must be indicated. Some characteristics can still be changed, such as whether it can be changed through HMI/OPC UA. In this case, those options are active for all variables.

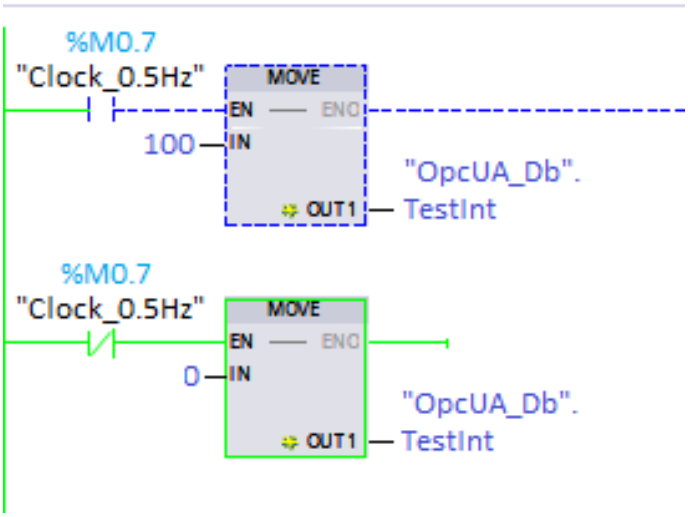

Figure 4.5: Example of a Toggle Function Block Program

4.5.2 OPC UA Client

To establish communication between the backend and the PLC, the UaExpert tool was used, a full featured OPC UA Client, designed as a general purpose test client supporting OPC UA features like data access, alarms and conditions, historical data and calling UA Methods.

- Browse OPC UA Adress Space.
- Reading and writing of variable and UA Attributes.
- Monitoring of data changes.
- Monitoring of events.
- Calling methods.
- Reading and updating of history data.
- Adding and removing nodes and references.

Figure 4.6 shows the window layout of UaExpert when first started:

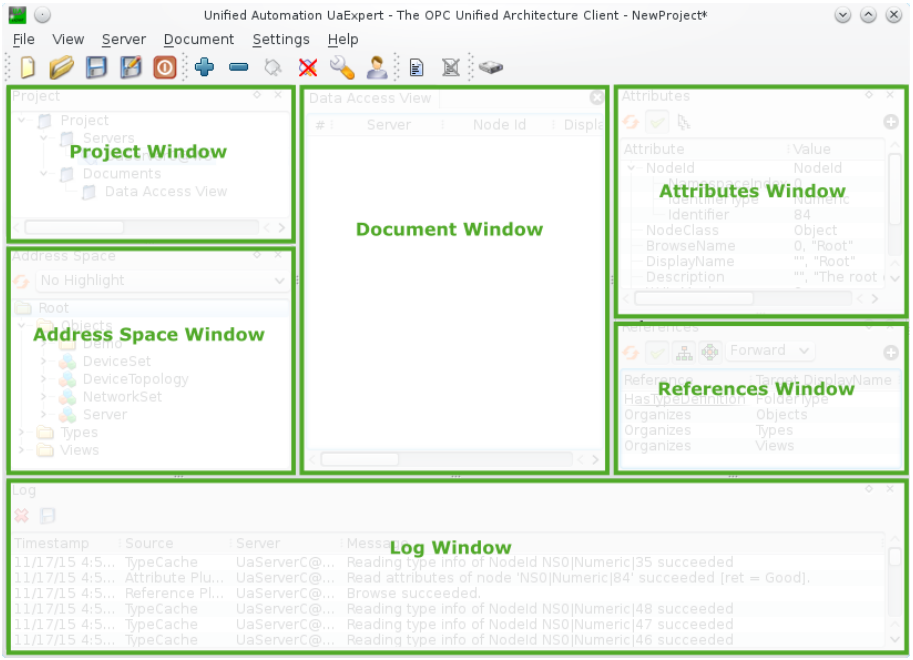

Figure 4.6: UaExpert window layout [2]

- Project Window - Can be used to manage multiple server connections.
- Address Space Window - Shows the adress space of the server which is currently selected in the Project Window.
- Document Window - Can display different view modes. By default, the Data Access View is selected.
- Attributes Window - Shows the attributes of the node which is currently selected in the Address Space Window.

- Reference Window - Shows the reference of the node which is currently selected in the Address Space Window.
- Log Window - Displays status and error messages.

The first step is to add a new connection to the OPC UA Server. The put URL and its port is the same as the one configured in TIA Portal OPC UA settings. After the successful connection, and in order to access the desired variables, in the Address Space Window, the database created previously is selected and those variables are dragged into the Document Window, for a real time visualization. According to the previous example, it is possible to see the variable *TestInt* toggle between the value 0 and 100, with a slight delay comparing to the PLC program.

4.5.3 Node-Red OPC UA Communication

In order to obtain the value of the variables in real time, or approximately, Node-Red nodes were used to communicate via OPC UA based on *node-opcua* library (installed package). Figure 4.7 demonstrates a flow that exemplifies how to collect information from the PLC and send it to the frontend of the application.

