pharaon

PILOTS FOR HEALTHY AND ACTIVE AGEING

Grant Agreement: 857188

D5.2

Pharaon ecosystem - Intermediate release

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No 857188.

Document Information

Deliverable number:	**D5.2**
Deliverable title:	Pharaon ecosystem - Intermediate release
Deliverable version:	1.0
Work Package number:	WP5
Work Package title:	Technology Ecosystem Integration
Due Date of delivery:	30/11/2022
Actual date of delivery:	15/12/2022
Dissemination level:	Public (PU)
Type	Demonstrator (DEM)
Editor(s):	Stefania D'Agostini, Pasquale Vitale, Gabriele Giammatteo (ENG)
Contributor(s):	Stefania D'Agostini, Pasquale Vitale, Gabriele Giammatteo (ENG), Carlos Fernández Sánchez, Monica Alvarez Leon (INDRA), Boris van Schooten, Dennis Hofs (RRD), Miran Mosmondor (ENT), Laura Fiorini (UNIFI). Marike Hendriks (MAIN), Antonio Sánchez (MIWenergía), Jose Luis Alfonso (ICE), Elisabete Pitarma (CDC), Mariana Camacho (SCMA), Ana Perandres (FAL). Mohamad Gharib (UTARTU). Nicolas Mayer (ASC), Telma Dantas (GLINTT) Miguel Ángel Beteta, Sergio Abbenante (CETEM), Jure Lampe (SENLAB), Michael Mrissa (IR CoE), Danny Jansen (ADS), Tarmo Pihl (SENTAB), Guillem Garí (ROB), Antonio Ferritto (CORO)
Reviewer(s):	Jose Luis Alfonso (ICE)
Project name:	Pilots for Healthy and Active Ageing
Project Acronym	PHArA-ON
Project starting date:	01/12/2019
Project duration:	48 months
Rights:	PHArA-ON Consortium

Document history

Version	Date	Beneficiary	Description
0.1	29.07.2022	ENG	Initial Table of Content
0.2	12.10.2022	ENG	Updated Table of Content
0.3	17.10.2022	ENG	Added content in Chapter 3
0.4	21.10.2022	ENG	Added content in Chapter 3 and Chapter 4
0.5	24.10.2022	ENG	Updated content in Chapter 2, Chapter 3 and Chapter 4
0.6	25.10.2022	ENG, INDRA	Updated content in Chapter 3, Chapter 4 and Chapter 5
0.7	26.10.2022	CORO	Updated content in Chapter and Chapter 4
0.8	31.10.2022	UNIFI	Draft content in Section 2.3.2
0.9	02.11.2022	RRD, ENG	Added content in Section 2.1 and Section 2.2
0.10	04.11.2022	CDC	Updated content on Section 3.2.3
0.11	07.11.2022	ENT	Updated IT and SI pilot related content in Chapter 3.1, 3.2 and Chapter 5.
0.12	16.11.2022	ENG	Updated content in Section 2.2
0.13	17.11.2022	MIW, MAIN	Updated content in Section 3.3.22, Section 4.1.22 and provided MISS activity input
0.14	18.11.2022	UNIFI	Updated content in section 2.3.2 related to the emotional goal in the italian pilot
0.15	21.11.2022	ROB	Updated ROB related content
0.16	22.11.2022	SENLAB	Updated SENLAB related content
0.17	23.11.2022	FAL	Updated 2.3.2,Section 4 Andalusian pilot information related to the emotional goal assessment
0.18	23.11.2022	RRD	Added section 4.2

0.19	23.11.2022	CETEM	Update of Amicare sections: 3.3.1 and 4.1.1
0.20	24.11.2022	ADS	Updated info related to the Dutch pilot
0.21	25.11.2022	SENTAB	Added to sections 3.3.19 and 4.1.19
0.22	27.11.2022	UTARTU	Updated section 2.3.2 related to the emotional requirement assessment
0.23	28.11.2022	CETEM	End-to-end testing section
0.24	29.11.2022	ASC	Added sections 3.3.2, 4.1.2 Updated sections 3.2.1, 3.2.2
0.25	30.11.2022	GLINTT	Added section 3.3.4 and 4.1.4
0.26	13.12.2022	IR CoE	Updated section 3.3.5
1.0	15.12.2022	ENG, ICE	Revised and finalized document

PM Efforts per Beneficiary having contributed to the Deliverable

#	Partner	PM effort in D5.2
1	UNIFI	0.7
5	CORO	0.1
7	CETEM	4.0
10	MIWENERGIA	0.2
15	INDRA	3
16	SCMA	0.2
18	CDC	0.2
20	MAIN	8.24
21	RRD	2.9
24	ADS	3.7
25	IRCoE	0.2
28	ENT	4.0
28	ASC	4.0
32	ICE	0.35
36	GLINTT	0.5
37	SENLAB	3.5
38	SENTAB	3.1
42	UTARTU	0.15
45	ENG	8.5
TOTAL	**Pharaon Consortium**	**47,54**

Acknowledgement: This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No 857188.

Executive Summary

The present document details the activities carried out within *WP5 - Technology Ecosystem Integration* and documents the second release of the PHArA-ON software. This deliverable is the second one within WP5 (the previous one is D5.1 - *Pharaon ecosystem - Initial release*) aiming to achieve the integration, release and testing of the PHArA-ON's components developed in WP3 and WP4.

This is a "Demonstrator" deliverable, therefore it is the result of the work done within the three tasks of the WP5. In particular, according to the continuous integration and continuous delivery (CI/CD) and testing approach described in D5.1, this document reports the work performed under *T5.1 - Pharaon release coordination* to set-up and deploy a "*project-wide*" CI/CD infrastructure hosted at ENG premises. This infrastructure can be used by the partners of the project and, optionally, by the new partners from the Open Call 1 (OC1) and Open Call 2 (OC2). Moreover, within this task, a CI/CD implementation approach based on CI/CD pipelines per-technology and per-pilot has been adopted. The deliverable reports the PHArA-ON Intermediate release of software that has been integrated, tested and released in the context of WP5. For each released component, a brief changelog and links to download the source code, artifacts and documentation are provided.

