UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Instituto de Ciência e Tecnologia de Sorocaba
Campus de São João da boa vista

José Zapata Neto

Development of a Business Interface based on RAMI 4.0 for Flexible Manufacturing in Industry 4.0

Sorocaba
2022

José Zapata Neto

Development of a Business Interface based on RAMI 4.0 for Flexible Manufacturing in Industry 4.0

Dissertação de defesa apresentada como parte dos requisitos para obtenção do título de Mestre em Engenharia Elétrica, junto ao Programa de Pós-Graduação em Engenharia Elétrica, interunidades, entre o Instituto de Ciência e Tecnologia de Sorocaba e o Campus de São João da Boa Vista da Universidade Estadual Paulista “Júlio de Mesquita Filho”.

Área de concentração: Automação
Orientador: Prof. Dr. Eduardo Paciência Godoy

Sorocaba
2022

Z35d

Zapata Neto, José
Development of a Business Interface based on RAMI 4.0 for Flexible Manufacturing in Industry 4.0 / José Zapata Neto. -- São João da Boa Vista, 2022
108 p. : il., tabs., fotos

Dissertação (mestrado) - Universidade Estadual Paulista (Unesp), Faculdade de Engenharia, São João da Boa Vista
Orientador: Eduardo Paciencia Godoy

1. Automação industrial. 2. Internet das coisas. 3. Interface de programação de aplicativos (software de computador). 4. Automação. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca da Faculdade de Engenharia, São João da Boa Vista. Dados fornecidos pelo autor(a).

Essa ficha não pode ser modificada.

UNIVERSIDADE ESTADUAL PAULISTA

Câmpus de Sorocaba

CERTIFICADO DE APROVAÇÃO

TÍTULO DA DISSERTAÇÃO: Development of a Business Interface based on RAMI 4.0 for Flexible Manufacturing in Industry 4.0

AUTOR: JOSÉ ZAPATA NETO
ORIENTADOR: EDUARDO PACIÊNCIA GODOY

Aprovado como parte das exigências para obtenção do Título de Mestre em Engenharia Elétrica, área: Automação pela Comissão Examinadora:

Prof. Dr. EDUARDO PACIÊNCIA GODOY (Participaçao Presencial)
Departamento de Engenharia de Controle e Automacao / Instituto de Ciência e Tecnologia - UNESP - Câmpus de Sorocaba

Prof. Dr. MARCIO ALEXANDRE MARQUES (Participaçao Presencial)
Departamento de Engenharia de Controle e Automação / Unesp - ICT Sorocaba

Prof. Dr. SERGIO SHIMURA (Participaçao Presencial)
Departamento de Indústria / Instituto Federal de Educação, Ciência e Tecnologia de São Paulo - Campus Soracaba

Sorocaba, 27 de outubro de 2022

THANK YOU TO

To God for strength and support at any time.

To Professor Eduardo Paciência Godoy for his guidance and dedication during all stages. To my colleague Joel Ravelli for the exchange of experiences throughout the conducting this research.

To my friend Yara Marinho for all the dedication to help me get through this challenge. To my family and my husband.

And to Unesp, for providing this amazing experience to build and share knowledge with our fellow.

ABSTRACT

The Industry 4.0, together with the Industrial Internet of Things (IIoT), enable to transfer information, analysis, and data over the internet, improving business productivity through rapid changes in production scope in an increasingly volatile market. In this scenario, the Reference Architecture Model for Industry 4.0 (RAMI 4.0) was defined the elements of I4.0 in a three-dimensional layer model. One of the major challenges for adopting RAMI 4.0 is the development of solutions that support the functionality of each layer and the necessary interactions among. Despite of many studies developing RAMI 4.0 compatible solutions, there are very few studies focusing on the top layers, such as Functional and Business. This research developed a business interface for flexible manufacturing in I4.0 focusing on the Functional and Business layers of RAMI 4.0. The interface uses concepts of Service-Oriented Architecture and Application Programming Interfaces (APIs) and provides a fully API-based interface to easily and scalable exchange information from the factory to clients. The Order Orchestrator is the main module for interaction to both sides, handling production in the factory, such as orders and items production updates from the stations, also, from the clients-side, using the API Gateway, the system is able to insert new orders, delete orders already in production and even modify an order that is currently in production to become a different final good, all of that receiving a response in matter of seconds. The API Gateway based in RESTful APIs is scalable and easy to integrate. The client can manage orders in its backend integration as required, using its own Enterprise Resource Planning (ERP) functions or via standard software connected to computer, gives a major advantage to the factory's clients as they can quickly respond to market changes, orchestrating changes in their production orders. The business interface was tested and validated in an I4.0 factory-based simulated scenario of clothing production, managing different production orders and changes. This research shows how a combination of the Order Orchestrator in the Functional layer and API Gateway in the Business layer brings a Product as a Service perspective for flexible production and manufacturing, opening space to improve the way of factories and production systems interact with clients.

Keywords: Industrial Automation, Internet of Things, Application Programming Interface (Software Computing), Automation.

RESUMO

A Indústria 4.0 (I4.0), juntamente com a Internet das Coisas Industrial (IIoT), possibilitou a transferência de informações, análises e dados pela internet, melhorando a produtividade dos negócios por meio de rápidas mudanças no escopo de produção em um mercado cada vez mais volátil. Neste cenário, no Modelo de Arquitetura de Referência para Indústria 4.0 (RAMI 4.0) foram definidos os elementos do I4.0 em um modelo de camada tridimensional. Apesar de muitos estudos desenvolvendo soluções compatíveis com RAMI 4.0, são poucos com foco nas camadas superiores, como Funcional e Negócios. Esta pesquisa desenvolveu uma interface de negócios para manufatura flexível em I4.0 com foco nas camadas superiores do RAMI 4.0. A interface usa conceitos de Arquitetura Orientada a Serviços (SOA) e Interfaces de Programação de Aplicativos (APIs) e fornece uma interface totalmente baseada em API para troca de informações fácil e escalável da fábrica para os clientes. O Orquestrador de Pedidos é o principal módulo de interação para ambos os lados, tratando da produção na fábrica, como atualizações de produção de pedidos e itens das estações, também, do lado do cliente, usando o API Gateway, o sistema é capaz de inserir novos pedidos, excluir pedidos já em produção e até modificar um pedido que está atualmente em produção para se tornar um bem final diferente, tudo isso recebendo uma resposta em questão de segundos. O API Gateway baseado em APIs RESTful é escalável e fácil de integrar. O cliente pode gerenciar pedidos em sua integração de back-end conforme necessário, usando suas próprias funções de Enterprise Resource Planning (ERP) ou via software padrão conectado ao computador, dando uma grande vantagem aos clientes da fábrica, pois podem responder rapidamente às mudanças do mercado, orquestrando mudanças no suas ordens de produção. A interface de negócios foi testada e validada em um cenário simulado baseado em fábrica I4.0 de produção de roupas, gerenciando diferentes ordens de produção e alterações. Esta pesquisa mostra como a combinação do Order Orchestrator na camada Funcional e API Gateway na camada Business traz uma perspectiva de Produto como Serviço para produção e manufatura flexíveis, abrindo espaço para evolução de fábricas e sistemas de produção com os clientes.

Palavras-chave: Automação industrial, Internet das coisas, Interface de programação de aplicativos (software de computador), Automação.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATION AND ACRONYMS

API	Application Programming Interface
AAS	Asset Administration Shell
AMQP	Advanced Message Queuing Protocol
BMM	Business Motivation Model
BPMN	Business Process Modeling and Notation
B&Q	Batch and Queue
CPS	Cyber-Physical Systems
CPPS	Cyber-physical Production Systems
COM	Component Object Model
CT	Cycle Time
CRM	Customer Relationship Manager
DB	Data Base
DPS	Dispersed Productive Systems
ERP	Enterprise Resource Planning
EPC	Electronic Product Code
FDI	Field Device Integration
GW	Gateway
ICT	Information and Communication Technology
IETF	Internet Engineering Task Force
I4.0	Industry 4.0
IOT	Internet of Things
IIOT	Industrial Internet of Things
IOS	Internet of Services
IT	Information Technology

ISA	**I**nternational Society of Automation
JSON	JavaScript Object Notation
HW	Hardware
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
KPI	Key Success Indicator
MaaS	Manufacturing as a Service
MBSE	Model-based System Engineering
MES	Manufacturing Execution Systems
MQTT	Message Queue Telemetry Transport
MTBF	Mean Time Between Failure
OPC UA	Open Protocol Communications Unified Architecture
OPC AE	Open Protocol Communications Alarms & Events
OPC HDA	Open Protocol Communications Historical Data Access
OPC DA	Open Protocol Communications Data Acquisition
OMG	Object Management Group
PAAS	Production as a Service
PN	Part Number
RAMI 4.0	Reference Architecture Model for Industry 4.0
REST	Representational State Transfer
RP	Relying Parties
SCADA	Supervisory Control and Data Acquisition
SW	Software
SDK	Software Development Kit
SEM	Structural Equation Modeling
SOA	Service-Oriented Architecture

SOC	Service-Oriented Computing
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
URI	Universal Resource Identifier
WS	Web Service

CONTENTS

1. INTRODUCTION

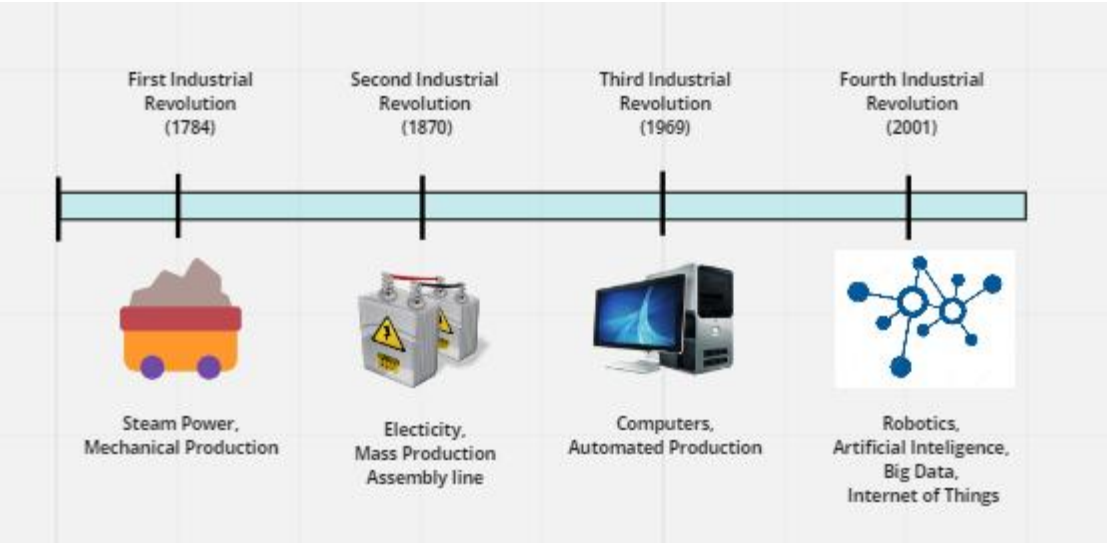
This section was created to give an introduction in the topic of Industry 4.0 and manufacturing as a service, to give the reader an opportunity to get more comfortable in the topic relevance and start to share this project main goals.

1.1. Relevance

Also known as “Smart Factory”, the term Industry 4.0 has been coined at the 2011 Hannover Fair. The number 4.0 refers to be the fourth Industrial Revolution to come. According to Rüttimann and Stock (2016), the first Industrial Revolution is largely considered to be the steam machine which made the steam power exploitable and opened the industry age. The second Industrial Revolution remotes to the discovery and application of electricity allowing, for example, automobile mass production. The third Industrial Revolution is related to the possibility of data processing for computer integrated manufacturing (CIM), leading to the present era of information technology (RÜTTIMANN e STÖCKLI, 2016).

In Figure 1 it is possible to see all four Industrial Revolutions, and the main advances brought by them. The development of new digital technologies allowed the creation of new production methods in global industries based on automation, robotics, artificial intelligence, IoT, and data intelligence, among other innovations (BIGHETI e GODOY, 2020).

Figure 1. General vision of Industrial Revolutions, and it’s important milestones.



Source: Own Authorship, adapted from (MELO, 2019).

The coordinated use of these technologies in the industrial context, making the business competitive, optimizing the efficiency of the production chain, adding value to the product, rationalizing the use of resources and customizing technological solutions is called Industry 4.0.

According to Suri et al. (2017), Industry 4.0 is one of the most prominent initiatives towards the vision of smart manufacturing to foster efficiency and synergy among suppliers, producers, and customers. The availability of a wide range of data and information in the context of Cyber-physical Production Systems (CPPS) provides a huge possibility to create novel business opportunities. However, there is an evident need to have efficient communication and clear visualization of the new strategies and the underlying operational process among all the stakeholders involved in the manufacturing value chain (SURI, CADAVID, *et al.*, 2017).

The majority of technologies necessary for the implementation of I4.0 already exists today, although in different stages of maturation. Zuehlke (2010) quotes the disruptive character that I4.0 brings as the result of the articulation and convergence of these technologies. First of all, the expansion of wireless networks to make communication and data acquisition flexible, combined with IPV6 protocol to expand the IP connection points; RFID for tracking products; cyber-physical systems (CPS); systems and services virtualization; the use of Industrial Internet of Things and Cloud Computing aiming at the integration and interoperable connection of all equipment on networks and the exchange of information in the cloud; and finally, Big Data and Artificial Intelligence for decision making (ZUEHLKE, 2010).

Several works have focused on studying the structure of RAMI 4.0. The big challenge for adopting RAMI 4.0 is found in the development of solutions, such as, equipment, systems, and standards, that support the functionalities of each layer and the interactions required between the elements of those layers (CONTRENAS, GARCIA e PASTRANA, 2017). As I4.0 currently represents a concept – not specifying, for example, communication technologies, integration tools and programming standards to be used – most works should look for different technologies that can be integrated and provide the functionality required by new applications. Kaggermann (2013) pointed out that solutions should be developed and made available to the market as soon as possible, considering their strategic implementation and the

adaptation of existing technologies and basic experiences to manufacturing requirements (KAGGERMANN, WAHLSTER e HELBIG, 2013).

On top the that, with the increasing demand for product customization, manufacturing systems are becoming more flexible, customizable, and intelligent. Therefore, product factories with small-batch custom manufacturing needs often struggle with the task of underused production lines and sufficient system flexibility to integrate new products into the production line. Developing a method for automatically matching product fabrication with clients will enable faster production and more efficient use of the available resources in the manufacturing industry.

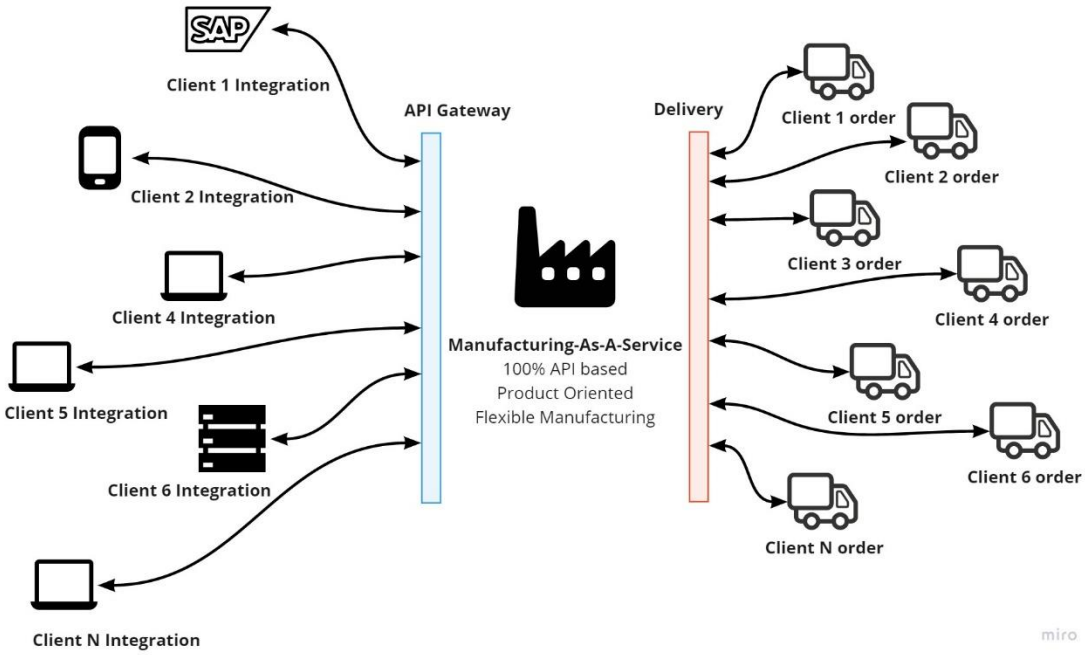
Delsing (2017) presents the concept of an on-premises automation cloud, in which automation components and devices are provided as services, allowing full interoperability for industrial applications. In this project, the entire integration and communication infrastructure was carried out through a Service Oriented Architecture (SOA), developed and called the Arrowhead Framework (DELSING, 2017). Leitão (2016) presents a review of some European Union-funded initiatives on industrial SOA applications. The developed SOA architectures are discussed and case studies in which these architectures have been validated are presented later in this research (LEITÃO, COLOMBO e KARNOUSKOS, 2016).

The concept of services allows any system to become clients or servers, depending on the need for the process or application, and for interaction, based on the exchange of messages, between any elements of the hierarchical levels of the industrial process. A service-oriented architecture consists of a collection of small autonomous services, where each service is independent and implements a single functionality.

Balta (2018), presented a Production as a Service (PaaS) framework and introduced the key structures and functionalities for PaaS realization. The PaaS framework enabled effective utilization of manufacturing resources for rapid realization of custom manufacturing needs of product developers (BALKA, LIN, *et al.*, 2018). The paper shows how precedence and batch splitting constraints efficiently evaluate optimal manufacturing solutions. The macro concept of Manufacturing as a Service (MaaS) can be seen in Figure 2. According to Annunziata (2019), extracted from *Forbes* Magazine, MaaS platforms bring three major benefits. First, they enable a higher degree of utilization of installed production capacity. Machines that would

otherwise sit idle can be used to meet the new demand for parts from other companies, raising the return on investment; and the companies that buy the parts can, in turn, increase their production thanks to an improved and faster access to essential components. Second, the platforms improve transparency and price discovery by providing instant quotes generated by AI engines drawing on a large set of data on which companies can produce a specific product and a specific cost (ANNUNZIATA, 2019). Price transparency is a big deal; a more transparent marketplace will foster greater competition, speed, and efficiency. Third and last, these platforms enable greater and more efficient localization. Manufacturing as a Service platforms give access to a wider range of suppliers, and for equal price and quality, the choice will gravitate towards local firms.

Figure 2. The general concept of Manufacturing as a Service.



Source: Own Authorship.

Among the publications concerning RAMI 4.0, most of them are focused on the bottom layers, and only a few addresses the Functional and Business layers. This research is part of a major research group, that consists of a full architecture proposition to engage all six layers of RAMI 4.0 showing all cycles of production system combined for a centralized production end-to-end perspective. A platform 100% API

based can bring a product-oriented perspective to a place that is client-oriented, and also, touchless connectivity to new clients and business opportunities. When the exterior is modern API based, the interior is ruled by OPC Unified Architecture and Molecular Framework, binging factory control and organization together.

This research is developed with other Master's Degree students project and the architecture developed and presented works as a unity and will be tested together end to end. This research is focused on the Functional and Business layers of RAMI 4.0, whilst the attention of (MELO, 2020), are the first three layers, Asset, Integration and Communication, and (RAVELLI, 2022) works with the Information layer.

1.2. Objectives and Contributions

This research develops a business interface for Industry 4.0 applications based on RAMI 4.0, Moleculer framework for microservices, and API Gateway, to exteriorize the control of production systems and also to enable flexible manufacturing *as a Service* in a Product-Client-oriented mass production.

1.3. Dissertation Organization

The structure of this dissertation was divided into 6 chapters considering this introduction and disregarding the Attachments, Appendix and Bibliographic References.

A brief introduction to Industry 4.0, its applications and potentiality are presented in Chapter 2. The highlights of the Service Oriented Architecture, clarifying the basic elements of the communication protocol for Industry 4.0 and RAMI 4.0, are also presented in Chapter 2.

In Chapter 3 is presented the first part of a conceptual review, addressing the main features of RAMI 4.0.

The Methods and Technologies used are shown in Chapter 4, and is described an architecture for the development of the project. It gives an overview of the technology used for the code development such as Node JS, Moleculer Framework, details of the API Gateway, and security protocol OAuth 2.0. Finally, it is given more details of the Functional and Business layers.

The development of this research and results are presented in Chapter 5. It is separated in the Functional layer, bring more perspective for the Order Orchestration, and the Business layer focusing more on the Gateway and in the API Swagger. A proof-of-concept scenario is also presented in Chapter 5 in order to test and validate the business interface developed in this project for flexible manufacturing in Industry 4.0.

And finally, in Chapter 6 is presented the remarks and conclusions drawn from the elaboration of the research.

2. LITERATURE REVIEW

In this section will be presented notions to better understand the main concepts applied in Industry 4.0 and Service Oriented Architecture.

2.1. Industry 4.0

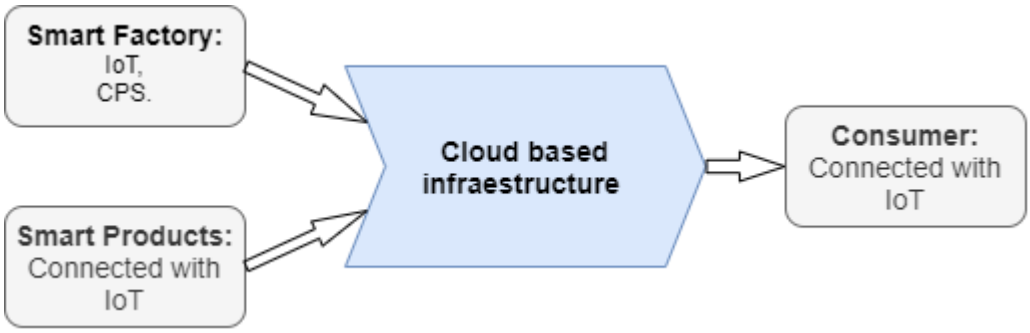
While the Toyota Production System (TPS) has shown to be the most performant manufacturing system, the Industry 4.0 initiative is still in the scoping phase with the demanding goal to become a highly integrated cyber production system (RÜTTIMANN e STÖCKLI, 2016). The word Industry 4.0 is used to refer to the “Fourth Industrial Revolution” (BASSI, 2017). The first Industrial Revolution is mostly considered to be the steam machine, which used the steam power exploitable, opening the industrial age; the Second Industrial Revolution is mostly seen as the use of electricity to create mass production, mainly in the new automotive industry; Third Industrial Revolution is generally associated to the wide use of electronics and information technology to automate the production process, also providing of data handling for computer integrated manufacturing (CIM), bringing to the present era of information technology and Operation technology integration (ZEZULKA, MARCON e VESELY, 2016).

The essence of IoT is the basic principle of I4.0 providing optimized and increasingly autonomous product manufacturing through communication between machines and products. In this way, a company creates collaborative networks across the value chain that communicate and control each other with a significant reduction in operator intervention. In this scenario, the main interaction between machines and products involves identifying the operations to be performed during the manufacturing process. An overview can be seen in Figure 3, showing the relationship between factories, products, and consumers, in a cloud-based infrastructure.

According to Zezulka et al. (2016), Industry 4.0 is used for three mutually interconnected factors:

1. Digitalization and integration of simple technical communication to complex communication and complex network;
2. Digitalization of products and services;
3. New market models.

Figure 3. Industry 4.0 is based on IoT, Big Data, Cloud, SOA, CPSs, and cloud computing to meet personalized consumer requirements.



Source: Own Authorship

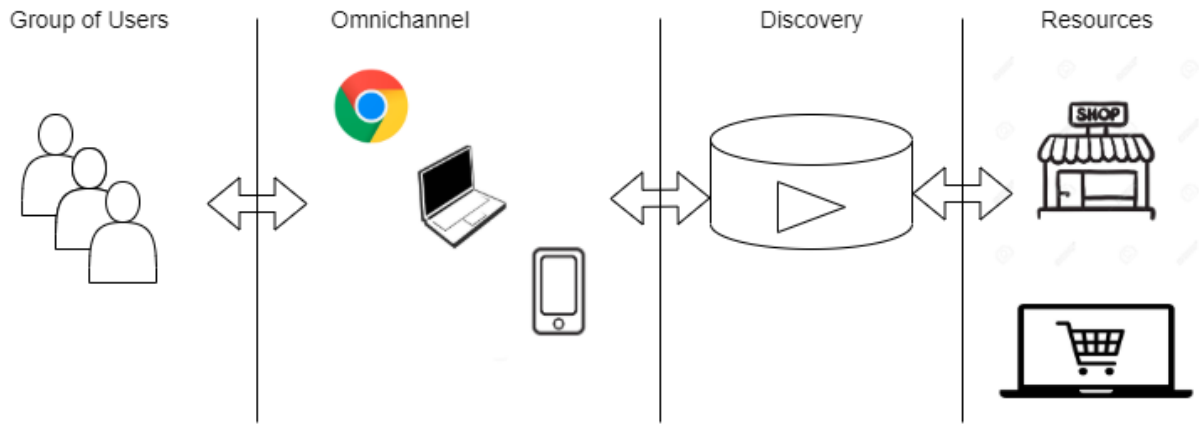
Those characteristics are a result of the integration of Industry 4.0 with technologies of the third Industrial Revolution; along with information processing components that allow management as information. Among the various technologies that are contemplated for a I4.0, there is augmented reality, cloud computing, Internet of Things (IoT), Big Data, Internet Services (IOS), and Cyber-Physical Systems (CPS) (WORKSHOP “PLATFORMS FOR CONNECTED FACTORIES OF THE FUTURE”, 2015).

Along with IIOT, the Cyber-Physical Systems (CPS) represent the integration of computer systems with physical systems. A CPS improves functionality, performance, security, management capability, adaptability and allows real-time applications by integrating communication and resources of computer storage with monitoring. A CPS is a control unit, usually one or more microcontrollers that control sensors and actuators, these being needed to interact with the physical world (TEODORA e LIVIU, 2012). This set of elements also requires a communication interface for exchanging data with other embedded systems or with the cloud (JAZDI, 2014).

Another import concept is the term Internet of Services (IoS) or Web 2.0. According to Schroth and Janner (2007), IoS describe a quickly growing set of Web-based applications. Consumers and providers can negotiate the services while engaged in a business interaction, which represents a support technology for IoS vision. In the IoS context, traditional value chains are broken up to a large extent and substituted by loose networks of providers and consumers. Using this network, the costumers’ needs can be fulfilled through a system where organizations work together, and values are created along the supply chain of the company, its suppliers, its customers, and aggregators (SCHROTH e JANNER, 2007).

Figure 4 describes IoS architecture based on a combination of principles and technologies from SOA standards. Resources that are accessible through the Web are registered in platforms and, thus, can be discovered, composed, and interlinked with the designing new resources, according to the users' requirements (SCHROTH e JANNER, 2007).

Figure 4. Internet of Services (IoS) architecture.

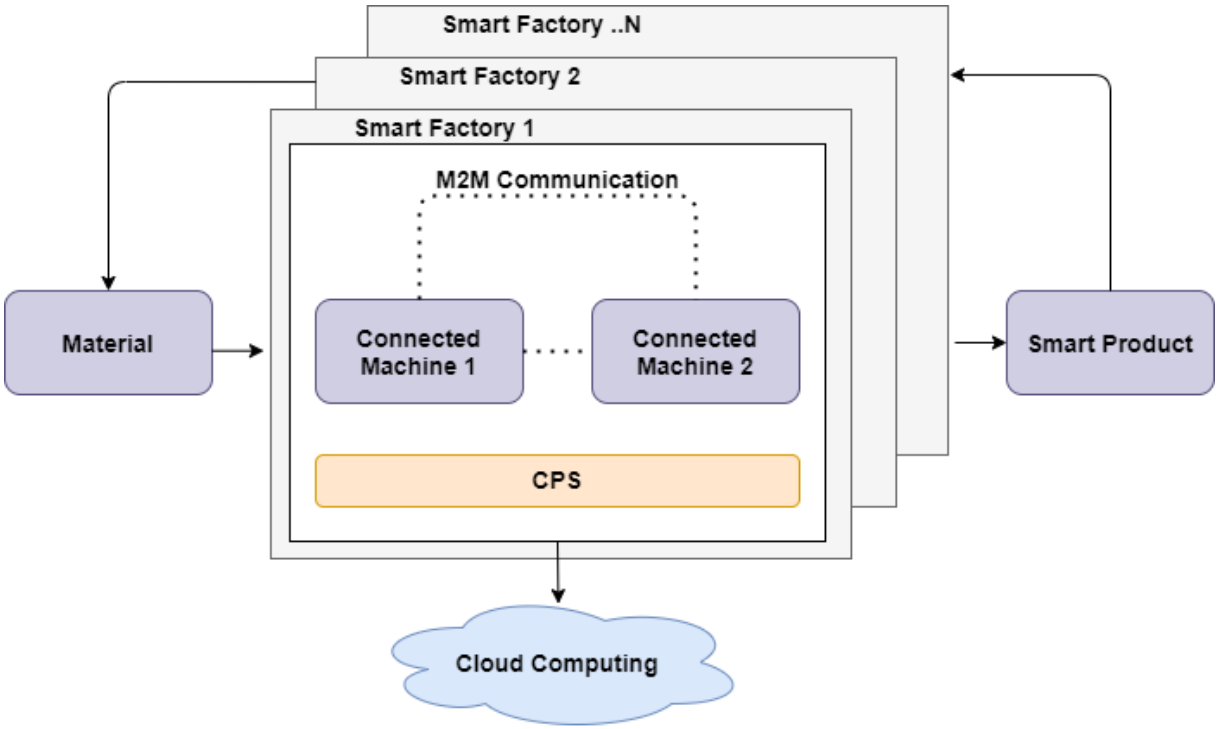


Source: Own Authorship

All those concepts combined start to mold the idea of a smart factory. A smart factory is composed of collaborative networks of industries supported by technologies based on cloud computing (LINS e OLIVEIRA, 2019). The industry is supported by CPSs IoT, and the machines can communicate with each other and with products in the manufacturing process, as can be seen in Figure 5. In the environment of the intelligent factory, the products carry information that helps control and monitoring, and also has the coordinates for each production process. The products and materials provided with this information offer their traceability, which enables effective control of the products' life cycle.