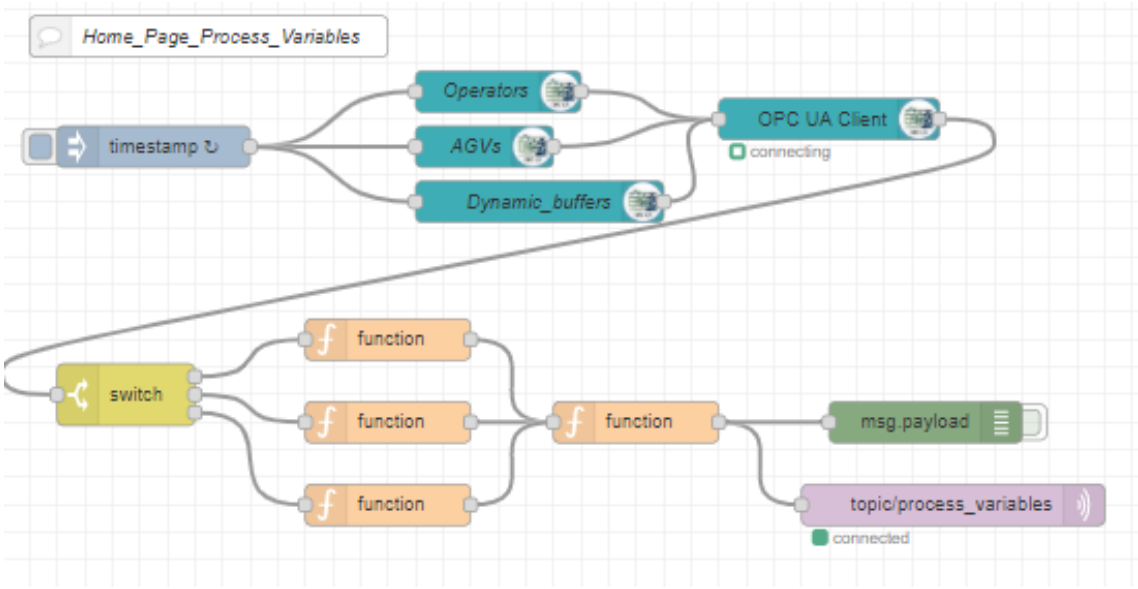

Figure 4.7: Node-Red flow using OPC UA nodes

First of all, a timestamp node is used, and there is a cyclic trigger, set every 2 seconds. The following nodes come from the package mentioned above:

1. OPC UA-Item - is used to define a variable:
   - Item Field - NodeId accessible in Attributes Window in UaExpert.
     - Example: *ns=3;s="OpcUA_Db"."Operators"*
   - Type - Data type.

2. OPC UA Client - is used to actually access the variables values, based on the OPC UA-Item properties:

    - Endpoint - the same one defined in UaExpert, that is the same as the one presented in TIA Portal settings.
    - Action - the action the user wants (read, write, subscribe, unsubscribe, etc) - in this case, the action selected was read.

The next nodes represent, respectively, a switch, which separates the variables present in a single String. The following function nodes put the value of each of them in a global variable and publishes in a given topic, the same that is being subscribed in the frontend of the application and making the values of the variables available on the Home Page. In this case the variables accessed are the number of Operators, AGV and Dynamic Buffers that at that moment exist on the shop floor.

4.6 Implementing MQTT in the Project

As mentioned in previous chapters, real-time data visualization is fundamental and one of the main requirements in the development of this project. In order to implement such a communication protocol in this project, MQTT nodes were used in Node-Red (section 2.8) and an MQTT library was also used in the Vue project. This makes the communication between the backend and the frontend of the application fast and efficient, if implemented correctly.

In the following subsections, an example is provided of how the application, when generating a page, sends to the backend certain parameters present in the URL and, according to them, generates a graphical representation of a palette on the application page.

4.6.1 Node-Red Implementation

The broker used is installed directly from the pallet manager, and has as main features:

- Standard TCP Support.
- WebSocket Support via port or path.
- Secure Sockets Layer (SSL) / Transport Layer Security (TLS).
- Message Persistence (In-memory or MongoDB).

Figure 4.8 represents the Broker node connected to a debug node, in order to keep and visualize messages from it.

This node has some important parameters that will be used when implementing the MQTT protocol in the frontend and also in the publish and subscribe nodes:

- MQTT port.

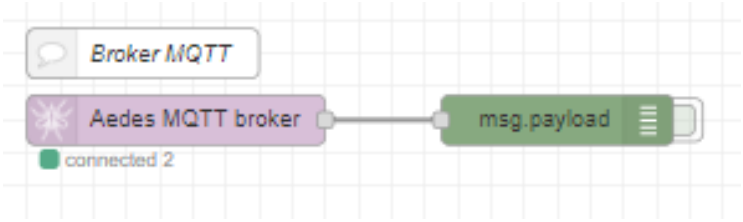

Figure 4.8: Broker Node

- WebSockets bind.
- WebSockets port.
- Security:
  - Username.
  - Password.

Figure 4.9 shows a flow that has as trigger the income of a message, performed by the subscription node.

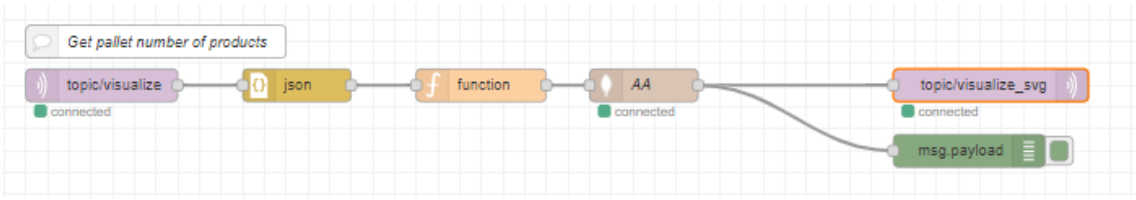

Figure 4.9: Example Node-Red flow using MQTT Subscriber and Publisher

This message is published in the topic *topic/visualize*, and contains information about the value of certain parameters of the application URL. This message is transformed into JSON and sent to the database, which indicates the number of products in the selected pallet, according to the *order_id* and *pallet_id* (these are the URL parameters). More information about each database table and its attributes can be found in section 4.

