Politécnico de Leiria
Escola Superior de Tecnologia e Gestão
Departamento de Engenharia Informática
Master's Degree in Eng.ª Informática – Computação Móvel

KNOWLEDGE-DRIVEN PRODUCTION LINES: SCALABLE SYSTEM FOR REAL-TIME MACHINE MONITORING AND CONTROLLING IN INDUSTRIAL ENVIRONMENTS

DIOGO JESUS GOMES FRANCISCO

Leiria, September de 2024

Politécnico de Leiria
Escola Superior de Tecnologia e Gestão
Departamento de Engenharia Informática
Master's Degree in Eng.ª Informática – Computação Móvel

KNOWLEDGE-DRIVEN PRODUCTION LINES: SCALABLE SYSTEM FOR REAL-TIME MACHINE MONITORING AND CONTROLLING IN INDUSTRIAL ENVIRONMENTS

DIOGO JESUS GOMES FRANCISCO
Number: 2220646

Project report carried out under the guidance of professors: Professor Nuno Carlos Sousa Rodrigues (nunorod@ipleiria.pt), Professor Alexandrino José Marques Gonçalves (alex@ipleiria.pt), Professor Roberto Aguiar Ribeiro (roberto.ribeiro@ipleiria.pt) and Professor Anabela Gonçalves Rodrigues Marto (anabela.marto@ipleiria.pt).

Leiria, September de 2024

ACKNOWLEDGMENTS

Firstly, I would like to profoundly thank all my mentors, namely Professor Nuno Carlos Sousa Rodrigues, Professor Alexandrino José Marques Gonçalves, Professor Roberto Aguiar Ribeiro, and Professor Anabela Gonçalves Rodrigues Marto, for all their support, teachings and guidance throughout the project. Certainly, they prepare me better for the world of work.

Furthermore, I express my appreciation to the ESTG and CIIC for the installations and materials made available during the development process of the present project. Without these units, specifically CIIC as a scientific unit funded by FCT - Fundação para a Ciência e a Tecnologia, I.P., through UIDB/04524/2020, I wouldn't have achieved the personal, academic, and scientific achievements. In collaboration with the mentors, it was possible to publish one scientific article related to the present work [1] on behalf of CIIC, where another one is currently under review.

This work was initiated under the project INOVMINERAL 4.0 - Tecnologias avançadas e software para os recursos minerais, project no. 46083, funded by COMPETE 2020 - Programa Operacional Competitividade e Internacionalização, under the program PORTUGAL 2020, through FEDER – Fundo Europeu de Desenvolvimento Regional and finished under the project Sustainable Stone by Portugal agenda funded by the European Union/Next GenerationEU (02/C05-i01.02/2022.PC644943391-00000051).

In addition, a special appreciation to my friends included in the CIIC scientific unit, specifically to the researchers Diogo Pinto, Bruno Rocha, Ruben Carreira, Tiago Ribeiro and Ana Cassia for all the knowledge shared and motivation given.

Finally, I express my deepest gratitude to my family for all their support throughout my studies, without whom all the knowledge and experience I have acquired would not have been possible.

ABSTRACT

Today, the industrial sector is already in the Industrial Revolution 5.0. However, some enterprises have not reached the fourth industrial revolution either because they don't know the potential of the technologies involved or because they don't have the monetary resources needed to integrate them into the manufacturing processes. One of the affected sectors by this lack of digitization is the stone industrial sector. These enterprises contain heavy machines within big warehouses, which work stones through robotic arms that move on several axes with cutting and milling tools to elaborate them and create several products, such as kitchen worktops and decorative pieces. Thus, a solution was developed to help these enterprises adapt and evolve. The present solution was initially conceptualized for the stone ornamental industry sector, but it can be the basis for other future solutions in this branch. The current developed solution integrates the OPC UA protocol as it is considered an industrial standard and has recently been incorporated into industrial machines, which promotes longevity for this solution. In terms of features, the developed system enables the industrial stakeholders in a factory to remotely visualize the parameters of the existing machines on the factory shop floor, as well as to modify their state from a Web application, increasing the operators' awareness and control over the manufacturing process. At the level of new machine integration into the developed system is possible to make it in a simple way and without advanced technical knowledge, which guarantees the independence of companies in the process. In addition, this solution took into account the concept of DT, which consists of having digital representations of one or more physical entities, such as industrial machines, where they are fully synchronized with each counterpart. These representations are made through a dynamic web platform, which adapts to each machine integrated into the system. Last but not least, the solution was tested at the level of transmission data latency, the usability of the web application, and coverage of several physical entities, which simulated a production line. As a result, the system has demonstrated the ability to keep operators and engineers synchronized on production lines remotely, as well as demonstrating the ability to offer versatility in terms of dynamically incorporating different machines without a great amount of knowledge and effort by the users.

RESUME

Atualmente, o sector industrial já se encontra na revolução industrial 5.0. Contudo, algumas empresas não conseguiram chegar à quarta revolução, quer por não conhecerem o potencial das tecnologias envolvidas, quer por não terem recursos monetários para as integrar nos seus processos de negócio. Um dos setores afetados por esta falta de digitalização é o sector da pedra. Algumas das características desta indústria são a existência de máquinas de grande porte em grandes armazéns que trabalham a pedra a partir de braços robóticos que se movimentam em vários eixos com ferramentas de corte e de fresagem para conseguir então a elaboração da pedra e criação de vários tipos de produtos, como bancadas de cozinha e peças decorativas. Desta forma, uma solução foi desenvolvida com o intuito de ajudar estas empresas a adaptaram-se, para que consigam evoluir. A presente solução foi inicialmente conceptualizada para o sector da indústria de ornamentação da pedra, no entanto pode servir como base a soluções de outros ramos. A presente solução integra o protocolo OPC UA, pois o mesmo é considerado um padrão da indústria e tem vindo a ser incorporado nas máquinas industriais mais recentes, o que promove longevidade à solução. Em termos de funcionalidades a solução permite às pessoas envolvidas na produção industrial em certa fábrica visualizar remotamente os parâmetros das várias máquinas existentes em chão de fábrica, como também modificar o estado das mesmas a partir de uma plataforma web, aumentando a consciência e o controlo dos operadores sobre os processos de fabrico. Ao nível da integração de novas máquinas na solução é possível fazê-la de uma forma simples e sem conhecimentos técnicos avançados, o que garante às empresas independência neste processo. Além disso, esta solução incorpora o conceito de gêmeos digitais, cujo objetivo é criar representações virtuais de uma ou mais entidades físicas, como máquinas industriais, totalmente sincronizada em tempo real com suas versões físicas. Esta representação é feita através de uma plataforma web dinâmica, adequada a cada máquina integrada no sistema. Por fim, a solução foi testada a nível de latência na transmissão de dados, usabilidade e de cobertura de várias entidades físicas, simulando uma linha de produção. Como resultado a solução demonstrou ter a capacidade para manter os operadores sincronizados com a linha de produção à distância e que consegue oferecer a versatilidade no que toca à incorporação de máquinas dinamicamente sem grande esforço e conhecimento por parte de utilizadores.

INDEX

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AI	Artificial Intelligence.
API	Application Programming Interface.
AR	Augmented Reality.
ASGI	Asynchronous Gateway Interface.
CCP	Cloud Computing Platform.
CIIC	Computer Science and Communication Research Center.
CNC	Computer Numerical Control.
CSS	Cascading Style Sheets.
DOM	Document Object Model.
DR	Digital Representation.
DSRM	Design Science Research Methodology.
DT	Digital Twin.
ERP	Enterprise Resource Planning.
ESTG	Escola Superior de Tecnologia e Gestão.
GUI	Graphic User Interface.
HMI	Human-Machine Interface.
HTML	Hypertext Markup Language.
HTTP	Hypertext Transfer Protocol.
I4.0	Industry 4.0.

I5.0	Industry 5.0.
ICT	information and communication technologie.
IETF	Internet Engineering Task Force.
IIoT	Industrial Internet of Things.
IoT	Internet of Things.
IP	Internet Protocol.
IT	Internet Technology.
JA	Java Application.
JS	JavaScript.
JSON	JavaScript Object Notation.
MES	Manufacturing Execution System.
MQTT	Message Queuing Telemetry Transport.
NASA	National Aeronautics and Space Administration.
NPM	Node Package Manager.
OPC Classic	Open Platform Communications Classic.
OPC UA	OPC Unified Architecture.
ORM	Object-Relational Mapping.
OS	Operating System.
P2P	Peer-to-Peer.
PLC	Programmable Logic Controller.
PT	Physical Twin.
Pub-Sub	Publish-Subscribe.
REST	Representational State Transfer.
RPM	Rotation per Minute.

SCADA	Supervisory Control and Data Acquisition.
SEQ	Single Ease Question.
SME	Small and Medium Enterprises.
SOAP	Simple Object Access Protocol.
SQL	Structured Query Language.
SUS	System Usability Scale.
UI	User Interface.
URL	Uniform Resource Locator.
UX	User Experience.
VCS	version control system.
VM	Virtual Machine.
WSGI	Web Server Gateway Interface.

1

INTRODUCTION

Currently, many industrial production companies in Portugal have not yet reached the popular industrial revolution called Industry 4.0 (I4.0). As a result, government projects have been created to help Small and Medium Enterprisess (SMEs) to evolve toward this transformation, some encompassing the stone industry [2, 3]. However, the next industrial generation is already defined as Industry 5.0 (I5.0), focusing on people's jobs and green production [4]. I4.0 aims to integrate digital technologies in the production lines sphere and connect the manufactories to the internet. It overlaps various other concepts, such as Smart Factories, Industrial Internet of Things (IIoT), Smart Industry and Advanced Manufacturing [5]. This revolution aims to create more controllable, predictable, and efficient production lines through several functionalities and technologies:

- Analysis of production data, carried out by workers on the shop floor or through technologies such as big data and cloud computing;
- Use of information and communication technologies (ICTs) to do the digitization of factory process data so that there is integration with other systems, like supply and logistics systems, promoting synchronization between various industrial units;
- Improved decision-making for workers in industrial production, using ICTs;
- Simulation of production processes to improve the prediction of manufacturing costs and duration, as well as to ensure synchronism between tasks. This process can consider the historical data retrieved from the production line, contributing to optimizations.

Therefore, with the application of I4.0, the productivity of the production processes can increase, and the downtimes can be reduced. Another concept that potentiates I4.0 is the creation of Digital Twins (DTs), which can help companies reach the referred revolution [6]. It was considered one of the emerging technologies of 2019 by Gartner [7]. This technology aims to represent a physical entity, which encompasses sensors and actuators, in the digital world from digital representation. The communication between these two entities must be bidirectional, where changes

in the state of the physical entity lead to immediate changes in the digital entity, and vice versa [8]. It can be applied to several sectors, like health, farming, and industrial manufacturing [9, 10, 11]. In the case of the manufacturing process, the digital model is responsible for replicating a production line or equipment inserted in it. Besides, it has the same objectives of the I4.0, which is to use the data generated by physical entities to find possible ways to improve the efficiency of the production and understand the future state of the operations to prevent issues [12]. These features can be reached by including other technologies, such as Big Data and Artificial Intelligence (AI). They enable a more thorough exploration of the data, thereby extracting more refined information to help industries improve their decision-making. In the case of the stone industry, the data generated by the machinery can be used to obtain the level of wear of the cutting saws, so that effective maintenance can be carried out, avoiding production errors [13].

There are several DT implementations in the scientific literature with different approaches. However, the following gaps were identified:

- The DT framework is created for a specific machine tool and cannot be scaled up to encompass an entire production line, which hinders production awareness across the entire line and production synchronization [9, 14];
- Many DT implementations only mention/implement unidirectional communication with the physical machine, which does not allow control of it from the upper layers. This aspect makes the concept of a DT not well covered [9, 14];
- Some developed solutions provide only a specific service, such as machine emulation and state prediction, resulting in incomplete DT solutions [14];
- Lack of unified digital interfaces, like protocols, hinders vertical interoperability, not allowing production systems meant for management to access or transfer data from/to several machines [11, 15]. Therefore, if this issue of unifying interfaces is not taken into account, horizontal interoperability and the addition of future modules could also be compromised [9];
- Lack of intuitive DT-based Human-Machine Interfaces (HMIs) that support both technical users and non-technical users so that stakeholders continue to contribute to the smooth running of the manufacturing process [9].

These gaps motivate the need to create a DT-based system capable of covering a production line of an industrial environment, using a means of communication capable of maintaining interoperability between information systems and machines on the factory floor. These communications should be supplier-independent to

promote affordability to the SMEs. This inclusion of machines should be easy for SMEs so that they can integrate new machines over time. These systems should include the principal features of a DT as emulation and control of the physical entities so that there is a bidirectional communication between entities. Therefore, the present work intends to create a DT-based system with these characteristics to promote digitization in the stone industry. In summary, considering the defined needs the following research questions are presented:

- RQ1 - How does the implementation of bidirectional machine communication systems can improve the working process of the operators and engineers in production environments?
- RQ2 - How can a DT-based system be designed to allow SMEs to easily integrate their different machines into a system?
- RQ3 - How can a system represent different machines through a Graphic User Interface (GUI) to be accessed and controlled remotely by factory operators and engineers?

Considering the outlined research questions, the present work proposes a system feasible to the SMEs, capable of encompassing several machines within a production line. The solution simplifies machine integration, allowing companies to dynamically add machines as production lines evolve through a developed Web application. Additionally, it utilizes a unified communication protocol called OPC Unified Architecture (OPC UA) to communicate with machines. This choice was made for the following reasons:

- The most commonly used protocol as standard in the industry area, which enables compatibility with previous solutions already developed with this specification [16, 17, 18];
- Communication interface that enables the bidirectional communication, i.e., allows the control and monitoring of parameters made available by physical entities [17];
- Communication protocol recommended by I4.0 standard architectures [19];
- Unified protocol that has been incorporated into industry machinery and vendor solutions to promote the interoperability among systems [20, 17, 21].

Furthermore, the present solution enables operators and engineers to control and monitor the different machines included in the system via a versatile web application GUI. The machines compatible with the developed system may differ in terms of parameters. However, the developed GUI allows stakeholders to integrate them. In

addition to this interface, the system was able to support other types of interfaces, such as Augmented Reality (AR) interfaces, to optimize the interaction with a user, despite the focus of this research is not on this sort of interface [22]. This type of interface enables users to interact with virtual elements overlaid on the real environment, using devices such as mixed-reality glasses. Lastly, this work presents a general architecture that can be valid in various industrial sectors and used as the basis for future works.

The present work is structured into several sections. Section 2 presents the several concepts and technologies that supported the present research. Section 3 mentions all the academic and commercial solutions, describing the principal features and handicaps of each one. Section 4 illustrates the roadmap planned with the several phases of the present work. Section 5 presents and characterizes the conceptual architecture in this research, while Section 6 describes all the modules involved in the developed solution. Section 7 shows the tests performed against the solution, as well as usability tests carried out. Also, the results obtained are discussed in that section. At last, Section 8 presents the conclusions, including the contributions of the present research, and the future developments that can aggregate value to this solution.

2

BACKGROUND

This section initially explains the organizational methods considered for the research lifecycle, covering project management techniques and workflow strategies that facilitate effective and timely delivery and development. In addition, the technologies considered in developing the intended solution are explained, highlighting them feature, the advantages and disadvantages, and the reasons for their consideration. Furthermore, the present section also explores relevant concepts linked to the solution as it allows readers to gain an insight into what the solution requires.

2.1 ORGANIZATIONAL TECHNIQUES

The present section explains the methodology used in the present research called Kanban. In most cases, this framework is used in the software development process in a business context, focused on principles like planning, iterative development with customer feedback and continuous adaptation. Methodology capable of enabling synchronization between the people involved in a given project, keeping track of the tasks carried out or not throughout the development process. In addition, some other frameworks were considered, such as Scrum and eXtreme Programming (XP). In addition, some other frameworks were considered, such as Scrum and eXtreme Programming (XP). The considered methodologies are agile methodologies as they tend to be more effective than traditional methodologies, such as waterfall methodology [23]. In addition, the Git version control system (VCS) is explained because it was the method used to save the increments of the developed software.

2.1.1 *Kanban*

Kanban is an Agile methodology that is utilized for visualizing workflow, which aims to improve efficiency across several domains, including software development. It facilitates synchronization among project teams throughout the development cycle by organizing and distributing tasks based on priority and available time.

In that framework, a tool called Kanban board is used to keep the normal workflow of a project [24]. There are several tools that allow the creation of this kind of boards based on Kanban methodology, such as Trello [25], Jira [26], and Notion [27]. These boards are filled with tasks, where each one is represented by one Kanban card, which always has a state that can change during the project development. For instance, a card can contain information about a feature that should be implemented in software. Futhermore, the Kanban boards have defined different stages, each one is represented by a column, which lets team know the status of a task at any time (e.g., if one task is being developed it is included in the "in progress" column). In addition, each column should have a limit of cards, this is called work in progress (WIP) limits, this allows companies to understand the level of productivity of a team and ensure that there is good quality development. Moreover, a possible Kanban board is illustrated in Fig. 1.

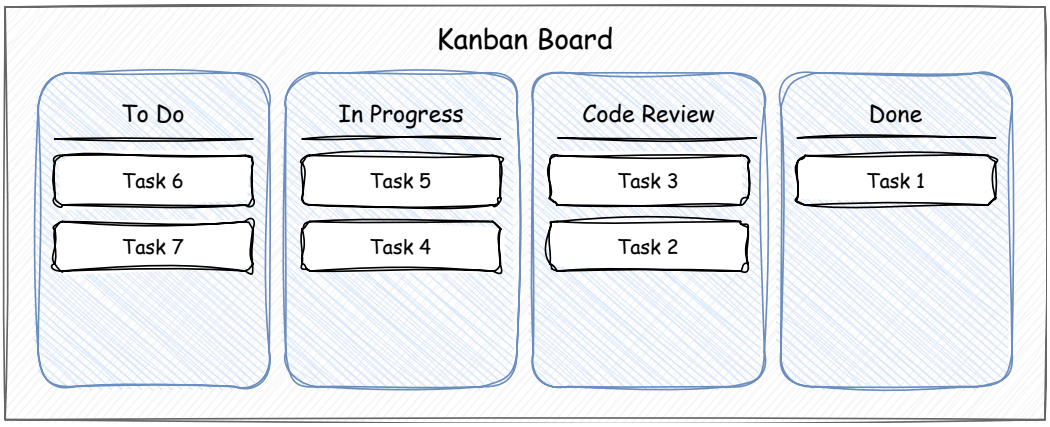

Figure 1: Kanban Board Example. Adapted from [28]

The Kanban board presented in Fig. 1 doesn't necessarily have to have just four columns. The number of columns will depend on the number of phases in which the team works better. In this example, there are the following columns: "To Do" which encompasses all the tasks still to be done, where the tasks are well-defined; "In Progress" which contains the tasks in the implementation phase; "Code Review", a good practice that allows to check whether the behavior of a given functionality is as expected; and "Done" column, where the card ends up when the task finished [28]. So, the tasks on this type of board flow from left to right.

2.1.2 *Git*

Git is a VCS that allows developers' teams to track changes made to their project code over time [29]. It is considered a major practice in professional software development, but it can be also used with non-software projects. It is a powerful tool, and it has the following benefits:

- It allows the developers to record all software changes as a snapshot of the work over time, which is called commits, in this way, is possible to check the old version of all files of the project.
- A Tool that helps the developer's teams to work more efficiently and faster because several members of the team can work in parallel on different features of a project, synchronized with already developed software. This is possible because this VCS can synchronize the project in a cloud repository, which is then shared between several team members [30].
- It contributes to DevOps workflow because has mechanisms that allow the project to be tested and deployed efficiently while the development process continues [31]. In this way, it is possible to improve the delivery of functional software to a company's customers.

Therefore, a Git local or cloud repository has several elements, such as the commits, which, as mentioned, are snapshots of files modified at some point. The commit has the timestamp and description, which is used to describe the project changes. Another element is the branches, which allow the teams to create different versions of the project under development, often serving to develop new project functionalities in an isolated way. In addition, the branches can be generated from others, as illustrated in Fig. 2.

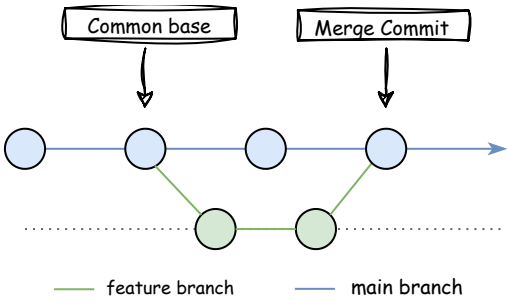

Figure 2: Git Workflow Entities.

In addition, there are several Git branching models, which define how a project team works over the branches, therefore, these models define the essential branches

and purposes of each one. The most popular models are: Git Flow [32], GitHub Flow [33] and GitLab [34]. One of these used by the GitHub platform is GitHub Flow, which has the branches Main/Master and n branches of Feature type, as presented in Fig. 3.

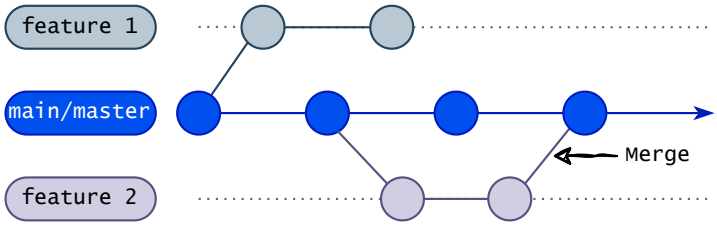

Figure 3: GitHub Flow Schema.

Therefore, according to Fig.3, when there is a new feature to be implemented in the project, a new branch is created with a descriptive name [33]. It used to derive from the master branch, which has a deployable and reliable project version. After that, all changes in the branch should be committed and updated constantly in the cloud project repository in the same named branch. When the feature is finished and tested, it can be added to the deployable version (master branch) from a merge procedure. It can be made from a pull request, which makes the changes go through a validation process. Finally, once the feature is approved by project managers, it is merged into the master branch. Ultimately, the master branch can be deployed in the production environment or not, depending on the requirements.

2.2 NETWORK PROTOCOLS AND COMMUNICATION ARCHITECTURES

Once the present solution needs communication between several modules, several network protocols should be chosen. These should be selected considering the requirements found in this industrial sector. For example, the synchronous network protocols were considered to allow static data to be changed between modules. Thus, a style of software architecture called Representational State Transfer (REST) is analyzed, as it offers more scalability and flexibility when it comes to handling user requests than other technologies such as Simple Object Access Protocol (SOAP) [35]. Also, asynchronous network protocols were considered to meet the demand for the efficient share of real-time data of the machines with clients.

2.2.1 *Representational State Transfer (REST)*

REST is a concept that describes a software architecture to implement an Application Programming Interface (API) that follows a client/server structure [36]. This structure is Operating System (OS) and programming language independent. It demands the use of standards like Hypertext Transfer Protocol (HTTP) to communicate with clients, and Uniform Resource Locators (URLs) for clients' application resource access (e.g., text file, HTML page, image, video, or any type of data) [37]. In addition, an application with this structure is called RESTful API, which follows a Peer-to-Peer (P2P) communication, based on the request/response model. Additionally, since the HTTP protocol is a requirement of this type of APIs and operates in a stateless way makes that each request to RESTful API from a particular client is handled independently of the previous ones. In addition, the data should be transmitted between API and the client in formats like JavaScript Object Notation (JSON) or XML [36, 37].

Regarding the requests done by clients, they have to specify their intention for a particular resource when they request a REST application using HTTP methods, such as GET, POST, PUT, and DELETE. In reverse, the server application should answer with a three-status code that indicates whether the request was well-processed. There are the following code status ranges:

- 2XX - code range that indicates success;
- 3XX - code range that indicates URL redirection;
- 4XX and 5XX - code range that indicates errors.

2.2.2 *WebSocket*

The WebSocket technology enables asynchronous communication between client and server, a long-lived connection. This communication can be used to support event messages without the client having to send requests to the server to obtain messages. It only has to initialize the communication from the handshake mechanism [38, 39]. Therefore, this protocol doesn't follow the request/response communication model. This way of communication is supported by several web browsers used by the community and JavaScript (JS) platforms, such as Node.js.

Standardized by Internet Engineering Task Force (IETF) organization under the standard RFC 6455 [40]. This protocol was developed to allow share real-time data

between servers and clients, therefore, it can be used by Internet of Things (IoT) applications, which demand a high data rate [37, 39].

Regarding the procedure for communication via WebSocket protocol, the client exchanges HTTP messages with the server to establish the communication, as shown in Fig. 4, as "Opening Handshake" phase. After that, payload messages can be exchanged among the client and server via WebSocket protocol, in full-duplex mode, while the session is active. This way, the two intervening nodes can send messages at any time and at the same time to each other. Finally, when any node involved in communication initiates the closing handshake, several messages are exchanged to end the session accordingly.

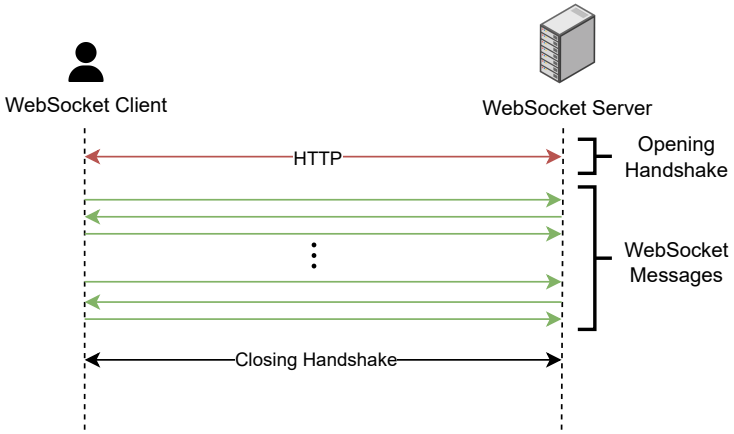

Figure 4: WebSocket Communication Flow. Adapted from [37].

2.2.3 *Open Platform Communications Unified Architecture (OPC UA)*

In 1996, an interoperability standard called Open Platform Communications Classic (OPC Classic) was released by OPC Foundation to provide a common platform for communication between HMIs systems and Programmable Logic Controllers (PLCs) [41]. These controllers allow the management of the industrial processes, i.e., allow communication with industrial machines or manufacturing entities in an industrial context for managing them. The problem with them is that they communicate via several industrial protocols (e.g. Modbus, Profibus,...), making it hard to connect to other modules. Therefore, it was created the OPC Classic protocol, thereby enabling applications to communicate in a unified way with machines. Thus, the applications can send requests to OPC Classic wrapper as presented in Fig. 5 with the name "OPC Classic Wrapper", which is prepared to translate OPC Classic write/read requests to specific device communication protocol without too much

effort on the part of the applications, working on a client-server architecture way. This standard, which defines a communication protocol and architecture, allowed features like real-time data access, historical data access, and monitoring of alarms and events.

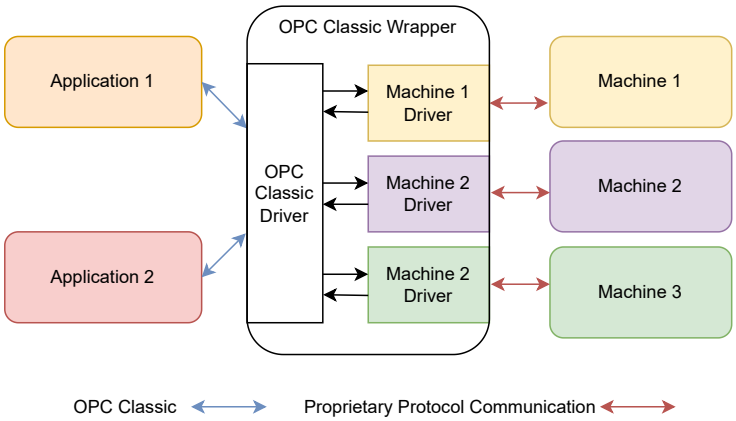

Figure 5: OPC Classic General Architecture.

However, the specification brought some disadvantages to the industrial sector, where it was created to run on the Windows platform, leaving the other OSs aside. Besides, the OPC Classic protocol had another handicap, where for each feature there are different specifications, which define the interface for server and client exchange data, as presented in Fig. 6 [42]. This aspect requires several implementations, causing some effort for the industries. There is the OPC Data Access (DA) specification that allows the data exchange in real-time in a specified format. Another specification is called OPC Alarms and Events (A&E), which defines the data exchange about alarms and events, which this kind of events are defined in the A&E server from real-time data available. The last important specification is called OPC Historical Data Access (HDA), which stores historical data and makes them available to OPC Classic HDA clients. An aspect of these specifications is that they all derive from the OPC DA specification to connect to OPC Classic servers.

In order to unify all OPC Classic functionalities in a unified service and platform independent, the OPC UA was created, an open specification (IEC 62541) that defines a service-oriented unified architecture that allows a unique OPC data model, which is called AddressSpace, where all OPC UA clients can access to data of industrial physical entities to make decisions and control them [43]. This architecture allows the clients to communicate from Client-Server and Publish-Subscribe (Pub-Sub) communication models. The first one enables clients to retrieve

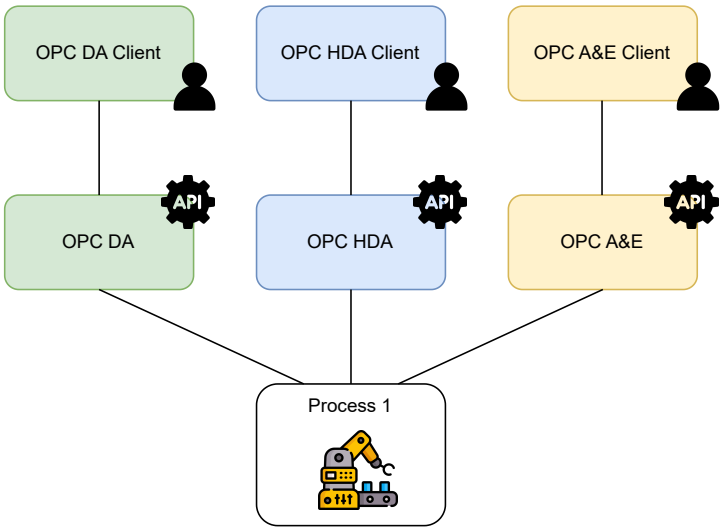

Figure 6: OPC Classic Specifications. Adapted from [42]

data from a device in a request-response model and the latter allows multiple clients to receive notifications when device parameters change [44].

