Universidade de Évora - Escola de Ciências e Tecnologia

Mestrado em Engenharia Informática

Dissertação

Implementation of an information system for resource and process management in an industrial facility

João Pedro Amaral Dias

Orientador(es) | José Saias

Évora 2021

Universidade de Évora - Escola de Ciências e Tecnologia

Mestrado em Engenharia Informática

Dissertação

Implementation of an information system for resource and process management in an industrial facility

João Pedro Amaral Dias

Orientador(es) | José Saias

Évora 2021

Évora 2021

To my parents

Acknowledgments

First, I would like to express my gratitude to Professor José Saias, my dissertation advisor, for his consistent availability, patience and support throughout the full course of this project. Without his guidance and encouragement, this dissertation would not be possible.

I would also like to thank the partner entity of this project for their participation and collaboration, even taking into account the challenging moments we are experiencing.

In addition, I thank my family for their unconditional support and, especially, my parents, for giving me the necessary strength not only throughout this dissertation, but also during my entire academic journey.

Finally, a special thanks to all my friends and colleagues, who in one way or another, helped me to be the person I am today, and to my amazing girlfriend who always encouraged me to never give up and to fight for my dreams.

Contents

List of Figures

List of Tables

Acronyms

ERP	Enterprise Resource Planning
HTTP	HyperText Transfer Protocol
IoT	Internet of Things
ISA	International Association of Automation
JSP	JavaServer Page
MES	Manufacturing Execution System
MESA	Manufacturing Enterprise Solutions Association
MRP	Manufacturing Resource Planning
MVC	Model View Controller
POJO	Plain Old Java Object
SME	Small and Medium-sized Enterprises
SOA	Service Oriented Architecture

Abstract

The increasing investment in information systems for industrial production control, combined with the advent of the fourth industrial revolution, allows to improve decision making, providing a more efficient manufacturing process. However, for small and medium-sized enterprises in this sector, the complexity of existing solutions and the need to perform, in a preliminary phase, a substantial investment makes it difficult to implement these types of systems, accentuating, in terms of competitiveness, the gap between small and large companies.

The proposed system, developed within the scope of this dissertation, aims to assist this types of companies, more specifically, a local business entity in the branch of the aeronautical industry, since it allows to optimize the monitoring of industrial resources and processes at the shop floor level.

To support the design and implementation of the solution, an analysis on the operation of industrial facilities and techniques adopted in the field of industrial information systems is carried out. The entire development phase encompasses software engineering processes and culminates with the delivery of a prototype of a Spring web application, deployed on the partner entity's server and positively evaluated in the execution of development and functional tests.

Keywords: Industrial, Production, Resources, Information System, Web Application

Sumário

Implementação de sistema de informação para gestão de recursos e processos industriais

O investimento crescente em sistemas de informação para controlo de produção industrial, aliado ao advento de uma quarta revolução industrial, permite aprimorar a tomada de decisões, proporcionando um processo de manufatura mais eficiente. No entanto, para pequenas e médias empresas deste sector, a complexidade das soluções existentes e a necessidade de efetuar um investimento preliminar substancial dificulta a implementação deste tipo de sistemas, acentuando, a discrepância competitiva entre pequenas e grandes empresas.

O sistema proposto, desenvolvido no âmbito do trabalho da presente dissertação, visa auxiliar esta tipologia de empresas, mais concretamente, uma entidade empresarial local no ramo da indústria aeronáutica, na medida em que permite otimizar a monitorização de recursos e processos industriais ao nível do chão de fábrica.

Para suporte da conceção e implementação da solução realiza-se uma análise do funcionamento de instalações industriais e de técnicas adotadas no domínio de sistemas de informação para aplicação industrial. A totalidade da fase de desenvolvimento engloba processos de engenharia de software e culmina com a entrega de um protótipo de uma aplicação web Spring, implantada no servidor da entidade parceira e, positivamente avaliada na execução de testes de desenvolvimento e funcionais.

Palavras chave: Industrial, Produção, Recursos, Sistema de Informação, Aplicação Web

1 Introduction

In this chapter an introduction to this dissertation theme is presented alongside some important topics that will be approached during this work. This introduction also includes the motivation for this project, the objectives proposed to achieve and also the structure that the dissertation will follow.

Throughout history, we have witnessed three major industrial revolutions that have completely changed the world economically and socially. According to Schwab, K. in [Sch16], we are now entering a fourth industrial revolution caused mainly by the increase in computer processing power that is allowing to take advantage of new fields such as big data, artificial intelligence or internet of things in industrial environments. This revolution can therefore affect all industry around the world, generating many opportunities, since there will be a need to make complete transformations in entire production systems to keep companies efficient and competitive in the market.

This dissertation will focus on the theme of information technologies applied to the industrial component, always with the objective of exploring new technologies and new concepts that aim to make the entire industrial production process more valuable and transparent. Through a partnership with an aeronautical industrial company, this dissertation also aims to implement a system that tightens the connection of this company with themes of industry 4.0.

1.1 Motivation

An initial partnership with a local industrial aeronautical facility served as the starting point for this dissertation. With this partnership, it was possible to identify the growing need for manufacturing companies to have a system for resource and process management on the shop floor. Currently, the absence of such system can lead to economic losses to the enterprise, since it becomes more difficult to manage the production schedule and the availability of resources in the facility. Therefore, the main motivation for this dissertation is to help the management of processes and resources of this company through the implementation of a software solution to support facility operations, with analytical features and interoperability with existing systems.

It is also in the interest of this project to battle the lack of resource and process management systems for manufacturing small and medium-sized enterprises (SMEs), since most of the solutions available on the market are very complex and difficult to implement in smaller industrial environments, making it more challenging for these enterprises to keep up with topics from the fourth industrial revolution.

The other source of motivation for this dissertation was the opportunity to apply all the knowledge acquired during the master's degree in computer science to develop a complete system in the form of a web application. This implementation also provides the opportunity to carry out other stages of a regular software development lifecycle, such as software design, deployment and testing, on a real-life project scenario, culminating in the end result of a fully functional application to be used in a real industrial environment.

1.2 Objectives

As previously stated, the goal of this dissertation is the study, design and implementation of a software solution prototype for resource and process management in a local industrial aeronautical facility. The system should allow registered facility operators to perform key manufacturing management activities, at the shop floor level, such as creating purchase orders, dispatching jobs to production, tracking availability of facility resources and more. It is also requested that this system can be linked to corporate-level applications to allow information flow between different enterprise levels.

To achieve this main objective, it is also necessary to conduct a research on information technologies applied to manufacturing to realize the operation mode of a standard industrial facility and, thus, understand the best way to address the requirements requested by the partner facility. It is also necessary to conduct a research on the available technologies, that can eventually be used to implement the desired solution, in order to determine the best ones to choose and to apply in this specific use case.

Thus, the list of objectives for this dissertation can be described as bellow:

- Research the working method of standard manufacturing facilities;
- Research on information technologies applied to the manufacturing sector or useful in implementing the components of the proposed solution;
- Describe and design the proposed solution in the form of a web application;
- Implement a software prototype that meets the requirements described;
- Test and evaluate the developed prototype, in different scopes.

1.3 Structure

The structure of this dissertation consists of five chapters that cover different sections of the work performed around the implementation of the proposed software solution, a web application, for resource and process management, in an industrial facility. Included in these five chapters is the introductory chapter, where the main theme of the work is presented together with its goals, motivation and structure.

Chapter 2 outlines the state of the art and some key concepts in the field of computer science when applied to industrial environments. This chapter is the result of an extensive research and analysis carried out with the aim of collecting relevant information to support the next stage of the project, which consists of the practical implementation of the proposed system. Also examined in this chapter are two systems similar to the one being implemented, with the objective of evaluating the offered features and detecting gaps in this type of solutions.

All previously investigated concepts are then applied in Chapter 3, where the system's design and implementation are described. The definition of goals and requirements for the system serves as a starting point for this chapter, which continues with an explanation of why the used technologies were considered the best for this specific use case. Then, the system architecture is illustrated with all decisions made on security and database issues being also explained. Lastly, in this chapter, three of the system modules are detailed with the support of some code snippets that demonstrate how the solution was technically implemented.

The results of multiple experimental evaluations are reported in Chapter 4, to support some analysis and conclusions about the system's behavior and performance. Three types of experiments are included in this section, in order to evaluate the system technically, through unit and performance tests, and functionally, through direct interaction with potential end users of the system. In this last experiment, an initial version of the system is provided to the users to collect their feedback and to discover possible problems with the system.

The last chapter concludes this dissertation by presenting a summary of all the work carried out through the investigation, the implementation and the testing phases. The obtained results are also discussed to assess whether all the initially proposed objectives have been achieved. Finally, an introduction to the future work of this dissertation is made, by detailing the necessary steps to be followed to complement the solution initially implemented.

2 State of the Art

This chapter begins with an overview of an industrial production system. This step is important to better understand the importance of an information system, for resource and process management, as a way to support and improve the manufacturing process in industrial environments. General functionalities and requirements for this type of system are then presented alongside some common standards used in the industry. Finally, a more in-depth analysis is made to multiple academic works carried out in the area and also of two systems currently available on the market and similar to the one proposed.

2.1 Industrial Production System Overview

Manufacturing can be defined as a value-adding process which consists in transforming raw materials into finishing goods, through the use of resources such as human labor, machines, tools or chemical processes. The use of information technology in industrial manufacturing has driven companies to adopt more agile processes and more powerful industrial production systems, which allow them to operate more efficiently and remain competitive in the manufacturing market [SBZ08]. This means having production facilities well structured and manufacturing support systems that boost production speed and product quality, according to standards. By having a proper production system, manufacturing companies can also guarantee not only a quality delivery for the final product to the clients but also a reduction of costs in all production chain [Gro15].

Production facilities, as an important part of production system, consist of the company's factories, its equipments, machines, tools and also the way everything is organized in space [Gro15]. They are influenced by the quantities of products produced by the factory itself, but as the scope of this dissertation is the development of an information system for SMEs, facilities that deal with mass production quantities will not be addressed, as they require more complex information systems that need to handle large amounts of data. On the other hand, SMEs are characterized by a more flexible and dynamic environment [SMMM09], thus requiring a more proactive production facility that is capable of reacting more easily to changes in manufacturing process that happen in short periods of time.

The second category of production systems are manufacturing support systems, which consist of "procedures and systems used by a company to solve the technical and logistics problems encountered in planning the processes, ordering materials, controlling production, and ensuring that the company's products meet required quality specifications" [Gro15]. This means that manufacturing support systems will perform activities within the scope of business functions, product design, manufacturing planning and manufacturing control. The last two functions are the focus of the proposed system, as they cover shop floor activities, like controlling execution of orders and resource monitoring, and quality control activities, like defect reporting or quality data analysis.