This same value from the database is then published in the topic *topic/visualize_svg*, to then be analyzed by the frontend and allow the visualization of a pallet with products resting on it.

Each subscriber/publisher node must be connected to the previously described broker. For this, in the server settings of the nodes, it is necessary to indicate the server IP. For example, if the broker is running locally, this field would be filled in as "ws://localhost: *WebSockets port*". If security exists, as is the case where there is a username and a password, it is necessary to fill in these fields in the security section of the node settings.

4.6.2 Vue Implementation

Vue is a progressive framework for building user interfaces. Unlike other monolithic frameworks, Vue is designed from the ground up to be incrementally adaptable. The core library is focused on

the view layer only, and is easy to pick up and integrate with other libraries or existing projects. On the other hand, Vue is also perfectly capable of powering sophisticated Single-Page Applications when used in combination with modern tooling and supporting libraries.

That said, the MQTT library was installed and implemented, which allows connecting to a broker, simulating the subscription, receiving and sending messages, unsubscribing and the disconnecting between the client and the broker.

The *connection* variable, used in the *createConnection()* function is composed of different attributes, such as the host, port, endpoint, connectTimeout, username and password. These fields are filled matching those previously defined above (subsection 4.6.1).

In order to satisfy the above requirements about reading header parameters when the page loads, some functions are called in the main function *beforeMount()* (Listing 4.3), that means that all functions inside this one are called right before the mounting begins: the render function is about to be called for the first time.

```
1     beforeMount() {
2     this.createConnection();
3     this.doSubscribe();
4     var site = new URL(window.location.href);
5     this.param_1 = site.searchParams.get("order_id");
6     this.param_2 = site.searchParams.get("pallet_id");
7     this.doPublish();
8     }
```

Listing 4.3: Functions called when page loads

The developed code for the functions to create the connection to the broker, subscription and publishing are present in Appendix A. The *doPublish()* function sends information to the topic *topic/visualize*. Note that when a message is received, it is advisable to check the topic and thus analyze the received data more easily, like it is possible to see in Listing 4.4 (also about the number of products on each pallet):

```
1     if(topic == "topic/visualize_svg"){
2     var obj = JSON.parse(message);
3     this.nr_products = obj[0].nr_boxes;
4     var products = this.nr_products;
5     var i = 0;
6         while(products > i){
7         // Pallet constructor code //
8         }
9     }
```

Listing 4.4: Topic analysis code

4.7 Web Application Pages Development

This section describes the implementation process and how the information is organized and where it comes from. It is also described what had to be implemented to overcome some adversities that arose throughout the development of the project. The web application pages not covered follow an implementation strategy similar to the one described. All pages have in common the menu with navigation options and a footer with information about the author of the work developed.

4.7.1 Home Page

The home page of an application translates the first impression that the user has. It must be clean and clear, and the information must be well structured and organized. Figure 4.10 illustrates the final result of the developed home page.

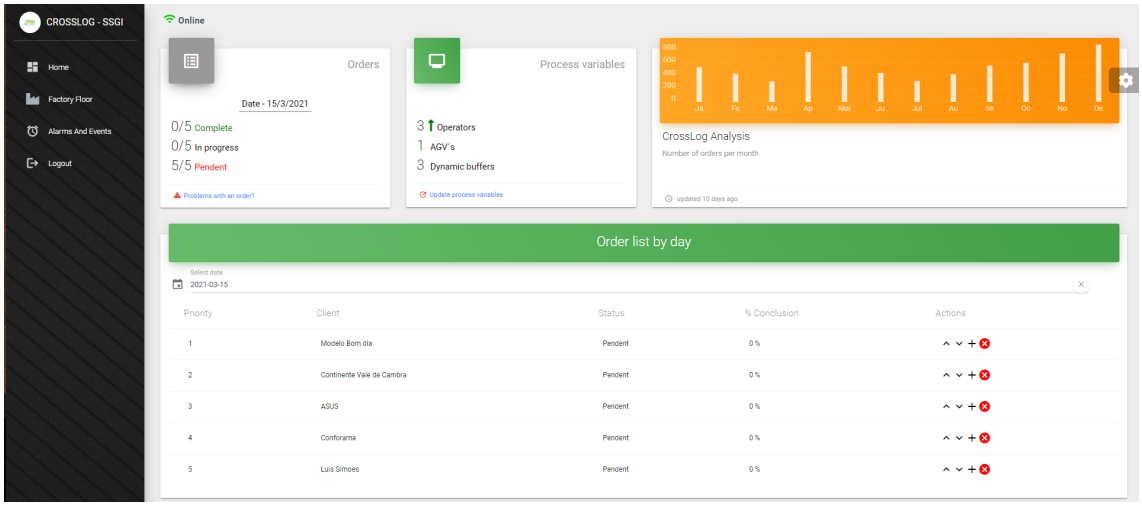

Figure 4.10: Application Home Page

Initially, and if the system is subscribing to the correct topics, the user has the possibility to change the day, and thus consult the quantity and status of orders and details about each one individually. By default, the date chosen when opening the application is the current day.

After choosing a certain date, the Vue application sends a message to Node-Red with the date that the user selected (date format: *dd/mm/yyyy*), and all the information about this day that is present in the database is returned via MQTT:

- Order list by day:
  - Priority.
  - Client.
  - Status
  - Percentage of conclusion
  - Actions:

    - Raise the priority.
    - Lower the priority.
    - More details about the order.
    - Cancel order.