Regarding the AddressSpace that each OPC UA server provides, it save several nodes that store data about the manufacturing entities [45]. Each node contains several attributes and references (Fig. 7), which depend on the "NodeClass" used to instantiate it. The attributes describe the node, for instance, the attribute "BrowseName" is a unique identifier of the node and allows your searching. The references allow to link nodes to each other.

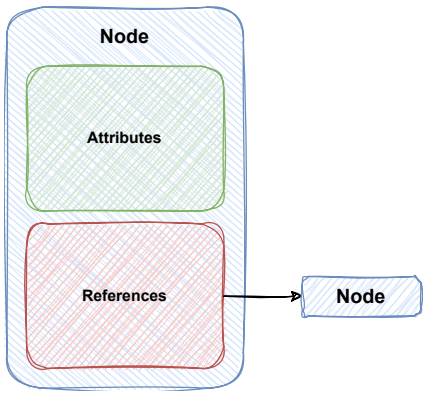

Figure 7: Node Model. Adapted from [45].

There are several types of node classes, where from them, it is possible to create several kinds of nodes in AddressSpace of an OPC UA server. The most important node classes are the following [46]:

- Object - used to represent parts of the physical device;

- Variable - node class that allows to the system specify its important data, dynamic and static. Instances of this node class belong to an Object node;
- DataType - node class that allows to define types of value stored by Variable instances;
- Method - node representing a method that can be called with some input values;

An Object node instance, as aforementioned, contains a set of Variable nodes as illustrated in Fig. 8, which can be of two types: Property or DataVariable. A DataVariable node represents the content of the Object and can contain dynamic data, e.g., a value of a sensor of some machine in a factory, while a Property node is more used to store static data, e.g., a measurement unit of some DataVariable node. The Property node can not belong to another Property node, so it can only belong to other nodes like Objects and DataVariables. Finally, an Object node can also include methods that can be called to change the state of the device linked.

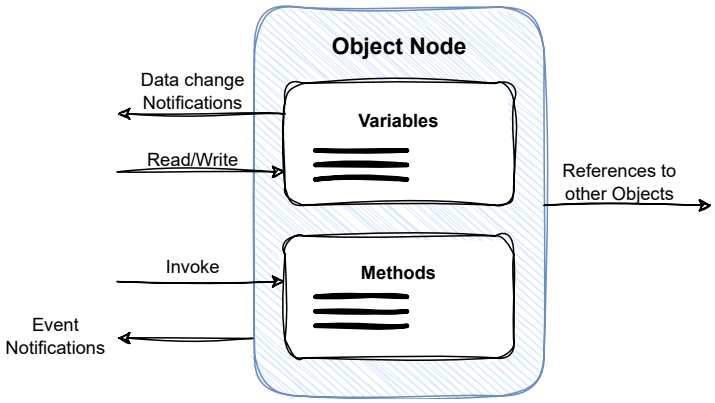

Figure 8: Object Node. Adapted from [45]

2.3 FRONTEND FRAMEWORKS

As previously mentioned, the solution will include a web application. Thus, technologies that can accelerate the development of such applications should be considered. This section explains the technologies that enable the development of JS applications as well as the technologies that support them and make them available to end-users.

2.3.1 *Vue*

Vue is a JS web framework that allows the creation of browser-accessible user interfaces [47]. This framework, which is in the third version, enables the creation of web applications from standard languages such as Hypertext Markup Language (HTML), Cascading Style Sheets (CSS), and JS. One of the advantages of this framework is a large documentation with many examples, so that novice Vue developers can learn and implement web applications. In addition, it offers good functionalities, such as the following:

- Single-page Application - It's the ability of a web application to only load one page and according to user interactions or data acquisition from APIs, refresh parts of the page that have to change without the need to load the entire page [48]. This brings advantages to the web application, like more performance in the loading page process, and consequently, a better user experience;
- Components - It is a functionality that allows the developers to create several pieces that can be reused into several HTML pages, which requires less implementation effort [49];
- Lifecycle Hooks - Each instance of the certain component has your Lifecycle, which includes stages such as "mounted", "updated", and "created". In each stage, the developers can manipulate responses to them (e.g., in the "created" stage, it can be retrieved data from such API, so that when the Document Object Model (DOM) of the page is rendered, the data already exists) [50];
- Intuitive Interfaces - This technology can be integrated with other JS packages that already have created components, helping improve applications' User Experience (UX) [51].

Furthermore, this front-end JS framework should run over JS engines, like Node.js ([52]), which creates a server-side environment, so that applications become available to web users, as represented in Fig.9. Vue usually creates an instance of the application for each client/browser window, distributed by Node.js.

In terms of support, it is a technology that has been improved ([53]) and has been used by developers many times, where a sign of that is the number of downloads. In 2023 there were around 467119728 downloads, according to statistics extracted from Node Package Manager (NPM) API [54].

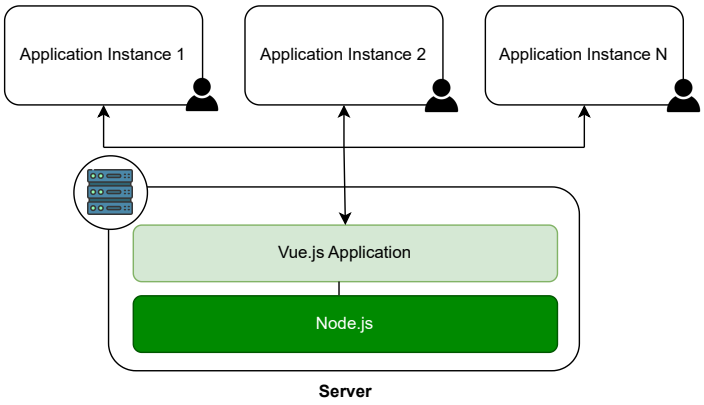

Figure 9: Vue over JS engine.

2.4 BACKEND FRAMEWORKS

Since this solution requires server-side applications to support user applications, related technologies are considered. These applications can be used to manage data in databases and can also act as a bridge between user applications and IoT devices, such as industrial machines. In addition, technologies that enable microservices approaches are covered, as these allow applications to be scaled and adapted quickly to production environments.

2.4.1 *Django*

Django is an open-source Python framework that allows the creation of web applications [55]. It has been used by reputed organizations, such as Instagram, Mozilla Firefox, Pinterest, and National Geographic [56]. This technology was thought for developers to create their applications quickly and with a certain robustness. It offers a structure to easily implement REST APIs ([57]) and WebSocket servers, which are called Django Channels [58]. Moreover, this framework offers built-in features such as the Object-Relational Mapping (ORM) mechanism, which allows data to be easily retrieved and persisted in relational databases [59]. In addition, this framework enables the integrations of other plugins such as user authentication [60]. Regarding the project folder structure in Django, it follows the folder structure pattern presented in Fig.10 [61].

As presented in Fig.10, the Django project has a root folder, which usually has the project's name. Moreover, there are the "app" subfolders, which represent the several applications included in the project, e.g., the project can have REST application,

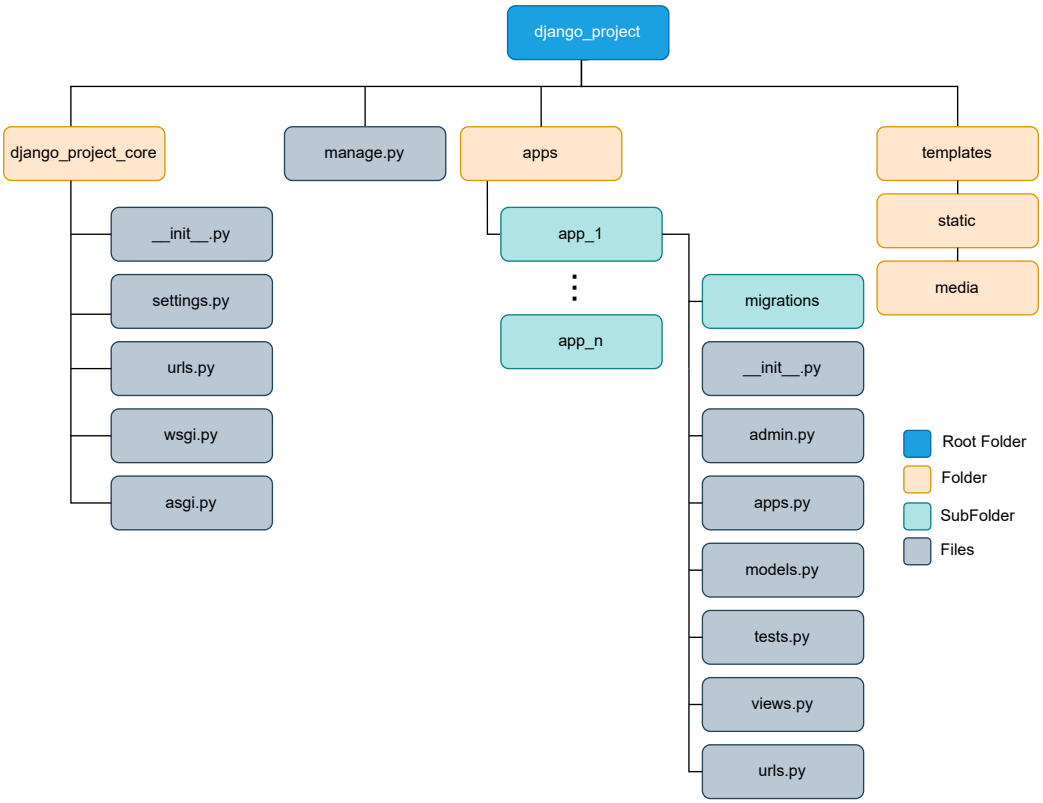

Figure 10: Django Project Folder Structure. Adapted from [62].

and another with a WebSocket application, using Django Channels. In addition, there are important files within the "django_project_core" folder, which allows developers to configure the global settings of the project. So, this subfolder has the following files [61]:

- django_project_core/setting.py - responsible for configuration of the main settings of the respective Django project, e.g., used to include applications that were created, plugins, network access settings, and more;
- django_project_core/urls.py - defines entry-point URLs for Django Web application, each one matching a specific application. This way, messages are processed by the application they are intended for.
- wsgi.py - the entry-point used by WSGI-compatible web servers to create one instance of the Django application, where the instance accepts requests and forwards them to the respective sub-applications to be replied;
- asgi.py - the entry-point used by ASGI-compatible web servers to create one instance of the Django application, where the instance accepts requests and forwards them to the respective sub-applications to be replied. The difference with the Web Server Gateway Interface (WSGI) instance is that it allows

clients to communicate via the WebSocket communication protocol, in addition to the HTTP protocol;

Moreover, so that the Django applications can communicate via the internet through HTTP web servers, two kinds of interfaces were created, one of them allows synchronous communications (where there is a request and a response, and after that, the connection terminates), and the other, that allows asynchronous communications (in the same connection there are several messages exchanged between client and server, which is usually a long communication) [63]. The first one called WSGI, stands for Web Server Gateway Interface. It allows the receiving of the HTTP requests from clients, firstly received by the HTTP web server, and after, redirected to the Python application through the WSGI interface. The problem with this first interface is that doesn't allow long-lived connections (i.e., asynchronous communications), making it impossible to use protocols such as WebSocket [63]. So, the Asynchronous Gateway Interface (ASGI) was created to also support that kind of communication, which has become a focus of web programming [64]. Moreover, to configure these interfaces the "asgi.py" or "wsgi.py" files should be configured depending on the specification needed. Considering these interfaces the possible general architecture of the Django projects is presented in Fig.11.

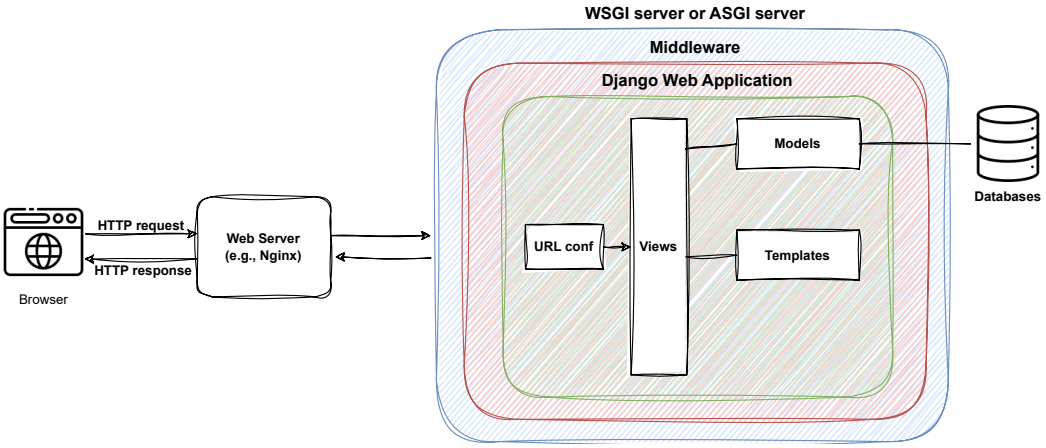

Figure 11: Django Generic Architecture. Adapted from [65].

As shown in Fig.11, the communication workflow starts with the user via Browser sending a request to reach the Django application, but first, it is caught by the Web server, which redirects the message to the ASGI or WSGI server. The choice between these kinds of servers should be based on the above reasons. After that, the request can be processed by middleware, such as Authentication Middleware, which validates the user's login. On the other hand, there can be several types of middleware processing requests with different objectives [66]. After passing through

the middleware, the request reaches the Django Web application. After that, the request is forwarded to the correct view (an entity that contains the business logic code to process the requests [67]) of one specific application, according to the route diagram defined by URL configuration ([68]). During this request processing, data can be persisted into the database via the ORM technique, using Models (a unit that can represent a certain object in the database, persisted or not [59]). They are defined in the "models.py" file, which is within the application folder. Finally, with the processed request, the Django Web application replies to the client with the response message following the reverse path. The response can be only data, like JSON or HTML pages via templates [69].

2.4.2 *Docker*

Docker is software that allows virtualization at the OS level, creating favourable environments for running applications with different dependencies in the same host [70]. Each environment is called a container and usually runs one application. Each container is created from an image that can be created manually or extracted from Docker Hub [71], which already brings the dependencies configured to run some sort of application. The containers work over the Docker engine, as seen in Fig.12, and all the containers share the same hardware and resources with the host OS [72].

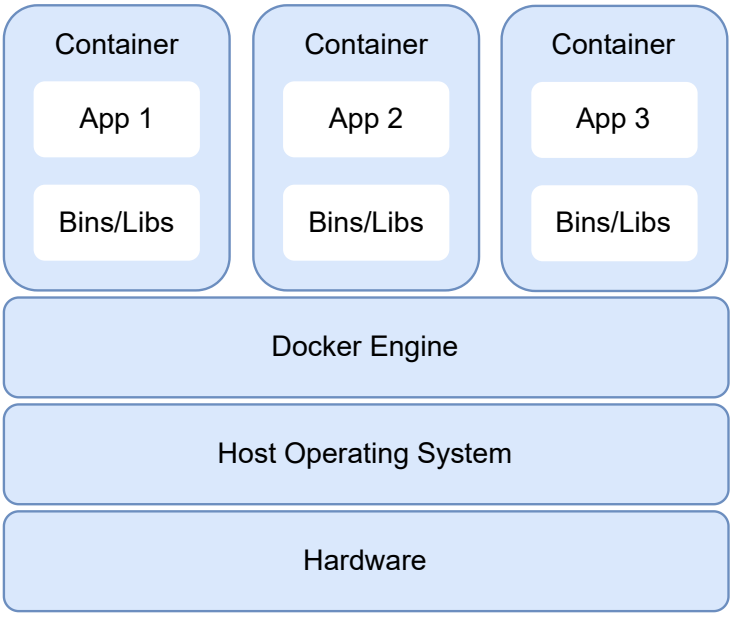

Figure 12: Generic Docker Architecture. Adapted from [72]

This technology can bring several advantages to applications, developers, and enterprises, such as the followings [73, 72]:

- The deployment of the applications is faster than other options, like Virtual Machines (VMs) because the image that gives rise to the environment of the execution of the application has to be configured one time to be replicated in any computer;
- As the containers work within an ambient portable and run on any operating system over Docker, they can be deployed in a variety of environments in development stages, keeping the normal application operation, such as development (the developers write code and test it in a replica production environment in their computers), staging (application test environment), and production (when an application becomes available to the users). Thus, the work becomes easier for development teams and more agile for enterprises;
- Decreases the degree of effort to enterprises to migrate the applications to a cloud environment because most of the cloud platforms offer resources to support containers;
- Containerization can be useful to keep the redundancy of applications due to the usage demand, that's why there are technologies like Kubernetes ([72]), which allows configuring the number of replicas of a certain container.

2.4.3 *WildFly*

WildFly, formerly called JBoss, is an application server that can make available Java Applications (JAs) on the Internet, more specifically, applications that follow the Java language specification, Java EE, now Jakarta EE. These specifications define protocols, APIs, annotations, and services to promote the environment for creating business applications, accessible via the network. Therefore, WildFly aims to help development teams build and run JAs quickly and efficiently [74]. Additionally, this technology is valuable for developing REST APIs.

One Java EE application is divided into 3 principal layers, as represented in Fig. 13 [75]. The first layer called Client Machine, represents the client side, which can include applications that request and receive responses from JavaEE Server via the sublayer Java Application Services. Also, the client can receive pages HTML that must be rendered on your side, which come from the aforementioned sublayer. Moreover, the sublayer Business contains a few units of processing, the beans, that

process all requests from clients via sublayer Java Application Services. After that, it can store data or retrieve data from the layer EIS, which stands for Enterprise Information System that represents the databases.

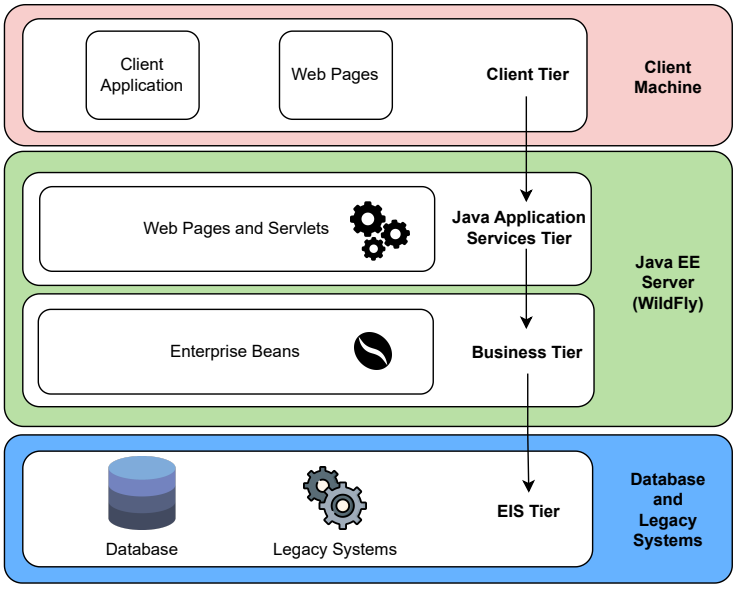

Figure 13: WildFly and Java EE General Architecture. Adapted from [75].

2.5 DIGITAL TWIN

The concept of DT has been studied by several business sectors, such as health and manufacturing [9, 76]. The concept has been a topic of discussion because the definitions of that can vary between researchers. However, the definition most accepted by the community was presented by National Aeronautics and Space Administration (NASA) [77], in which a DT is "an integrated multiphysics, multiscale, probabilistic simulation of an as-built vehicle or system that uses the best available physical models, sensor updates, fleet history, etc., to mirror the life of its corresponding flying twin" [78]. Although there are different definitions, they have commonalities, such as the fact that there has to be a physical system, i.e., a Physical Twin (PT) (e.g., one machine or production line with several machines), and your Digital Representation (DR) mirrors the behavior of the PT. The data comes from physical entities from attached sensors and are forwarded via the network to a DR. On the other hand, any changes to the DR will be conveyed to the PT. Therefore, there is a two-way communication between the two entities.

According to Kritzinger et al. [8], the DTs are "digital counterparts of physical objects". In turn, this author divides this concept into three subcategories, according

to data integration between DR and the PT, these are presented in Fig.14. The first subcategory classifies the DR as a Digital Model instead of DT when there is no data exchange automated in the two directions, i.e., the DR doesn't mirror any behavior of the PT in real-time, and any change in DR doesn't reflect in physical space directly. The second definition classifies a Digital Shadow when the DR reflects the behaviors of the PT, automatically, but the reverse flow doesn't exist. Finally, the DT classification is when there is a two-way communication, where the changes in DR or PT are reflected in your counterpart.

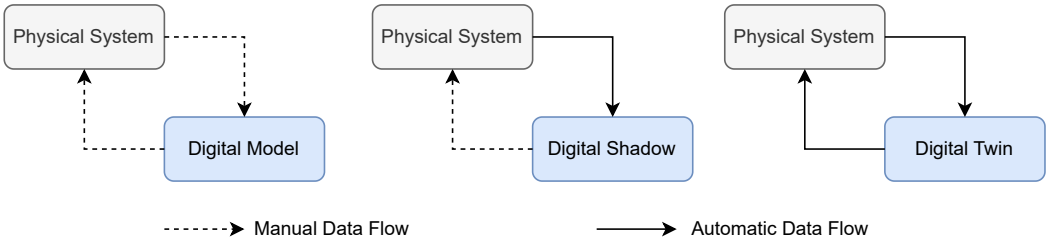

Figure 14: Digital Twin Subcategories. Adapted from Kritzinger et al. [8].

The DT technology can offer several services to help decision-making taken by the enterprise stakeholders, such as:

- Obtaining information related to the lifecycle of a product from the real-time and historical data extracted from it, deriving valuable information to optimize production and the product [79];
- In the context of industry, it can be used by operators/enterprises to follow the production process of physical entities, to maintain the normal production process, or to understand certain aspects, such as failures. There are services that allow emulation over 3D models of the working process to help operators understand the production flow [80, 20];
- The DT can help enterprises optimize the production line process, either the level of flexibility or performance, through the dynamic simulations from data retrieved [80, 81, 82];
- This technology also can be used to keep the machines operational with maintenance in time from predictive analysis [80, 81].

The DT concept can be illustrated as in Fig.15, in which the PT makes its processing data available to DT, and with the help of its services the enterprise takes data-driven decisions, which subsequently, will be reflected in physical space from DT.

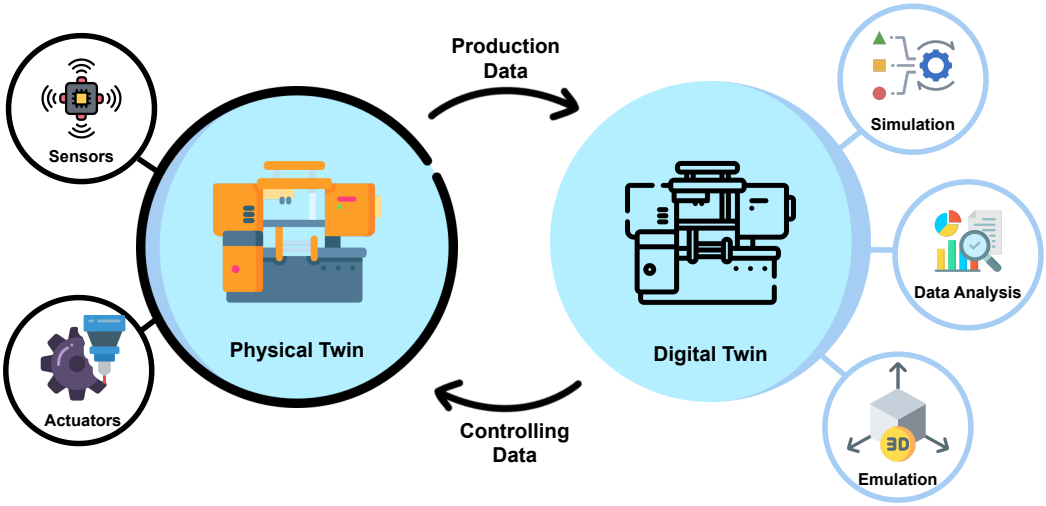

Figure 15: DT General Architecture.

3

RELATED WORK

This section presents various academic and commercial solutions found in the industrial area to improve production lines. The academic covered papers are the result of several searches in scientific databases with keywords such as “Digital Twin”, “Industry 4.0”, “Internet of Things”, and “OPC UA”, between the years 2017 and 2024. The commercial solutions were found in academic articles and from searches on web search engines. In addition, the technologies and approaches employed in these works are discussed so that advantages and disadvantages can be known.

3.1 SCIENTIFIC RESEARCHES

The present section presents several scientific researches from platforms such as Google Scholar, Scopus, MDPI, and IEEE Explorer. Many of these studies mention the concept of DT, not all implement the full features of a DT system [14], and some only propose theoretical architectures to support this technology [10, 18]. However, these solutions are valuable as they make it possible to identify gaps that have not been overcome, as well as to understand the difficulties encountered by researchers in implementing this concept.

In [20], Redelinghuys et al. proposed a 6-layer architecture with a case study implementation. This architecture was inspired by a 5-layer architecture, which intends to create a virtual space/DT and a real space/PT, where there is communication between both spaces. In addition, it aims to adapt to legacy systems and different vendor interfaces and enables emulation (action of replicating in near real-time the behaviour of a physical system digitally), simulation (predicting the future of the physical system over certain conditions), analysis, and decision-making support based on historical data. The architecture proposed by the authors is presented in Fig. 16 with a set of layers.

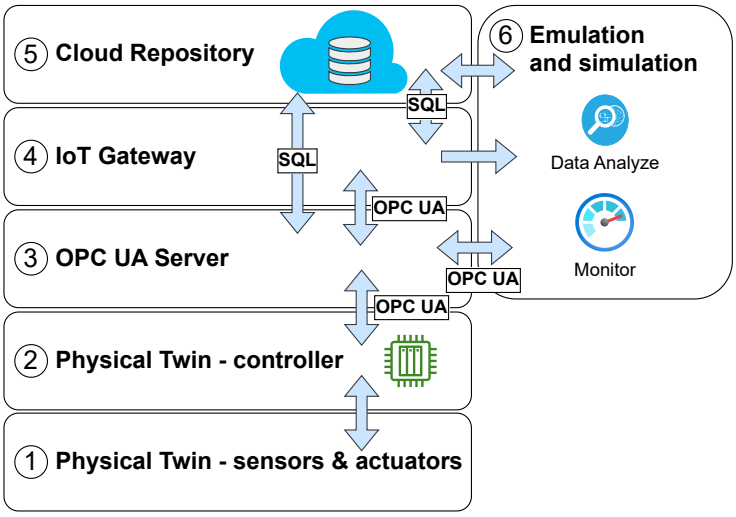

Figure 16: 6-Layer Architecture for Manufacturing. Adapted from Redelinghuys et al. [20].

This architecture described is intended to be applied in any industrial context, where the layers represent parts of a complete industrial system, each one with its own well-defined function. Thus, these layers are explained below:

- **Physical Twin** – It represents physical entities that execute manufacturing processes. Each physical entity includes sensors and actuators. Therefore, these entities referred belong to layer 1. Data can be exchanged with physical entities from controllers/data acquisition devices, where these edge devices belong to layer 2;
- **OPC UA Server** – It corresponds to layer 3, and it is the bridge between Physical Twin (Layer 2) and upper layers, providing a vendor-neutral communication interface, where in this case, is used the OPC UA protocol, which facilitates vertical and horizontal interoperability. This layer sometimes has to be adapted to the Physical Twin protocol because some machines on the shop floor are provided by different suppliers, which leads to different communication interfaces in the production line;
- **IoT Gateway** – The fourth layer contains the application that can communicate via OPC UA protocol with Physical Twin from layer 3. In addition, this module can process the data to make it more valuable to upper layers, and after that, persist them in Layer 5 (Cloud Repository);
- **Cloud Repository** – Represents the fifth layer, which aims to store all sensing data received from layer 4, thus the historical information of the physical twin can be accessed via layer 6. This feature can help industries improve their decision-making;

- **Emulation and Simulation** – In this layer, software capable of simulating and emulating are considered, so that the behavior of the Physical Twin can be replicated digitally, for instance, through a 3D representation. Furthermore, these software programs can help enterprises to predict behaviours of the assets from experimental conditions. The authors used the Siemens Tecnomatix Plant Simulation software, which enables low-level data retrieval from cloud database (Layer 5) and OPC UA Server (Layer 4).

Finally, the solution presented by Redelinghuys et al. was developed and tested on a robotic gripper, a physical asset that is part of a manufacturing process, using the OPC UA protocol, which shows that the solution is prepared to communicate with physical entities. However, this implementation demands a high configuration always that a new machine has to be included in the digital space which may not be so comfortable for enterprises with low financial resources.

Another 6-layer architecture was proposed, but this time by Hazrathosseini et al. [10], which aims to support the mining industry. That architecture was neither implemented nor tested. The first layer (Physical Space) presented in Fig. 17 covers the machine and any physical asset. The second layer (IoT Gateway) encompasses the controllers, actuators and sensors, which are copulated with the physical assets, like trucks and processing plants. The third layer (Cloud Repository) receives and stores raw data and pre-processed data from the previous layer via communication technologies. Layer 4 (Virtual Space) is responsible for mirroring the behaviour of the physical assets through the real-time data received from the third layer. This process as suggested can be carried out using platforms such as Azure Digital Twins and Ansys Twin Builder. The next layer, called Cognition, is where the simulations, predictions, optimizations, decision-making and diagnostics are made through data mining and AI algorithms. These optimizations and decision-making are reflected via layer 3 in physical assets. Finally, layer 6 is responsible for generating reports so that engineers and managers are aware of the manufacturing operations.