The evolution of information systems in manufacturing culminated in three distinct types of systems with different purposes and that deal with data with different levels of detail and in different time intervals. Figure 3.1 lists these three types of systems. There are business systems, like Enterprise Resource Planning (ERP) systems, manufacturing systems, like Manufacturing Execution Systems (MES), and Process Control Systems, like Programmable Logic Controller systems. A manufacturing company may not have all these types of systems in its production chain, but the implementation and integration of all of them can significantly increase the efficiency of the manufacturing process [SBZ08].

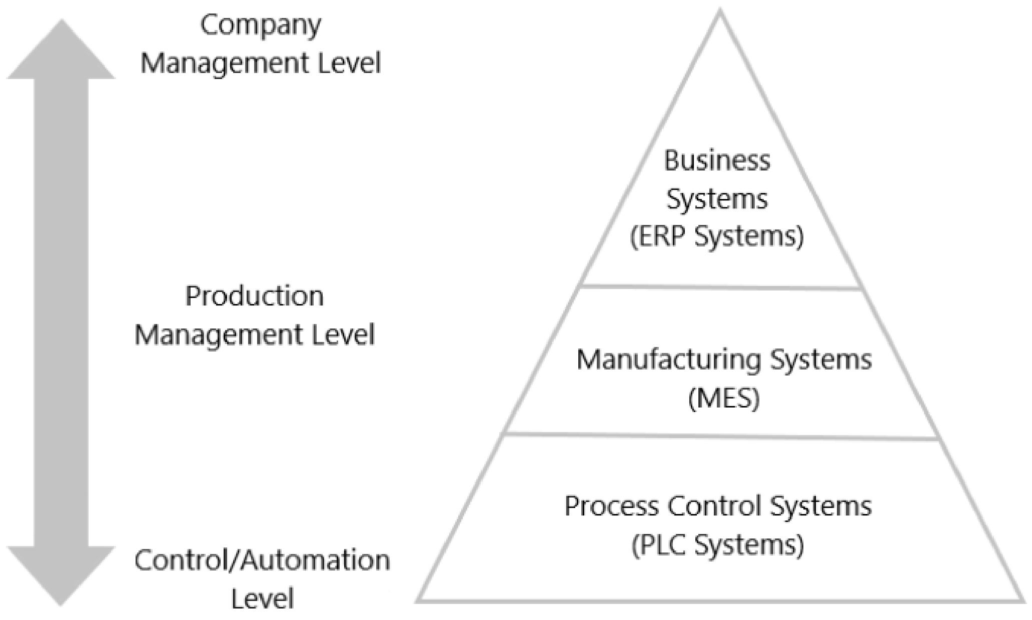

Company Management Level
Production Management Level
Control/Automation Level
Business Systems (ERP Systems)
Manufacturing Systems (MES)
Process Control Systems (PLC Systems)

Figure 2.1: Types of information systems present in a manufacturing company

The future of manufacturing companies relies on smart manufacturing due to increased reliance on information systems for manufacturing process control, rather than human operation. Smart manufacturing is an approach that aims to provide real-time solutions to "changing demands and conditions in the factory, in the supply network and in customer needs", by using fully integrated and collaborative manufacturing systems [Kus18]. This can be achieved with the help of technological innovations such as Internet of Things (IoT), Service Oriented Architecture (SOA), data science, cloud computing or Artificial Intelligence (AI), which are all linked to the new concept of Industry 4.0.

2.2 Manufacturing Execution System

The MES concept was first introduced in the manufacturing industry in the early 1990s, with the aim of merging multiple data collection systems, with overlapping functions, into a single system that contained real-time production data [Kle07]. A MES is, therefore, able to control production operations based on real-time data, collected by sensors or operators in the shop floor.

Until the introduction of MES, specialized custom applications were used to handle specific tasks in functional groups like production planning, personnel management or quality assurance. But with the increased demand for these systems to be connected and integrated with each other, it was no longer feasible to have separate applications, as synchronizing all data was very complex and time consuming [Kle07]. So, the only solution was to combine all of these isolated systems into a single, standardized one, where all relevant data for production process would be available when needed.

This type of system was also designed to complement corporate level tools, for instance ERP systems, which performed company functions in an end-to-end scope, and Manufacturing Resource Planning (MRP), used to schedule production operations according to available resources [Cro11]. Unlike MES, which has the characteristic of handling real-time production data, corporate systems are not provided with the same instant data. Instead, they rely on an intermediary to maintain the data flow from shop-floor to business level. MES offered to automate this data exchange, while allowing to restrict the visibility of data according to predefined roles, which was not possible in top level systems, since they were only available for management roles.

In the context of globalization, manufacturing companies faced several challenges to be faster and to improve quality while reducing costs. This increase in pressure to be more efficient was also fought with the addition of MES to the production chain [MFT09]. This system increased transparency and the power to draw conclusions and make correct decisions at an early stage. The standardization of this system was also an important step on the path to achieve a solution that could be implemented in different manufacturing situations.

2.2.1 Models and Standards

Since the appearance of computer systems for manufacturing, several organizations and institutions have sought to implement standards and guidelines to optimize the use of these tools in production activities. Regarding MES, the first organization to ever work upon the subject was the Manufacturing Enterprise Solutions Association (MESA). This association was also the first to propose a formal definition for MES [Int97].

MESA Models

In 1997, MESA gathered a set of rules and published its first model, which defined the core operations that should be present in a MES and also its relations with external systems. Functions such as operation scheduling, resource allocation status and product tracking were some of the eleven operations mentioned in the model [Kle07]. Over time, these operations were redesigned and business operations were also added, giving rise to a new version of the model, where the focus was on collaborative manufacturing and how different systems, related with manufacturing process, could interact with each other.

It was thanks to this growing interest in collaborative manufacturing, that MESA continued to shift its focus to how a manufacturing enterprise should work to achieve its strategic goals, throughout all available systems. These strategic objectives, or initiatives, are the ones that drive the current version of the MESA model, as illustrated in figure 2.2. Although operations at the business and manufacturing level remain practically the same from previous versions, new relationships between both levels and strategic initiatives have been added, through the means of objectives and results [Int20]. This means that if we seek to develop a MES solution that follows the guidelines of the current MESA model, we must take into account not only the manufacturing operations, but also the links with systems at different levels of the manufacturing enterprise, because only this way its possible to achieve the strategic goals defined.

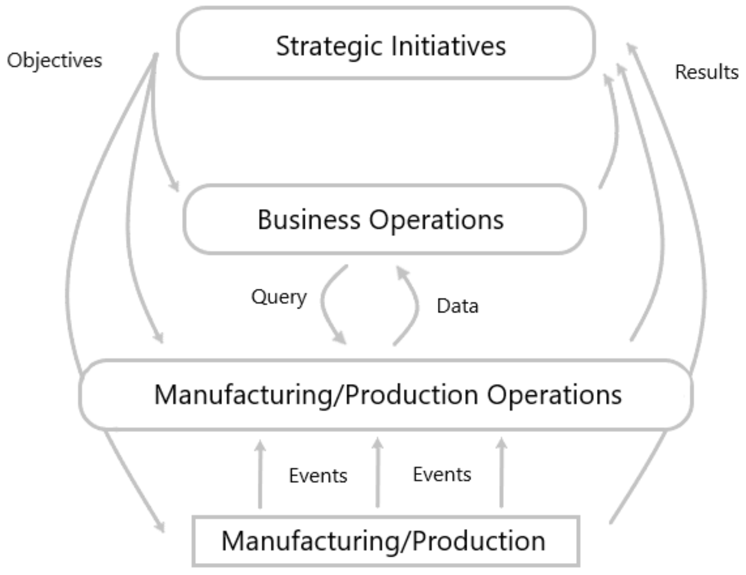


Figure 2.2: Mock of current version of MESA Model, developed in 2008

ISA Standards

The International Society of Automation, or ISA, is another example of an organization, composed by multiple volunteers, that develops global standards widely used in the manufacturing industry. These standards aim to improve performance in production processes and are used by others as base concepts for their projects [ISA20].

When developing a MES, two ISA approved standards can be considered, the ISA88 and the ISA95. The first standard establishes some guidelines for batch control systems in manufacturing industries. It is divided into four parts, the first of which addresses the concept of recipe, as a set of production instructions for building a designated quantity of products. This standard also addresses topics such as data models structure, standard data analysis and consequent results and, finally, the subject of data archiving [MFT09]. All of these concepts are fundamental for an efficient MES, as most manufacturing companies work with batch processes.

The other standard mentioned, ISA95, is also very important, as it defines an interface between the enterprise and control systems of a manufacturing company of any type [MFT09]. Throughout its five levels, this standard describes the three layers of a manufacturing company (business management, production management and automation) and their interactions with each other to provide a continuous data flow. The functions and objectives of each layer are also exposed to assist in the efficient implementation of a system solution, so that in this way, duplicated and missing functionalities can be avoided during integration of all systems. Usually this standard is illustrated as it can be seen in Figure 2.3, where level 0 to 2 correspond to the first layer and level 3 and level 4 correspond to the second and third layers respectively.

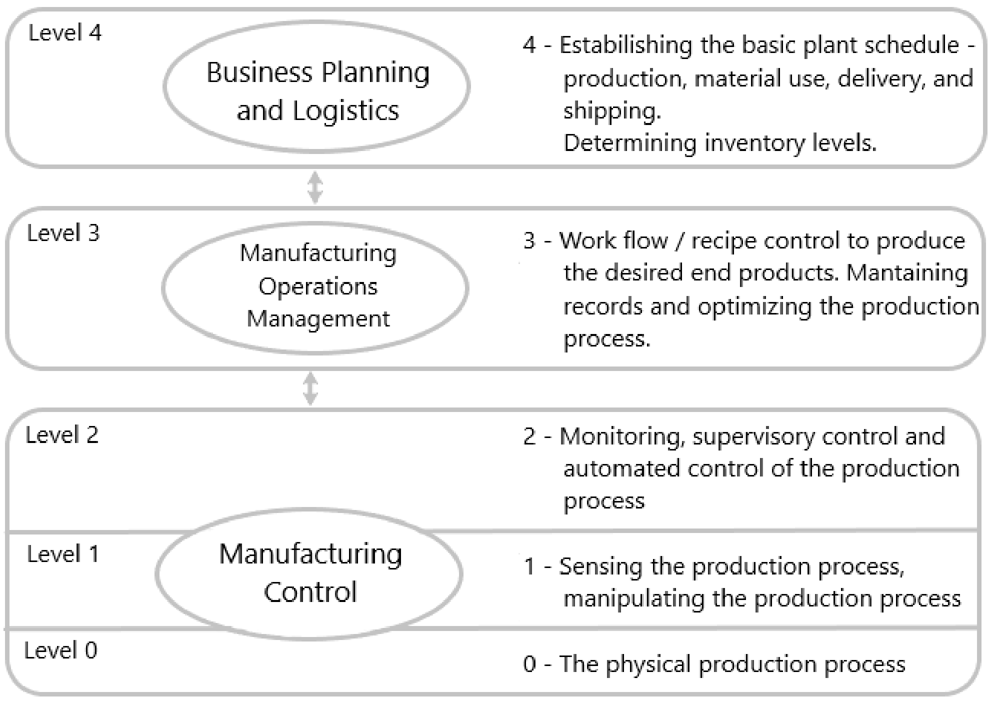

Level 4
Business Planning and Logistics
4 - Estabilishing the basic plant schedule - production, material use, delivery, and shipping.
Determining inventory levels.
Level 3
Manufacturing Operations Management
3 - Work flow / recipe control to produce the desired end products. Mantaining records and optimizing the production process.
Level 2
2 - Monitoring, supervisory control and automated control of the production process
Level 1
Manufacturing Control
1 - Sensing the production process, manipulating the production process
Level 0
0 - The physical production process

Figure 2.3: Representation of the model defined in standard ISA95

2.2.2 Functional Areas

The scope of MES can include activities from three different functional groups. These groups are production management, quality management and personnel management and, although each serves different areas of a manufacturing company, they can be linked through common activities between them [Kle07]. The implementation of these activities in the system, however, will be dependent of a contextual analysis of the manufacturing environment where the system will act, since the requirements may differ between factories [AP18].