- Day Status:
    - Complete orders.
    - In progress orders.
    - Pendent orders.

In addition to information about orders, the home page provides data from the shop floor (simulation with the PLC) in real time, such as the number of Operators, AGV and Dynamic Buffers. As a corrective measure (subsection 3.2), it was defined that if the number of Dynamic Buffers is higher than the sum of Operators and AGV, a warning will appear on the screen to increase the number of Operators. As soon as the system receives information that this number has been raised, the warning disappears.

So that the application user (administrator or operator) can report errors or problems, there are buttons that, when pressed, emit an alarm or event, depending on the location and the problem, which is entered into the database and later analyzed in the respective Alarms and Events page.

4.7.2 Pallet 3D Visualization

Initially, it was defined the integration in this project of a tool already developed for the visualization of each pallet, after intelligent calculation of the position of each product, through its dimensions, order of arrival and priority of orders. However, due to the nonexistence of a Planning System and due to privacy and copyright policies regarding the software that was to be implemented, an achievable solution had to be thought within the remaining implementation time and that would satisfy the imposed requirements.

It was then defined, for testing purposes, that all pallets were Europallets, with fixed dimensions of 1200 mm x 800 mm and that each order would have a certain number of pallets, each one consisting only of products of equal dimensions and characteristics (600 mm x 267 mm). Therefore, when the user is provided with the details page of each order, he has the possibility to click and view each pallet individually. This process redirects the user to a preview page, as can be seen in Figure 4.11. The left page is for a pallet with 6 products and the right page is for a pallet with 16. Both pallets belong to the same order.

The web page sends to the Node-Red tool the *order_id* and *pallet_id* that the user wants to view. After receiving the number of products that will be on that pallet (by collecting information from the database), the application generates a dynamic image that visually demonstrates how the pallet should look after it is filled.

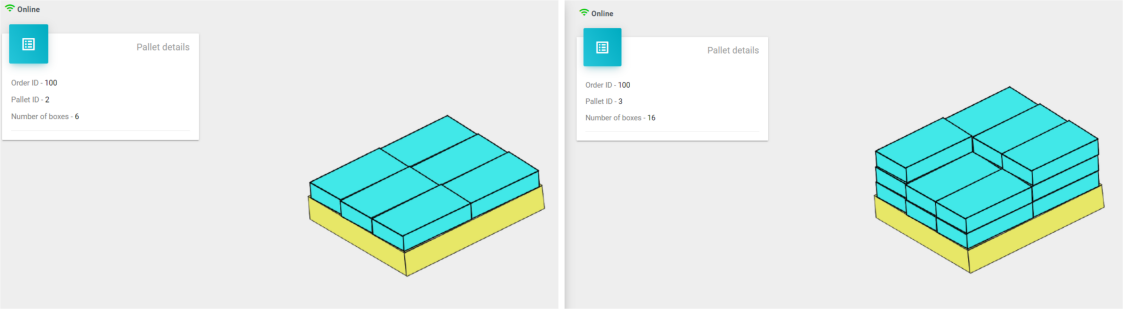

Figure 4.11: Dynamic pallet visualization

4.7.3 Factory Floor

One of the main requirements of this project was to be able to visually represent the factory floor, in (approximately) real time, so that any kind of user would be able to draw conclusions about what is happening at that moment in the factory. Therefore, the overall system, as well as each module inherent to it, is accompanied by information provided next to the graphical representation:

1. Global Factory Floor:
    - Status of each module - represented in full and by the use of industrial traffic lights.
    - Legend of the icons used.
    - Problem report.
2. Identification and Weighing Station:
    - Status of the Control Station and Warehouse - represented by the use of colors.
    - Mode switch from Automatic to Manual.
    - Conveyors speed.
    - Problem report.
3. Dynamic Accumulator:
    - Status of each accumulator - represented by the use of colors.
    - Mode switch from Automatic to Manual.
    - Conveyors speed.
    - Problem report.
4. Palletization:
    - Position, status (when working, indication of the order number) and battery level of each AGV.
    - Robotic arm position of one AGV.

- Mode switch from Automatic to Manual.
- Problem report.

5. Progressive Involvement:

- Status of each wrapper - represented in full and by the use of colors.
- Mode switch from Automatic to Manual.
- Problem report.

Figure 4.12 represents the application page regarding the shop floor state of the global process. As can be seen, each of the modules is accompanied by a hyperlink (represented by an icon) that redirects the user to the page for each of the subsystems; this can also be done using the bottom menu containing the full name of each of the modules.

