WHITE-LABEL SHOP FOR DIGITAL INTELLIGENT ASSISTANCE AND HUMAN-AI COLLABORATION IN MANUFACTURING

Title	D2.1 Joint WASABI demonstrator - version 1
Document Owners	BIBA
Contributors	BIBA, UNIMORE, SYXIS, ICCS, ATLANTIS
Dissemination	Public
Date	29/11/2023
Version	V0.7

Co-funded by the Horizon Europe programme
of the European Union under Grant Agreement
№ 101092176

VERSION HISTORY

Nr	Date	Author (Organisation)	Description
0.1	05/10/2023	Mina Foosherian (BIBA)	Deliverable structure
0.2	23/10/2023	Mina Foosherian (BIBA)	Updated table of content
0.3	09/11/2023	Mina Foosherian (BIBA) Stefan Wellsandt (BIBA) Indah Lengkong (BIBA) Dena Arabsolgar (SYXIS) Ioakeim Fotoglou (SYXIS) Silvia Ghidini (SYXIS) Federica Mandreoli (UNIMORE) Umair Haider (UNIMORE) Alireza Rahimi (UNIMORE) Enrico Taglini (REINOVA) Nicola Becchelli (REINOVA) Maria Papaspyropoulou (ATLANTIS) Zoi Chatzichristodoulou (ATLANTIS) Mattheos Fikardos (ICCS) Katerina Lepenioti (ICCS) Alexandros Bousdekis (ICCS)	Input from all partners
0.4	13/11/2023	Mina Foosherian (BIBA)	Comments from BIBA for all partners
0.5	16/11/2023	Mina Foosherian (BIBA) Ioakeim Fotoglou (SYXIS) Federica Mandreoli (UNIMORE) Umair Haider (UNIMORE)	Input from partners
0.6	20/11/2023	Alexandros Bousdekis (ICCS) Bernhard Lutzer (TTTECH)	Comments from ICCS and TTTECH for all partners
0.7	29/11/2023	Mina Foosherian (BIBA) Bernhard Lutzer (TTTECH) Ioakeim Fotoglou (SYXIS) Federica Mandreoli (UNIMORE) Zoi Chatzichristodoulou (ATLANTIS) Alexandros Bousdekis (ICCS)	Final input from partners

REVIEWERS

Name	Organisation
Alexandros Bousdekis	ICCS
Bernhard Lutzer	TTTECH

DISCLAIMER

ACKNOWLEDGEMENT

This document is a deliverable of the WASABI project. This project has received funding from the European Union's Horizon Europe programme under grant agreement N° 101092176

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviation	Description
SME	small and medium-sized enterprises
DIA	Digital intelligent assistant

1. EXECUTIVE SUMMARY

WASABI focuses on intelligent digital assistant (DIA) solutions to help humans achieve their goals without marginalizing them - this will contribute to human-centered manufacturing. The WASABI vision is that DIA and conversational AI become standard practices to reach sustainability goals in manufacturing. Humans will use it, for instance, to identify and assess opportunities to turn waste into a resource and to reorganize work to minimize carbon footprints. Access to these benefits will be as simple as selecting and configuring Apps from an online store, and interoperability minimizes vendor lock-in and maximizes information valorization. New AI-focused training services for employees will be a general practice, too. They let workers experience solutions and teach them about AI's capabilities, risks, and limitations in manufacturing. DIAs will blend into Europe's emerging legal framework for AI, and they will be affordable and manageable even for small producers. The WASABI project aims to demonstrate DIA solutions for SMEs and mid-caps to help them achieve their sustainable goals and faster onboarding of new workers.

The WASABI project is structured around five use case partners: CROMA (sterilization of surgical equipment), EPISCAN (production of personal protective equipment), REINOVA (testing and validation of e-mobility components), SILK-BIO (solubilisation and casting of silk fiber) and TRIMEK (metrology systems, solutions, and machines). Some of these partners are involved in more than one use case.

This document presents Deliverable 2.1, "Joint WASABI Demonstrator – Version 1", which demonstrates the results of the first phase of the project (M1-M9) in the development and deployment of the WASABI solution across three distinct use cases:

1. Augmented waste management and valorization for TRIMEK and CROMA (Task 2.2)
2. Assisted workforce management for EPISCAN (Task 2.3)
3. Assisted quality assurance for sustainable products for REINOVA, TRIMEK, CROMA, EPISCAN, and SILKBIO (Task 2.4)

This deliverable reports the main activities and results carried out from M1 to M9 in Tasks 2.1, 2.2, 2.3, and 2.4.

Task 2.1, "WASABI's core infrastructure for digital assistance" aims to set up the WASABI core infrastructure as a Docker environment, mainly based on the results from the Horizon 2020 RIA COALA project[1]. This core infrastructure has been deployed in three demonstrators developed within T2.2, T2.3, and T2.4. This task has ended, and the progress of this task is reported in Section 3 of this deliverable.

In the first phase of Task 2.2, "Augmented waste management and valorisation", the first demo prototype for the waste inspector assistant for TRIMEK and CROMA have been deployed and configured using the COALA stack in a Docker environment running on a server in BIBA. A connection between the Waste Inspector skills and the rEUse waste management platform has been developed. An introduction to the rEUse waste management platform and a demonstration of the Waste Inspector skills and their initial dialog model in English are presented in Section 4 of this deliverable.

In the first phase of Task 2.3, "Assisted workforce management", the first demo prototype for EPISCAN onboarding assistant has been deployed and configured using the COALA stack in a Docker environment running on a server in BIBA. Technical components, as well as a demonstration of the Onboarding Assistant skill and its initial dialog

[1] https://cordis.europa.eu/project/id/957296

model for the onboarding processes and the knowledge base in English and Spanish are presented in Section 5 of this deliverable.

Finally, in the first phase of Task 2.4, "Assisted quality assurance for sustainable products", a digital assistant framework for product quality testing was defined. The first prototype of the status monitoring skill for REINOVA was developed based on this framework by deploying and configuring REINOVA's existing digital twin for product testing. This skill is based on the COALA base stack in a Docker environment running on a server in UNIMORE, uses REST APIs to access sensor data, and has an Analytics module to provide insights about quality testing procedures. By applying the defined digital assistant framework at REINOVA, an evidence of its feasibility to other business cases in provided. Additionally, an overview of how the design principles adopted for the REINOVA prototype can be adapted and extended to support the other involved business cases, e.g., TRIMEK, COMA, EPISCAN, and SILKBIO are presented in Section 6 of this deliverable.

Tasks 2.2., 2.3, and 2.4 are scheduled to continue till M24 with the second and third phases. During the second phase, our focus will be on enhancing the capabilities of the assistants by close collaboration with the end users and incorporating feedback obtained from users during evaluations. Additionally, findings from the first round of the open-call experiments will be valuable for this phase. The progress and achievements of the second phase will be detailed in D2.4 in M15. In the third and final phase, we will integrate the developed solutions into the end users' infrastructure and finalize the demonstrations. The result of the final phase will be detailed in D2.6 in M24.

2. INTRODUCTION

2.1 Purpose and scope of the deliverable

The WASABI project centers on advancing Digital Intelligent Assistant (DIA) solutions to facilitate goal achievement for humans without marginalization, aligning with the principles of human-centered manufacturing and sustainability. The vision involves integrating DIAs and conversational AI as standard practices to address sustainability goals in manufacturing. The deliverable 2.1 "Joint WASABI Demonstrator – Version 1", presents outcomes from the first phase (M1-M9) of the project across three distinct use cases.

- Use case 1: Augmented waste management and valorisation for TRIMEK and CROMA (Task 2.2)
- Use case 2: Assisted workforce management for EPISCAN (Task 2.3)
- Use case 3: Assisted quality assurance for sustainable products for REINOVA, TRIMEK, CROMA, EPISCAN, and SILKBIO (Task 2.4)

This deliverable reports the main activities and results carried out during M1 to M9 across Tasks 2.1, 2.2, 2.3, and 2.4. This deliverable is the first of three Joint WASABI Demonstrators; version 2 is scheduled for publication in M15, and the final version set to be released in M24.

2.2 Relation to other WPs and Tasks

This deliverable receives input from WP1 (T1.1, T1.2, T1.3, T1.4), WP3 (T3.4), WP4 (T4.2, T4.4), and WP5 (T5.5). Figure 1 shows how D2.1 relates to other WPs and Tasks. There are minor relations to the data management plan (D7.2 and D7.5) and the periodic progress reports (D7.3 and D7.4) because they contain information about privacy.

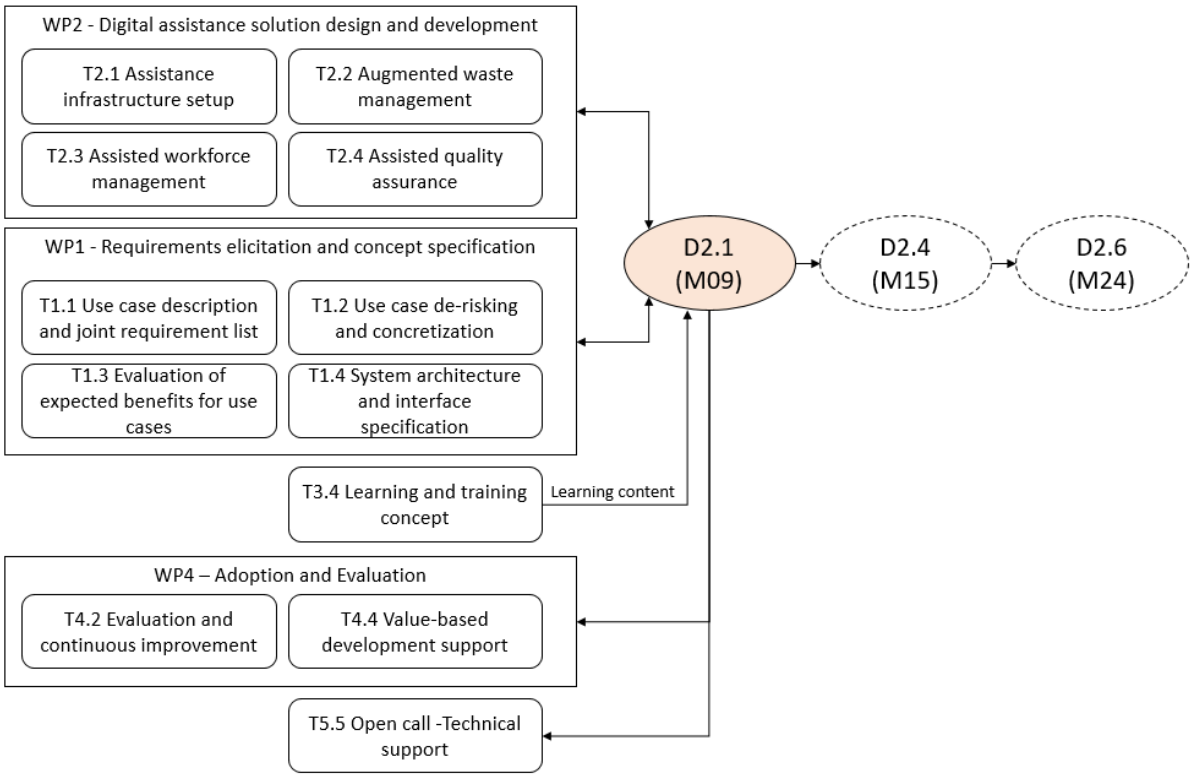

Figure 1: D2.1 relation to other work packages and tasks

3. ASSISTANCE INFRASTRUCTURE SETUP

The Horizon 2020 RIA COALA developed and tested a basic setup for a voice-enabled digital intelligent assistant. This setup consists of several Docker containers running open-source software as micro services. This document uses the term "COALA stack" for it. WASABI modified and updated the COALA stack in several aspects to make it more maintainable and faster to deploy. The technical description of the stack is confidential and described in D1.3 "System architecture and interface specification". The following list summarizes the main changes compared to the COALA stack.

- Replaced the Rasa X Docker stack (deprecated) with a simplified Rasa setup (Rasa server, action server, model server, tracker store). The model server can receive models from a continuous machine-learning process.
- COALA's anonymization and user services were removed in favor of a KeyCloak-only solution.
- Updated all service versions, especially KeyCloak.
- Started migration from Mycroft (discontinued) to OpenVoiceOS (Mycroft fork).
- Created a Docker Compose project with detailed deployment instructions.
- Updated the port exposure to minimize public ports (only Nginx's 80 and 443 ports are public).
- Added HiveMind client to simplify debugging
- Improved documentation for the COALA Android App
- Replaced Graphene GraphQL server with Apollo GraphQL server.

Figure 2 illustrates the main elements of the WASABI assistance infrastructure. Most components use open-source software as a basis and a proprietary configuration. Many parts of the configuration are specific for each application case.

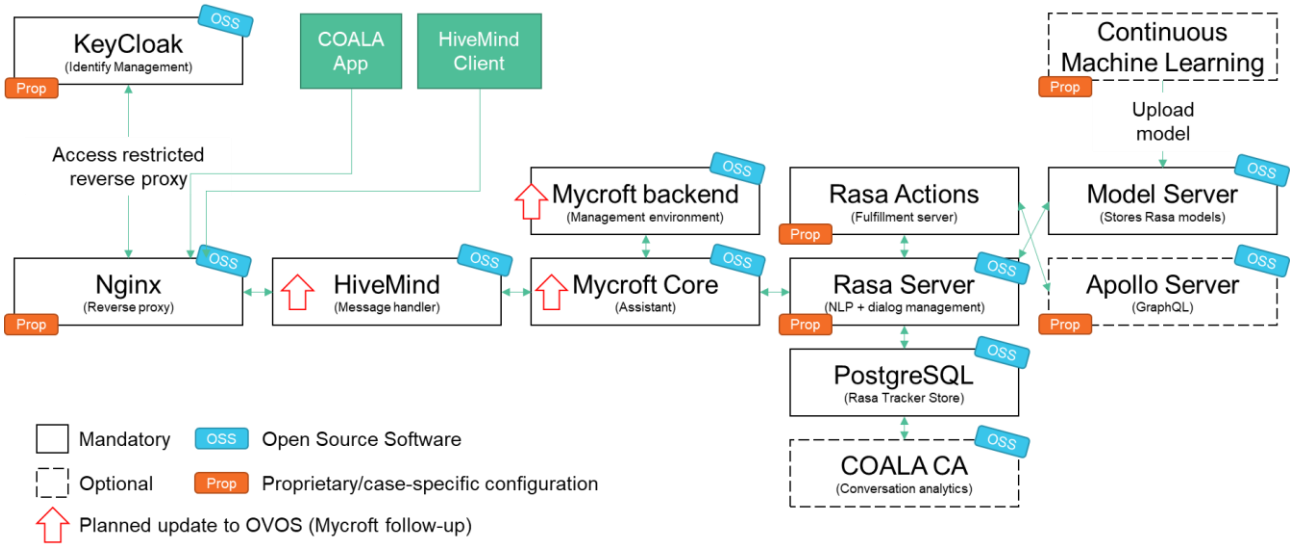

Figure 2: WASABI Assistance Infrastructure

WASABI's assistance infrastructure will be deployed (installed) in three different ways. The first deployment is on a **local computer** for testing and debugging. This setup may publish more container ports to grant developers access to development tools like a GraphQL query browser.