The IoT concept is essential for the smart factory (TEODORA e LIVIU, 2012). IoT tends to be more and more pervasive in the industrial sector, where the flow of information and decision-making are more efficient, with greater market receptivity for the connection of computing devices and software in operational and business processes. In automation, the use of IoT and communication networks, such as the Internet, allows the connection of equipment within the industrial plant and the optimization of production systems. In this way, machines and systems are connected within the industry, and manufacturers use the information to automate the workflow, maintaining and optimizing the production system with less human intervention.

Figure 5. Smart Factory elements.



Source: Based on (PISCHING, 2018)

2.1.1. Industry 4.0 Applications

Many organizations still discredit how Industry 4.0 affects their business and they are seeking to find talent or knowledge to understand how to best use it in their unique use cases. Several other organizations are implementing today's transformations and preparing for future information (MARR, 2018). Some possible applications are listed as follow.

- Opportunity Identification: collecting substantial amounts of data could provide information for maintenance, performance, and other issues. A deep analysis of this data may be useful to patterns and insights that humans cannot reach within a reasonable timeframe. These opportunities lift operations quickly and efficiently, understanding what needs to be addressed. By using this collected data, an African gold mine identified a problem with the oxygen levels during leaching. Once fixed, they were able to increase their yield by 3.7%, which saved them $20 million annually (BAUR e WEE, 2015).
- Optimize logistics and supply chain: connected supply chains can adapt as new information is provided. If the weather delay causes the goods to be bundled

together, the connected system can actively adapt to reality and modify manufacturing priority.

- Automatic equipment and vehicles: shipping yards are leveraging autonomous cranes and trucks to streamline operations as they accept shipping containers from the ships (BBC, 2018).
- Robots: from product assortment in the warehouse to shipment preparation, automated robots can provide manufacturers with fast and secure support and are now more affordable and available to organizations of every size. Robots move goods around Amazon warehouses while reducing costs and giving online retailers a better footprint (CNET NEWS -, 2014).
- Additive Manufacturing (3D Printing): this technology has been greatly improved over the past decade and has been used primarily for prototyping and actual production. The advancement in the use of additive metal manufacturing opens up many possibilities for production.
- Internet of Things and the Cloud: a key component of Industry 4.0 is the Internet of Things, which is characterized by connected devices. This not only helps internal operations but, by using a cloud environment that stores data, it's possible to optimize devices and operations by leveraging other people's ideas using the same devices or by allowing small businesses to access technologies they don't have.

Industry 4.0 is still evolving and its total potential is still not fully known. The companies that are using I4.0 technologies are also working to address the issue of how to upgrade existing workforces to take on the new Internet 4.0 job responsibilities and recruit new employees with the right skills (MARR, 2018).

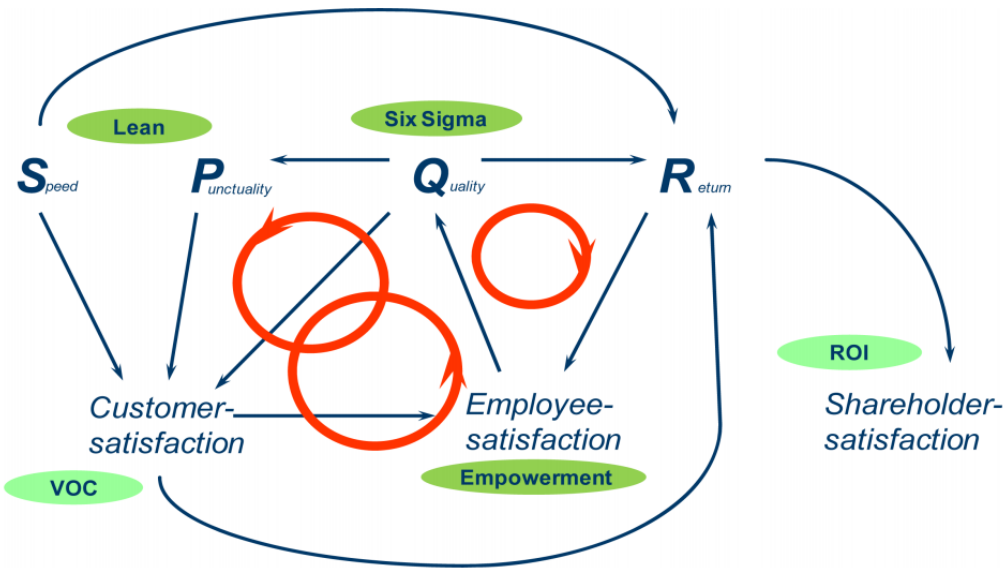
2.1.2. Industry 4.0 Potentiality

Industry 4.0 combines the strengths of traditional industries with cutting-edge internet technologies. It embraces a set of technologies enabling, smart products integrated into intertwined digital and physical processes (SCHMIDT, MÖHRING, *et al.*, 2015). On the other hand, business process complexity has a negative influence as quoted in (ABBOTT, 2014).

To explore the potential use of industry 4.0, Schmidt et al. (2015) conducted empirical research with some interesting findings of the potential use of Industry 4.0. The use of current technologies like Big Data or Cloud are drivers for the individual potential of the use of Industry 4.0. Furthermore, Mass Customization as well as the use of idle data and production time improvement are strong influence factors to the potential of Industry 4.0. Business Process Complexity has a negative influence (SCHMIDT, MÖHRING, *et al.*, 2015).

Rettmann (2016), shows a Lean approach on Industry 4.0. The concepts are very interesting in the construction of this works proposal when it comes to client orient architecture, and it's important to dig more into their contribution. Rüttimann show how the manufacturing performance of a production system must meet customer requirements. Customer requirements are a set of different needs. Apart from product quality requirements to be mandatorily observed, there are also service performance requirements in terms of e.g., speed – as well as punctual deliveries to be observed. The simplified representation and intrinsic dynamic behavior of the basic target system can be modeled with the SPQR model, depicted in Figure 6. This simple model explains the systemic interactions between the main customer-perceivable performance variables – Speed, Punctuality, Quality, and Return (i.e., price), with the systemic stakeholder variables customer, employee – and shareholder. It shows with the customer the most important element and with the employee the most vulnerable element, as well as the systemic effects on the other system variables.

Figure 6. SPQR model



Source: Rettmann (RÜTTIMANN, 2001), as quoted in [1].

The author points out how the production system itself has evolved from B&Q to SP transfer-line manufacturing, to fully automated manufacturing cells, complemented with lean manufacturing techniques following the TPS as quoted by Rüttimann (2016). The supply chain integration extends lean concepts to outbound logistics. Industry 4.0, on the other hand, adds the IoT possibilities to an existing production system in order to create an integrating cyber-physical dimension to install a supply chain integration and IoT-controlled manufacturing system.

A derived simplified but necessary roadmap and structure to be successful with the Industry 4.0 initiative is shown in Figure 7. It clearly shows the necessity to get clarity about the requirements and constraints of the system before developing the subsystems. It will rather develop from “version to version”. Standardization and internet security will be one of the biggest challenges; however, the aspect of the routing and scheduling control, influencing the performance aspect of manufacturing systems, is crucial and shows that certain objectives may have to be revised.

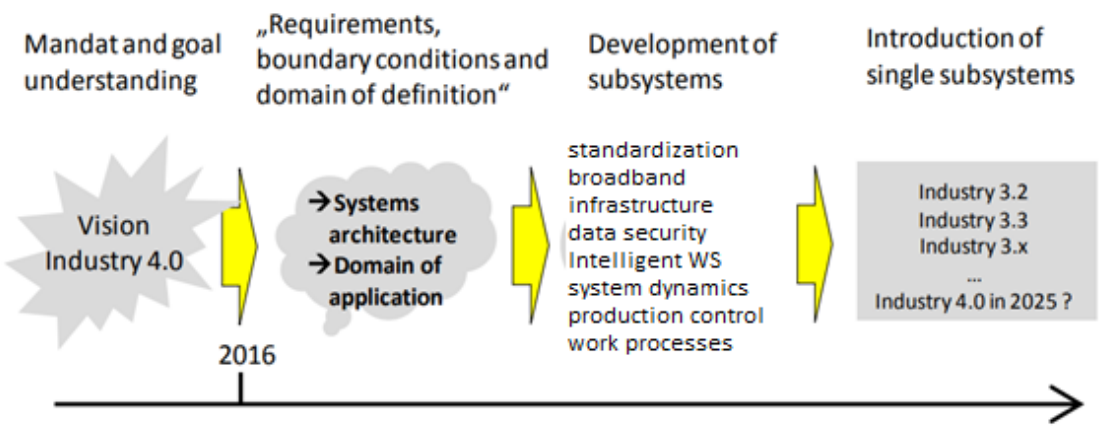
An important part the work is transcript here:

> *“… Industry 4.0 initiative as a whole has a high probability to fail, such as cybernetics in the sixties trying to automate socio-economic systems, if it is not put into the right context by considering fundamental manufacturing laws.”*

Industry 4.0 manufacturing, consequently, has to find the right domain of application. The right applicability and how to integrate the CPPS in the Lean theory are key elements for Industry 4.0, which makes Lean Production more flexible; whether it makes it faster, smoother, and more stable, and more accurate has to be proven. Whether or not enterprises will survive depends at the end heavily on manufacturing cost and performance.

In summary, Rüttimann (2016), has made a deep contribution to the elaboration of a proposition for I4.0 and top layers of RAMI 4.0. Business layers has to be customer-satisfaction orient, and, as quoted in Figure 7, there is a roadmap provided to help that achievement.

Figure 7. The industry 4.0 journey.



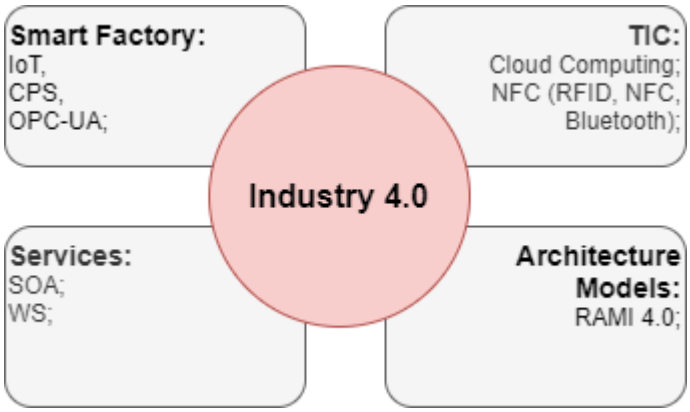
Source: (RÜTTIMANN, 2015), modified.

2.2. Service Oriented Architecture (SOA)

Services, in most cases, are digitalized to become integral parts of a virtual world. With that, the virtual and physics interact so that real-world events affect their virtual representation and vice versa. As show in the computational context, services are operations that range from simple responses to process requests to the execution of complex business procedures, which require communication between providers and consumers of virtual services (DUSTDAR e PAPAZOGLOU, 2008). Virtual services are independent processes that are described, made available, located, and invoked through communication networks.

This concept is associated with Service-Oriented Computing (SOC) which concerns a type of distributed computing platform (ERL, 2009). According to Wang (2013), the service-oriented computing paradigm is transforming traditional workflow management from a centralized and closed system to a system with decentralized and open business processes (WANG e XU, 2013). According to Kumar (2012), SOA is defined where is possible to understand the relationship of services, concepts, technologies, and architecture to enable Industry 4.0. Figure 8 brings a visual representation of the Services and Technologies of I4.0 (KUMAR, P. e NG, 2012).

Figure 8. Services and Technologies surround Industry 4.0.



Source: Based on (PISCHING, 2018)

SOA aims to improve the efficiency, agility, and productivity of a company with a focus on activities seen as all services (ERL, 2009). The goal of SOA is to offer an architecture to support the proliferation and adoption of reusable digital services (KUMAR, P. e NG, 2012). In this context, information and resources are available to all interested parts in the form of services and data that are accessed in a standard way. According to Sheng (2014) (SHENG, QIAO, *et al.*, 2014), SOA is an architecture for organizing infrastructures and software applications in a set of services that can be interacted with. SOA establishes an architecture to allow services to be published, discovered, and consumed by applications or other services (MELO, SOUIT, *et al.*, 2010). SOA represents a paradigm that involves the organization and use of services distributed in the form of software that is under the control of different owners. The idea is to ensure a uniform environment to offer, discover, interact, and use services (computational software) to produce desired effects (MELO, JUQUEIRA e MIYAGI, 2010).

2.2.1. SOA Approach and Support:

SOA is recognized as an approach to support the integration and collaboration of distributed resources. It adopts a distributed architecture based on the concept of service. In this architecture, services are invoked independently by any service requester (external or internal) to process simple functions or work together through coordinated implementations with other entities to develop new features for existing processes (KUK, KIM, *et al.*, 2008). SOA is adequate in manufacturing to handle

resources, mainly due to the heterogeneous nature of these environments (MELO, SOUIT, *et al.*, 2010). It leads to the idea that each programmable hardware is considered as a potential device whose functionality is published in the form of services and that allows its programming and or configuration through applications. The functionalities of the devices are operated by different system modules independent of the heterogeneity of the hardware/software, allowing the interoperability of the systems (VEIGA, PIRES e NILSSON, 2009).

2.2.2. SOA Implementation:

For SOA implementation, the most used technology is WS (SHENG, QIAO, *et al.*, 2014), which is a solution that groups a basic set of standards that specify the definition, exchange, and transportation of messaging and service coordination (FATTORI, JUNQUEIRA, *et al.*, 2011). A WS is essentially a semantically defined abstraction of a set of physical or computational activities involving resources, intention to meet consumer needs or business requirements (SHENG, QIAO, *et al.*, 2014).

In the context of I4.0, the recommended technology for the implementation and integration of SOA services is also a WS, mainly because it allows the implementation of technology-independent services, ensuring the interoperability of systems. Kuk et al. (2008) and Sheng et al. (2014) claims that SOA is implemented through a set of flexible and interoperable standards for distributed systems, enabling the composition of services. There are already works that use WS for the composition of services and collaboration between PS (Productive Systems) distributed geographically, the so-called DPSs, and for Dagerman and Riemensperger (2014) quoted for Pisching (2018) this is a fundamental technique to make I4.0 viable (KAGERMANN e RIEMENSPERGER, 2014) (PISCHING, 2018).

According to Pisching (2018), WS offers a standard of interoperability between different software applications, and which run on a variety of platforms. They are characterized by their great interoperability and extensibility due to the use of XML (Extensible Markup Language) and, therefore, they are flexibly combined to achieve relatively complex operations. WSs are developed in two ways: WS based on SOAP (Simple Object Access Protocol) and WS RESTful (REST - Representational State Transfer). The WS based on SOAP depends on three standardized elements: WSDL,

SOAP, and UDDI (Universal Description, Discovery, and Integration). SOAP-based WSs are protocol independent and maintain an operational state (stateful), which demands more computational resources, mainly when handling SOAP messages. Typically, SOAP-based WSs are used to integrate complex enterprise applications.

The RESTful-based WSs are low-level services to access devices and resources, using REST (MEISSNER, DOBRE, *et al.*, 2012). This form of WS implementation was introduced for the construction of large-scale distributed hypermedia systems. REST WSs interact through a uniform interface that includes a fixed set of Web operations and the HTTP (HyperText Transfer Protocol) protocol and is identified by URI, which offers a global address space for discovery services and resources. REST-based services interact via the exchange of request and response messages, each including sufficient information to determine how to process the message.

WS REST tends to be lighter from the point of view of a computational load because it does not maintain the operational state (stateless) of transactions (SHENG, QIAO, *et al.*, 2014). The services can thus be implemented on different computers and be distributed in a local communication network, or in distributed and heterogeneous networks (DUSTDAR e PAPAZOGLOU, 2008), or even in collaborative networks arranged in infrastructure based on cloud computing (PISCHING, 2018). Services are classified as atomic web services and composite web services. The first one refers to an elementary service and does not depend on other Web services, but it meets all the requirements of any service, following the consumer's requests. The second is a structure composed of atomic services and other composite services that, together, implement a set of operations to meet a specific demand (PISCHING, 2018).

3. RAMI 4.0

RAMI 4.0 is short for Reference Architecture Model for Industry 4.0. It was developed by the German Electrical and Electronic Manufacturers' Association (ZVEI) to support Industry 4.0 initiatives, which are achieving broad acceptance around the world (LYDON, 2019). Industry 4.0 is an integrated view of manufacturing enterprises, started in Germany, with many worldwide cooperative efforts including China, Japan, and India. Industry 4.0 concepts, structure, and methods are being adopted worldwide to modernize manufacturing.

RAMI 4.0 provides a view of all the important aspects of Industry 4.0 that are needed by different stakeholders. It does so by mapping all critical aspects of Industry 4.0 to three different axes making it a kind of 3D mapping of Industry 4.0 solutions. Moreover, RAMI 4.0 gives general guidelines and relevant views that should be taken into account in the overall domain organization but does not prescribe a specific modeling approach (LYDON, 2019). The six layers of the vertical axis define the structure of the Information and Communication Technology (ICT) representation for an Industry 4.0 component. This vertical axis represents the various ICT perspectives, such as the business applications, the functional aspects, information handling, communication, and integration capability, and finally the hardware/assets involved in a CPPS. The life cycle of products, machines, and factories is captured along the life cycle and value stream axis. The third axis of RAMI 4.0 describes the functional classification of various circumstances within Industry 4.0 from the product to the factory level and then the connected world (SURI, CADAVID, *et al.*, 2017).

Given this integration, entities active within the Industry 4.0 may seek mutual communication throughout the life cycle, regardless of the boundaries between companies, nations, and states (LYDON, 2019). With this, all entities within the production chain will be able to have all the necessary data. It is also noteworthy that participants in the production and commercial chain – including industrial machine manufacturers and software developers – will have the opportunity to develop their products based on prior knowledge of the latest components, not yet developed and tested by producers (ZEZULKA, MARCON, *et al.*, 2018).

Due to the effects of such comprehensive chains, leading German companies launched in 2015 RAMI 4.0, which is a three-dimensional structure. This reference

model enables step-by-step migration from today's world to 4.0 Industry and the definition of application domains with stipulations and special requirements (ZEZULKA, MARCON, *et al.*, 2018).

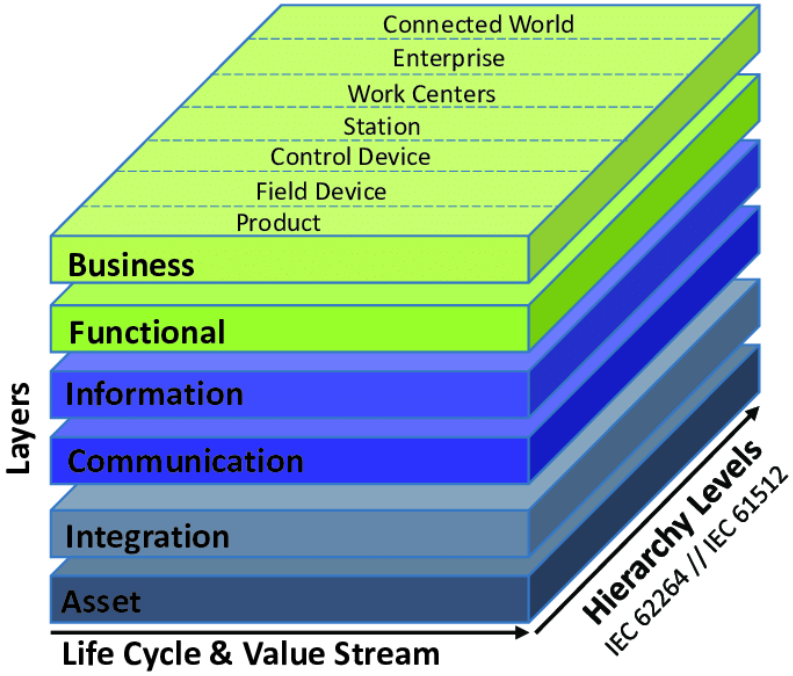
Adopting RAMI 4.0 as the design and application standard in I4.0 provides some benefits such as the use of a Service Oriented Architecture (SOA), the combination of IT components at each layer, and the division of processes into components, making communication and processing easier.

Recently, several researchers have focused on developing solutions that integrate different technologies to implement the functionality required by Industry 4.0 applications, based on RAMI 4.0. Suri et.al. (2017) proposed a model-based approach for creating and communicating business strategies and bridge the gap between the business strategy and its operation using Business Motivation Model (BMM) from Object Management Group (OMG). To achieve this, they have created BMM models to communicate business ideas and connect corresponding business tactics to the process's models in Business Process Modeling and Notation (BPMN), simulate processes to check achievable Key Performance Indicator (KPIs) corresponding to different strategies (modeled in BMM), and define organizational units in BMM that perform tasks modeled in BPMN. Contreras et. al. (2017) addresses the essential features that allow a manufacturing system to be adapted to become an I4.0 application, following the RAMI 4.0 model. To this end, the authors specify a standards-based intelligent manufacturing system such as Field Device Integration (FDI), AutomationML 2, and Open Platform Communications Unified Architecture (OPC-UA).

For the RAMI 4.0 structural axes, the reference model can be divided, as shown in Figure 9, into the following axes: Hierarchy Levels, Life Cycle & Value Stream, and IT Layers.

On the left horizontal axis is represented the life cycle value stream axis of facilities and products, based on IEC 62890, Life-cycle management for systems and products, used in industrial-process measurement, control, and automation. Furthermore, a distinction is made between *types* and *instances*. A *type* becomes an *instance* when design and prototyping have been completed and the actual product is being manufactured. The model also combines all elements and IT components in the layer and life-cycle model (LYDON, 2019).

Figure 9. RAMI 4.0 model.



Source: from Zezulka et al. (2016) (WEIß, KOELMEL e BULANDER, 2016).

As quoted by Adolph et. al, (2015), a type is always created with the initial idea. This includes placing project orders, development, and testing ranging from the first sample to prototype production. The product type, machine, etc. can be created during this phase. The authors also point out that products are manufactured industrially based on the general type, with each manufactured product representing such an instance, which may have, for example, a unique serial number.

3.1. Hierarchy Levels Axis

On the right horizontal axis of RAMI 4.0 model are hierarchy levels from IEC 62264, the international standards series for enterprise IT and control systems. The hierarchy levels represent the different functionalities within factories or facilities. To represent the Industry 4.0 environment, these functionalities have been expanded to include work pieces, labeled *Product*, *Field Device*, *Control Device*, *Station*, *Work Centers*, *Enterprise* and *Connected World*. (LYDON, 2019).

A more precise description of the hierarchy levels is of RAMI 4.0 is described as (ADOLPHS, BEDENBENDER, *et al.*, 2015):

- Product: this allows a homogeneous view of the product to be manufactured, the ease of production, and the interdependencies between them;

- Field Device: represents the functional level of an intelligent device, for example, an intelligent sensor. They are electronic devices used to detect and identify sensor components and technologies;
- Control Device: considered the brain of manufacture. These are usually machines/sensors used to manage input/output commands such as Programmable Logic Controller, Distributed Control Systems, and Graphical Interface;
- Station: where operators perform administrative activities to examine the operation of events and processes, such as the use of SCADA;
- Work Centers: maintains manufacturing information, defines the stage of production, and oversees the renewal of raw materials for refined products;
- Business: this is generally defined in terms of ERP, which is business management software used in business processes such as production planning, service delivery, marketing, sales, finance modules, retail, and other expenses;
- Connected World: this item has been added at the top end of the hierarchy levels to meet Industry 4.0 needs that describe the factory group and collaboration with outside engineering firms, component suppliers, and customers.

3.2. Layers Axis

The IT layers axis on the vertical of the RAMI 4.0 model, describe the decomposition of a machine into its properties, structured layer by layer as the virtual mapping of a machine. Such representations originate from information and communication technology, where properties of complex systems are commonly broken down into layers.

According to Wang et al. (2017), and Adolphs et al. (2015), RAMI 4.0 Layers axis can be described in the following aspects:

- Business Rules: covers abstract business models and process logic, as well as ensure the integrity of functions along the value chain. The layer ensures the integrity of functions in the value stream, enables mapping business models and the resulting of the overall process. It contents legal and regulatory Framework

conditions, enables modeling of the rules which the system must follow. The layer also creates a link among different business processes;

- Functional: enables a formal description of roles and creates a platform for horizontal mixing of various functions. Contains run time and modeling environment for services for support of business processes and a run time environment for applications and technical functionality. Rules and decision-making logic are generated in the Functional layer. Some use cases can be executed in lower layers as well. But remote access and horizontal integration can take place within the Functional layer only because of the necessity of data integrity;
- Information layer: provides run time for preprocessing of events and execution of event-related rules. It enables a formal description of the rules and event preprocessing. For this, data is checked for its integrity, summarized in new, higher quality data, and made available to superordinate layers such as functional. It also receives events and transforms them to match the data which are available for the higher layer;
- Communication layer: this layer provides standardization of communication through uniform data format in the direction of the Information layer. This layer provides services to control the integration layer;
- Integration: providing information related to technical assets such as hardware and software for the overlapping layers. Wang et al. (2017), and Adolphs et al. (2015) point out that this layer contains all elements associated with IT, including HMI generating events based on information acquired and performing final process control (WANG, TOWARA e ANDERL, 2017). This layer makes provision of information on the assets (HW/SW, components) in a form that is available for computer processing. It is also responsible for computer control of the process, generation of events from assets and it contains elements, which relate to IT, such as RFID readers, sensors, HMI, actuators. Integration of persons is a part of Integration layer functions as well, via HMI;
- Asset layer: this is the lowest layer within the RAMI 4.0 framework. It represents physical reality, containing objects such as processes, machines, sensors, actuators, and documentation. This layer identifies products and parts concerning processing requirements for machines and workstations. Humans

are a part of the Asset layer, by being connected with the virtual reality world by the Integration layer. Passive connection of the assets to the higher Integration layer is done by for instance means of QR codes.

3.3. RAMI 4.0 in the Context of I4.0

Life Cycle & Value Stream, Hierarchy Levels and Layers, all crucial aspects of Industry 4.0, can be mapped, allowing objects such as machines to be classified according to the model. Highly flexible Industry 4.0 concepts can thus be described and implemented using RAMI 4.0. The model allows for step-by-step migration from the present into the world of Industry 4.0 (LYDON, 2019).

Kannan et al. (2017) dive into the context of the automotive industry, in an evident gap between the requirements supported by the current automotive Order and the requirements proposed by industrial standards from the International Society of Automation (ISA). The authors bridge this gap by following a model-based requirements engineering approach along with a gap analysis process. In this work, the authors follow five basic steps described here.