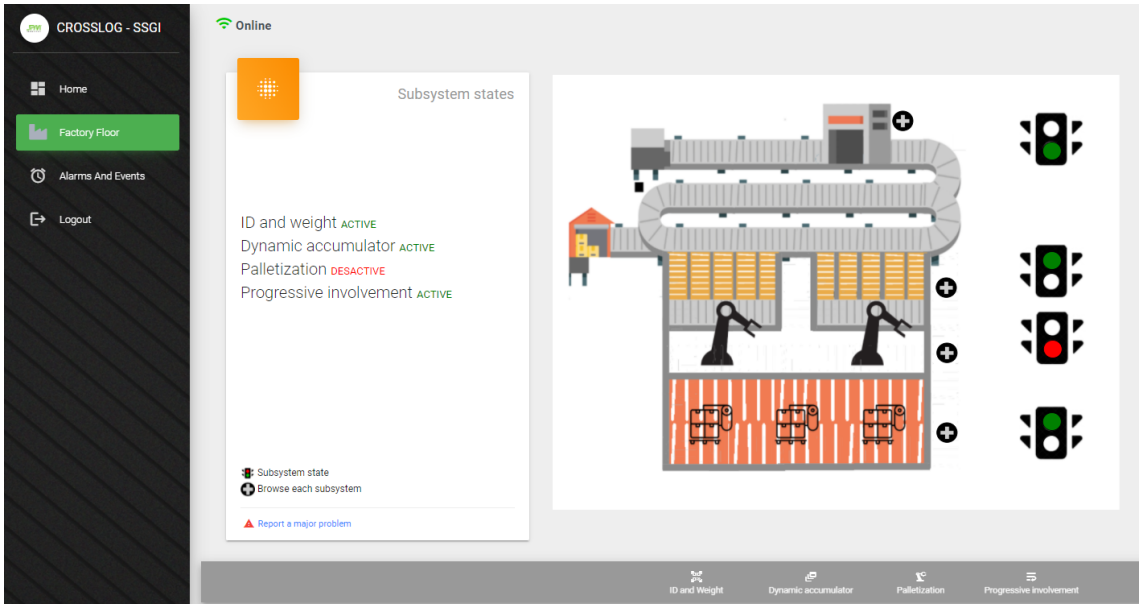

Figure 4.12: Full System Factory Floor Status

4.7.4 Palletization AGV Controller Software

In order for the administrator to be able to simulate and visually represent characteristics of an AGV, a software was created, capable of sending commands to the application and thus customize the position, the status, the battery level, and also the position of the robotic arm, if it exists.

Figure 4.13 represents the AGV Control Software, also implemented in VueJs, which sends messages to topics read in the main application. Note that, for this communication to be possible, the connection to the same broker is fundamental, and the only parameter that changes is the user, so that no two users with the same name are connected to the same broker (resulting in conflict), assuming that the communication protocol used is the same as the one mentioned before, MQTT.

Figure 4.13: AGV Control Software

Motion buttons have been developed for each AGV, allowing it to move around in the Cartesian coordinate system. Each time a position change command is sent, corresponding changes are made in the frontend, namely in the SVG parameters. The sensitivity of each movement iteration was set as 10 units. This value can be reduced if it is desired to move the robot more precisely.

The sliders change the battery level and, in the case of the AGV with the gripper, there is one that controls the position of the robotic arm. Figure 4.14 represents the page of the Palletization module using the Control Software.

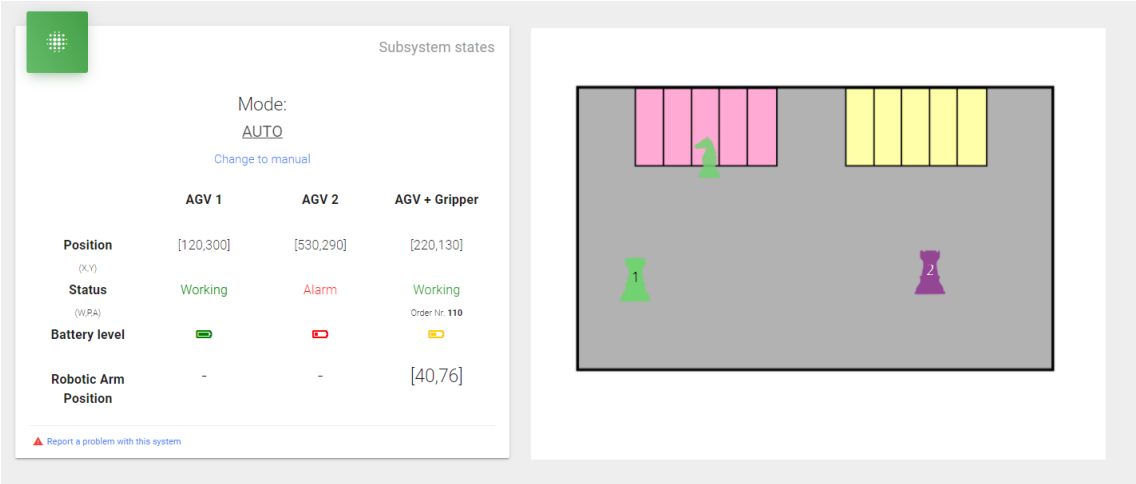

Figure 4.14: Palletization Application Page

4.7.5 Alarms and Events

Any type of user, when using the application, can report problems and errors, as mentioned above. Therefore, each of these warnings is sent to the database, along with the date, location where it occurred, type of problem, and a short message about the details of the anomaly.

All these alarms and events are described in the page of the Alarms/Events application, where the information is organized and separated, as depicted in the Figure 4.15.