In terms of production management, the system should include functions that deal with the management of product life cycle and take place on the factory shop floor, thus ensuring control of the entire production process. Production process begins with the planning and dispatching of orders, as the system should allow to establishing the best sequence to deal with the orders on the working queue. Other functions provided by MES should be the schedule and monitoring of all tasks needed to deal with an order, and these actions should also include the management of resources that should be allocated for each operation to be performed. On this category, the system shall be able to monitor performance and status of the full production process through outputting machines and operations states [MFT09].

Quality management activities are important to ensure the quality of the product delivered by the manufacturing company to the customer and can help to detect bad practices and defects on the shop floor level, during the production process. The system should have a strong data collection mechanism in order to store and exchange a maximum amount of data related to everything that is happening on the shop floor [Kle07]. This raw data can then be worked on and provide useful information to assess the efficiency of the factory. Modern MES systems have a statistical component to display this information through tables or control charts, to facilitate analysis [Cro11].

The final functional area where MES can have an impact is the management of personnel or human resource. Usually, activities related to this, such as employee management or payroll processes, are handled at company level by ERP systems. Nonetheless, manufacturing companies can benefit from implementing some of these activities at shop floor level with MES. For example, by recording employee working hours in MES, its easier to plan the production schedule, as the visibility of human resources available is better [MFT09]. For matters of transparency, managing employee data directly on shop floor can help access productivity according to which employee was working at different times.

2.3 Related Work

The rise of industry 4.0 coupled with the use of new advanced technologies, such as cloud computing, IoT, artificial intelligence or virtualization, has been shown to be the main reason for an increase of interest in the development of information systems solutions in the manufacturing sector.

One of the first studies to explore these technologies applied to manufacturing, was [TZV^{+}11], where a new concept of cloud manufacturing was proposed, as a service-oriented manufacturing model that allows for the sharing of manufacturing resources and capabilities over the internet and on demand. This topic was further discussed in [HLYZ13] with the introduction of a cloud manufacturing service platform aimed at SMEs. With this work, it was possible to look deeper at the transformation process from production-oriented manufacturing to service-oriented manufacturing, since this type of enterprises gained access, according to their needs, to virtualized manufacturing resources, stored in different manufacturing clouds, that were made available through the platform.

Still in scope of cloud computing and SOA, in 2014, Helo et al. [HSHA14] developed a prototype of a MES for distributed manufacturing. This system was designed to make up for the lack of solutions in situations where the production chain had more than one participant in different factories and there was a need to exchange information between them. The proposal was based mainly on cloud computing technology, because in this way it was possible to centralize data in a single location, making it accessible for multi-factory management.

Other important aspects in recent information systems for the manufacturing sector are device interface and accessibility, on the shop floor, through mobile devices. As web-based solutions are gaining popularity, works like [JZLZ15], demonstrate the possibility of implementing a MES as a web application, which is accessible through the browser on a regular computer or mobile device. This paper also presents a multi-user role configuration designed to handle the wide range of operator permissions that can be found on shop floor.

The benefits of having information systems for monitor production and controlling operations at the shop floor have led to an increase in the number of MES solutions on the market. The degree of complexity is not always the same, making it difficult to choose the most appropriate alternative for each situation. In addition to cloud hosted solutions mentioned earlier, there are also types of systems that need to be installed on-premises. The way that the system is made available to end users is also an important characteristic, as it can be open-source or proprietary software. MPDV Hydra and Qcadoo are two of these systems, and they are seen in detail bellow in order to analyze the differences between both.

MPDV Hydra MES

Hydra MES, developed by MPDV, is a modular MES, which follows the ANSI/ISA-95 standard. Its modularity allows Hydra's end user to select only the necessary modules that meet all business requirements. In addition to the modules, in Hydra it is also possible to incorporate platform tools to strengthen the system [MPD20].

Within the module structure there are three functional groups: manufacturing control, quality management and human resources. Each of these groups is composed of several applications that serve different purposes. It is also important to note that this system follows the SOA guidelines, as each application has access to a wide range of services, that in turn are combinations of software modules that can be modified independently [KD18].

The complexity of the system can be seen as the main challenge for implementation in a manufacturing environment, especially for companies that require a considerable number of modules to meet their needs. There may also be an integration problem with existing systems, as they may not be compatible, thus requiring a complete restructuring of the entire IT infrastructure. If this is not the case, the installation process will consist in setting up, as a central part, Hydra MES server (which will contain the logic for functionalities and database for real-time production data), integrating the system with shop floor terminals and company ERP system and finally connecting the server to operation centers, which should be distributed through facility computers [KD18].

Qcadoo MES

Qcadoo MES, developed by Qcadoo Limited, is a MES solution that, in addition to having a commercial edition, also offers an open source community edition, available in a Github repository [Git20], which allows for programmers outside of the company to see the source code and request changes. This approach is very popular to reduce the development costs, however, when using the community edition the end user will not have any kind of official support [Qca20].

The system is cloud hosted and delivered through a web application built on top of Spring framework. As long as the web application is running on a local server or in the cloud, the end user does not need to perform any type of installation and the system can be accessed through any internet browser. The system architecture is based on a modular structure, similar to Hydra, but because it offers less functionalities, it is more suitable for SMEs. It is also important to note that the system can also be integrated with other solutions, such as ERP, but once again the compatible applications are more limited when compared to Hydra [Qca20].

3 The Proposed System

In this chapter, an overview of the proposed system is presented along with the goals and objectives of this implementation. The solution requirements are also described taking into account insights from a manufacturing company. The steps and decisions taken during the implementation of some of the main modules of the system are also detailed, as well as the deployment process.

3.1 Overview and Goals

As it was mentioned previously the main goal of this dissertation is the implementation of a system for resource and process management in an industrial environment. A local aeronautic industrial facility will serve as a partnership for the implementation of the application and consequent execution of tests. The main target for this system are SMEs, since mass production companies require a more complex solution as there are a lot more data to be handled and processed.

A web application, deployed in the facility server, is the method chosen to make the system available to all operators, since this option allows any user to access the application, from any device connected to the factory's internal network, as long as they have an internet browser. This approach is more beneficial than a desktop application solution, because developers don't have to make different versions for multiple platforms and also there is no need to distribute application updates to each individual user.

Regarding functional features, the system should satisfy the common needs of a shop floor operator and follow the guidelines of a MES, mentioned in the previous chapter. The scope of this solution should therefore include features from production, personnel and quality management.

3.2 Requirements Definition

With the support of the partner manufacturing company, it was possible to outline the system requirements and divide them into several sections in order to cover the different areas of a regular manufacturing workflow. These sections specify what the correct system behavior should be and can be changed in the future to accommodate further requirements. They are described bellow:

- **Purchase Orders Module**. Purchase orders are an essential component of the manufacturing process. They are the initial item of a production workflow and are what determine the work that is queued and ready to be executed. The system must allow, for a pre-defined group of users, the ability to view, create, edit and delete orders. The system should also validates if the proper configurations to work on the order are met by the facility before executing it.

- **Production Jobs Module**. Production jobs are a set of manufacturing operations that take place on the shop floor to create a final product. To control this aspect of manufacturing, in this module, the system should provide, for a set of users, the ability to track these operations and consult the real time status of the jobs. The analytic component is also an important element of this module, since dashboards and tables can facilitate the management of facility's production jobs.

- **Equipment Module**. The management of factory equipment, like machines and production tools, must also be included in the system in order to provide to the operators the ability to check the status of a specific machine or to check if is there any problem. A representation of the shop floor layout can be present in the application, to help planning and control a product manufacturing process in a production line located in the shop floor.

- **Personnel Module**. The personnel module is more relevant to the human resource sector of the industrial factory and to the corporate level. Here it should be possible to register facility operators and include their functions as workers. This will help restrict operators visibility across the application by giving to the users only the permissions compatible to the functions they perform. It is also required to track the user activity in time for specific jobs, to generate production performance data that can be used to improve the current production process and to detect anomalies.

- **Statistic Module**. The statistics module can encompass the jobs dashboard already mentioned and all other charts and tables that handles production data for performance and quality assessment. The purpose of this module is targeted at the corporate level of the factory, as it can help them make better decisions to improve manufacturing process according to data collected in real time. It also gives an idea of how the factory is working without having to be present on the factory floor.

3.3 Choice of Technologies

The development of a web application needs to include a server-side section, to handle requests and process any logic that the application may require, and a client-side section, to render responses that can be understood by the end user. For building each one of these sections there are a wide range of technologies such as tools, platforms and frameworks, with different characteristics, available for use. The task of choosing the best ones is fundamental, and should always be done taking into account the application requirements.

One of the options to consider when planning to build a web application is Node.js and its multiple available web frameworks, such as Express. Node.js is essentially a JavaScript runtime environment, built on top of Google Chrome's JavaScript V8 Engine, where it is possible to execute Javascript code on server-side [Nod20]. This feature makes it perfect for full stack development, as the programmer can use a single language, Javascript, for front-end and back-end development.