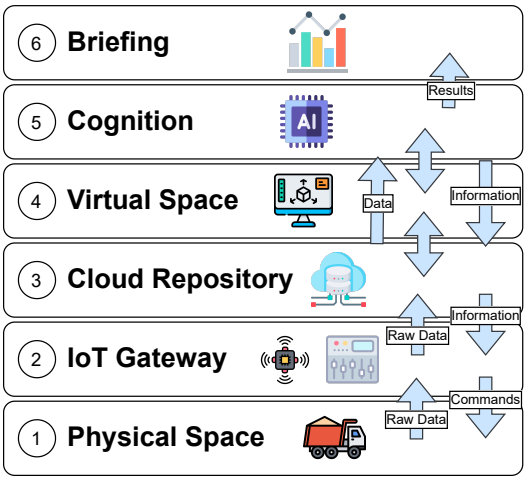

Figure 17: 6-Layer Architecture for Mining Industry. Adapted from [10].

Habib et al. [17] proposed another solution for managing physical space. Its architecture is depicted in Figure 18. This presented solution was capable of aggregating information from various information sources, i.e., from devices that communicate from different communication interfaces. The module responsible for communicating with those devices was called Server OPC UA, a central server, and it uses the protocol for sharing the related information with OPC UA clients. These clients can be vendor software and any module capable of communicating via OPC UA protocol. One this type of client was created and was named REST Based OPC UA Middleware, where it's responsible for extracting all real-time values of several devices periodically from the server and sharing them with compatible Restful APIs, more specifically with 3 web applications, to be monitored. This implies that there is a data transfer rate between the Middleware and the web applications, which can cause unnecessary communications on the network and delays in receiving the values of device parameters when they change.

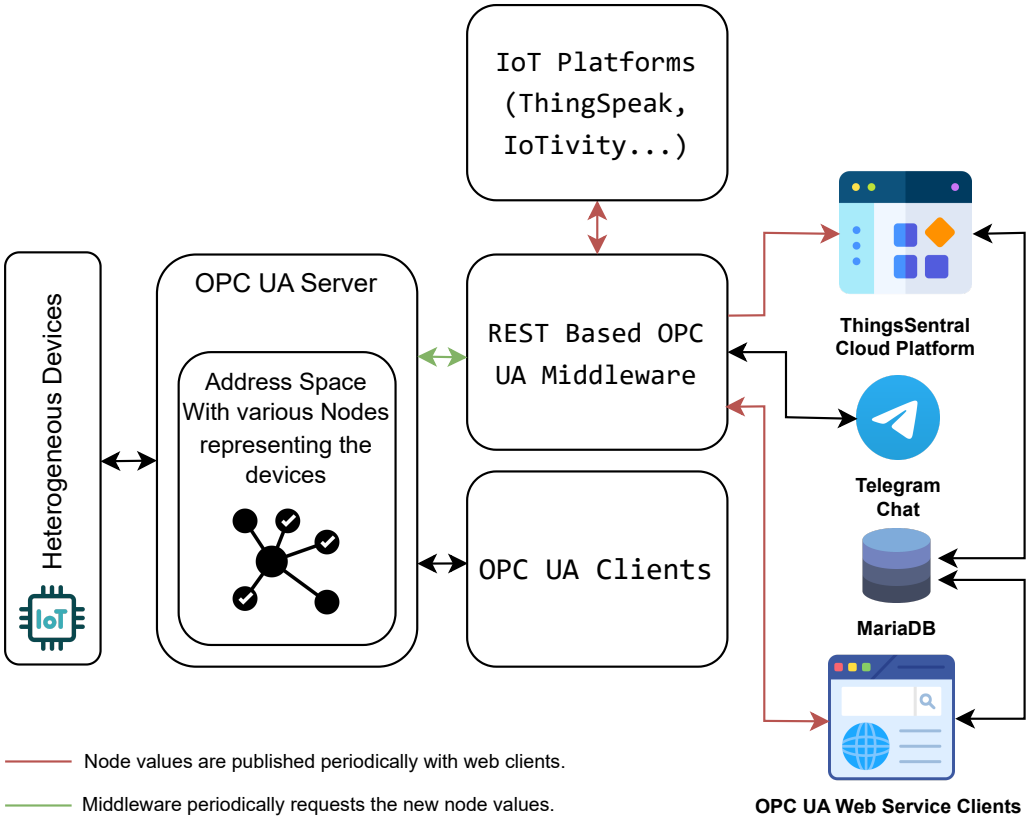

Figure 18: Rest Middleware Solution Architecture. Adapted from [17].

Other solutions follow the same approach to implement a central OPC UA server to distribute information about several physical entities [16, 83, 84, 85]. A notable solution with this approach was presented by Liua et al. [86]. The authors created a single OPC UA server to monitor several industrial machines. Some of these machines communicate via the OPC UA protocol, while others use the industry standard MTConnect protocol. In this manner, the OPC UA server was adapted to receive information from machines that communicated via the MTConnect protocol. In addition, an AR application was developed to improve the UX compared to traditional 2D control panels and to improve understanding of the manufacturing process. The application was used with AR glasses called HoLolens. The solution allowed users from AR glasses to monitor the various status of the machines from their sensorial parameters, e.g., cutting force, axis position, vibration, and power status. Furthermore, the solution could simulate different statuses of a machine.

Additional services, such as data analytics, remote monitoring, and storage, can be integrated into DT architectures using cloud services. These functionalities allow companies to prevent future events and optimize production. Furthermore, the use

of these cloud services enhances their ability to integrate new emerging technologies, thereby optimizing business processes.

A solution linked to cloud services was presented by He et al. [87], where the architecture is presented in Fig. 19. This solution provided a digital representation of a physical device, more specifically, an experimental tower crane, allowing real-time monitoring and the possibility of processing simulations of possible conditions. Moreover, the physical entity included external sensors, in which the values measured by them were made available over the Internet. These values were collected and analyzed in the following Cloud Computing Platforms (CCPs): Firebase Real-time Database and Microsoft Azure, respectively. The authors noted that the AR application allowed users to monitor the digital entity and helped to increase the operator performance in aspects such as the best reaction to abnormal behaviour of the monitored entity.

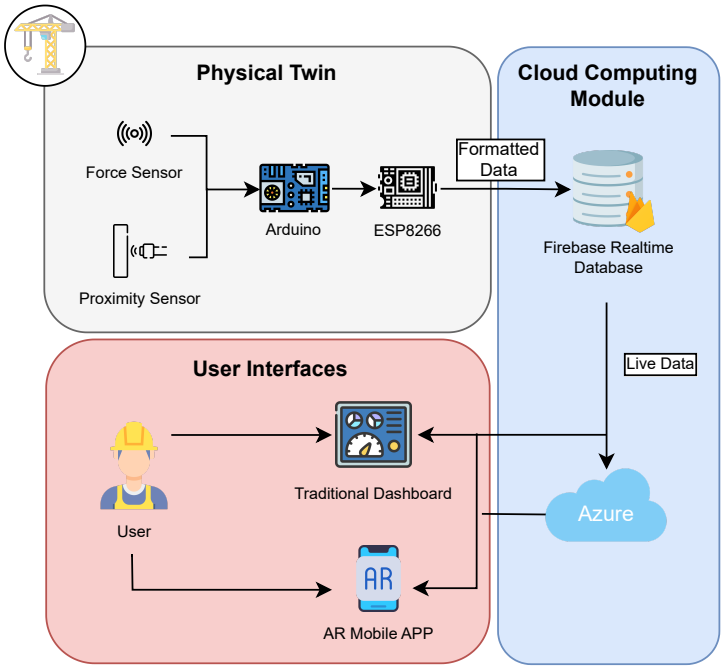

Figure 19: Tower Crane Solution Architecture. Adapted from [87].

Another solution that used cloud services technology was presented by Mourtzis et al. [81] so that sensor values from various machines of a mould factory could be consulted remotely and anywhere. Also, the authors implemented a predictive maintenance algorithm based on the mean time between failures of each machine so that the stakeholders can carry out maintenance in time. Furthermore, the engineers and operators were able to consult the different states of each machine from a platform (e.g., busy mode, idle mode, down mode...) that received values of external wireless sensors attached to each machine. Another relevant feature of this solution

was the ability to respond to unexpected events in production, e.g., a machine that stops due to a fault, so it is in down mode and, in response, the system efficiently can reschedule production tasks automatically and quickly. This procedure reduces production downtime, and the time the people involved would lose doing the same rescheduling process manually.

In addition, more algorithms have been created to aggregate more services to DT solutions [88, 89], such as the algorithm proposed by Polini et al. [13], which aims to predict the wear of cutting saws and milling tools, using sensory data.

A related solution had the goal of detecting faults in physical space and optimizing the management of the resources on the production line using AI algorithms was presented by Răileanu et al. [90]. The authors of that solution presented a 4-layer architecture (Fig. 20), where the first covers all sensors and actuators, which generate and receive data. The second layer is a gateway that forwards the data to the upper layers. The third layer aggregates all essential extracted data. The last layer helps in the analysis and decision-making processes. So, with this architecture, the authors could create one DT of the factory floor conveyor system, which contained several stations to be monitored, where the products passed to be assembled, and some behaviours were predicted. Therefore, various services can be added to developed DT solutions to add value to companies' production lines, where some may be more suitable for certain application areas than others.

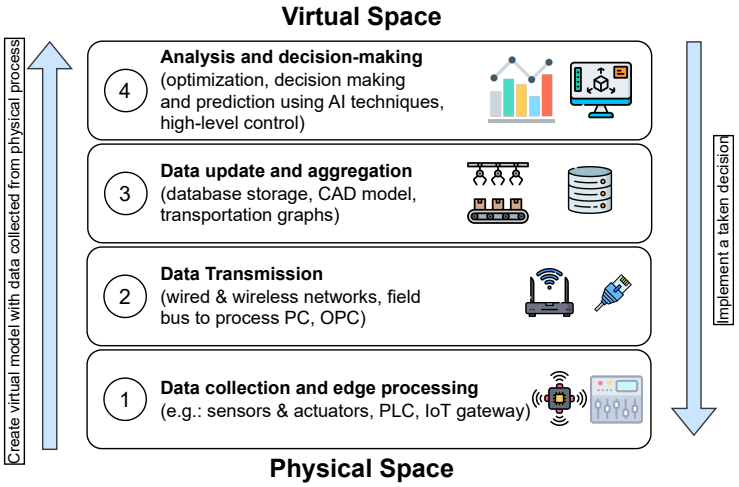

Figure 20: 4-Layer Architecture for Production Line Conveyor. Adapted from [90].

A notable solution with a different approach to those already mentioned was presented in [91]. The authors created a solution around OPC UA communication protocol. A solution created a central entity, presented in Fig. 21 with the name OPC UA Web Platform, capable of distributing information related to various

OPC UA servers through synchronous and asynchronous communication. The synchronous communication was done from a REST API, and the asynchronous from an Message Queuing Telemetry Transport (MQTT) broker, which works over a Publisher/Subscriber model. The solution had the advantage of being compatible with client applications where the communication follows the widely used REST pattern. In addition, the solution enabled user applications to receive notifications from the MQTT broker when the values of physical entities changed, where the users initially had to identify one broker and one topic to the Middleware module publish those data, and then connect to the broker. This system also facilitated the process of connection with OPC UA servers as the users don't need to know that specification because the Middleware treats that automatically, whereas the user applications only needed to know the machine and the parameters to monitor. Finally, the authors focused on the core module, OPC UA Web Platform, and not on web applications, so they did not present any user application.

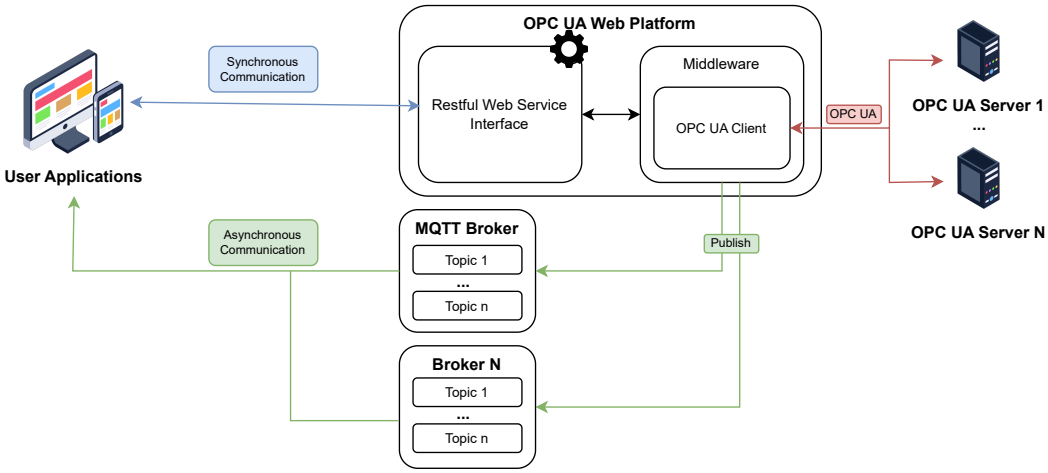

Figure 21: Middleware Solution Architecture. Adapted from [91].

3.2 COMMERCIAL SOLUTIONS

A search was made for licensed solutions, some of which were found through scientific research and others through web search engines. Moreover, it should be noted that these solutions imply monetary costs for companies, which for certain companies may not be attractive.

Prosys enterprise has been in the market for a few years, focused on building software around OPC UA technology, and has launched several products for the I4.0. One product called OPC UA Forge shows great potential for overcoming

the difficulties experienced by companies [92]. This product, still in version 1.2.0, released on 10.06.2024, involves a central module capable of providing information about the underlying data sources to the Internet Technology (IT) systems, such as Manufacturing Execution System (MES), Enterprise Resource Planning (ERP), and IoT, as presented in Fig. 22. In addition, this module can extract the information via industrial protocols, such as OPC UA and MQTT, and make it available to the other systems via the same protocols and also via a REST API. The notable features of this solution are the following [93]:

- Through a GUI, the possibility to integrate new sources of information, and aggregate them into a unified address space, which allows data to be made available to IT systems, but demands some knowledge;
- Generation of events according to the values of the elements integrated in your address space, subsequently forwarding them in the form of notifications to the various communication channels connected to the clients;
- It incorporates communication patterns such as Publish/Subscribe that allows clients to receive data in real-time. This based asynchronous communication is a better approach when a system relies on real-time data from another system, ultimately reducing network traffic, unlike other patterns like REST architectural standard, which force clients to make new requests in order to receive new available values;
- Through a GUI the possibility to set up the central module to record real-time data into databases for later analysis.

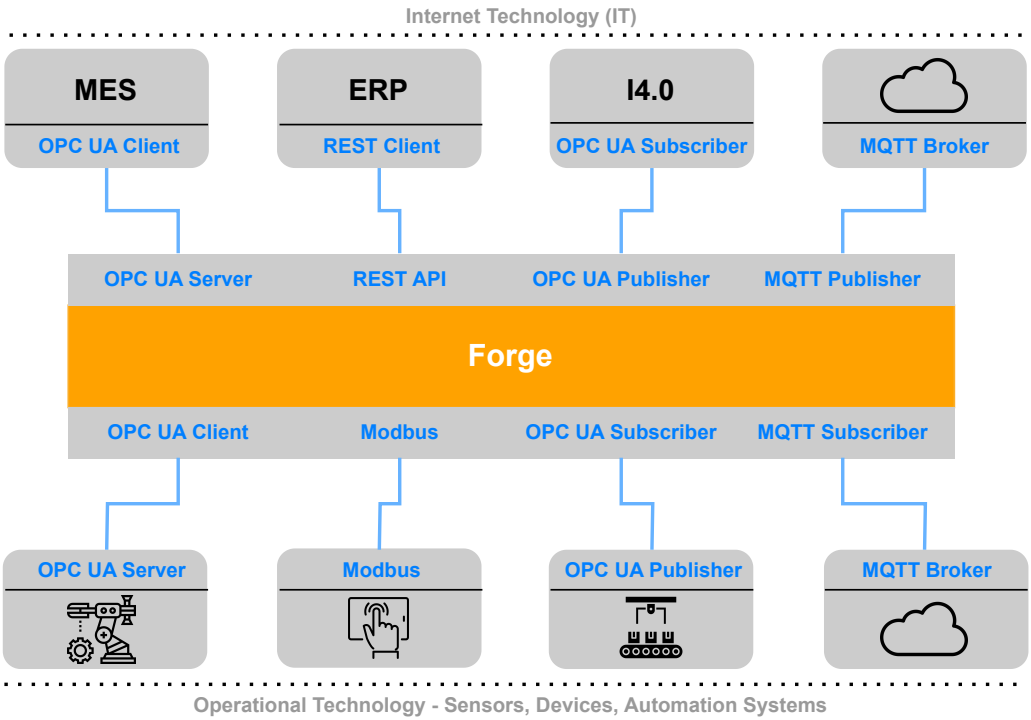

Figure 22: Prosys OPC UA Forge Architecture. Adapted from [92].

Another enterprise focused on the creation of automation software programs called ICONICS developed a solution similar to the previous. The solution named DeviceXPlorer OPC Server (DxpSERVER) works as a OPC UA server capable of communicating with various different industrial devices, around 400 types of devices of more than 100 vendors [94]. The enterprise offers other modules that can communicate with this central model, like GENESIS64, which basically is one HMI Supervisory Control and Data Acquisition (SCADA) that allows enterprises to collect, visualize, and analyze real-time data of the industry equipment, as presented in Fig. 23. The ICONICS on November 30, 2022, launched the 7th version of the DxpSERVER solution. This new version includes support to MQTT and HTTP protocols, joining to already included interface OPC UA [95]. These new interfaces will provide the possibility to integrate industry floors with cloud services to explore the data extracted, such as AWS IoT Core [96] and Azure IoT Hub [97] solutions.

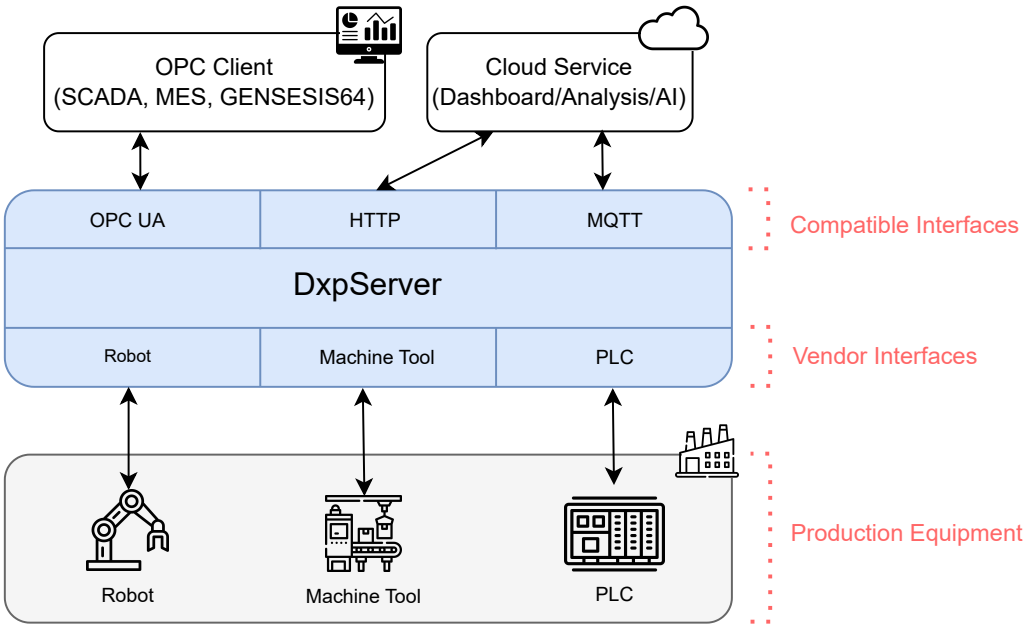

Figure 23: DeviceXPlorer OPC Server Architecture. Adapted from [98, 95].

3.3 DISCUSSION

After presenting the solutions related to this research, several conclusions can be drawn. The following discussion addresses the research questions previously defined and referenced in Section 1. Consequently, all the identified solutions are compared in Table 1, considering key features and the research questions defined.

Table 1 also shows the solutions that contributed to understanding the current state of the DT technology, helping to understand the main gaps that exist in its implementations. Commercial and scientific research is included in the table, with commercial solutions marked in blue. In addition, the conceptualization of the table was done based on the table presented in research [14].

Each solution presented in Table 1 has a description of your use case, as well as the level of digitization, where the values were outlined based on categories presented by Kritzinger et al. [8]. Therefore, the values are: "DS" stands for , "DT" stands for DT, and "DT*", which is assigned in three cases: when the solution presented one DT architecture with bidirectional communication, but in implementation the solution was not capable of including that feature; when the solution only presented a DT architecture, without any implementation of it and when the solution didn't represent a digital representation of physical devices. In addition, the table contains the column "DT Features", which presents the principle and extra features of DT technology, which they are:

- **Emulation** - It determines how real-time information about the production line is presented to the user;
- **Remote Control** - It indicates whether the solution considered control feature through DT;
- **Temporal Visualization** - It indicates if the solution shows the historical data to the users;
- **Analyzes for Optimization** - It determines if the solution is concerned with optimizing the manufacturing process;
- **Fault Detection** - It specifies whether the solution considers predictive fault detection or real-time fault detection;
- **Simulation** - It indicates if the solution allows production simulations to be carried out in different scenarios.

These columns are filled with the symbol "✓✓", indicating that the solution implemented the feature, or with the symbol "✓" if the feature is only mentioned. Following these, there are columns under the column "Implementation Features", which refer to more specific features. The first column in this group is titled "Communication Protocol" and indicates which protocol the authors selected to gather information from machines/physical devices. Additionally, in cases where the OPC UA is used, two descriptions are provided where the "CS" stands for "Central Server", which means that the system has one OPC UA server that covers several machines and makes available their real-time data from it. The description "SSPM" stands for "Single Server Per Machine" and means that the solution supports the topology of one OPC UA server per machine. Next, the columns included in the column "Recover Data Technique" describe what techniques are used to retrieve information from physical devices. This column covers the following columns:

- **Pub/Sub** - It stands for Pub-Sub, where this model of communication-based on events is used to capture real-time data from physical devices;
- **Polling** - A technique in which data is periodically extracted from the data source;
- **Streaming** - A technique in which data is periodically sent to the client modules to be updated.

The "Scalability" column indicates whether the implemented system supports multiple machines or not. The "Machine Type" column specifies the kinds of physical devices covered by the solution. The "Adaptability Level" column shows if the

solution can be adapted to changes in the production line, such as the integration of new machines. This column can have the following descriptions:

- **Ad-hoc** - It indicates that a solution was developed for a specific use case and can not be adapted to new machines;
- **Adapt.** - The description stands for "Adaptable" and indicates that the solution can be adapted to integrate new machines but requires a huge effort from the enterprise, where it has to change several modules of the solution to integrate new devices. Also, with this description are included the solutions that present one conceptual architecture capable of representing a DT;
- **Gen.** - It stands for "Generic" and means that the system is designed to integrate machines dynamically, with some setup and knowledge;
- **Gen.*** - It is a variation of the "Gen." description, involving some characteristics but the difference of requiring fewer setups and knowledge to integrate one machine, such as knowing only a couple of parameters.

Additionally, the table includes the "Test Environment" column, which indicates where the solution was tested, and the "Cloud Usage" column, which shows if cloud services were utilized. Finally, the columns related to the defined research questions aim to summarize each study and compare it with the present work, offering a clear comparison.

Table 1: Comparative Table of Related Work. "-" - not available or not understood, "✓" - a feature only mentioned, "✓✓" - feature implemented, "CS" - Central Server, "SSPM" - Single Server per Machine, "Adap." - Adaptable, "Gen." - Generic Solution.

Article	Use Cases	Digitization Level	DT Features					Implementation Features								RQ1	RQ2	RQ3
			Emulation	Extra Services				Communication Protocol	Recover Data Technique		Scalability		Machine Type	Adaptability Level	Test Environment	Cloud Usage		
				Remote Control	Temporal Visualization	Analyzes for Optimizations	Fault Detection	Simulation		Pub/Sub	Polling	Streaming	Single	Multiples				
Tower Crane Solution [87]	Tower Crane	DT	AR & 3D Model & Dashboard	✓✓	✓✓	-	✓✓	-	HTTP	-	-	✓✓	✓✓	-	-	Ad-hoc	Lab.	✓✓	X	-
6-Layer Architecture [20]	Catalytic Converter Assembly	DT	3D Model & Dashboard	✓✓	✓✓	-	✓✓	✓✓	OPC UA (CS)	✓✓	-	-	✓✓	✓	Robotic Gripper	Adapt.	Lab.	✓✓	X	-
Conceptual 6-layer Architecture [10]	Mining Industry	DT*	3D Model & Dashboard	✓	✓	✓	✓	✓	-	-	-	-	✓	✓	-	Adapt.	-	✓	-	-
6-Layer Architecture, including AR [85]	Production Management	DS	AR & Dashboard	-	-	-	-	-	OPC UA (CS)	-	-	-	✓✓	✓	CNC	Adapt.	Ind.	-	-	X
Robot Control [16]	Production Management	DT	3D Model	✓✓	-	-	-	-	OPC UA & Vendor Interfaces (CS)	-	-	-	✓✓	✓✓	Robots	Adapt.	Lab.	-	X	X
Digital Shadow of a CNC System [83]	Production Management	DS	Dashboard	-	✓✓	-	-	-	OPC UA (CS)	-	-	-	✓✓	✓	CNC	Adapt.	-	✓✓	-	-
Industrial Machines & AR Application [86]	Production Management	DT*	AR & Dashboard	-	✓✓	-	-	-	OPC UA, MTConnect (CS)	✓✓	-	-	✓✓	✓✓	CNC	Adapt.	Ind.	✓	-	X
Monitoring of Production Line Stations [14]	Mobile Assembly	DT*	Dashboard	✓	✓✓	-	-	-	OPC UA (SSPM)	-	-	-	✓✓	✓	CNC	Adapt.	Lab.	✓	-	-
4-Layer Architecture for Conveyor [90]	Production Line	DS	-	✓	-	✓	✓	-	OPC UA (SSPM)	-	-	-	✓✓	-	Factory floor Conveyor	Ad-hoc	Lab.	✓✓	-	X
Adaptive Shop-floor Scheduling [81]	Molds Production Line	DS	Dashboard	-	-	✓✓	✓✓	-	OPC UA (CS)	-	-	-	✓✓	✓✓	CNC	Adapt.	Lab.	✓✓	-	X
Online Predictive Maintenance [88]	Production Line	DS	Dashboard	-	-	-	✓✓	-	OPC UA (SSPM)	-	-	-	✓✓	-	Machine Simulation	Ad-hoc	Lab.	-	-	-
Autonomous Guided Vehicle [84]	Battery Assembly	DT	Dashboard	✓✓	✓✓	-	-	-	OPC UA (CS)	-	✓✓	✓✓	✓✓	✓✓	Robots	Adapt.	Lab.	-	X	-
Predictive Maintenance for CNCs [89]	Production via CNC	DS	Dashboard	-	✓✓	✓	✓✓	✓	OPC UA(CS)	-	-	-	✓✓	-	CNC	Ad-hoc	Ind.	-	-	-
Rest Middleware Solution [17]	Industrial Environment	DT	Dashboard	✓✓	✓✓	✓	✓✓	-	OPC UA(SSPM)	✓	✓✓	✓✓	✓✓	✓✓	PLCs	Gen.	Lab.	✓✓	X	X
MQTT Middleware Solution [91]	Industrial Environment	DT*	Only an API available	✓✓	-	-	-	-	OPC UA(SSPM)	✓✓	✓✓	-	✓✓	✓✓	PLCs	Gen.*	Lab.	-	X	-
Prosys OPC UA Forge [92]	Industrial Environment	DT*	Only an API available	✓✓	-	-	-	-	OPC UA (SSPM)	✓✓	✓✓	-	✓✓	✓✓	-	Gen.	-	✓✓	X	-
Device DeviceXPlorer OPC Server [94]	Industrial Environment	DT	Dashboard	✓✓	-	-	-	-	OPC UA (CS)	✓✓	✓✓	-	✓✓	✓✓	-	Gen.	-	✓✓	X	X
Present Work	Stone Industrial	DT	Dashboard	✓✓	✓✓	✓	✓	✓	OPC UA (SSPM)	✓✓	✓✓	-	✓✓	✓✓	OPC UA Devices	Gen.*	Lab.	✓	X	X

As shown in Table 1 and considering RQ1, it is evident that solutions are evolving towards true DTs, allowing bidirectional communication, which makes it possible to control and monitor production lines. Also, the table shows that 6 out of 17 solutions are considered true DTs ("DT"), where 8 solutions have this feature related to RQ1. This capability is represented by the "Emulation" and "Remote Control" columns, where several solutions demonstrate a DT through AR applications and 3D models. However, two solutions don't provide emulation and cannot be considered true DTs as they do not provide a digital representation of physical assets, despite offering APIs for real-time machine data extraction [91, 92]. Furthermore, it should be noted that only one solution didn't use the OPC UA communication protocol, highlighting the impact of OPC UA on the industrial environment for constructing DTs with the major features.

Regarding RQ2, which examines whether the solution can dynamically include new machines without effort by part of enterprises, only one solution was capable of this [91]. The solution in question has one central module capable of integrating automatically the OPC UA machine servers on the network without efforts and knowledge by part of enterprises. Thus, the level of adaptability of this solution is considered "Gen*". In contrast, there is the generic solutions ("Gen"), in total of 4, which demand users to configure all integration with underlying systems and have knowledge about the technologies involved, making life difficult for SMEs. Moreover, there are 9 solutions that fall outside the scope of this feature, with "Adapt." adaptability level, as they demand modifications at level of solution whenever a machine has to be integrated. Lastly, the rest of solutions with an 'Ad-hoc' adaptability level are designed to cover only a single machine or a specific use case and were not intended for any adaptation [87].