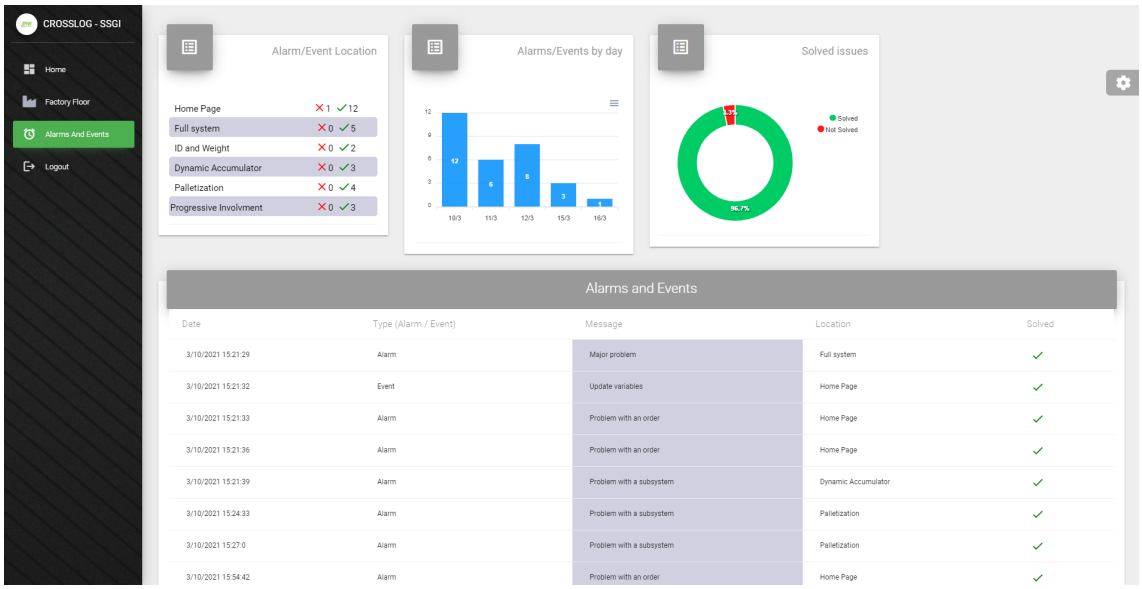

Figure 4.15: Alarms/Events Application Page

The alarms and events page of an industrial application is quite important, as is the layout of the information. It is necessary for the user to act as quickly as possible when an anomaly appears in the system, to be able to resolve it and avoid delays or increased costs. To this end, events and alarms have been separated by subsystem, allowing the user to know which errors have not yet been resolved or have already been resolved. It is also possible to see the information organized by days and by its total resolution success.

4.8 Network implementation of the results

In order to make the application executable in a real environment, it was implemented in the company's network. Initially, the application was developed locally, but in order for it to be used by people in the company, it had to be exported to a new machine within the company's network. A virtual machine has been made available within the company's private network, which runs Windows 10, has 8 GB RAM and an Intel Core i7 processor.

In terms of security, besides the existing one, there was no added attention (for example, the use of the HTTPS protocol instead of HTTP), due to the company's network being private and

secure. In addition to installing the various programs that need to run in parallel to the application server (Keycloak, MongoDB, Node-Red), it was necessary to reconfigure the IP of the new machine to the required parameters.

4.9 Tests and Validation

After implementing and carrying out the activities described above, it can be said that the results meet the requirements initially defined. Research is needed about the possibility of adapting the application to different types of customers, with different cross-docking structures and needs. For now, and given the requirements imposed by the company and those involved in the dissertation, data transmission is reliable and the organization of information is well structured and perceptible. Although the results obtained were simulated and could not have been tested in a real situation with all the existing modules present, evaluation metrics were defined in order to draw conclusions about the viability and robustness of the application. A few tests were performed simultaneously to prove the robustness of the application:

- Functionality:
  - Adequacy - Capability of the software product to provide an appropriate set of functions for specified tasks and user objectives: **OK**.
  - Accuracy - Capability of the software product to provide, to the degree of accuracy required, correct or agreed-upon results or effects - **OK**.
  - Interoperability - Capability of the software product to interact with one or more specified systems - *NOT OK* - The case of communication with the Plaiting System and other modules still under development was not feasible. In this case the communication with the PLC was given as correctly implemented
  - Security: The software product's ability to protect information and data so that unauthorized persons or systems cannot read or modify it, and so that authorized persons or systems are not denied access: **OK** - The application ensures security because it is on the company's protected network, but changes would be needed if it were not (security certificates would have to be implemented).
- Efficiency:
  - Time behavior - Evaluates whether the response (or processing) times are within specifications: **OK**.
  - Resource utilization - Measures both the resources consumed and the system's ability to utilize the available resources: **OK** - Table 4.2.

Table 4.2 shows some metric evaluation parameters and their respective results. Note that the application developed is lightweight, and much is due to the type of communication between modules (MQTT protocol), which presents very favorable characteristics.

Table 4.2: Metric Evaluation

Metric	**Result**
% of Memory occupied	36
% of CPU used	15-20

From a more practical point of view, the following requirements imposed at the beginning of the work have been validated:

- Communication with PLC and availability of the variables in real time.
- Inserting and updating information in the database.
- Authentication and authorization by the different users created.
- Possibility to report errors.
- Sending motion commands, getting battery levels and status of the robotic units.

All the points reviewed above were analyzed and verified by the parties involved in the development of the project and some of JPM's employees.

Chapter 5

Conclusions and Future Work

The development of this project allowed the deepening of knowledge regarding different concepts related to the creation of an application for industrial use, such as activity planning, security, communication between different modules and PLC, Backend and Frontend programming. Note that the application was created during the implementation time of the *CrossLog* project, so the results and some modules were simulated to check the reliability and consistency of the application. The inclusion of Node-RED as a tool also allowed for an introduction to other areas of IoT such as data analytics and communications management. The use of the MQTT protocol allowed the study of its advantages, disadvantages, and the transport of data between modules and applications in parallel. Through the results obtained, it can be verified that the transmitted data are not lost and are correctly stored and made available to users.

It was possible to create a functional application that monitors and controls in real time processes related to the Cross-Docking concept. Requirements regarding communication between modules, security and information available to different types of users were met.