The document also presents the activities carried out under *T5.2 - Pharaon platform integration* and under *T5.3 - Pharaon platform testing* to integrate and validate the PHArAON ecosystem components implemented within the WP3 and WP4. In particular, the document highlights the updates made to the component's interactions and their implementation status (also considering the new interactions due to the introduction of the new technologies from the OC1). Moreover, this document reports the CI/CD pipelines that have been defined and implemented in the context of T5.2 activities and the work carried out within the T5.3 to improve the testing approach providing more details on end-to-end testing and on the evaluation of emotional requirements.

The links and resources providing the evidence of the work reported here can be found within the document.

Progress Beyond the State of the Art and Play

PHArA-ON adopts the DevSecOps principles and guidelines to achieve high quality releases. As highlighted in the previous deliverable D5.1, the main challenges of the WP5 activities are:

- To apply the DevSecOps approach to a large project where are involved several organizations owner of several technologies that need to be integrated in order to achieve the PHArA-ON's users and pilot requirements;
- Build a CI/CD process flexible enough to meet the technologies constraints (e.g., availability of source code and binaries, licenses).

Therefore, the work presented in this document follows the CI/CD and the Quality Assurance (QA) process defined in D5.1 for which also a "hybrid" approach is considered in order to realize CI/CD pipelines using both project-level and private tools, with the objective to left more flexibility to the technical partners.

The tools and technologies reviewed and selected to support WP5 activities have been deployed in a cloud environment dedicated to research projects and hosted at ENG premises. This "project-wide" CI/CD and testing infrastructure has been made available to the partner to perform WP5 activities. Since no project-level solutions were available during the first phases of the project we conducted an assessment to verify the status of realization of the CI/CD pipelines and to collect information on specific constraints related to the involved technologies. Then we used this information to provide some guidelines on how to proceed to implement CI/CD pipelines, i.e., using private tools on premises, the "project-wide" ones or a combination of both options. Moreover, where possible, some partners have planned to migrate their CI/CD and testing pipelines to the project-wide infrastructure.

This document also presents the CI/CD implementation approach based on the definition and realization of CI/CD pipelines *per-technology* and *per-pilot*. The “per-technology” pipeline manages the set-up of build, package, test and deployment phases for a specific PHArA-ON component. The “per-pilot” pipeline, instead, enables the possibility to realize CI/CD across multiple partners (i.e., *federated* CI/CD). In this case the partners are able to deploy other partners’ components on the staging environment, and even perform partial integration tests on their own private build servers using the pilot-wide docker-compose.

Contents

Figures

Tables

Acronyms & Abbreviations

Term	Description
CI	Continuous integration
CD	Continuous Deployment or Continuous Delivery
DevOps	Development and Operations
ERQ	Emotion Regulation Questionnaire
HTTP	HyperText Transfer Protocol
JSON	JavaScript Object Notation
MQTT	Message Queuing Telemetry Transport
MSC	Most Significant Change
OC1	Open Call 1
OC2	Open Call 2
PSS	Perceived Stress Scale
QA	Quality Assurance
REST	REpresentational State Transfer
SUS	System Usability Scale
UCLA Loneliness Scale	University of California Los Angeles - Measure of Loneliness
UEQ	User Experience Questionnaire
VM	Virtual Machine

1 Introduction

1.1 Overview

The main purpose of this deliverable is to outline the Intermediate Release of PHArA-ON ecosystem at M36. The document provides an update with respect to the Initial Release (M26) on the integration and testing activities performed within the context of WP5.

In particular, the document focuses on the progress of tasks T5.1, T5.2 and T5.3. Within T5.1 the CI/CD process and infrastructure has been set-up and deployed, providing a set of "*project-wide*" tools that can be used by the partners to achieve the WP5 tasks. According to the CI/CD process defined in D5.1, within this task has been also defined a CI/CD implementation approach based on the realization of CI/CD pipelines per technology (to automate as much as possible the DevOps phases) and per-pilot (to realize *federated* CI/CD). In order to achieve the user and pilot requirements identified in the deliverables D2.1 and D2.3, the PHArA-ON components and services developed in WP3 and WP4 have been integrated and tested within T5.2 and T5.3. This document provides an update on (i) the interactions between the PHArA-ON components and their implementation status, (ii) on how the components are involved in the pilots, also considering the introduction of the new technologies from the Open Call 1 (OC1), and (iii) it provides a snapshot of the functionalities provided by each individual component in the context of this Intermediate Release. Moreover, this document reports the CI/CD pipelines that have been defined and implemented in the context of T5.2 activities in order to cover the different phases of DevOps approach (i.e., build, package, deploy and test).