Product Analysis: the four-top shortlisted Order orchestrator tools are used to benchmark the product features for a state-of-the-art Order Orchestrator tool development. All these features are reviewed and recommended for the development of Order Orchestrator:

- Platforms: The supported platforms are into two groups (server and client systems);
- Database: Oracle and SQL Server are dominant at the supported database systems;
- Technology: it is suggested to support technologies such as Com-technology, MS-platform, JAVA platform, ODBC, and HTML5 for MES product development;
- Architecture: it is recommended to apply Client/Server, distributed, and SOA architecture as all of the benchmarking tools supports the same;
- Validation and Regulatory Compliance: it is endorsed to comply to GAMP regulated environment, 21 CFR part 11 and validation dossier MES development;
- Languages: it is recommended to support Full-NLS, Full National Language Support for improved user comfort and Unicode standard;

- Incident Management: it is suggested to support Detecting latent incidents and have both Activate tracing and De-activate tracing;

It is interesting to notice how the technology has changed since this article came out, and still is a solid reference for IT layers of RAMI 4.0

Requirement Modeling Phase: The ISA-95 is one of the building blocks for RAMI 4.0. Additionally, the authors cross-checked with the ISA-88 standard. Once collected all the needs from ISA-95, the requirements are framed from the needs and modeled using SysML. A total of 106 requirements were found, and grouped by functionality given in the ISA-95 standard (ISA-95.00.03 STANDARD, 2013). They are:

- FR01 - Managing the Manufacturing instructions - The software may be able to the work masters, manufacturing instructions, recipes, product structure diagrams, manufacturing bills, and product variant definitions;
- FR02 - Manage New Product Definition - The software shall manage the new product definitions;
- FR03 - Manages Product Definition Changes – The software shall be able to make changes to product definition;
- FR04 - Production Rules to Personnel - The software shall provide the product production rules to personnel and relevant activities;
- FR05 - Maintaining Assembly Steps - The software shall maintain the feasible detailed production routing for products;
- FR06 - Provide Detailed Route - The software shall provide the detailed route to the manufacturing operations;
- FR07 - Exchange Product Definition Information – The software shall manage the exchange of product definition information with Level 4 as per the business operation requirement.

Gap Analysis Phase: In this Section, the authors detail the steps involved in the gap analysis:

Product - Standard Compliance Details: Before starting the gap analysis, it is necessary to know how Automotive MES products fulfill the requirements of the ISA-95 standard. To get inputs for this process, technical reports and literature surveys act as the main source of information. MES functionalities are divided into four categories

as per the ISA95 standard that is*, (i) Production Operations Management, (ii) Maintenance Operations Management, (iii) Quality Operations Management, and (iv) Inventory Operations Management.* Using the data from the reports, compliance percentages of MES products w.r.t standards for each category are found ((CGI, 2015) as quoted by (KANNAN, SURI, *et al.*, 2017)).

Steps Involved in Gap Analysis: For the gap analysis, a list of 37 vendors was selected. Using the vendor compliance data from the survey, a scale range of 1 to 5 would be assigned to each of the 33 requirements for each of 19 MES vendors

Gap Analysis Result: For the gap analysis, out of a total score of 95 for each requirement, the requirements that had a score of 85 and above were segregated and prioritized. Thus, if an MES tool satisfies these prioritized requirements, it will help in solving the challenges faced by the automotive industry.

Kannan (2017), work was successfully able to perform the model-based requirements using engineering technique for finding, modeling, and relating the needs to develop a state-of-the-art MES tool using the ISA95 standards. Also, to perform a gap analysis between the modeled needs and the available MES tools, their orientation and product were used as a base to model production perfective applied in flexible manufacturing.

Almada-Lobo's (2016) contribution shows how Industry 4.0 dictates the end of traditional centralized applications for production control. Act In Response to customer demands for customized products, these plants fueled by technology enablers such as 3D printing, Internet of Things, Mobile Devices, and Big Data, among others, create a new environment. The manufacturing systems of the future, including manufacturing execution systems (MES), will have to be built to support this paradigm shift.

The reaction of MES providers**:** the direct consequence for centralized systems is that they will simply cease to exist. For Manufacturing Execution System providers, this will become a quite challenging scenario. The first group, representing the big majority is simply ignoring Industry 4.0 and doing business as usual. A second group is paying more attention to it. However, they claim Industry 4.0 defines a target model which will most likely take years or decades to reach. A third group, however, argues that decentralized systems will always need a centralized system due to compliance, optimization, and monitoring, which is contradictory;

Manufacturing Execution Systems of the Future**:** manufacturing Execution Systems have been pivotal in the performance, quality, and agility needed for the

challenges created by a globalized manufacturing business and will most likely continue to be. However, a completely new generation is required to cope with the new challenges created by Industry 4.0. The following are the four main pillars these systems shall consider:

- Decentralization: this decentralization doesn't need to be physical, but instead a logical one. With cloud computing, it's even arguable if such a system can be considered physically centralized;
- Vertical Integration: all services which the different CPS and CPPS entities can provide are exposed, allowing their orchestration in business processes that may be simple or complex for compliance or more broadly related to quality, logistics, engineering, or operations. The MES must then be truly modular and interoperable, logically decentralized, so that all functions or services can be consumed by smart materials, smart equipment, or any other shop-floor entity;
- Connectivity and mobile: advanced manufacturing environments have had such connectivity for a long time. On the more operational MES front, connectivity and mobile combined shall allow more adaptable interfaces. MES will consist of different applications, making the vision of getting to a piece of equipment, downloading, and later using an app specifically built to operate that equipment will become a reality;
- Cloud computing and Advanced Analysis: CPS and CPPS will generate huge amounts of data, which needs to be stored and processed. The Smart Factory vision of Industry 4.0 requires achieving a holistic view of manufacturing operations. Advanced analytics are then needed to fully understand the performance of the manufacturing processes, quality of products and supply chain optimization.

The work of Almada-Lobo (2016), brings key aspects of MES and Industry 4.0 for a successful IT implantation. It is well known that centralized and monolithic production monitoring and control applications will eventually come to an end because it's not scalable and become unable to grow. Solutions using these principles already exist today and are the ones that shall support manufacturers in creating their production comprehensive picture and roadmap, with step-by-step actions, leading to the ultimate vision of Industry 4.0. As manufacturers build their Industry 4.0 roadmaps,

they must understand these core principles so they are not faced with difficult replacement decisions.

4. METHODS AND TECHNOLOGIES

This section describes the project proposal, main discussed architecture, methodologies and technologies used for business and functional layers.

4.1. Project Proposal

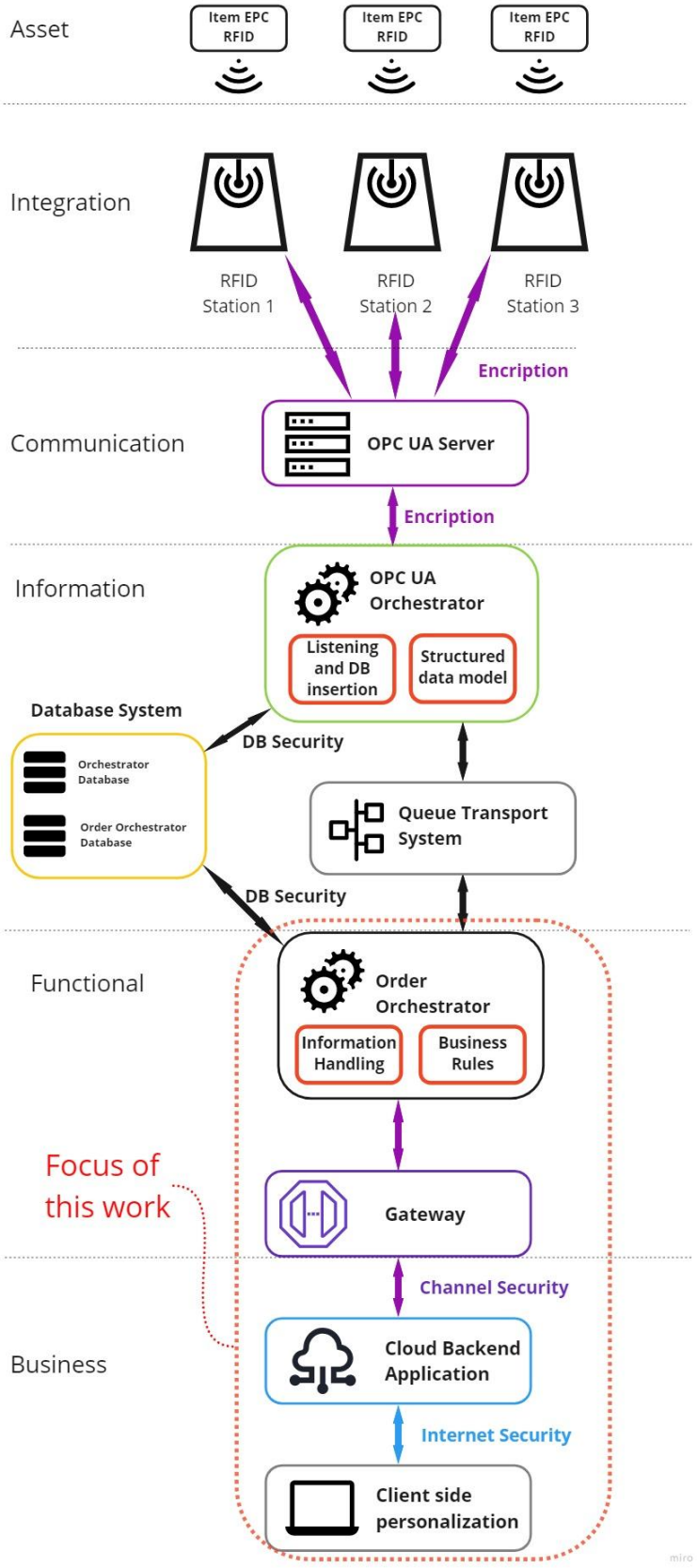
This research is part of a greater research project that aims to develop an microservice-oriented architecture based on RAMI 4.0 for flexible manufacturing applications in Industry 4.0. A macro-overview of the architecture layers proposed and presented in Figure 10. It is important to clarify that this Master's project contemplates the development of the parts in Functional and Business layers in Figure 10 (highlighted in the red box), which in resume considers the Order Orchestrator, the API Gateway (API GW), the Swagger documentation and Client application.

Note that the architecture describes a flow to help the reader to understand the production flow. It starts in the RFID tag in a factory floor environment (Asset layer), followed by the reading station, in which the RFID tag is read by an RFID antenna; the data is interpreted and sent to the OPC UA Server to link factory information to system data. The RFID stations are part of the manufacturing stations (Integration layer), which are responsible for the production logic control. The station's communication with the OPC UA is bi-directional since it is always sending information that reached it and questioning what should be done with that item (Communication layer). The parts related to Asset, Integration and Communication layers of the project architecture have been covered in previous researches (MELO, 2020).

The OPC UA standard is described in this work by the server and the orchestrator (client). The OPC UA Server is the block for capturing and sending information to the manufacturing stations. It is responsible for knowing where each data should go using the OPC UA topics; OPC UA topics are channels to organize information. The OPC UA Orchestrator is a client responsible for the ingestion of the information and decision making, based on business rules. While the OPC UA Server knows the data flow at the bottom layers, the Orchestrator also knows the data flow at the top layers, based on business and decision-making rules. All decision-making is based on the information that comes from the factory crossed with the information contained in the database. In addition, the production rules bring the vision of production on-time, and history of the registration of the tags. All information passed

through the OPC UA is made secure by encryption using a certificate.

Figure 10. Microservice Architecture based on RAMI layers for Industry 4.0.



Source: Own Authorship.

The OPC Orchestrator and the database are related to the Information layer. These elements are being developed in another research (RAVELLI, 2022) of the project. The database is a no-SQL and temporal language.

The Order Orchestrator and the API Gateway are related to the Functional layer. The Order Orchestrator is responsible to bring data from the Business layers to the Integration layer. In other words, refers to the customer's manufacturing requests. It is in this layer that new orders are inserted into production, manufacturing status is monitored and production upgrade wishes are deemed feasible, or not, in order to enable the capacity of reconfigure the production.

To make it more comprehensive, the Order Orchestration, shown in the Figure 10, was separated into two main blocks, the first for handling information from top and bottom layers and the second to business rules management. Order ingestion validates the inputs from the API GW, ingests, validates the data, and direct internally; the business rules is responsible for prioritization, and parameterization of the order in the database.

The communication from the Order Orchestrator to the Client is done through an API gateway that takes this information in an organized and structured way to the customer interface. The API Gateway delivers information in a very structured way to enable any client to connect using it, which is very important to guarantee scalability, and security. For security, the information is encrypted with the OAuth 2.0 protocol. Using Transport Security Layer (TLS), both OPC UA and Order orchestrator communicate with the database, without offering security gaps.

All clients are identified in the DB with usernames, and any Order is always related to a client. A client has an API-based integration through factory API Gateway, so it can place new Orders, follow Order fabrication status and modify or delete ongoing Orders, this factory-client integration is represented in Figure 10 by the Cloud backend. Cloud backend applications can take care of client orchestration and internal integration. This part is related to the Business layer. From the factory perspective, the Order orchestrator is responsible to organize, handle and apply business rules to client and factory-side information. This is made to guarantee that production is been correctly followed and centralized in the DB.

The perspective of this architecture consists in the integration, standardization and communication of the production process. Even if is one or thousands of clients,

manufacturing one or thousands of products, integration is standardized through the RESTful APIs. The client application is seen as any integration platform that makes sense to the business, whether via mobile application, website, or via dedicated software (like an ERP for example). Through the APIs, any interface can be connected to the factory, sending and consuming information from it.

Considering the information flow during production, after an Order is placed and production starts, items identified by EPC (which is a combination of the Part Number and the Serial Number) start to flow in the production line. Every time an item is near a decision point or a manufacturing station, an RFID antenna identifies this EPC, validate their route, and direct in which way it may go. The flexible manufacturing is defined in this proposal as a sequence of manufacturing stations that an item is required to go, where the stations and the production flow may change without harming the production process. The Order Orchestrator is responsible to guide this particular item through its final station in accordance to the production execution (using a factory flow map or knowledge route for product production) and client placed orders.

The following are some technologies that made the project possible, such as programming language Node.js environment, Moleculer framework for microservices, Mongo DB Database, API Gateway, and OAuth 2.0.

4.1.1. Node.js

Node.js is an event-oriented language. It is an event-driven language, which orientation is the work with I / O events from the server, for example, the connect event of a database, a file open, a data streaming data, among others. Event-Loop is the trigger, which is listening/waiting for something and launching execution events for the modules. In practice, it is an infinite loop that each iteration checks in a queue of events if a particular event has been emitted. When it occurs, an event is emitted and sent to an execution queue, and new actions can be programmed and executed in a call-back function. The programming languages .NET, Java, Python systems have a common feature of blocking I/O on the server or Blocking-thread. On a web server, it can be understood in a wide way, considering that each process is a request made by the user. With the increase in users accessing and generating a request on the server, the system queues each request and then processes them, one by one. While one request is processed, the others are on hold, maintaining a queue of idle requests for a while (PEREIRA, 2014).

Node js is a highly scalable and low-level platform since programming is done directly with network and internet protocols or using libraries that access operating system resources. JavaScript is the basic programming language, and it was made possible through Google Chrome's JavaScript V8 engine (CANTELON, HARTER, *et al.*, 2013).

The Node.js platform has some important characteristics, such as (KOTA e PRABHU, 2014):

1. Asynchronous execution: the most suitable event model for the creation of highly scalable Web applications through callbacks;

2. Less learning curve: developers are familiar with JavaScript and asynchronous programming;

3. Practicality: advances in execution speed have made it more objective to write server-side software;

4. Code Sharing: client and server development can be reused, which helps to reduce the switching context between client and server development, allowing code sharing;

5. NoSQL: JavaScript is the language used in several databases, such as CouchDB, MongoDB.

6. JSON: it is a simple, efficient, and popular data format nowadays;

4.1.2. Moleculer Framework

Moleculer is a fast, modern, and powerful microservices framework for Node.js. It helps to build efficient, reliable, and scalable services. Moleculer provides many features for building and managing microservices. The Moleculer framework presents a base structure or a development platform, which contains tools, libraries, systems, and components that streamline the process of developing a specific solution. The Moleculer framework is a development framework with the JavaScript language for developing applications oriented to microservices. The framework is open-source and runs on Node.js (MOLECULER JS, 2021).

Some features about Moleculer are presented on their official website (MOLECULER JS, 2021):

- Promise-based solution (async/await compatible);
- Request-reply concept;

- Support streams;
- Support the event-driven architecture by balancing;
- Built-in service registry & dynamic service discovery;
- Load-balanced requests & events (round-robin, random, CPU-usage, latency);
- Many faults tolerance features (Circuit Breaker, Bulkhead, Retry, Timeout, Fallback);
- Supports middleware;
- Supports versioned services;
- Service mixins;
- Built-in caching solution (memory, Redis);
- Pluggable transporters (TCP, NATS, MQTT, Redis, NATS Streaming, Kafka);
- Pluggable serializers (JSON, Avro, MsgPack, Protocol Buffers, Thrift);
- Pluggable validator;
- Multiple services on a node/server;
- All nodes are equal, no master/leader node;
- Parameter validation with fastest-validator;
- Built-in health monitoring & metrics;
- Official API gateway module and many other modules.

Moleculer is a microservice and has an automatic registration and discovery feature available to use in development. All existing services are informed after the creation of a new service or the availability of new functionality. Another feature is automatic load balancing, which has the function of dynamically distributing the communication load between microservices evenly (PONTAROLLI, BIGHETI, *et al.*, 2020).

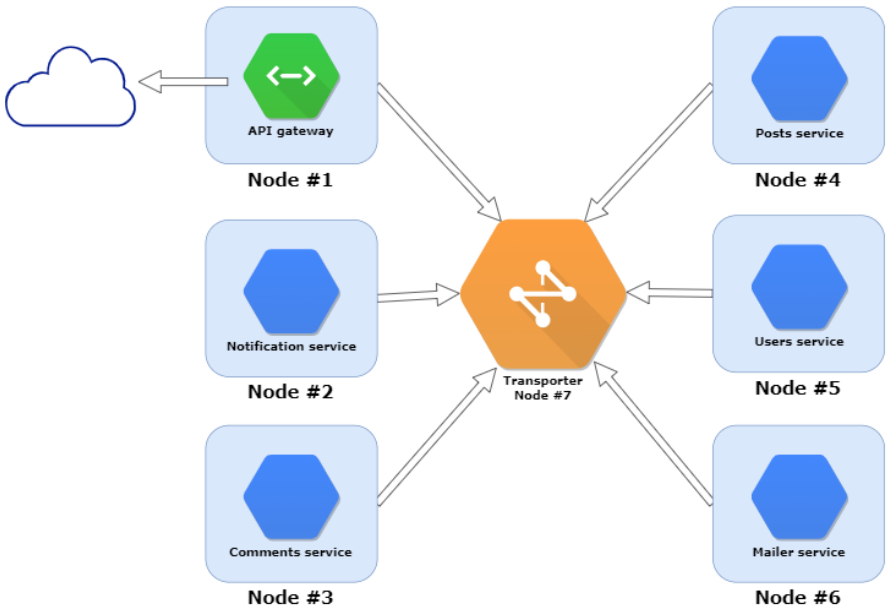
In Moleculer framework, services are executed on individual nodes that communicate via a communication protocol (transporter) through a message broker. In this distributed architecture, the latency of communication between services is not insignificant, but services can be replicated to provide resilience and fault tolerance.

In the Moleculer architecture seen in Figure 11, all microservices are the same, with no hierarchy or priority. A great advantage of this framework is that all developed microservices have an automatic registration and discovery feature available. In this way, all existing services are informed after the creation of a new service or the

availability of new functionality in a service. Another important feature is automatic load balancing, which has the function of dynamically distributing the communication load between microservices evenly.

The configuration of the Transporter type is done in the ServiceBroker method of the microservice in the Transporter option. After configuring the type of message broker (Transporter), the Communication between microservices occurs transparently, so, no matter what communication mechanism is used, no additional specification or configuration is required for communication between microservices. Internal stock calls are performed by the broker call function. External calls to action occur through the Gateway microservice. The messaging pattern used is the publish/subscribe method.

Figure 11. Moleculer Microservices architecture.



Source: Moleculer official documentation (PONTAROLLI, BIGHETI, *et al.*, 2020).

4.1.3. API Gateway

Customers want to have access to company data and services through digital devices and channels. To run into customer expectations, enterprises need to open their assets in an agile, flexible, secure, and scalable manner. APIs allow the access into an enterprise's data and services. They allow applications to easily communicate with each other using a lightweight protocol like HTTP.

According to Brajesh (2017), the API gateway is an API management tool that

stands between the customer and a collection of back-end services. It works as a reverse proxy, which accepts all calls from the API, aggregates the various services necessary to perform them, and returns the appropriate result. Most company's APIs are deployed through API gateways. These gateways typically manage common tasks that are used in a system of API services, such as user authentication, rate limiting, and statistics.

An API gateway is part of the API management system. It intercepts all incoming requests and sends them through this system, which processes several necessary functions. The exact role of the gateway varies from one implementation to another. Some common functions include authentication, routing, rate limiting, billing, monitoring, analysis, policies, alerts, and security (RED HAT, 2021).

According to with Amazon Web Services documentation (2021), the main benefits are:

- Efficient API development: running multiple versions of the same API simultaneously with the API Gateway, allowing you to iterate, test, and quickly launch new versions;
- Performance at any scale: provide end users with the lowest possible latency for API requests and responses by leveraging our global network of points of presence with Amazon CloudFront;
- Easy monitoring: monitor performance metrics and information about calls to the API, data latency, and error rates on the API Gateway dashboard;
- Flexible security controls: authorize access to your APIs with AWS Identity and Access Management (IAM) and Amazon Cognito. If using OAuth tokens, API Gateway will natively support OIDC and OAuth2;
- RESTful API: create RESTful APIs using HTTP APIs or REST APIs. HTTP APIs are the best way to create APIs for most cases (AWS, 2021).

Moleculer API Gateway is a Service that makes other Moleculer Services available through REST. It allows the creation of high-performance, non-blocking, distributed web applications. Web request processing can be fine-tuned using server-independent middleware. The Moleculer Gateway service (API) is responsible for the standardized interface connecting Moleculer services to external applications. To access the Gateway microservice, the device's IP must be used on port 3000. Some

features of moleculer-web to publish your services as RESTful APIs are described as (MOLECULER JS, 2021):

- support HTTP & HTTPS;
- serve static files;
- multiple routes;
- support Connect-like middleware in global-level, route-level, and alias-level;
- alias names (with named parameters & REST routes);
- whitelist;
- multiple body parsers (JSON);
- CORS headers;
- rate limiter;
- before & after call hooks;
- buffer & Stream handling;
- middleware mode (use as a middleware with Express).

4.1.4. Oauth 2.0

According to the Internet Engineering Task Force (IETF) official website of standardization of OAuth 2.0 (2012) (HARD, 2012), The OAuth 2.0 authorization framework enables a third-party application to obtain limited access to an HTTP service, either on behalf of a resource owner by orchestrating an approval interaction between the resource owner and the HTTP service, or by allowing the third-part application to obtain access on its behalf. This specification replaces and obsoletes the OAuth 1.0 protocol described in RFC 5849.

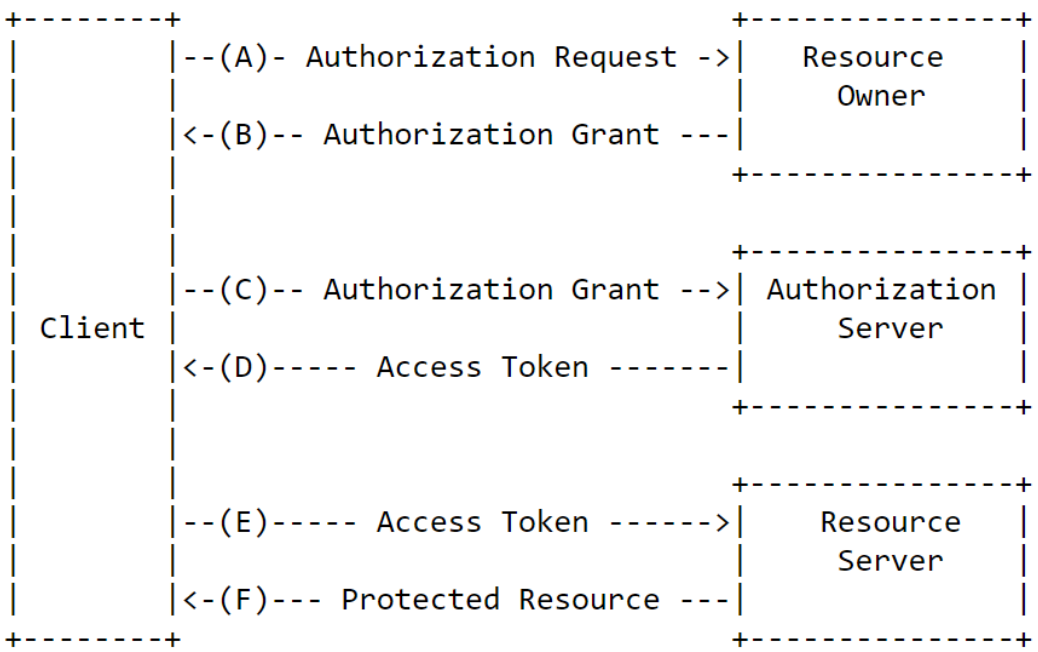
OAuth defines four roles:

- Resource owner: entity capable of granting access to a protected resource
- Resource server: server hosting the protected resources, capable of accepting and responding to protected resource requests using access tokens;
- Client: making protected resource requests on behalf of the resource owner and with its authorization. The term *client* does not imply any particular implementation characteristics (e.g., whether the application executes on a server, a desktop, or other devices);
- Authorization server: the server issues access tokens to the client after successfully authenticating the resource owner and obtaining authorization.

The interaction between the authorization server and resource server is beyond the scope of this specification. The authorization server may be the same as the resource server or a separate entity. A single authorization server may issue access tokens accepted by multiple resource servers. The abstract OAuth 2.0 flow illustrated in Figure 12 describes the interaction between the four roles.

First, the client requests authorization from the resource owner. The authorization request can be made directly to the resource owner. Then, the client receives an authorization grant, which is a credential representing the resource owner's authorization. The authorization grant type depends on the method used by the client to request authorization and the types supported by the authorization server. The client requests an access token by authenticating with the authorization server and presenting the authorization grant. The authorization server authenticates the client and validates the authorization grant, and if valid, issues an access token. The client requests the protected resource from the resource server and authenticates it by presenting the access token. The resource server validates the access token, and if valid, serves the request.

Figure 12. OAuth 2.0 Protocol Flow.



Source: IETF (HARD, 2012).

Many API gateways use OAuth 2.0 as the basis of a single sign-on service to

ease password management for users. However, little attention has been paid to security in practice. The paper of Li and Mitchel (2014), reveals two critical vulnerabilities present in many implementations. Both of them allow an attacker to control a victim user's accounts at a relying party without knowing the user's account name or password. The authors listed the following recommendations:

- Deploy countermeasures against CSRF attacks: one reason the OAuth 2.0 systems investigated are vulnerable to CSRF attacks is that the RPs do not implement countermeasures;
- Do not use a constant or predictable state value: some RPs include a fixed state value in the OAuth 2.0 authorization request. In this case, an attacker can forge a response, since the RP cannot distinguish a legitimate response produced by a valid user from a forged response;
- Check the state value: RPs that includes an opaque state value in their OAuth 2.0 request should check the state value in the response before completing binding, they recommend that RPs use a session-dependent state value;
- Require the user to input account information: definitely, the simplest way to prevent the CSRF attack is to require users to input their account names and passwords before completing binding.