In the future, and if more time and resources existed for the implementation phase, the following future activity points can be considered:

- Integration of 3D visualization tools and optimization of the position of the different objects present in each of the pallets.
- Integration of an interface for real-time visualization of the robots' position and status via Robot Operating System (ROS).
- Switching from HTTP to HTTPS in order to give even more security to the application and to the information provided.

Appendix A

Code Appendix

A.1 Keycloak Vuejs

```
1
2  import Vue from "vue";
3  import VueRouter from "vue-router";
4
5  //Keycloak library
6  import VueKeycloakJs from "@dsb-norge/vue-keycloak-js";
7
8
9  // configure router
10 Vue.use(VueKeycloakJs, {
11   init: {
12     // Use 'login-required' to always require authentication
13     onLoad: 'login-required',
14     silentCheckSsoRedirectUri: window.location.origin + "/silent-check-sso.html"
15   },
16   config: {
17     url: 'http://192.168.40.231:8080/auth',
18     clientId: 'vue-test-app',
19     realm: 'keycloak-demo'
20   }
21 })
22 Vue.use(VueMaterial)
23 Vue.use(VueRouter);
24 const router = new VueRouter({
25   mode: 'history',
26   base: process.env.BASE_URL,
27   routes:
28
29     // ROUTES
30
31     }
```

```
  ],

});
function sleep(ms) {
  return new Promise(resolve => setTimeout(resolve, ms))
}

router.beforeEach(async (to, from, next) => {
  if (to.matched.some(record => record.meta.requiresAuth)) {
    // We wait for Keycloak init, then we can call all methods safely
    while (!router.app.$keycloak.ready) {
      await sleep(100)
    }

    if (router.app.$keycloak.authenticated) {
      next()
    } else {
      const loginUrl = router.app.$keycloak.createLoginUrl()
      window.location.replace(loginUrl)
    }
  } else {
    next()
  }
})

new Vue({
  router,
  data: {
    Chartist: Chartist
  },
  render: h => h(App)
}).$mount("#app");

let payload = {
  idToken: keycloak.idToken,
  accessToken: keycloak.token
}
if (keycloak.token && keycloak.idToken && keycloak.token != '' && keycloak.idToken
    != '') {
  store.commit("login", payload);
  console.log("User has logged in: " + keycloak.subject)
}
else {
  store.commit("logout");
}
```

A.2 Node-Red function node - insert a new alarm/event

```
var type = msg.payload
var id_alarme  = Math.floor(Math.random() * 10000);

var d = new Date()
    dformat = [d.getMonth()+1,
                d.getDate(),
                d.getFullYear()].join('/')+' '+
               [d.getHours(),
                d.getMinutes(),
                d.getSeconds()].join(':');

var newMsg = {};
newMsg.collection = 'alarms_events';
            newMsg.payload = {
                "type":msg.payload.type,
                "msg":msg.payload.msg,
                "location": msg.payload.location,
                "data": dformat,
                "solved": msg.payload.solved,
                "id_alarme": id_alarme


                  }
return newMsg;
```

A.3 MQTT Vuejs functions

```
// Create connection to the MQTT Broker
    createConnection() {
      const { host, port, endpoint, ...options } = this.connection;
      const connectUrl = `ws://${host}:\${port}${endpoint}`;
      try {
        this.client = mqtt.connect(connectUrl, options);
      } catch (error) {
        console.log("mqtt.connect error", error);
      }
      this.client.on("connect", () => {
        console.log("Connection succeeded!");
        connection_verify = 1;
      });
      this.client.on("error", error => {
        console.log("Connection failed", error);
```

```
16          connection_verify = 0;
17        });
18        this.client.on("message", (topic, message) => {
19          this.receiveNews = this.receiveNews.concat(message);
20        });
21      }
22
23
24      // Subscribe to a certain topic - String defined in variables
25          doSubscribe() {
26        const { topic, qos } = this.subscription;
27        this.client.subscribe(topic, { qos }, (error, res) => {
28          if (error) {
29            console.log("Subscribe to topics error", error);
30            return;
31          }
32          this.subscribeSuccess = true;
33          console.log("Subscribe to topics res", res);
34        });
35      }
36      // Publish to a certain topic - String defined in variables
37
38          doPublish\_order() {
39        const { topic, qos, payload } = this.publish_order;
40        this.client.publish(topic, payload, qos, error => {
41          if (error) {
42            console.log("Publish error", error);
43          }
44          alert("An alarm / event was set!");
45        });
46      }
47
48      //Destroy the connection
49
50          destroyConnection() {
51        if (this.client.connected) {
52          try {
53            this.client.end();
54            this.client = {
55              connected: false
56            };
57            console.log("Successfully disconnected!");
58          } catch (error) {
59            console.log("Disconnect failed", error.toString());
60          }
61        }
62      }
```

Bibliography

[1] Inductive Automation. What is scada? Available at https://www.inductiveautomation.com/resources/article/what-is-scada (Last access: 03/02/2021), 2018.

[2] Unified Automation. Uaexpert - first steps. Available at http://documentation.unified-automation.com/uaexpert/1.4.2/html/first_steps.html (Last access: 19/04/2021).

[3] A. Azizi. *Modern manufacturing. In SpringerBriefs in Applied Sciences and Technology 1st Edition Pages 7-17*. Springer, Singapore, 2019.

[4] Francois Botha. Testing keycloak with a simple vue.js client. Available at https://francoisbotha.io/2019/11/03/testing-keycloak-with-a-simple-vue-js-client/ (Last access: 07/04/2021).