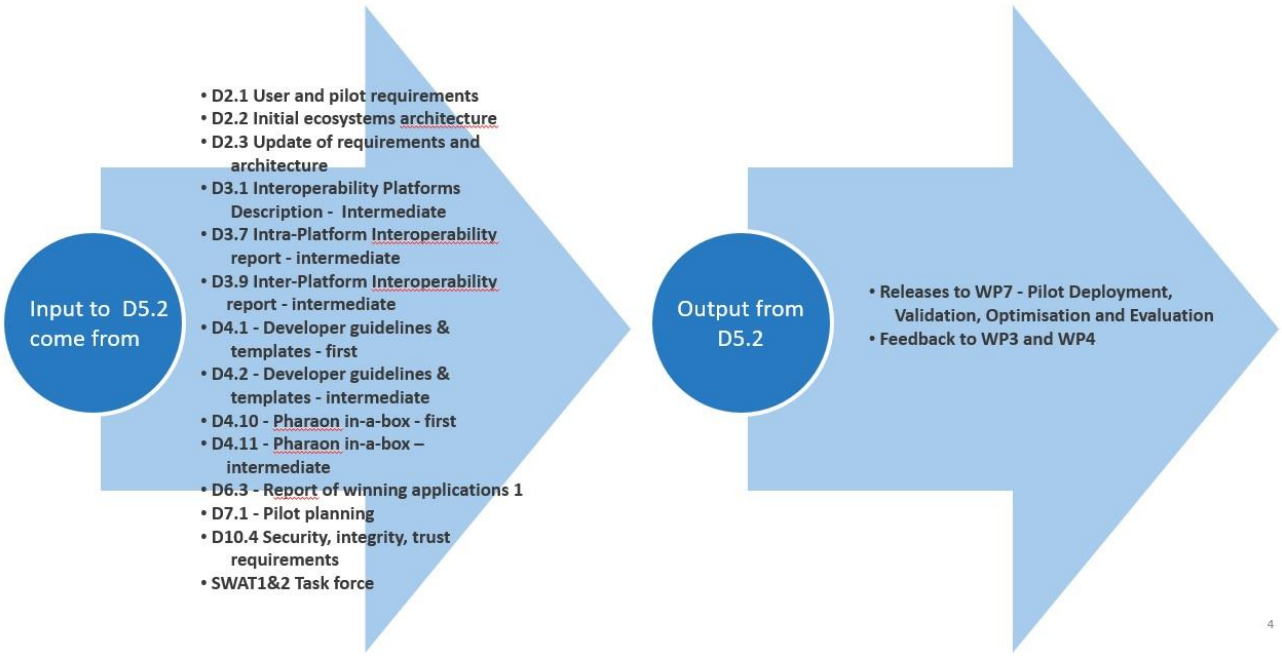
As mentioned in the previous deliverable (D5.1) the testing activities performed within the T5.3 are based on a multi-level testing considering both functional, non-functional, and emotional dimensions, to release high-quality software. Within this task some additional work has been done to improve the testing approach, analyzing how to perform end-to-end testing and how to assess emotional requirements in the pilot context.

Finally, this document reports the staging environment for the Intermediate Release, where the technologies (components and services) are deployed to test their integration and functionalities before the deployment in production, and provides an update of the release plan.

1.2 Relation to other tasks and deliverables

Input to this deliverable comes from WP2, WP3, WP4, WP6, WP7 and WP10 activities. Figure 1.1 reports the deliverables and the main activities that serve as input to the work carried out within WP5 and the output expected from that work.

Figure 1.1: Input and output for D5.2 in relation to other WPs



1.3 Structure of the deliverable

This document describes the Intermediate Release of the PHArA-ON ecosystem, and it is organized in the following sections:

- **Chapter 2** describes how the CI/CD has been implemented, how the *project-wide* CI/CD infrastructure has been set-up and provides some update on the testing approach respect the previous deliverable (i.e., D5.1);
- **Chapter 3** describes the Intermediate Release of the PHArA-ON ecosystem at M36 providing an update of the interactions implemented between the PHARA-ON components and on how they are involved in the pilots. Moreover, the preliminary work for the integration of the new technologies from the OC1 is highlighted;
- **Chapter 4** reports the CI/CD pipelines that have been defined and/or implemented to cover the DevOps phases and the performed testing activities;
- **Chapter 5** provides an update (from D5.1) of the staging environment adopted by the pilots to deploy the PHArA-ON technologies (e.g., components, services, equipment) for test purposes;
- **Chapter 6** provides an update of the release plan with respect to the one reported in the previous deliverable D5.1. This update takes into consideration the pilot and open calls plans.
- **Chapter 7** draws the conclusions and the next steps.

2 Release process and environment

This section describes (i) how the CI/CD approach described in the D5.1 has been implemented (section 2.1), (ii) how the *project-wide* CI/CD infrastructure has been set-up and deployed and (iii) provides updates on the testing approach (section 2.3) adopted in the project.

2.1 CI/CD Implementation approach

In our CI/CD approach, we left up to the partners what CI/CD tools they use, providing the needed flexibility for diverse technical partners. Nevertheless, we provided recommendations and tools to make it easier for partners to get started with CI/CD, and provide a unified approach where needed. We chose two technologies as our primary technologies: **Jenkins** for CI/CD and **Docker** for virtualization. Alternative technologies are, for example, Github CI/CD and virtual machines.

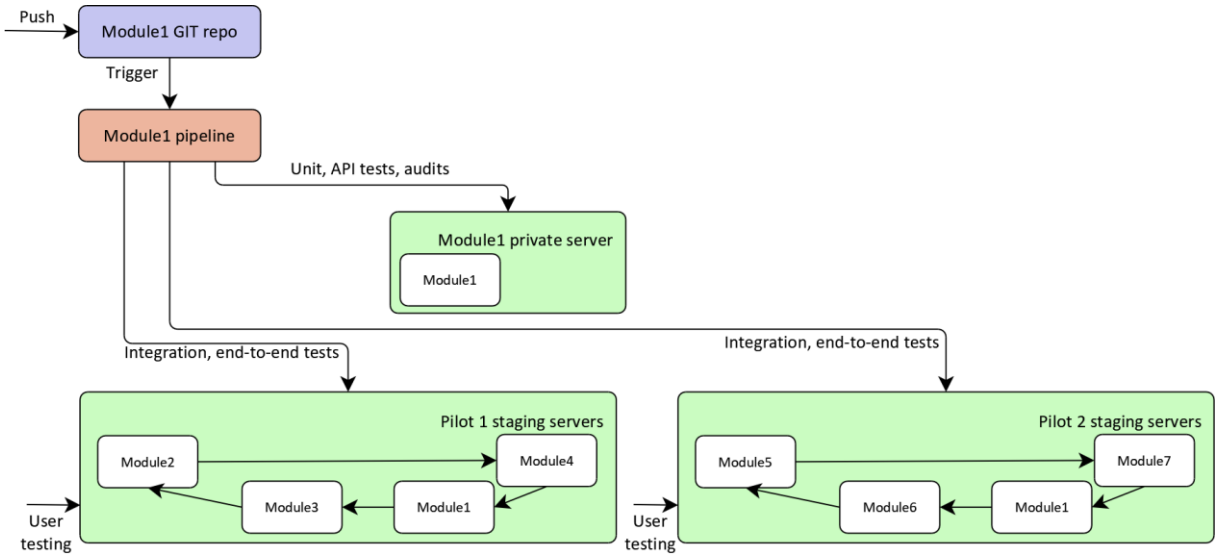
Each technical partner typically has one or more CI/CD pipelines. These can be run on the partner's private servers, but we also provided a project-wide Jenkins server and a project-wide binary repository for Docker images and other binaries.