About the RQ3, there is one commercial solution that makes available a GUI. However, it is not part of the DxpSERVER solution but can be integrated from solutions presented by ICONICS to manipulate and monitor the underlying systems [94]. In addition, despite existing solutions with a GUI, some solutions were not considered because they only cover one physical entity, like [20, 83, 87]. Another solution but compatible with various industrial machines, more specifically, Computer Numerical Control (CNC) machines, presented an AR interface and showed that this kind of interface adds more understanding of the industrial process [86]. Finally, solutions with this functionality of presenting dynamically several machines/physical entities in the same GUI are 7 out of 17 solutions, so some effort should be considered in this regard in future solutions not to incorporate just one machine but several, as production processes are carried out between several machines.

Another notable feature, unrelated to the research questions but pertinent to this sort of solution is related to the architecture presented by some solutions [17, 86, 83, 16]. In 9 solutions, an OPC UA central server ("CS") is used to aggregate information from underlying systems. However, this approach can overload the central server when multiple clients attempt to retrieve data from the connected systems simultaneously. Therefore, in that case, architectures where one server per machine is used ("SSPM") should be considered, so that the workload is distributed and to prevent the failure of one server from compromising the others. In short, 6 out of 17 used one architecture with "SPPM" topology.

In conclusion, as shown in Table 1, there is still work to be done to develop industrial solutions in a sustainable manner for SMEs. There is space to add new services to the existing solutions, such as temporal visualization, analyses for optimizations, fault Detection, and simulation. However, a good structure must be created to support these services. Therefore, the present solution intends to develop a system that supports essential services, as well as extra services, and a system that follows the evolution of the production lines of SMEs, i.e., a system that allows enterprises to include new machines dynamically without effort and knowledge. In contrast to other solutions, such as [83], which uses Node-RED [99] technology, the entire solution must be changed whenever a new machine is integrated or new machine parameters are considered. Additionally, the OPC UA communication protocol is used, so that the current solution can be adapted to different industries and remain compatible with future machines, which increasingly have this feature. In addition, this solution considers that sometimes the network in factories is not very favourable to the exchange of information, so unnecessary communications should not be used, but asynchronous communications based on changes in the states of physical entities should be adopted to notify interested parties of what happens. Ultimately, the solution aims to help operators and engineers manage the shop floor from a GUI that adapts to each machine on the production line.

4

PROJECT METHODOLOGY

Nowadays, a solid plan is essential for successful and timely software development. Thus, a well-defined roadmap was created. The roadmap is shown in Fig. 24, outlining the project's phases, which were structured following the Design Science Research Methodology (DSRM) approach [100], with a problem-centred initiation. Although the phases were clearly defined, the workflow is not always linear, occasionally requiring tasks to be repeated.

Fig. 24 shows the various phases involved in this work, the achievements of which can be understood from the description below:

- **Phase 1** - After identifying a general problem in a specific sector, it was analyzed to determine whether there was an opportunity to solve it. In the case of this work, it was possible to detect a lack of digitization of industrial processes in the manufacturing sector. Therefore, meetings were organized with people involved in this sector. In addition, scientific articles were read to identify gaps and needs in the industrial sector;
- **Phase 2** - The Technology Exploration & Feasibility phase included understanding the flaws found and whether they could be overcome with current technologies. This involved studying the technologies used by existing solutions and exploring others that could improve these types of systems. After that, it was studied whether there was a way forward to create a solution that would align with the problems found and add value to the industry.
- **Phase 3** - In the Planning phase, the goals of the project are described, as well as the functional and non-functional requirements. Thus, the goals were defined and specified in Section 1. Also, the requirements were defined and are described in Section 5;

  Regarding working methodologies, the Kanban agile methodology was selected to support the project as it can become more flexible than others like Scrum, which requires that a task or a set of tasks should be implemented within a well-defined time frame, called Sprint, which can be not suitable to this project because some tasks may not be so predictable [101]. Another reason is

that certain tasks can sometimes be blocked, e.g., the task to test the system in the factory, where the execution depends on the availability of the factory. Thus, it was possible to continue developing the solution without stopping. In addition, using this methodology, it's easier to stop tasks and focus on more important ones and keep this more traceable [102]. The Kanban board created is illustrated in Fig. 25 with following columns:

  - **Backlog** - The column that stores all not-so-priority tasks planned that need to be done. These tasks can be software functionalities, errors to fix, software tests, or documentation. It may be changed according to priority and the addition of new tasks. This type of column is traditionally used in the Scrum methodology, known as Product Backlog, but it can add value to the Kanban methodology as it allows the stakeholders not to forget certain tasks during the project Lifecycle [101, 103]. In addition, it is good for keeping tasks that can be carried out when priority tasks have already been completed, contributing to a continuous flow of work. This mix of methodologies can be called Kanplan [104];
  - **To Do** - Phase that contains the priority tasks, defined in a weekly meeting by the stakeholders of the project;
  - **In Progress** - Column that includes the tasks that are being carried out;
  - **Blocked** - List populated with tasks dependent on something;
  - **Done** - Column that contains all the tasks completed.

- **Phase 4** - The "Design Architecture" phase is used to plan the architecture of the solution and to specify the modules needed. In addition, the technologies were defined from the knowledge acquired in **Phase 2**. This process sometimes can be repeated because the technologies used in practice may not satisfy the needs, which leads the team to discuss new technologies again or to look for other ways to meet the requirements;
- **Phase 5** - In this stage and according to the plan, the functionalities chosen are developed and integrated into the solution. Here, Git VCS was chosen along with the GitHub cloud repository to store all the content related to this work, such as the documentation, the conceptualized schemes, and the developed software. In addition, it was chosen the git methodology called GitHub Flow, explained in Section 2. This approach of storing all work increments was chosen because it enables stakeholders to manage related content efficiently using just

two types of branches: the main branch, which contains all production-ready code, and the feature branch, which is used to develop new functionalities before merging them into the main branch upon conclusion;

- **Phase 6** - In this stage, a new version of the system was released, ready to be tested. This system version supported an AR application that would be tested in the upcoming phase. The application allowed factory operators to consult information on a particular machine, such as educational content (e.g., a video about how to change a cutting saw) and real-time machine (e.g. temperature, position, and current in the axis of a machine). This content was distributed by the solution presented in this work. It is important to note that this application is not the subject of the present research;
- **Phase 7** - The solution was tested with users as planned. In this stage, only the AR user application was available because the web application was still in development, so that application and the system that supports it were tested. A form was created where it included several sections: a biography section to extract information about the participants, including questions about the stone sector; a section with various tasks that the participants should realize in the AR application, which allowed the stakeholders to understand if the application was intuitive; another section to evaluate the usability of the application through the System Usability Scale (SUS) method and finally, the last section, which aimed to determine if the industry participants were interested in using the developed application and system. This test phase was precious, as it enabled the researchers to extract various insights about the developed solution and thus adjust and improve the solution so that it produced more value for the users;
- **Phase 8** - One of the main goals of the research was also to publish an article to spread the knowledge learned. A paper was written, published, and presented at the 2023 conference called International Conference on Graphics and Interaction (ICGI). This article was published together with another researcher, and it is indexed in IEEE platform under the name "Augmented Reality and Digital Twin for Mineral Industry" [1];
- **Phase 9** - After the tests were performed and the article written, new insights about the developed solution were extracted. Therefore, from these phases, it was possible to obtain information about the usability of application AR, as well as information about the performance of their support modules,

which belongs to the current solution presented in this work. Based on this information, new goals, improvements, and functionalities were outlined;

- **Phase 10** - In this phase, the adjustments to the solution were realized, as well as new functionalities were implemented, as planned in the previous phase;
- **Phase 11** - A second version of the solution was released. This new release was launched with new features. The developed solution had modules to support an AR application and a Web application. The AR application, as previously mentioned, is not the goal of the present research;
- **Phase 12** - Second testing phase was responsible for testing the entire solution, including the web application. In this phase validation and usability tests should be performed. Therefore, in this phase was possible to test the web application with users where a form was created with several sections. The biographical section to extract information about the participants, including questions about the stone sector. Another section with questions related to the industry, to extract more valuable information about the sector, answered only by people linked to the sector. The third section contained tasks to be carried out by the participants, some linked to the industry and others not. Also to assess the usability of the application, a post-task section was created that included the SUS scale. Finally, the last section contained questions aimed at extracting the usefulness and interest of the participants to use the Web platform. In addition, the validation tests were performed and are explained in Section 7 too;
- **Phase 13** - This phase involved the writing of a new article to integrate the findings from the reviewed articles, as well as the recent changes made to the solution and the obtained test results. Up to the date of project submission, the new article was submitted for revision;
- **Phase 14** - Phase where the present project was written and delivery for evaluation.

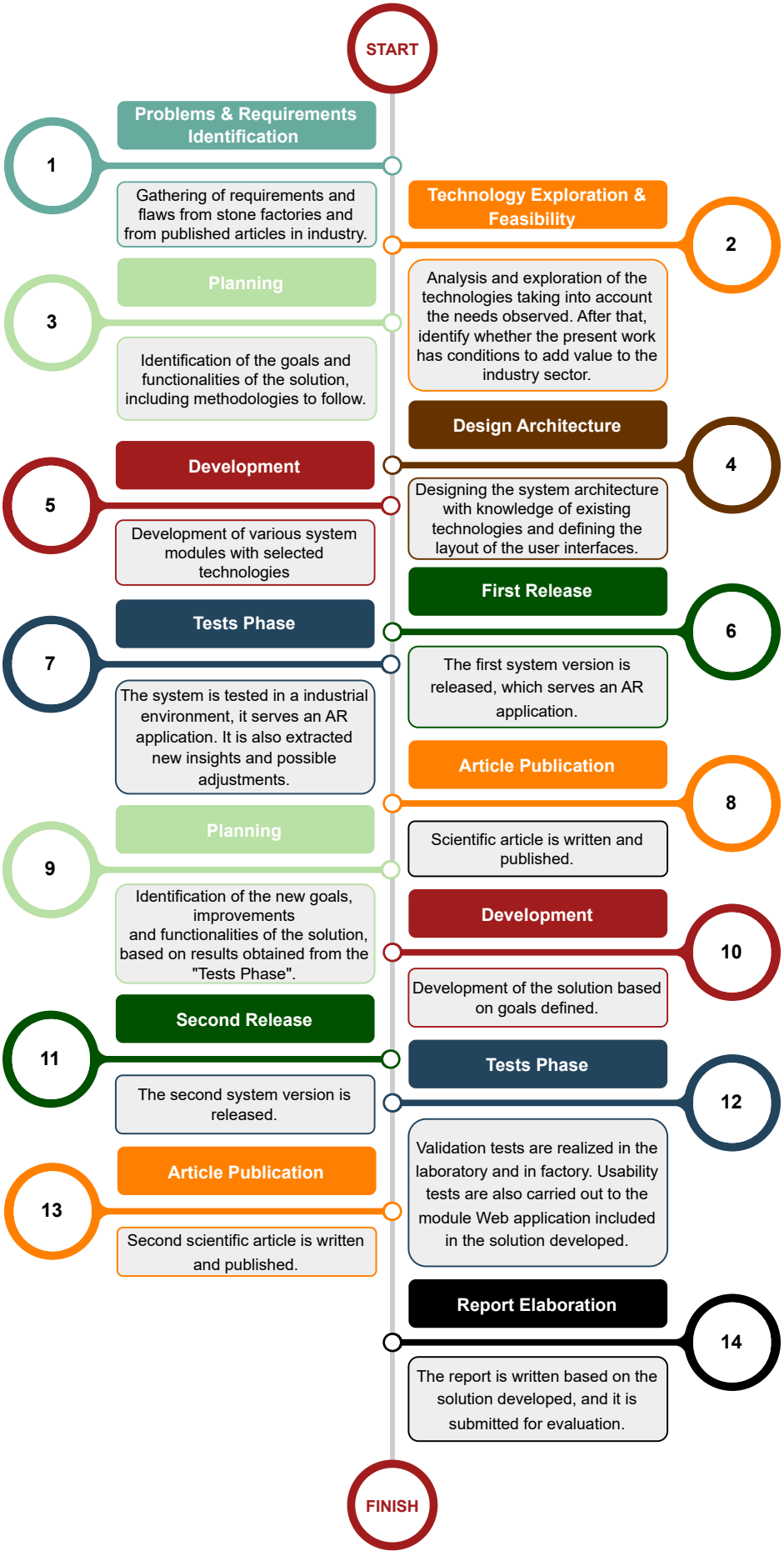

Figure 24: Project Roadmap

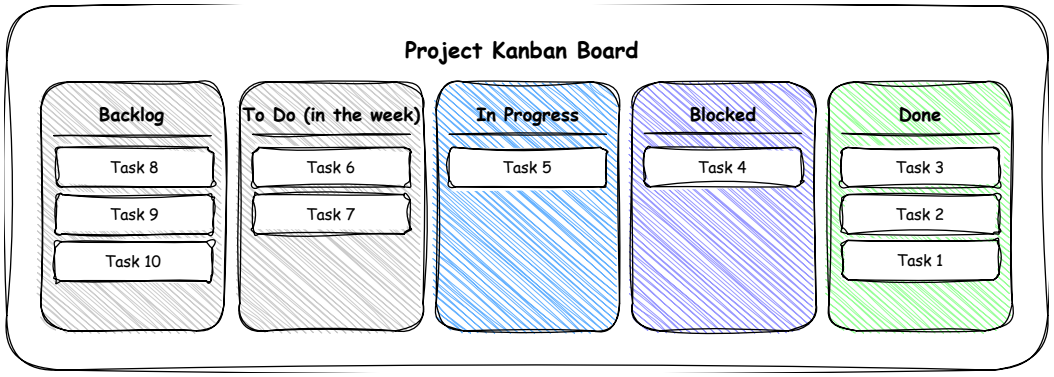

Figure 25: Project Kanban Board

5

PROPOSED SOLUTION

The principal weaknesses in existing solutions and the conditions and challenges faced by companies were identified through an analysis of related work. Thus were defined non-functional and functional requirements of the solution. Additionally, with these requirements and functionalities outlined, which the solution should accomplish, an architecture is proposed. Also, your constituent modules are explained in this section. Lastly, the views of one web application are proposed.

5.1 REQUIREMENTS

The present section presents the list of requirements that were defined considering the gaps found and the conditions of the people involved in this sector, where there are machine operators on the shop floor and engineers at the office, controlling and monitoring all factory operations. Consequently, the requirements for the current solution are outlined below.

NON-FUNCTIONAL REQUIREMENTS

- The factory should have configured network access points that cover its entire infrastructure;
- The machines on the shop floor should have their OPC UA servers configured, i.e., they have to be ready to communicate via OPC UA protocol;
- The users should have access to the machine network from internet-enabled devices, such as tablets, smartphones, and computers;
- There are two roles in this platform considered, the administrators and the shop floor operators, with different privileges;
- One machine can have its dedicated OPC UA server, consequently, a machine is identified by its Internet Protocol (IP) address and port.

FUNCTIONAL REQUIREMENTS

- Administrators should be able to integrate and dissociate new machines to the solution;
- Administrators should be able to select the most important parameters of a machine to be monitored on the shop floor and decide how they are presented;
- Administrators should be to delete/edit machine parameters selected;
- Administrators must be able to update the list of machine parameters to select them;
- Administrators should be able to integrate informative content related to certain machines, like video, text, and audio, where they aim to instruct operators on the shop floor;
- Administrators should be able to delete/edit informative content;
- An administrator should be able to manage all user access permissions, such as restricting some users from changing the state of machines and allowing others to change the state of only one machine;
- An administrator should be able to list all machines integrated into the solution;
- Administrators, who may be office workers, should be able to see which machines are available and under maintenance;
- Administrators should be able to activate the procedure of collecting historical information;
- Users should be able to monitor and control a machine through a GUI;
- Users should be able to consult the reference values of a certain machine, which allows them to see what its normal values are;
- Users should be able to put a machine in maintenance status, to the administrators know that a certain machine is not operational;
- Users must be able to consult the previous status of machines, in a given period.

5.2 PROPOSED ARCHITECTURE

In this section, the proposed architecture is explained, having been designed to address the previously identified challenges. The architecture, as presented in Fig. 26, is divided into eight layers, each representing a crucial module within the system, each with a specific function. Additionally, the idealized communications between these modules are illustrated by arrows, indicating how the modules interact with each other.

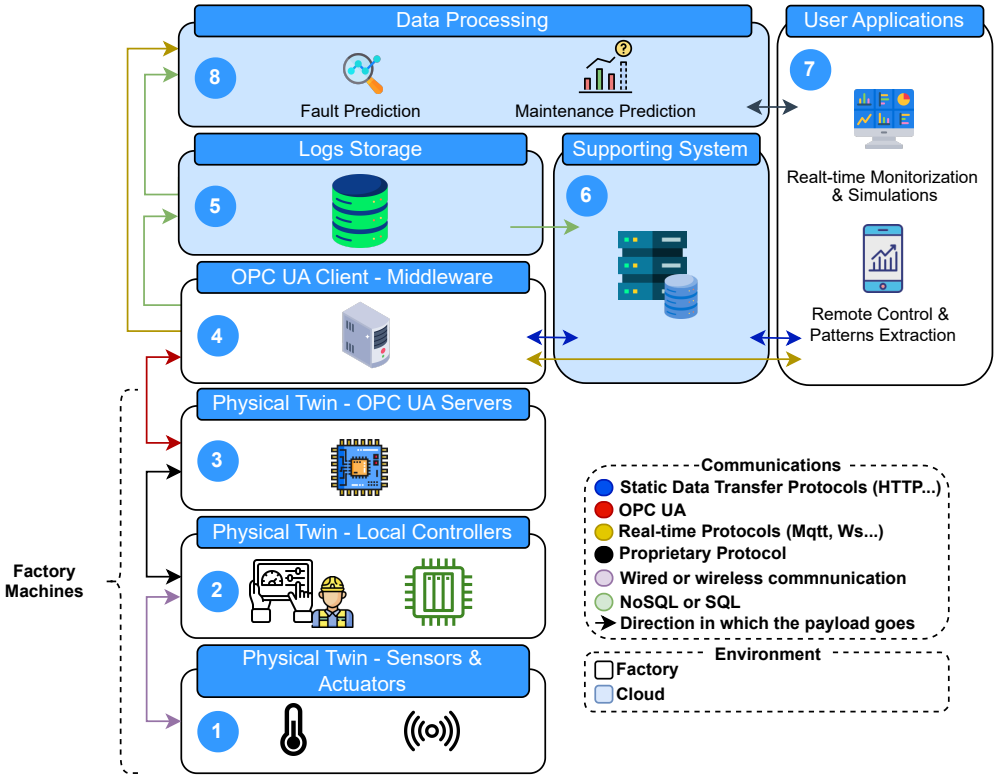

Figure 26: Layered Proposed Architecture

5.2.1 *Physical Twin - Sensors and Actuators*

This layer includes all sensors and actuators, such as robotic arms on CNC machines that operate along different axes to perform various tasks. Additionally, external sensors can be added to these machines to gather more data, making it easier to understand the machine's status, e.g., temperature sensors, vibration sensors, and sound sensors ([81]).

5.2.2 *Physical Twin - Local Controllers*

The PLCs/machine controllers belong to layer 2, which are units responsible for controlling all machine movements via actuators and obtaining information about the machine units from sensors. Each machine can have a controller responsible for its production processes. Some controllers on the market already incorporate the OPC UA server module, which belongs to layer 3, allowing information from a covered machine to be made available via the OPC UA protocol. One controller with this capacity is the controller provided by Simatic [105].

If there is this type of controller in the factory, they occupy layers 2 and 3. Other controllers come with vendor interfaces that allow client units to extract data about a machine, but in this scenario and considering this study, these machines need an additional module that belongs to level 3 so that there is an integration of them with this solution.

5.2.3 *Physical Twin - OPC UA Servers*

This layer includes all OPC UA servers in the factory. Ideally, each machine should have its own OPC UA server, although there are cases where a central OPC UA server communicates with multiple machines. The present solution is designed to function in both scenarios but performs optimally when each machine has its own OPC UA server. This is because edge devices, where such servers typically run, have limited resources, and if multiple services or clients rely on a single server for information, it can lead to server overload and loss of communication with the underlying machines.

Therefore, having one OPC UA server per machine is ideal for distributing the workload and ensuring consistent communication. Thus, if an OPC UA server is not already embedded within a machine, a separate module should be created, with all relevant machine parameters mapped into the OPC UA server's address space. In addition, the OPC UA servers configured should incorporate all the security OPC UA built-in mechanisms, so that there are secure communications [106].

5.2.4 *OPC UA Client - Middleware and Logs Database*

The layer 4 includes the OPC UA client capable of communicating with OPC UA servers of the respective machines. In this solution was conceptualized one module called middleware capable of making the bridge between end-user applications and OPC UA servers. This approach is more suitable than having all end-user application instances directly access the OPC UA servers. The disadvantageous approach increases the workload of the machine servers due to multiple clients accessing information simultaneously, e.g., an engineer programming the production line and a factory floor operator monitoring the production process, which leads to server overload and possible unavailability.

Also, it is important to note, as before mentioned, that many of these servers operate on hardware with limited capabilities, which can compromise their performance when faced with multiple demands. To mitigate this, an intermediary module, labeled Middleware, presented in Fig. 26, can be developed to handle singular communications for each OPC UA server, as needed. This approach allows the same information to be distributed to several user applications with minimal overload on the OPC UA server of the respective machine, which often want the same information of the underlying system.

This module can be developed using several open source packages OPC UA, and its selection should be made based on the specific use case and evaluation metrics, such as energy consumption, computing load on the devices and available features [107].

Furthermore, this module is not limited to sending real-time data made available by the machines/underlying systems, but should also be able to execute commands requested by clients over the machines through the OPC UA servers. In addition, to spread the real-time data of the machines for all end-user applications that require, this module should use good protocols, i.e., protocols and technologies focused on low latency and effective message distribution, e.g., technologies like socket.io, a JS package, which uses WebSocket protocol, which allows the same message to be distributed to several clients, grouping them in one channel, this, from its rooms feature [108].

These clients can be applications such as cloud services that perform data analysis (layer 8) and end-user applications used by shop floor operators to monitor the status of the production line (layer 7). Regarding the protocols, several fit in this layer, many of them are used in IoT technology, such as MQTT, but the choosing should

be done based on the needs of the use case, so the advantages and disadvantages of each one should be analyzed [109]. For instance, the WebSocket communication protocol is great to maintain the interoperability with browser applications, so it's should used in systems that use this type of applications. Another protocol mentioned earlier is the MQTT communication protocol, which is particularly useful for integrating modules with cloud services like Azure IoT Hub [110]. These services enable data processing to extract more detailed information [111]. Therefore, the protocols and technologies should be carefully analyzed and integrated based on the specific requirements of the solutions to develop.

Additionally, this module should be capable of persisting data in databases (layer 5) to save the history of all operations performed in underlying systems. This data can be normalized and stored efficiently in intended databases. Finally, this data will be useful for the analysis services included (layer 8) to extract patterns, helping the industries optimize their production lines.

5.2.5 *Supporting System*

This layer includes a supporting system that assists in managing the machines integrated into the system and handling them related static content. This content may include instruction manuals and worker training materials in text, audio, and video formats, helping to integrate new machine operators. Administrators should regularly consult and manage all this information via end-user applications (layer 7). Additionally, this module enhances engineers' awareness of the machines on the factory floor by providing comprehensive machine information and status updates, such as identifying those under maintenance and unavailable for production. Furthermore, it enables administrators to manage user permissions, such as controlling machine data access and remote control functionalities.

5.2.6 *User Applications*

The applications should be developed and integrated for use by factory stakeholders (layer 7). These applications should be capable of representing the DT of the monitored systems, displaying shop floor machine parameters such as axis temperature and current, state of the present manufacturing work, detected errors, and tool status, which are some information available by stone manufacturing machines. Additionally, the applications should be designed according to the environment and

devices used by the users. For example, if the applications are intended for use on the factory floor by operators, a smartphone is likely the most suitable device, requiring an application optimized for mobile use. In contrast, users in an office environment, typically involved in production line planning, will more commonly use a computer, necessitating an application tailored for desktop use. Therefore, considering these requirements, a web-based application, accessible via a browser was designed to adapt to the user's device. This application should allow administrators to manage all machines integrated into the system, select the major parameters of the machine to present to the operators and manage all educational content related to machines. As mentioned, this application will be used on the shop floor, so the application should allow operators to monitor the parameters made available by the machines through layer 4 (e.g., sensor data, errors ...), consult educational contents provided by layer 6 and the history of each machine integrated stored in layer 5.

Currently, some HMIs have been developed that fit in this layer and could be integrated, and some of them use AR, which improves interaction with factory floor operators and makes some tasks easier to perform [112, 113].

Additionally, modules included in layer 7 can interface with modules in layer 6, which includes modules capable of processing data and providing more refined feedback on production operations to stakeholders. For example, these modules can employ AI algorithms to predict machine status based on collected data, aiding in maintenance scheduling and timely fault detection [111]. They can also monitor tool conditions, such as the wear level of cutting saws [89]. Moreover, vendor software like Tecnomatix PS [20] and Simulink [14] can be integrated into layer 7 to simulate and emulate production processes, though these modules primarily function as OPC UA clients, directly communicating with OPC UA servers.

5.3 WEB APPLICATION MOCKUPS

This section presents the mockups of the web application, displaying its User Interface (UI) and highlighting the main functionalities. This mockups serves as a medium-fidelity design, guiding the development of the web application's layout.

Beginning by initial view of the application, there is a dashboard where the main goal is to represent summarized data in brief. For example, the number of machines integrated into the system and the number of machines in maintenance process, as illustrated in Fig. 27.

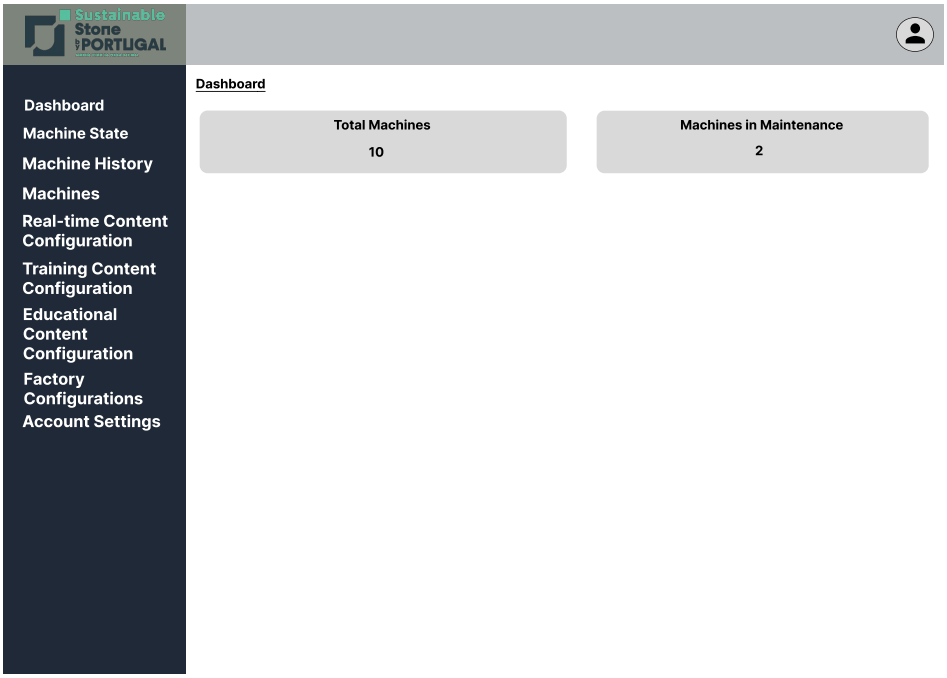

Figure 27: Dashboard Layout

Another important UI is presented in Fig. 28, which aims to list all machines available to users. This UI is rendered when a user wants to select one machine to monitor its state ("Machine State"), visualize its production process history ("Machine History"), select/configure its real-time parameters ("Real-time Content Configuration"), and when the user wants to configure educational content ("Educational Content Configuration" or "Training Content Configuration").

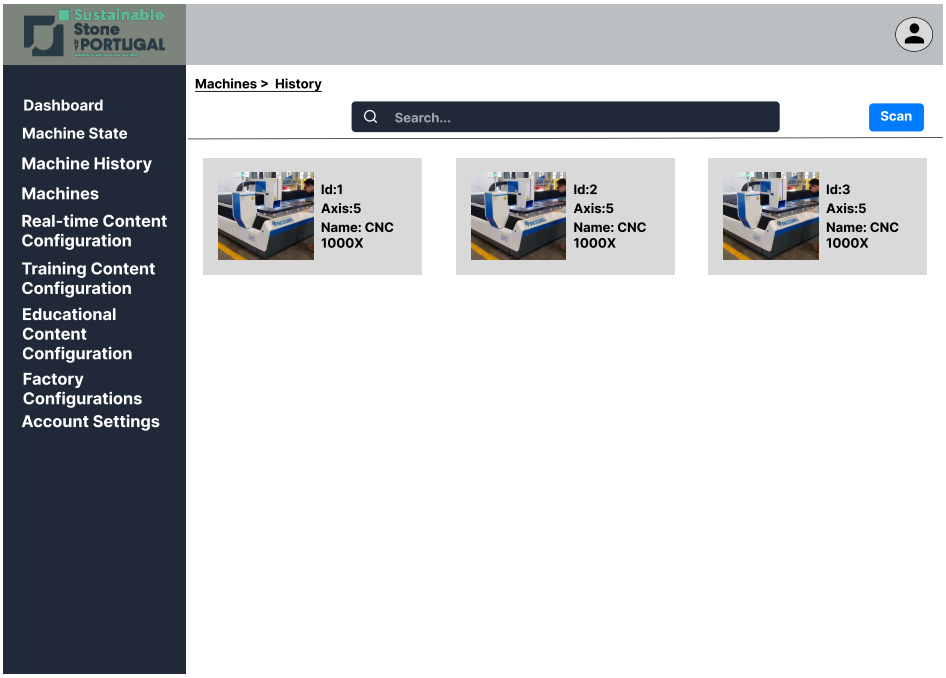

Figure 28: Machine List Layout

A page to manage the machines was designed, as shown in Fig. 29, where a user can add, edit, and delete machines. Additionally, a form for integrating new machines was designed, which is illustrated in Fig. 30.