Node.js is also known for its event-driven architecture, perfect for handling large-scale I/O applications in an asynchronous way. However, this feature can become a problem when the goal is to build applications with computationally intensive tasks, as the event loop can be blocked and other requests can be impacted [D'm17]. That's why using Node.js is most recommended when building real-time web applications. Other topic, regarding Node.js, that deserves to be highlighted is the big community, that supports this technology, and the large marketplace that offers a wide range of tools and open source libraries to be reused by any programmer. Nonetheless, some of the solutions that we can find in the marketplace may not be of good quality and harm the project where they are integrated, since the marketplace is quantity-driven and not quality-driven.

An older alternative to Node.js is Django web framework. Django is an open source Python web development framework, where the main focus is to build fast web applications and avoid code repetition. This framework is very popular, as it makes use of Python programming language, which is a versatile and beginner friendly programming language. This doesn't mean that Django is not robust enough, since large websites, like Instagram or Bitbucket are built on top of this framework [Vin18].

Django has a vast range of modular tools and packages that facilitate aspects of web development, such as user authentication, administrator interface or security aspects. It is also regarding security that this framework has one of its main advantages, as it can alert and prevent common attacks such as Cross-site request forgery or SQL Injections [Poc20]. On the negative side, Django can generate web applications with bad performance, if they are not well planned at an early stage. It may also not be the right choice when developing small applications because the amount of pre-installed components can make the application slower than it should be.

The last web development framework covered in this dissertation is Spring, which is the most popular Java framework. This framework allows programmers to build applications with Java programming language more quickly, easily and securely, as it offers a wide variety of modules and dependencies [Spr20c]. Inversion of control and dependency injection are also two other important features that can be used in Spring in order to achieve more modular and easy to test applications.

In the field of web development, Spring provides an extension, called Spring Boot, that facilitates the process of setting up a new application from the beginning, reducing any upfront configuration. In the modular aspect of the framework, there are three almost mandatory modules that should be part of a Spring web application. Spring web, for basic web-oriented features [Spr20a], Spring Data JPA, to connect and interact with any database, and Spring Security, to handle security aspects like authentication. Despite all these pre-built modules, Spring can be a bad choice for web development if the development team has no previous experience with the framework and Java language, as the learning curve can be difficult for this tool.

After reviewing each one of these technologies in detail, some differences can be identified when all of them are compared to each other. Even though all of them are good options for web development, the scope and requirements of the application to be built will determine which one would generate an application with better performance and quality [vsC20]. For example, Node.js, due to its single threaded processing characteristic, is best suited for web applications that do not require heavy CPU processes. On the other hand, if the goal is to build complex applications with enterprise focus or elaborate logic, Spring can be seen as the most appropriate, due to its reliability and libraries maturity [DZo19]. Finally, if the goal is to build an application as quickly as possible, Django should be the best solution, as it uses a well-known and easy-to-learn programming language and because it provides many components that speed up the development process.

Finally, after all the comparison, the choice of the most appropriate technology to implement the proposed system fell over to Spring framework. Spring was chosen based on all the features mentioned above, but also due to the author's familiarity with the Java programming language, which avoids any learning complications that could block or delay the implementation process. In addition, this framework was already used in previous projects and ended up being a good approach for implementing Model View Controller (MVC) systems, which will be the design pattern in use for the proposed system.

3.4 System Architecture

The system architecture can be explained as a web application that follows the MVC design pattern. This pattern divides the application into three different layers: model, view and controller, which act as independent components and allow developers to work in different parts without impacting the entire system [KSB16]. The model component represents application data structure and in this specific case it is represented as Plain Old Java Objects (POJO), which correspond to the entities present in the related databases. This data is then presented to the user in the view layer of the application, for interpretation. In this project, the view layer is composed of JavaServer Pages (JSP), that are dynamic web pages based on HTML. Any action performed by the user in the application is then handled by the controller layer, that acts as an intermediary between model and view, since it builds the appropriate model and passes it to the view upon any request.

The database structure for the proposed system is based upon two distinct data sources, one for business data and other for authentication data. The database for business data, like orders, jobs, and operations, was already implemented in SQL Server for this use case, so the structure was not changed and the application was built around the existing tables. For authentication and authorization purposes, there was still lacking a viable solution that kept credentials and permissions safe, so it was decided to create a different database in PostgreSQL, in order not to disturb the existing one. This last database contains tables for users, roles and the relationship between both. The decision to have a solution with two different data sources was also supported on the fact that this approach can be seen as more secure, since all data is not in the same place, and also this way allows the authentication database to be used by other applications without sharing visibility of business data present in the existing operations database (in Microsoft SQL Server database).

Other important topic to be mentioned in the proposed system architecture is the list of dependencies used by the application. As the system is a Spring project based on Maven, we can find all the dependencies used in the pom.xml file. Here, each one of the dependencies names are listed next to its specific version. Among the most crucial dependencies, it is possible to find Spring Data JPA for data access and persistence topics, Spring Web for essential web application developments and Spring Security for security aspects.

In Figure 3.1 it is possible to see the complete architecture of the proposed system and also the workflow of actions for a standard HTTP request. The chain starts when a user sends an HTTP request from a web browser. This request is received by the dispatcher servlet, that consults the Handler Mapping in order to redirect the request to the appropriate controller. The controller then takes the request and invokes the correct method according to the type of request. The purpose of the method is usually to return a view name and a model as well. To build the model, the controller may require some logic from the service or repository layer. The next step is for the dispatcher servlet to interpret the view name received and choose the correct view, with the help of View Resolver. Once the view is complete the dispatcher servlet passes the model built by the controller to it, and renders the page in the user's browser.

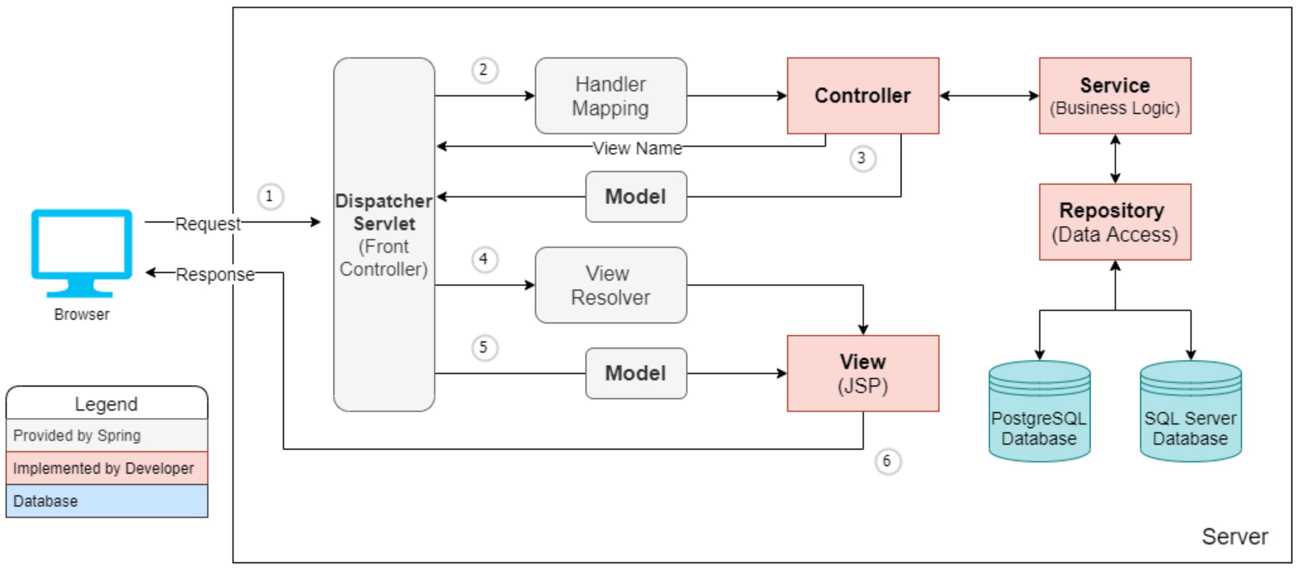
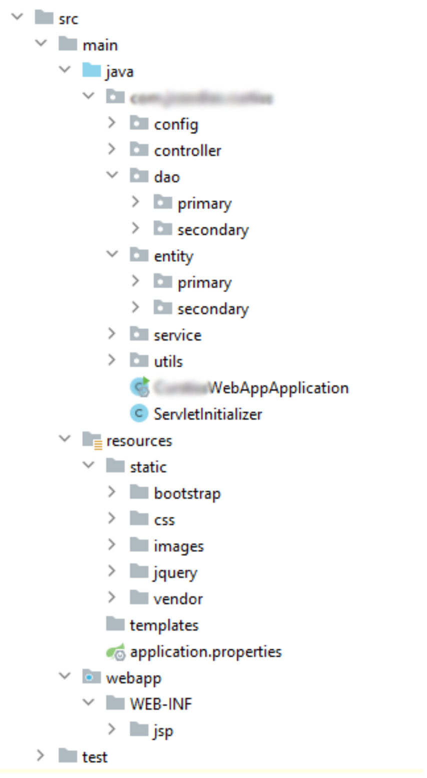


Figure 3.1: Architecture of the proposed system

Like the system architecture, the defined project structure also follows an approach divided by layers, as shown in Figure 3.2. There are a total of six main packages for storing java classes separately related to configuration aspects, controller implementations, data access operations, entities definition, service and utility logic. In this structure, it is important to note that dao and entity packages are also divided into two separate packages, that correspond to each one of the database systems in use by the application, primary package for manufacturing operational database (SQL Server) and secondary package for authentication and authorization database (PostgreSQL). All JSP, which correspond to the view layer, are stored together in a unique folder in the path *webapp/WEB-INF*. Finally, there is also a resource folder for storing items such as images, style sheets or bootstrap files used in the system's user interface.

3.4.1 Database

The typical MES, which was studied in chapter 2, makes use of a database to store real-time production data collected on the shop floor, and the proposed system is no different. As mentioned earlier, the database used for the intent of storing production data was already created in the associated factory server. This database was implemented with Microsoft SQL Server and had the essential table structure for this use case. It contains tables for the following entities: purchase orders, customers, prices, jobs, operations,



```
src
  main
    java
      [redacted]
        config
        controller
        dao
          primary
          secondary
        entity
          primary
          secondary
        service
        utils
        [redacted]WebAppApplication
        ServletInitializer
    resources
      static
        bootstrap
        css
        images
        jquery
        vendor
      templates
      application.properties
    webapp
      WEB-INF
        jsp
  test
```

Figure 3.2: Application code structure

configurations, machines, quality certificates and also others not used during the implementation of the proposed system. It also contains a one-to-many relation between jobs and operations table, to keep track of which operations were executed for each job performed on the shop floor.