We considered different build environments. The *production environment* is the set of integrated components that together form a PHArA-ON pilot, as is used in production. The *staging environment* also contains the full set of integrated components, but is used for integration testing before the components go into production. Components also typically have a private build server where single-component tests can be performed. In PHArA-ON, the production environments are managed within WP7 and are considered "frozen" during the evaluation periods (updates are avoided as much as possible). Currently CI/CD concerns the staging environments fthat, on the contrary, are managed in WP5 and can be updated at any time for testing. Once the evaluations are done, the same processes and CI/CD tools used in WP5 can be applied for the production environments as well.

Distinction between staging and production can be done using particular tags or branches in the source repository. For example, a "master" branch can be used for staging, and "stable" for production. A similar tagging scheme can be used for Docker images or other binary artefacts.

The basic setup for a technical component is to have one CI/CD pipeline that builds, tests, and deploys each component. As shown in Figure 2.1 we have a component called *Module1*. A build is typically triggered by a Git repository operation, like pushing an update. Then, the CI/CD pipeline can build Module1, and perform tests on its own private build server. If successful, it can then be deployed to the staging environment, and integration and pre-validation testing can be performed. Module1 may feature in multiple pilots, in which case it is deployed to all pilots, with a specific configuration for each pilot.

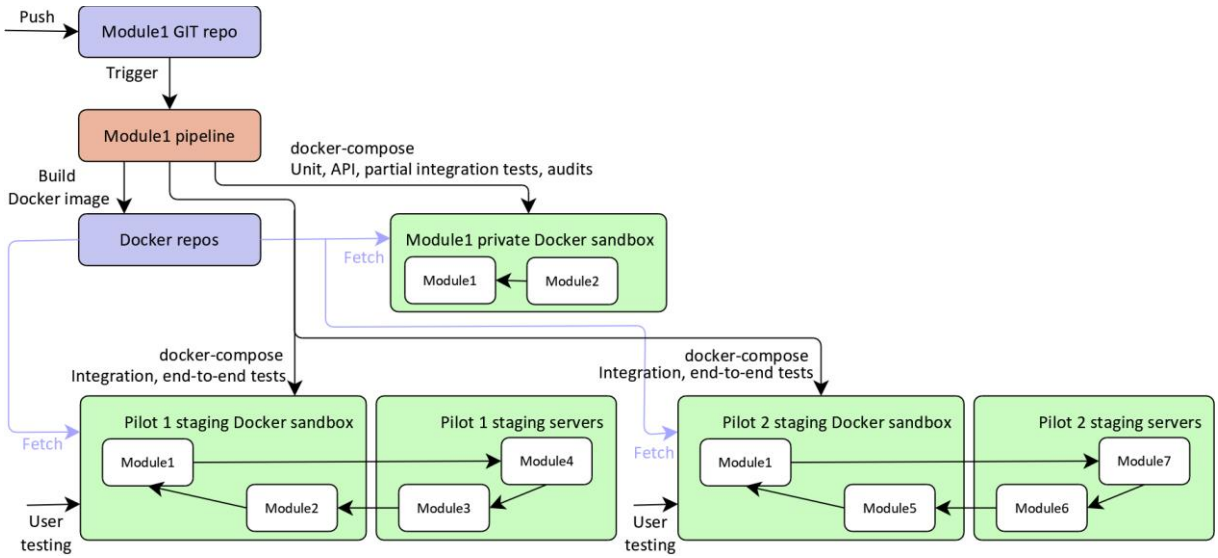
Figure 2.1: Basic CI/CD setup



The staging environment consists of multiple servers, maintained by the different partners. Additionally, it can contain a dedicated Docker server, which can deploy some or all "Dockerized" components using a pilot-wide Docker configuration file (docker-compose.yml). In this scenario, partners are able to deploy other partners' components on the staging environment, and even perform partial integration tests on their own private build servers using the pilot-wide docker-compose. We call this *federated CI/CD*: CI/CD across multiple partners. A prerequisite is that partners are willing to share binaries with the project. The pilot-wide docker-compose will probably have to be maintained by a pilot technical manager to ensure continuity and consistency.

The basic CI/CD setup for "Dockerizable" components with Docker-based federated CI/CD is shown in Figure 2.2. The difference with the non-Docker setup is that the build results in a docker image that is stored in a Docker repo. Deploy is then performed by invoking docker-compose, which will fetch the Docker images from the repo.