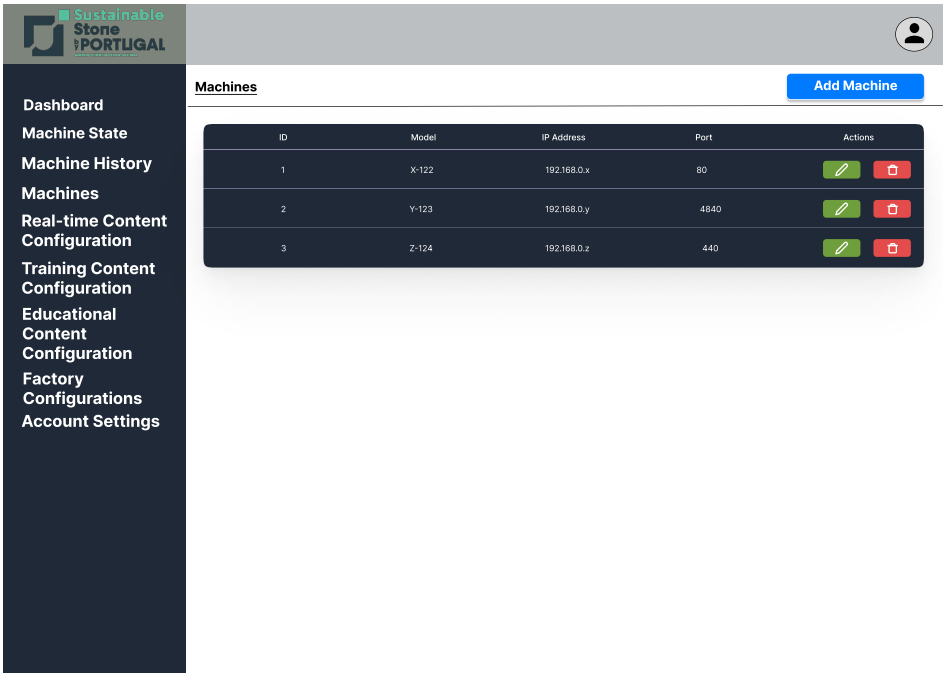

Figure 29: Machine Management Layout

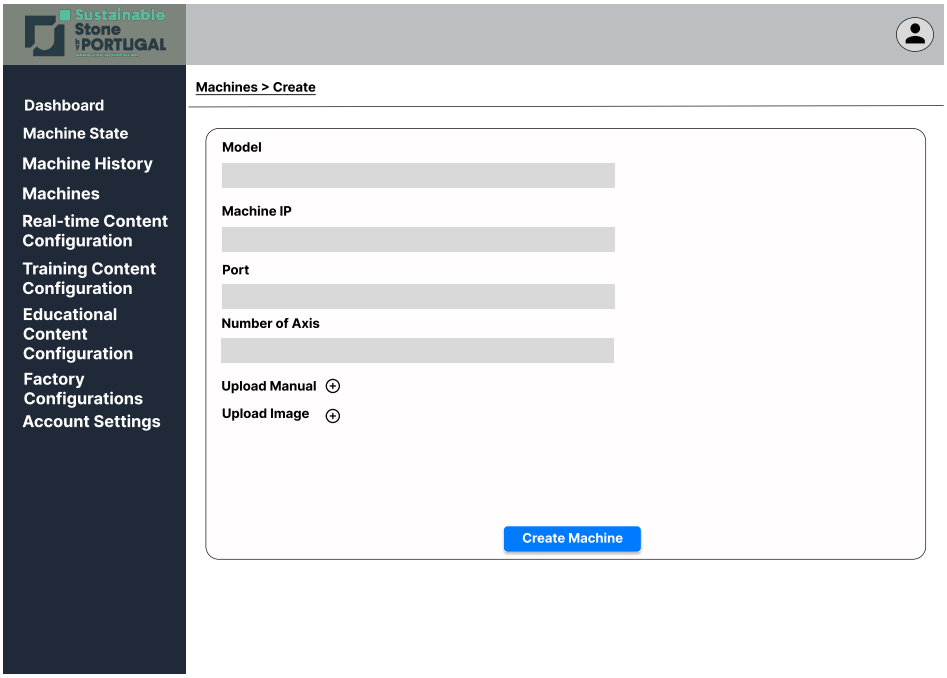

Figure 30: Machine Integration Layout

Regarding the consultation of the machine state, a page was designed, and it's shown in Fig. 31. This page should adapt to different machine parameter types, for example, if the parameter is a number, then the chart to be displayed should be a gauge chart that includes limits of values, allowing operators to understand the normal functioning of the machine. Another example is when the parameter is boolean, a red or green symbol should be displayed. These considerations enhance clarity and enable a quick understanding of a certain machine's status. In addition, the historical data of all parameters could be visualized through a page illustrated in Fig. 32.

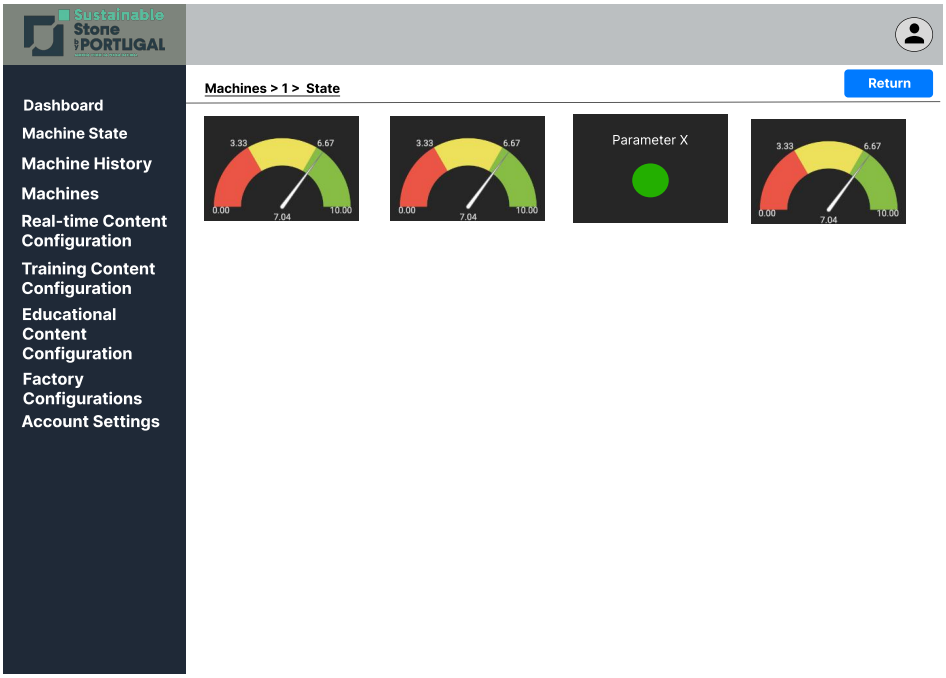

Figure 31: Machine State Layout

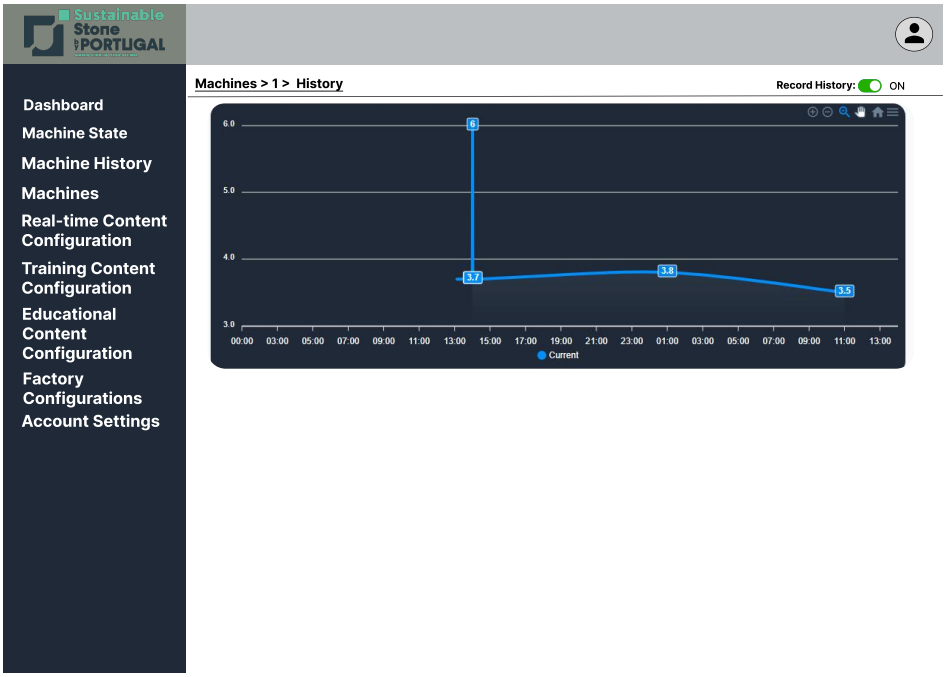

Figure 32: Machine History Layout

To select and configure real-time parameters of a certain machine the pages presented in Fig. 34 and Fig. 33 were designed. The first one allows users to manage all relevant machine parameters to the monitoring process. The last page enables users, specifically administrators, to select a machine parameter to consider on the shop floor and to configure how it should be displayed.

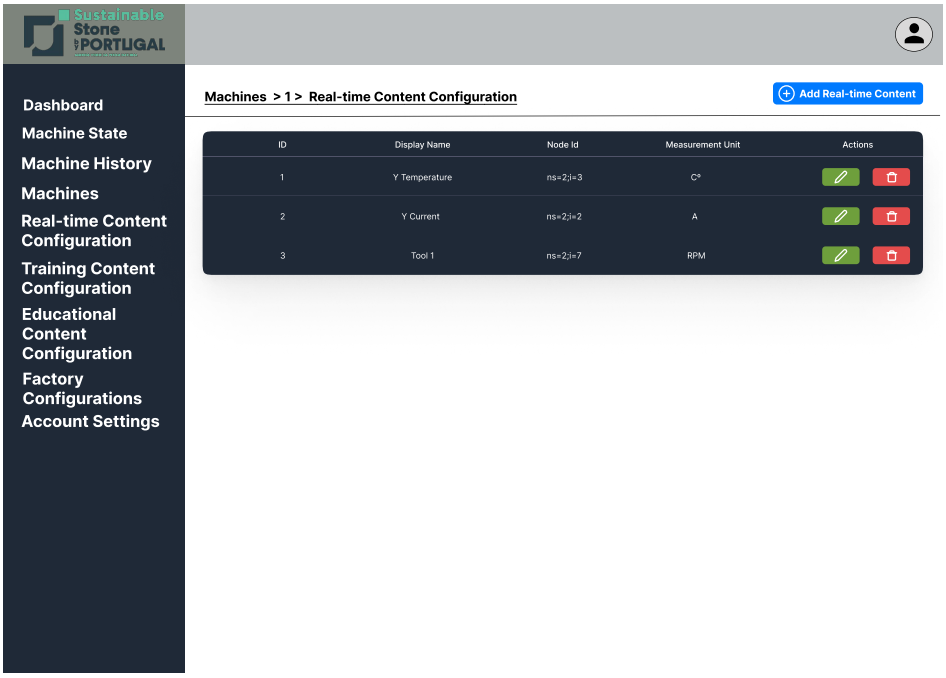

Figure 33: Machine Parameter List Layout

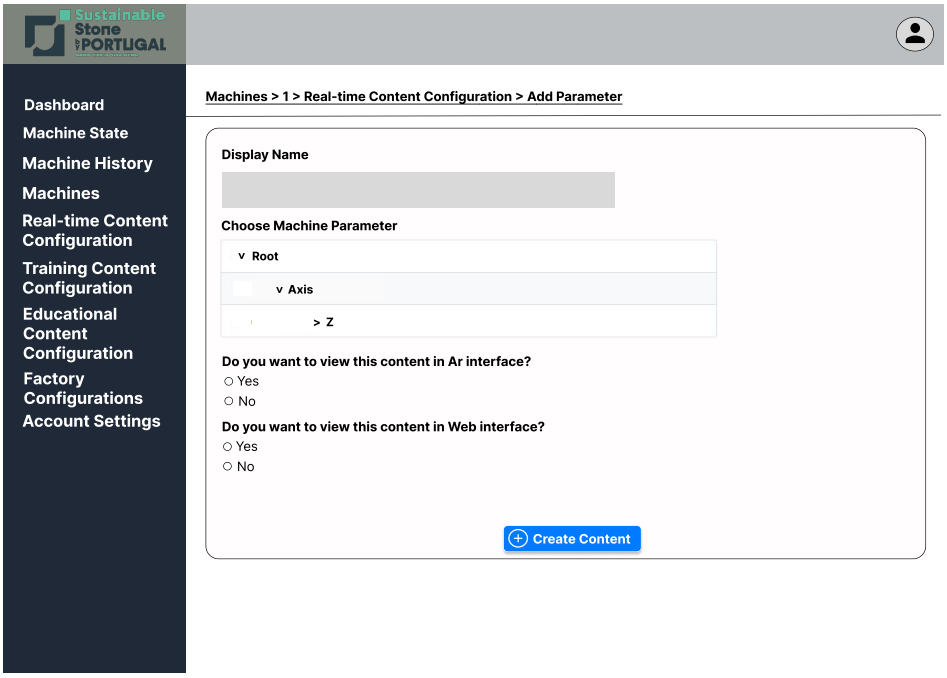

Figure 34: Real-time Parameter Configuration Layout

6

IMPLEMENTATION

This section presents the architecture of the developed solution, built using the technologies outlined in Section 2. Additionally, the requirements specified in Section 5 were carefully considered during its development. The implemented architecture is illustrated in Fig. 35, where the communication between the key modules is clearly analyzed. In the following sections, each module will be explained in detail, covering its functions and the technologies utilized.

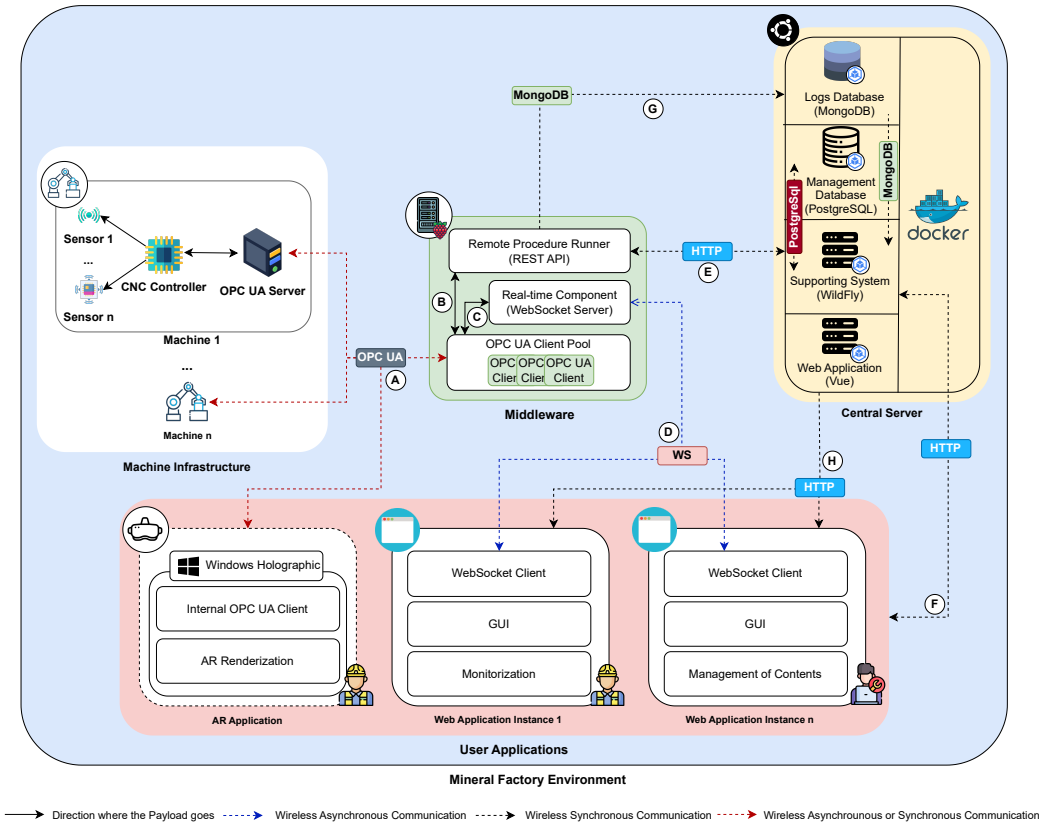

Figure 35: Implemented Architecture

6.1 MACHINE INFRASTRUCTURE

This module includes all machines in the factory, representing the Physical Twin of the machines. The covered machines in this solution used to have a robotic arm capable of performing 3D translations and rotations. More specifically, each robotic arm is moved between the axes of the machine and performs cuts in raw stone with the cutting saw for stone elaboration. Each machine, in the case of stone factories, contains its sensors and actuators, e.g., as sensors, temperature sensors, current in axis and velocity of cutting tools (Rotation per Minute (RPM)), and as actuators, the position of the axis, cutting velocities and tools used by robotic arms to work in different axis. Other parameters can be obtained and redefined about machines, such as machine state, machine turned on or off, kind of material used by machine, and machine errors. Another important component of the machine is the controller, which is linked to the sensors and actuators and allows the synchronization of the production line. Furthermore, the subcomponent OPC UA Server allows the external modules to communicate with each machine from OPC UA interface to retrieve information or control. In addition, as mentioned earlier, some vendor controllers have already integrated the OPC UA functionalities, making the development and integration of the OPC UA Server to each machine unnecessary. Therefore, to include a machine in the solution developed, it must be capable of communicating via OPC UA from an external module (OPC UA Server) or not.

6.2 MIDDLEWARE

The core of this solution is the module Middleware, which aims to help the other modules communicate with physical entities on the shop floor so that they can be controlled and monitored. This edge module operates near the machines and can forwards messages between other modules within the factory and outside the factory environment, more specifically, the cloud environment. Moreover, it was developed from a Python framework called Django Channels [114]. Technology that not only allows the creation of applications that handle HTTP requests but also the creation of applications that handle long-running connections, such as chats and chatbots using the WebSocket communication protocol. This framework was chosen to enable the simultaneous distribution of the same message across multiple modules and allows the integration of business logic to manage specific events, such as connecting to a particular machine and maintaining that single connection, even

as new clients connect. In addition, this technology makes it possible to carry out simple procedures on the machines, such as changing their values.

The current module integrates two key components that can be considered applications within the same Django project. The first is a API called Remote Procedure Runner, which facilitates the execution of simple procedures on machines, as previously mentioned. The other component, Real-time Component, is responsible for treating communication in real-time with machines, where it receives the new values of the parameters of each machine in monitoring and forwards them to clients interested in them. Both the components referred use the OPC UA client instances, represented within the OPC UA Client Pool element in Fig. 35. The number of client instances depends on the number of active connections with factory machines. In addition, these instances are not considered independent components because they are managed within each application. Therefore, the pool referred to in Fig. 35 is only for representational purposes. Last but not least, a Python and open-source OPC UA package was used to create the OPC UA client instances, called Opcua-asyncio [115]. This Python package offers a more comprehensive implementation of the OPC UA specification compared to other solutions, overcoming the JavaScript-based and Node-Opcua-Ua solutions [17]. Another advantage of this package is that it continues to be actively maintained by the community.

6.2.1 *Real-time Component*

Regarding the Real-time Component, it functions as a WebSocket broker application capable of maintaining connections with clients interested in machine data. This module creates an OPC UA client instance for a specific machine when the first client connection is established, serving all clients that wish to monitor that machine. This is made possible by the group feature provided by the Channels framework, which groups various clients simultaneously, allowing messages to be delivered in a broadcast fashion [116]. In addition, each client connected has one consumer instance, which in turn has one channel attributed, managed by one abstract layer called channel layer. This layer allows the consumers share information with groups or other consumers, as illustrated in Fig. 36.

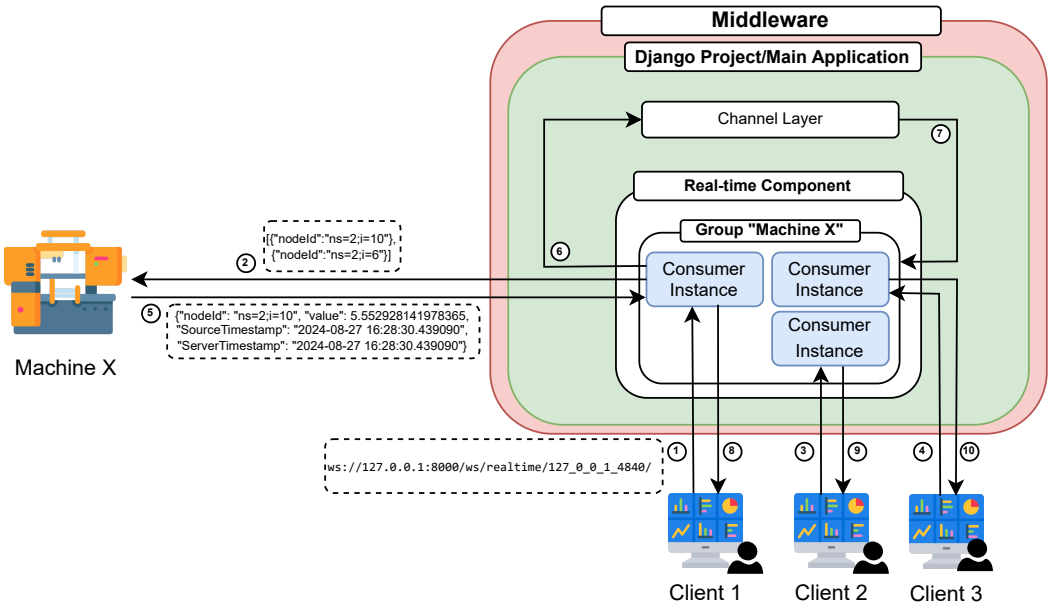

Figure 36: Consumers Relationship

As shown in Fig. 36 there are several involved processes so that the clients can retrieve real-time information related to machines, so the processes are the following:

- **Step 1** - The client application connects to the middleware to the Real-time Component application through the WebSocket communication protocol. It specifies via URL the machine IP address and port, and sends a message that identifies the parameters to be monitored of the respective machine;
- **Step 2** - The Real-time Component application receives the message from the client and creates an OPC UA client instance from the client consumer instance, which connects to the OPC UA server on the respective machine. This connection is based on Pub-Sub communication model, where the parameters are subscribed using node IDs of them, as specified in Fig. 36;
- **Step 3 and 4** - Other client instances can connect to the Real-time Component application to retrieve information from the same machine, but in these cases, if the specified parameters to monitor are the same, the OPC UA client instance created in step 2 remains running. Otherwise, a new OPC UA client instance with new parameters replaces the first one;
- **Step 5** - Once established the communication between OPC UA client instance and OPC UA server, and subscribed the parameters, the Consumer Instance, owner of that OPC UA client instance, will receive the new values of the subscribed parameters. Therefore, the consumer that created the OPC UA client instance receives the new values in the format shown in Fig. 36. Thus,

there is one communication more optimized, where only the messages are exchanged when the values of parameters change. It is the best approach comparatively to other techniques, such as the polling technique, which in that case would force the Real-time Component application to probe each parameter without knowing if the parameters changed, increasing the number of messages exchanged and decreasing the performance of the middleware;

- **Step 6 and 7** - Once the new values are received by the OPC UA client instance, the consumer that owns it, sends the parameter values to the Channel Layer that is responsible for forwarding the messages to channels connected to the consumers, which are grouped by machine;
- **Step 8, 9 and 10** - The messages that reach the consumers are forwarded to clients. The cycle continues until the client applications disconnect. After that, the OPC UA client instance created is deleted, and consequently, the connection to the machine OPC UA server is closed.

6.2.2 *Consumer*

Concerning the consumer as an important part of this work, it is explained in more detail, showing the code associated with that module. Therefore, the code related to the consumer is shown in Code Snippet 1 and 2. Thus, it's possible to understand the processes associated with the connection between consumer and client when it establishes a connection with middleware, more specifically, with the Real-time Component application.

Code Snippet 1: Consumer Module - Part 1

```
1
2           global rooms
3           rooms = dict() #dictionary to control and verify all rooms created
4
5           #handler to treat messages received from OPC UA server machine
6           class SubscriptionHandler:
7               def __init__(self, consumer):
8                   self.consumer = consumer
9
10              #method responsible for formatting the json data and sending the new
                ↪ parameters values to the machine group via the channel layer
11              async def datachange_notification(self, node: Node, val,
                ↪ notificationData):    ..4..
12                  await self.consumer.channel_layer.group_send(self.consumer.room_
                    ↪ group_name,{"type": "send.message", "message":
                    ↪ json_string_message},)
13
14          #executed within a thread
15          #method reponsible for creating the subscriptions/connections with OPC UA
            ↪ machine server
16          async def subscribe_parameters(consumer):                    ..3..
17
18              try:
19                  async with client:
20                      handler = SubscriptionHandler(consumer)
21                      #create subscription with all parameters requested by client
22                      subscription = await client.create_subscription(4000,
                        ↪ handler)
23                      ...
24                      # Stay in loop verifying if there is clients in room and
                        ↪ allows the consumer to receive the changes on the
                        ↪ machine.
25                      while(rooms[consumer.room_group_name]["room_has_clients"]):
26                          await asyncio.sleep(1)
27                          await client.check_connection()
28
29                      #delete subscription
30                      await subscription.delete()
31              ...
32          #method invoked to create one thread responsible for creating a OPC UA
            ↪ client instance
33          def thread_execution(consumer):
34              asyncio.run(subscribe_parameters(consumer))
35
36          ... continue ...
37
```

Code Snippet 2: Consumer Module - Part 2

```
class WSConsumer(AsyncWebsocketConsumer):    #Consumer class
    #method responsible for treating the WebSocket connection with client when it connects
    async def connect(self):          ..1..
        #extract room = machineOPCUA_IP (e.g.,192_168_1_1)
        self.room_group_name =
        self.scope['url_route']['kwargs']['room_name']
        ...
        #it validates if the room already exists of the respective machine, if yes, the method validate if the connection is being closed, if yes, the consumer doesn't let the client connects,otherwise, it accepts to establish the connection
        if not self.room_group_name in rooms:
            pass
        elif rooms[self.room_group_name]["room_has_clients"]==False:
            await self.close()#disconnect client, it goes to the disconnect method
            return
        #accept connection
        await self.accept()
    #method invoked when the client send the parameters of machine to be monitored
    async def receive(self, text_data=None, bytes_data=None):  ..2..
        ...
        try:
            if self.room_group_name in rooms and monitor_the_same_parameters(list_of_parameters,self.room_group_name):  #There is room and the parameters are the same
                #join consumer channel to the machine room
                await self.channel_layer.group_add(self.room_group_name,
                self.channel_name)
                rooms[self.room_group_name]["nr_channels"]+=1
            elif self.room_group_name in rooms: #rooms exist but the parameters are different
                ...
                #delete thread and OPC UA client instance connected to respective machine server
                rooms[self.room_group_name]["room_has_clients"]=False
                #start new thread with a new OPC UA client instance to monitor more parameters
                t1 = threading.Thread(target=thread_execution,
                args=(self,))
                t1.start()

            else: #there is no room
                ...
                #initiate thread, with new subscription/ OPC UA client instance
                t1 = threading.Thread(target=thread_execution,
                args=(self,))
                t1.start()
            ...
    #receive message from channel layer of the type "send_message" and send it to the client
    async def send_message(self, event):         ..5..
        message = event["message"]
        await self.send(text_data=message)
    #method called when the client disconnects
    async def disconnect(self, close_code):                  ..6..
        ...
        #update the number of channels in room
        exit_room['nr_channels']-=1
        ...
        #validate the number of clients in room, if 0, the room is deleted
        if exit_room['nr_channels']==0:
            exit_room["room_has_clients"]=False  #this stop, the while in the code above
            time.sleep(1)
            del rooms[self.room_group_name] #delete room from the dictionary
```

In Code Snippet 1 and 2 is possible to understand the following steps:

- **Step 1** - The consumer instance receives a connection from a client and establishes it, considering whether the room is being closed or not. If so, the connection is rejected;
- **Step 2** - The consumer instance receives the parameters of the machine to be monitored, where if this list differs from the parameters subscribed by the existing OPC UA client instance, the current instance is closed, and a new one is created with both the old and new parameters;
- **Step 3** - Following the before step the OPC UA client instance is created and the parameters are subscribed;
- **Step 4** - Receiving from OPC UA machine server the new values of the parameters, they are treated by one unit called SubscriptionHandler that constructs one message and forwards it to the channel layer, where this layer sends the message to all consumers included in a machine-related group/room;
- **Step 5** - Since the message is received by the consumer or group consumers, the method with the name of the event type is executed within the consumer, which in this case is called "send.message". After that, the message is forwarded to the client conected to its consumer;
- **Step 6** - When the client disconnects, the disconnect method is called. It can update the number of consumers in the room/group, and if there are no consumers within it, the OPC UA client instance is deleted. Otherwise, the OPC UA client instance maintains the connection with the machine's OPC UA server.