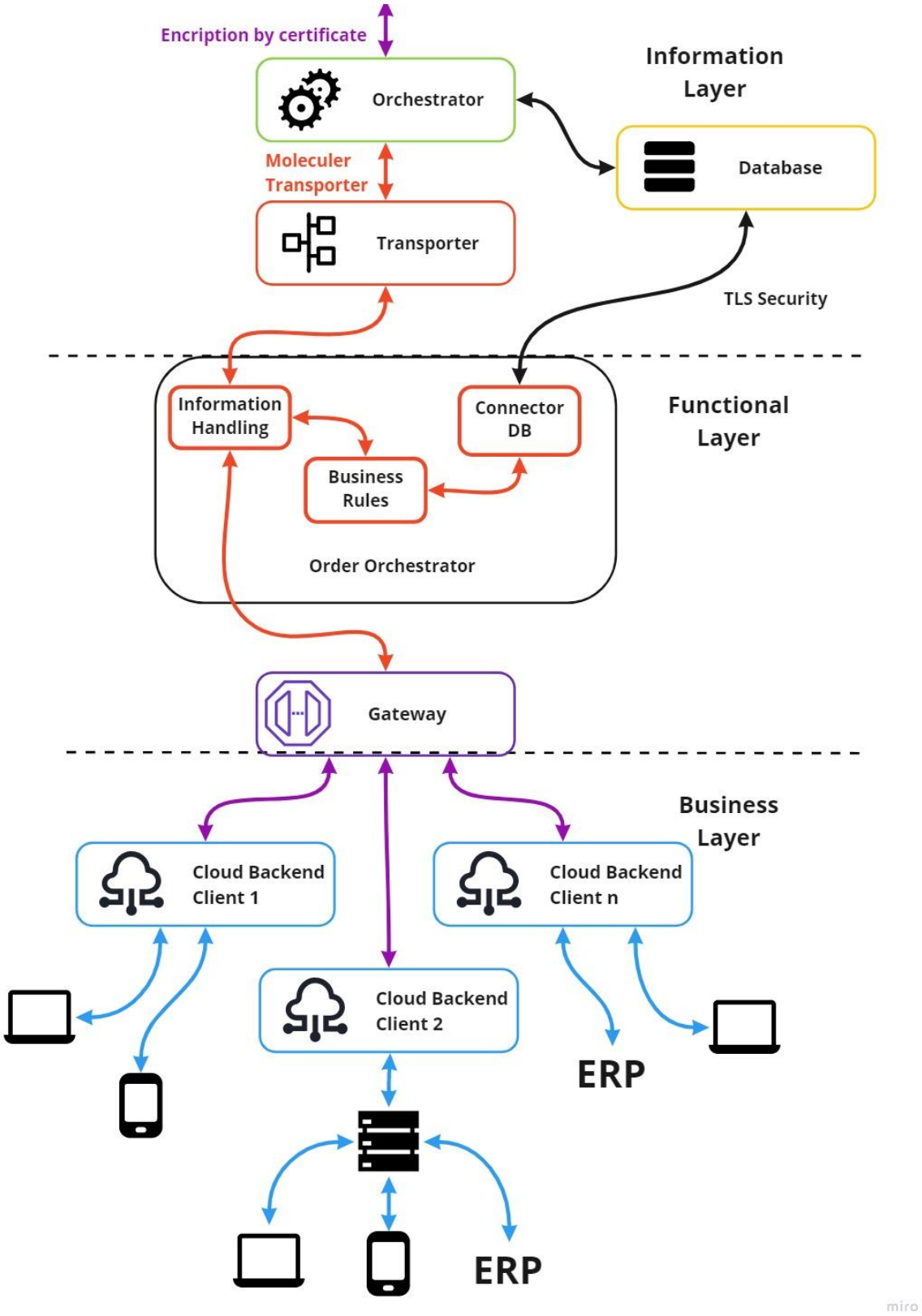
5 DEVELOPMENT AND RESULTS

To demonstrate the concept explained in this proposal, the author in accordance with his working partner (RAVELLI, 2022), have develop 3 main programs, running on node.js, and the front-end using Node-Red. The software programs use Docker Environment (DOCKER WEBSITE, 2022) to run the virtual machine where the program runs, and MongoDB (MONGODB, 2022) Database software to factory data storage, and Robo 3T (ROBO3T - ROBOMONG, 2022) to access the data stored in MongoDB Software. Rabbit MQ was the Moleculer framework transporter defined and is responsible for queue control and management, and is also running on Docker environment. To organize and create a microservice code, the programs were divided in three parts:

- The Order Orchestrator – The responsibility of this software is to organize order production, controlling how they are stored, how the item orders are associated, and how they change and flow between orders when the client requests;
- OPC UA Orchestrator – Responsible for factory production organization, it is responsible to guide item production based on stored knowledge routes and to guarantee production success by item efficient command and control process;
- API Gateway – Responsible for client-factory interface. Deliveries a secure and efficient contract to allow clients to easy and securely connect to the factory, to insert, view, update and delete orders and items. Also, Node-Red was used for frontend development. In Node-Red pallet was created the orchestrations on calling API Gateway to organize the data to be shown to the client. Node-Red dashboard pallet was used to help creating the frontend. More details about the implementation and deployment are presented in APPENDIX A, while, a few cases will be replicated here in order to help with the explanation.

Now, it is important to understand the flow, responsibility of each layer. Functional layer's main characteristic is orchestration and guarantee reliable integration. From the factory-side perspective, the Functional layer is responsible to address every EPC coming from and to the DB. Also, it handles every information from the client-side perspective. In Figure 13, a full vision of Information, Functional and Business layers of RAMI 4.0, and the flux of information between components can be seen.

Figure 13. RAMI 4.0 layers of the Architecture, emphasis on Functional and Business layers.



Source: Own Authorship.

All factory-side information is handled by the OPC UA using the Transporter to

get it through Order Orchestrator. The Orchestration in Functional layer handle this information, understanding where it comes from, where it should go, and which Business Fabrication Rules are applied to it, gathering factory floor data, Business Rules and DB logged information. After that, the Order Orchestrator sends a response to the service that has been questioned and inserts this new status on DB. For example, for the factory-side, if an EPC in a particular station is read by the RFID antenna, the Order orchestrator will evaluate the next station for this EPC. An example on the client side: if a client is requesting a new order, the Order orchestrator will receive, ordinate, and send to DB this new order to start production.

The API gateway contemplated in the Integration layer has a description of the major attributes created to implement a 100% API-based client connection and all the API endpoints can be seen in the Business layer session in a few.

In Figure 13, a detailed diagram of the architecture proposition is depicted. Even though the Information layer is not part of this work, the piece is presented to help to understand its interaction with the Database. From the top to the bottom of Figure 13, notice that the Database is connected to the Order Orchestrator in the Functional layer and in OPC UA Orchestrator in the Information layers.

Two of the motivations of this work is the Functional layer Order Orchestrator. The Order Orchestrator is responsible to handle information from the Client and Factory-side. To handle the information, the Order Orchestration has to validate all the requests, based on the factory and client's business rules, after validation and if the request makes sense; it uses Moleculer connector to CRUD in the Database.

The Order Orchestrator is the essence of this work; it's the main module to communicate to both sides, handling production in the factory and handling orders and updates from the clients. This proposition was though because the market perspective is changing faster each day, orders have to be produced quickly and be available to change when market changes, and the technology has to be prepared to help production to react to change. So, in this work, the client can change an ongoing order in any desired point. The system will calculate how much of the production can change without lost, and will return it to the client.

The second most important feature in this work is the API Gateway. API is today's state-or-art integration technology, and, as demonstrated in 2.2, a microservice path to scalable and easily integrate the solution. The API Gateway provides from one to an uncountable number of clients to access and fabricate items in the factory through

an easy and touchless setup. The client can manage its backend integration and orchestration by any means desired to, using its own ERP, like SAP or other providers, or connecting to its standard software, such as a PC or a Smartphone. The key is to allow the client to creates his interface, obeying factory business rules.

5.1. Functional Layer

The Functional layer is responsible for the interaction between the Business Layer and the Information Layer, it dictates how the business is transferred in to computer interpretable information. In the context of the RAMI 4.0, the Functional layer enables a formal description of roles, contains the modeling environment for services for support of business processes, rules and decision-making logic. In this research the Functional layer dictates how any client will interact with the factory, it standardizes the interaction flow of communication and guarantee that the communication is safe, reliable and scalable to be open to communicate with any client.

The Functional layer of this research contains the Order Orchestrator and the API GW. API GW consults frequently the Order Orchestrator for decision making. A set of endpoints of the API GW is resumed in Table 1. When an API interacts with another system, the elements of this communication flow are considered endpoints. In this research, an endpoint will define the interactions and communication flow among the elements such as the Client Application (Business Layer), API Gateway and Order Orchestrator (Functional Layer) and Database (Information Layer).

The endpoints created in this research were divided in two main groups: the Clients endpoints and the Factory endpoints. The following subsections provide more information about these endpoints, detailing and explaining one example of an endpoint for better comprehension. In order to avoid repetitive content, the other endpoints were explained in APPENDIX B – Endpoint Diagram, that shows every API flow in a different diagram.

Table 1. Endpoints created and their functionalities.

Method	Endpoint	Description
POST	Client	Create a new client in the factory to enable production
	Items	Create a new Item in an existing Order
	KnowledgeRoute	Create a new production route (set of stations to follow) to enable fabrication of a new PN
	Order	Create a new Order of Production requested from a client
GET	Clients	Retrieves client stored information by username
	Items	Retrieves all Item data by passing idOrder or idClient
	KnowledgeRoute	Retrieve set of station from a particular PN
	Order	Retrieve Order(s) from a particular Client
	NextStation	Search the next station of an Item by its PN, passed station and set of stations
DELETE	Client	Delete client's data and make it unable to produce in the factory
	Items	Delete production Item from an Order or Client
	KnowledgeRoute	Delete a knowledge route from a PN
	Order	Delete a production Order (Don't automatically delete the Items bellow)
PUT	Client	Update client's informed data
	Items	Update Item information, changing the PN and the final good the Item will become
	KnowledgeRoute	Update the set of Stations an Item will follow

5.1.1. Functional Interface – Client Endpoints

The communication between client and factory is done by the integration with the Client Endpoints. The Client Endpoints allow the client to Create, Read, Update and Delete (CRUD) information from the factory of Order, Items, Client profile information and Knowledge Route to help to cross order information.

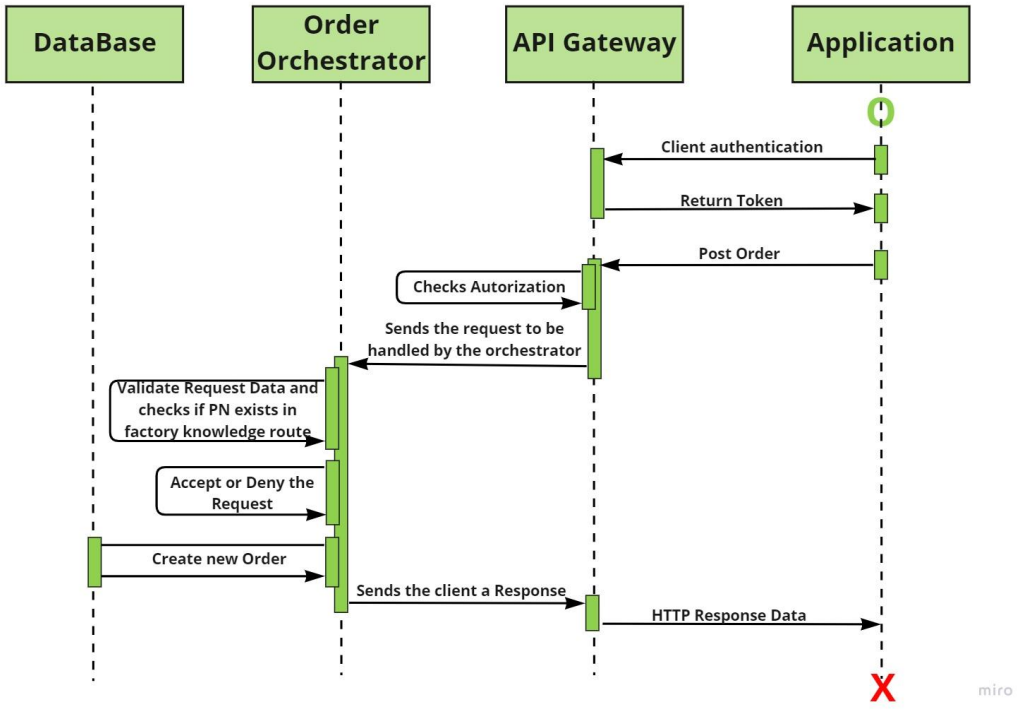
In order to explain one case, in Figure 14 the state diagram for Post Order can be seen.

Every time the API is called, it is checked for authentication to validate permission and the access token is returned; then the Order Orchestrator validates if a knowledge Route exists for this particular PN to validate if the factory is all set up to manufacture

the product. Then after the success order creating in the DB, a message is returned to show the client that the process was successful.

The permission to call the API in the client-side is different from the permission to call from the factory-side. While the client can see and change only their own orders, and have limited access in terms of functionalities, the factory token scope have complete access to change production routes, such as to modify knowledge route for PN. The factory has to manage a way to never show data from one client to another, in order to avoid conflict of interests and compliance matters between clients. And to address that, all that needs to be done is a token parametrization of access and information API access scope permissions.

Figure 14. State Diagram of Post Order endpoint.



Source: Source: Own Authorship

5.1.2. Functional Interface – Factory Endpoints

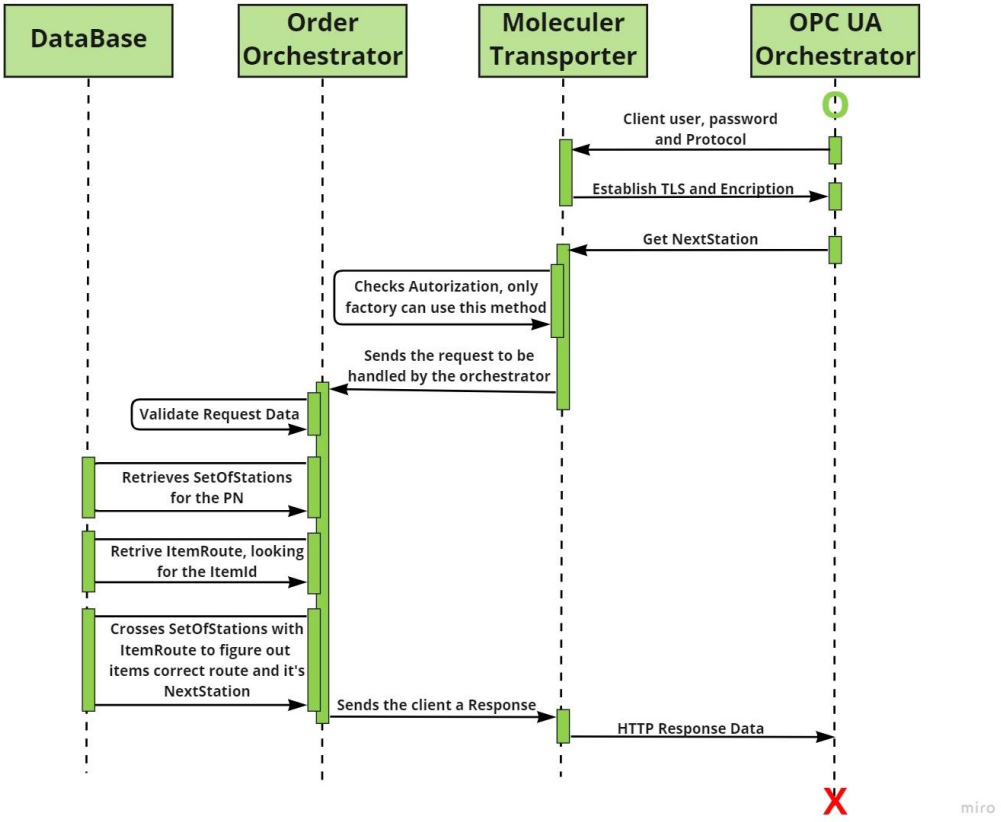
The Factory Endpoints are mostly related to PN, Client, Knowledge Route setup (setup means a way to prepare the system to run by itself, something that is done only once to prepare to run automatically) in the factory. This is one of the APIs most precious to the correct function of the system, because it validates if an Item that just arrived in a workstation in the factory is in the correct production route and to which station it should go next to continue its production path. This API is the Get NextStation.

In

Figure 15 the state diagram for Get NextStation can be seen. For the reader to understand the function of it: first, the authentication is always checked; second, a series of validation is made to answer three questions:

1. If the Item is on the correct track;
2. What must be done to that Item in current Station;
3. Where must the Item go next.

Figure 15. State Diagram of Get Next Station endpoint.



Source: Source: Own Authorship

To answer, the system is searched the SetOfStations in the KnowledgeRoute collection, the correct route for that product (or PN), then it is retrieved Items path (from SetOfStations) of already passed station for the item, using the history-time collection ItemRoute (from KnowledgeRoute), searching ItemId.

With these two queries, it is possible to cross information to answer the three questions:

1. If the station in place is in the Set of Stations, and the item is passing by the first time, the item is in the correct route, if not, item is in the wrong station and should be direct to the correct one;
2. Searching in the Knowledge Route, the information of what needs to be done is this station is collected and returned to the stations;
3. Searching in the Knowledge Route the item can see in the sequence where it is and which is the next station to go to.

5.2. Business Layer

The Business layer defines the factory documented business rules, the ecosystem of production environment, and how a client can connect and gain production collaboration with the factory. In the context of the RAMI 4.0, the Business layer covers abstract business models and process logic, as well as ensure the integrity of functions along the value chain. In this research the Business layer explain how any client will interact with the factory, it ensures the integrity of functions in the business, enables mapping business models and the resulting of the overall process of the production ecosystem.

The API GW of this research is sited in the intersection of Functional and Business layers, but considering that the GW is a connectivity method, it can be interpreted as closer to Functional layer. The API GW and the swagger is a valuable part of the development of this research, and what the Business layer of RAMI 4.0 must deliver: a set of documented and clear integration methods that are flexible to the client decides how to interact with the factory in order to get the best result from the integration.

That means there are uncountable ways that one client to interact with the factory, and it has to be flexible, because each client has their business internal characteristics and the integration must adaptable to its business. This is what the swagger and the APIs achieve in this work.

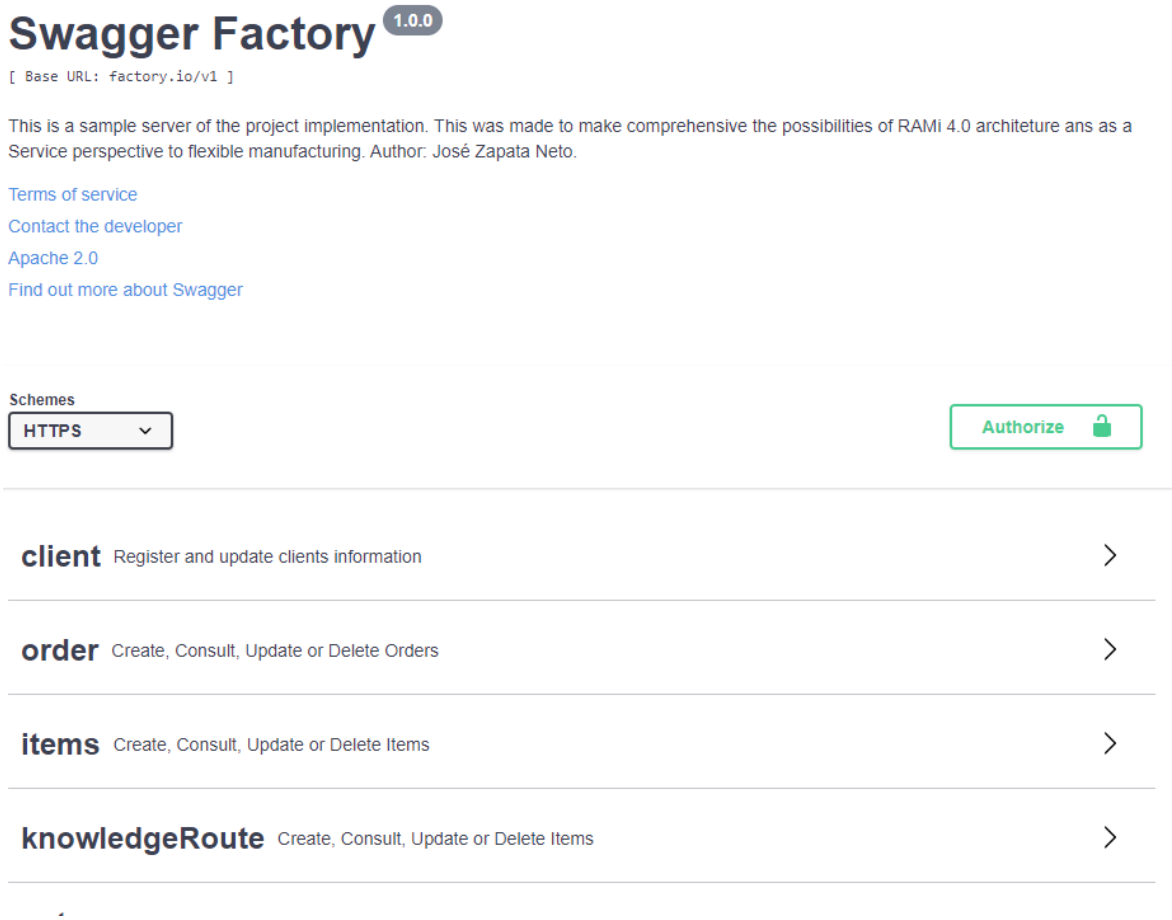
The clients are connected to the factory using the API GW and they are led to do that by a set of business rules explained in the Swagger. Swagger can be used to automatically generate an Open API document based on the code itself. This embeds the API description in the source code of a project and is informally called code-first or bottom-up API development. The Swagger has its business rules explained in a more

business related and less technical but still technically reliable, so the client can understand how to efficiently integrate with the APIs. It is crucial that a client has a clear swagger to integrate with factory and the technical team needs to understand how to get functional integration to find how is the best way to integrate and have information going back and forth to make the factory a part of their business, like a software, or platform they hire.

5.2.1. Business Interface – API Gateway Swagger

This work was not published on the Internet but it is running locally on the computer local host. But for purpose of demonstration was crated a mocked swagger shown in Figure 16, to give the reader a clear view of what the developer of the factory client would seem to understand how the factory works and how to talk to it. The figure has a definition of all external endpoints created. As explained with all details on APPENDIX B, some endpoints are external and orchestrated for the client -side, and some are internal (factory-side), orchestrated for the factory to change internal production rules.

Figure 16. Factory Swagger.



Source: Own Authorship.

The swagger defines the Client, Items, KnowledgeRoute and Order endpoints, showing what the client can and cannot do. Every box can be clicked to show details; for example, if the client API is clicked, it will open more details, as shown on in Figure 17.

A way to go even further on business potentiality is to create another page focused on external business rules. It would contain a set of documentation focused on training clients to talk with the factory, focused on product flows and linked with the swagger to demonstrate in a more high-level way, instead of a developer interpretation of business-to-business integration.

This set of endpoints allows the factory control to setup a new client in the factory production line. This can be done by the methods POST, PUT, GET and DELETE; some examples are described in Figure 17.

Figure 17. Factory swagger, CRUD Client Methods.

Source: Own Authorship.

The methods are quite similar, variating on the CRUD (Create, Read, Update and Delete) methods. The overall parameters for methods, and using POST client as an example, are:

POST`/client`
Body:

```
{
            "_id",
            "clientUsername",
            "password",
            "clientName",
            "phone",
            "statusClient",
            "createdAt",
            "updatedAt",

}
```

The PUT method can update all fields but clientUsername and statusClient, as the first is the index field for that company, and status field can only be managed by the factory, not by the client as this status is used to know if the client is ok or not ok to place orders and fabricate items. Client can have a bill overdue for example and be blocked by the factory to produce items.

The GET and DELETE methods are quite straightforward, being used to consult and delete client data.

Post Order is used to create a production order in the factory, to call use this API contract:

POST/order
Body:

```
{
            "_id",
            "clientUsername",
            "createdAt",
            "updatedAt",

}
```

The PUT method doesn't exist; this is a business rule to not change the order. Orders are created for a PN exclusive or for one type of product, so changing, does not make sense. The GET and DELETE methods are quite straightforward, being used to consult and delete client data. It is important to considerer that *on_production* may not be deleted, but order does not have a status, order reflects the item status.

Post item is used to create a production item in the factory, can be orchestrated by the order or not, to call use this API contract:

POST/item
Body:

```
{
            "_id",
            "idOrder",
            "partNumber",
            "EPC",
            "statusItem",
            "createdAt",
            "updatedAt",

}
```

The PUT method can update item to be a different PN, changing the item to a new one and this can only be done if there is a match on production and knowledge route.

The GET and DELETE methods are very simple to understand, they can be used to consult and delete item if the production has not started.

Post Knowledge Route is used to create a new route of production in the factory, to call use this API contract:

POST/knowledgeRoute
Body:

```
{
            "_id",
            "partNumber",
            "setOfStations",
            "createdAt",
            "updatedAt",

}
```

The POST, PUT and DELETE method are only used from the factory, client can only use GET method. They are used for insert new Knowledge Routes in the factory, update and delete, to update factory with new items and change production for route when required. Remember that is the instruction to fabricate products in the factory, using a set of stations the item must pass to become a final good. The Information in the DB is given an overview on APPENDIX C.

5.3. Proof of Concept (PoC)

In this section, experimental results will demonstrate how the proposed architecture and its components can work in scenarios of a factory with clients, workstations and products, working in flexible manufacturing. It is complicated to explain this work in a linear way as the system works asynchronously. For the reader to understand: first (5.3.1) it will be presented the scenario defined for the PoC and the scenario of flexible manufacturing that was considered in the factory; second (5.3.1.1), it be explained the systems used to input orders and the factory stations; third (5.3.2) it will go through the order production flow with details in the scenario presented to move to the final discussions and remarks.

5.3.1. Description of PoC and Flexible Manufacturing Scenarios

For this PoC, a scenario was chosen of clothing production factory. It is only an example and do not limit this research, it could be chosen a factory of any product such as shoes, computers, vehicles, any product. In this example, it was chosen to be fabricated 5 different items (products), 2 t-shirts, 2 shirts, and 1 tank top. There are items with and without similarities, in order to explore the possibility of changing items during its production process. This change represents one of the flexibilities in a flexible

manufacturing system. In addition, the products are different in their production, which means each product needs to pass through different workstations in order to be produced correctly.

Also in this example, it was defined 5 workstations in the production line of the factory. The number five is random and it was chosen to not be too big to bring complexity and to not be too small to limit the explanation of how flexible the manufacturing system is. Three (3) of 5 workstations on the production line are responsive for cutting the fabric, one cuts and sew in a way to make a t-shirt, other in to make a shirt, and other in to make a tank top shape. The other 2 workstations are responsible for dye the fabric in the desired color. In this PoC example, it was chosen 2 color possibilities: to dye in red and black colors.

According to the product, the desired item will pass through the workstations in the production line to become the final good it supposed to be. For example: a black tank top first goes to the tank top cut and sew station, them to the black dying station to finish its production process. It is important to explain that the PoC will explore production scenarios in which the items will be changed during the production (ex: a black shirt order will be changed to a red shirt in the middle of production) and items that have not started its production will be canceled. The objective is to show the flexibility that is given to the clients to manage their order using the architecture developed in this research.

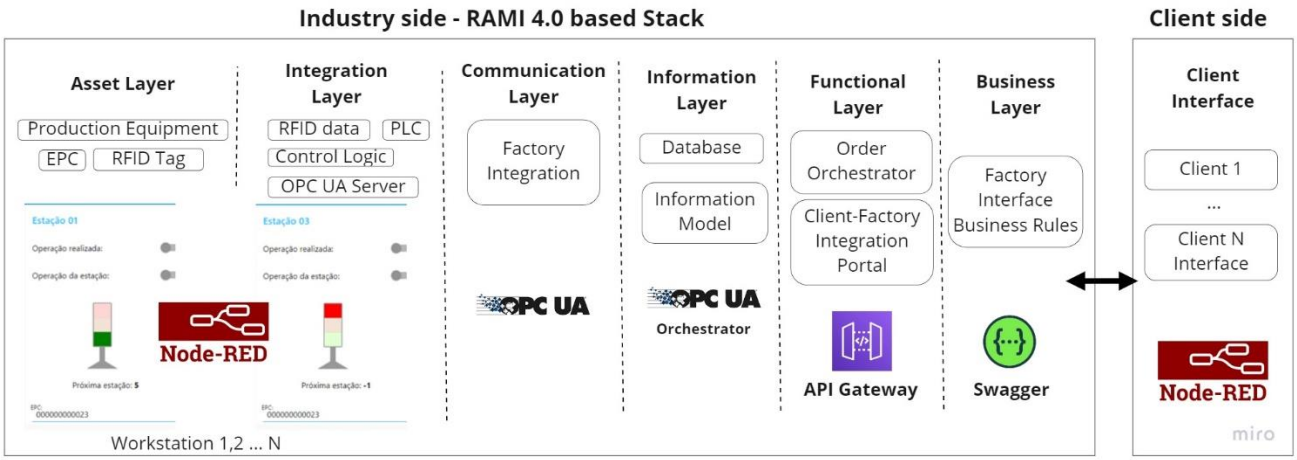
The flexible manufacturing scenario proposed was a factory with a production line where each workstation is responsible for a particular action. So, it is independent on the position/where the production line is, as the product is guided inside the factory to the station it must go to be finished. In linear production lines, there are a sequential linear path of workstations until the item is finished. In this example of flexible manufacturing, there is no linearity because the stations can change in place or actions and the production can be reprogramed to guarantee item integrity to finish its production. Is important to notice that this architecture could also be applied in linear production lines, because in linear production if item is in a workstation that has nothing to do, it would just continue to reach the next one where it should stop.

For this PoC scenario, one client was created in the factory with access to place orders in the production. The client is identified by its e-mail as its username, and it was chosen *jzapataneto@gmail.com* (authors e-mail) only for demonstration. The reader will be able to follow it in the next figures. The username is the key for clients

to place orders, query its information and manage production inside the factory. It will be shown how the client can manage their order inside the factory using the client interface, in which it will be possible to create, search, cancel and change orders.

All the developed tools and systems in this PoC, from the workstation to the client interface, are shown in Figure 18. The industry side represents the factory elements that are allocated in accordance with the RAMI 4.0 layers of the developed architecture. The client side represents the external world, in which the client will interact with the factory to put and manage its production orders. It is really important to reinforce that this PoC is just one example of a flexible manufacturing system that microservice architecture developed in this research could be applied. There is an infinite implementation possibility, depending on the client, the production line and workstations, the products and order specifications.