Figure 2.2: Basic CI/CD setup for Dockerizable components

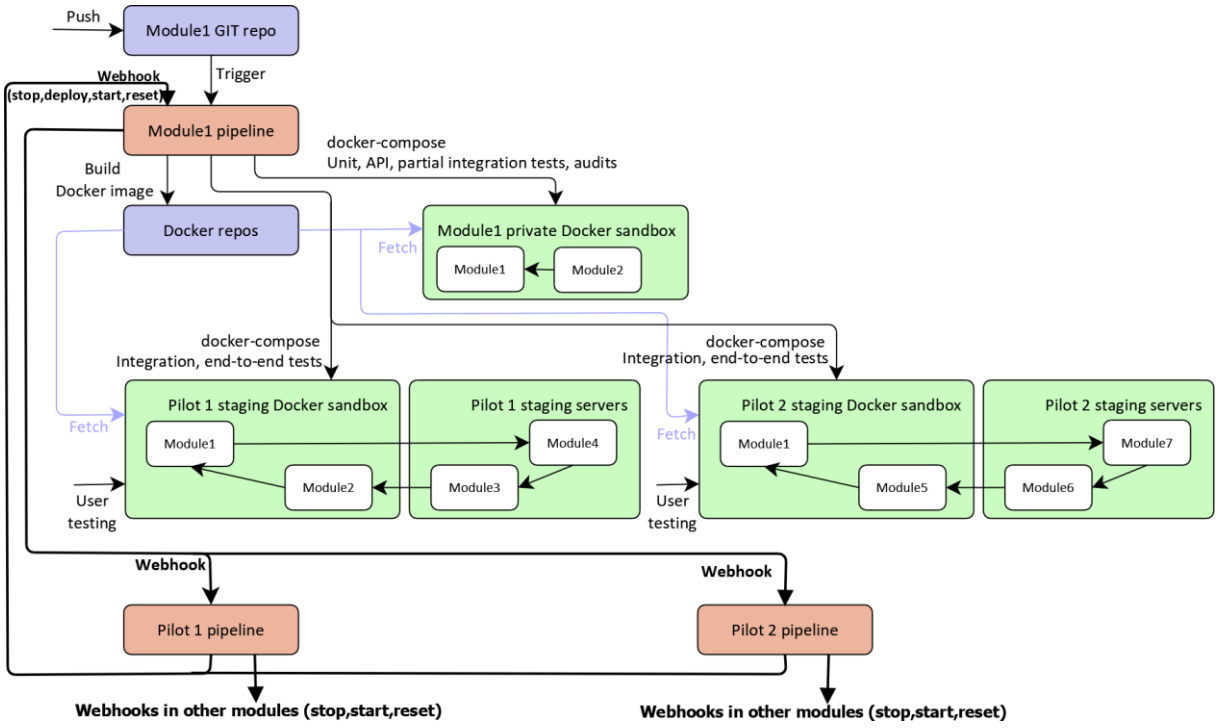


Theoretically, if all components are Dockerizable, the entire pilot could be deployed using docker-compose. Most partners do not have Dockerizable components, though. In some cases, federated CI/CD is still desirable, in particular:

- to stop and restart partner components that may malfunction when deploying a component while they are running;
- to be able to restart critical partner components that are prone to crashing;
- to reset the pilot to a reliable start state for automated testing.

Therefore, we offer an alternative approach to federated CI/CD, using a pilot-wide pipeline. This pipeline coordinates the deploy process for all relevant components. This approach is illustrated in Figure 2.3, which shows the case in which the component is Dockerized. The non-Dockerized case is similar.

Figure 2.3: Federated CI/CD using pilot-wide pipelines



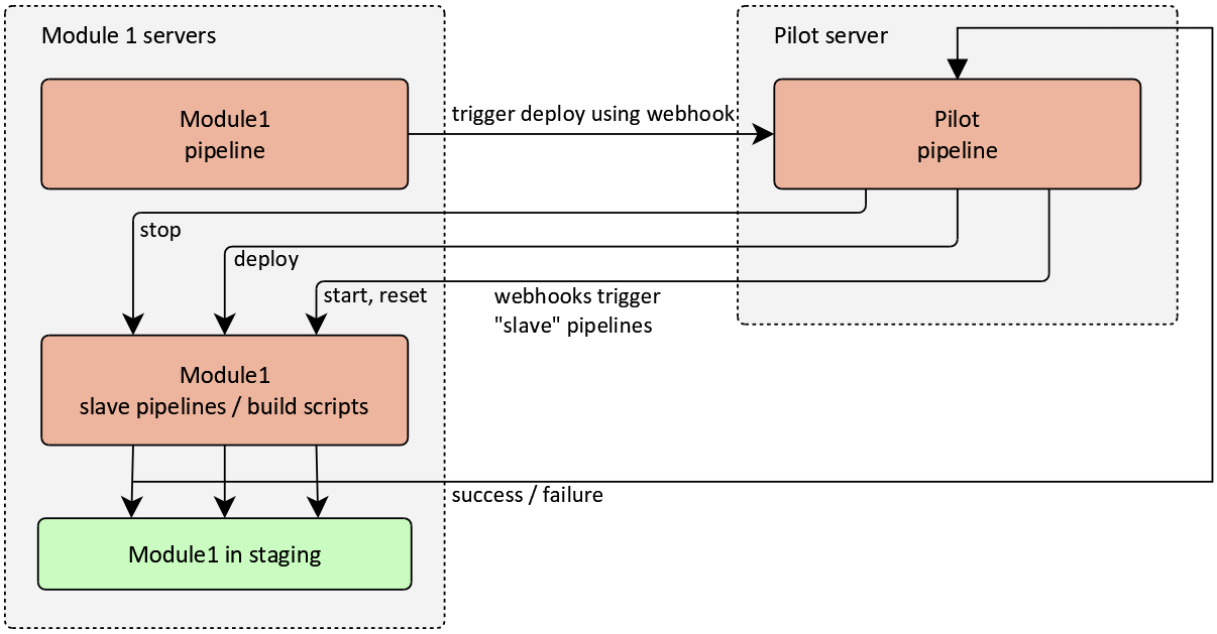
In the pilot pipeline approach, each component pipeline will build and perform private tests the same as in the basic case. Once the tests pass, it calls the pilot pipeline to start deployment on the staging environment. The pilot pipeline then calls each component to stop, then calls back the component to indicate that the environment is ready to deploy the component, then restarts each component. This way, the entire pilot can remain in a valid state while one component is being deployed. After that, the component pipeline can perform integration tests as needed.

This approach leaves all details on *what* should be done to the individual component pipelines. The pilot pipeline only determines *when* particular pipeline stages should be done. This approach retains maximum control over the CI/CD process by the technical partners, while the pilot pipeline needs little to no changes if something changes in the CI/CD process. Also, the partners need only to provide third-party access to webhooks on their servers to make this work.

Communication between the pipelines is done via webhooks. The most straightforward implementation is by splitting the component pipeline into multiple sub-pipelines for the stop, start, reset, and deploy stages. This can be readily implemented using the Jenkins webhook and webhook-step plugins, together with parallel Jenkins pipelines. Alternatively, if specified properly, stop, start, reset, and deploy can be optional stages in a single pipeline, with the pilot pipeline telling which stage(s) to execute via a parameter. This can be done by adding conditions to the appropriate Jenkins pipeline steps, and allowing at least two instances of the pipeline to run, since the component pipeline is called via a callback while waiting for the pilot pipeline to finish, and needs to run as a second instance in that case.