[5] Rajkumar Buyya and Amir Vahid Dastjerdi. *Internet of Things: Principles and Paradigms 1st edition Pages 23–29*. Morgan Kaufmann, May 25, 2016.

[6] Dipali Chaudhari. Top 25 advantages and disadvantages of plc | pros and cons. Available at https://dipslab.com/plc-advantages-disadvantages/(Last access: 03/02/2021).

[7] Direktoratet. vue-keycloak-js plugin. Available at https://github.com/dsb-norge/vue-keycloak-js (Last access: 07/04/2021).

[8] OpenJS Foundation. The core nodes. Available at https://nodered.org/docs/user-guide/nodes (Last access: 22/03/2021).

[9] Karan Gandhi. Authentication & authorization in web apps. Available at https://blog.jscrambler.com/authentication-authorization-in-web-apps/ (Last access: 22/03/2021).

[10] Shreya Ghosh and Aruna Srinivasan. *Manufacturing Analytics and industrial internet of things. IEEE Intelligent Systems Pages 74–79*. 2017.

[11] Paul Hinz. What is cross-docking - understanding the concept and definition. Available at https://www.adaptalift.com.au/blog/2011-12-23-what-is-cross-docking-definition (Last access: 06/02/2021), 2011.

[12] Maria Antonietta Perna Ivaylo Gerchev. What is svg? your guide to svg files. Available at https://www.sitepoint.com/svg-101-what-is-svg/ (Last access: 24/03/2021).

[13] JPM. *Relatório Técnico E2 31/05/2020 – Relatório de Requisitos técnicos do Sistema.* (documento reservado), 2020.

[14] John Klaess. 6 manufacturing dashboards for visualizing production. Available at https://tulip.co/blog/manufacturing-apps/manufacturing-dashboards/ (Last access: 22/03/2021).

[15] Deviprasad Kotian. Securing asp.net web api using custom token based authentication. Available at https://www.ecanarys.com/Blogs/ArticleID/308/Token-Based-Authentication-for-Web-APIs (Last access: 22/03/2021).

[16] Shancang Li, Li Da Xu, and Shanshan Zhao. *The internet of things: a survey. Information Systems Frontiers Pages 243–259.* 2015.

[17] Chen Choplou Li Z. *A Solution for Cross-docking Operations Planning, Scheduling and Coordination. Journal of Service Science and Management 05(02):111-117.* 2012.

[18] Yang Lu. *Industry 4.0: A survey on technologies, applications and open research issues. Journal of Industrial Information Integration Volume 6, June 2017, Pages 1-10.* 2017.

[19] Rachaelle Lynn. What is just-in-time manufacturing? Available at https://www.planview.com/resources/guide/what-is-lean-manufacturing/just-in-time-manufacturing/ (Last access: 10/02/2021).

[20] Critical Manufacturing. Manufacturing execution systems. Available at https://www.criticalmanufacturing.com/en/critical-manufacturing-mes/what-is-manufacturing-execution-system (Last access: 03/02/2021), 2018.

[21] Node-Red. Securing node-red. Available at https://nodered.org/docs/user-guide/runtime/securing-node-red (Last access: 07/04/2021).

[22] SAICA NPO. The stages of industrial revolution and its impact on jobs. Available at https://www.accountancysa.org.za/the-stages-of-industrial-revolution-and-its-impact-on-jobs/(Last access: 05/02/2021), 2008.

[23] Okta. What is token-based authentication? Available at https://www.okta.com/identity-101/what-is-token-based-authentication/ (Last access: 22/03/2021).

[24] U. A. Pozdnyakova, V. V. Golikov, I. A. Peters, and I. A. Morozova. *Genesis of the revolutionary transition to industry 4.0 in the 21st century and overview of previous industrial revolutions. In Studies in Systems, Decision and Control.* German Research Center for Artificial Intelligence (DFKI),, 2011.

[25] SAGE. Como melhor gerir o inventário e fazer crescer o seu negócio. Available at https://www.sage.com/pt-pt/gestao-de-inventarios-e-stock (Last access: 9/02/2021).

[26] Padraig Scully. The top 10 iot segments in 2018 – based on 1,600 real iot projects. Available at https://iot-analytics.com/top-10-iot-segments-2018-real-iot-projects/(Last access: 03/02/2021).

[27] SWOOP. Web application authentication: 5 best practices to know. Available at https://swoopnow.com/web-application-authentication-best-practices/ (Last access: 22/03/2021).

[28] INESC TEC. Concretização de sistema físico e software para paletização mista automática em centros logísticos cross-docking para cadeias de valor responsive demand-driven. Available at https://www.inesctec.pt/pt/projetos/crosslog (Last access: 01/02/2021).

[29] Diogo Luís Rey Torres. *Increasing the Feedback on IoT Development in Node-RED*. PhD thesis, FEUP, 2020.

[30] Upkeep. What is the difference between plc and scada? Available at https://www.onupkeep.com/answers/asset-management/plc-vs-scada (Last access: 04/02/2021), 2019.

[31] Cattrysse D Van Belle J, Valckenaers P. *Cross-docking: State of the art. Omega 40(6):827-846*. 2012.

[32] VENDUS. O que é a industria 4.0 e qual o impacto nas empresas. Available at https://www.vendus.pt/blog/industria-4-0/(Last access: 04/02/2021).

[33] Marcio Venturelli. Rami 4.0: Modelo de referência para arquitetura da indústria 4.0. Available at https://www.automacaoindustrial.info/rami-4-0-modelo-de-referencia-para-arquitetura-da-industria-4-0/ (Last access: 24/03/2021).