Figure 18. Proof of Concept Scenario: Details of Technologies and Components Interactions for the Microservice Architecture based on RAMI layers for Industry 4.0.



Source**:** Own Authorship

From the Asset layer to the client side represented in Figure 18, the components of the architecture can be resumed as:

- Asset layer: it is related to the production equipment (machinery to produce clothes in this PoC), the RFID tag in the item and the EPC (Electronic Product Code) that contains the product specification code. In this PoC, it is represented by the workstations (detailed in the next section), which are composed by the RFID tags (recorded with the EPC inside them), the RFID antenna, the

conveyors, stack lights and sensors. The 5 workstations were created using Node-Red programming language;

- Integration layer: it is related to the control equipment (Ex: PLC) and control logic of the production equipment, the RFID data and the communication support. In this PoC, it is also represented by the workstations, containing the EPC code (RFID data) and the communication network OPC UA Server. It is also included in the Node-Red programming of the workstation;
- Communication layer: the RAMI 4.0 recommends the OPC UA protocol for the communication network. This layer is responsible to link factory data to Information layer (ex: database), providing the interface between the factory (low level) and the production management (Functional layer);
- Information layer: contains the information modeling and the system database. It is a crucial part for system efficiency and reliability of information. In this PoC, it was chosen the MongoDB and was created the OPC UA Orchestrator (Client). The OPC UA Orchestrator is an OPC Client that connects to the OPC UA of each workstation for the network integration;
- Functional Layer: it is composed by the Order Orchestrator that interacts with both (client and factory) sides for decision making about the production management of orders done by clients. Also, it provides the client integration portal using the API GW to open factory information to the Internet, receiving and sending information with security through APIs. The system was created using Node.js programming language;
- Business layer: it is composed by the factory business rules to orient clients on how to integrate with the factory, put orders and manage them. All the knowledge was put together in the Swagger;
- Client Interface: it shows an example of how a client could build an interface using the Swagger orientation and integrating with the API GW to place orders, search, update and delete orders. In this PoC it only represents one client, but the architecture would support several clients. The interface was created using Node-Red programming language.

Next section sub-sections will present in details the workstation and client interface usage, the order procedure and the production workflow in the stations of the

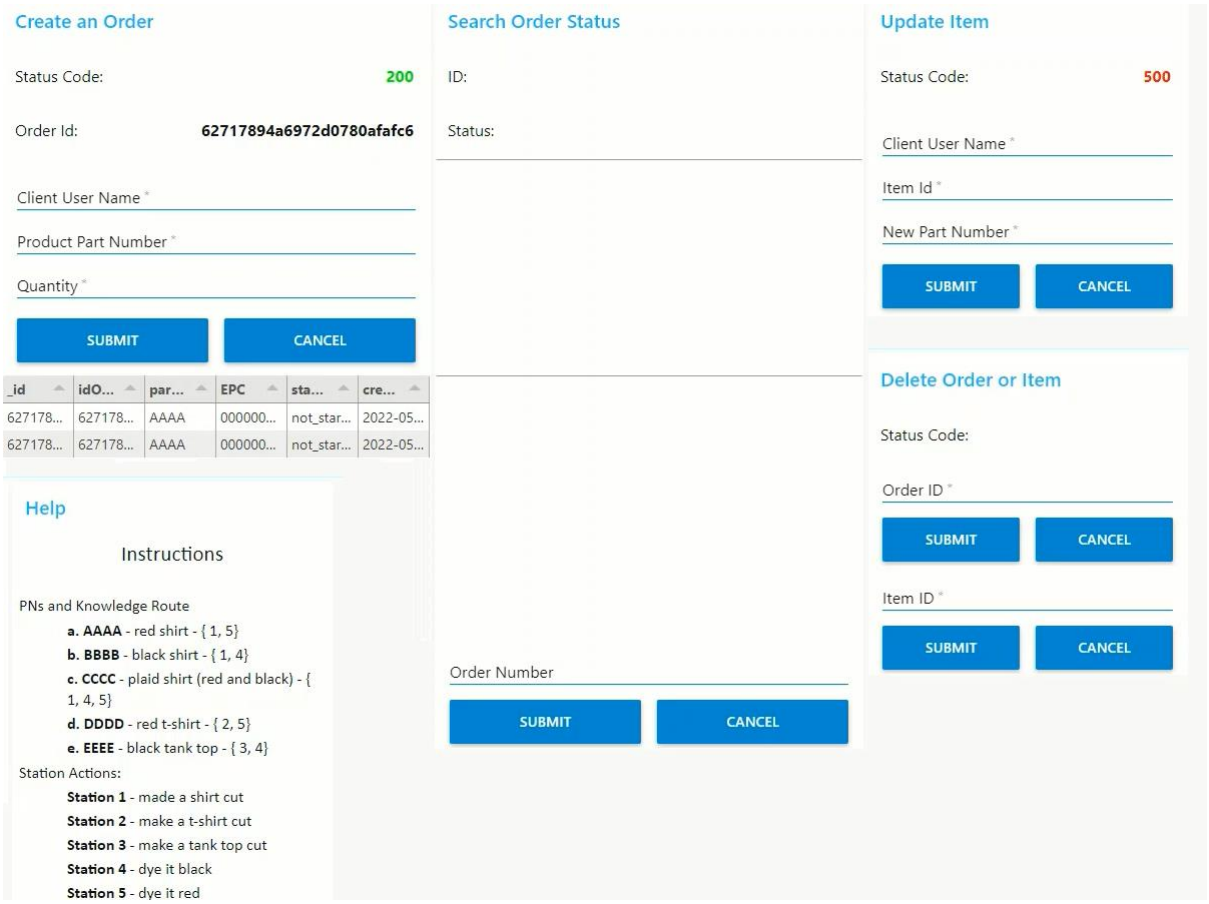
PoC factory.

5.3.1.1. Description of the Client Interface and Factory Workstations

In order to be able to have a proper scenario of a flexible manufacturing system for testing the architecture and the components developed in this research, a client interface and a standard workstation have been created in this PoC. The client interface resumes the client interaction to the factory focused in its order. The workstation resumes the all elements necessary for producing an item in the production line.

Figure 19 shows the Client Interface created for the PoC and show in a more interactive way how the client can interact with the factory.

Figure 19. Overall view of Client Interface developed for this research (modified to suit the screen).



Source: Own Authorship.

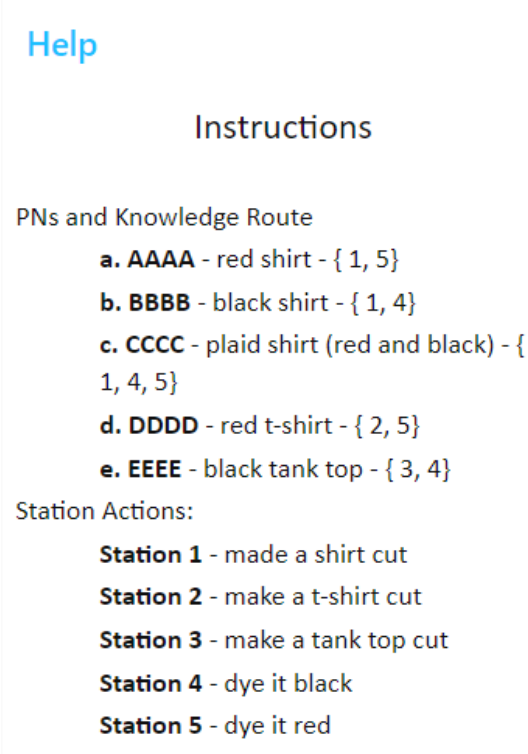
Every important part in Figure 19 will be explained in details. This interface was created to demonstrate how the client can:

- place new orders screen: using Client User Name (*e-mail*), Product Part Number (the final good they want to produce) and quantity of items to be produced, this will place a new order for a client of a specific PN with the requested number of items to be fabricated. After requesting a new order, the Order id is returned to the client consult the production details. This part can see in details in Figure 22;
- search for order in production screen: by passing the Order id. It will be searched the Order and Items associated to that order in production details. It can be seen in more details on Figure 23;
- update items screen: by passing the Client User Name (*e-mail*), the item id and the new PN, the system will make a try to change the item from one PN to another (changing a PN is an effort to change this item from a final good to another). This allows the client to change its order and try to best suit the market. The system will calculate based in stage of production if the can change or not, and return a success or error. Normally an order has only one type of PN, but if the client changes it will have 2 or more. The details of this screen are shown in Figure 24;
- delete items screen: The client can delete an Item or Order in production. It is recommended to do that before production to starts, but the client can do during the production if they don't want this production anymore. The item can be deleted using item id, and order by order id. The details of the screen are shown in Figure 25.

Some instructions were given in the Client Interface, only to help the user to understand which Part Number and knowledgeRoutes are ready to use in the factory. It is shown in the right of Figure 19, and in details in Figure 20. It is a summary of everything it was explained above and, it can be seen in detail in to have like a quick access place to remember what everything does and are.

Now that the reader understood the client interface of this research, or the client side of interaction with the architecture, the author will explain the factory-side of interaction. As clients place orders, search, update and delete, and it is all manage by the architecture: the factory's workstations (production and control equipment and RFID data) are all sending data and receiving instructions from the Functional layer.

Figure 20. Instructions set on Client Interface for quick access.



Source: Own Authorship.

In order to represent this interaction and for the PoC purpose, a set of 5 stations were created, as shown in Figure 21. Each station has represented in Figure 21 has:

- a text field called EPC to be used to insert the EPC number for simulation the RFID antenna reading a tag;
- a stack light (with green, yellow and red from the bottom to the top respectively), to show when the item is in correct route (green), wrong route (red) and when the station is waiting (yellow);
- two buttons (“Operação Realizada” “Operação Estação”) used only for the reset on the system.
- the “Proxima Estação” field for system to return that means Next Station field where the system returns the next station the item should follow; returns 0 if it is its items last station, -1 if it is a route error but the EPC exists in the base, and -2 if the EPC does not exist in the DB.

Figure 21. Stations Dashboard, simulating factory production.

Source**:** Own Authorship

In the next sub-section, some scenarios of production will be explored. An example of production will be given for better comprehension of production scenarios of this research PoC.

5.3.2. Production Verification

Considering the PoC scenario created, an order creation by a client and its production process will be described and verified. This production example will help understanding the whole architecture operation. For this PoC scenario, it will be created an order with PN AAAA. First the order is created with 2 Items, them it will be shown the searching using the Order id, to explain how the Status of production are displayed. And them, one of the 2 items of the order will be changed from PN AAAA to BBBB, demonstrating how it is possible to change an order in the system. Finally, the Order and the Item will be deleted, this example will demonstrate all the functionalities

of the client-business interface in an example.

More exploratory scenarios using other PN will be shown after. Just a quickly reminder on system knowledge routes that is performed at each station:

- Station 1: cuts the fabric and sews it into a shirt;
- Station 2: cuts the fabric and sews it into a t-shirt;
- Station 3: cuts the fabric and sews it into a tank top;
- Station 4: dyeing the fabric in black;
- Station 5: dyeing the fabric in red.

The PN chosen as example, are manufactured according to PN and the route of stations needed to finish production:

- PN: AAAA, produces: red shirt, and must pass through station 1 and 5;
- PN: BBBB, produces: black shirt, must go through station 1 and 4;
- PN: CCCC, produces: red and black plaid shirt, must go through station 1, 4 and 5;
- PN: DDDD, produces: red t-shirt, must go through station 2 and 5;
- PN: EEEE, produces: black tank top, goes through station 3 and 4.

Starting with PN AAAA, it will be created containing 2 items (Quantity =2). Observe in Figure 22 that after inserting client username, PN and Quantity and clicking on submit, and order is created and it is possible to see:

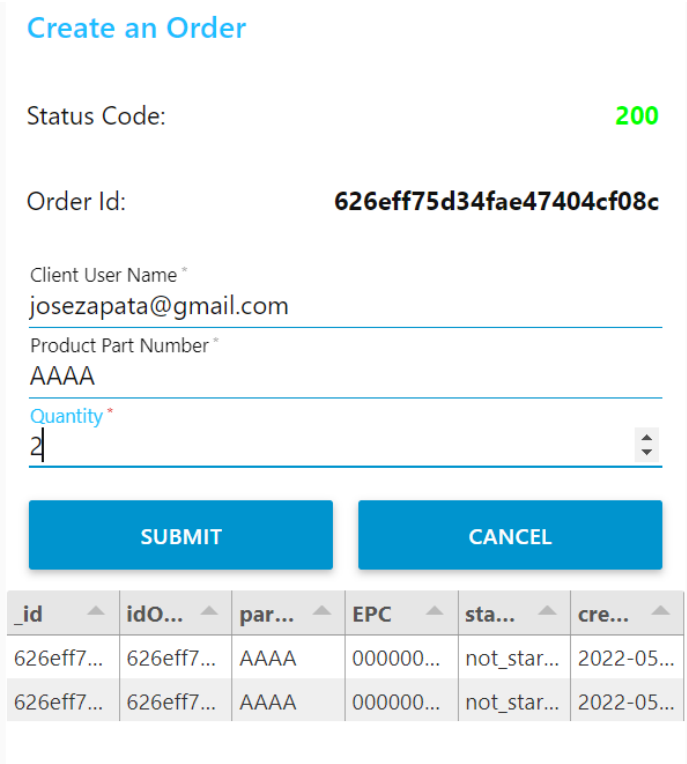
- Status 200 hundred, which means success;
- Order Id for our control,
- A table of items created containing _id, Id_Order, partNumber, EPC, statusItem, and createdAt to give production details;

Also, let’s explore some error scenarios developed that the client may receive if does something wrong:

- If a client User Name is not found in the clients DB, and all other fields are correct, an Error 500 will occur, showing that clients without an account cannot produce;

- If client Username and quantity are correct but is passed a Product PN that the factory doesn't have a knowledge Route will appear an Error 422, showing that the factory cannot produce an item that factory is not ready to produce;
- If is a correct client username and Product PN but user tries to insert a Quantity equal or below zero, the system just ignores the request.

Figure 22. Create Order on Client Interface.



Source: Own Authorship.

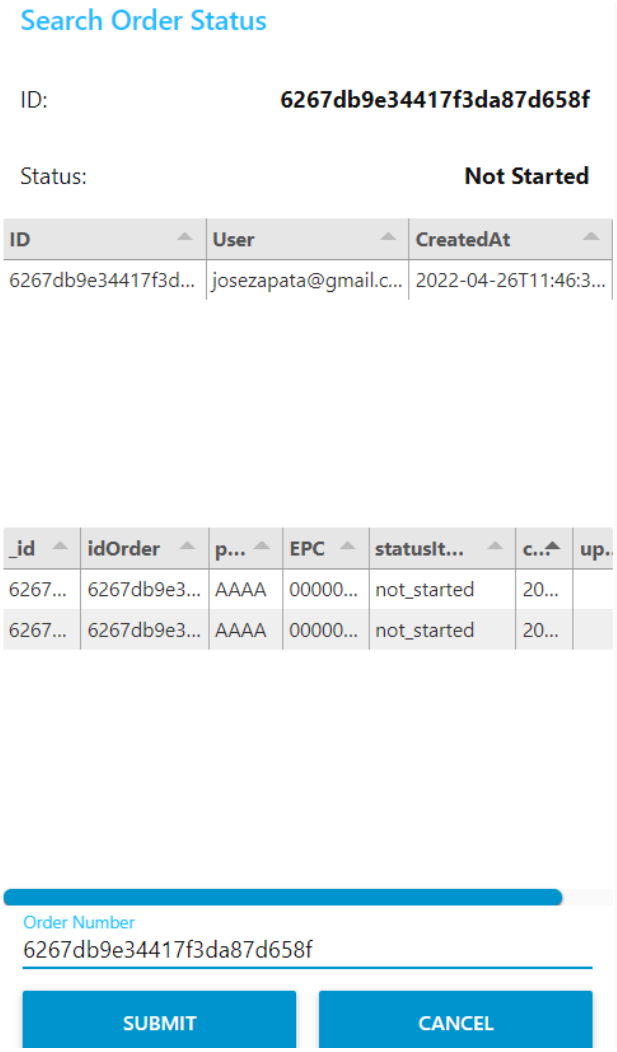
If the client search Order information just after order is created, it will notice that the Order status is "Not Started", also the Items will have the same status, as it is shown in Figure 23. For this screen the only input is the Order Number, and the outputs are Order Status, Order Id, User, CreatedAt, and for Item details are: _id, Id_Order, partNumber, EPC, statusItem, and createdAt to give item details. The screen does not treat errors; if a wrong Order id is inserted, it is just ignored.

The Order Status follows this logic to set Status in Not Started, In Production, or Finished:

- If there are no items in the order started to be manufactured, the items are in not_started and the order is Not Started;
- If an item went through any station, only that item changes to In Production, and the order also changes to In Production as well.

- If an item has finished its production, only that item changes to Finished, and the order continues in In Production until all items are finished production.
- If all items are finished, the order changes to Finished.

Figure 23. Search Order on Client Interface.



Source: Own Authorship.

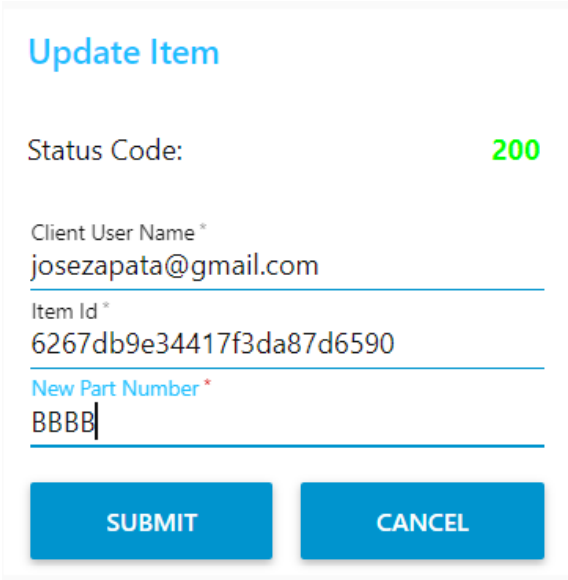
Now, to demonstrate how to change an Item and make it become a different final good, it is possible to update the item knowledge route. For example, in Figure 24, one of the items of the Order of PN AAAA (Station 1 and 5) will be transformed in PN BBBB (Stations 1 and 4), which means, instead of making a red shirt, a black shirt is made. It is possible if:

- Item has not start production
- Items Knowledge Route and current station have similarities to enable the change, for example, if the Item has passed for station 1 which is similar from

PN AAAA and BBBB, the system will enable because the change does not harm the final Item.

If the Items has no similarities, the system will block the change to not cause harm in the Item; in this case the system blocks and returns and Error 500. This allows quick client factory detailed interaction and it is the core delivery of this research, without losing production governance.

Figure 24. Update Order on Client Interface.



Source: Own Authorship.

If the client lost interest in production, Item and Orders can be deleted, as shown in Figure 25. To do so, the client just inserts the Order Id and submit, or insert the Item Id and submit. This helps the client to reduce order quantity if needed.

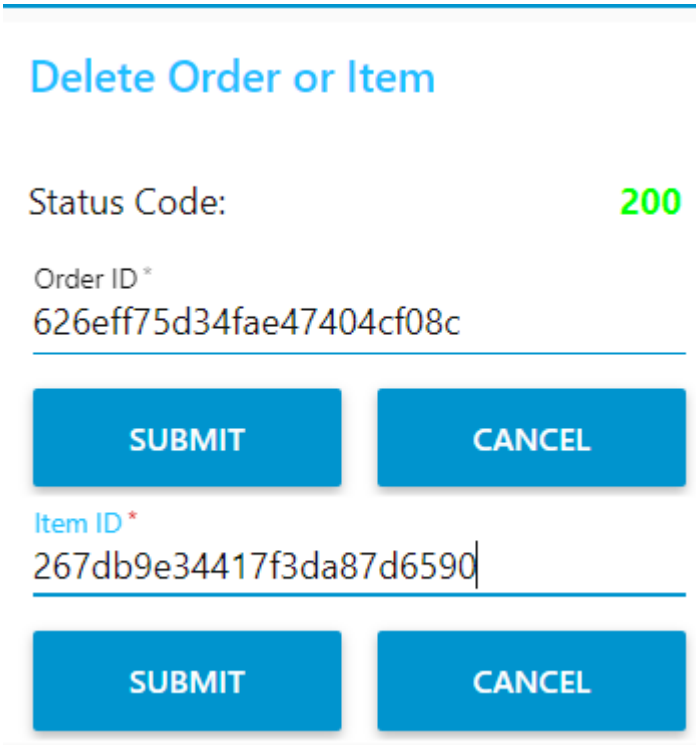
Using the example explored above it is possible to see all system functionalities in one example, but to explore more scenarios it was created and example with 3 PN, producing in all 5 Stations. It was created an order with 2 items for each PN (AAAA, BBBB, and EEEE);

In the first example with PN AAAA, it was created and order with 2 items to demonstrate a scenario of normal production. The Order at start has status of Not Started, and their items as well, so after the first Item of the Order passes to Station 1, the Item Status is changed to In Production, and after that Order Id is searched and can be seen that Order Status is in production as well.

Them, the first item is passed on station 5, the last station, and their status is

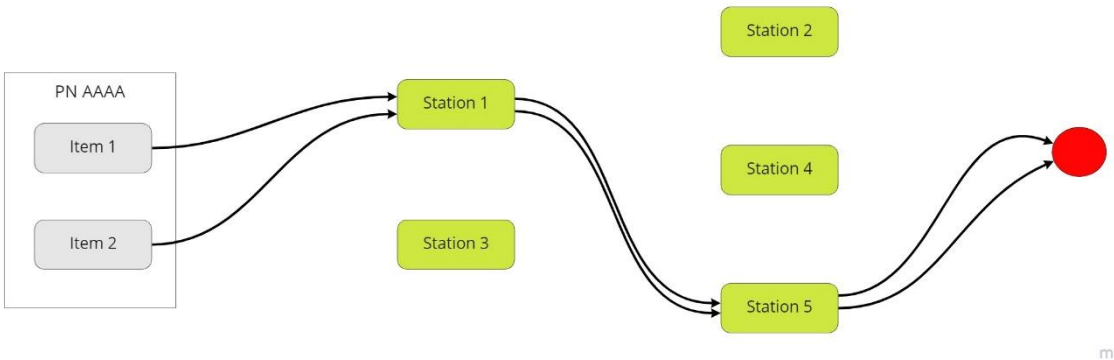
changed to Finished, but order Id keeps in Production because the other Items hasn't finished. So, the second item passed in station 1, is changed to In Production, passes on 5 and is Finished, and now can be seen, that the Order is also changed to Finished. This is the more common and happy scenario, but very important to validate. The flow is also shown in Figure 26, and detailed in the video available at the end of the session.

Figure 25. Delete Order on Client Interface.



Source: Own Authorship.

Figure 26. Order production of 2 items and PN AAAA.



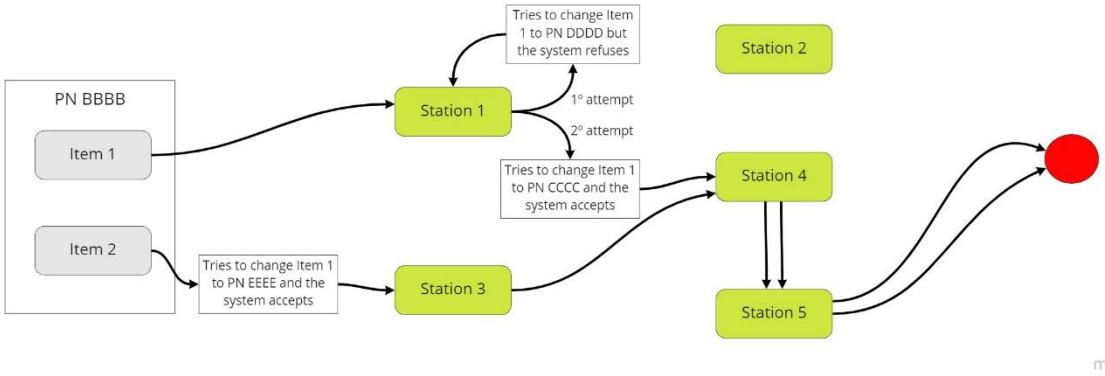
Source: Own Authorship.

In the second example, with PN BBBB, will be explored an attempt to change items from one PN to another. So, the order also has 2 items and the first item have passed on station 1, and the second hasn't started. First it tries to change the first item to PN DDDD, but the system does not allow and gives an error, because production route for PN DDDD is 2 and 5, and this item have already passed on 1.

So, it is made another attempt to change to PN CCCC which production route is 1, 4 and 5, and the system returns success because even the item has passed in station 1, for PN BBBB and CCCC station 1 is common and the change means no harm for the production. For the second item that have not passed by any station, when it is tried to change to PN DDDD it returns a success because the item has not started production, showing that in this early stage, any change is possible.

This example shows that a client that regrets change order from one good to another. The flow is also shown in Figure 27, and detailed in the video available at the end of the session.

Figure 27. Order production of 2 items and PN BBBB.



Source: Own Authorship.

On the third example, it will be explored a scenario where the client have made a very large order but wants to cancel partially or totally by any reasons they might have, so using PN EEEE, with 2 items, the client tries on the deletion screen and delete the first item that has not start production, them deletes the second and them deletes the order that was in Status Not Started, showing that system allows the client have flexibility to change orders and Items as they wish when factory authorizes.

In order to consider the explanation of the 3 examples above in a very complete vision of the developed architecture and its components, as well as the PoC scenarios, two videos were made:

Portuguese:
https://drive.google.com/file/d/1PJvMrNu8z3IkG5dJPs1bWfpVj_26zkCE/view?usp=sharing
English:
https://drive.google.com/file/d/1tAo6bW7kEUqogtgiQA_gdokW5khTxpXm/view?usp=sharing

5.4. Final Discussions and Remarks

The results shown in the previous section make possible to understand how flexible manufacturing production lines can make use the microservice architecture based on RAMI 4.0 layers developed in this research. The architecture was able to deliver a simple way to manage flexible producing lines, using the concept of Part Number, Serial Number and Production Route, covered by a service-based controlling to manage the production. As demonstrated, the architecture fulfils the requirements for flexible manufacturing. Orders can be changed during production; the items could change from one to another and the production route could change without warm the production. Another point of flexibility of the developed architecture is that the workstation's physical locations and Items movement can change without affecting the factory. If more workstations are added or subtracted from the production line in order to produce more or less different products, the knowledge route in the database can be updated to adequate production. In a nutshell, all of that shows the concepts of flexible manufacturing addressed in this research.

The architecture delivers a very stretchy and efficient way to produce in the factory, and also, delivering value to the factory clients, that using this feature can respond quickly to market changes and easily make modifications on items and products. It is very simple to adapt the system to a lot of production environment, production lines and different scenarios, due to the use of microservices architecture.

A point of disadvantage of this proposal is that the control is not decentralized, which can harm the production in case of interruption as all systems are depending on the unique point of control. This a very common problem when data reliability is crucial for the system to work. A possible way to continue this work would be explore a way to use blockchain and consensus mechanisms to decentralize the DB, like finance, energy and retail institutions are doing.

It may be observed in this research that it uses of all layers of RAMI 4.0, which is rarely identified in the literature related to this research. RAMI 4.0 allowed a way of structuring the organization of the solution in an organized and comprehensible way, facilitating the understanding of roles and responsibilities of each system, software and asset in a RAMI 4.0 layer inside the factory solution in the context of Industry 4.0. A disadvantage of RAMI 4.0 is that, being an abstract model, it causes a great divergence of interpretations, mainly in the Business layer, what makes it more challenging to converge and compare with other researches. The author believes that Business layer cannot be easily explained and defined, as it can be done in the Functional Layer. The Business layer in authors view is more of an ecosystem on how other systems can be perceived and quickly interact with others when opened to network. It is how it is inserted, how easily synergy can be gain with other software and systems when they are connected.

When it is looked into the architecture and software used in this research, it can be noticed that the choice of using SOA promoted a more scalable way to deploy the software, making it organize and independent pieces of software. But on the other hand, it was a lot more development to build the ground basis of the software when compared to a monolith, what can be a disadvantage if time of development is concise.

The combination of the Order Orchestrator and API Gateway provided the PaaS and MaaS concepts discussed in 2.1.2 (Industry 4.0 Potentiality), enabling an effective utilization of manufacturing resources for rapid realization of custom production needs of product developers as discussed by Balka and Lin et al. (2018). Also, it also enabled a higher degree of utilization of installed production capacity, stations can be reused to create new production lines to increase production, raising the return on investment, as explained by Annunziata (2019). All that combined brings a Product as a Service perspective for Production and Manufacturing, opening space to transform the way that factories and production systems are perceived.

With all of that said, this proposal of architecture was successfully implemented. All the demonstration and scenarios show that the control of the factory can be delivered to the client, without losing production governance and process, allowing efficiency, scalability for flexible manufactory in context of Industry 4.0.