The second deployment can be on an **edge computer**, i.e., a computer close to manufacturing. An edge deployment can be beneficial when a use case requires a local, stand-alone island-system. In this case, the deployment will likely have no or highly constrained access to external resources through the Internet. The edge

computer's hardware is predefined. The partner TTTECH will deploy assistants on edge computers in WASABI using their commercial platform Nerve.[2]

The third deployment is on a (virtual) computer provided by a **cloud service provider**. This deployment can benefit from the flexible adjustment of infrastructure (e.g., CPU, GPU, and RAM) and software scalability (e.g., replicas of a service). A downside is that the costs are hard to anticipate because they likely depend on the actual resource usage. The latter remains largely unknown for prototypes. Taking full advantage of cloud functionality requires additional configuration information (e.g., how many replicas per service) and a definition of additional components like a load balancer.

In order to deploy the containers using Nerve, the whole stack shall be defined using Docker Compose. **Docker Compose** is a tool for defining and running multi-container Docker applications. It uses a YAML file, which serves as a configuration file where you define all the services, networks, and volumes for your Docker application, containing the following parts:

1. **version**: Specifies the version of the Docker Compose file format. This determines which features will be available for use.
2. **services**: This is where you define the different containers you want to run as part of your application. Each service can represent a microservice or a module of your application. For each service, you can specify:
    - **image**: The Docker image to use for the container.
    - **restart**: The policy for restarting the container, for instance, **always** to ensure it restarts if it stops.
    - **environment**: Environment variables for the container, which can be hardcoded or taken from the host environment.
    - **ports**: The ports that need to be exposed, mapping from host to container.
    - **volumes**: Mounts paths from the host to the container, allowing for data persistence or sharing data between the host and container.
    - **depends_on**: Specifies dependencies between services, ensuring they start in the correct order.
3. **networks**: Defines the networks your containers will use to communicate with each other. Containers on the same network can talk to each other using container names.
4. **volumes**: Declares volumes that services can use for persistent storage independent of the container's lifecycle.
5. **configs** and **secrets**: For managing sensitive information and configuration files that should not be sent to every image user.

Nerve only supports Docker compose files validated against the 3.5 schema specification (Compose file version 3 reference | Docker Docs). Figure 3 illustrates the activities to run a stack from a Compose file.

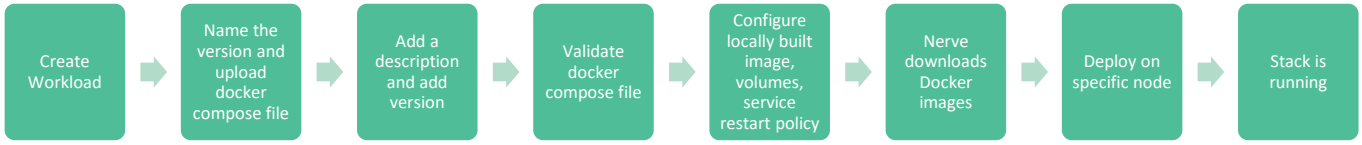

Figure 3: Activities to run a Compose file with the Nerve platform

Figure 4 illustrates the workload view from the Nerve management platform.

[2] www.tttech-industrial.com/nerve

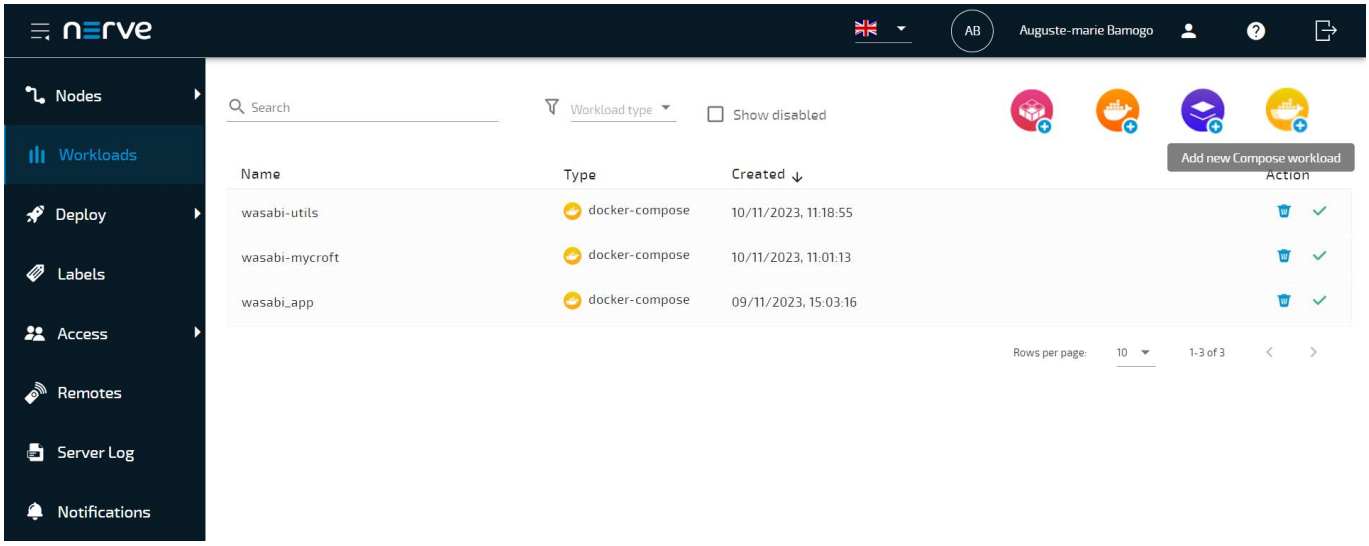

Figure 4: Workload view of the Nerve management platform

Figure 5 illustrates what a workload looks like for the user after deployment.

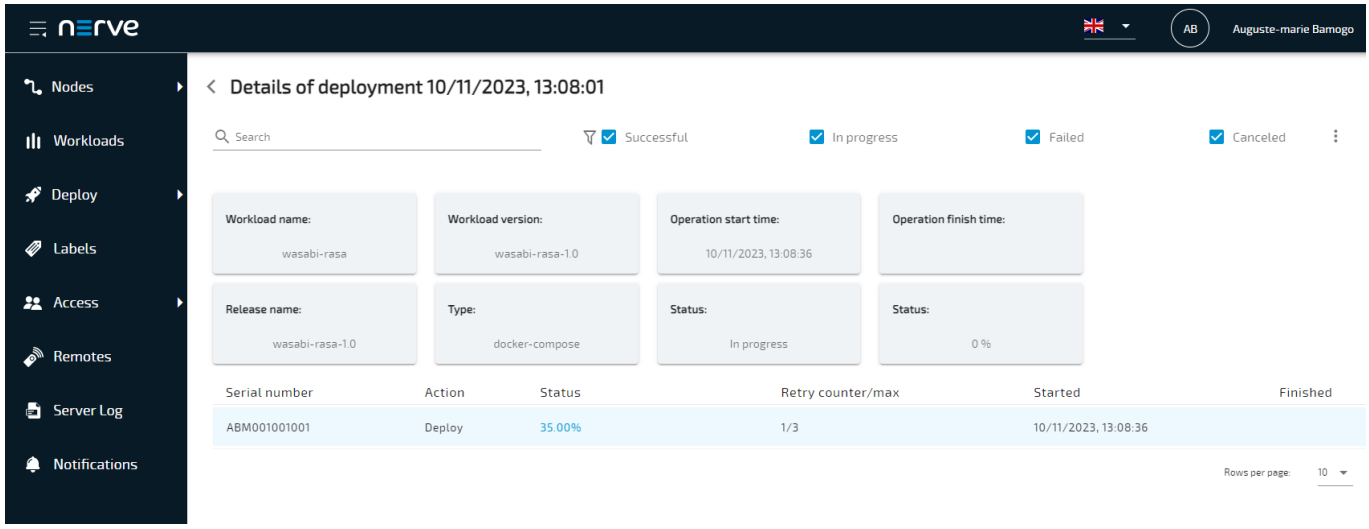

Figure 5: Deployed workload view of Nerve management platform

4. AUGMENTED WASTE MANAGEMENT AND VALORISATION

This use case concerns the better utilization and management of dismissed items in manufacturing to minimize those found to be suboptimal based on their specific manufacturing or tolerance criteria. The central assumption is that a production process cannot avoid generating some non-conforming items for technical reasons. These items may still be reusable or refurbished by first-parties or third-party organizations, provided the characteristics can meet the appropriate requirements.

This task provides workers with advanced ways to help describe dismissed items and provides third parties with ways to identify if they qualify for reuse. Workers can use a combination of the powerful tools provided by the rEUse platform to describe these dismissed items, have them stored in the rEUse environment, and enrich the description with specific descriptive attributes to facilitate the reuse process. Consequently, the WASABI's voice assistant can interact with rEUse to enrich and draw upon this information or rapidly gain insights on the items. rEUse has REST endpoints to facilitate automatic responses to queries received from the WASABI ecosystem, which can be used to automate parts of this process.

In the sections below, we examine the assistant approach and how rEUse is being adapted to the use cases of CROMA and TRIMEK. The latter includes defining reference attributes and circular entities, which we implemented in cooperation with them for their purposes.

In deploying the rEUse part of the waste management solution described in the deliverable, we communicated with the partners from TRIMEK and CROMA to determine their needs in compiling the list of appropriate circular entities and properties modeled in the rEUse platform. In achieving this, workshops presenting the processes followed by the two partners were also held in the summer of 2023 and were invaluable in demonstrating the industrial approaches of the partners. These discussions culminated in the understanding that neither company deals with actual waste but more with items that need to be recycled, repurposed, or otherwise reintroduced for other purposes; hence, we adjusted the waste management part to encompass a circular approach, although, at the time of the writing of this deliverable, we are still referring to it as waste management. Future versions of this deliverable will adopt the new terminology.

4.1 Components

4.1.1 rEUSe Waste Management Platform

The technical object used to create the Waste Management Platform is based on the data modeler tool called "rEUse". It is a Web application where users can define data entity models using the Topic and Attribute meta-concepts.

An **entity definition** (name and attributes) is captured by an instance of the Topic meta-concept and instances of the Attribute meta-concept. For example, the Topic can be Calibrated Artifacts, while Attributes are properties of Calibrated Artifacts, e.g., Calibration Laboratory, Item, and Mark. rEUse provides a user interface to list, add, and update Topic and Attribute instances and to manage relationships between Topic and Attribute instances. Besides, it offers front-end elements for creating and removing actual data entities and list-based and single-entity renderers. D1.3 details the technical aspects and the deployment mechanism of rEUse.

4.1.2 Rasa Chatbot

Rasa is an open-source chatbot framework using a configurable NLU pipeline, a dialog manager based on rules and probabilistic models, and a fulfillment server performing custom actions via Python code. The action server

queries information from the rEUse platform through a REST API and uses the results to decide how to respond (e.g., using a template with parameters filled by the query result).

Our implementation of the Rasa chatbot has two groups of features:

- Base Features: These encompass universally applicable intents and conversation patterns suitable for generic queries and tasks.
- Circular entity traceability features: Tailored to specific use case circular entities, these features are designed to allow users to interact with the rEUse platform by creating new entities and retrieving information about already registered entities.

4.2 Demonstration scenarios

The demonstration focuses on TRIMEK and CROMA. Each case has its entity definition and a specific set of attributes. We developed the initial definitions using demonstrative sessions showcasing the abilities and functionalities of rEUse and an early assistant mockup. The following definitions will likely evolve with further progress in WP2.

4.2.1 Waste inspector skill demo for TRIMEK

4.2.1.1 Entity model and basic interaction with rEUse

Table 1 summarizes the schema of properties communicated by TRIMEK. This information provides the basis for how rEUse represents dismissed items in this case.

Table 1: Fields and types in rEUse for the TRIMEK use case

Field	Type in rEUse
Calibration laboratory	Alphanumerical text – Location
Item	Alphanumerical text – Name
Mark	Alphanumerical text
Model	Alphanumerical text
Serial number	Alphanumerical text
Code No.	Alphanumerical text
Manufacturer	Alphanumerical text
Identification	Alphanumerical text
Applicant	Alphanumerical text – Name
Date/s of test	Alphanumerical text **with** dates and **not** date: according to input from TRIMEK there can be multiple dates if an artifact is calibrated multiple times throughout its usable lifetime
Laboratory technician	Alphanumerical text
Date of issue	Date – Selectable field
Name of the measurement equipment	Alphanumerical text
Code of the measurement equipment	Alphanumerical text
Measurement field (mm)	Numerical field, dimensions
Calibration procedure	Alphanumerical text – description
Temperature	Numerical field, float number
RH	Numerical field, percentage
Measuring range	Alphanumerical text
Hole interval	Numerical field, float number

Number of holes	Numerical field, non-negative integer
Linear expansion coefficient	Numerical field, float
Uncertainty	Numerical field, float
Link to the calibration certificate (pdf, word)	Alphanumerical text, direct link to document stored in TRIMEKs infrastructure

These properties were then adjusted and implemented into the "Calibrated Artifacts" circular entity in rEUse and attached to the calibrated artifacts topic. Figure 6 illustrates the properties from the rEUse attributes management page.