Since SQL Server was linked to enterprise systems at corporate level, it was also possible to use this database as an integration point to link the proposed system, on the shop floor level, with other business applications, at a higher level within the company. This, however, created a challenge, because as this database was linked to other systems, it was not possible to make changes in any aspects in order to avoid the risk of disrupting business workflows already implemented in other systems. This challenge was overcome by developing the entire system around the existing database structure. The option of building a new database was not viable because this would eliminate integration with enterprise systems.

Although the database already implemented contained a user table, its properties were not suitable for implementing a secure authentication system with an authorization permissions component. Fields such as username or password were absent or poorly configured in the original database, and there was also no table to save user permissions. Thus, it became necessary to create a new data model for this section. This action involved the creation of a new database on a PostgreSQL server due to the impossibility of making changes in the Microsoft SQL Server instance, already configured before the implementation of the proposed system.

The data model for this new database can be seen in Figure 3.3. It contains three related tables, with the objective of having a table to store users credentials, a table to store the permissions available for all users and finally a table to link multiple permissions to each user of the system.

For both databases, Spring Boot Starter Data JPA dependency was used to map database tables into Java entities, in the form of POJOs, and also to simplify the complexity of performing database operations

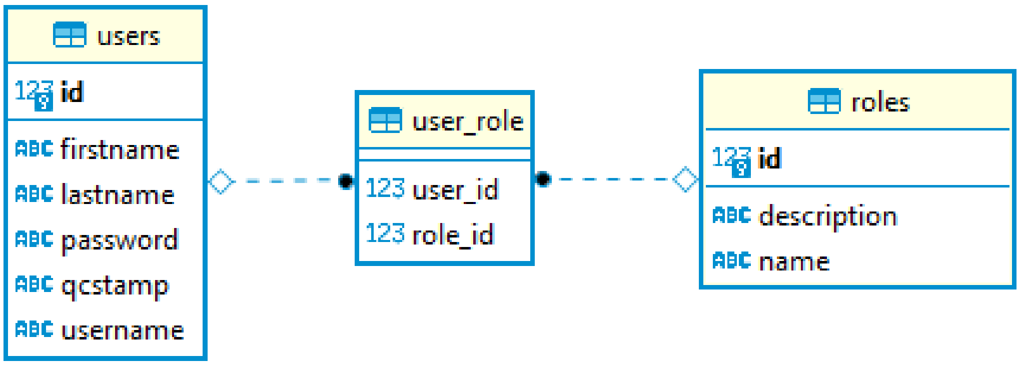


Figure 3.3: Data model for authentication database create with PostgreSQL Server

through code, by reducing the amount of 'boilerplate' code used by the JPA dependency. Listing 3.1 shows the structure of how a table can be represented as a Java entity, with all the table fields represented as Java variables. Through implementation of repository abstraction, as shown in listing 3.2, it is possible to see how database CRUD (create, read, update, delete) operations or database query executions are performed.

```java
import javax.persistence.*;
import java.io.Serializable;
import java.util.List;
import java.util.Objects;

@Entity
@Table(name="users")
public class User implements Serializable {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    @Column(nullable = false, unique = true)
    private String username;
    private String password;
    private String firstName;
    private String lastName;
    private String qCStamp;

    @ManyToMany(fetch = FetchType.EAGER, cascade=CascadeType.MERGE)
    @JoinTable(name="user_role", joinColumns={@JoinColumn(name="user_id",
        referencedColumnName="id")}, inverseJoinColumns={@JoinColumn(name="
        role_id", referencedColumnName="id")})
    private List<Role> roles;

    // Implementation of get and set methods
    // Implementation of equals and hashCode override methods
}
```

Listing 3.1: Implementation of entity User.java mapped to table users in PostgreSQL database

```java
import com.*.*.entity.secondary.User;
import org.springframework.data.jpa.repository.JpaRepository;

public interface UserRepository extends JpaRepository<User, Long> {
    User findByUsername(String username);
}
```

Listing 3.2: Implementation of UserRepository.java for database operations in users table

Regarding database structure of the proposed system, it is also important to note that, as the system is connected to two distinct data sources there was the need to add both Microsoft SQL Server and PostgreSQL Server drivers and to configure Spring application to know which of the data sources to use according to the operation to be performed. This was accomplished by creating two new Java configuration classes that mapped the two database credentials, located in the properties file, to the correct package that contained the corresponding entities.

3.4.2 Security

Security aspects are a major concern in systems implemented in manufacturing environments, as these systems work with confidential data and unauthorized access can cause serious damage to the entire company and not just to the facility where the system is implemented. Spring security allows to set restrictions in Spring web applications according to programmer needs, especially in important areas such as authentication and authorization [San20].

To use Spring Security for web security, it was necessary to implement a Java security configuration class that enabled aspects of web security through annotations, such as *@Configuration* and *@EnableWebSecurity*. In this class, the method *configure(AuthenticationManagerBuilder auth)* was overridden to deal with aspects of authentication with data from PostgreSQL database. This method makes use of a service class that searches for user records by their username. If there is in fact a user with the mentioned username, Spring Security validates the password inserted against the password present in the database. It is also important to note that passwords are always encrypted using BCryptPasswordEncoder [Par20], which uses BCrypt hashing algorithm, to prevent passwords from being stored in plain text on the system and also in the database.

To deal with authorization matters, the method *configure(HttpSecurity http)* has also been overridden in the same configuration class, as seen in listing 3.3. This allowed the configuration of filters to restrict access to system resources, such as HTTP endpoints, according to user permissions. Login and logout endpoints behavior are also configured in this same method. To validate that a user has the necessary permissions, Spring has the *hasAuthority* method, which queries the list of permissions assigned to the logged in user. This method is also used in JSP to restrict the visibility of specific sections in the front end.

```java
@Override
protected void configure(HttpSecurity http) throws Exception {
    http.authorizeRequests()
        .antMatchers("/css/**").permitAll()
        .antMatchers("/images/**").permitAll()
        .antMatchers("/register").permitAll()
        .antMatchers("/orders/**").hasAuthority("ORDER_MANAGEMENT")
        .antMatchers("/bookIn/**").hasAuthority("JOB_MANAGEMENT")
        .antMatchers("/users/**").hasAuthority("USER_MANAGEMENT")
        .antMatchers("/roles/**").hasAuthority("ROLE_MANAGEMENT")
        .anyRequest().authenticated()
        .and()
            .formLogin()
                .loginPage("/login")
                .defaultSuccessUrl("/menu", true)
                .permitAll()
        .and()
            .logout()
                .logoutRequestMatcher(new AntPathRequestMatcher("/logout"))
                .logoutSuccessUrl("/login")
                .invalidateHttpSession(true)
                .clearAuthentication(true);
}
```

Listing 3.3: Configuration of authorization and authentication filters for system HTTP endpoints and resources

3.5 Detailed Approach per Module

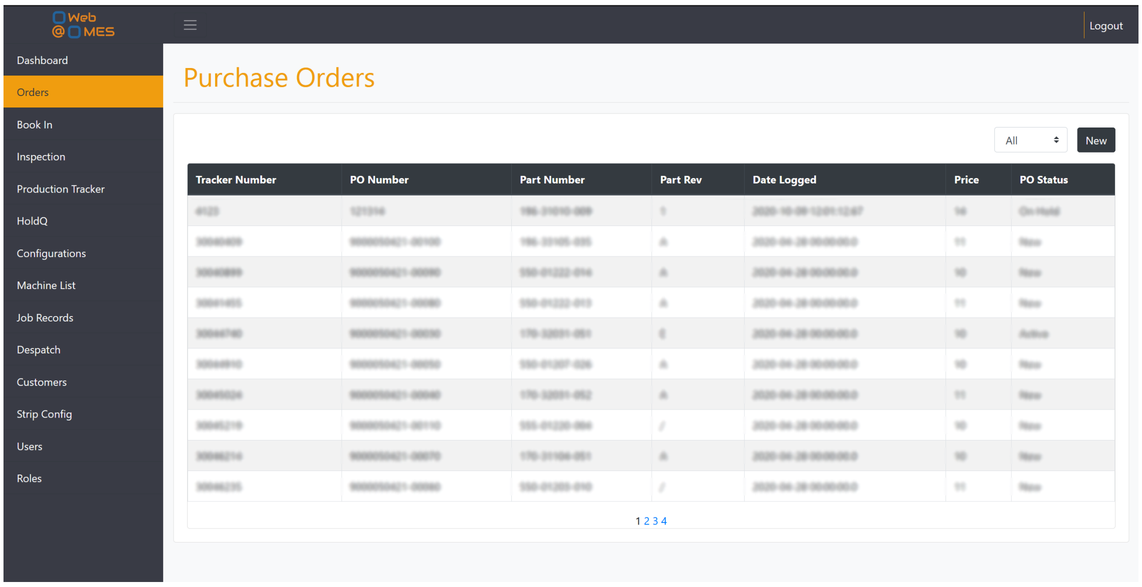
3.5.1 Order Management Module

The requirements for the order management module state that only a user with the correct permissions should be able to view, create, edit and delete purchase orders in the system. In the original database, a purchase order is characterized by a set of fields that include a unique tracking number, configuration details, a registration date, a status and a price. In the system, the table is mapped into a Java class that references these same fields in the form of object properties.

The result of implementing a consult mechanism for a list of purchase orders is represented in the Figure 3.4. On this screen, the operator can view various information about diverse orders, present in the system, in a single table and can also filter the list by order status. This filter logic was implemented by adding to the model (which will be rendered in the view) a list of purchase order instances that represent a portion of the records retrieved by a query performed over the database, in the order repository class. Because this class extends *PagingAndSortingRepository* interface, its possible to use sorting and pagination features in the executed queries [Sri20].

Regarding the purchase order creation process, the requirements stated that an operator, with the correct permissions, should be able to create a single order manually or insert into the system an excel file with multiple orders inside. Figure 3.5 shows the screen where the operator can chose between this two methods.

The manual insertion component is implemented using the form tag provided by Spring framework. This tag renders an HTML form and also binds the form fields to Java object properties [Spr20b]. For this



Purchase Orders

Tracker Number	PO Number	Part Number	Part Rev	Date Logged	Price	PO Status
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]

Figure 3.4: View of table with filtering and pagination options of multiple purchase orders

specific use case the form created is linked into a purchase order entity. After an operator submits the form, if all mandatory fields are filled, a validation is performed to verify if the tracking number is unique and also to verify that the price entered is in accordance with the ones mentioned in the system, for the configuration asked in the order. If the price is not referenced in the system, the order is created, but placed on hold until someone with the appropriate permissions confirms the information. Both validations were requested by the partner company and implemented in the service layer in order to be invoked by the controller when necessary.