6.2.3 *Remote Procedure Runner*

Concerning the Remote Procedure Runner component, it is an application with REST API responsible for making operations in the factory machines. It is capable of executing four procedures, which can be accessed from routes presented in Code Snippet 3.

Code Snippet 3: Procedures/URLs of REST API

```
1
2         urlpatterns = [
3             path('machines/<int:id>/opcuatree', views.machineTreeScan),  #PUT
4             path('machines/<int:id>/node', views.changeValue),   #POST
5             path('machines/<int:id>/history', views.activateHistoryRecording),
              ↪  #PUT
6             path('machines/<int:id>/history/isActive',views.isMachineHistoryActi
              ↪  ve),
              ↪  #GET
7         ]
```

The first procedure is responsible for scanning the tree of a certain machine from an created OPC UA client instance, and after that, forwarding the tree collected to a module called Supporting System, which persists that information in PostgreSQL database. In this way, the clients can consult from client applications which parameters are available for monitoring a particular machine. The code of the first procedure is showed in Code Snippet 4.

Code Snippet 4: Tree Scanning Procedure

```
1
2           class My_OPCUA_Client():
3               async def create_client(self):   ..1..
4                   ...
5                   self.opcuaClient = Client(url) #creating OPC UA client instance
6                   ...
7
8               #method for initializing the tree scanning
9               async def persist_history_or_update_machine_tree(self):
10                  ...
11                  await self.constructTreeDict(list(self.treeDictionary.keys())[0],
                    ↪  self.treeDictionary, self.opcuaClient.get_objects_node())
                    ↪  ..2..
12
13                  #the dictionary with a list of parameters is sent to module
                    ↪  "Supporting System"
14                  response = requests.put(url=f"{central_api_url}{self.machine_id}
                    ↪  /tree",json=json.dumps(self.treeDictionary))
                    ↪  ..4..
15                  ...
16              #method to recursively get the list of parameters
17              async def constructTreeDict(self,dictKey, treeDictionary, node):
                ↪  ..3..
18
19                  for childNode in await node.get_children():
20
21                      node_class = await childNode.read_node_class()
22                      browseName = await childNode.read_browse_name()
23
24                      if node_class == ua.NodeClass.Variable:  #if the node is of
                        ↪  type Variable (Properties or DataVariable)
25                          #value for read timestamp
26                          read_value = await childNode.read_data_value()
27                          data_type = await childNode.read_data_type()
28
29                          try:
30                              #  retrieve user access level, each access level
                                ↪  operate over one position bit
31                              accessLevelList = await
                                ↪  childNode.get_user_access_level()
32                              accessLevelValue = 0
33
34                              for access in accessLevelList:
35                                  accessLevelValue = accessLevelValue |
                                    ↪  int(access)+1
36
37                              #add node to the dictionary
38                              treeDictionary[dictKey][browseName.Name] = {"nodeid":
                                ↪  childNode.__str__(), "node_class":
                                ↪  node_class.__str__(),"value":read_value.Value.Va
                                ↪  lue.__str__(),"sourceTimestamp":read_value.Sourc
                                ↪  eTimestamp.__str__(),"serverTimestamp":read_valu
                                ↪  e.ServerTimestamp.__str__(),"midTimestamp":datet
                                ↪  ime.datetime.now().strftime("%Y-%m-%d
                                ↪  %H:%M:%S").__str__(),"data_type_identifier":data
                                ↪  _type.Identifier,"user_access_level":accessLevel
                                ↪  Value}
39                          except Exception as error:
40                              logging.error(f"concatenating parameters:
                                ↪  {browseName}, error: {error}")
41                              continue
42
43                      else:  # if the node is an object
44                          treeDictionary[dictKey][browseName.Name] = {"nodeid":
                            ↪  childNode.__str__()}
45                          await self.constructTreeDict(browseName.Name,treeDiction
                            ↪  ary[dictKey],
                            ↪  childNode)
46
```

Code Snippet 4 shows the steps carried out in this first procedure. So the steps are:

- **Step 1 -** The OPC UA client instance is created;
- **Step 2 and 3 -** A Python dictionary is created, and the Object node is obtained from the OPC UA server of a respective machine, a typical node that contains the machine parameters. Thus, the parameter list can be browsed from the Object node. This one is a predefined node that stores parameters related to sensors and actuators. After that, the tree/parameter list is extracted recursively;
- **Step 4 -** At the end of the capturing process, the tree is persisted in a database PostgreSQL from the module presented in Fig. 35, called Supporting System.

The second procedure allows client modules to change the machine's parameters. Part of this procedure is represented in Code Snippet 5, it can be explained in 4 steps:

- **Step 1 and 2 -** The client module calls the route with information on the node that needs to change. The "setNodeData" method is called. It validates if the node is writable and if the information given by the client matches the original type of the value of the parameter;
- **Step 3 -** Once the information is well defined, the value defined by the client is converted into a compatible type, as it comes in String type;
- **Step 4 -** Finally, the value of the parameter is modified in OPC UA server of the respective machine.

Code Snippet 5: Tree Scanning Procedure

```
        #method for changing the value of a parameter
        async def setNodeData(self,nodeWithRequestedValue:models.Node):   ..1..

            #only create a node instance, it is not synchronized
            nodeInstance =
            ↪  self.opcuaClient.get_node(nodeid=nodeWithRequestedValue.node_id)
            #read the original node
            nodeDataTypeNode = await nodeInstance.read_data_type()
            nodeAccessLevelList = await nodeInstance.get_user_access_level()
            originalNodeDataTypeId = int(nodeDataTypeNode.Identifier)


            if self.isWritable(nodeAccessLevelList) and originalNodeDataTypeId ==
            ↪  int(nodeWithRequestedValue.value_type_id):    #it can be modified
                await nodeInstance.set_value(ua.DataValue(ua.Variant(convertValu
                ↪  eToDataTypeCompatible(originalNodeDataTypeId,nodeWithRequest
                ↪  edValue.value),
                ↪  ua.datatype_to_varianttype(int_type=originalNodeDataTypeId))
                ↪  ))
                ↪  ..4..
            else: #it can not be modified
                raise BaseException(f"writing value in node. Is Writable:
                ↪  {self.isWritable(nodeAccessLevelList)}; Original Node Data
                ↪  Type: {originalNodeDataTypeId} ")

        #method responsible for identifying if the node is writable
        def isWritable(self,accessLevelList:set):   ..2..
            """ Returns information about whether the node can be written (User
            ↪  permissions)"""
            for accessLevel in accessLevelList:
                if accessLevel == ua.AccessLevel.CurrentWrite:

                    return True
            return False
        #convert values (They are received as a String)
        def convertValueToDataTypeCompatible(nodeDataType:int,valueToWrite):
        ↪  ..3..
        """Convert value to type requested. Raise exception when the value is not
        ↪  convertable to format similar to nodeDatatype. 1 - Boolean; 6 -
        ↪  Int32;8 - Int64; 10 - Float; 11 - Double; 12- String"""
            try:
                if nodeDataType == 1:
                    return str_to_bool(valueToWrite)
                elif nodeDataType == 6:
                    return int(valueToWrite)
                elif nodeDataType == 8:
                    return int(valueToWrite)
                elif nodeDataType == 10:
                    return float(valueToWrite)
                elif nodeDataType == 11:
                    return float(valueToWrite)
                elif nodeDataType == 12:
                    return str(valueToWrite)
                else:
                    raise Exception("Invalid Data Type.")
            ...
```

The third procedure stops or initiates the retrieval of all parameter values from a specific machine. Thus, the historical data of each machine is stored in a database, presented in Fig. 35 as Logs Database. So, it's responsible for maintaining the logs of each machine. The procedure involves the same code presented in Code Snippet

4, but in this case the scanning of the elements is executed regularly, every 60 seconds from one Thread Python feature. This procedure only lets clients create one OPC UA client instance to collect the historical data per machine. Finally, the fourth procedure only validates the active state (active or stopped) of the historical procedure for a given machine.

6.3 CENTRAL SERVER

The Central Server module is a server, in this case, a laptop, which hosts 4 important components: Logs Database, Management Database, Supporting System and Web Application. All of them are containers running over the Docker [117]. Each container was downloaded from Docker Hub and had the environment configured to run each application used. This technology enabled the solution owners to deploy the four containers into cloud environments from platforms like Google Cloud Platform, which can house these containers easily [118].

Concerning the Logs Database subcomponent, it represents a database of type MongoDB [119], a NoSQL database, responsible for saving all historical data of the machines, i.e., values of sensors and actuators over time. This data is persisted by the module Middleware. In addition, this type of database was also selected because the data from the machines may not conform to a uniform data model, as there are several machines with different data models, which makes it difficult to store their data in a relational database that requires a structured schema. Furthermore, the NoSQL database was used as it is more optimized for frequent data retrieval and updates [120].

A relational database, specifically PostgreSQL, was integrated for that purpose. This database was selected to store relational data that complies with the predefined model illustrated in Fig. 37. It saves static information to support the whole system, such as machine data, e.g., IP addresses, port numbers, available parameter list of each machine/machine tree, services, and associated data. Therefore, the services of each machine are saved, where there is one service of monitoring per machine that saves the nodeId of each parameter chosen to be monitored from the user applications. Another service is the training service, where one machine can have several, where they guide novel operators to carry out certain operations in the factory. Therefore, this type of service stores a sequence of steps for manufacturing processes along with 3D models that can be visualized in an AR application. This AR application was developed using a separate solution that can be integrated

with the present system. The last service, the instructing service, offers educational content like audio, video, PDF, and text, e.g., how a user should proceed to change the cutting saw of a machine. This service is not as interactive as a training service because the instructions in AR environment show up more intuitive in the real world.

The Supporting System module includes a JA, running over the WildFly application server. It handles all business logic for managing system data, including machine-related data stored in the previously mentioned PostgreSQL database. The Supporting System also provides services to user applications, allowing stakeholders to access and manage machine information. Crucial information includes the connection parameters, required for user applications to connect to the machines through Middleware module.

Finally, the Web application module is a Vue application, which is used by users to monitor in real-time the processes of the machines and to consult and manage information related to machines. An instance of this application is delivered to each browser client on the client side.

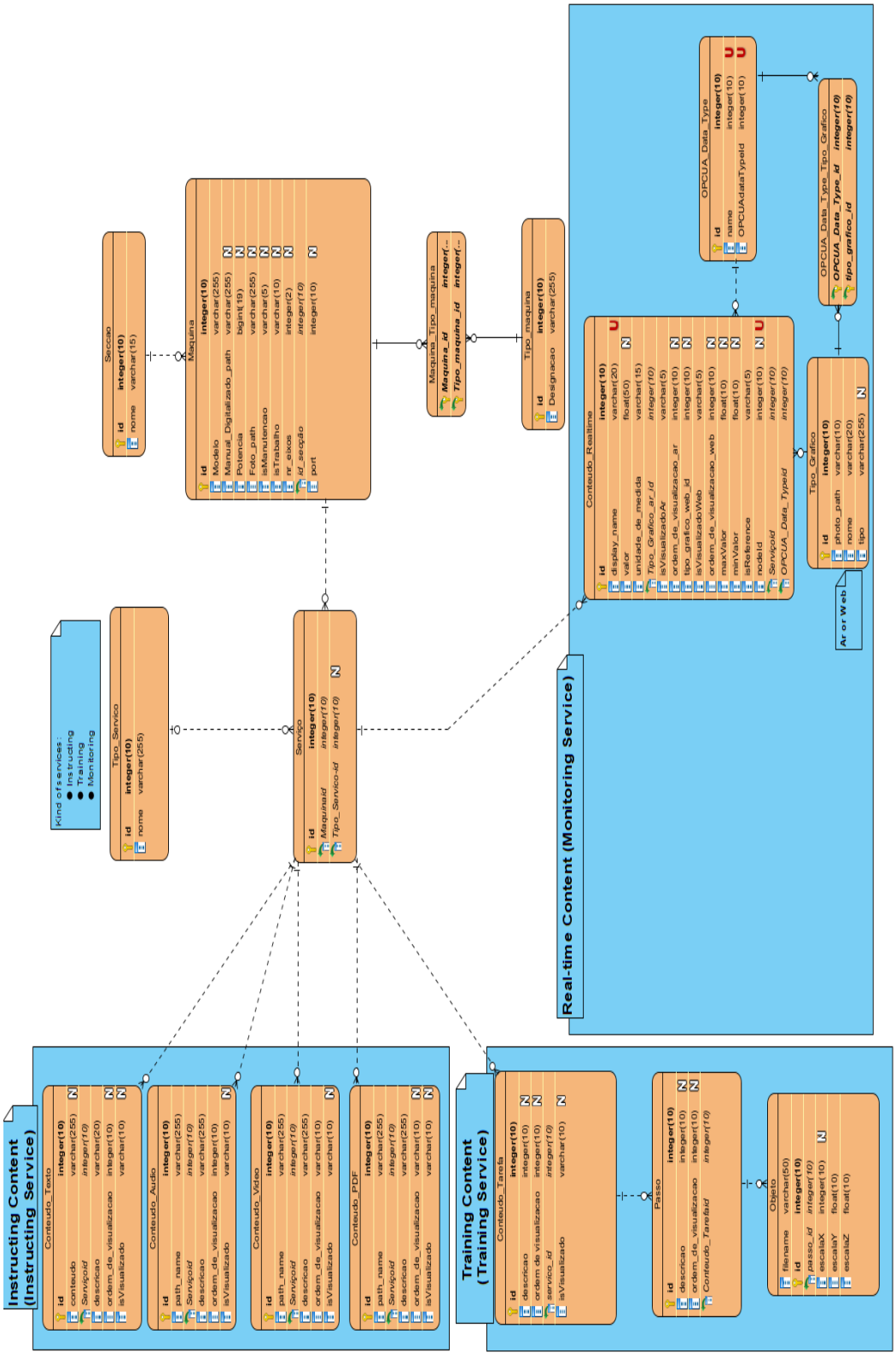

Figure 37: Entity Relationship Diagram

6.4 USER APPLICATIONS

The User applications model represents two types of applications, an AR application to be used in HoloLens (Fig. 38), a mixed-reality headset created by Microsoft, and an application that is accessible via a web browser. Although the communication between OPC UA servers and AR application is different from Web Application instances, that communication was just a proof of concept for use of the AR application on the factory floor and is not the subject of the present research. Therefore, all modules that want to receive data from machines must always communicate with Middleware because of the reasons mentioned.

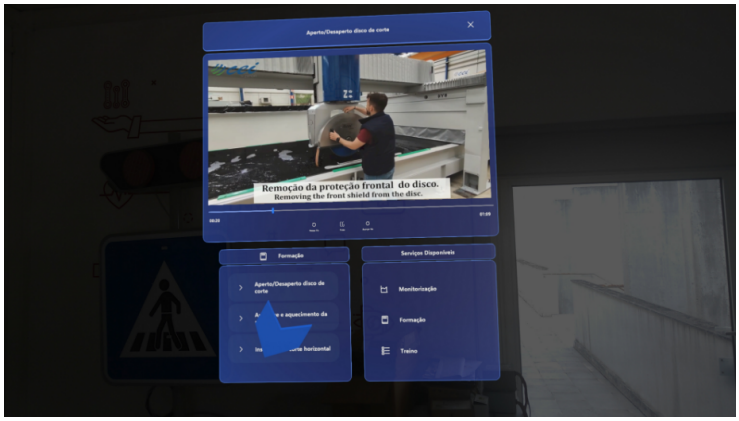

Figure 38: AR Application Interface

As mentioned, a Web Application Instance in this module is one instance of Web Application, which runs over Central Server. This application aims to support operators on the shop floor to keep the normal function of the production line from real-time charts (Fig. 39) and historical charts (Fig. 40) of the machines in the factory. This application was designed to be browser-based to accommodate the diverse needs of factory users. Office people, primarily using laptops, and shop floor operators, where mobile devices are more suitable, can both access the application. A browser-based approach ensures compatibility with those devices. In addition, the application has other functionalities, where the users can visualize which machines are available for work and under maintenance through the page presented in Fig. 41 and Fig. 42. Therefore, the page shown in Fig. 42, which is used for selecting a machine to monitor (Fig. 39), allows the users to understand which machines are in maintenance from rounded symbols for each machine, where a green rounded symbol means that the machine is not under maintenance and red vice versa.

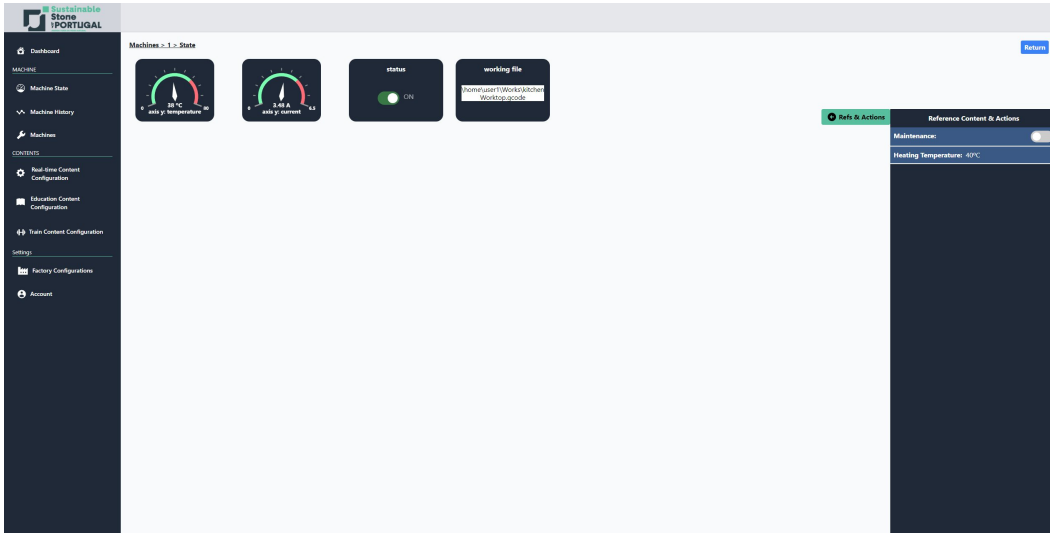

Figure 39: Web Application: Real-time Data Charts

Figure 40: Web Application: Historical Data Charts

In addition, the Web application allows the users to integrate new machines into the system, from the page presented in Fig. 43. Moreover, when a user adds one from this page, a set of communications among modules of the system are realized, these are shown in the sequence diagram in Fig. 44.

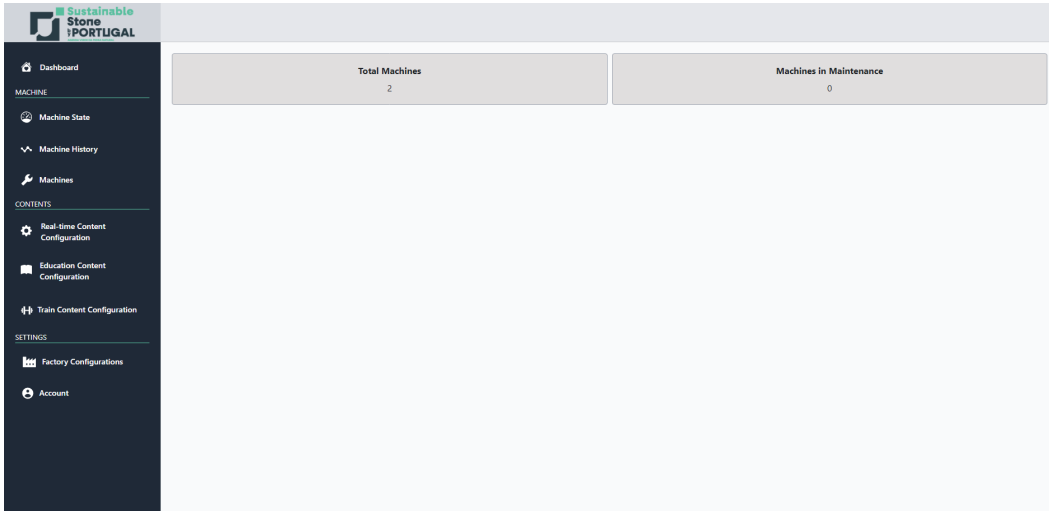

Figure 41: Web Application: Dashboard

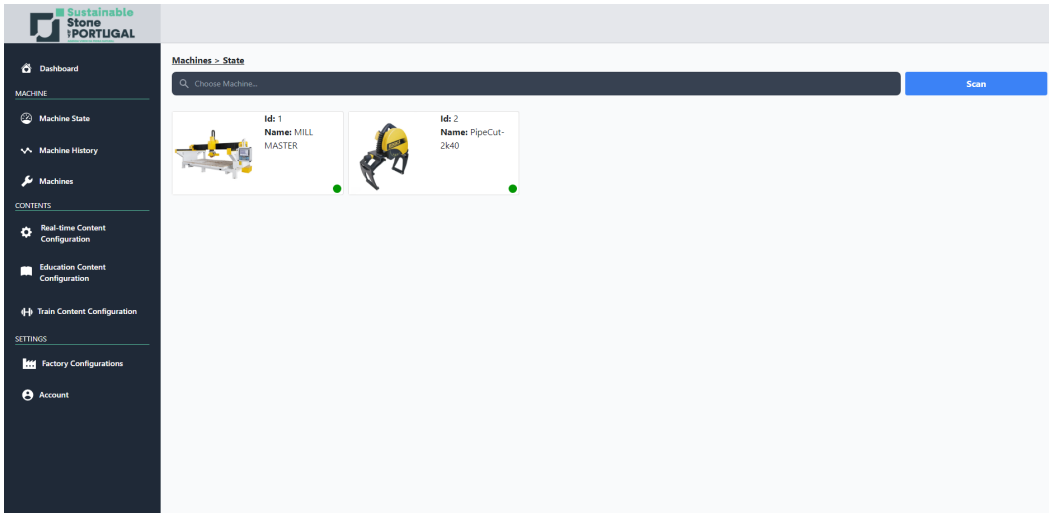

Figure 42: Web Application: Selectable List of Machines

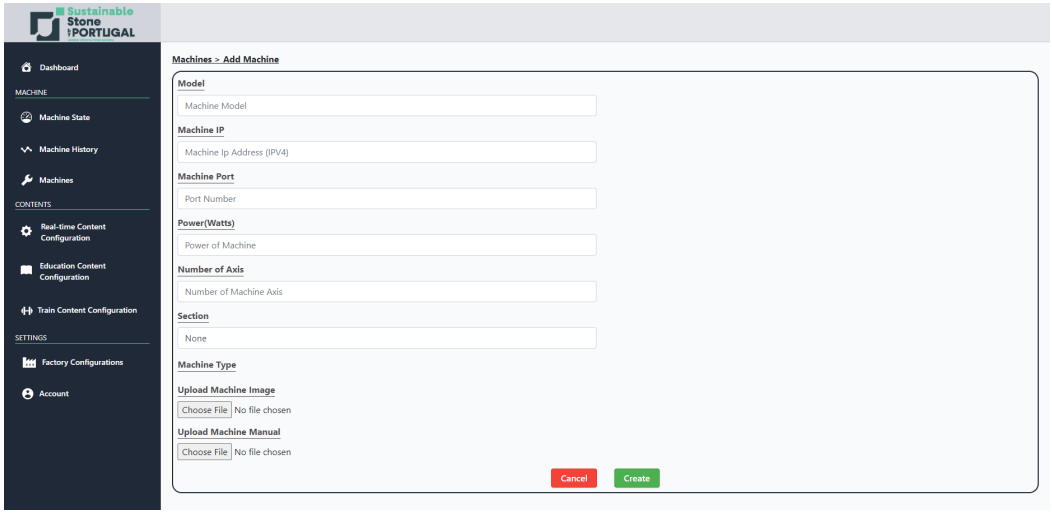

Figure 43: Web Application: Integrate Machine

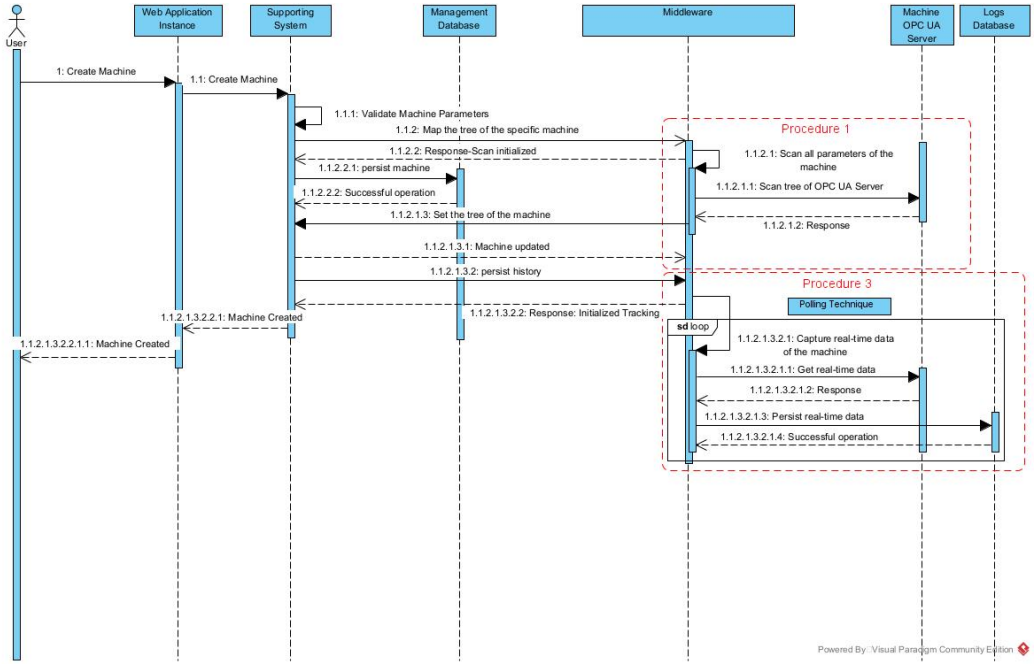

Figure 44: Machine Integration Flow - Sequence Diagram

As shown in Fig. 44 from sequence diagram, it is possible to identify the main procedures of the Middleware module, explained in Subsection 6.2.3: the first to obtain all parameters of the machine from Machine OPC UA server of the respective machine and third to capture all values of parameters over time.

In short, considering the sequence diagram, the workflow starts when the machine is added through the Web application (Fig. 43). After that, during the process of creating the machine in the Supporting System, this module notifies Middleware to capture the list of parameters of a machine. Ending this first procedure, the

Supporting System persists the machine into the Management Database. After that, the second procedure is activated where the Supporting System notifies Middleware to activate it, which from the polling technique persists historical data into the Logs Database. After these two procedures, the parameters can be selected on the Web application to be monitored, which can be made from the page presented in Fig. 45.

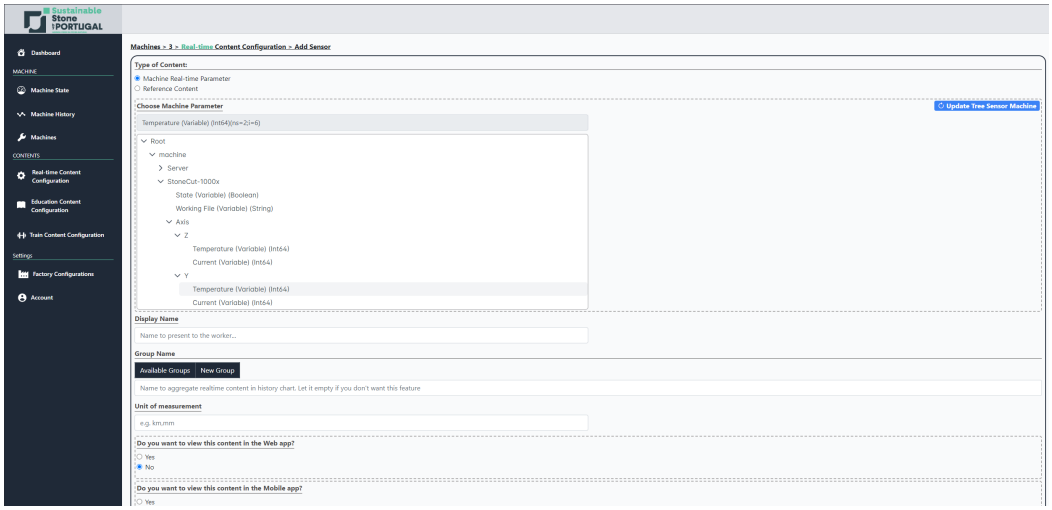

Figure 45: Web Application: Selecting of a Machine Parameter to Monitor.

7

TESTS

This section evaluates the capabilities of the developed solution. Technical validation tests assess the functionality of features like monitoring and controlling the physical twin. Performance indicators, such as communication delays, are measured to validate real-time capabilities so that comparisons can be made with other solutions with similar objectives. Additionally, usability tests are conducted with various users on the web application to understand user experience. In addition, the results of these tests are analyzed and discussed. Finally, it should be noted that the tests were carried out in a laboratory environment, as it was not possible in a factory environment due to restrictive rules.

7.1 TECHNICAL VALIDATION TESTS

Regarding the technical validation tests, are executed three tests: one to measure the communication delay between OPC UA Server and Middleware, the second to understand if the solution allowed the users to change the state of the physical twin from OPC UA Server attributed, and the last, to evaluate the capacity of the solution to cover multiple OPC UA servers simultaneously. The test setup is presented in Fig. 46, and the configurations of each hardware used are described in Table 2. Moreover, all modules mentioned communicate using wired connections in these tests.

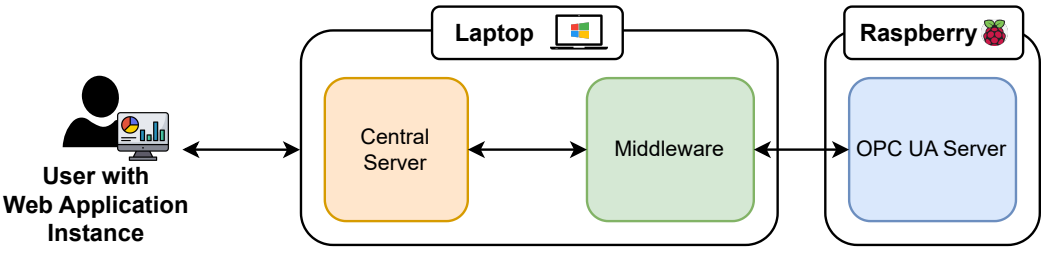

Figure 46: Setup for the Technical Validation Tests.