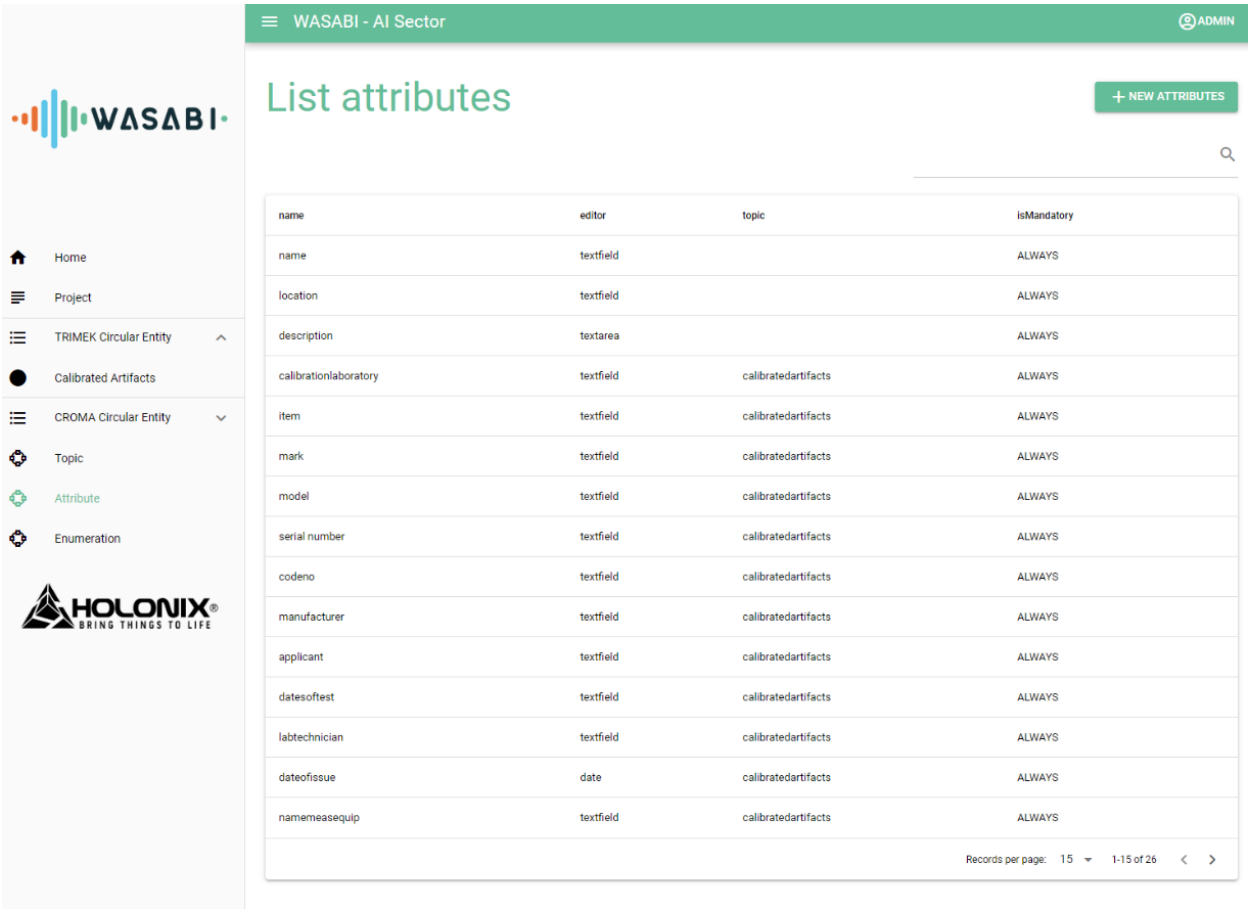

Figure 6: rEUse admin menu GUI with a list of attributes

The fields demanding alphanumerical and numerical inputs were assigned a "textfield" value in the editor, whereas "dateofissue" was assigned a "date" value. Adding the circular entity properties creates the appropriate fields and enables TRIMEK to populate the Calibrated Artifacts circular entity.

Once the administrator created the model, users and other software (like the WASABI assistant) can populate the environment with data entries. Figure 7**Fehler! Verweisquelle konnte nicht gefunden werden.** presents the page where a user can fill out the form to record a circular entity "Calibrated Artifact".

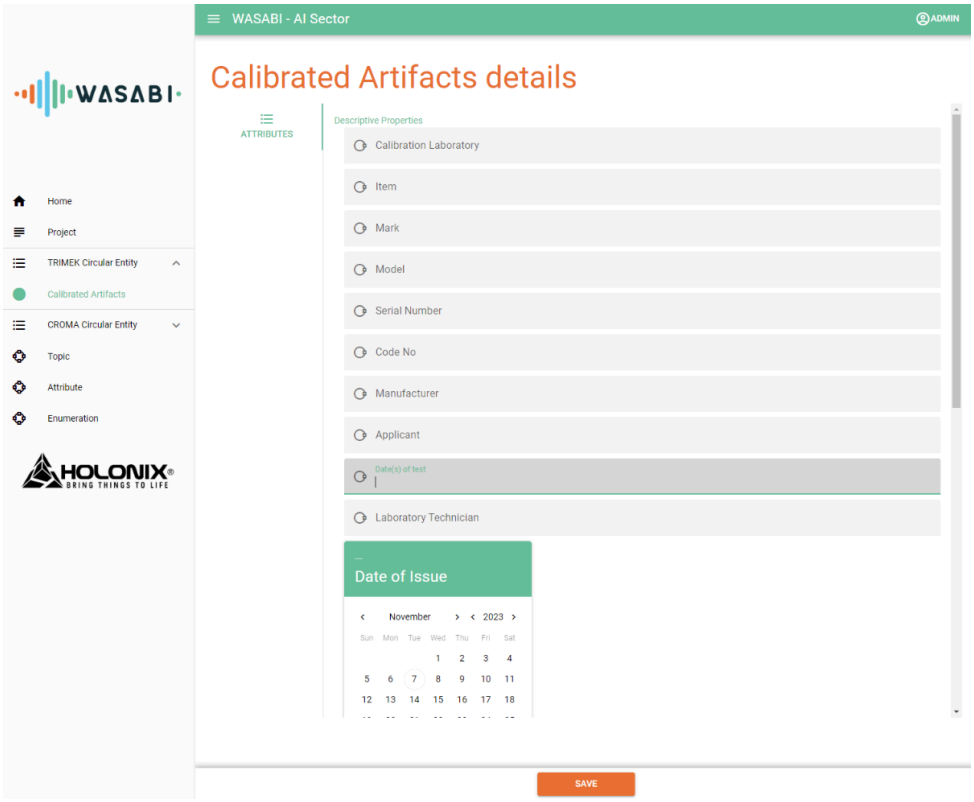

Figure 7: Create a Calibrated Artifact entity in rEUse

Once the user submits the form, rEUse records the artifact. Figure 8**Fehler! Verweisquelle konnte nicht gefunden werden.** shows the list of created artifacts.

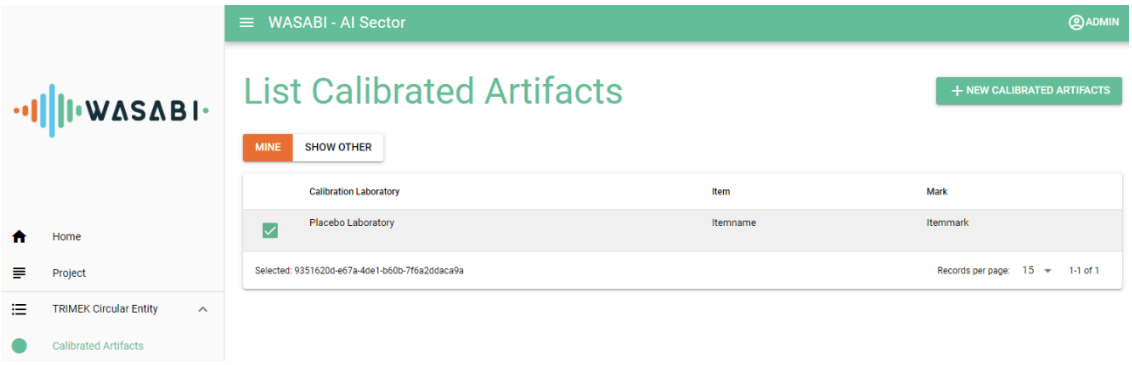

Figure 8: List of calibrated artifacts

Each record receives a universally unique identifier (UUID) indicated below the selected item. This identifier is used for all internal processes and interactions with the REST API. The backend of rEUse is flexible enough to handle thousands of items and circular entities, which allows for scale if needed.

4.2.1.2 Skill interaction scenario

The actual population of the rEUse platform follows the idea of a **multiexperience** (sic) interaction, where users can select the most appropriate interface for their task and situation. The WASABI assistant will help users fill out the form, but when users must interact with many entries quickly, they may prefer using rEUse's graphical interface.

The waste inspector skill initial prototype for TRIMEK has been developed based on the COALA assistant configuration. It runs in a Docker environment on a server maintained by BIBA (detailed in D1.3). Examples of

dialog models for the waste inspector skill in English are developed based on the required attributes to describe calibrated artifacts entered into the rEUse platform.

This section outlines the first demonstration scenario for the TRIMEK waste inspector skill. In this demo, we focus on an introductory scenario to collect the required attributes from the user. Figure 9 outlines an actual conversation where the assistant asks one question per attribute and collects the user's answer.

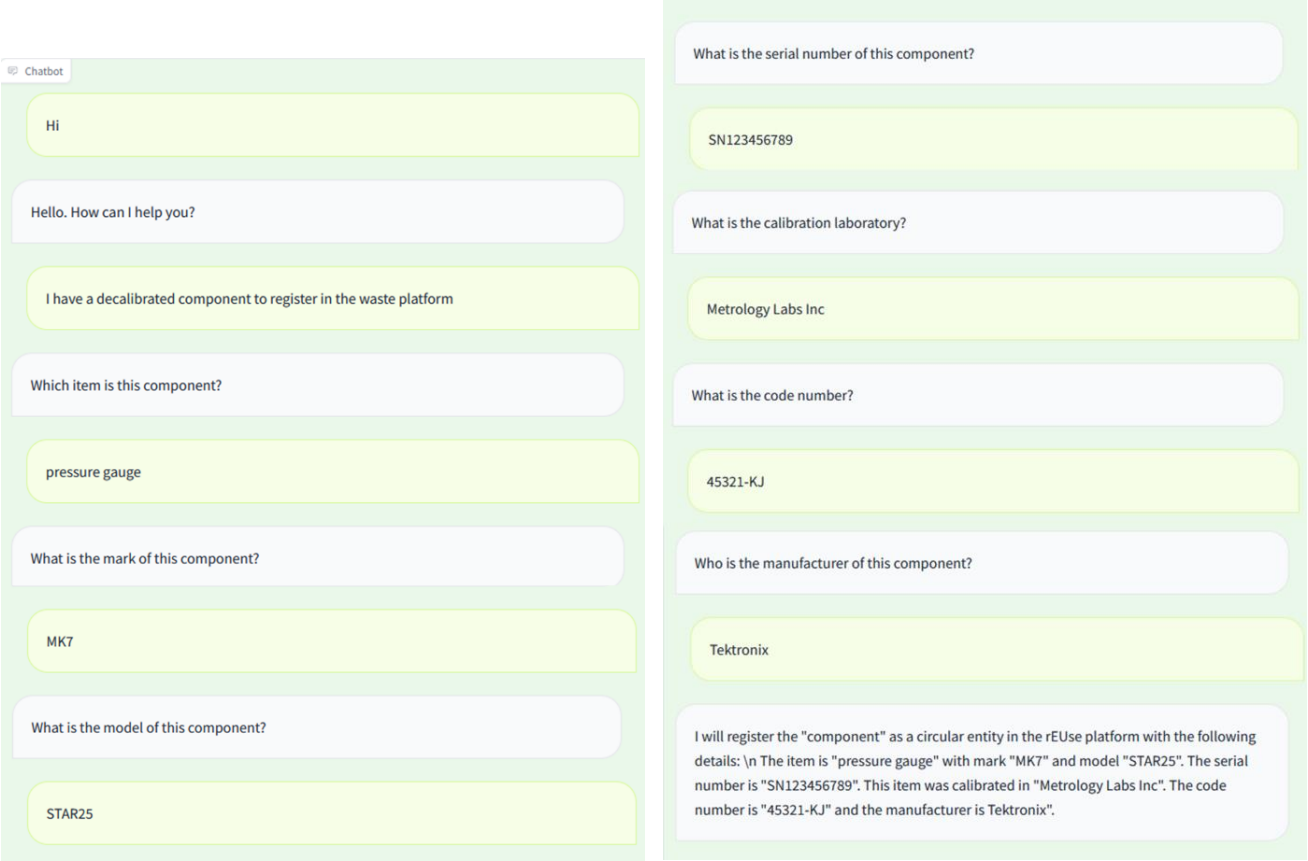

Figure 9: Example of registering a calibrated artifact

The scenario above includes the intents and named entities summarized in Table 4. The **intent** is what the user intends to do with the assistant (e.g., describe waste). **Named entities** are exceptional information the assistant has to recognize in the user utterances. Rasa stores their values in variables to access them later in the conversation.

Table 2: Intents and named entities used for TRIMEK

Intent name	Named entities	Description	Example utterance
describe_waste	waste_mark waste_model waste_serial_number waste_calibration_laboratory waste_code_number waste_manufacturer waste_identification waste_applicant waste_date_of_test waste_laboratory_technician	This intent aims to recognize and prompt the required entities for the rEUse platform from the user.	I have a decalibrated component to register in the waste platform I want to register my waste in rEUse platform

	waste_date_of_issue waste_name_of_measurements_equipment waste_code_of_the_measurement_equipment waste_measurement_field_in_mm waste_calibration_procedure waste_temperature waste_RH waste_measuring_range waste_hole_interval waste_number_of_holes waste_linear_expansion_coefficient waste_uncertainty waste_link_to_the_calibration_certificate		Please check if this waste is valuable

4.2.2 Waste inspector skill demo for CROMA

4.2.2.1 Entity model and basic interaction with rEUse

We have received feedback and are working to determine the numbers and types of circular entities and what fields they will encompass. CROMA has provided sample documents demonstrating what they would like to achieve. Table 3 summarizes the entities for CROMA.

Table 3: Fields and types in rEUse for the CROMA use case

Entity	Fields	Type in rEUse
SET	• Set name • Code • Number • List of instruments (multiple)	• Alphanumerical text – Name • Alphanumerical text • Number, non-negative integer Linked to the "INSTRUMENT" entity
INSTRUMENT	• Type of instrument • Instrument name • Code • Estimated number • Real number • Type of actions	• Alphanumerical text – Name • Alphanumerical text • Alphanumerical text • Number, non-negative integer • Number, non-negative integer • Selectable List (content of list needs to be defined; communication with CROMA still open)

This context highlights the flexibility of adopting circular entities in our approach, demonstrating how two different circular entities can be interlinked in their purpose. In this case, the SET entity comprises items inherent in a separate entity (INSTRUMENT) with their own attributes and yet part of the attributes of the former. As of this time, we are still investigating the required attributes to determine the final objects that will be used to populate the platform, so screenshots from the rEUse platform are preliminary and may not conform to the final version that will be operable after CROMA provides their final feedback on the selection of attributes for the circular entities on rEUse.