For the import section, it was necessary to add the Apache POI dependency to the project, in order to be able to manage Microsoft documents in the Java source code. By offering a Java API, this dependency allows to read and write on XLS spreadsheets [Fou20], which is the type of files used by the operators in this use case. As the import files always follow the same structure the system only reads the necessary columns, while iterating all lines, to create a list of purchase orders entities. This list is then placed in the model and rendered in the next view so that the operator can confirm the information read by the system and change anything, if necessary. After submission, tracking number and price validations are performed on all orders and, if there is a problem, an error message will be displayed mentioning unsaved orders.

3.5.2 Job Management Module

In the system's job management module, the main requirement is the creation of jobs from queued purchase orders and the consequent dispatch of new jobs into production. This means that the operator, with permission to create and dispatch jobs, needs to have visibility, in the system, to all purchase orders available to be worked on (status equals to new).

When selecting one of these orders to dispatch, the system validates the configuration referred in the purchase order, in order to notify the operator if there are enough resources to work on the job, in the facility. If this configuration is not present in the system, the order is rejected and cannot go forward in the production process. Otherwise, if the configuration is available, the new job can be created and dispatched as normal.

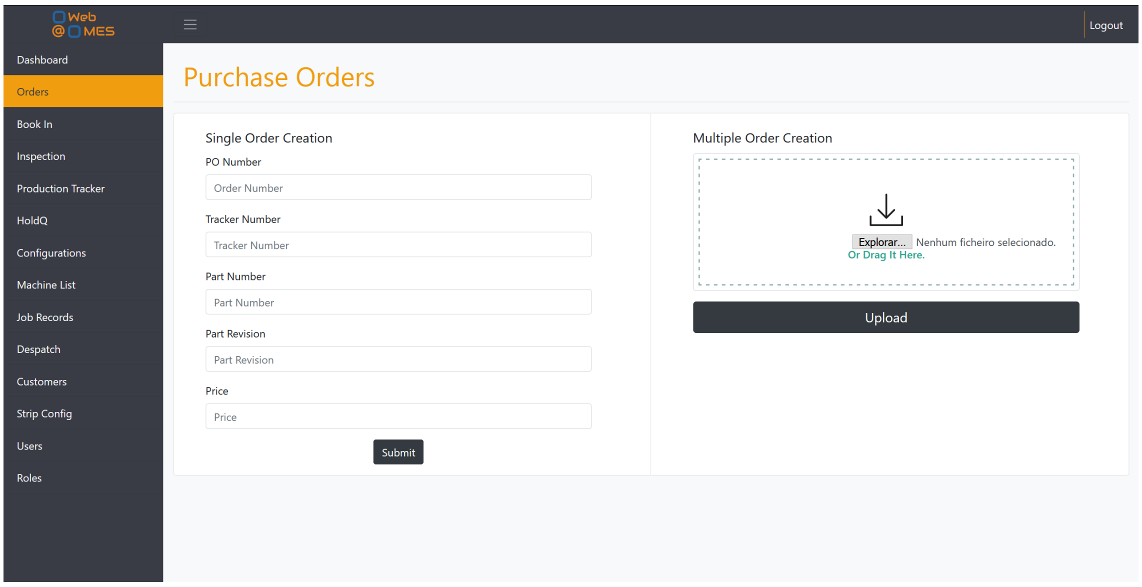


Figure 3.5: Page where the operator can choose the method to insert orders in the system

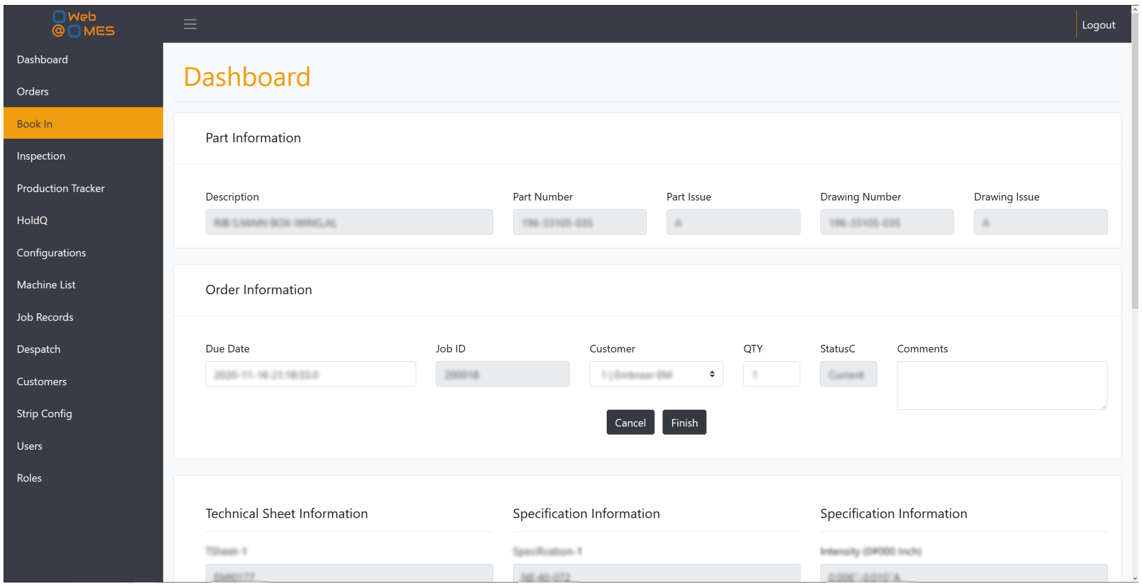
The behavior of showing the purchase orders ready to be worked on is replicated from the previous module with a small difference regarding the query performed in the database. To avoid sending to production jobs related to orders already picked up by another operator, in the repository abstraction only purchase orders with status equal to new are fetched.

After the operator selects a purchase order from the queue, the validation process begins. In the system, this logic is implemented by querying the configuration table, in the database, for records with the same values as the configuration fields specified in the order. If there isn't a match, the system will remain on the same page and display an error message. If there is indeed a record in the database for such configuration, the purchase order is sent to the next view and the booking in process can start.

The job creation page, shown in Figure 3.6, with which the operator interacts in the next step can be divided into three sections. It has a section for configuration information, one for the job details and, finally, a section for technical information about the operations that will be performed in this new job. Only the job details section can be edited to allow the operator to enter information related to dates and quantities.

In a more technical perspective, this view renders a model, built in the controller, that contains an instance of a purchase order and configuration record related to the job, and also an instance of the job itself to be inserted in the system. The job instance is initialized with some predefined fields such as order number, configuration number, part number and part version. Other fields, like quantity and due date are also predefined during the construction of the model, but these can be edited in the view afterwards.

After the operator confirms all the information and clicks the finish button, the new job instance is sent as an argument to the service method *enqueueNewJob()*. This method receives the job to be created and also the logged in user, who will be linked to the new job for quality and tracking purposes. Here, the new job instance is also linked to the customer record chosen in the booking in page. Before saving the job, the appropriate status is set on the related purchase order and on the job record. Finally, it is also inserted in the system a record in the certificates table to track the various steps through which the job will go to.



Dashboard

Part Information

Description	Part Number	Part Issue	Drawing Number	Drawing Issue
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

Order Information

Due Date	Job ID	Customer	QTY	StatusC	Comments
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	

Technical Sheet Information

Specification Information

Specification Information

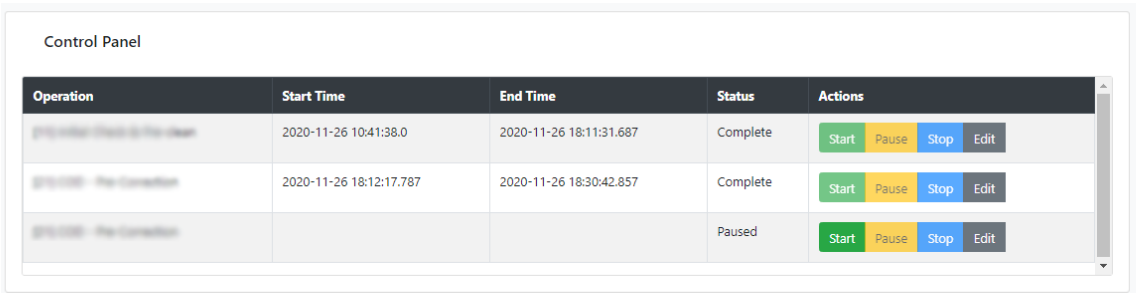
Figure 3.6: Page where the operator can confirm information related to a job and dispatch it to production

3.5.3 Task Monitoring

Also related to the job management module, another relevant component was implemented with the purpose of monitor the execution progress of each production job operation. This feature was necessary to meet the requirement that operators had on tracking the progress of each activity they work on in the shop floor. By being able to store the information related with start and end times of all operations, the transparency regarding production jobs increases, and the management level of the factory gains more information in order to make better decisions.

Figure 3.7 shows the final result of this component user interface. The operator can see in this section all the operations, for a specific job, that have been completed or are to be processed. The displayed information is retrieved from an operations table in the database, where a single row corresponds to an activity for a unique job. On each table entry, in the system, operators can access the start, pause, stop and edit actions. These actions are available according to the current status of the operation. If the operation has not yet started, the operator can press the start button and then, the start time information is stored in the database along with a status transition. When the operation is completed on the shop floor, the operator can press the stop button and the end time is recorded in the database. For this action, the system also validates if there are still operations to be done in this job and creates a new row in the table, if necessary. If there is a need to pause an operation, the operator can use the pause button. In this specific case, an end time is recorded, but a new row is created for the same operation, with the status set to paused. In this way it is possible to record all the times that this task was paused and restarted.

It is also important to detail that each operation status update is also reflected on the parent job record. This means that in the parent job record is recorded the last operation that suffered an update, the new state and the timestamp when this action occurred. By providing this information directly in the production job, it becomes easier to group all information in a single dashboard or table.



Control Panel

Operation	Start Time	End Time	Status	Actions
[redacted]	2020-11-26 10:41:38.0	2020-11-26 18:11:31.687	Complete	Start Pause Stop Edit
[redacted]	2020-11-26 18:12:17.787	2020-11-26 18:30:42.857	Complete	Start Pause Stop Edit
[redacted]			Paused	Start Pause Stop Edit

Figure 3.7: Page component that allows operators to record the progress of job operations.

3.6 Application Deployment

Delivering the proposed web application, making it accessible on the facility for the operator, proved to be quite an easy process thanks to Spring. Spring web applications are packaged into jar or war files and can include an embedded servlet container, making it possible to execute it as a standalone application. However, it was necessary to take a few steps before deploying and running the proposed web application on the company facilities' server.

The purpose of the first step in the list was to configure the authentication database in a PostgreSQL server, installed in the same server as the web application. So, it was necessary to create a user in PostgreSQL and a database where the user previously referenced was the owner. It is in this database that the Spring application creates all authentication tables on the first time it runs, using the PostgreSQL user mentioned in the application properties file.