Table 2: Test-bed to assess the performance of the system developed.

Hardware	Software	Environment
Lenovo Legion 5 15ARH05H - 16 RAM, AMD Ryzen 7 4800H	Windows 11 Version 23H2	Lab.
Raspberry Pi 3 Model B V1.2	Raspberry Pi OS Lite (64-bit)	Lab.
Router Fiber Gateway GR241AG	Default	Lab.

Concerning the module "OPC UA Server", it was created from the package provided by FreeOpcUa GitHub repository [115], and the parameters created within it simulated the parameters of one stone machine on a factory floor, where the parameters were: the temperature for each axis, state of the machine, working or not, the current in each axis, and the working file, which describe what work the machine are performing. All the values of these parameters were generated randomly. Therefore, the tree/list of parameters of the present OPC UA server for the test is presented in Fig. 47, extracted from OPC UA client [121].

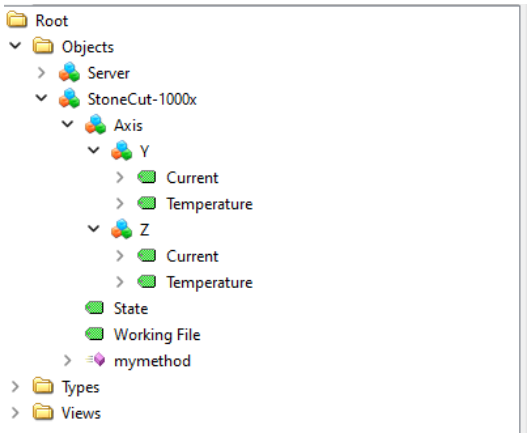

Figure 47: Tree of OPC UA Server Developed for the Tests.

7.1.1 *Latency Test*

The performance test explained here aimed to measure the latency between the OPC UA Server and the Middleware, more specifically, the latency between layers 3 and 4, considering the architecture presented in Fig. 26. Therefore, the Middleware was modified so that latency could be measured. In addition, the test had two situations: in the first, the Middleware had to retrieve the value of a sensor several times, and in the second, the Middleware had to scan the server's parameter list OPC UA, also several times. The time elapsed between sending the request and

receiving the response was recorded in both tasks. The times obtained are shown in Table 3.

Table 3: Latency between Middleware and OPC UA Server.

Environment	Task	Experiment Number	Number of Requests	Mean Time Elapsed per Request (ms)	General Mean Time Elapsed (ms)	Standard Deviation (ms)	Mean Standard Deviation (ms)
Laboratory	Retrieve Sensor Value	1	10	5.312	5.424	0.723	0.616
		2	100	5.481		0.530	
		3	1000	5.433		0.600	
		4	10000	5.471		0.611	
	Scan Machine Parameters	1	10	4908.441	4949.967	15.786	43.168
		2	100	4913.174		22.322	
		3	1000	4964.018		30.518	
		4	10000	5014.233		104.044	

As shown in Table 3, the first task had the average elapsed time between communication between the two layers was 5.424 ms with a standard deviation of 0.616 ms, which makes the performance of the two modules very stable compared to the number of requests sent in a row. The second task, as predictable, took more time, with an average time of 4949.967 ms and a mean standard deviation of 43.168 ms.

7.1.2 *Scalability Test*

The present test consists of having the Middleware covering multiple physical entities simultaneously. Therefore, three OPC UA Servers were initialized in three different devices: one on the initial laptop, another on Raspberry, and the last one on a different laptop (Toshiba SATELLITE-C850-1G3,Linux Mint OS 20.3, 4GB RAM, Celeron 1000M). In addition, 3 Web Application instances were started on initial laptop, one per OPC UA Server, as shown in Fig. 48. The result of the experiment is that the solution is capable of covering various machines/physical twins in parallel.

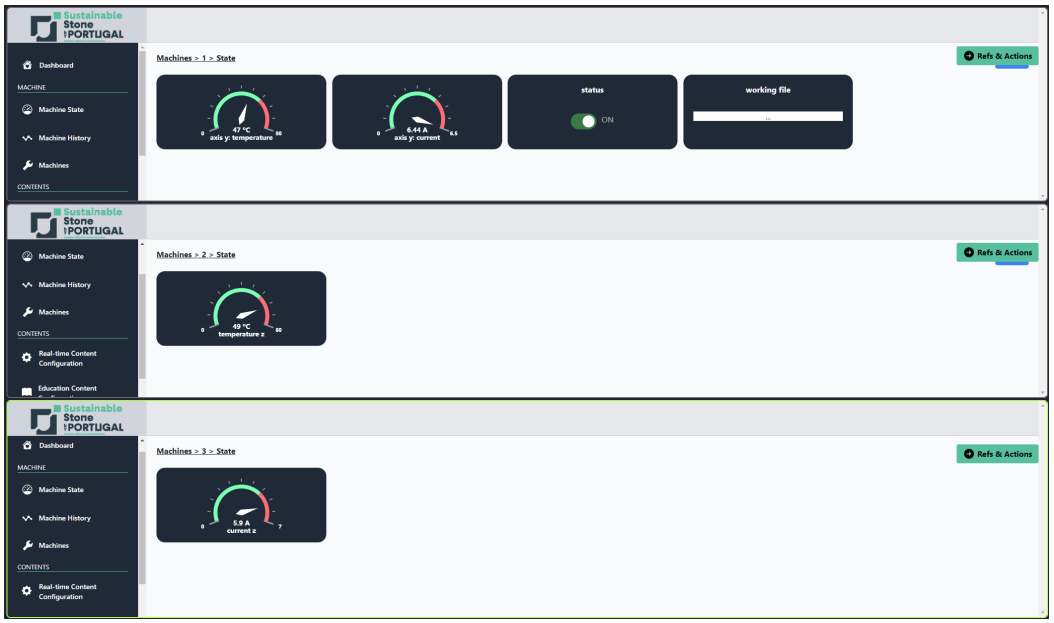

Figure 48: Interaction between 3 OPC UA Servers and Middleware, serving 3 Web Clients.

7.1.3 *Control Test*

To evaluate the solution's ability to control a physical twin, one LED was added to Raspberry Pi, as shown in Fig. 49, and the OPC UA server running on the Raspberry Pi was modified to include two additional parameters. As depicted in Figure 50, these parameters are associated with a single LED connected to the Raspberry Pi. The 'Blinking Interval' parameter lets users specify the duration for which the LED remains illuminated. The 'State' parameter provides control over the LED's on/off status.

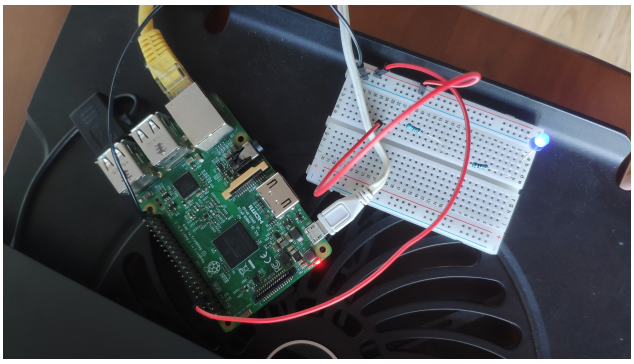

Figure 49: Raspberry Pi and Led Setup.

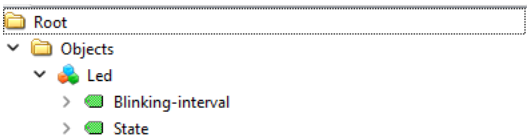

Figure 50: Raspberry's Pi OPC UA Stack for the Control Test.

From this test, it was possible to see that the solution allows users to modify the parameters of the physical twin/device integrated into the system. This is demonstrated in a video file that accompanies this document, which shows the process of changing the status of the LED connected to the Raspberry Pi.

7.2 USABILITY TESTS

To assess the usability of the web application and the entire system, a usability test was created, based on the usability testing guidelines presented by the Portuguese government entity called TicAPP [122]. The instrument used to help evaluate the application was a questionnaire, which was divided into 5 sections and answered by 20 participants

The first section of the questionnaire aims to extract demographic data related to participants. As shown in Fig. 51, the participants' group consisted of 13 men and 7 women, with an average age of 27. Also, most of them had the high school level, a total of 16 participants. In addition, the questionnaire also included questions about industrial and mobile experience. As shown in Fig. 52, all participants were familiar with smartphones, and 4 had prior industrial experience.

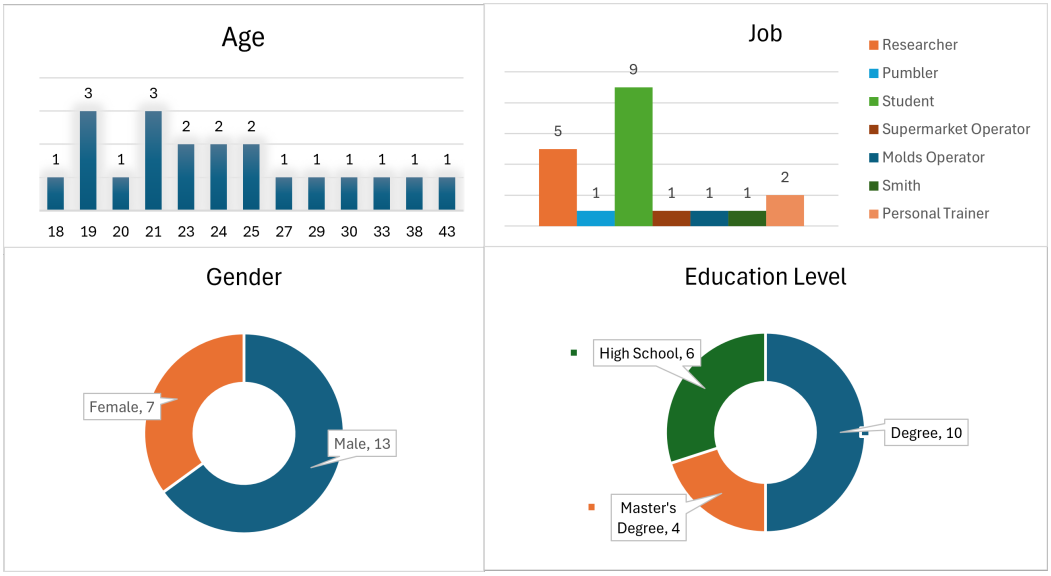

Figure 51: General Demographic Section - Part 1

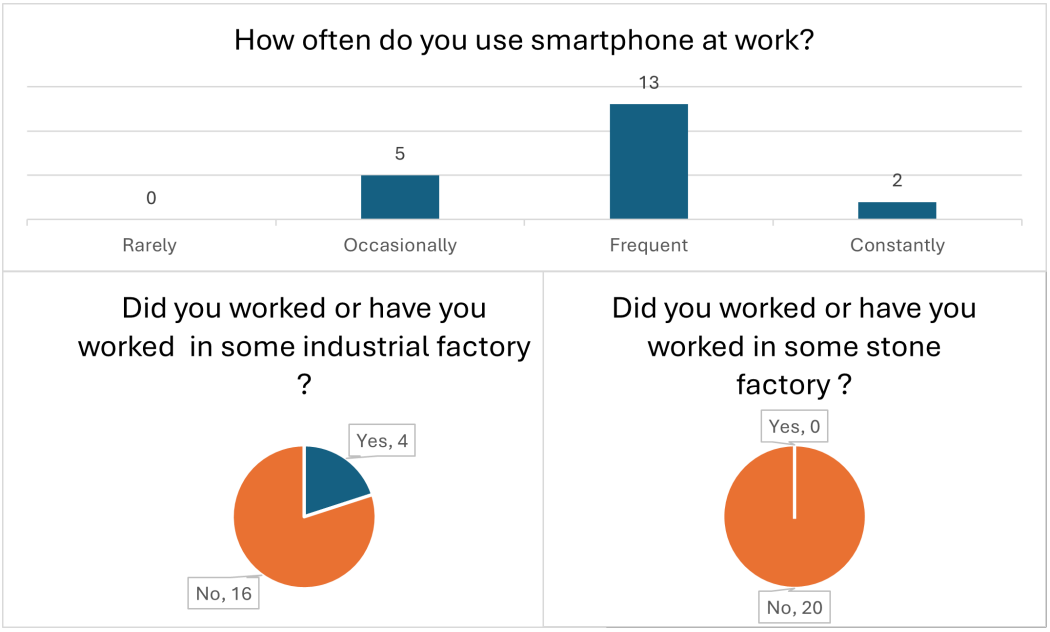

Figure 52: General Demographic Section - Part 2

The second section of the questionnaire focused on industrial environment questions, answered only by those with industry experience. The results, depicted in Figure 53, show three people with extensive experience in this sector, with 2, 3, and 5 years in the profession and with a neutral comfort level for operating industrial machinery. In addition, some time is wasted in this sector monitoring various machines, around 2 to 3 minutes, 0 to 20, and 21 to 40 seconds. However, there is a person who doesn't monitor more than one machine, and also a person who doesn't do any monitoring at all.

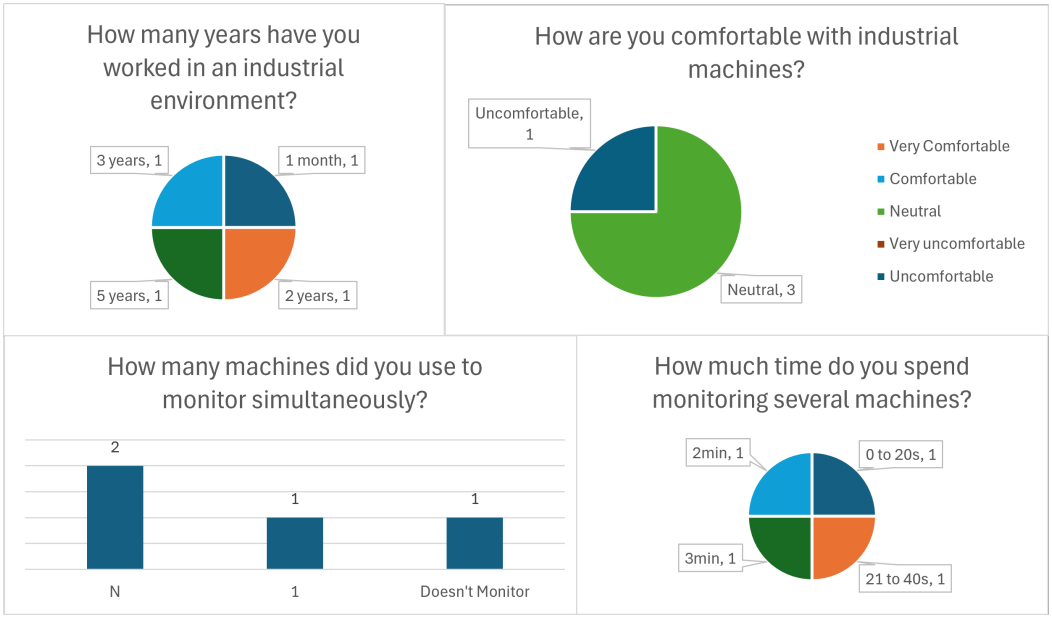

Figure 53: Industrial Demography Section - Questions and Results

The third section included 9 tasks, shown in Table 4, which the participants had to perform over the Web application developed, where this process aimed to understand the level of user experience of the application. Each task was rated in terms of its ease of execution through a 7-point Likert scale (1 - very difficult and 7 - very easy), called Single Ease Question (SEQ), which is a recommended scale to measure the ease of performing a given task [123]. In addition, some tasks contained questions to be answered by the participants so that the researcher could see if the task had been carried out successfully. This section forced the researcher to be around the participants, to catch the reactions of the participants and register some comments said out loud from them. Also, this section included one free-response question, where the participants could give suggestions and register some bugs found.

Table 4: Questionnaire Tasks

Task Id	Description	Contains Question
1	With the web application open, enter the number of machines integrated into the system.	Yes
2	Integrate a new machine into the system with a few parameters provided.	Yes
3	Once the machine has been integrated, check and indicate how many machines are available in the application.	No
4	Access to the page where you can select a parameter and create real-time content to monitor the added machine.	No
5	As a result of the previous task, view all the parameters of the added machine.	No
6	On the same page, select the “Temperature” variable, belonging to the axis Z, and indicate the variable value type.	Yes
7	Create and configure the real-time content with the rest of values.	Yes
8	Once the content has been added, find and indicate the range of its value in real-time.	Yes
9	Consult the historical data of the parameter added to the machine and indicate the range in which its value is inserted.	Yes

As a result of the level of ease of the task, an average of 5.75 and a standard deviation of 0.62 were obtained. These results were obtained by discarding the classification of the level of ease of incorrectly answered user tasks, to be more accurate. In addition, the participants left comments on the free-response question, where some users identified that the page intended to create new real-time content (Fig. 45) should be more intuitive, as when it’s necessary to select one parameter of a machine, the user doesn’t understand that have to expand the root node to view the others parameters. Another observation related to this page was that the unit type of the parameter defined by a user should be based on a predefined list. Additionally, a participant commented that the dashboard (Fig. 41) should be more useful and with better UI. Regarding the positive comments, some participants thought the web application had a good layout and a good UX. Also, there was an observation in which the participant indicated that the application would be useful in the situation he worked in where he had to constantly check machines to see if there was a lack of raw materials to process. Finally, most of the participants could complete the tasks, with 11 of the participants answering well to all the questions, which some tasks included.

This questionnaire also included the post-task questions in section 4. This question belongs to a method called SUS, which makes it possible to evaluate the global level of usability of the UI developed. Also, this type of evaluation serves as a complement

of SEQ evaluation [124]. The questions in SUS methodology were adapted to the European Portuguese version, according to an article published for that purpose [125]. From the results presented in Fig. 54 is possible to extract the average score of 68.5 and a standard deviation of 13.3 can be obtained, with a minimum of score of 40 and a maximum of 92.5. In addition, translating the average score into an adjective, the usability of the interface can be considered OK, meaning it has space for improvements [126, 127]. This classification is based on the scale presented in Fig. 55.

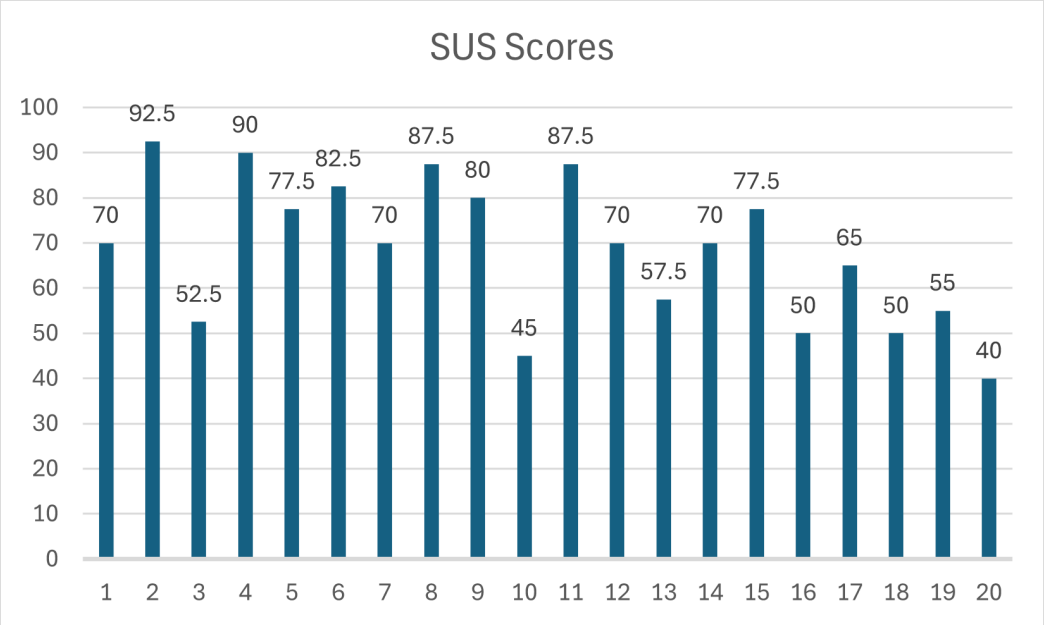

Figure 54: SUS results.

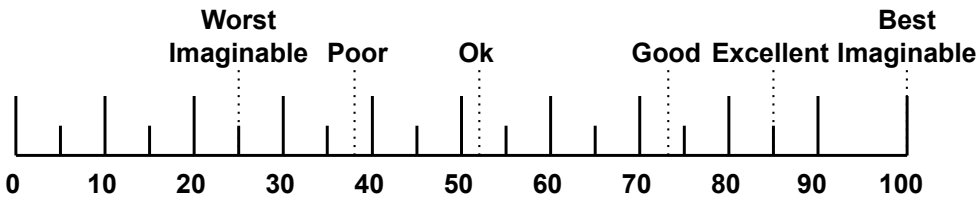

Figure 55: SUS Score. Adapted from [126].

The last section, section 5, included questions related to the web application and industrial environment, which were answered by people with some experience in industrial environments. The questions related to this section are presented in Table 5.

Table 5: Final Consideration Section - Questions.

Id	Question	Possible Responses
1	What are the principal benefits of using the present solution in an industrial environment?	It increases productivity; It contributes to timely maintenance and repair; It allows factories to find functioning patterns; It increases awareness of the operational process; It increases the effectiveness of the operational process.
2	What obstacles do you foresee in the process of adopting the solution?	The Web application is not very intuitive; Technical limitations; Security and privacy of personal data; Lack of instruction and support; Resistance from coworkers; Discomfort and inconvenience when using the devices; The quality and resistance of equipment in an industrial environment.
3	How interested are you in including the solution in your daily work?	5-point Likert Scale (1 - Not very interested and 5 - Very interested)
4	Would you use the solution in your daily work to support the production process?	5-point Likert scale (1 - Totally disagree and 5 - Totally agree)
5	Do you think you will save more time during the monitoring process by using this application?	5-point Likert scale (1 - Totally disagree and 5 - Totally agree)

Fig. 56 shows that most of participants agree that the solution helps to increase awareness of the operational process (Question 1). About question 2, the participants agree that the difficulty of integrating this solution into the production line is due to the lack of instructions and support to educate the operators. Question 3 shows that all 4 participants involved in the industrial environment are interested and would use the solution in the production line. Also, both agree that this solution saves some time in production processes (Question 5).

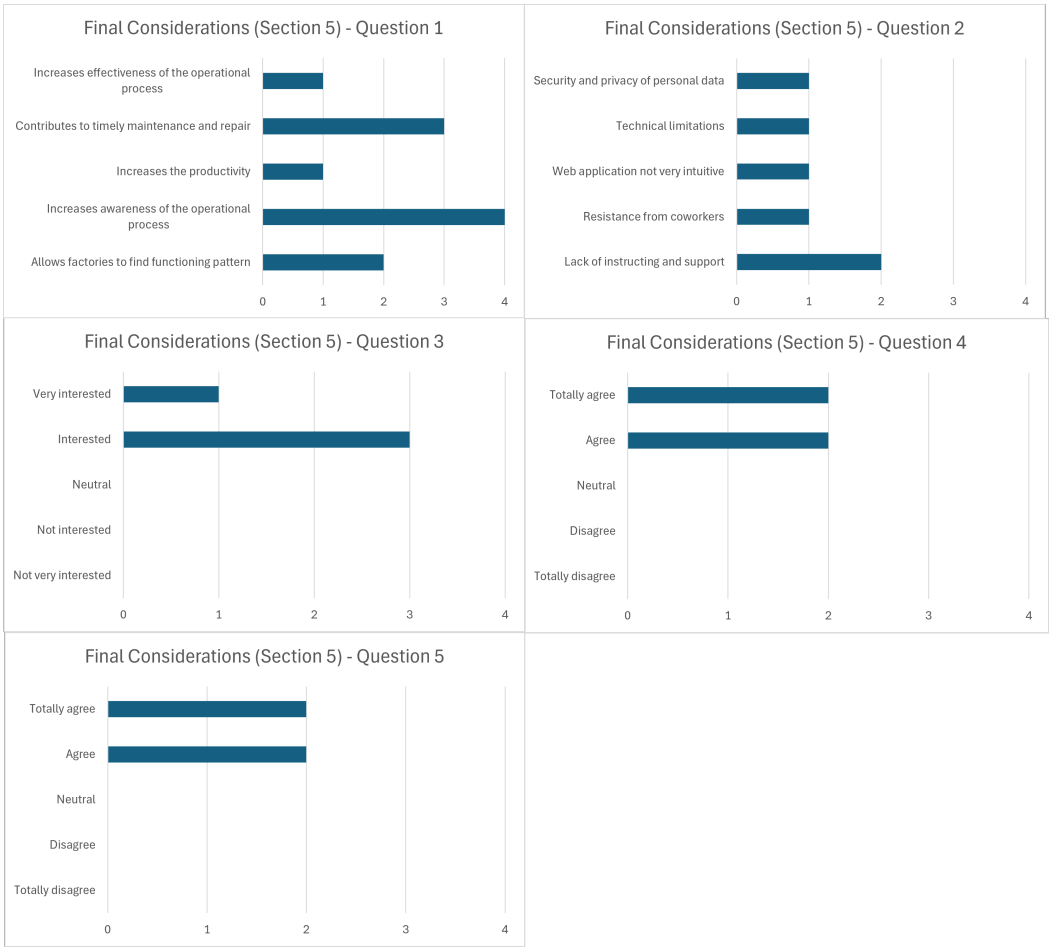

Figure 56: Final Consideration Section - Results

7.3 DISCUSSION

Regarding the results obtained technical validation and usability tests, several conclusions can be drawn. These will help future developments in this industrial area.

Starting with the technical validation tests results, regarding the latency, it was demonstrated real-time communication intervals results obtained to acquire real-time values, the first task of Table 3, can be considered a good result with low latency [128], having less latency than other solutions.

Starting with the results of the technical validation tests, the latency demonstrated effective real-time communication intervals for acquiring real-time values. The first task in Table 3 shows a low-latency result, which can be considered a strong outcome, outperforming other solutions in terms of latency. For example, the solution presented by Habib et al. [17], where the latency declared between the data

source and the module functioning as a middleware (Fig. 18) was in the range of 35.7 ms and 52.6 ms, against the range of the present solution of 5.31 ms and 4.48 ms. Also included in the validation tests was the scalability test, which validated the middleware's ability to handle multiple communications. This test showed that the middleware is capable of communicating with several machines and updating the clients according to changes in machine values. Thus, the solution is capable of interact with machines/physical entities dynamically, not being the solution programmed to a set of machines, which happens with several solutions [87, 20]. The last test included in validation tests is called the control test. It test proved that the solution is capable of changing the state of physical entities at the request of a user, a requirement much discussed in academic papers about digital twins, as mentioned in Section 1. Many of the academic solutions published only allow one-way communication, i.e., they are only concerned with extracting information from machines [81].

Concerning the usability tests, these were carried out with some people involved in industrial environments, which made it possible to obtain more information about the difficulties felt in industries. It was also possible to extract that there is time wasted by operators in the machine monitoring process. In this way, the solution developed can help, as it makes the operator move only when necessary. Also, from the tasks carried out by the participants, it was possible to extract that the web application has space to evolve, given the 68.5 rating acquired by the SUS method, even though several participants completed all the tasks. Therefore, future improvements at the level of forms and UI can be improved. Finally, industry participants showed interest and considered that the solution offers significant potential to enhance the production process by increasing manufacturing awareness, ensuring timely maintenance, and improving overall production efficiency. However, issues such as lack of instruction about the application must be taken into account to produce a good solution. These aspects can be resolved by inserting tutorials into the application to guide users and thus help people understand the platforms developed.

8

CONCLUSIONS AND FUTURE WORK

Currently, it is difficult for SMEs to keep up with developments due to monetary costs, which is why there is a lack of digitization in the stone industry. In this way, companies can't be as aware of their production line, making improvements difficult. Thus, several DT solutions have emerged to overcome this need. However, these solutions don't meet all the requirements demanded, as a lot of them are created to accommodate a single or a limited set of machines with some characteristics. Therefore, they cannot adapt to cover all machines within a production line. Another issue with these solutions is that they do not allow stakeholders to perform essential tasks. For example, the solutions do not enable stakeholders to control the production line and monitor it simultaneously. Thus, the present work intends to present one solution to overcome these gaps and help industries to be more aware of the manufacturing process.

The developed solution allows the companies to integrate machines into a digital system without much effort and knowledge, creating a DT of each machine. Thus, the statuses of each machine can be accessed via a developed Web application, available on any device connected to the internet, to fit every work environment. This way, stakeholders are more in tune with the work done. In addition, they can adjust the production lines according to demand, setting the parameters of machines via Web application. This way, the operators who monitor the several machines on the shop floor have their work made easier, as they don't have to go to the machines to check their status so regularly, saving time for other production processes.

Furthermore, the solution is prepared to save the data of the machine over time. This data can be consulted via the developed Web application, which can help operators extract patterns in the machines, contributing to the detection of machine maintenance signals, which can avoid problems. In addition, with this feature of saving historical data, the solution in the upcoming iterations can integrate new services that can explore these data from technologies such as AI, giving more understanding of the production process to stakeholders.

Regarding the tests performed on the solution, the technical validation demonstrated that it has a low real-time delay, allowing users to be synchronized with the production process of each machine effectively. Additionally, the tests showed that the solution can dynamically cover multiple machines simultaneously, addressing a gap found in other research. The solution also demonstrated that physical devices can be reconfigured through a web application, independent of the data model of each machine. As for the user tests executed against the Web application, the SUS metric highlighted that some UI elements could be made more intuitive to reduce resistance from certain users. Despite this, most participants completed the tasks without issues, and the overall UI and UX were rated as satisfactory.