Data flow between the component servers and the pilot pipeline is illustrated in more detail in Figure 2.4. Consider a component called Module1. If a build is triggered, it builds, then triggers a deploy on the pilot pipeline, which will call sub-pipelines at appropriate times for each component. It will call stop for all components, then deploy only for Module1, then start for all modules. So, Module1 eventually gets callbacks from the pilot pipeline to its sub-pipelines to perform certain steps before its main pipeline is finished. Note that any action on a sub-pipeline can result in success or failure. The result is sent back to the pilot pipeline via webhook parameters. Note that any step can fail, which should result in an error report. We recommend that the system will not abort the operation altogether in case of error, and instead should try to restart all modules.

Figure 2.4: Data flow between component servers and pilot pipeline

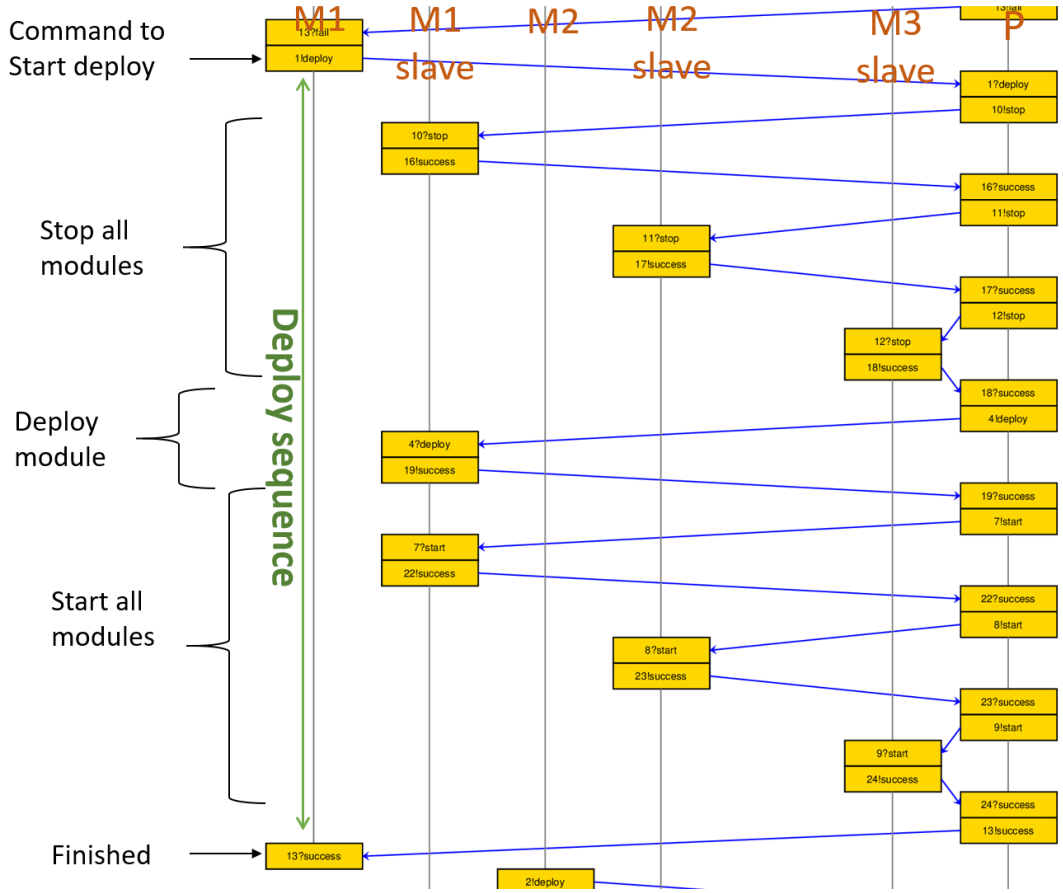


A detailed sequence diagram of a deploy sequence is given in Figure 2.5. Note that this is output from the iSpin[1] verification tool, which we used to specify and verify our solution in the early stages of development. This example shows a system with three federated components M1, M2, and M3. Component M1 starts its deployment, after which all components are stopped (“stop all modules” sequence), then M1 gets the signal to deploy, then all components are started again (“start all

[1] http://spinroot.com/spin/Man/3_SpinGUI.html

modules" sequence). Note that the steps are executed sequentially for simplicity. The stop and start phases could be parallelized for efficiency.

Figure 2.5: Sequence diagram of deploy sequence



Federated CI/CD with the pilot pipeline is not mandatory for component developers, but rather, an option if they need it for deployment. For example, in many cases it is acceptable that a component is deployed while others are running. There are three levels of participation: no participation (private pipelines only), passive participation (component pipelines do not trigger the pilot pipeline, but can be called by the pilot pipeline), and full participation (they can trigger and be called by the pilot pipeline).

In the context of T5.2, we conducted an assessment to verify, for each technology provider, the status of realization of the CI/CD pipelines (i.e., if the CI/CD was already in place or not) and to collect information on specific constraints related to the involved technologies (i.e., source code and binaries available or not, components dockerizable or not etc.). Then we used this information to provide some guidelines on how to proceed to implement CI/CD pipelines. These guidelines are reported in Figure 2.6.

Figure 2.6: Guidelines to implement CI/CD pipelines

CI/CD Avail.	Code Avail.	Binary Avail.	Actions
No	No	No	Create CI/CD with private tools
		Yes	Create CI/CD with private tools Create binary checks + tests + CD with project tools
	Yes	No	--
		Yes	Create CI/CD with project tools
Yes	No	No	Report CI/CD activities and results in D5.2
		Yes	Create binary checks + tests + CD with project tools
	Yes	No	--
		Yes	Migrate existing pipelines to project tools

According to the described approach, Chapter 4 reports the CI/CD *component pipelines* defined and implemented by each technology provider using privates and/or the project-wide tools and also an example of *pilot pipeline* (implemented for the Ditch pilot).