After configuring the authentication database and verifying that Microsoft SQL database (which contains production data) is also configured, the next step was to edit the application properties file in order to insert the data source path for both databases, in the form of a jdbc url, and the user credentials to be used in each database operation. In this file, more generic application settings have also been added, such as the server port where the application will be available.

The decision to package the application as a war file, to be used in the deployment process, was taken as a preference for jar files because war files can also be deployed into traditional servlet containers, making the solution more versatile during delivery. After moving the war file into the facility server the application is executed using the following command in the terminal: *java -jar <fileName>.war –spring.config.location=file:"<path to file.properties>"*. This command makes the web application online and accessible from the designated port in the server.

Finally, the last step performed before allowing operators to use the web application, was to insert initial data, related with users and permissions, into the authentication database. It is necessary to have at least one user created in the system, with user management permission, to be able to login a first time and create the remainder of the users. After completing all configuration steps, the deployment process is complete and the application can be accessed from all the machines connected to the same internal network as the server.

4 System Testing and Evaluation

In this chapter, the results of several tests performed on the system are discussed to understand if the proposed solution works as expected and meets the requirements defined at the beginning of this dissertation. This evaluation process is divided into three types of tests, which include unit testing, functional testing and performance testing. In each one of the experiments the results are discussed and compared with the expected behavior.

In the software development life cycle, after completing the implementation of any module or functionality, a series of tests are carried out to assess the quality of the solution. This happens through the execution of pre-designed test cases, with fictitious data, where the response of the system is compared with the expected system behavior. Although with testing it is possible to reveal the presence of errors or defects in the system, it is not possible to prove their absence, as it is never feasible to test the system with all possible input-output combinations, as they are unlimited.

The solution implemented in the context of this dissertation was subject to evaluation in three different testing categories. Development testing, for matters of unit testing, where isolated system components were validated, user testing, to retrieve feedback from potential system users in work environment, and release testing, where system performance was analyzed when subject to high demand. In the next sections, for each one of these testing experiments, it is demonstrated the methodology followed and the evaluation of the results obtained.

4.1 Unit Testing

Unit testing is a type of development test and is normally used by the programming team, internally, to find logical problems and improve the reliability of the system implementation. The main characteristic of this testing process is the fact that all tests are focused on individual units of the system. Which means that for components like classes, functions or methods, isolated test cases are designed and executed to ensure that the software functionality is working as intended [SBM13].

In the proposed solution, unit tests were performed using the Spring Boot Starter Test dependency. Above all, this dependency, added to the project at an early stage, includes JUnit which is a test automation framework where it is possible to write and execute repeatable unit tests [JUn20]. The structure used for all unit tests implemented followed a division into three layers: controller, service and data. With this approach, it became easier to separate each implementation and to cover all layers of the MVC web application. The test cases developed are also designed to cover two experiments, the first of which aiming to replicate normal system operations and the second one to replicate cases where problems normally arise.

Listing 4.1 shows an example of a simple test case implementation for the controller layer. This test method validates the response of accessing the endpoint */orders/viewAll*, responsible for rendering a model with all purchase orders in the system, in pages of ten. The test is successful if the correct JSP is returned and if the information in the 'orders' attribute is the same as the one used to simulate the result of calling the service method *getAllOrders()*. In each test class, is also necessary to setup some configurations, such as an instance of the MockMvc class, to be able to execute test HTTP requests.

```
@Test
public void testViewAll() throws Exception{
    Pageable pageRequest = PageRequest.of(0, 10, Sort.by("dateLogged").
        descending());

    List<TPurchaseOrders> orders = new ArrayList<>();
    orders.add(new TPurchaseOrders("1", "1", "P1", "R1", null, "New", "1"));
    orders.add(new TPurchaseOrders("2", "2", "P2", "R2", null, "Closed", "2"));

    when(tPurchaseOrdersService.getAllOrders(pageRequest)).thenReturn(orders);

    mockMvc.perform(get("/orders/viewAll").param("pageId", "1"))
        .andExpect(status().isOk())
        .andExpect(view().name("OrdersListviewAll"))
        .andExpect(model().attribute("orders", hasSize(2)))
        .andExpect(model().attribute("orders", hasItem(
            allOf(hasProperty("poNumber", is("1")))
        )))
        .andExpect(model().attribute("orders", hasItem(
            allOf(hasProperty("poNumber", is("2")))
        )));
    }
```

Listing 4.1: Implementation of a simple unit test case in controller layer

4.2 Usability Testing

Knowing how real users will use the system is a crucial aspect of implementing a successful software solution. Through usability tests, it is possible to have target system users carry out pre-designed test case scenarios directly in the application, simulating a regular work environment. Usually in these tests the aim is to simulate everyday tasks with artificial data, similar to real data. By replicating these day-to-day use case scenarios, it becomes possible to not only detect potential system problems, but also to understand whether all system requirements are met and whether any changes may be necessary to satisfy additional needs of end-users.

With the assistance of users from the partner company, it was possible to test some use case scenarios in an initial release of the system implementation. These scenarios were made available to several test users and, after their execution, a report was generated with the results of all tests and feedbacks provided by these users. With this data, it was possible to compare the results with the expected system behavior and draw conclusions regarding the fulfillment of system requirements and user experience for the web application. Table 4.1, shows the results obtained for a list of four test case scenarios performed in an early system release.

#	Scenario Description	Expected Result	Comments
1	A user, with user management permission, logs in to the system and creates a new user. The new user then logs in to the system.	New user created in the system, with the possibility to login with his user credentials.	Test successful. Good user experience reported on the user creation form.
2	A user, with order management permission, connected to the system is able to create a single new purchase order.	New order is created with the parameters entered by the user	Test successful.
3	A user, with order management permission, connected to the system is able to create multiple purchase orders by importing an Excel file containing the order details.	Multiple orders are entered into the system with the parameters specified in the Excel file.	The test was successful, however, a user witnessed abnormal behavior due to an incorrectly configured Excel file, which was not rejected by the system.
4	A user, with jobs management permission, connected to the system is able to select a new order and dispatch it to the shop floor, creating a new production job for it.	A new job is created and the related purchase order is changed to active status.	The test was successful, but a user had trouble figuring out which purchase orders, present in the initial book in page, had enough resources available to be worked on. This information was only displayed after selecting each order for book in.

Table 4.1: Usability tests results

The results of the executed usability tests proved that the system met the initial requirements, since the users were able to complete the assigned tasks. Although some minor flaws have been identified, users still reported a good user experience when interacting with the application. Therefore, even with the presence of room for improvement, the system can be considered ready for use in the facility production environment.

4.3 Performance Testing

The last testing experiment, performed on the proposed system, is called performance testing and can be included as part of release testing phase. Usually, they are executed on a final release version of the system with the aim of measuring some performance metrics, like response times and throughput rates, when there is an increase of workload in the application. By gathering such information, it is possible to verify if the system will maintain the desired behavior over time, and to inform end users whether the solution, to be delivered, is in accordance with pre-defined parameters in terms of efficiency, robustness or reliability.

First, in order to access the initial state of the solution, a set of generic test cases were defined and executed on application, running in the facilities server. The results of such tests, shown in table 4.2, were very positive and revealed that the system could endure day-to-day activities efficiently with the available resources.

Test case scenario	**Test result**
A user logs into the application and after one hour of inactivity, it tries to perform a task and the system asks for login again.	A new login action was requested to the user.
A user tries to login in the system after 48 hours of inactivity, with the application always running in the server.	The system responded and the user was able to login.
Two different users try to login at the same time from different machines.	Both users were able to login in the application.
A user logged in the system loses network connection for ten seconds. After regaining connection, the session should not be lost.	User was able to continue using the system after regaining network connectivity with same session.

Table 4.2: List of initial performance tests and results.

The next phase of performance tests carried out included the evaluating of the system's response when subjected to an above normal demand. These tests, often called stress or load tests, were designed to measure the system's response in such situations and to detect any anomalies that could be caused by the lack of available resources to process requests. To support the execution of these types of test case scenarios, the JMeter tool was used, which is an open source Java application capable of simulating a heavy workload on a system while evaluating its response [Fun20]. With this tool, a test plan was created that included the creation of all HTTP requests needed to perform an end-to-end test case scenario starting with a user login and ending with the creation of a new purchase order. The test plan was then executed several times for a large number of users simultaneously. Finally, the time and efficiency results for the login and order creation task were calculated, giving the result displayed in the table 4.3 and table 4.4.

Nr Users	**Min(ms)**	**Max(ms)**	**Avg(ms)**	**Req/sec**
1	341	350	342	2,81
10	350	523	464	17,47
50	415	2774	1243	21,01
100	456	4309	1954	21,06

Table 4.3: Performance tests for login action

It is important to note that the gathered results, for the test case scenarios described above, are average values of execution of each request per user, simultaneously, a total of ten times. By repeating each test case ten times, with the JMeter tool, it was possible to obtain more consistent results and avoid unique

Nr Users	**Min(ms)**	**Max(ms)**	**Avg(ms)**	**Req/sec**
1	5	9	6	3,11
10	5	49	11	18,60
50	6	1620	552	22,00
100	5	3784	1525	21,78

Table 4.4: Performance tests for new purchase order creation action

case situations.

The results themselves are very satisfactory, since it was not detected any operational error and all requests were completed successfully. Regarding speed values, for both actions, it was recorded the minimum and maximum time for a request, as well as the average values. The longest time recorded was 4.3 seconds for a login request, when there where 100 users trying to use the application at the same time, and although it seems a bit long, it happened in a very rare situation, since the target number of users is well below value of 100. Lastly, the number of requests handled per second was also recorded and with this information it was possible to conclude that when the number of users reaches 50, the application performance stagnates, because the number of requests per second remains between 21 and 22.

5 Conclusion

This chapter presents a final review of all the work carried out in the scope of this dissertation together with a conclusion regarding the obtained results. Section 5.2 concludes this document with an outline of the work to be further developed, in order to extend or improve the proposed system.

The work developed throughout this dissertation sought to describe the process of implementing an information system for resource and process management in an industrial facility. All the requirements of this system were discussed in partnership with a local industrial aeronautical facility, which served as the final destination and real use environment for the built solution. The need for this system can be explained by the growing importance on tracking production processes and inventory on the factory floor, to improve response times and consequently the efficiency of production processes. The relevance of developing the proposed solution is also explained by the great effort that SMEs need, to adopt the available systems on the market, since most of them have a high degree of complexity and cannot be adapted to the factory business model and specific needs.