6. CONCLUSIONS

This research presented a full microservice architecture to engage all six IT layers of RAMI 4.0 showing all cycles of production system combined for a service-based production end-to-end perspective but focusing on Functional and Business layers. A platform 100% API-based was able to bring a product-oriented perspective to a place that is client-oriented; also, touchless connectivity to new clients and business opportunities. When the exterior is modern API based, the interior is ruled by OPC Unified Architecture and Moleculer microservice Framework, bringing factory control and organization together to help the factory react to market changes.

The research has delivery an API Gateway and Order Orchestrator in the Functional layer of RAMI 4.0, an explained swagger in the Business layer, and an interpretation of a Client Business Interface. The Order Orchestrator was able to maintain control from both factory and client side, intermediating all communications talking to DB, to return reliable and quick data. API GW delivered a safe, organize, scalable way to connect the factory to the world, using state-of-art technology, and the Swagger gave the Business layer structured comprehensive documentation to improve client speed for integration and help them get the most of the factory services.

In this example for Proof of Concept, Order and Items were created, had their production Status searched, were deleted, and, most important, Items were converted from one PN to another, showing quick client-factory interaction and adaptation, allowing the client to respond quickly from market changes without the factory losing production governance, an important advantage to the client and a plus to the factory. In a market that is changing so fast, production has to be able to adapt, producing the items that the client needs very fast. Factory must not force the client to have orders requested in a different market time, that’s why this service is one of the biggest contributions of this research in relation to flexible manufactory where production line could be changes without affecting factory production. Side by side, the Order Orchestrator, Swagger, and API GW from top layers of RAMI 4.0 allows business-to-business simple, adaptable and scalable connectivity to clients.

This research is sited in a place of literature that does not have too many contributions. There are very few projects applied in the top layers of RAMI 4.0, so this project was able to delivery one of few implementations of RAMI 4.0. It also shows a

full case and adaptability to work on flexible manufacturing and full set of OPC UA orchestration to have the factory as a production platform that can be orchestrated on client desires to produce for one or uncountable clients, and one or uncountable items, using the microservices and APIs.

5. Referências

ABBOTT, D. **Applied predictive analytics:** principles and techniques for the professional data analyst. 1. ed. [S.l.]: John Wiley & Sons, v. 1, 2014. 427 p.

ADOLPHS, P. et al. Reference Architecture Model Industrie 4.0, 01 April 2015. Disponivel em: <https://www.zvei.org/en/press-media/publications/the-reference-architectural-model-industrie-40-rami-40/>. Acesso em: 07 October 2020.

ALMADA-LOBO, F. The Industry 4.0 revolution and the future of Manufacturing Execution Systems (MES). **Journal of Innovation Management**, v. 3, n. 4, p. 16-21, 2016.

ANNUNZIATA, M. Manufacturing-As-A-Service Platforms: The New Efficiency Revolution. **Forbes Magazine**, 13 may 2019. Disponivel em: <https://www.forbes.com/sites/marcoannunziata/2019/05/13/manufacturing-as-a-service-platforms-the-new-efficiency-revolution/?sh=68f34dc357fd>. Acesso em: 17 March 2021.

AWS. Amazon API Gateway, 2021. Disponivel em: <https://aws.amazon.com/pt/api-gateway/>. Acesso em: 18 March 2021.

BALKA, E. et al. Production as a Service: A Digital Manufacturing Framework for Optimizing Utilization. **IEEE Transactions on Automation Science and Engineering**, v. 15, n. 4, p. 1483-1493, 27 jun. 2018. Disponivel em: <https://ieeexplore.ieee.org/document/8398543>.

BASSI, L. **Industry 4.0:** hope, hype or revolution? Modena, Italy: IEEE. set. 2017. p. 11-13.

BAUR, C.; WEE, D. Manufacturing's next act. **Mckinsey Company**, 1 June 2015. Disponivel em: <https://www.mckinsey.com/business-functions/operations/our-insights/manufacturings-next-act#>. Acesso em: 07 October 2020.

BBC. Will Robots Take Our Jobs? **Youtube**, 26 march 2018. Disponivel em: <https://www.youtube.com/watch?v=nSjLzte1_ps&feature=youtu.be&ab_channel=BBCClick>. Acesso em: 7 October 2020.

BIGHETI, J. A. **Arquitetura de Automação e Controle Orientada a Microserviços para a Indústria 4.0**. Universidade Estadual Paulista "Julio de Mesquita Filho" Sorocaba, Brasil, Tese de Doutorado, Programa de Engenharia Elétrica (FEB), p. 146. 2020.

BRAJESH, D. **API Management:** An Architect's Guide to Developing and Managing APIs for Your Organization. 1. ed. Bangalore, India: Springer, 2017.

CANTELON, M. et al. **NodeJs Action Manning**. 1. ed. Nova Iorque: Manning Publications Co, 2013.

CGI. MES Product Survey. **MESA International**, v. 16, n. 1, p. 1-768, 2015.

CNET NEWS -. Meet the robots making Amazon even faster. **Youtube**, 30 November 2014. Disponivel em: <https://www.youtube.com/watch?v=UtBa9yVZBJM&feature=youtu.be&ab_channel=CNET>. Acesso em: 07 October 2020.

CONTRENAS, J. D.; GARCIA, J. I.; PASTRANA, J. D. Developing of Industry 4.0 Applications. **International Journal of Online Engineering (iJOE)**, 13, n. 10, 07 nov. 2017. 30-47. Disponivel em: <https://online-journals.org/index.php/i-joe/article/view/7331>. Acesso em: 29 set. 2022.

DELSING, J. Local Cloud Internet of Things Automation: Technology and Business Model Features of Distributed Internet of Things Automation Solutions. **IEEE Industrial Electronics Magazine, Institute of Electrical and Electronics Engineers Inc**, v. 11, n. 4, p. 8-21, 2017.

DOCKER WEBSITE. Develop faster. Run anywhere., 2022. Disponivel em: <https://www.docker.com/>. Acesso em: 22 set. 2022.

DUSTDAR, S.; PAPAZOGLOU, M. P. Services and Service Composition – An Introduction (Services und Service Komposition – Eine Einführung). **it - Information Technology**, 50, n. 2, 2008. 86-92. Disponivel em: <https://www.degruyter.com/document/doi/10.1524/itit.2008.0468/html>.

ERL, T. **SOA:** Princípios de Design de Serviços. São Paulo: Pearson, 2009.

FATTORI, C. et al. Modeling of manufacturing execution in disperse productive systems using service oriented technique. **IFAC Proceedings Volumes**, 44, n. 1, jan. 2011. 6413–6418.

HARD, E. D. The OAuth 2.0 Authorization Framework, October 2012. Disponivel em: <https://tools.ietf.org/html/rfc6749>. Acesso em: 18 March 2021.

ISA-95.00.03 STANDARD. ANSI/ISA-95.00.03-2013. Enterprise-control system integration - Part 3: Activity models of manufacturing operations management. **International Society of Automation**, 2013.

JAZDI, N. **Cyber physical systems in the context of Industry 4.0**. IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR). Cluj-Napoca, Romania: IEEE. 2014. p. 1-4.

KAGERMANN, H.; RIEMENSPERGER, F. Recommendations for the Strategic Initiative Web-based Services for Business, 2014. Disponivel em: <http://www.acatech.de>. Acesso em: 12 October 2020.

KAGGERMANN, H.; WAHLSTER, W.; HELBIG, J. Recommendations for implementing the strategic initiative INDUSTRIE 4.0. **acatech**, 2013. Disponivel em: <https://en.acatech.de/publication/recommendations-for-implementing-the-strategic-initiative-industrie-4-0-final-report-of-the-industrie-4-0-working-group/>. Acesso em: 26 nov. 2020.

KANNAN, S. et al. **Towards Industry 4.0:** Gap Analysis between Current Automotive MES and Industry Standards using Model-Based Requirement Engineering. IEEE International Conference on Software Architecture Workshops. Gothenburg, Sweden: [s.n.]. 2017.

KOTA, A. K.; PRABHU, D. M. Node.js: Lightweight, Event driven I/O web development. **Informatics**, 2014. Disponivel em: <https://informatics.nic.in/article/287>. Acesso em: 29 set. 2022.

KUK, S. H. et al. An e-Engineering framework based on service-oriented architecture and agent technologies. **Computers in Technology**, v. 59, n. 9, p. 923-935, 2008.

KUMAR, B. V.; P., N.; NG, T. **Implementando SOA Usando Java EE**. Rio de Janeiro: Alta Books, 2012.

LEITÃO, P.; COLOMBO, A. W.; KARNOUSKOS, S. Industrial automation based on cyber-physical systems technologies: Prototype implementations and challenges. **Computers in Industry**, Bragança, Portugal, 81, 04 jun. 2016. 11-25. Disponivel em: <https://www.sciencedirect.com/journal/computers-in-industry/vol/81/suppl/C>.

LI, W.; MITCHELL, C. J. Security Issues in OAuth 2.0 SSO Implementations. **Information Security Group**, p. 529-541, 2014. ISSN doi:10.1007/978-3-319-13257-0_34.

LINS, T.; OLIVEIRA, R. A. R. Cyber-physical production systems retrofitting in context of industry 4.0. **Computers & Industrial Engineering**, Ouro Preto, Brazil, 03 dez. 2019. Disponivel em: <https://www.researchgate.net/publication/337449226_Cyber-Physical_Production_Systems_Retrofitting_in_Context_of_Industry_40>. Acesso em: 29 set. 2022.

LYDON, B. RAMI 4.0 Reference Architectural Model for Industrie 4.0. **ISA**, 2019. Disponivel em: <https://www.isa.org/intech-home/2019/march-april/features/rami-4-0-reference-architectural-model-for-industr>. Acesso em: 7 October 2020.

MARR, B. What is Industry 4.0? Here's A Super Easy Explanation For Anyone. **Forbes Magazine**, 02 September 2018. Disponivel em: <https://www.forbes.com/sites/bernardmarr/2018/09/02/what-is-industry-4-0-heres-a-super-easy-explanation-for-anyone/#4922bf469788>. Acesso em: 2020 September 2020.

MEISSNER, S. et al. Internet of Things Architecture IoT-A Project Deliverable, 2012. Disponivel em: <http://www.meet-iot.eu/caf/?ses=Y3JlPTE2MDI1MjIyMTgmdGNpZD13d3cubWVldC1pb3QuZXU1Zjg0OGM2YTc1MTQ3NS44NTU4NDUyOSZ0YXNrPXNlYXJjaCZkb21haW49bWVldC1pb3QuZXUmbGFuZ3VhZ2U9ZW4mYV9pZD0zJnNlc3Npb249UUFlM2F4QjZYZGlKeXVOM3hSZjY=&query=IOt%20Solution&afdToken=3B1gqTay>. Acesso em: 12 October 2020.

MELO, J. et al. **A systematical approach to expose manufacturing system as a**

service. 9th IEEE/IAS International Conference on Industry Applications - INDUSCON. Sao Paulo, Brazil: IEEE. 2010. p. 1-6.

MELO, J. I. G.; JUQUEIRA, F.; MIYAGI, P. E. Towards Modular and Coordinated Manufacturing Systems Oriented To Services. **Dyna-Colombia**, Medellin, Colombia, v. 77, n. 163, p. 201-210, 2010. Disponivel em: <https://repositorio.usp.br/item/002691840>. Acesso em: 29 set. 2022.

MELO, P. Open Source Control Device for Industry 4.0 Based on RAMI 4.0. **IEEE International Workshop on Metrology for Industry 4.0 and IoT**, 10, n. 7, 01 jun. 2019. 229-234. Disponivel em: <https://repositorio.unesp.br/handle/11449/190616>. Acesso em: 29 set. 2022.

MELO, P. **Dispositivo de Controle para a Indústria 4.0 baseado no RAMI 4.0 e OPC UA**. Faculdade de Engenharia - São João da Boa Vista, Brasil, Tese de Mestrado, Programa de Engenharia Elétrica São João da Boa Vista, p. 127. 2020.

MOLECULER JS. What is Moleculer?, 17 March 2021. Disponivel em: <https://moleculer.services/docs/0.13/index.html>. Acesso em: 17 March 2021.

MONGODB. MongoDB Atlas. Fully managed MongoDB in the cloud, 2022. Disponivel em: <https://www.mongodb.com/>. Acesso em: 22 set. 2022.

PEREIRA, C. **Aplicações web real-time com Node.js**. 1. ed. São Paulo: Casa do Código, v. 1, 2014.

PISCHING, M. **Arquitetura para descoberta de equipamentos em processos de manufatura com foco em Industria 4.0**. Universidade de São Paulo. São Paulo, Brasil, p. 189. 2018.

PONTAROLLI, R. et al. **Microservice Orchestration for Process Control in Industry 4.0**. Metrology for Industry 4.0 & IoT 2020 IEEE International Workshop. Roma, Italia: IEEE. 2020. p. 245-249.

RAVELLI, J. **Arquitetura de Dados baseada no RAMI 4.0 para Manufatura Flexível na Indústria 4.0**. Universidade Estadual Paulista "Julio de Mesquita Filho".Tese de Mestrado, Programa de Engenharia Elétrica interunidades São João da Boa Vista e

Sorocaba, p. . 2022.

RED HAT. APIS, 18 March 2021. Disponivel em: <https://www.redhat.com/pt-br/topics/api/what-does-an-api-gateway-do#:~:text=O%20gateway%20de%20API%20%C3%A9,de%20servi%C3%A7os%20de%20back%2Dend.&text=Esses%20gateways%20normalmente%20gerenciam%20tarefas,limita%C3%A7%C3%A3o%20de%20taxa%20e%20estat%C3%ADst>. Acesso em: 18 march 2021.

ROBO3T - ROBOMONG. Robo 3T is now Studio 3T Free, 2022. Disponivel em: <https://robomongo.org/>. Acesso em: 22 set. 2022.

RÜTTIMANN, B. SPQR, The Central Importance of Quality. **ALUMINIUM Int. Journal**, p. 7-8, 2001.

RÜTTIMANN, B. **Von Lean zu Industrie 4.0—Eine Evolution? Von einer visionären Idee zum realen Verständnis**. Fertigungstechnisches Kolloquium. Zurich/IWF, Switzerland.: [s.n.]. 2015.

RÜTTIMANN, B.; STÖCKLI, M. Lean and Industry 4.0 - Twins, Partners, or Contenders? A Due Clarification Regarding the Supposed Clash of Two Production Systems. **Journal of Service Science and Management**, Inspire AG, Zurich, Switzerland., 09, n. 06, dez. 2016. 485-500.

SCHMIDT, R. et al. Industry 4.0 -Potentials for Creating Smart Products: Empirical Research Results. **18th International Conference on Business Information Systems, Lecture Notes in Business Information Processing (LNBIP)**, Poznań, Poland, v. 208, p. 16-27, June 2015.

SCHROTH, C.; JANNER, T. Web 2.0 and SOA: Converging Concepts Enabling the Internet of Services. **IT Professional**, 3, n. 9, 05-06 2007. 36-41. Disponivel em: <https://ieeexplore.ieee.org/document/4216107>. Acesso em: 29 set. 2022.

SHENG, Q. Z. et al. Web services composition: A decade's overview. **Information Sciences**, v. 280, n. 1, p. 218-238, 2014.

SHENG, Q. Z. et al. **Web services composition:** A decade's overview. [S.l.]: [s.n.].

2014. p. 218-238.

SURI, K. et al. Modeling business motivation and underlying processes for RAMI 4.0-aligned cyber-physical production systems. **2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)**, Limassol, Cyprus, 12-15 set. 2017. 1-6. Disponivel em: <https://ieeexplore.ieee.org/abstract/document/8247702>.

TEODORA, S.; LIVIU, M. Cyber-physical systems - Concept, challenges and research areas. **Control Engineering and Applied Informatics**, Romania, 14, 2012. 28-33. Disponivel em: <https://www.researchgate.net/publication/280228151_Cyber-Physical_Systems_-_Concept_Challenges_and_Research_Areas>. Acesso em: 29 set. 2022.

VEIGA, G.; PIRES, J. N.; NILSSON, K. Experiments with service-oriented architectures for industrial robotic cells programming. **Robotics and Computer-Integrated Manufacturing**, 25, n. 4-5, 08-09 2009. 746-755.

WANG, V.; XU, X. An interoperable solution for Cloud manufacturing. **Robotics and Computer-Integrated Manufacturing**, v. 29, n. 4, p. 232-247, jan. 2013.

WANG, Y.; TOWARA, T.; ANDERL, R. **Topological Approach for Mapping Technologies in Reference Architectural Model Industrie 4.0 (RAMI 4.0)**. World Congress on Engineering and Computer Science. San Francisco, USA: [s.n.]. 2017. p. 25-27.

WEIß, P.; KOELMEL, B.; BULANDER, R. Reference architecture model for the Industrie 4.0 approach in Germany (RAMI4.0) adapted from (VDI, 2015, 7)., Sep 2016. Disponivel em: <https://www.researchgate.net/figure/Reference-architecture-model-for-the-Industrie-40-approach-in-Germany-RAMI40-adapted_fig3_309807447>. Acesso em: 02 Oct 2021.

WORKSHOP “PLATFORMS FOR CONNECTED FACTORIES OF THE FUTURE”. LOGISTICS 4.0 AND THE INTERNET OF THINGS, 5 ago. 2015. Disponivel em: <https://ec.europa.eu/information_society/newsroom/image/document/2015-44/8_huelsmann_11945.pdf>. Acesso em: 06 out. 2020.

ZEZULKA, F. et al. Communication Systems for Industry 4.0 and the IIoT, Czech Republic, 2018. 23-25. Disponivel em: <https://www.sciencedirect.com/science/article/pii/S2405896318308899?via%3Dihub>. Acesso em: 05 set. 2022.

ZEZULKA, F.; MARCON, P.; VESELY, I. S. O. **Industry 4.0 – An Introduction in the phenomenon**. Brno-střed, Czech Republic: ELSEVIER. 5-7 out. 2016. p. 8-12.

ZUEHLKE, D. Smart Factory — Towards a factory-of-things. **Annual Reviews in Control**, Kaiserslautern, Germany, v. 34, n. 1, p. 129-138, mar. 2010.

APPENDIX A – Code, Softwares and Station Dashboards

Order Orchestrator program

The Order orchestrator code was divided in client, item, knowledgeRoute, order services. To run it, do *npm run dev* and run the mes_orchestrator on visual code software. Then it is possible to access the following services that were important for this research.

Client Services is responsible to validate if the entered username is valid and ready to manage orders. It will respond 200 if found and if not found, with a correct message instruction, and if correct, return order details.

Item Services is responsible item related services, such as:

1. Change partNumber: used to change a partNumber, required client username, itemId, and newPartNumber;
2. getOrderbyId: gets order details by its Id;
3. getLastEpcbyPartNumber: get last EPC by a PN to create a correct id on the new EPCs;
4. GetPartNumberByEpc: discoveries PN searching by a particular EPC.

KnowledgeRoute Services is responsible for production item orientation in the factory.

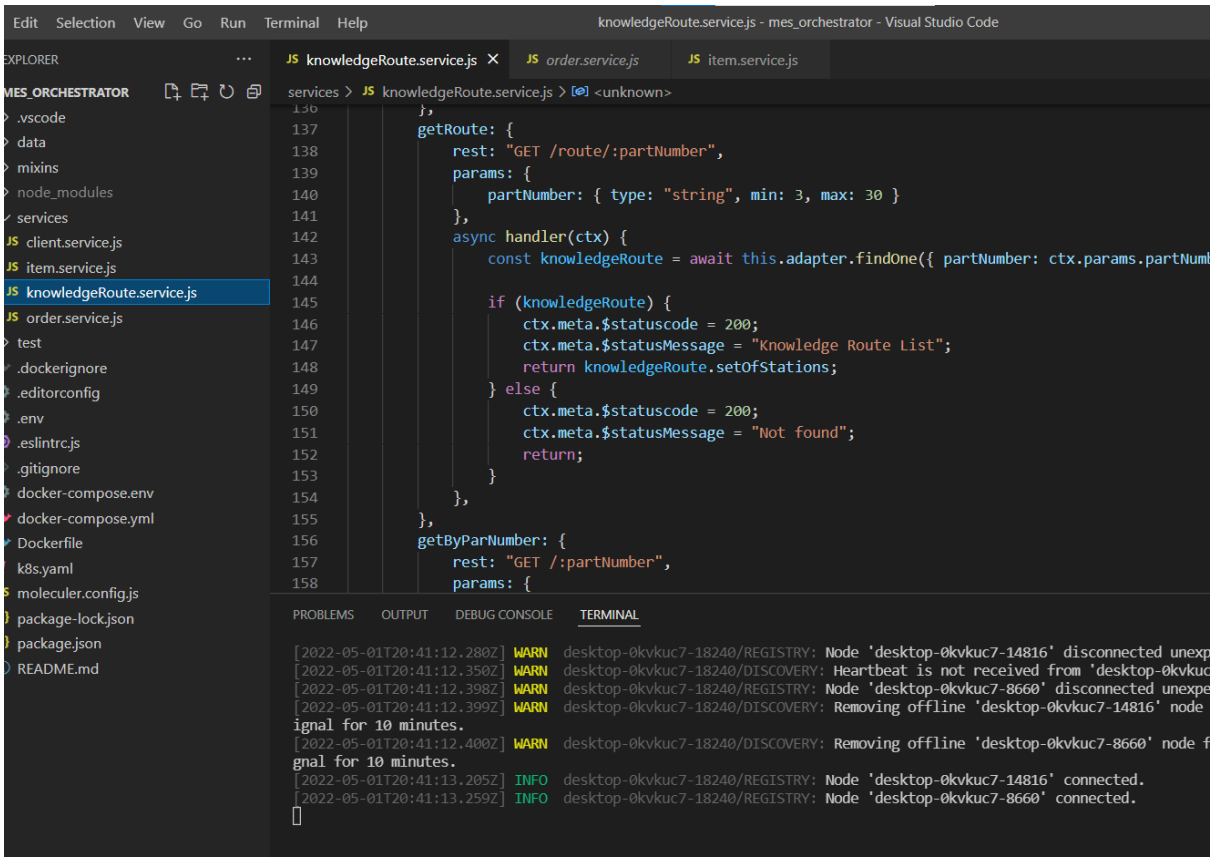
1. getNextStation: used in the station to help the item to find next station, receives the partNumber, the EPC and the current station, and returns the next Station;
2. getRoute: receives a partNumber and return the set of station to produce an item.

Order Services: responsible for order service management

1. createOrder: receives a username, partNumber and a number of items that has to be crated in the DB;
2. get orderAndItems: receives an orderId and return its items and statuses.

The program running can be seen in details in Figure 28. On the left, it is possible to see the services described above. To understand how the program runs, is very important to understand them.

Figure 28. Order Orchestrator, formerly call MES, running.

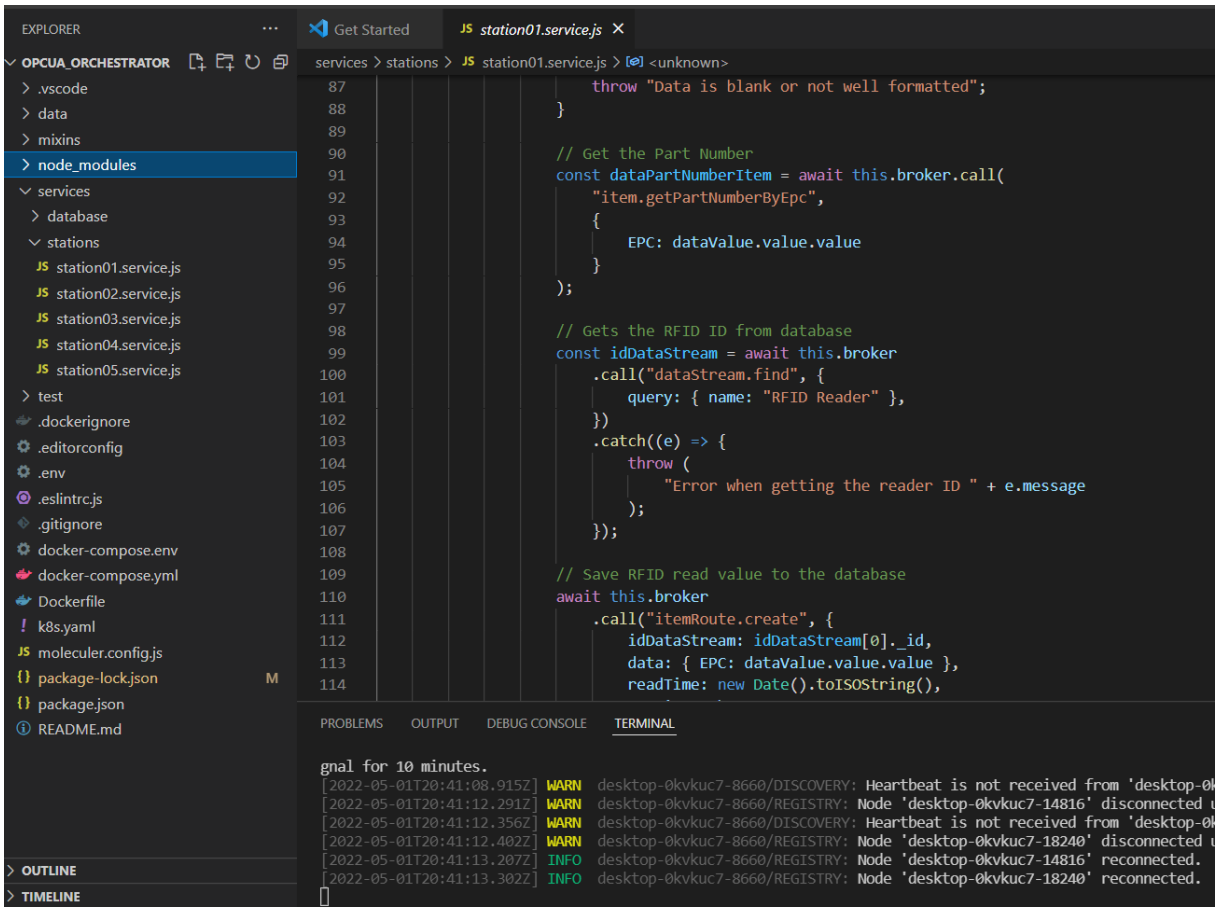


Source: Own Authorship

OPC UA Orchestrator program

The OPC UA orchestrator is not on the scope of this research, it is responsible for the Station structured communication with the server; details are explained in (RAVELLI, 2022). But to validate this research, his program must be running. To run it, do *Npm run dev,* and run the opcua_orchestrator on visual code software. The OPC UA orchestrator is shown on Figure 29; in the left is possible to see all stations from 1 to 5 to represent this test case.

Figure 29. OPC UA Orchestrator running.

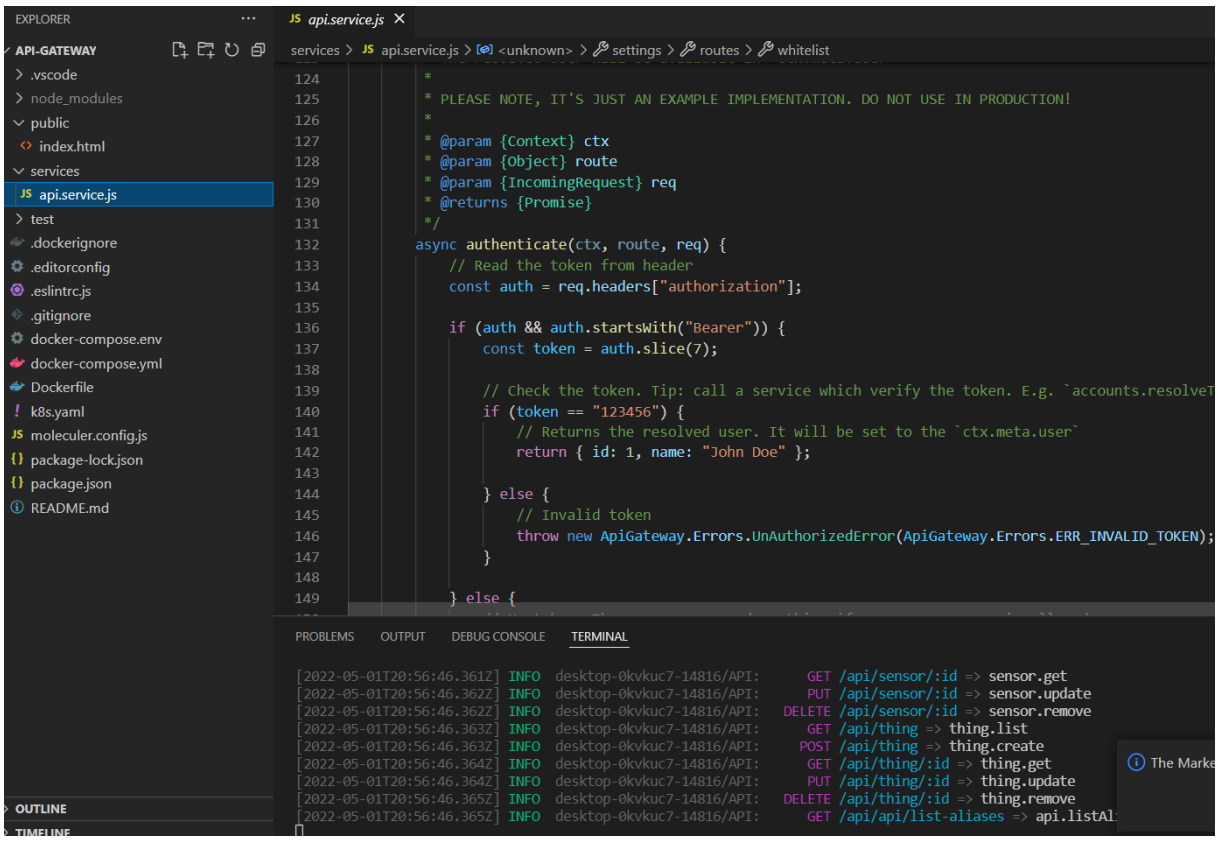


Source**:** Own Authorship

API Gateway program

This software is responsible for frontend and backend integration. It is responsible for authentication and authorization methods to validate permission control using oAuth 2.0 protocol. It contains the following APIs explained and described in APPENDIX B, that represents the logic of the services of the program, and, in Figure 30 it is possible to see the API GW running and the API services.