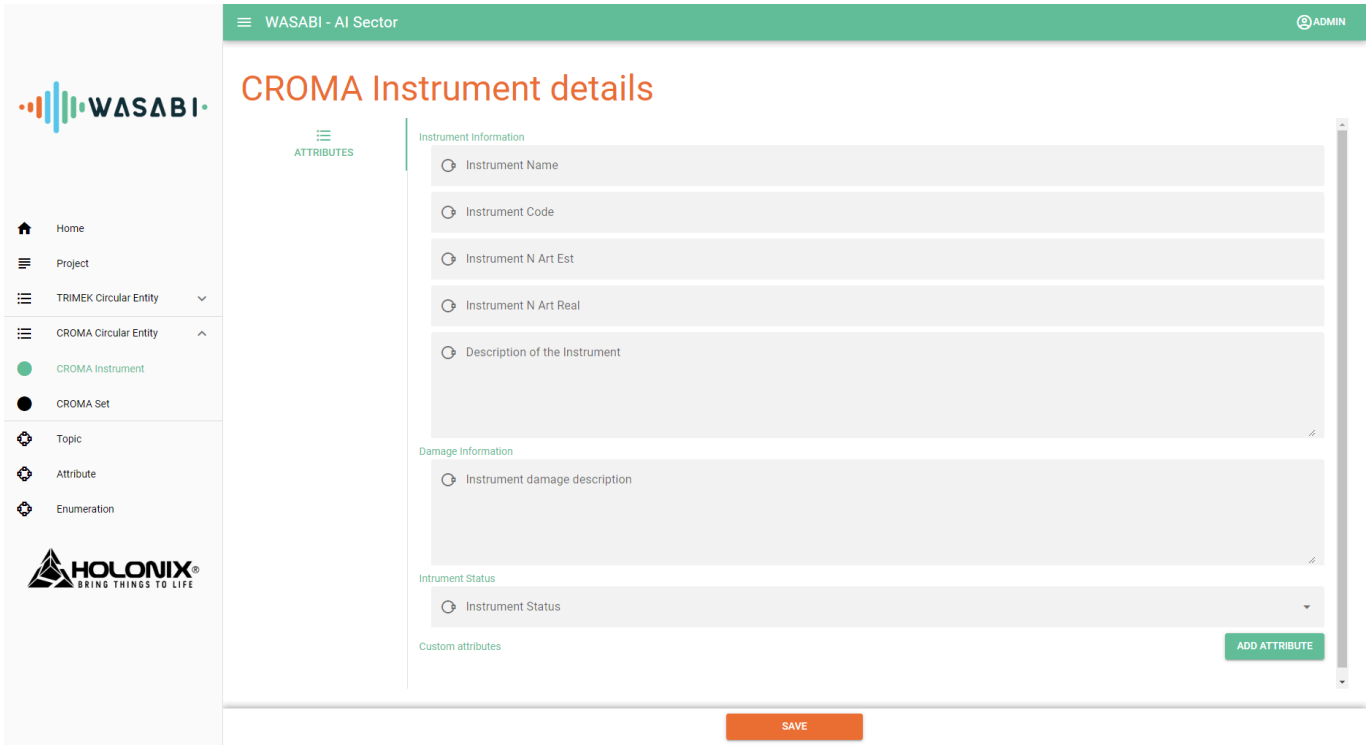

Figure 10: An initial conception of attributes in the CROMA use case

4.2.2.2 Skill interaction scenario

The initial CROMA waste inspector skill prototype has been developed based on the COALA assistant configuration in a Docker environment running on a server in BIBA (detailed in D1.3). Examples of dialog models for the waste inspector skill in English are developed based on the required attributes to describe sets of surgical instruments or the instruments to be entered in the rEUse platform.

This section outlines the first demonstration scenario for the CROMA waste inspector skill. In this first demo, we focused on an introductory scenario to record the required attributes from the user and implemented key functionalities for it.

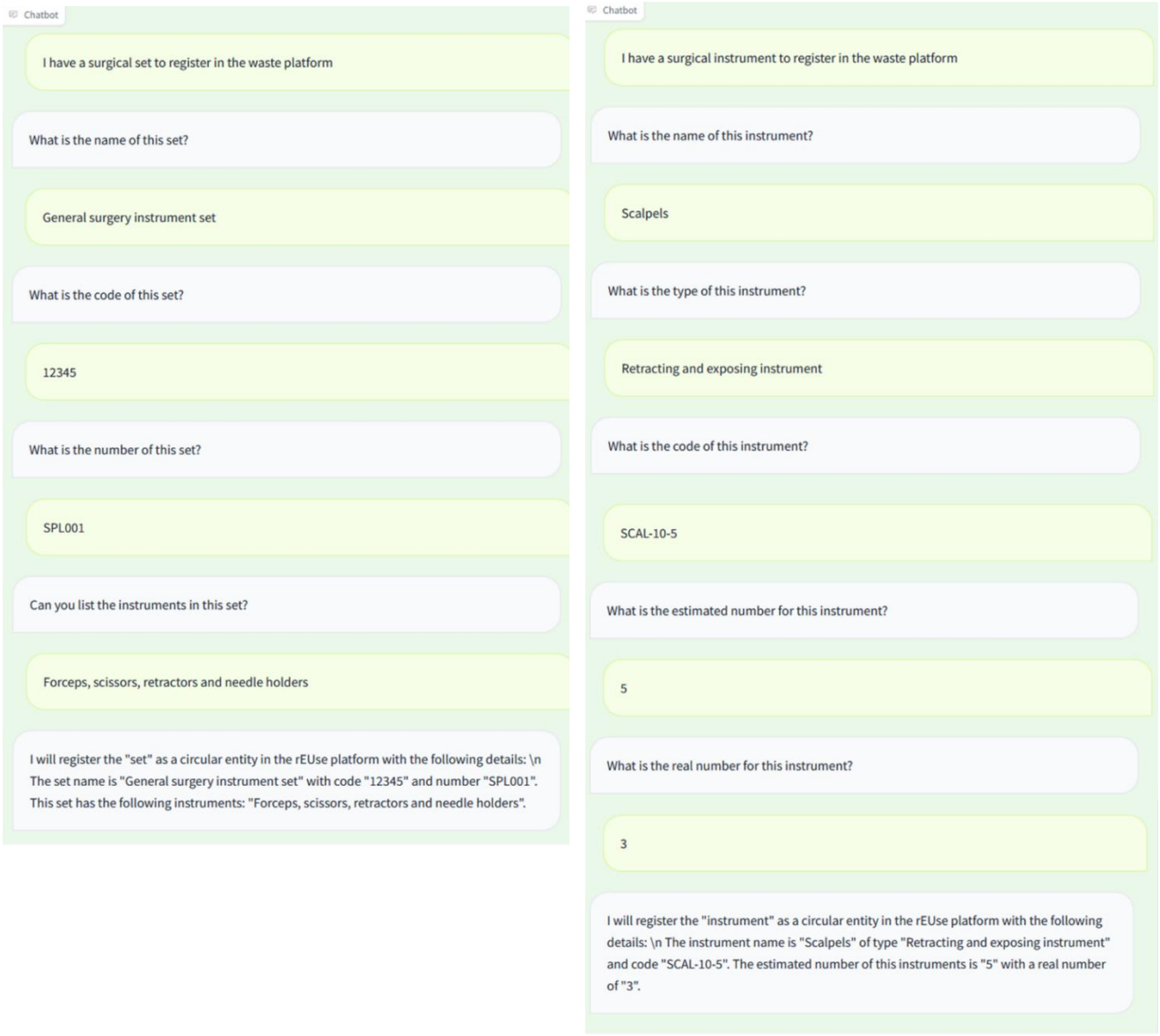

Figure 11: Example of registering a surgical set (left) and a surgical instrument (right)

The scenario above includes the intents and named entities summarized in Table 4. The intent name is identical to the TRIMEK case, but the named entities differ.

Table 4: Intents and named entities used for CROMA

Intent name	Named entities	Description	Example utterance
describe_waste	waste_set_name waste_set_code waste_set_number waste_set_list_of_instruments waste_instrument_name waste_instrument_type waste_instrument_code waste_instrument_estimated_number waste_instrument_real_number	This intent aims to recognize and prompt the required entities for the rEUse platform from the user.	I have a set of surgical instruments to register in the waste platform I want to register my waste in rEUse platform Please check if this waste is valuable

5. ASSISTED WORKFORCE MANAGEMENT

In the assisted workforce management use case, the WASABI solution aims to help workers follow the manufacturing process and give arrival technical instructions to new employees. This will reduce training times for workers in manufacturing processes. EPISCAN, a Canarian-based company that produces personal protective equipment (PPE), is the involved partner in this use case. The main products produced by EPISCAN are masks (surgical and FFP2 without exhalation valves). These are produced in two separate production lines at EPISCAN's facilities. Masks are produced in very high volumes. In the surgical mask production line, a total of 35,000 masks can be produced in an 8-hour working day. In the FFP2 mask production line, a total of 15,000 masks can be produced in an 8-hour working day.

The initial prototype for Assisted Workforce management for EPISCAN has been developed based on the COALA assistant configuration in a Docker environment running on a server in BIBA (detailed in D1.3). Examples of dialog models for the onboarding process and the knowledge base in English and Spanish are developed based on fifteen user stories (described in D1.2) and the instructions for five training processes from the machine manuals.

This section outlines the skill components and the demonstration scenarios for the EPISCAN onboarding assistant.

5.1 Components

This section explains the components used for the Assisted Workforce Management skill, including the Neo4j database, Apollo server and GraphQL, and the Rasa chatbot.

5.1.1 Neo4j

Neo4j is a graph database management system that provides an efficient way to store, manage, and query data in terms of entities and the relationships between them. Unlike traditional relational databases, where data is stored in structured tables, Neo4j uses nodes, relationships, and properties to represent and store data. This graph structure allows for flexible and complex queries to be executed with efficiency, making it a popular choice for applications that require complex data relationships.

Our Neo4j database is designed for the EPISCAN onboarding process of new employees. It is structured to guide and monitor the training process of employees, ensuring they are familiarised with various machine operations. Below are the capabilities of the database:

- **Dynamic Training Paths**: The database offers a structured training pathway, guiding technicians through various machine operations in a logical sequence.
- **User Progress Monitoring**: By tracking the active training stages of users, the database allows for the real-time monitoring of individual progress.
- **Multilingual Support**: Training processes and steps are documented in both English and Spanish.
- **Media Integration**: Various training steps are complemented with associated images, videos, or documents to enhance comprehension.

Nodes and their attributes: In our graph database, the primary entities are represented as nodes. Each node belongs to a specific type, and each type has its own set of attributes that provide detailed information about that entity. For the purpose of this project and its initial development stages, we have introduced hypothetical users

and media files. This allows us to simulate real-world scenarios and interactions, even though we do not have actual user details at the moment. The table below summarises the various node types in our database, the attributes associated with each type, and the total number of nodes of each type:

Table 1: Nodes and Attributes

Node Type	Attributes	Number of Nodes
Training Process	name_en, name_es, description_en, description_es	5
Step	id, description_en, description_es, isMainStep	36
User	id, role, expertise, training_is_started, active_training_name_en, active_training_name_es	4
Image/Video/Documents	id, description_en, description_es, url	2

Relationships between nodes: Relationships are the links or connections between nodes, establishing context and meaning to the entities within the database. Each relationship has a direction and a type, which describes how nodes are related to each other. By traversing these relationships, we can derive insights, patterns, and connections that would be complex or impossible to deduce in traditional relational databases. The table below provides an overview of the various relationship types in our database, detailing the nodes they connect and any additional attributes or properties associated with these relationships.

Table 2: Relationships between Nodes

Relationship Type	From Node	To Node
STARTS	TrainingProcess	Step
HAS	TrainingProcess	Step
NEXT	Step	Step
HAS_FOLLOWUP	TrainingProcess	TrainingProcess
ACTIVE	User	Step
ANSWERED	User	Step
HAS_MEDIA	Step	Image/Video/Documents

The database can be queried using the Cypher Query Language, which is specially designed for querying graph databases. For integration with our application layer, we leverage the Apollo Server, which acts as an intermediary between the Neo4j database and our front-end application. By using a schema generated from Neo4j, Apollo Server ensures that our GraphQL layer is tailored to our data model. This integration facilitates real-time data retrieval and updates, enhancing the user experience.

The structure and relationships of the database can be understood by the below figure, which illustrates the database graph:

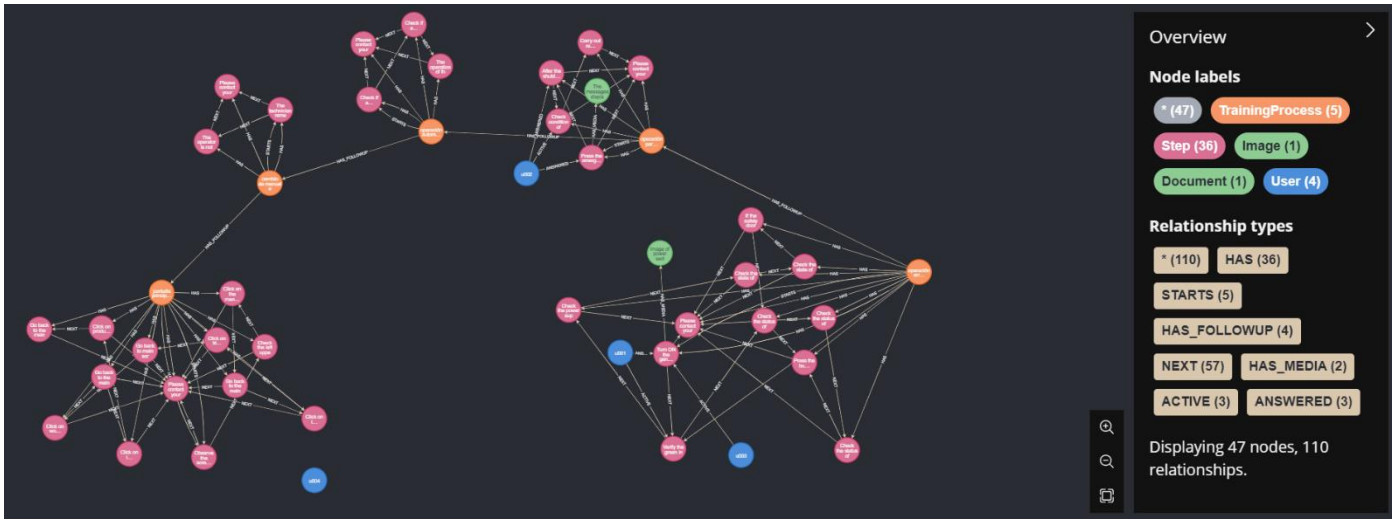

Figure 1: Neo4j Database

5.1.2 Apollo server and GraphQL

The Rasa chatbot's performance relies on how it can access data from external databases. We use the Apollo server to query data from the Neo4j graph. Apollo Server is an open-source, spec-compliant GraphQL server that's compatible with any GraphQL client, including Apollo Client. It acts as the foundation for our data operations using a schema generated from Neo4j. This ensures that the GraphQL layer is perfectly tailored to our data model, allowing precise and efficient queries.

5.1.3 Rasa Chatbot

Rasa is an open-source chatbot framework using a configurable NLU pipeline, a dialog manager based on rules and probabilistic models, and a fulfillment server performing custom actions via Python code. The action server queries information from GraphQL and uses the results to decide how the chatbot responds (e.g., using a template with parameters filled by the query result).

Our implementation of the Rasa chatbot is twofold:

- Base Features: These encompass universally applicable intents and conversation patterns suitable for generic queries and tasks.
- Training-specific Features: Tailored to specific use case needs, these features are designed to handle particular queries, manage domain-specific interactions, and execute custom actions. These actions interact with other components of our system, such as Neo4j for data storage and retrieval and Apollo Server for seamless API interactions. The specifics of these interactions will be detailed in the section that follows.