5.1 Work Overview

The first stage of this work was fundamental to understand the circumstances in which the system would be implemented and the functional patterns that the system should follow. The research conducted on this stage of the work made it possible to identify the different types of information systems usually present in an industrial enterprise and the necessary connections between them. The survey on shop floor level systems led to the concept of MES. The proposed solution fits this type of system, therefore, it was possible, by looking into different MES models, created by MESA, to describe some guidelines and features that should be implemented in the new system, to support regular manufacturing activities. This stage of the work was concluded with the analysis of some existing solutions, and the outcome of this analysis revealed common patterns in the implementation of both systems. Features such as modularity or the ability to connect to external systems, like ERP, are often used in such systems and must be taken into account when building a new one from scratch.

Regarding the system implementation phase, although not all system requirements defined during the system design have been met, the main objective of creating a first and fully working release version of the system has been achieved. A functional prototype of the proposed web application, containing a variety of features, was delivered to the partner's facility at the time of writing. This prototype was connected to two databases in the facility server (one for business data linked to corporate applications and the other for authentication and authorization purposes) and can now be used by any registered operator directly from the facility. The authentication and authorization aspects, stated in the initial requirements, were also successfully implemented in the system prototype, and gave the possibility of user management actions and the possibility of granting access permissions based on the operator's role in the factory.

The prototype implementation was completed for this phase with the inclusion of several operations from the order management and job management modules. With these features, operators can perform a complete chain of actions, starting with the creation of a purchase order in the system and continuing with the dispatching of that order to production, only if there are enough resources in the facility. With this development, one of the main requirements for the solution was met, since users gained the ability to track their work efficiently and automatically validate whether a job can be performed or not, according to defined business criteria.

Finally, this work was concluded with a system assessment, performed in different circumstances. Several tests were carried out to evaluate the system prototype quality in relation to the robustness of the implemented code and the functional behavior of the application. The results were very positive, since no significant defects were raised when testing day-to-day activities. Spring framework also allowed to build a solution that achieved good performance results when subjected to an above-normal demand.

5.2 Future Work

Despite having delivered a first version of the system with this dissertation, the work regarding this system implementation does not end here. There are still some requirements mentioned in this dissertation that have not yet been implemented. New requirements can also be created in the future to expand the current system or to improve its performance.

Even though job module requirement was deeply developed on this project, there is still room for improvement on this functionality. The option for editing a job is still missing and the operators are unable to change the operations related to production jobs. This feature is very useful in a facility where the requirements for manufacturing can change in the middle of the production workflow. This implementation can be similar to the one developed on purchase orders, but there will be a need to introduce additional logic to change the related operations, as this is a different entity then the job entity. Nevertheless, before implementing such functionality, the partners facility needs to be contacted, as some validations may be imposed to restrict some changes on these actions.

The further development of an analytic component should also be top priority for a next release. In this component, the information retrieved from the database should be displayed in a friendly interface, through charts or graphs. The implementation of such feature will provide to the users relevant information to assess the facility performance. One example is handling the information retrieved from the process of job operation tracking to understand how the facility is performing in time metrics, since it can be possible to display this information in a way that facilitates the read by the operators. This component can be also important for quality aspects, as it can be developed to display jobs with defects or machines with issues, improving the capability of quick problem detection and response.

Also as future work, is intended to be explored connections with other external systems, like ERP systems. These connections, whether at a higher or lower level of the enterprise, will bring more advantages, as the system will have more data to process and analyze. This feature can be used alongside the existing databases that feed the system, or used as a replacement for these databases. This last approach would need deeper developments as a refactoring in the model component of the system would be needed.

It should always be taken into consideration that any increment made to the system, such as those mentioned above, must be followed by a testing phase that should validate not only the new developments, but also evaluate the quality of the complete solution. To do so, a series of end-to-end tests should be planned to verify that the system didn't lose any property or suffered some regression.

Ultimately, this project proved to be a pleasant challenge in the field of software engineering. Throughout the work, several steps regarding the study, design, development, deployment, testing and monitoring of the solution have been accomplished. These steps are essential to build a stable software solution and in this project revealed to be crucial to secure the quality of the delivered solution. The design of the proposed solution also allowed to explore and apply interoperability with other systems, through connections to shared data sources and networks.

Bibliography

[AP18] Emrah Arica and D. J. Powell. Status and future of manufacturing execution systems. *IEEE International Conference on Industrial Engineering and Engineering Management*, 2017-Decem:2000–2004, 2018.

[Cro11] Bill Crowley. Manufacturing execution systems vs. ERP/MRP.pdf. *Global SMT & Packaging*, 11(9):10–16, 2011.

[D'm17] Bruno Joseph D'mello. *Web Development with MongoDB and Node - Third Edition*. Packt Publishing, 3 edition, 2017.

[DZo19] DZone. Web development comparison: Spring boot vs. express.js. `https://dzone.com/articles/web-development-comparison-springboot-vs-expressjs`, 2019. Accessed: 2020-08-06.

[Fou20] Apache Software Foundation. Poi-hssf and poi-xssf/sxssf - java api to access microsoft excel format files. `https://poi.apache.org/components/spreadsheet/index.html`, 2020. Accessed: 2020-10-18.

[Fun20] Apache Software Fundation. Apache jmeter - apache jmeter. `https://jmeter.apache.org/`, 2020. Accessed: 2020-11-03.

[Git20] Github. qcadoo/mes: qcadoo mes - friendly web manufacturing software. `https://github.com/qcadoo/mes`, 2020. Accessed: 2020-07-19.

[Gro15] Mikell P Groover. *Fundamentals of Modern Manufacturing*. Wiley, 5 edition, 2015.

[HLYZ13] Biqing Huang, Chenghai Li, Chao Yin, and Xinpei Zhao. Cloud manufacturing service platform for small- and medium-sized enterprises. *International Journal of Advanced Manufacturing Technology*, 65(9-12):1261–1272, 2013.

[HSHA14] Petri Helo, Mikko Suorsa, Yuqiuge Hao, and Pornthep Anussornnitisarn. Toward a cloud-based manufacturing execution system for distributed manufacturing. *Computers in Industry*, 65(4):646–656, 2014.

[Int97] MESA International. Mes explained: A high level vision. 1997.

[Int20] MESA International. Mesa international - mesa model. http://www.mesa.org/en/modelstrategicinitiatives/MESAModel.asp, 2020. Accessed: 2020-07-14.

[ISA20] ISA. About isa - the home for automation. https://www.isa.org/about-isa/, 2020. Accessed: 2020-07-14.

[JUn20] JUnit. Junit - about. https://junit.org/junit4/, 2020. Accessed: 2020-10-31.

[JZLZ15] Pingyu Jiang, Chaoyang Zhang, Jiewu Leng, and Jinwei Zhang. Implementing a WebAPP-based software framework for manufacturing execution systems. *IFAC-PapersOnLine*, 28(3):388–393, 2015.

[KD18] Jürgen Kletti and Rainer Deisenroth. *MES Compendium: Perfect MES Solutions based on HYDRA*. Springer, 2018.

[Kle07] Jürgen Kletti. *Manufacturing Execution System - MES*. Springer, 2007.

[KSB16] Shameer Kunjumohamed, Hamidreza Sattari, and Alex Bretet. *Spring MVC: Designing Real-World Web Applications*. Packt Publishing, 1 edition, 2016.

[Kus18] Andrew Kusiak. Smart manufacturing. *International Journal of Production Research*, 56(1-2):508–517, 2018.

[MFT09] Heiko Meyer, Franz Fuchs, and Klaus Thiel. *MES - Optimal design, plannig and deployment*. McGraw-Hill, 2009.

[MPD20] MPDV. Mpdv: Increase productivity with the modular mes hydra by mpdv. https://www.mpdv.com/en/products-solutions/mes-hydra/, 2020. Accessed: 2020-07-19.

[Nod20] Node.js. About | node.js. https://nodejs.org/en/about/, 2020. Accessed: 2020-08-09.

[Par20] Eugen Paraschiv. Password encoding with spring | baeldung. https://www.baeldung.com/spring-security-registration-password-encoding-bcrypt, 2020. Accessed: 2020-10-02.

[Poc20] Michał Poczwardowski. Pros and cons of django - when to use python framework (updated) | netguru blog on python. https://www.netguru.com/blog/pros-and-cons-of-django, 2020. Accessed: 2020-08-18.

[Qca20] Qcadoo. qcadoo mes - functions overview. https://www.qcadoo.com/functions-overview/, 2020. Accessed: 2020-07-19.

[San20] Sandhya Sandy. What is spring security? how does it work? | by sandhya sandy | javarevisited | med. https://medium.com/javarevisited/what-is-spring-security-how-does-it-work-9d561fe3f92a, 2020. Accessed: 2020-09-30.

[SBM13] Corey Sandler, Tom Badgett, and Glenford J Myers. *The art of software testing*. Wiley, 3 edition, 2013.

[SBZ08] Yoshiaki Shimizu, Rafael Batres, and Zhong Zhang. *Frontiers in Computing Technologies for Manufacturing Applications*, pages 221–250. Springer-Verlag London Limited, 2008.

[Sch16] Klaus Schwab. The Fourth Industrial Revolution: what it means and how to respond. *World Economic Forum*, 2016.

[SMMM09] Baljinder Singh, Jason Matthews, Glen Mullineux, and Tony Medland. Product development in manufacturing smes: Current state, challenges and relevant supportive techniques. In *DS 58-6: Proceedings of ICED 09, the 17th International Conference on Engineering Design, Vol. 6, Design Methods and Tools (pt. 2), Palo Alto, CA, USA, 24.-27.08. 2009*, 2009.

[Spr20a] Spring. 1.2 modules. `https://docs.spring.io/spring/docs/3.0.0.M4/reference/html/ch01s02.html`, 2020. Accessed: 2020-08-18.

[Spr20b] Spring. 18. view technologies. `https://docs.spring.io/spring-framework/docs/3.2.x/spring-framework-reference/html/view.html#view-jsp-formtaglib-formtag`, 2020. Accessed: 2020-10-18.

[Spr20c] Spring. Spring | why spring? `https://spring.io/why-spring`, 2020. Accessed: 2020-08-18.

[Sri20] Shubhra Srivastava. Pagination and sorting using spring data jpa | baeldung. `https://www.baeldung.com/spring-data-jpa-pagination-sorting`, 2020. Accessed: 2020-10-18.

[TZV$^+$11] F. Tao, L. Zhang, V. C. Venkatesh, Y. Luo, and Y. Cheng. Cloud manufacturing: A computing and service-oriented manufacturing model. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 225(10):1969–1976, 2011.

[Vin18] William S. Vincent. *Django for Beginners: Build websites with Python and Django*. WelcomeToCode, 2018.

[vsC20] vsChart. Spring vs. django vs. node.js comparison. `http://vschart.com/compare/spring-framework/vs/django-framework/vs/node-js`, 2020. Accessed: 2020-08-17.