Figure 30. API GW running.

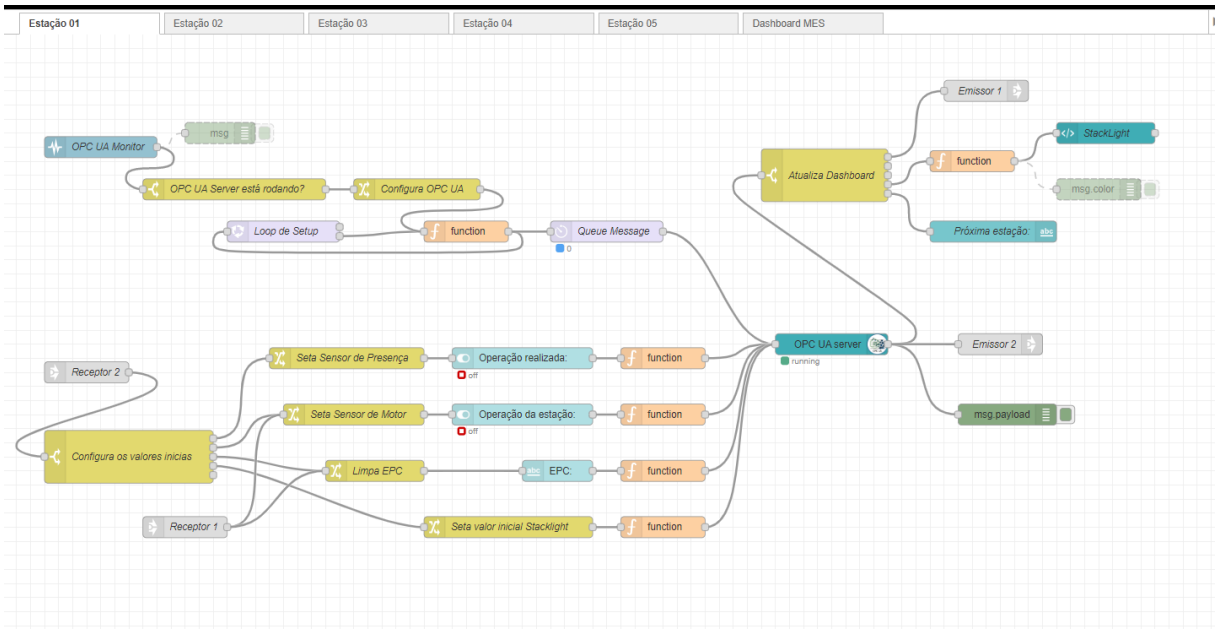


Source**:** Own Authorship

Station Dashboard

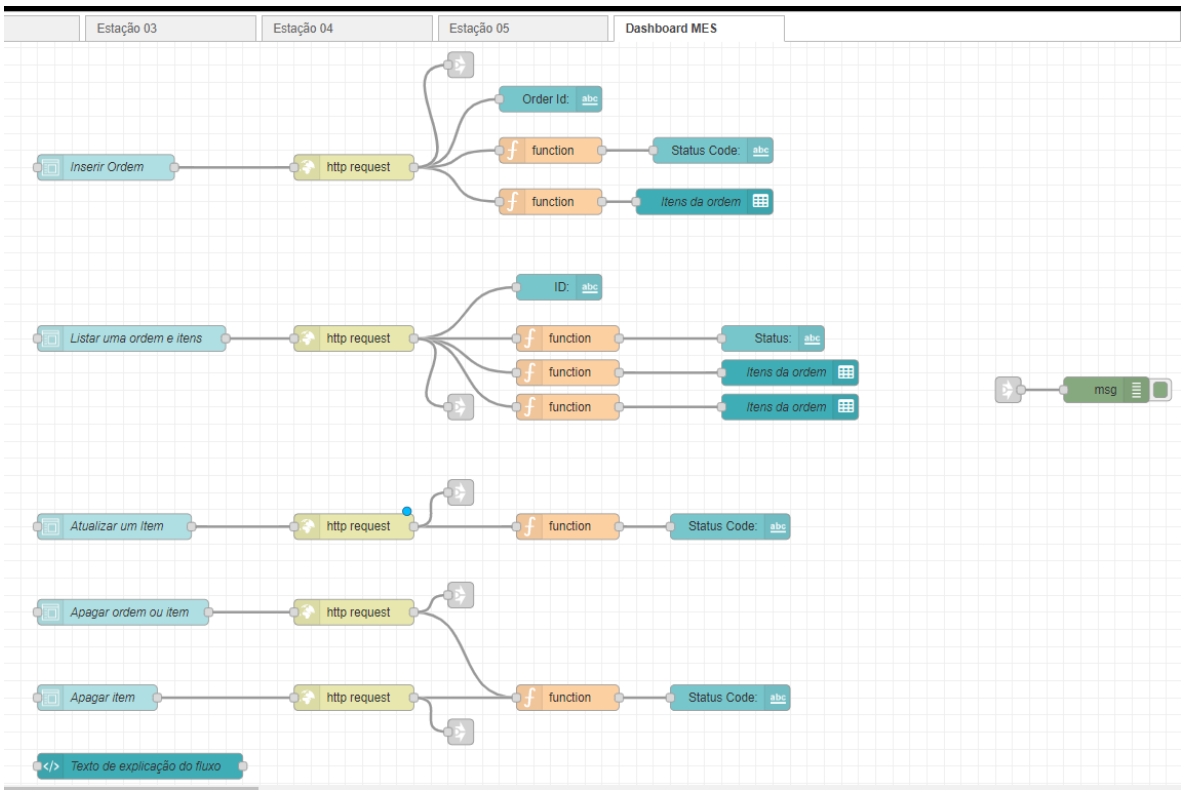
In frontend persperctive, developed in Node-Red, it is a sequence of nodes, communicating with the user and orquestrating the API GW to interact with the factory. The sequence of blocks and nodes to create a station flow is represented in Figure 31. Only Station 1 out of 5 stations created, are represented in the figure.

Figure 31. Stations Node-Red program, showing 1 of 5.



Source**:** Own Authorship

Figure 32. Dashboard Node-Red program.



Source: Own Authorship

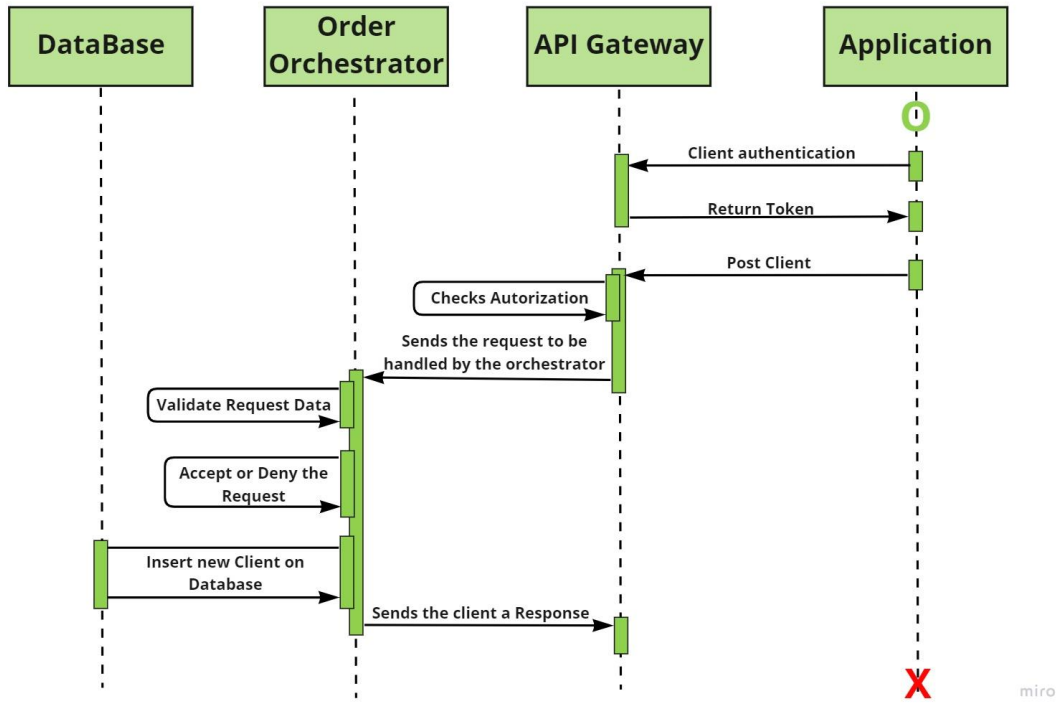
In Figure 32 the node flow to create the dashboard of Business Interface, featured in this research can be seen. All the screens developed are represented on those nodes.

APPENDIX B – Endpoints Diagram

Clients Endpoints

In Figure 33 the state diagram for Post Client can be seen. It is checked authentication as only factory requests can create new clients. After, the request is validated and proceed to be inserted into the database. This allows the client to create orders and items to be fabricated.

Figure 33. State Diagram of Post Client endpoint.

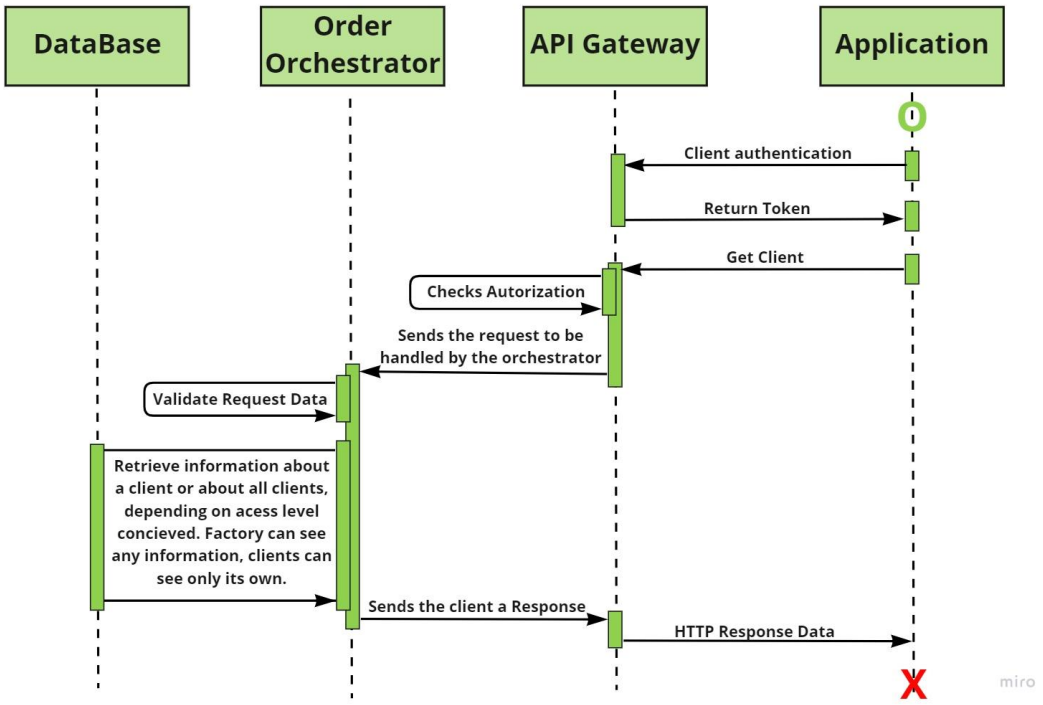


Source: Own Authorship

In Figure 34, the state diagram for GET Client is shown. It is checked authentication to validate if it is a valid client or factory itself. After the client information is queried to be returned to the requester, clients can get their information, and the factory can access data from every client in the DB.

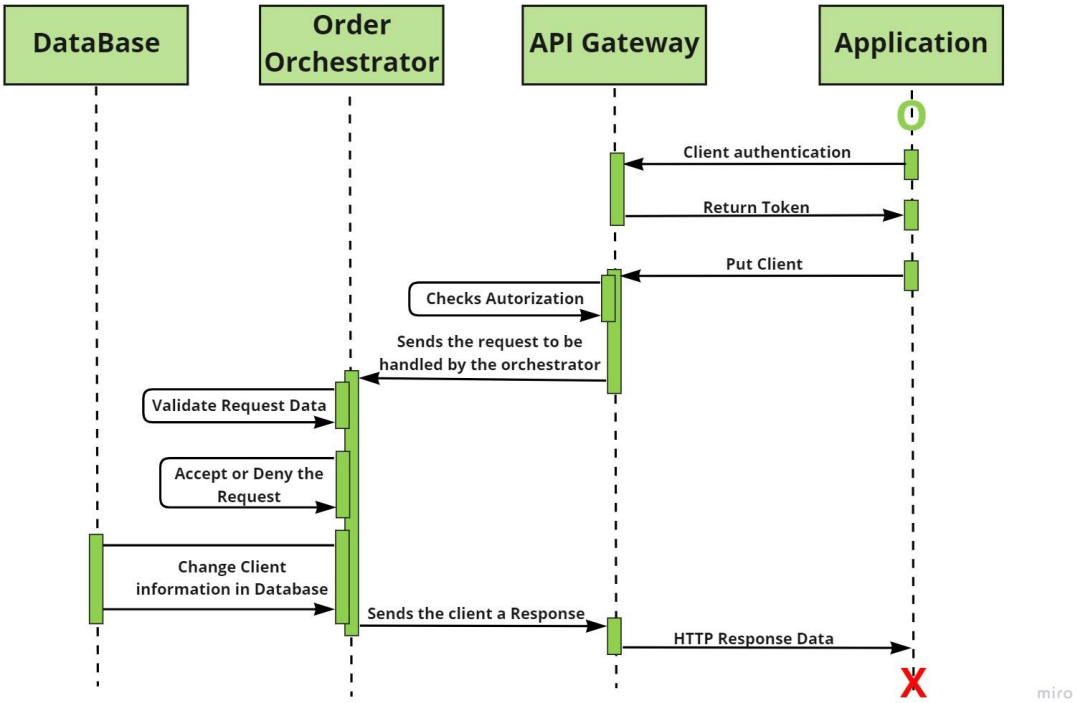
The state diagram for Put Client can be seen in Figure 35. It is checked authentication to validate if it is a valid client or factory itself. Clients can update partial information such as an address, email, phone number, and factory can update any field. This was set so the factory can put a client on *Hold* or blocked as payments is delayed for example, so if a client has any due payment, it cannot request items and orders.

Figure 34. State Diagram of GET Client endpoint.



Source: Own Authorship

Figure 35. State Diagram of Put Client endpoint.

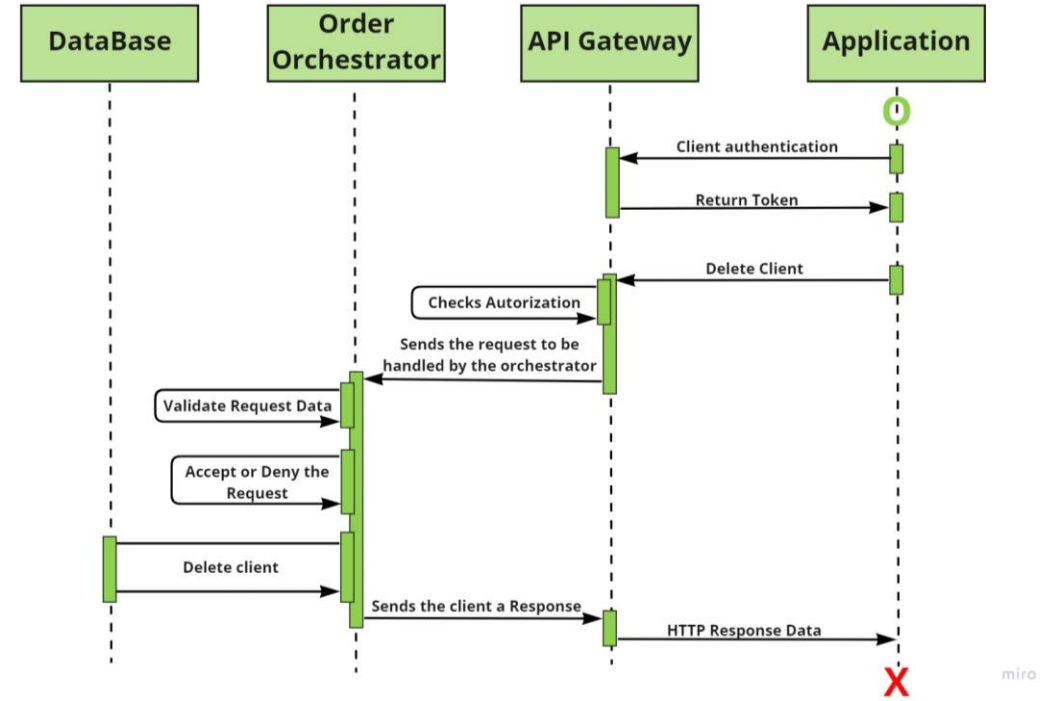


Source: Own Authorship

In Figure 36, the state diagram for Delete Client is depicted. It is checked

authentication to validate if it is a factory request. The client cannot delete itself from the database. Is important to remind that the delete does not delete the data but put client inactive to request new orders.

Figure 36. State Diagram of Delete Client endpoint.

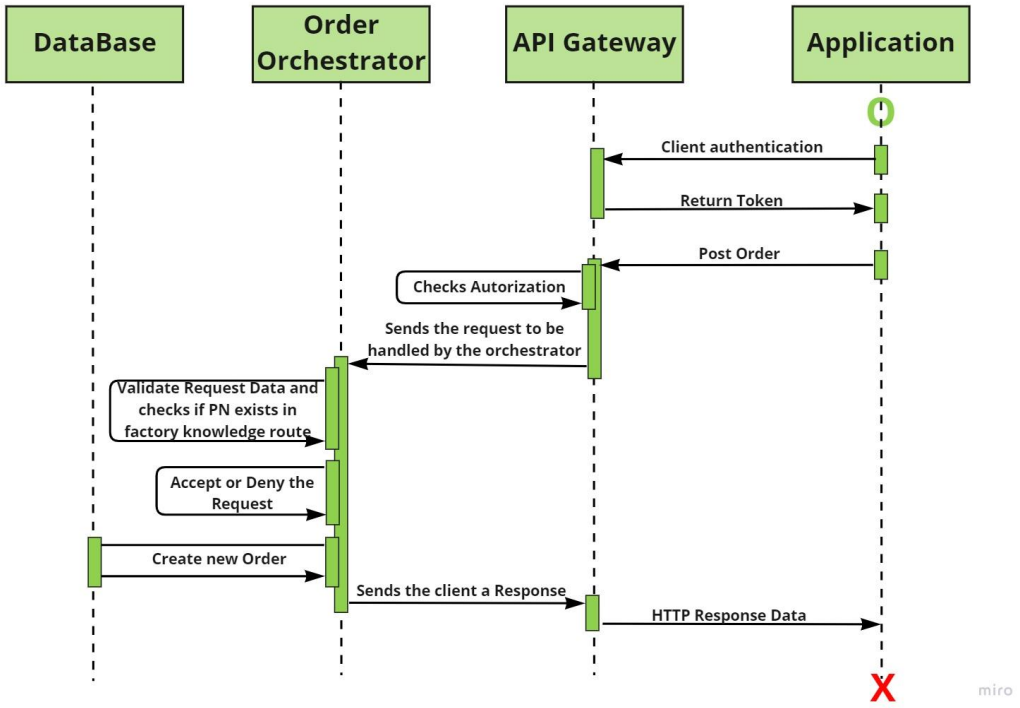


Source: Own Authorship

Order endpoints

In Figure 37, the state diagram for Post Order is shown. It is checked for authentication to validate if it is a factory or client's request, then the Order Orchestrator validates if a knowledge Route exists for this particular PN to validate if the factory is all set up to factory the product. Then success message is returned.

Figure 37. State Diagram of Post Order endpoint.

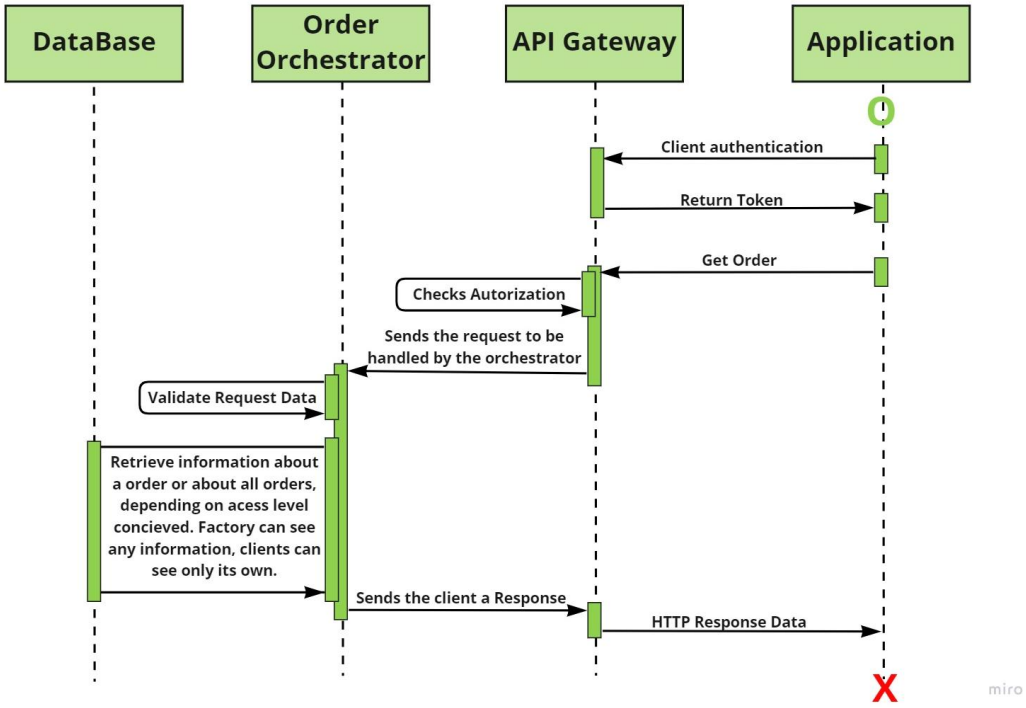


Source: Source: Own Authorship

The state diagram for GET Order can be seen in Figure 38. It is checked authentication to validate if it is a factory or client's request, then the Order Orchestrator retrieves information about Order Collection. If a single Order is passed through, Order status and all Items associated with this particular Order are returned. If no Order is passed, all Orders related to that particular Client and Status are returned. After the search, a message is returned.

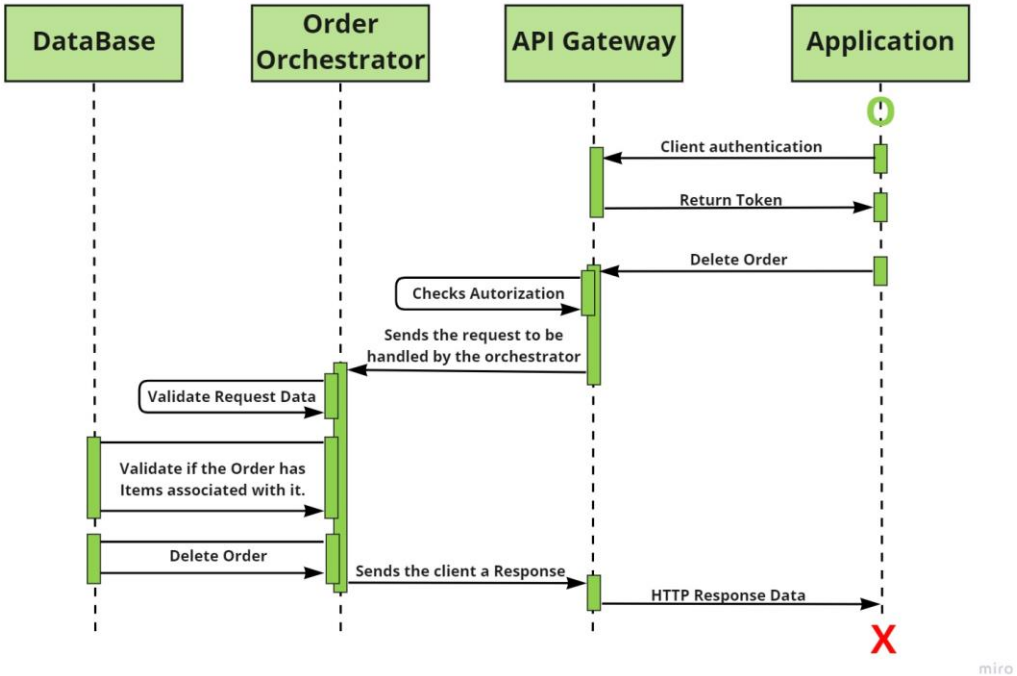
Figure 39, the state diagram for Delete Order is shown. It is checked for authentication to validate if it is a factory or client's request, then the Order Orchestrator will do a simple validation. To delete an Order, it is required that the Order have no Items associated with it. So, it is necessary to delete Items first, then delete Order. This validation is made after a message is returned.

Figure 38. State Diagram of Get Order endpoint.



Source**:** Own Authorship

Figure 39. State Diagram of Delete Order endpoint.



Source**:** Own Authorship

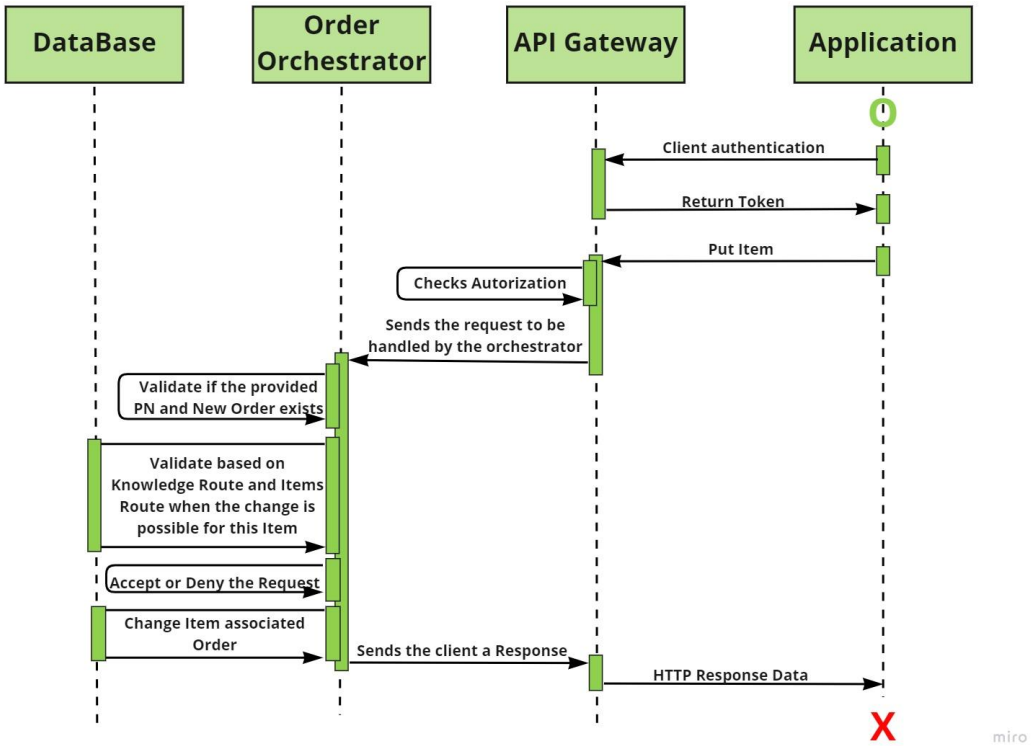
Item endpoints

The state diagram for Post Items, is depicted in Figure 40. It is checked for

authentication to validate if it is a factory or client's request, then the Order Orchestrator validates if the request has a Client and an Order associated with it, then return success.