2.2 CI/CD Infrastructure

The project-wide CI/CD infrastructure is hosted at ENG premises in a cloud environment dedicated to research projects in one of the company's data-centers. The infrastructure is currently composed of five virtual machines with a total of 34 vCPUs, 160GB of RAM, 136GB of internal disks and 350GB of network storage. Additional computational and storage resources can be added at any time if needed. At infrastructure level, regular backups of the storage volumes are performed and network traffic is secured through a firewall and regular security vulnerability scans.

On top of the VMs, a Kubernetes cluster (v. 1.21) has been deployed to manage the resources and all the software installed in the infrastructure. Kubernetes ensures a higher level of manageability of the entire infrastructure improving at the same time also the availability, scalability, fault tolerance and load balancing of the services hosted.

The services currently deployed on top of the Kubernetes cluster and offered to the PHArA-ON project for the activities in WP5 are (Figure 2.7):

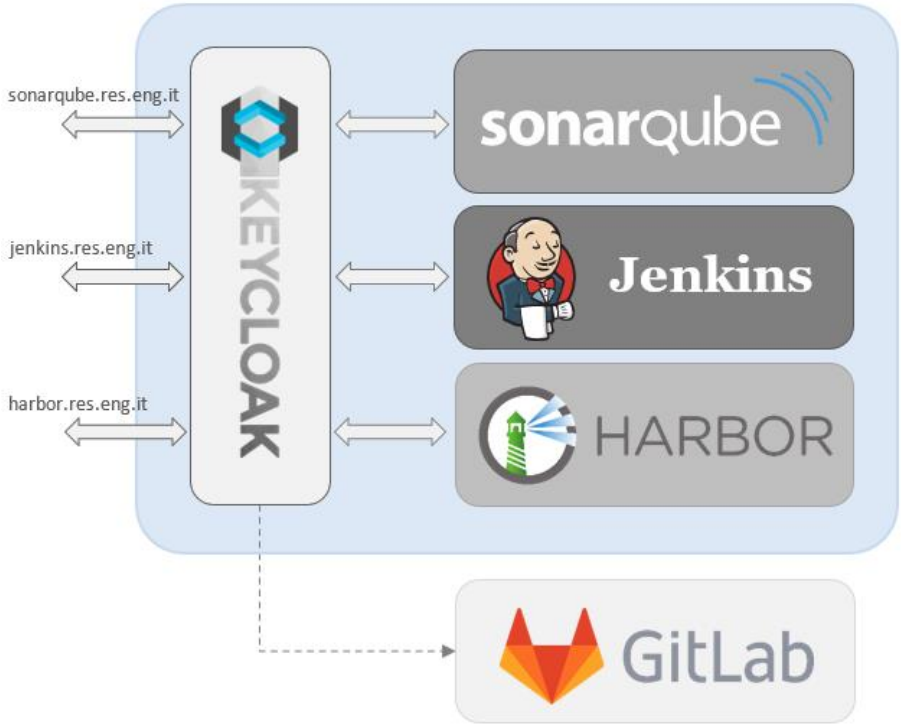
- **Jenkins** - the CI/CD server at link https://jenkins.res.eng.it
- **Sonarqube** - the code analysis and detect vulnerability at link https://sonarqube.res.eng.it
- **Harbor** - the registry of docker images at link https://harbor.res.eng.it

In addition to these user-facing services, other services are deployed for the maintenance and to guarantee the functionality of the cluster:

- **Keycloak** - to provide centralized user management and single-sign-on in all the services. The instance is available at https://idp.res.eng.it
- **Velero** - to backup daily all the application's data on an external storage
- **Postfix and Dovecot** - to allow applications to send and receive emails
- **Prometheus and Grafana** - to collect and visualize monitoring data from the applications
- **Crowdsec** - to monitor the network traffic and block potential attacks and malicious requests

The next subsections describe more in detail the main tools that will be used by the PHArA-ON partners to define and implement CI/CD pipelines using the project-wide infrastructure.

Figure 2.7: Project-wide CI/CD Infrastructure



2.2.1 Keycloak

Keycloak represents the **central identity provider** for all services in the PHArA-ON ecosystem. Any external client which wants to access services (Sonarqube, Jenkins, or Harbor) within the infrastructure needs to perform login through Keycloak. The login in the infrastructure is federated, i.e., if you try to access these services, you will be redirected to the login page (keycloak) and after authentication is successful, you will be back to the service.

The login authentication (keycloak) has been configured to provide **two authentication ways**:

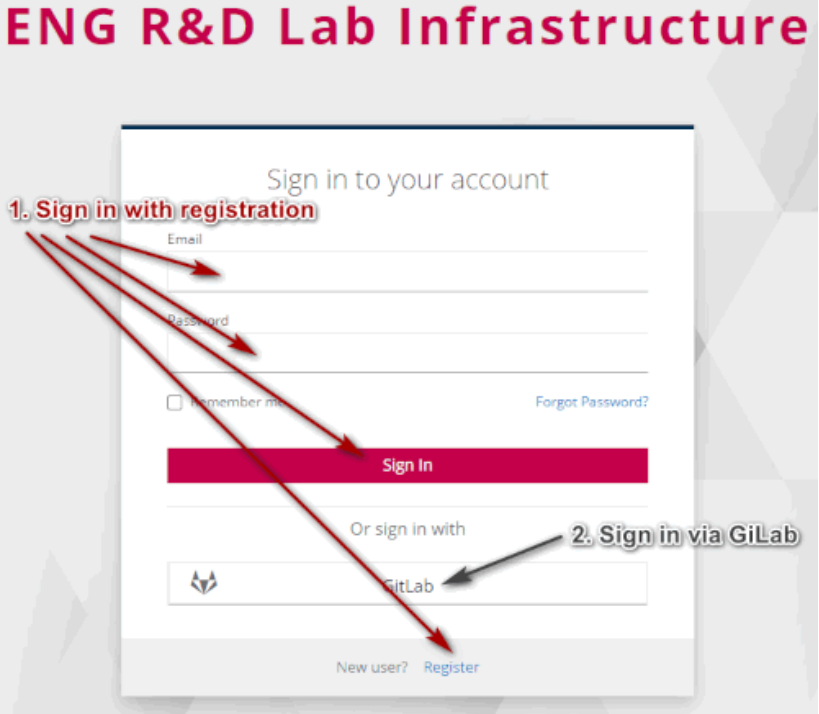
1. through user registration and login,
2. or login via GitLab credentials.