5.2 Demonstration scenarios

This section delves into the demonstration scenario tailored around the Augmented Manufacturing Training Process. We looked at fifteen user stories and machine manuals to understand how new employees learn to operate the machine. The focus is on walking the new employees through the steps of operating the machine and the touch screen.

The identified training processes, crucial for onboarding new employees, are as follows:

- Starting Operation
- Emergency Stop-Reset Operation
- Automatic Operation
- Change from Manual to Automatic Operation
- Machine's Touchscreen

Our implementation has been carried out in both English and Spanish. At the moment, we have orchestrated the 'happy path' scenarios and have ventured into some facets of the 'unhappy path'. A **happy path** is a conversation without breakdowns or other obstacles. We will consistently implement these dialogs first and then incrementally add happy and unhappy paths and mechanisms to resolve problems. The more features we implement, the more happy and unhappy paths we need to identify and address to maintain a good user experience. In general, the bulk of the evaluation and improvement will be dedicated to these activities.

For demonstration purposes, we showcase the processes of:

1. Starting – continuing the training process
2. Navigating the training process

The following paragraphs outline the interaction from the users' perspective. We use screenshots from a chat interface built using Gradio to interact with the Rasa bot (but not the new COALA app). The assistant can recognize the same or semantically similar user utterances - i.e., inputs do not have to be the same as indicated in the following sections. Variations in wording may result in misunderstanding (conversational breakdown).

Below is a detailed explanation of the EPISCAN demonstration scenarios for the onboarding assistant.

5.2.1 Starting – continuing the training process

In this scenario, the assistant facilitates the employee's journey in initiating a new training process or proceeding with an already started training process. This process is designed to ensure that employees have a seamless experience while accessing the training resources they need.

Upon expressing the desire to start/continue a training process, the assistant first authenticates the user by asking for their user ID. Once the employee's identity is confirmed, the rasa bot retrieves the user connection to the training processes in the Neo4j graph to see if the user has an active training process. This scenario can be divided into two distinct paths:

1. Case one (Fresh Start): If the employee is initiating the training process for the first time, they will need to select a desired training process from the list provided. Once selected, the assistant will confirm the choice and guide the employee through the initial steps of the chosen training.
2. Case two (Resuming): For employees who have previously initiated a training process but did not complete it, the assistant recognizes their progress. Instead of starting over, these employees are automatically directed to the next step of the training process they have to complete. This ensures a seamless continuation, minimizing repetition and enhancing the learning experience.

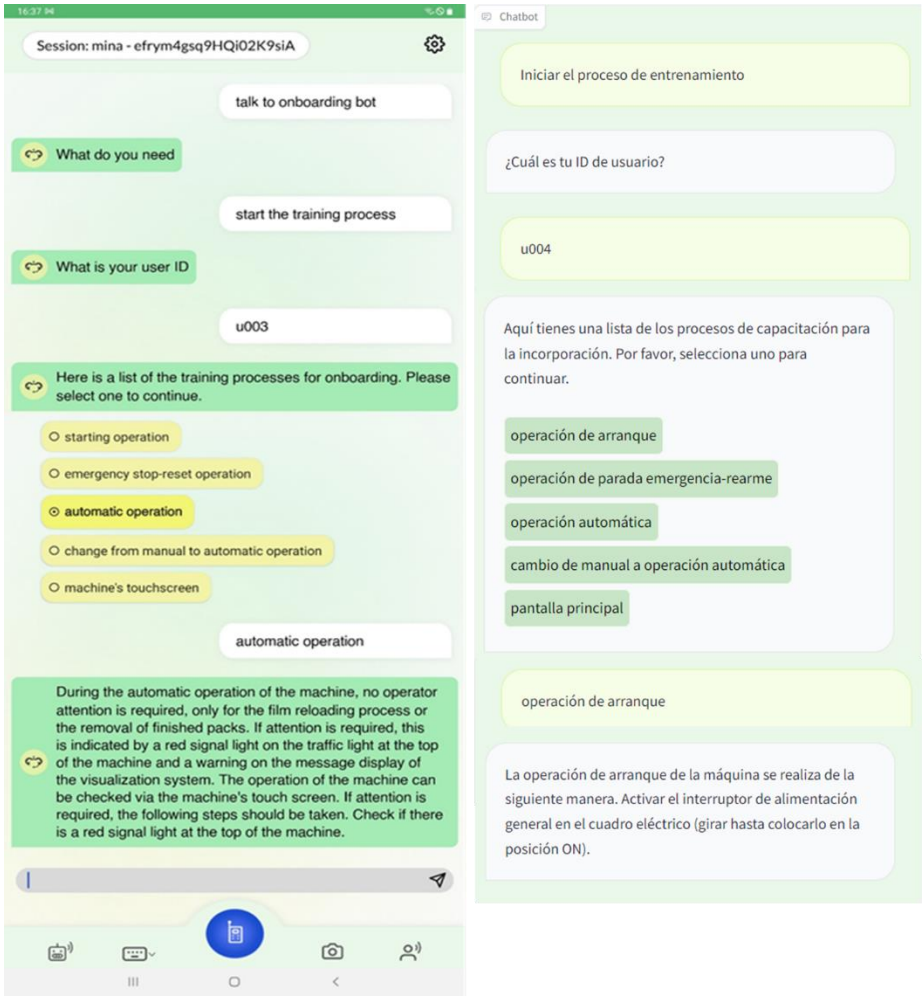

Figure 12: Starting a new training - English chatbot using COALA android app (left) and Spanish chatbot (right)

Table 5: View of start training process intent

Intent name	Entities	Description	Example utterance
request_start_training_process	user_id training_process_name	Users can ask the bot to start/continue a training process. They should then provide the required entities.	- Let's start the training. - Start the training for the automatic operation - Continue the training process

Table 6: View of request_start_training_process entities

Entity name	Description
user_id	The id of the user
training_process_name	The name of the training process

5.2.2 Navigating the training process

At each training step, users could do one of the following:

1. The user confirms the step and goes to the next one

2. The user encounters errors or issues or can not confirm the step during their training. In such cases, the assistant will either guide them through a series of sub-steps designed to rectify the problem or suggest contacting a supervisor for further assistance.

Table 7: View of training process navigation intents

Intent name	Entities	Description	Example utterance
Affirm	-	User confirms the step or request going to the next step.	- Yes - I confirm the sound - Yes, the light is green - Go to the next step
Deny	-	User rejects the step completion.	- No - No I could not hear the sound No the light is red

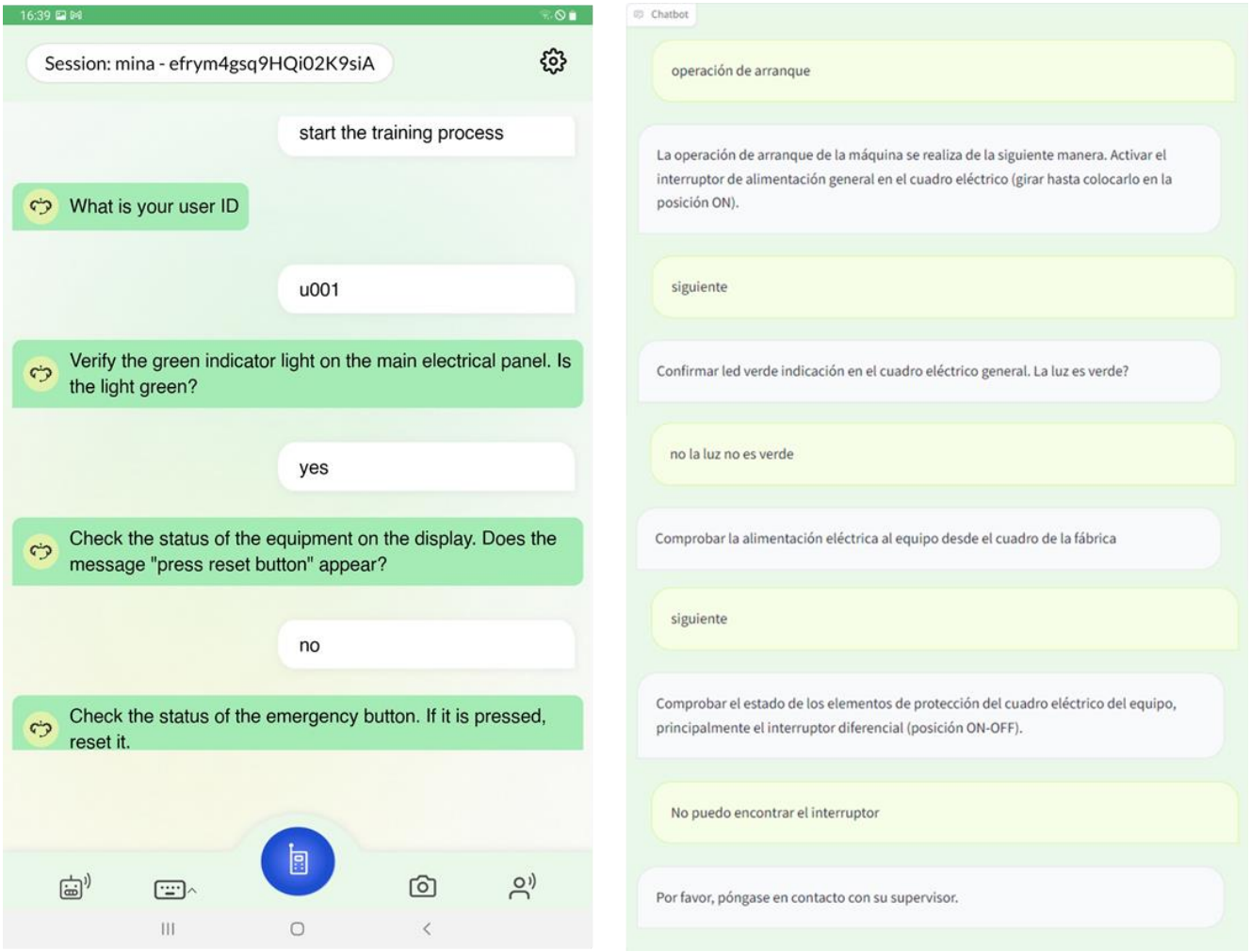

Figure 13: Navigating a training process - English chatbot using COALA android app (left) and Spanish chatbot (right)

6. ASSISTED QUALITY ASSURANCE FOR SUSTAINABLE PRODUCTS

This use case aims to enhance product quality testing processes by leveraging a digital assistant that supports workers in executing validation protocols for safer and more sustainable products. This assistant integrates machine learning models to optimize testing, reduce energy consumption, and provide real-time guidance, fostering collaboration between the assistant and operators. WASABI aims to demonstrate this solution across the REINOVA, TRIMEK, EPISCAN, CROMA, and SILK-BIO business cases corresponding to spanning automotive battery testing, dimensional metrology, EPI material testing, and prosthetics quality testing, respectively, to improve overall product quality and worker efficiency.

Starting from the existing digital twin for product testing currently in use at REINOVA, in the first phase, task T2.4 deployed and configured the digital twin in a Docker environment hosted in the cloud. Then, it customized and integrated the dialogue component released by T2.1 and added the data synthesizer component that will enable the digital twin to assist the operator in conducting product quality assurance procedures.

This section describes the outcomes of the activities performed during the first phase of the project. Specifically:

- It defines a digital assistant framework for product quality testing;
- It provides evidence of its feasibility through the REINOVA prototype;
- It provides an overview of how the design principles adopted for the REINOVA prototype can be adapted and extended (i.e., with additional skills) to support the other involved business cases, e.g., TRIMEK, COMA, EPISCAN, and SILK-BIO.

6.1 The digital assistant architecture for assisted quality assurance for sustainable products

Figure 14 shows the general architecture of the digital assistant for quality testing procedures leveraging the COALA technology stack (detailed in D1.3). The legacy infrastructure is meant to grant access to the quality testing data that are available in the data management system of the involved company. A wrapper rest API module is included to this end. Moreover, the architecture supports analytics and machine learning processes using the dedicated **Analytics Module** accessible through GraphQL servers.

Finally, its design aims to support workers in all the phases that characterize the quality testing procedures in supporting business case settings. To this end, it includes two distinct skills:

- **The status updating skill**: this skill is devoted to the support of all the updating procedures that are necessary to have quality testing data available and up to date. To this end, it assists workers during data insertions and updating by asking for the required feature values, checking for their correctness and completeness, and calling the related insertion and updating procedures through the rest API module;
- **The status monitoring skill**: this skill is devoted to the support of the quality testing procedures. To this end, it assists workers by providing up-to-date quality information on the procedures and products that are under testing in a user-friendly manner by querying the quality-related data through the rest API. Moreover, it optimizes testing, reduces energy consumption, and provides real-time guidance through the integration of information provided by the Analytics module.

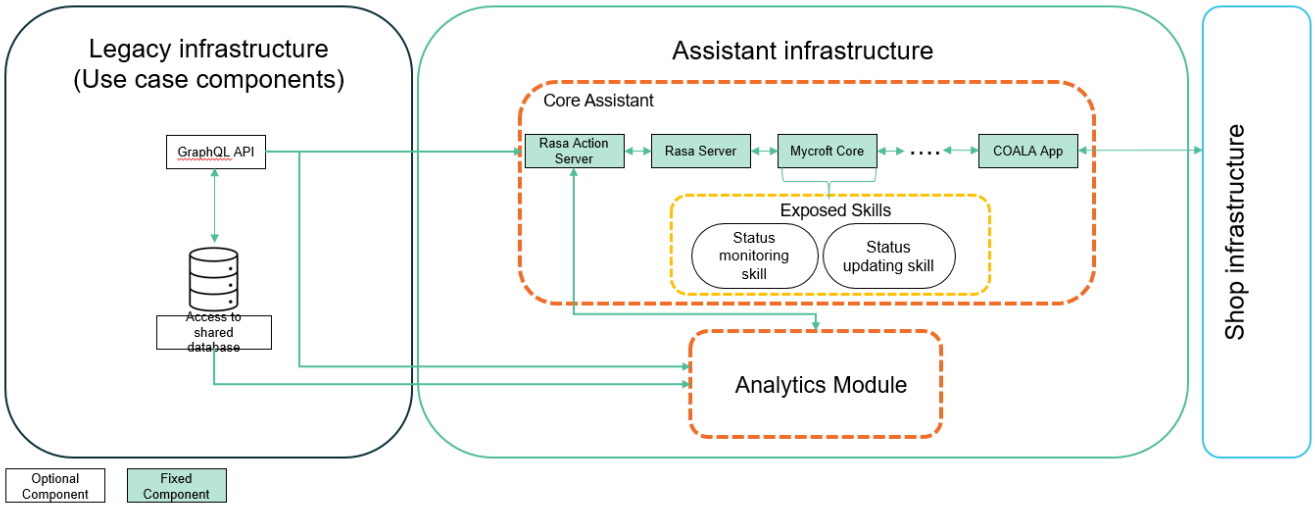

Figure 14: Architecture of the digital assistant for quality testing

It is worth noting that such general architecture can be applied to the diverse quality assurance procedures of the involved business cases (details are provided in deliverables D1.1 and D1.2) through concrete instantiations of the exposed skills and the enablement/disablement of components based on each company's specific requirements. The skills support required by the five business cases is summarized in Table 8.