In Figure 41 shows the state diagram for GET Items. It is checked for authentication to validate if it is a factory or clients request, then the Order Orchestrator will do similar validation explained in GET Orders, idItem is passed, return Items production status and if idOrder is passed, will return a list of all idItems and status associated with the Order. After all, a return with the search criteria is sent.

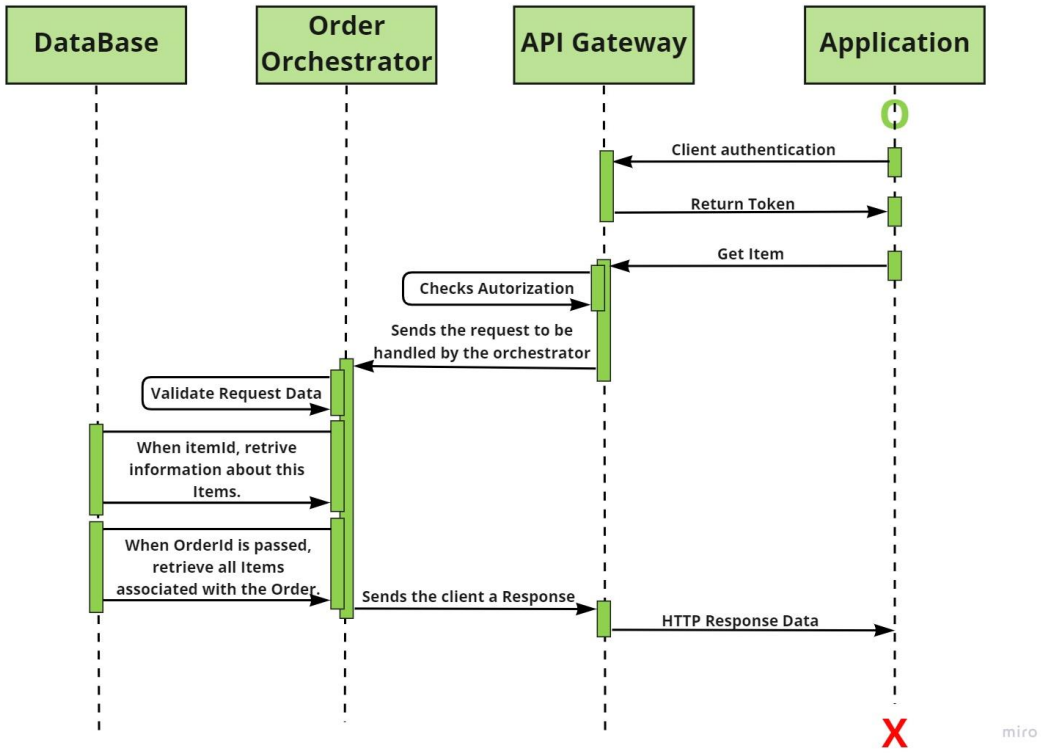
Figure 40. State Diagram of Post Items endpoint.



Source**:** Own Authorship

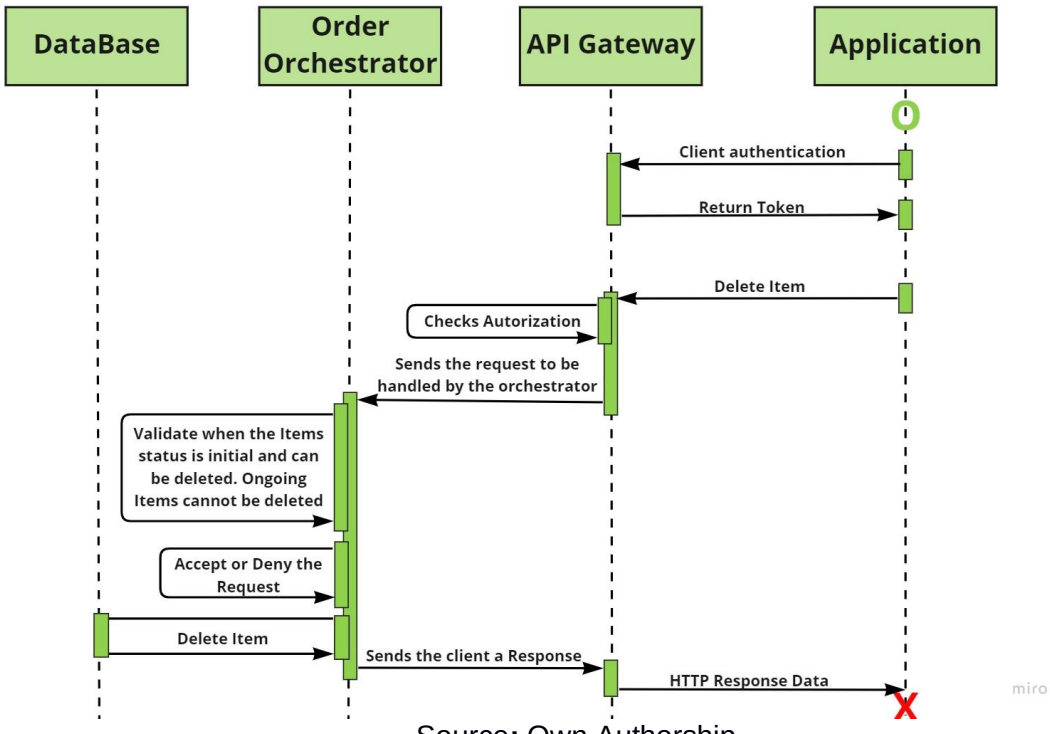
The state diagram for Delete Items is illustrated in Figure 42. It is checked for authentication to validate if it is a factory or client's request, then the Order Orchestrator will validate if the Item has started in production; if not, so it can be deleted but if has started, so it cannot be deleted. A return is sent with this information.

Figure 41. State Diagram of Get Items endpoint.



Source: Own Authorship

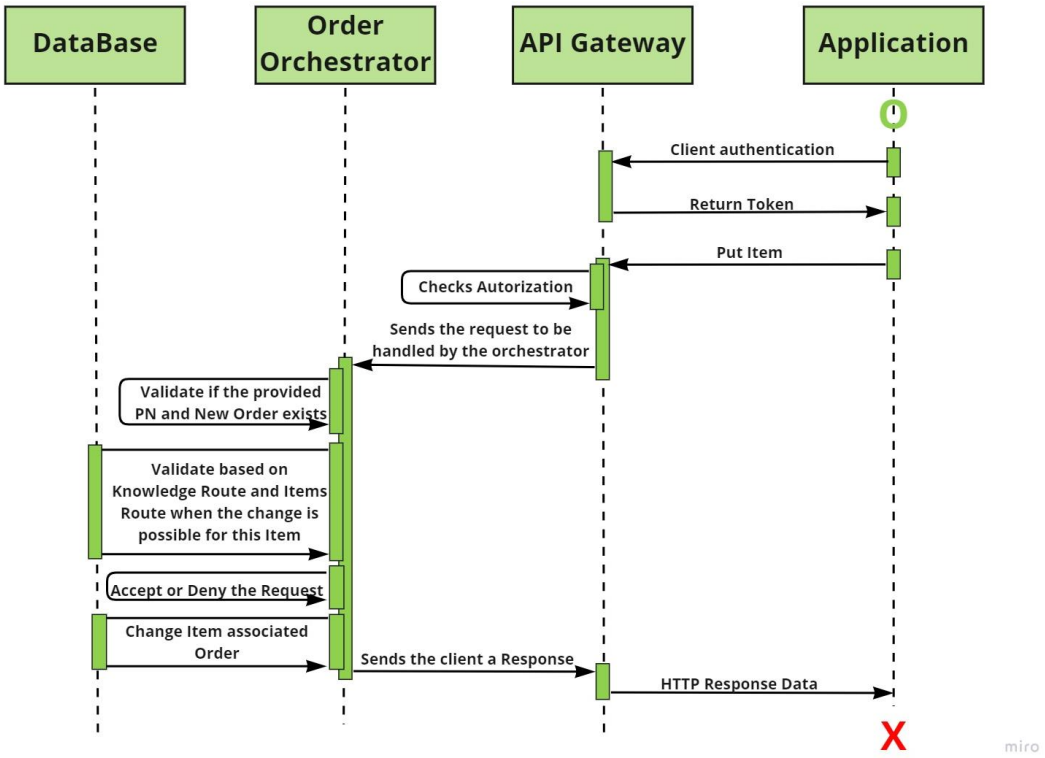
Figure 42. State Diagram of Delete Items endpoint.



Source: Own Authorship

In Figure 43, the state diagram for Put Items is shown. It is checked authentication to validate if it is a factory or client's request, then the Order Orchestrator will run a few validations.

Figure 43. State Diagram of Put Items endpoint.



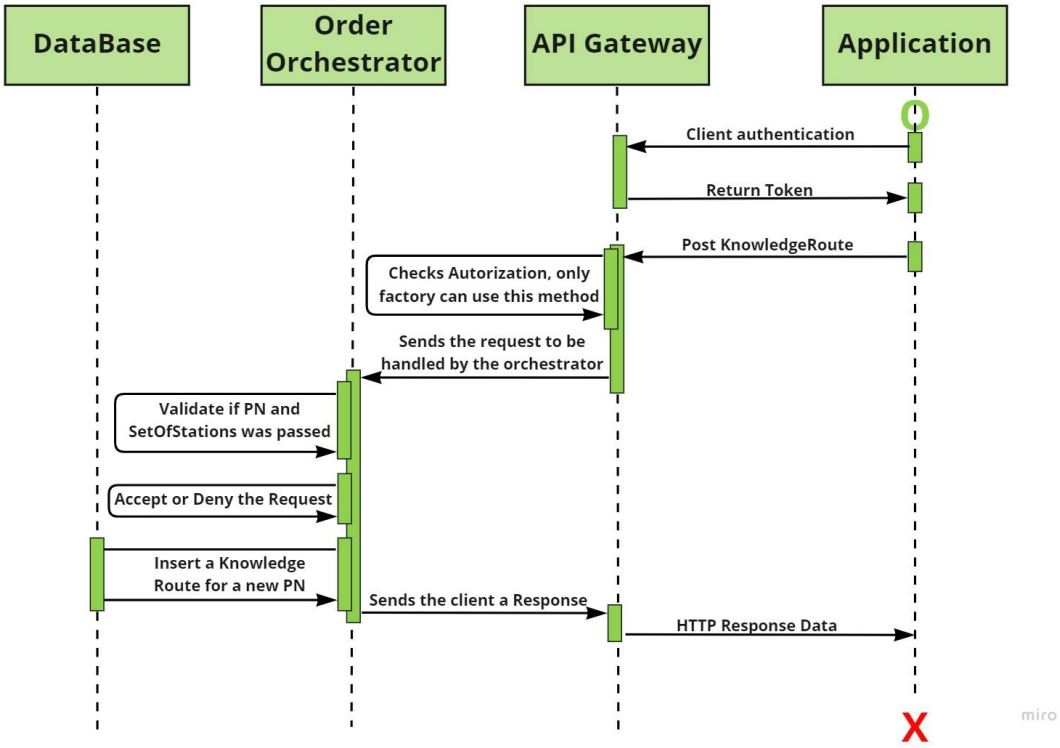
Source**:** Own Authorship

Knowledge Route endpoints

The state diagram for Post KnowledgeRoute is depicted in Figure 44. It is checked authentication to validate if it is a factory request, as only the factory can insert a Knowledge Route. The insertion is quite straightforward, validates if the PN does not already have a route, and inserts it in the DB.

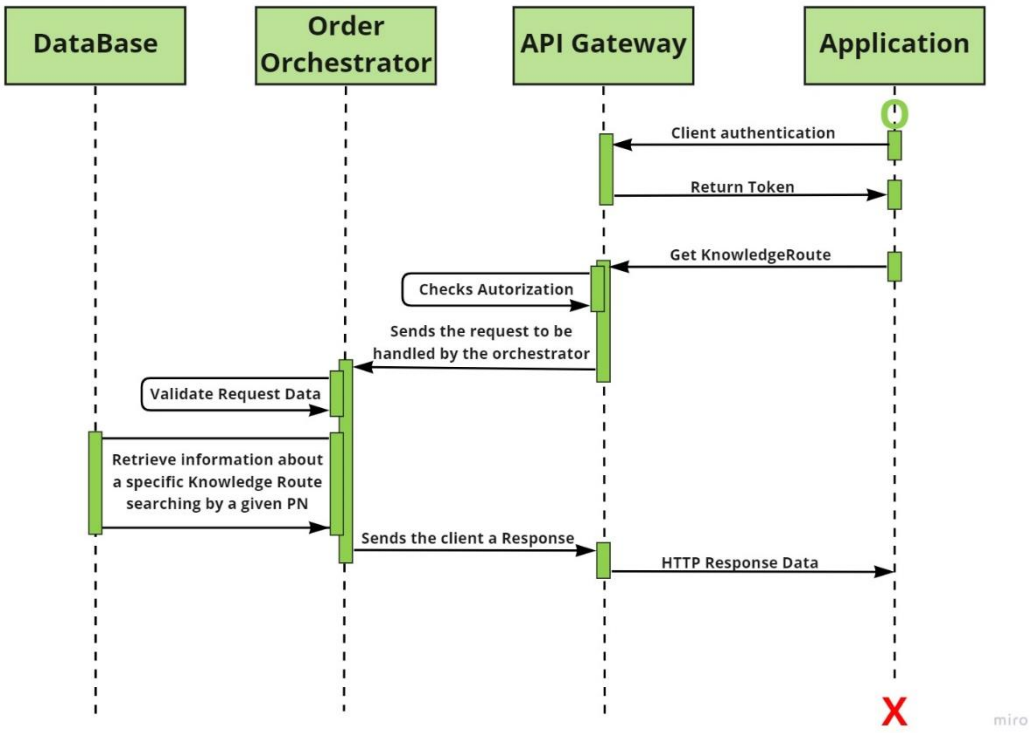
In Figure 45 the state diagram for GET KnowledgeRoute is shown. Is checked authentication to validate if it is a factory request, and then retrieves setOfStations related to the route of that PN.

Figure 44. State Diagram of Post KnowledgeRoute endpoint.



Source**:** Own Authorship

Figure 45. State Diagram of Get KnowledgeRoute endpoint.

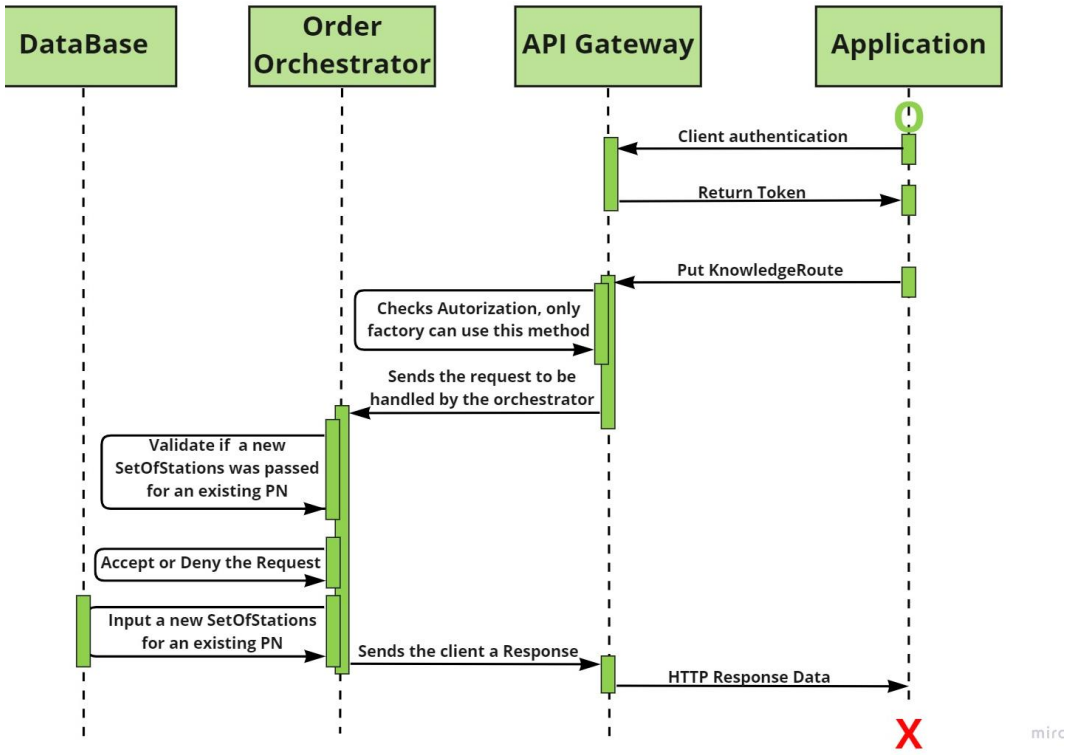


Source**:** Own Authorship

The state diagram for Put KnowledgeRoute is illustrated in Figure 46. It is checked authentication to validate if it is a factory request, as only the factory can

change a Knowledge Route, then validate if the SetOfStations provided is valid and change it in the DB. Ongoing Items are managed to follow the new KnowledgeRoute. Is important to validate if this will not affect ongoing production Items, because if it does so, it will generate route errors in the production line.

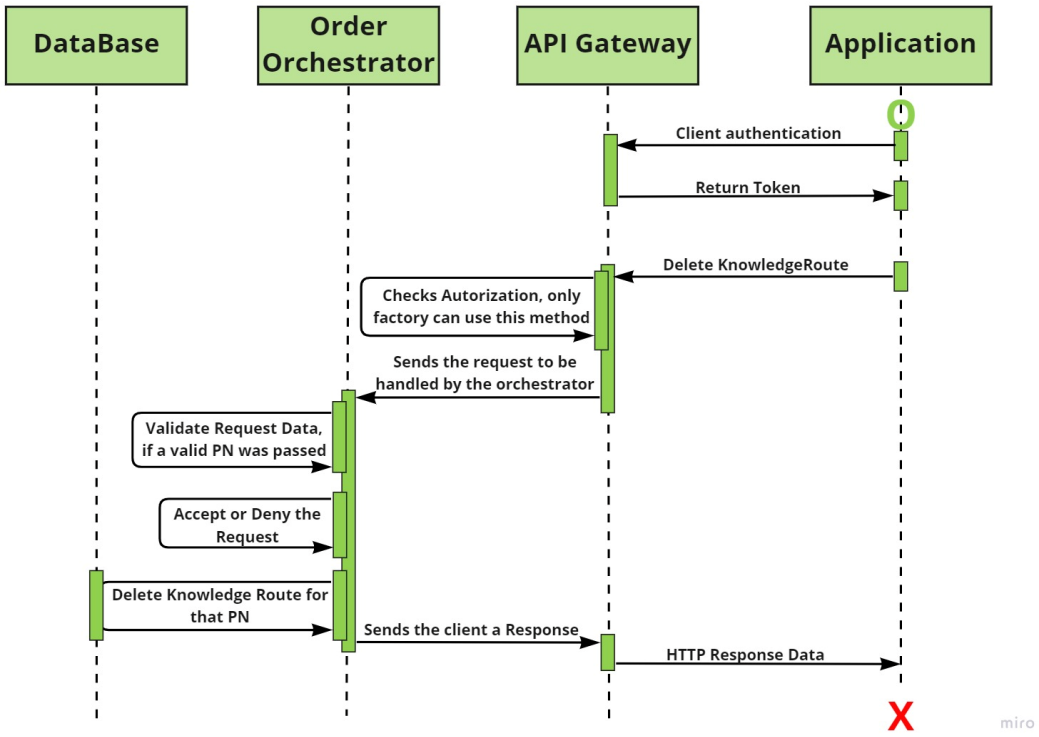
Figure 46. State Diagram of Put KnowledgeRoute endpoint.



Source**:** Own Authorship

In Figure 47, the state diagram for Delete KnowledgeRoute can be seen. It is checked authentication to validate if it is a factory request, as only the factory can delete a Knowledge Route. The validation is quite straightforward, validates if is the PN passed is valid and deletes the requested Route. Such as Put method, it will affect ongoing methods, that's why it can only be done by factory request, since deleting a route will affect ongoing Items.

Figure 47. State Diagram of Delete KnowledgeRoute endpoint.



Source**:** Own Authorship

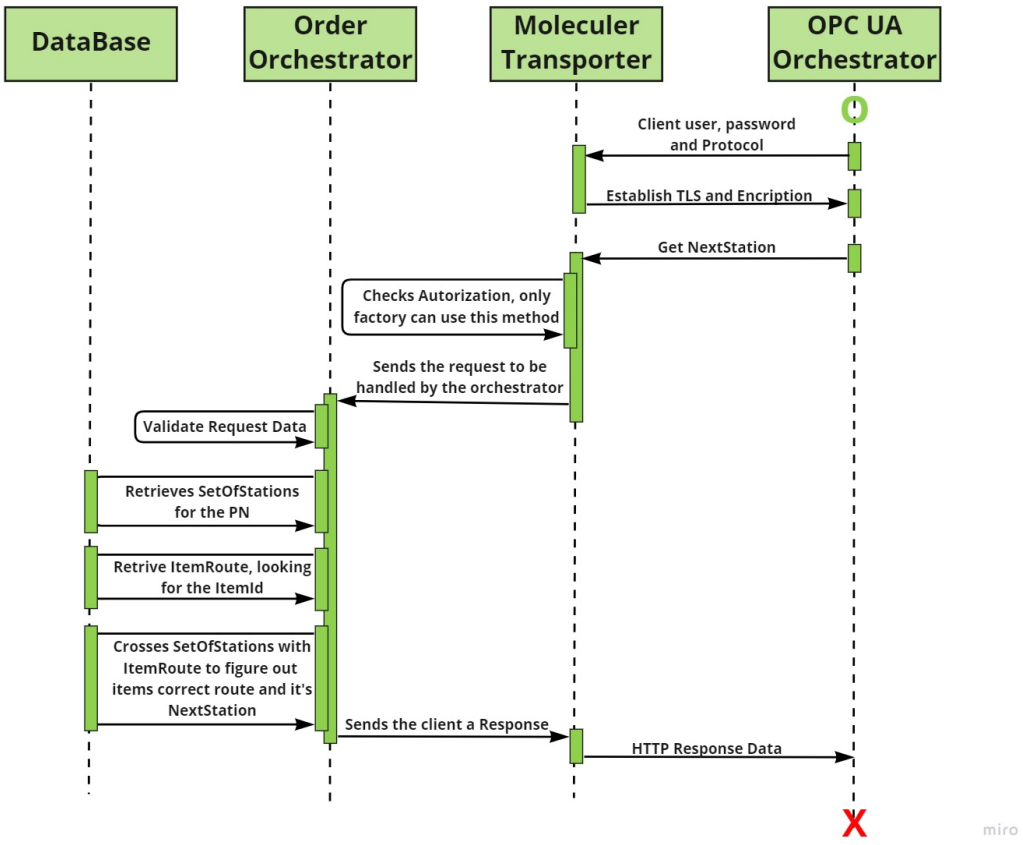
Get NextStation endpoint

The state diagram for Delete Client is presented in Figure 48. It is checked authentication to validate if it is a factory request as this method is only used by factory OPC UA Server to guide Items through the fabrication process. This is an interaction with the intermediate layers of Communication and Information of RAMI 4.0. When the OPC UA sends a PN, a series of validation are made to answer three questions:

- If the Item is on the correct track,
- What must be done in that Item in current Station,
- What are Items next Station.

To do so, it is searched the PN SetOfSations in the KnowledgeRoute collection, then it is retrieved Items path of Station using the history-time collection ItemRoute, searching ItemId. With these two queries it is possible to cross information to answer the three questions. Afterward, it is returned to the OPC UA Orchestrator if the Items are in the correct path or not and if it is also returned the instruction of what must be done in the Station and which Station it may go next.

Figure 48. State Diagram of Get Next Station endpoint.

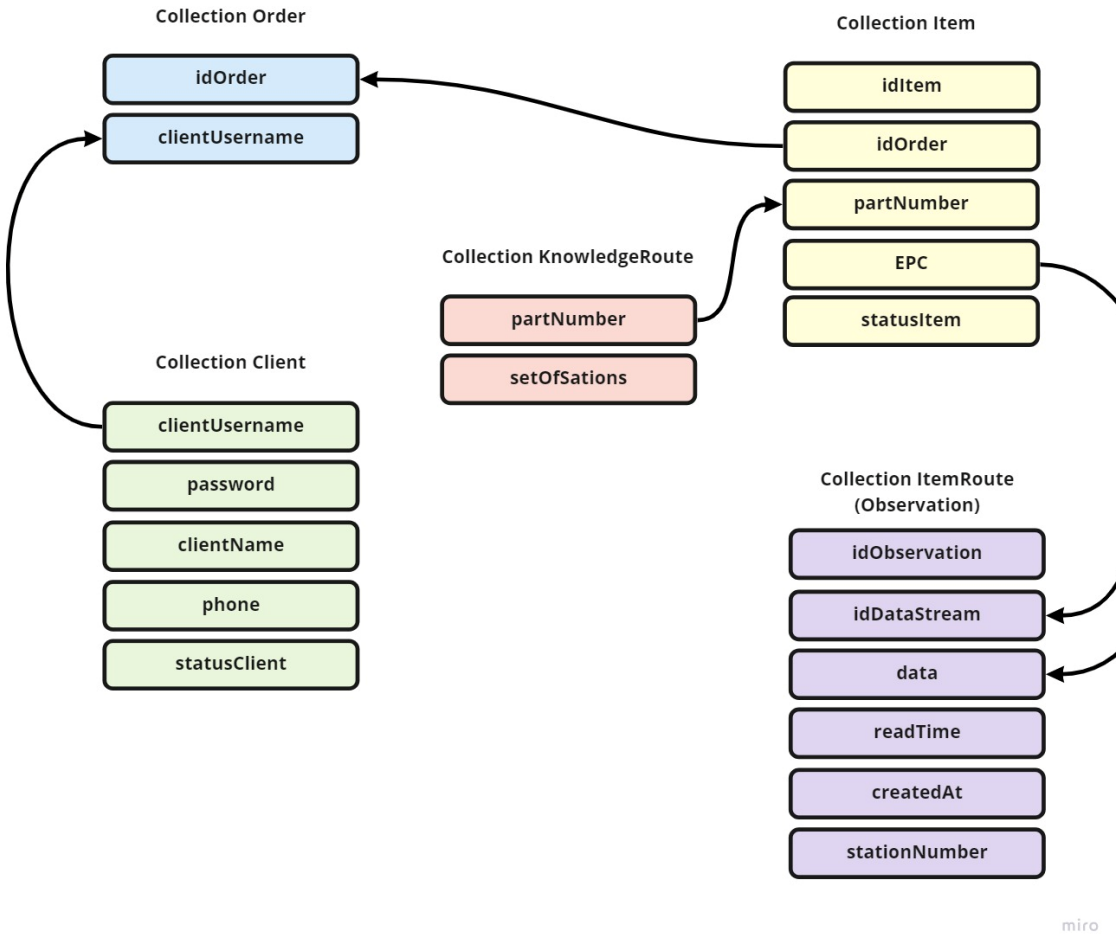


Source: Own Authorship

APPENDIX C - Information Layer: DB overview

The Database is part of the Information layer. In Figure 49, it is possible to see the structure of the Collections, the standard MongoDB database tables. The information is segregated to reflect customer data in the Collection Client, order data in the Collection Order, the more specific information of the Order, the Item Table, and a time view of recording everything that went through the shop floor in the Collection ItemRoute, and finally the knowledge base, that is, the manufacturing rules for any item in the Collection KnowledgeRoute.

Figure 49. Database Structure.



Source: Courtesy of Ravelli and Godoy (2022).

1. Collection Order:

a) idOrder is incremental, primary key and lists a specific order from a specific customer;

b) clientUsername is the primary key, used to identify the customer since only registered customers can request orders.

2. KnowledgeRoute Collection:

a) partNumber is the primary key, used to specify the type of item;
b) setOfStations provides the stations that this specific PN must navigate.

3. Collection Item:

a) idItem is an incremental, primary key, used as the unique identifier of an item;
b) idOrder is an incremental, primary key, used to correlate an item to a particular order;
c) partNumber, is an incremental, primary key, used to identify the item by the part number;
d) EPC is a unique value stored on the RFID tag that represents a combination of PN and SN;
e) statusItem can be 0 if the order has not yet started to be produced (PENDING), 1 if it has already started (PROCESSING), 2 if it has been completed (FINISHED).

4. Collection Client:

a) clientUsername is used to identify the client by an email;
b) password for password storage clients;
c) clientName stores the name of the clients;
d) phone stores customers phones;
e) the status is 0 if the customer is suspended and 1 if the customer is active. Only active customers can request new orders.

5. ItemRoute Collection:

a) idObservation;
b) idDataStream;
c) data is the EPC read in the station;
d) readTime is the time that the read is collected;
e) createdAt is the time that was saved in the DB;
f) stationNumber is the station that the EPC has passed.