In short, the current work presents a possible implementation of a digital twin DT-based system that encompasses its key features, including bidirectional communication and the ability to manage multiple machines within the production lines. Additionally, the system enables the representation of various physical devices in a virtual space, achieving a true digital twin implementation.

In terms of scientific contributions, this work offers an overview of the state of the art in digital twins and proposes an architecture that, while originally designed for the stone industry, can serve as a foundation for future solutions across several industrial sectors. The scientific content was published in the proceedings of the International Conference on Graphics and Interaction (ICGI) [1].

Furthermore, as future work, the middleware module of the solution must be adapted to maintain resilience, i.e., in any state of unavailability of the module, another instance of it is created. This can be achieved using technologies such as Kubernetes [129], a container orchestration system capable of managing all the modules of a solution, keeping them operational and redundant in case of failure. In addition, with this sort of technology, several instances of the Middleware module can be created to increase the performance of the solution in cases where there are many machines in the factory and multiple access to them by the stakeholders. In this case, to maintain consistency, one real-time communication per machine should exist, i.e., for a set of instances, there should only be one communication for each machine to avoid overloading the OPC UA servers of machines. Thus, a reverse proxy must be incorporated, which aims to forward the communication of clients to the correct instance of the Middleware module.

In addition, AI algorithms can be integrated into this solution to predict equipment faults and maintenance from the historical data collected from each machine [81]. Also, the malfunction of machine tools can be predicted using algorithms, such as

the level of wear of a cutting saw or drill [13]. Therefore, these estimates would give stakeholders more valuable information about the production lines, contributing to a more optimized workflow.

Last, for future work, a complete AR application can be integrated into this solution so that the level of UX is better because as demonstrated in article [87] the users tend to be more efficient with that sort of interface. This technology integrated with AR glasses can help the specialized operators to keep up to date with the production line process in hands-free mode and perform procedures in some machines following guides.

BIBLIOGRAPHY

[1] D. Francisco et al. "Augmented Reality and Digital Twin for Mineral Industry". In: 2023 International Conference on Graphics and Interaction (ICGI), Nov. 2023. DOI: 10.1109/ICGI60907.2023.10452719.

[2] *Indústria 4.0 | Fase II*. Accessed: Apr. 30, 2024. URL: https://www.iapmei.pt/Paginas/Industria-4-0-Fase-II.aspx.

[3] Stone by Portugal. *Sobre – Stone by Portugal*. https://sustainable.stonebyportugal.com/sobre/. Accessed: Sep. 24, 2024. 2024.

[4] J. Kraaijenbrink. *What Is Industry 5.0 And How It Will Radically Change Your Business Strategy?* Accessed: Mar. 10, 2024. May 2022. URL: https://www.forbes.com/sites/jeroenkraaijenbrink/2022/05/24/what-is-industry-50-and-how-it-will-radically-change-your-business-strategy/.

[5] *Industry 4.0 - Digitalisation for productivity and growth*. Accessed: Mar. 11, 2024. Sept. 2015. URL: https://www.europarl.europa.eu/RegData/etudes/BRIE/2015/568337/EPRS_BRI(2015)568337_EN.pdf.

[6] M. Javaid, A. Haleem, and R. Suman. "Digital Twin applications toward Industry 4.0: A Review". In: *Cognitive Robotics* 3 (Jan. 2023), pp. 71–92. DOI: 10.1016/j.cogr.2023.04.003.

[7] Kasey Panetta. *Gartner Top 10 Strategic Technology Trends For 2019*. Accessed: Mar. 16, 2024. 2018. URL: https://www.gartner.com/smarterwithgartner/gartner-top-10-strategic-technology-trends-for-2019.

[8] W. Kritzinger et al. "Digital Twin in manufacturing: A categorical literature review and classification". In: *IFAC-Pap.* 51.11 (Jan. 2018), pp. 1016–1022. DOI: 10.1016/j.ifacol.2018.08.474.

[9] Y. K. Liu, S. K. Ong, and A. Y. C. Nee. "State-of-the-art survey on digital twin implementations". In: *Adv. Manuf.* 10.1 (Mar. 2022), pp. 1–23. DOI: 10.1007/s40436-021-00375-w.

[10] A. Hazrathosseini and A. Afrapoli. "The advent of digital twins in surface mining: Its time has finally arrived". In: *Resources Policy* (2023). DOI:

10.1016/j.resourpol.2022.103155. URL: https://doi.org/10.1016/j.resourpol.2022.103155.

[11] F. Tao and M. Zhang. "Digital Twin Shop-Floor: A New Shop-Floor Paradigm Towards Smart Manufacturing". In: *IEEE Access* 5 (2017). DOI: 10.1109/ACCESS.2017.2756069.

[12] Bernard Marr. *7 Amazing Examples of Digital Twin Technology In Practice*. Accessed: Apr. 30, 2024. URL: https://www.forbes.com/sites/bernardmarr/2019/04/23/7-amazing-examples-of-digital-twin-technology-in-practice/.

[13] W. Polini and A. Corrado. "Digital twin of stone sawing processes". In: *International Journal of Advanced Manufacturing Technology* (2020). DOI: 10.1007/s00170-020-06384-6. URL: https://doi.org/10.1007/s00170-020-06384-6.

[14] Chiara Cimino, Elisa Negri, and Luca Fumagalli. "Review of Digital Twin Applications in Manufacturing". In: *Computers in Industry* 113 (2019), p. 103130. DOI: 10.1016/J.COMPIND.2019.103130.

[15] "Digital Twins: A Survey on Enabling Technologies, Challenges, Trends, and Future Prospects". In: *IEEE Communications Surveys & Tutorials* 24.4 (2022). DOI: 10.1109/comst.2022.3208773.

[16] H. Arnarson, B. Solvang, and B. Shu. "The application of open access middleware for cooperation among heterogeneous manufacturing systems". In: *2020 IEEE 7th International Conference on Industrial Engineering and Applications (ICIEA)*. 2020, p. 6. DOI: 10.1109/SIMS49386.2020.9121537.

[17] K. Habib et al. "An Aggregated Data Integration Approach to the Web and Cloud Platforms through a Modular REST-Based OPC UA Middleware". In: *Sensors* 22.5 (Jan. 2022). DOI: 10.3390/s22051952.

[18] Y. K. Liu, S.-K. Ong, and A. Nee. "State-of-the-Art Survey on Digital Twin Implementations". In: *Journal of Intelligent Manufacturing* 10.1 (2022). DOI: 10.1007/s40436-021-00375-w.

[19] *OPC UA in the Reference Architecture Model RAMI 4.0 – OPC Connect*. Accessed: Jul. 15, 2024. URL: https://opcconnect.opcfoundation.org/2015/06/opc-ua-in-the-reference-architecture-model-rami-4-0/.

[20] A. J. H. Redelinghuys, A. H. Basson, and K. Kruger. "A Six-Layer Architecture for the Digital Twin: A Manufacturing Case Study Implementation". In: *Journal of Intelligent Manufacturing* 31.6 (Aug. 2020), pp. 1383–1402.

[21] OPC Connect. *Should I Use OPC UA or MQTT or AMQP?* Accessed: February 2, 2024.

[22] P. T. Ho et al. "Study of Augmented Reality Based Manufacturing for Further Integration of Quality Control 4.0: A Systematic Literature Review". In: *Applied Sciences* 12.4 (Jan. 2022). DOI: 10.3390/app12041961.

[23] Atlassian. *Agile vs. waterfall project management.* Accessed: Jun. 27, 2024. URL: https://www.atlassian.com/agile/project-management/project-management-intro.

[24] Atlassian. *Kanban.* Accessed on: Jun. 13, 2024. URL: https://www.atlassian.com/agile/kanban.

[25] Trello. *Manage Your Team's Projects From Anywhere | Trello.* Accessed on: Jun. 13, 2024. URL: https://trello.com/.

[26] Atlassian. *Continuous delivery starts on the Jira kanban board.* Accessed on: Jun. 13, 2024. URL: https://www.atlassian.com/software/jira/features/kanban-boards.

[27] Notion. *Your connected workspace for wiki, docs projects.* Accessed on: Jun. 13, 2024. URL: URL:%20https://www.notion.so.

[28] Atlassian. *Working with WIP limits for kanban.* Accessed on: Jun. 13, 2024. URL: https://www.atlassian.com/agile/kanban/wip-limits.

[29] Atlassian. *What is version control.* Accessed on: Jun. 17, 2024. URL: https://www.atlassian.com/git/tutorials/what-is-version-control.

[30] GitHub. *Build software better, together.* Accessed on: Jun. 17, 2024. URL: https://github.com/.

[31] Atlassian. *What is DevOps?* Accessed on: Jun. 17, 2024. URL: https://www.atlassian.com/devops.

[32] GitKraken. *What is Git Flow.* Accessed on: Jun. 17, 2024. URL: https://www.gitkraken.com/learn/git/git-flow.

[33] Git Hub. *GitHub Flow.* Accessed on: Jun. 17, 2024. URL: https://githubflow.github.io/.

[34] GitLab University. *GitLab Flow - Development workflow in GitLab.* Accessed on: Jun. 17, 2024. URL: https://university.gitlab.com/learn/course/gitlab-flow/gitlab-components-and-workflows/development-workflow-in-gitlab?page=3.

[35] Amazon Web Services, Inc. *SOAP vs REST - Difference Between API Technologies - AWS*. Accessed: Jun. 28, 2024. URL: https://aws.amazon.com/compare/the-difference-between-soap-rest/.

[36] Amazon Web Services, Inc. *What is RESTful API? - RESTful API Explained - AWS*. Accessed: May 27, 2024. 2024. URL: https://aws.amazon.com/what-is/restful-api/.

[37] C. Bayılmış et al. "A survey on communication protocols and performance evaluations for Internet of Things". In: *Digital Communications and Networks* 8.6 (Dec. 2022), pp. 1094–1104. DOI: 10.1016/j.dcan.2022.03.013.

[38] MDN Web Docs. *WebSockets - APIs da Web | MDN*. Online. Accessed: May 27, 2024. URL: https://developer.mozilla.org/pt-BR/docs/Web/API/WebSockets_API.

[39] D. Glaroudis, A. Iossifides - Iosifidis, and P. Chatzimisios. "Survey, Comparison and Research Challenges of IoT Application Protocols for Smart Farming". In: *Computer Networks* (Nov. 2019), p. 107037. DOI: 10.1016/j.comnet.2019.107037.

[40] A. Melnikov and I. Fette. *The WebSocket Protocol*. Request for Comments RFC 6455. Internet Engineering Task Force, Dec. 2011. DOI: 10.17487/RFC6455.

[41] *What is OPC?* Accessed: 02 March 2024. OPC Foundation. URL: https://opcfoundation.org/about/what-is-opc/.

[42] M. H. Schwarz and J. Borcsok. "A survey on OPC and OPC-UA: About the standard, developments and investigations". In: 2013, p. 6. DOI: 10.1109/ICAT.2013.6684065.

[43] OPC Foundation. *Classic*. URL: https://opcfoundation.org/about/opc-technologies/opc-classic/.

[44] OPC Foundation. *Unified Architecture*. Accessed: Jun. 14, 2024. 2024. URL: https://opcfoundation.org/about/opc-technologies/opc-ua/.

[45] *UA Part 3: Address Space Model - 4 AddressSpace concepts*. Accessed: Jun. 14, 2024. URL: https://reference.opcfoundation.org/Core/Part3/v104/docs/4.

[46] *C++ UA Server SDK: OPC UA Node Classes*. Accessed: Jun. 14, 2024. URL: https://documentation.unified-automation.com/uasdkcpp/1.5.2/html/L2UaNodeClasses.html.

[47] Vue.js. *Introduction*. Accessed: May 09, 2024. URL: https://vuejs.org/guide/introduction.html.

[48] MDN Web Docs. *SPA (Single-page application) - MDN Web Docs Glossary: Definitions of Web-related terms | MDN*. Accessed: May 09, 2024. URL: https://developer.mozilla.org/en-US/docs/Glossary/SPA.

[49] Vue.js. *Components Basics*. Accessed: May 09, 2024. URL: https://vuejs.org/guide/essentials/component-basics.html.

[50] Vue.js. *Lifecycle Hooks*. Accessed: May 09, 2024. URL: https://vuejs.org/guide/essentials/lifecycle.html.

[51] Vuetify. *Why Vuetify?* Accessed: May 09, 2024. URL: https://vuetifyjs.com/en/introduction/why-vuetify/#what-is-vuetify3f.

[52] *Node.js — Introduction to Node.js*. May 14, 2024. URL: https://nodejs.org/en/learn/getting-started/introduction-to-nodejs.

[53] Vue.js Core Contributors. *Contributors to vuejs/core*. Accessed: May 09, 2024. URL: https://github.com/vuejs/core/graphs/contributors.

[54] npm. *api.npm*. Accessed: May 09, 2024. URL: https://api.npmjs.org/downloads/point/01-01-2023:31-12-2023/vue.

[55] Django Project. *Django*. May 16, 2024. URL: https://www.djangoproject.com/.

[56] Amazon Web Services, Inc. *What is Django? - Django Framework Explained - AWS*. May 16, 2024. URL: https://aws.amazon.com/what-is/django/.

[57] encode. *encode/django-rest-framework: Web APIs for Django*. May 16, 2024. URL: https://github.com/encode/django-rest-framework.

[58] *Django Channels — Channels 4.1.0 documentation*. May 16, 2024. URL: https://channels.readthedocs.io/en/latest/.

[59] Django Project. *Models | Django documentation*. Accessed: May 21, 2024. URL: https://docs.djangoproject.com/en/5.0/topics/db/models/.

[60] Django Project. *User authentication in Django | Django documentation*. May 16, 2024. URL: https://docs.djangoproject.com/en/5.0/topics/auth/.

[61] Django Project. *Writing your first Django app, part 1 | Django documentation*. May 17, 2024. URL: https://docs.djangoproject.com/en/4.2/intro/tutorial01/.

[62] R. S. Khan. *Creating a Django Project — The Right Way*. Accessed: May 17, 2024. 2024. URL: https://rai-shahnawaz.medium.com/creating-a-django-project-the-right-way-14d230358d72.

[63] *Introduction — ASGI 3.0 documentation*. Accessed: May 20, 2024. 2024. URL: https://asgi.readthedocs.io/en/latest/introduction.html.

[64] *ASGI (Asynchronous Server Gateway Interface) Specification — ASGI 3.0 documentation*. Accessed: May 20, 2024. 2024. URL: https://asgi.readthedocs.io/en/latest/specs/main.html.

[65] Will Vincent. *The Django Jigsaw Puzzle: Aligning All the Pieces with Will Vincent - DjangoCon US 2022*. Online Video. Accessed: May 21, 2024. Nov. 2022. URL: https://www.youtube.com/watch?v=rIt0uj8TaKg.

[66] Django Project. *Middleware | Django documentation*. Online. Accessed: May 22, 2024. URL: https://docs.djangoproject.com/en/5.0/topics/http/middleware/.

[67] Django Project. *Writing views | Django documentation*. Online. Accessed: May 21, 2024. URL: https://docs.djangoproject.com/en/5.0/topics/http/views/.

[68] Django Project. *URL dispatcher | Django documentation*. Online. Accessed: May 21, 2024. URL: https://docs.djangoproject.com/en/5.0/topics/http/urls/.

[69] Django Project. *The Django template language | Django documentation*. Online. Accessed: May 21, 2024. URL: https://docs.djangoproject.com/en/5.0/ref/templates/language/.

[70] *What is Docker? | AWS*. May 14, 2024. Amazon Web Services, Inc. URL: https://aws.amazon.com/docker/.

[71] *Docker Hub Container Image Library | App Containerization*. May 14, 2024. URL: https://hub.docker.com/.

[72] *Overview*. May 14, 2024. URL: https://kubernetes.io/docs/concepts/overview/.

[73] *Containers vs. virtual machines (VMs)*. May 14, 2024. Google Cloud. URL: https://cloud.google.com/discover/containers-vs-vms.

[74] *About WildFly*. Accessed: Jun. 28, 2024. URL: https://www.wildfly.org/about/.

[75] Java Platform, Enterprise Edition. *1.3 Distributed Multitiered Applications - Java Platform, Enterprise Edition: The Java EE Tutorial (Release 7)*. Accessed: Jun. 28, 2024. URL: https://docs.oracle.com/javaee/7/tutorial/overview003.htm.

[76] J. Qu et al. “Digital twins in the minerals industry – a comprehensive review”. In: *Mining Technology* 132.4 (2023), pp. 267–289.

[77] M. Shafto et al. *Modeling, simulation, information technology amp; processing roadmap*. https://www.nasa.gov/sites/default/files/501321main_TA11-ID_rev4_NRC-wTASR.pdf. Available: https://www.nasa.gov/; Accessed on: 2019-09-24.

[78] Y. K. Liu, S. K. Ong, and A. Y. C. Nee. “State-of-the-art survey on digital twin implementations”. In: *Advances in Manufacturing* 10.1 (Mar. 2022). Accessed: Jun. 02, 2024, pp. 1–23. DOI: 10.1007/s40436-021-00375-w. URL: https://link.springer.com/article/10.1007/s40436-021-00375-w?error=cookies_not_supported&code=1cec4d40-646c-4754-a9da-fc2206777520.

[79] F. Tao et al. “Digital twin-driven product design, manufacturing and service with big data”. In: *Int. J. Adv. Manufact. Technol.* 94 (2018), pp. 3563–3576. DOI: 10.1007/s00170-017-0233-1.

[80] C. Cimino, E. Negri, and L. Fumagalli. “Review of digital twin applications in manufacturing”. In: *Computers in Industry* 113 (2019), p. 103130. DOI: 10.1016/j.compind.2019.103130.

[81] D. Mourtzis and K. Vlachou. “A Cloud-Based cyber-physical system for adaptive shop-floor scheduling and condition-based maintenance”. In: *Journal of Manufacturing Systems* 47 (July 2019), pp. 179–198. DOI: 10.1016/j.jmsy.2018.05.008.

[82] R. Rosen et al. “About The Importance of Autonomy and Digital Twins for the Future of Manufacturing”. In: *IFAC-PapersOnLine* 48.3 (2015), pp. 567–572. DOI: 10.1016/j.ifacol.2015.06.141. URL: https://www.ifacoc.org/papers_online/vol48_num3/art0567.

[83] Georgi Martinov, Akram Al Khoury, and Ahed Issa. “Development and Use of OPC UA Tools for Data Collection and Monitoring of Technological Equipment”. In: *Presented at 2023 International Russian Smart Industry Conference (SmartIndustryCon)*. 2023. URL: https://ieeexplore.ieee.org/document/10110757.

[84] F. Yao et al. “Optimizing the Scheduling of Autonomous Guided Vehicle in a Manufacturing Process”. In: *2018 IEEE 16th International Conference on Industrial Informatics (INDIN)*. Accessed on: Feb. 8, 2024. 2018, pp. 537–542. DOI: 10.1109/INDIN.2018.8471979.

[85] Viktor Chekryzhov, Ilya A. Kovalev, and Anton S. Grigoriev. "An Approach to Technological Equipment Performance Information Visualization System Construction Using Augmented Reality Technology". In: *Presented at MATEC Web of Conferences 224, 02093 (2018)*. 2018. DOI: 10.1051/matecconf/201822402093.

[86] C. Liu et al. "A Cyber-Physical Machine Tools Platform using OPC UA and MTConnect". In: *Journal of Manufacturing Systems* 51 (Apr. 2019), pp. 61–74. DOI: 10.1016/j.jmsy.2019.04.006.

[87] F. He, S. K. Ong, and A. Y. C. Nee. "An Integrated Mobile Augmented Reality Digital Twin Monitoring System". In: *Computers* 10.8 (Aug. 2021), p. 99. URL: https://www.mdpi.com/2073-431X/10/8/99.

[88] S. Centomo, N. Dall'Ora, and F. Fummi. "The Design of a Digital-Twin for Predictive Maintenance". In: *Presented at International Conference on Emerging Technologies and Factory Automation (ETFA)*. Sept. 2020. DOI: 10.1109/ETFA46521.2020.9212071.

[89] Weichao Luo et al. "A hybrid predictive maintenance approach for CNC machine tool driven by Digital Twin". In: *Robotics and Computer-Integrated Manufacturing* 65 (2020), p. 101974. ISSN: 0736-5845. DOI: https://doi.org/10.1016/j.rcim.2020.101974. URL: https://www.sciencedirect.com/science/article/pii/S0736584519306660.

[90] S. Răileanu et al. "Integrating the Digital Twin of a Shop Floor Conveyor in the Manufacturing Control System". In: *Service Oriented, Holonic and Multi-agent Manufacturing Systems for Industry of the Future*. Ed. by T. Borangiu et al. Cham: Springer International Publishing, 2020, pp. 134–145. DOI: 10.1007/978-3-030-27477-1_10.

[91] S. Cavalieri, M. G. Salafia, and M. S. Scroppo. "Integrating OPC UA with web technologies to enhance interoperability". In: *Computer Standards & Interfaces* 61 (Jan. 2019), pp. 45–64. DOI: 10.1016/j.csi.2018.04.004.

[92] *OPC UA Forge - Connect IT and OT for Secure Operations*. https://prosysopc.com/products/opc-ua-forge/. Accessed: Jul. 31, 2024.

[93] *OPC UA Forge - Connect IT and OT for Secure Operations*. https://prosysopc.com/products/opc-ua-forge/. Accessed: Jul. 31, 2024.

[94] *DeviceXPlorer OPC Server | ICONICS Software Solutions*. https://iconics.com/en-us/Products/DeviceXPlorer-OPC-Server. Accessed: Aug. 01, 2024.

[95] OPC Foundation. *New DeviceXPlorer OPC Server as DX Platform for Smart Factory – OPC Connect*. https://opcconnect.opcfoundation.org/2022/12/new-devicexplorer-opc-server-as-dx-platform-for-smart-factory/. Accessed: Aug. 01, 2024.

[96] Amazon Web Services, Inc. *AWS IoT Core*. https://aws.amazon.com/pt/iot-core/. Accessed: Aug. 01, 2024.

[97] Microsoft. *IoT Hub | Microsoft Azure*. https://azure.microsoft.com/en-us/products/iot-hub. Accessed: Aug. 01, 2024.

[98] OPC Foundation. *DeviceXPlorer OPC Server Ver.7 will be released in 2022 – OPC Connect*. https://opcconnect.opcfoundation.org/2022/06/devicexplorer-opc-server-ver-7-will-be-released-in-2022/. Accessed: Aug. 01, 2024.

[99] *Node-RED*. https://nodered.org/. Accessed: Aug. 05, 2024.

[100] K. Peffers et al. "A Design Science Research Methodology for Information Systems Research". In: *Journal of Management Information Systems* 24.3 (Dec. 2007), pp. 45–77. DOI: 10.2753/MIS0742-1222240302.

[101] Atlassian. *Scrum Sprints: Everything You Need to Know*. Accessed: Jun. 21, 2024. URL: https://www.atlassian.com/agile/scrum/sprints.

[102] Atlassian. *Kanban vs Scrum*. Accessed: Jun. 21, 2024. URL: https://www.atlassian.com/agile/kanban/kanban-vs-scrum.

[103] Atlassian. *Product Backlog Explained [+ Examples]*. Accessed: Jun. 21, 2024. URL: https://www.atlassian.com/agile/scrum/backlogs.

[104] Atlassian. *Kanplan: where your backlog meets kanban*. Accessed: Jun. 21, 2024. URL: https://www.atlassian.com/agile/kanban/kanplan.

[105] Siemens. *SIMATIC S7-1500 CPUs*. url=https://www.siemens.com/global/en/products/automation/s7-1500/cpus.html. Accessed on: Jan. 31, 2024.

[106] O. Gilles et al. "Securing IIoT Communications Using OPC UA PubSub and Trusted Platform Modules". In: *Journal of Systems Architecture* 134 (2022), p. 102797. DOI: 10.1016/j.sysarc.2022.102797.

[107] A. Morato et al. "Assessment of Different OPC UA Implementations for Industrial IoT-Based Measurement Applications". In: *IEEE Transactions on Instrumentation and Measurement* 70 (2021). DOI: 10.1109/TIM.2020.3043116.

[108] Socket.IO. *Rooms | Socket.IO*. url = https://socket.io/docs/v3/rooms/. Accessd on: Feb 1, 2024.

[109] C. Bayilmis et al. “A Survey on Communication Protocols and Performance Evaluations for Internet of Things”. In: *Digital Communications and Networks* 8 (2022). DOI: 10.1016/j.dcan.2022.03.013.

[110] kgremban, KennedyDMSFT timlt, and v-gpettibone dominicbetts Usha-Rathnaveland paweenatongbai. *What is Azure IoT Hub? - Azure IoT Hub*. url = https://learn.microsoft.com/en-us/azure/iot-hub/iot-concepts-and-iot-hub. (Accessed on: Apr. 26, 2024).

[111] S. Hossain Sani, M. Shopon, and S. Hossen Rakib. “Air Quality Index Prediction Using Azure IoT & Machine Learning for Smart Cities”. In: *Proceedings of the Proceedings of International Conference on Computational Intelligence, Data Science and Cloud Computing*. Accessed on: Feb. 9, 2024. Apr. 2021. DOI: 10.1007/978-981-33-4968-1_56.

[112] D. Mourtzis et al. “Integrated Production and Maintenance Scheduling Through Machine Monitoring and Augmented Reality: An Industry 4.0 Approach”. In: *Presented at Advances in Production Management Systems. The Path to Intelligent, Collaborative and Sustainable Manufacturing*. Aug. 2017. DOI: 10.1007/978-3-319-66923-6_42.

[113] Z. Zhu, C. Liu, and X. Xu. “Visualisation of the Digital Twin Data in Manufacturing by Using Augmented Reality”. In: *Presented at Procedia CIRP*. Vol. 81. Accessed on: Feb. 2, 2024. 2019. DOI: 10.1016/j.procir.2019.03.223.

[114] *Introduction — Channels 4.0.0 documentation*. https://channels.readthedocs.io/en/stable/introduction.html. Accessed: Aug. 27, 2024.

[115] *GitHub - FreeOpcUa/opcua-asyncio: OPC UA library for python >= 3.7*. https://github.com/FreeOpcUa/opcua-asyncio. Accessed: Aug. 27, 2024.

[116] *Channel Layers — Channels 4.0.0 documentation*. https://channels.readthedocs.io/en/stable/topics/channel_layers.html. Accessed: Aug. 27, 2024.

[117] *Docker: Accelerated Container Application Development*. https://www.docker.com/. Accessed: Aug. 30, 2024.

[118] Google Cloud. *Cloud Run*. https://cloud.google.com/run. Accessed: Aug. 30, 2024.

[119] MongoDB. *MongoDB: The Developer Data Platform*. https://www.mongodb.com. Accessed: Aug. 30, 2024.

[120] B. Diène et al. “Data management techniques for Internet of Things”. In: *Mechanical Systems and Signal Processing* 138 (Apr. 2020), p. 106564. DOI: 10.1016/j.ymssp.2019.106564.

[121] *OPC UA Clients - Unified Automation*. https://www.unified-automation.com/downloads/opc-ua-clients.html. Accessed: Sep. 03, 2024.

[122] TicAPP. *TicAPP2021-TestesUsabilidade*. url = https://ticapp.gov.pt/wp-content/uploads/2021/03/TicAPP2021-TestesUsabilidade.pdf. Accessed on: May 2, 2024. May 2021.

[123] P. Laubheimer. *Measuring Perceived Usability*. url = https://www.nngroup.com/articles/measuring-perceived-usability/. Accessed on: May 2, 2024. Feb. 2018.

[124] W. Laubheimer. *Beyond the NPS: Measuring Perceived Usability with the SUS, NASA-TLX, and the Single Ease Question After Tasks and Usability Tests*. url = https://www.nngroup.com/articles/measuring-perceived-usability/. Accessed: May 30, 2024. Nielsen Norman Group, 2018.

[125] Ana Isabel Martins et al. “European Portuguese Validation of the System Usability Scale (SUS)”. In: *Procedia Computer Science* 67.67 (2015), pp. 293–300. DOI: https://doi.org/10.1016/j.procs.2015.09.273.

[126] Aaron Bangor, Phil Kortum, and James Miller. “Determining What Individual SUS Scores Mean: Adding an Adjective Rating Scale”. In: *J. Usability Stud.* 4 (Apr. 2009), pp. 114–123.

[127] John Brooke. “SUS: A quick and dirty usability scale”. In: *Usability Eval. Ind.* 189 (Nov. 1995).

[128] Luciano Floridi. “Digital Time: Latency, Real-time, and the Onlife Experience of Everyday Time”. In: *Philosophy & Technology* 34.3 (Sept. 2021), pp. 407–412. DOI: 10.1007/s13347-021-00472-5.

[129] Kubernetes. *Production-Grade Container Orchestration*. https://kubernetes.io/. Accessed: Sep. 10, 2024. 2024.

DECLARATION

I declare on my honor that my work presented in this project report, with the title *"KNOWLEDGE-DRIVEN PRODUCTION LINES: SCALABLE SYSTEM FOR REAL-TIME MACHINE MONITORING AND CONTROLLING IN INDUSTRIAL ENVIRONMENTS"*, is original and was carried out by Diogo Jesus Gomes Francisco (2220646) under the guidance of the professors: Professor Nuno Carlos Sousa Rodrigues (nunorod@ipleiria.pt), Professor Alexandrino José Marques Gonçalves (alex@ipleiria.pt), Professor Roberto Aguiar Ribeiro (roberto.ribeiro@ipleiria.pt) and Professor Anabela Gonçalves Rodrigues Marto (anabela.marto@ipleiria.pt).

Leiria, September de 2024

Assinado por: **Diogo Jesus Gomes Francisco**
Num. de Identificação: 14938067
Data: 2024.12.23 12:48:15+00'00'

Diogo Jesus Gomes Francisco