Moreover, it will be required from the digital assistant to offer different access levels (through an authorization tool) based on different organizational roles, and adapt its architecture to the hierarchy and role-based architecture of the organization, i.e., once a user is signed in, the digital assistant will **redirect** them to the relevant skills.

Table 8: Overview of the involvement of skills in each business case

Business case	Status updating skill	Status monitoring skill
REINOVA	-	X
TRIMEK	-	X
EPISCAN	X	X
CROMA	X	X
SILK-BIO	-	X

6.1.1 The Analytics Module

6.1.1.1 Scope & Objectives

The Analytics Module is designed as a part of the **analytics core** of the "Quality Assurance for sustainable products" WASABI assistant. By operating passively, the Module is in a position to dive deep into field data to provide **real-time**, **accurate insights** to facilitate the product quality process. Specifically, the Module is designed to process the time-series **sensorial and status data** in real time and apply the predictive quality algorithms that are suitable for each use case.

The analytics functionalities in WASABI utilize the Analytics-as-a-Service (AaaS) paradigm over the whole lifecycle of data analytics (descriptive, predictive, and prescriptive analytics). They can be used to derive insights and build predictive models to support and assist quality assurance. The scope of the Analytics Module encompasses the

systematic **analysis** of incoming data to proactively identify and address potential issues and **timely inform** the end-user while **raising trust** by providing interpreted answers.

Therefore, the objectives of the Analytics Module include the following:

- The encapsulation of **suitable algorithms**
- The precise **dataflow** to have **timely results**,
- The incorporation of **interpretability techniques**

The above objectives are translated into three main conceivable functionalities that are depicted in Figure 15. The features can be utilized simultaneously or separately, depending on the use case.

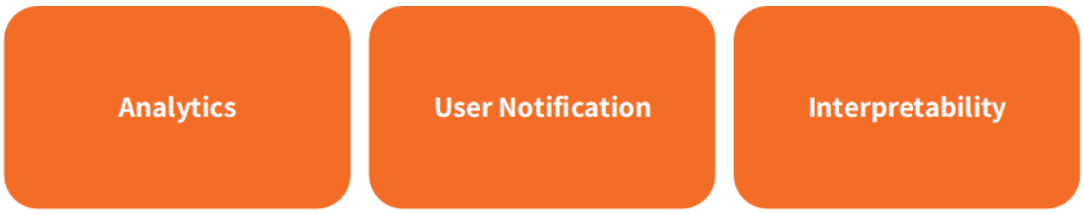

Figure 15: Analytics Module Features

In the following sections, these features will be presented and elaborated alongside the architecture of the AC and the connection and interaction with the Wasabi Assistant.

6.1.1.2 Features

Analysis Feature

The analysis feature encapsulates the algorithms that perform any kind of major or minor analysis task on the data of each pilot. This feature aims to enrich the capabilities of the Digital assistant with predictive quality insights.

User Notification Feature

The timely and accurate notification of end users constitutes a paramount aspect of **predictive quality management**, given that the efficacy of algorithmic outcomes may diminish over time. Conversely, an excess of notifications has the potential to disrupt the user experience and erode trust in analytics functionalities outcomes, thereby compromising valuable insights. Additionally, diverse job roles may necessitate or possess varying levels of authorization to access information commensurate with their data science proficiency or job requirements.

To address these considerations, analytics in WASABI offers a customizable reporting tool designed to furnish notifications pertaining to anomaly detections and operation summaries for a specified timeframe. This solution aims to strike a balance between timely communication of pertinent information and avoiding undue disruption to the user experience, ensuring that notifications are tailored to the specific needs and authorization levels of distinct user roles.

Interpretability Feature

To build up confidence and trust in the outcomes of analytics, an understanding of the process is critical. This feature is enhanced by a **customizable and configurable dashboard** that can be adjusted to each pilot's needs, visualizing the raw data and the algorithm outcomes in suitable ways to a) be easy to understand each machine's condition and b) be able to elaborate on details if needed. The role hierarchy that is presented in the previous section is also applicable to the dashboard, rearranging the information depth according to the user's background and needs.

Figure 16: AC Customizable Dashboard

Furthermore, the interpretability feature is enhanced via the algorithm selection and breakdown of the fusion mechanism that combines the outcomes of the algorithms.

6.1.1.3 Integration with the status monitoring Skill

Integration with the Digital Assistant Skill will provide the Analytics Module with essential usability within the WASABI assistant framework. The integration process is intricate, demanding a robust interface to guarantee smooth interaction between the components and the assistant framework. Furthermore, it is essential to ensure the **online availability of analytics outcomes that are continuously produced upon the arrival of new data**. To achieve this synergy, a secure and efficient API has been developed over the stored AC outcomes. This API serves as the linchpin, enabling secure access to meticulously processed information. Ensuring both **data integrity** and **confidentiality**, this connection acts as a gateway, allowing the Digital Assistant Skill to harness the analytical insights provided by the component.

As the developed solution progresses, to optimize the integration, specific security protocols or encryption standards could be incorporated into the API architecture. A deeper understanding of the processed data and validation procedures will facilitate tailoring the integration within the digital assistant context for an enhanced user experience. The fully integrated solution is expected to enhance the productivity of production lines and, therefore, customer satisfaction.

6.2 Assisted quality assurance skills for REINOVA

In the first phase, we deployed and configured the existing digital twin for product testing currently in use at REINOVA[3] in a Docker environment hosted in the cloud. Then, we customized and integrated the dialogue component released by T2.1 by following the architecture described in Section 3.

[3] A description of this platform has been uploaded on the WASABI project repository for inspection as needed.

Specifically, REINOVA provides testing and validation of e-mobility components such as modules and battery packs; the values of different testing parameters are continuously collected from the sensors located in chambers and batteries under testing.

In this context, the digital assistant aims to assist technicians and engineers in a laboratory setting where advanced test machines are running, often long-lasting, tests on various devices (Device Under Tests, DUT). By providing real-time status updates and presenting retrospective data, the digital assistant can contribute to the safety of the work environment and improve work efficiency in general. To this end, the first digital assistant prototype implemented the status monitoring skill.

The following subsections describe the following prototype modules:

- The RASA component;
- The REST API developed to get access to the sensor data;
- The Analytics module aiming to provide useful insights about quality testing procedures to support decision-making processes.

6.2.1 Rasa component

The RASA component is designed to interpret and respond to specific user queries by recognizing intents and extracting relevant entities. This section outlines the intents and entities that form the core of the system's natural language understanding capabilities.

6.2.1.1 Understanding Intents

An intent represents the purpose behind a user's input. In our system, intents are categorized based on the type of information the user is seeking from the status monitoring. The primary intents identified for our system are:

1. **Status Query**: This intent is triggered when a user inquires about the general status of a chamber. It is designed to provide a comprehensive update on the chamber's condition based on the incoming data from the APIs.
2. **Temperature Test**: When a user asks about the temperature of a specific chamber, this intent is invoked to fetch and relay temperature details.
3. **Battery Status Query**: This intent is used when the user wants to know the battery status of a specific tester. It delivers information about the battery's current state.
4. **Chamber List Query**: This intent is connected with the action which is responsible for fetching the list of all the chambers.
5. **Battery List Query**: Similar to the chamber list intent, it is connected with the action to fetch the list of battery testers available.
6. **Criticality Status Query**: This intent is a part of the analytics component, and it indicates the criticality of an inquired chamber. It uses the chamber_id entity to return the results.
7. **Show DEFCON Levels**: This intent is part of the analytics module. Its purpose is to handle the queries related to the DEFCON levels of the chambers.
8. **Show Specific DEFCON Levels**: This intent will handle the queries related to specific DEFCON levels, i.e., Alarm, Ready, and Maintenance. It uses defconValue as an entity in the conversation.
9. **Show DEFCON Levels with Date Filter**: This intent is used while querying DEFCON level's occurrence with a specific date and time. It can also involve the startDate and endData entities in the conversation.

10. **Show DEFCON Levels with Relative Time Filter**: This intent is used to resolve the relative date and time mentioned in the conversation while asking about the DEFCON levels. It uses relativeDate entity in the conversation.
11. **Show DEFCON Levels with Count Filter**: This intent is used to resolve the count of DEFCON level occurrences.
12. **Show DEFCON Levels with Complex Filter**: This intent can contain both DEFCON level and time filter in it. It can catch both defconValue and specificDate or relativeDate entities to apply the filters and to get the answers.

Each intent is associated with a set of user utterances that the system can recognize. These utterances are phrased in various ways to ensure the system is robust in understanding the user's language.

Additionally, two optional intents are introduced in the system to query the list of the entities (i.e., chamber_id, battery_tester_id). It might be useful to see the list of already active chambers and battery testers in the system.

6.2.1.2 Defining Entities

Entities are crucial components that the system identifies and uses to provide a context-specific response. These entities are usually used as parameters to query data from the APIs. For the intents mentioned, the entities are:

- **chamberId:** A unique identifier for a chamber within the system. It allows the system to fetch and provide data for a specific chamber. (Example: B1, B2, B3)
- **batteryTesterId:** A unique identifier for a battery tester. This entity enables the system to retrieve and communicate the battery status for the specified tester. (Example: BT1, BT2, BT3)
- **defconValue**: This entity is used to catch the DEFCON level mentioned in the conversation (Example: Ready, Alarm, Maintenance)
- **startDate**: This entity is used in the show_defconlevel_date_filter intent to apply a filter in the query.
- **endDate**: This entity is also used in the show_defconlevel_date_filter.
- **relativeDate**: This entity is used in the show_defconlevel_relative_date_filter entity to catch relative time occurrences in the conversation related to DEFCON levels.
- **topValues**: This entity is used in the show_defconlevel_sum_filter intent to catch the count.
- **specificDate**: This entity is used in show_defconlevel_complex_filter intent to store a specific mention of date and time in the conversation.

6.2.1.3 Example of User Utterances for Each Intent

For each intent, multiple user utterances are mapped to ensure flexible and natural interactions. Below are examples for each intent:

- **Status Query Intent**
  - "Can you tell me the status of chamber {chamber_id}?"
  - "What's the current status of {chamber_id}?"

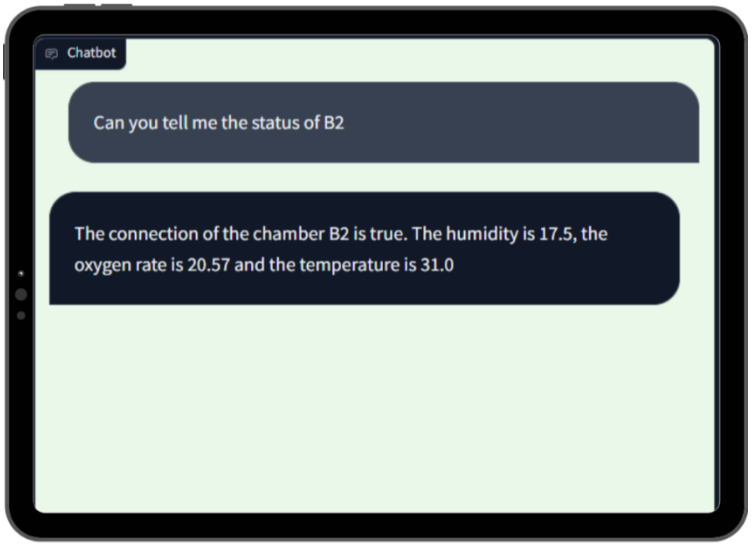

Figure 17: Status query Interaction

- **Temperature Test Intent**
  - "What is the temperature of chamber {chamber_id}?"
  - "How warm is chamber {chamber_id}?"

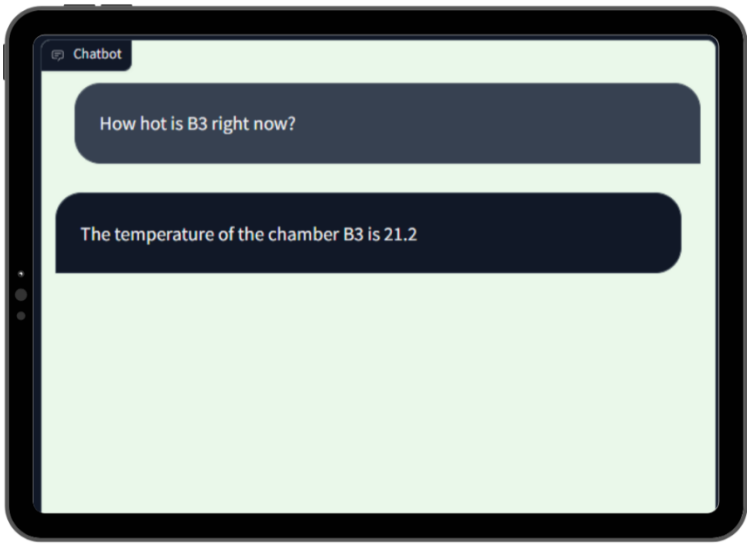

Figure 18: Temperature query interaction

- **Battery Status Query Intent**
  - "Check the battery for tester {battery_tester_id} for me."
  - "What is the battery status of tester {battery_tester_id}?"

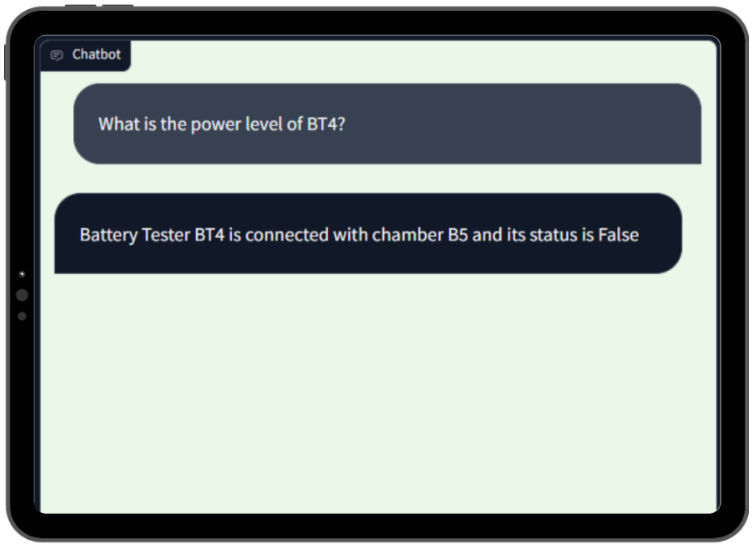

Figure 19: Battery tester query interaction

- **Chamber List (Optional)**
  - "Can you list all the chambers?"
  - "Show me all the chambers we have."

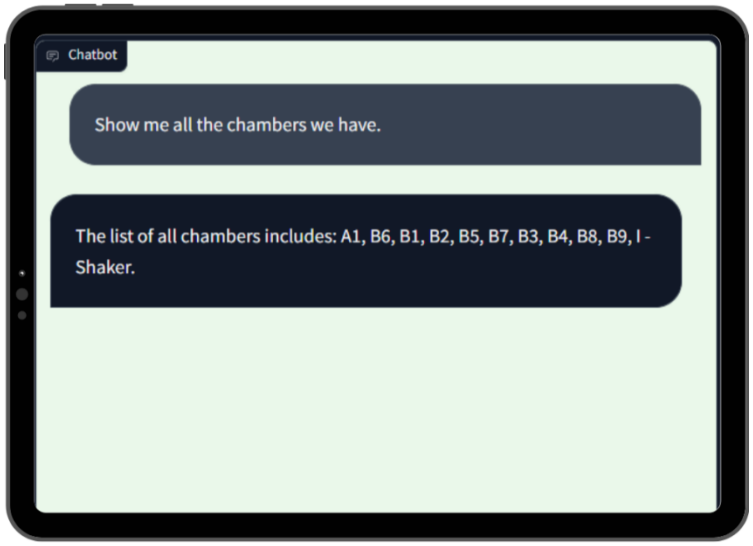

Figure 20: Chamber list query interaction

- **Battery Tester List (Optional)**
  - "Can you list all the battery testers?"
  - "I need a list of all battery testers."

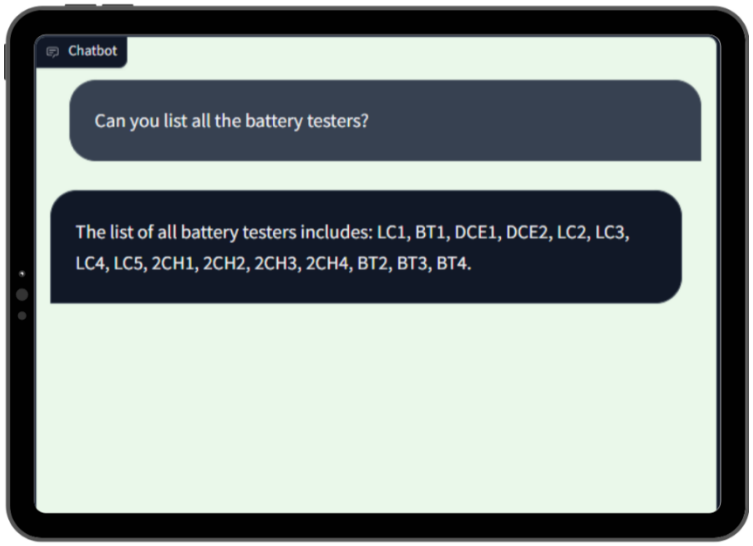

Figure 21: Battery tester list interaction

- **Criticality Level Intent**
  - "What is the criticality level of chamber B2?"
  - "Criticality of B17"

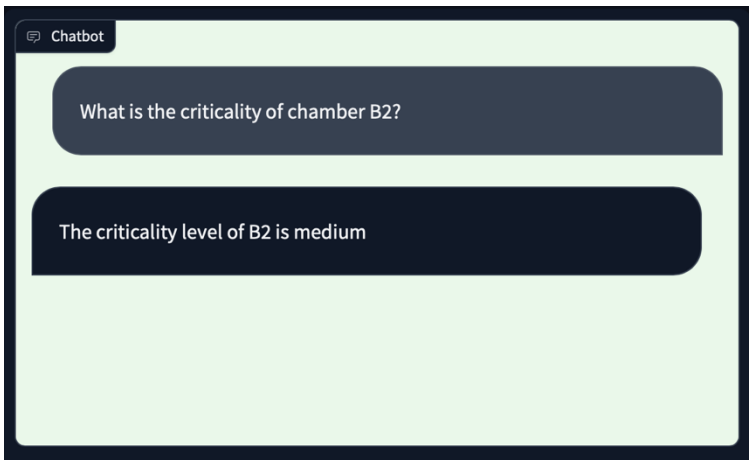

Figure 22: Battery tester list interaction

- **Show DEFCON Levels Intent**
  - "What are the occurrences of DEFCON level?"
  - "Show me the DEFCON level counts of the chambers."
- **Show Specific DEFCON Levels Intent**
  - "How many times did DEFCON level equal to Maintenance?"
  - "Please provide the number of Alarm level occurrences."
- **Show DEFCON Levels with Date Filters Intent**
  - "Show me DEFCON level occurrences after 2023-07-01"
  - "I need information on DEFCON levels from 2023-05-01 to 2023-05-31"
- **Show DEFCON Levels with Relative Time Filters Intent**
  - "Show DEFCON levels since morning"
  - "Show DEFCON level summary 6 months ago"
- **Show DEFCON Levels with Count Filters Intent**
  - "What is the maximum number of DEFCON level occurrences on any day?"
  - "Can you list the top 5 of the highest DEFCON level counts?"

- **Show DEFCON Levels with Complex Filters Intent**
  - "How many times did DEFCON level equal to Maintenance occured on 2023-07-01?"
  - "Please provide the number of Alarm DEFCON level occurrences on 2023-09-30"

6.2.1.4 Entity Extraction and Intent Processing

The system uses Rasa NLU to extract entities from the user's speech. When a user speaks one of the utterances, Rasa NLU parses the sentence to identify the intent and any entities present. For example, if a user says, "How much battery does tester B123 have left?" the system recognizes the intent as a "Battery Status Query" and extracts "B123" as the battery_tester_id entity.

6.2.1.5 Intent Handling and Response Formulation

Once an intent and its associated entities are identified, the system formulates a response. This involves invoking external REINOVA APIs, receiving the data, and then constructing a human-readable message that is relayed back to the user through Mycroft.

6.2.1.6 Rasa Action Server

The Rasa Action Server is a crucial component of the Rasa component, responsible for executing actions in response to user intents. It operates in conjunction with Rasa Core to perform actions that are too complex to handle with simple responses directly within Rasa Core. These actions can include querying databases, calling external APIs, and processing data before responding to the user. In the context of this system, the Rasa Action Server is tasked with:

- **Executing Custom Actions**: When a predefined action is triggered by an intent, the Action Server executes the corresponding custom action.
- **Communicating with External GraphQL APIs**: For intents that require real-time data, such as the status of a chamber or battery level, the Action Server handles the communication with the GraphQL APIs. It also communicates with analytics module.
- **Data Processing**: After fetching the required data, the Action Server processes it to create a user-friendly response.
- **Responding to Users**: It sends the processed information back to Rasa Core, which then delivers the response to the user through Mycroft.

6.2.1.7 Custom Actions for Intents

For each intent within the Rasa Action Server, a corresponding custom action is defined:

- **Status Query Intent Action**: An action that queries the GraphQL database via the Apollo Server to retrieve the status of a chamber.
- **Temperature Test Intent Action**: An action that fetches the current temperature of a chamber from the database or REINOVA APIs.
- **Battery Status Query Intent Action**: An action that checks the battery status of a tester by making an API call to the REINOVA APIs.
- **Criticality Level Intent Action**: An action that checks the criticality level for a specific chamber.
- **Show DEFCON Levels Intent Action**: An action to get the DEFCON level occurrence for multiple chambers.

- **Show Specific DEFCON Levels Intent Action**: An action to get the response for specific DEFCON level i.e. Ready, Alarm or Maintenance.
- **Show DEFCON Levels with Date Filter Intent Action**: An action that queries the date for DEFCON levels with date filters.
- **Show DEFCON Levels with Relative Time Filter Intent Action**: An action that resolves the relative date and time, queries data from the analytics module, and formulates a response.
- **Show DEFCON Levels with Count Filter Intent Action**: An action that applies count and applies the filter on the DEFCON level and generates the response.
- **Show DEFCON Levels with Complex Filter Intent Action**: An action that can include DEFCON level values and a specific date and time to query the data from the analytics module.

6.2.1.8 Communicating with REINOVA and Analytics GraphQL APIs

The Action Server handles GraphQL API interactions by:

- **GraphQL API Requests**: Making GraphQL requests to the endpoints with the necessary parameters.
- **GraphQL API Response Processing**: Parsing the JSON responses from the APIs to extract relevant data.

6.2.2 REINOVA APIs


```
{
    "data": {
        "getChamberInfo": {
            "chambername": "B1",
            "datetime": "2023/11/14 17:31:52'
            "connectionstatus": "true",
            "defconlevel": "Ready",
            "activebattery": [
                "BT2"
            ],
            "temperature": 24.3,
            "humidity": 57.6,
            "pressure": 0,
            "emergencyrelay": false,
            "doorstatus": false,
            "maintenance": false,
            "fault": false,
            "oxygen": 20.57,
            "gn2valvestatus": false,
            "gn2presflowstatus": false,
            "gn2keystatus": false,
            "washingvalvestatus": false,
            "washingpresflowstatus": false,
            "fire": false,
            "toxic": false,
            "explosive": false,
            "toxicpre": false,
            "explosivepre": false,
            "fgfault": false,
            "drainvalve": false
        }
    }
}
```


(a)


```
{
    "data": {
        "getBatteryTesterInfo": {
            "btname": "LC1",
            "commonalarm": false,
            "connectedchamber": "B6",
            "connectionstatus": true,
            "datetime": "2023/11/14 17:30:51
            "emergencybutton": true,
            "holdtestsequence": false,
            "isulationfault": true,
            "releaseoutputcontactor": false,
            "statusoutputcontactor": true
        }
    }
}
```


(b)

Figure 23: Samples of REST API answers

The legacy infrastructure of the digital assistant includes a wrapper for a number of GraphQL APIs that can be called to gain updates on the statuses of the company's instruments. Internally, the Wrapper implements all communication drivers with the REINOVA systems (Modbus, ADS, TCP, REST, etc.). The historical data regarding

the status of both the climatic chambers and the battery testers is stored in tables in the dataset provided by REINOVA.

The following list contains some documentation samples regarding the different APIs implemented. Samples of the possible answers in JSON format are shown in Figure 23.

- **getChamberInfo(chamber: String!)**: returns the latest status information of the selected chamber
- **listChamberInfo**: returns the latest status information of all the chambers present in the database
- **getBatteryTesterInfo(batterytester: String!)**: returns the latest status information of the selected battery tester
- **listBatteryTesterInfo**: returns the latest status information of all the battery testers present in the database

6.2.3 Analytics in the REINOVA use case

Analytics functionalities in WASABI are fully customizable and modular based on the needs of each use case. For REINOVA's use case, the **assisted quality features** were developed by gaining information and data related to the health of the equipment that they mostly use.

Therefore, for the first Demo, the flow was implemented to preprocess and analyze the data via simple rule-based heuristic algorithmic techniques that a primal Exploratory Data Analysis has reinforced to calculate **the criticality level** of each equipment (chamber) regarding three major threats: **fire, toxicity, and explosivity**. The early detection of this divergent behavior of the equipment is crucial as, firstly, timely correction adjustments may decrease the downtime, and secondly, it could help avoid dangerous situations for the workforce and clients' assets. The analysis is accompanied by a dashboard that depicts the raw data alongside the outcomes of the analysis.

6.2.3.1 Assistant interaction

The whole data flow was based on two main pillars:

- The criticality level of the equipment regarding major threats, and
- The current user's role and background.

Therefore, there are three main question categories that the AC can address:

- What is the fire criticality level of the #chamber_name chamber?
- What is the toxicity criticality level of the #chamber_name chamber?
- What is the explosivity criticality level of the #chamber_name chamber?

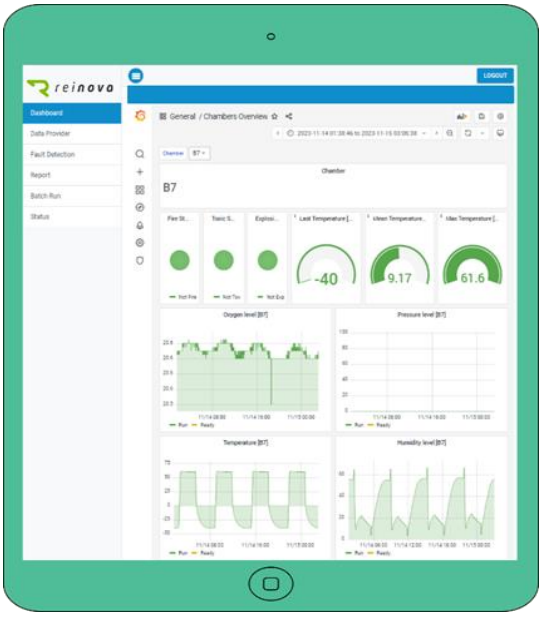

Figure 24: Analytics dashboard view for REINOVA

Moving forward, to address the role hierarchy order, three roles were established: the Operator, the Manager, and the Engineer. Each one of them receives information about the same analysis outcomes, hence in a different interpretation approach as the following list and Figure 25 depicts.

- **The Operator**. The operator receives a response with simple descriptive words of the situation.

- **The Manager.** The manager receives a response with a more detailed answer that indicates the exact situation of the equipment.
- **The Engineer.** The engineer receives the most detailed response that also includes interpretability features referring to the analysis details that have led to the outcome of the AC.

Figure 25: AC Role Hierarchy Responses

6.2.3.2 Examples of queries

To support the skills for REINOVA in the first demo, the analyses that were developed are exposed through related queries. We hereby present three queries: (i) instrument monitoring, (ii) duration of alarm, and (iii) duration of emergency. The "Status Monitoring" skill provides insights about the status of the instruments used, i.e., Chambers and Battery Testers. In addition, in the following examples, the filters implemented in the Analytics Results Query Engine can be applied to retrieve more sophisticated results and thin out the response.

1. Instrument monitoring

   The Instrument monitoring analysis counts the recorded entries from the Chamber dataset based on their status (defconlevel) per day. In this case, the filters can be used to get the results for a specific status, time interval, or the top/min/max counts.

Table 9: Instrument monitoring query

Query	Response

query alarms{
resultRequest(request:{
request: "GROUP_BY_DEFCONLEVEL"
requestFrom: "DA"
}){
results
reason
}}
	{"data": {
	"resultRequest": [
	{
	"results": [
	{
	"Date": "2023-08-20",
	"defconlevel": "Maintenance",
	"count": 2364
	},
	{
	"Date": "2023-08-20",
	"defconlevel": "Ready",
	"count": 11245
	},
	{
	"Date": "2023-08-20",

	<pre>"defconlevel": "Run",
	"count": 11713
	},……..</pre>

2. Duration of alarm

In this analysis, the duration of the flagged alarms are calculated. For each chamber separately, the start and end timestamps are recorded for the entries with the "Alarm" status following two formats: i) as a timestamp (HH:MM:SS) and ii) in seconds. In this example, the filters can be used to get the results for a specific chamber, time interval, or based on the duration.

Table 10: Duration of alarm query

Query	Response

<pre>query alarms{
resultRequest(request:{
request:
"DURATION OF CHMABER ALARM"
requestFrom: "DA"
}){
results
reason
}
}</pre>
	<pre>{"data": {"resultRequest": [{
	"results": [
	{
	"chambername": "A1",
	"defconlevel": "Alarm",
	"start": "2023-08-21 14:27:01",
	"end": "2023-08-21 14:27:32",
	"duration": "00:00:31",
	"seconds": 31.095
	},
	{
	"chambername": "B",
	"defconlevel": "Alarm",
	"start": "2023-08-22 17:08:35",
	"end": "2023-08-22 17:08:35",
	"duration": "00:00:00",
	"seconds": 0
	},
	{
	"chambername": "B",
	"defconlevel": "Alarm",
	"start": "2023-08-21 09:51:10",
	"end": "2023-08-21 09:51:10",
	"duration": "00:00:00",
	"seconds": 0
	},……..</pre>

3. Duration of emergency

In this analysis, the duration of the flagged emergencies are calculated. For each battery tester separately, the start and end timestamps are recorded for the entries with the emergency flag set to true in the following two formats: i) as a timestamp (HH:MM:SS) and ii) in seconds. Similarly to the aforementioned examples, the filters can be used to get the results for a specific battery tester, time interval, or based on the duration.

Table 11: Duration of emergency query

Query	Response

```
query alarms{
      resultRequest(request:{
        request: "DURATION OF TESTER EMERGENCY"
        requestFrom: "DA"
      }){
        results
        reason
      }
    }
```


```
{"data": {"resultRequest": [{
        "results": [
           {
            "btname": "BT1",
            "emergencybutton": true,
            "start": "2023-06-22 07:33:04",
            "end": "2023-06-22 11:12:02",
            "duration": "03:38:57",
            "seconds": 13137.935
          },
          {
            "btname": "BT1",
            "emergencybutton": true,
            "start": "2023-06-22 11:12:57",
            "end": "2023-06-22 11:13:32",
            "duration": "00:00:34",
            "seconds": 34.776
          }, …….
```


4. Duration of defconlevel

   In this analysis, the duration of all the chamber statuses are calculated. For each chamber separately, the start and end timestamps are recorded for each status, and the durations are calculated in two formats: i) as a timestamp (HH:MM:SS) and ii) in seconds. In this example, the filters can be used to get the results for a specific chamber, status, time interval, or duration.

Table 12: Duration of chamber defconlevel query

Query	Response
`query alarms{ resultRequest(request:{ request: "DURATION OF CHMABER DEFCONLEVEL" requestFrom: "DA" }){ results reason } }`	`{"data": {"resultRequest": [{ "results": [{ "chambername": "A1", "defconlevel": "Alarm", "start": "2023-08-21 14:27:01", "end": "2023-08-21 14:27:32", "duration": "00:00:31", "seconds": 31.095 }, { "chambername": "A1", "defconlevel": "Ready", "start": "2023-08-20 04:14:21", "end": "2023-08-20 23:57:07", "duration": "19:42:45", "seconds": 70965.418 }, { "chambername": "A1", "defconlevel": "Maintenance", "start": "2023-08-20 23:58:09", "end": "2023-08-21 01:03:57", "duration": "01:05:48", "seconds": 3948.301`

	}, …….

5. Example with filters

The following example provides a query leveraging the power of all three types of available filters. The scenario here represents the results retrieval from the "Duration of defconlevel" analysis, where we specifically want to request the top 2 durations for chamber A1 with the status Ready after the date 2023-08-22.

Table 13: Example with filters

Query	Response

query alarms{
resultRequest(request:{
request: "DURATION OF CHMABER DEFCONLEVEL"
requestFrom: "DA"
requestFilters: [{
name:"chambername"
action: "equals"
values: "A1"
},
{
name: "defconlevel"
action: "equals"
values: "Ready"
}]
dateFilters: {
name: "start"
action: "greater"
start: "2023-08-22"
}
sumFilters: {
name: "seconds"
action: "topn"
numberOfValues: "2"
}
}){
results
reason
}
}
	{
	"data": {
	"resultRequest": [
	{
	"results": [
	{
	"chambername": "A1",
	"defconlevel": "Ready",
	"start": "2023-08-23",
	"end": "2023-08-23 10:27:18",
	"duration": "00:54:58",
	"seconds": 3298.528
	},
	{
	"chambername": "A1",
	"defconlevel": "Ready",
	"start": "2023-08-22",
	"end": "2023-08-22 21:11:53",
	"duration": "00:02:02",
	"seconds": 122.688
	}
	],
	"reason": []
	}
	]
	}
	}

6.2.3.3 Future possible expansion

The first version of analytics presents a complete analysis solution that adds an intelligence level to the WASABI Assistant. However, the Reinova business case offers many possibilities. The following list summarizes the expansion thoughts about the exploitation of the full capabilities of analytics:

- Employment of more sophisticated algorithms upon more data arrival.
- General monitoring of the health of the equipment based on unsupervised anomaly detection analysis of new data.
- Employment of predictive analytics algorithms for further exploitation of the data that will become available.

6.3 Skill design principles for the other business cases

Starting from the business case requirements presented in D1.1 and the use cases and dialogue samples contained in D1.2, in the following sections, we provide the main design principles that can be adopted to deal with the quality assurance procedures of the other business cases. It is worth noting that all such business cases still need to be investigated, especially with reference to the features of the company's legacy infrastructure and quality-related data that could be exploited to assist quality assurance procedures. These activities will be carried out during the second phase in cooperation with WP1.

6.3.1 Assisted quality assurance for TRIMEK

TRIMEK manufactures and sells Coordinate Measuring Machines (CMMs) that can be used with the Metrology Software, of which TRIMEK is also proprietary. These products fit in several quality control processes in the manufacturing industry. However, the use of the software and the machine requires specialized metrology knowledge. TRIMEK provides training when selling a product, but the use of an assistant can help in the optimization of these procedures. TRIMEK personnel also use these devices and need to learn how to perform the measurement process correctly.

The WASABI quality assurance assistant should act as a complete interactive usage instruction or user's manual of the CMM in a step-by-step manner, i.e., it should encompass a "decision tree" of the intra-organizationally defined "templates" (e.g., template 1 in the dialogue box of deliverable D1.2) based on the visual characteristics of the object of interest. The categorization of the objects and the intentions of measurements (e.g., part inspection, standard QC of mass production, verification of design possibility) should be the defining criteria upon which the decision tree is created.

Therefore, the digital assistant, in the case of TRIMEK, will contribute to the quality of the operator's performance through the resolution of the procedure and clarification of the complex machine operation steps for the operator. In this sense, the measurement process has been divided into six major steps:

1. Define the coordinate system
2. Alignment of the coordinate system
3. Verification of alignment
4. Global inspection
5. GD&T inspection
6. Reporting

For the scope of the WASABI project, it was decided to work with the first three steps. For the application of a specific 'template of guidance', it is necessary also to define the part to inspect and the intention of the measurement process.

The operator first defines the object of interest geometrically (e.g., cylindrical, conical, etc.). In relation to what is described above, the idea is that the assistant could guide the worker in the steps to follow in order to define, align, and verify the coordinate system correctly, depending on the part and the intention for the measurement process (e.g., 1st part inspection, standard QC of mass production, verify the possibility of a design). Describe the product quality testing process and how data are collected and used by operators.

Based on the information above, possible concrete instantiations of the defined skills are the following:

1. **Status update skill:**

- Report delivery skill (by the operator), keeping track of the operator measurement actions performed and measurement processes completed in a self-reporting and interactive manner.

2. **Status monitoring skill:**
   - Template of guidance retrieval skill (by the operator);
   - Report monitoring skill (by the manager, supervisor, or the technical decision-making entity).

6.3.2 Assisted quality assurance for CROMA

CROMA manages the sterilization process of surgical instruments in a sterilization plant at the Hospital of Burgos (Spain). There are two steps in the sterilization process that require the use of equipment with validated work cycles, which must be verified by the operator. These processes are automated washing, which is carried out through washer disinfectors, and sterilization, through autoclaves. In washer-disinfectors, the cycle time, temperature, and detergent concentration must be checked, and in autoclaves, the cycle time, temperature, and pressure. The sterilization process is traced by the operator with the support of a proprietary instrument tracking software. Similar to the case of TRIMEK, we could define:

1. **Status update skill:**
   - Report delivery skill (by the operator), keeping track of the operator actions performed and tasks completed in a self-reporting and interactive manner. This information must to be stored in the database of the tracking software.
2. **Status monitoring skill:**
   - Report monitoring skill (by the manager, supervisor, or any technical decision-making user).

6.3.3 Assisted quality assurance skills for EPISCAN

In EPISCAN, the process of quality assurance is quantitative. The initial materials, such as PP meltblown, are provided by ERHARDT, etc., as noted in the forms. These materials already comply with standards EN 14683, etc. (more info at https://erhardtnw.com/en/pp-meltblown/). Therefore, the quality assurance in regard to EPISCAN mainly relates to the quality of the production process. Specifically, it is quantitatively measured concerning the optimality of manufacturing the end-product, that is, masks, from the initial materials. These numbers include the quantity of each initial material, the number of masks produced from a certain amount of initial materials, and the amount of waste of each initial material after production. This information is demanded with their temporal details, e.g., on a specific date, daily, weekly, and monthly report. In addition to operator shift forms and detailed Excel files, commercial software is used for desired statistics.

In the case of Episcan, therefore, it is of utmost importance to implement a legacy infrastructure supporting a data exchange protocol in a popular format, e.g., JSON, to join operators' shifts information that is stored in a legacy manner (shift forms), and excel files related to the inventory management, production parameters, alarms, and emergencies, as well as statistics.

Moreover, the specification of the skill support for this business case could be as follows:

1. **Status update skill**:
   - Report delivery skill (by the operator), keeping track of the operator actions performed and tasks completed in a self-reporting and interactive manner - also the rate of mask-production based on the adjustments made on the production line instruments, report of technical defects in an instrument, the amount of the initial materials in the inventory, etc.
   - Recording an alarm event and emergency situation, through which a track of such situations is kept and stored for future reference.
2. **Status monitoring skill**:

- Report monitoring skill (by the manager, supervisor, or the technical decision-making entity)- this skill provides the manager with all the aforementioned information reported by the operators.
- Alarm events and emergency situations' monitoring skill, based upon which the production may be halted or planned at different rates, machines can be inspected/replaced, etc.

6.3.4 Assisted quality assurance skill for SILKBIO

For SILKBIO, the digital assistant will support the process of solubilisation and casting of silk fiber. Specifically, it will support the operators' manual activity and their interaction with the equipment involved. Potentially, it can be applied in all process steps where the operator has to make a check or where he has to give a confirmation of activity. Today, this is done by an operator writing up steps performed and actions taken on a piece of paper, documenting each step of the procedure and signing off the paper when the process/subprocess has been completed. There are at least 15 such subprocesses to be documented for each "production batch". All documentation is then entered manually into a spreadsheet. It is necessary to keep the documentation for years to be able to retrieve it and present it when clients, customers, authorities, or others with a right-to-know ask. Therefore, the digital assistant can support the real-time recording of the information above.

To this end, we could introduce the following skill instantiations:

1. **Status update skill**:
   - Report delivery skill (by the operator), keeping track of the operator actions performed and tasks completed in a self-reporting and interactive manner, i.e., for every action performed by the operator, there must exist an acknowledgment (that yields the next step)
2. **Status monitoring skill**:
   - Solubilization and casting process retrieval skill (by the operator)- which support the process selection;
   - Report retrieval skill (by the manager, supervisor, or the technical decision-making entity)- which provides the user with the records of the interactions between the operators and the digital assistant and the production statistics.

7. CONCLUSION AND OUTLOOK

The WASABI project aims to demonstrate digital intelligent assistant (DIA) solutions for SMEs and mid-caps to promote sustainability and streamline the onboarding of new workers, featuring collaborative efforts with five partners (CROMA, EPISCAN, REINOVA, SILKBIO, and TRIMEK). This report introduces Deliverable 2.1, "Joint WASABI Demonstrator – Version 1", highlighting achievements from the initial project phase (M1-M9) and detailing the application of WASABI solutions within three use cases:

- Augmented waste management and valorisation for TRIMEK and CROMA (Task 2.2)
- Assisted workforce management for EPISCAN (Task 2.3)
- Assisted quality assurance for sustainable products for REINOVA, TRIMEK, CROMA, EPISCAN, and SILKBIO (Task 2.4)

In the first phase of the project, we created early prototype versions of the DIA skills and their involved components based on the scenarios, user stories, and requirements collected from the partners in D1.1 and D1.2.

In the first use case, augmented waste management and valorisation, two main components are involved: the waste inspector DIA based on the COALA base stack and the rEUse platform rEUse is a Web application that provides the ability to define models of data entities by modeling and describing them using the Topic and Attribute meta-concepts. We first identified circular entity types and their attributes for TRIMEK and CROMA to be entered into the rEUse platform. This platform allows TRIMEK and CROMA to trace and retrieve information about their circular entities. Workers can directly interact with the rEUse web interface, or they can use the DIA to do so. We created the first prototype of the waste inspector skill that allows users to register an entity in the platform via natural language. Once the DIA knows the entity type, it will ask users for the required attributes. Upon collecting all the required information, DIA sends this information to the rEUse platform via APIs.

In the second use case, assisted workforce management, we created the first prototype of the onboarding DIA for EPISCAN in English and Spanish. The assistant has access to a Neo4j knowledge graph database containing five training processes. These training processes are based on machine manuals and user stories we received from EPISCAN. Each training process contains steps and sub steps. The onboarding skill can track the progress of users, take them to the next corresponding step, or suggest available training processes.

In the third use case, assisted quality assurance for sustainable products, we first defined a digital assistant framework for product quality testing. We then developed the first prototype of the status monitoring skill for REINOVA by deploying and configuring their existing digital twin for product testing. This skill is based on the COALA base stack, uses REST APIs to access sensor data, and has an Analytics module to provide insights about quality testing procedures. By applying the defined digital assistant framework at REINOVA, we provide evidence of its feasibility to other business cases. Additionally, we provide an overview of how the design principles adopted for the REINOVA prototype can be adapted and extended (i.e., with additional skills) to support the other involved business cases, e.g., TRIMEK, COMA, EPISCAN, and SILKBIO.

Tasks 2.2., 2.3, and 2.4 are scheduled to continue till M24 with the second and third phases. During the second phase, our focus will be on enhancing the capabilities of the assistants by close collaboration with the end users and incorporating feedback obtained from users during evaluations. Additionally, findings from the first round of the open-call experiments will be valuable for this phase. The progress and achievements of the second phase will be detailed in D2.4, "Joint WASABI demonstrator – version 2" in M15. In the third and final phase, we will integrate the developed solutions into the end users' infrastructure and finalize the demonstrations. The result of the final phase will be detailed in D2.6, "Joint WASABI demonstrator – final" in M24.