The first login way includes a preliminary registration; the user has to fill in some fields (i.e., email, password, first name, last name, and so on) to save user credentials in the internal database (i.e., Postgres). After registration, users can log in by using their own credentials and they can use them for the next login.

The second login way uses GitLab credentials (i.e., login with GitLab). This feature is provided by the OAuth protocol configured in the Keycloak.

Figure 2.8 shows these two authentication ways. The interface allows you to sign in with username and password (after registration) or sign in via GitLab: in red **sign in with registration** and in black s**ign in via GitLab**.

Figure 2.8: Two way of authentications



After the login, the username must be provided to the ENG team via email (at support@res.eng.it) to authorize the user access to the server. This is a required step for security and to avoid not authorized user accesses

2.2.2 Jenkins

Jenkins is the engine adopted in the infrastructure to perform the **CI/CD processes**. Basically, the most important Jenkins usage in the PHArA-ON infrastructure consists to create **Jenkins pipelines**; any pipelines can do particular tasks (i.e., checkout of code, build of code, scanning of code, testing of code, and so on) to be included in the CI/CD process.

Jenkins also provides a set of plugins to add other functionalities, i.e., **sonar** (to scan the code), **JUnit** (to test source code), **performance** (to display JMeter results like throughput, response time, percentage of errors), **email** (to send email). These features are already included in the server. If it is

missing a feature then it can be installed easier through a helm chart after restarting the infrastructure without losing any data.

To help users in the creation of the pipeline, a list of pipeline templates (i.e., script) has been provided in the internal wiki[2]. This list provides a set of (generic) use cases which can be used as Jenkins stages in order to create a composed CI/CD pipeline.

In general, a **pipeline script** consists of three sections:

- ***Agent***: a list of docker images you need to use in the stage section (if necessary);
- ***Stages***: a list of tasks you need to perform in the pipeline; within the stages can be inserted one of multiple stage (according with tasks you have to perform);
- ***Post actions*** (optional): to do something at the end of the pipeline (i.e., send an email for notification about the success or the failure of pipeline)

Listing 1 shows a **pipeline script** with the agent, stages and post actions sections.

Listing 1: Example of pipeline script

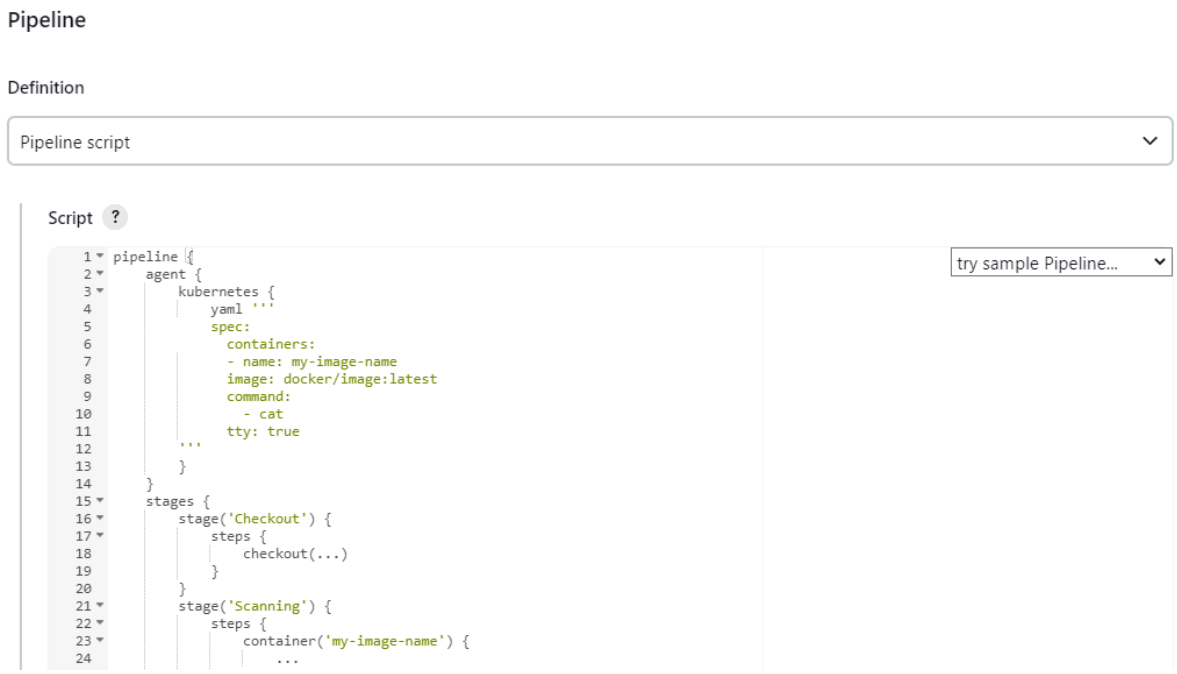
```
pipeline {
    agent {
        kubernetes {
            yaml '''
                  spec:
                    containers:
                        - name: my-image-name
                        image: docker/image:latest
                        command:
                          - cat
                        tty: true
               '''
        }
    }
    stages {
        stage('Checkout') {
            steps {
                checkout(...)
            }
        }
        stage('Scanning') {
            steps {
                container('my-image-name') {
                    ...
                }
            }
        }
    }
    post {
        success {
            script {
                emailext subject: "SUCCESSFUL",
                recipientProviders: [[$class:'RequesterRecipientProvider']],
                body: "Pipeline - successful."
            }
        }
        failure {
            script {
```

[2] https://gitlab.com/pharaongroup/developers-handbook/-/wikis/Getting-started/Jenkins-CI

```
                emailext subject: "FAILED",
                recipientProviders: [[$class: 'RequesterRecipientProvider']],
                body: "Pipeline - failed."
            }
        }
    }
}
```

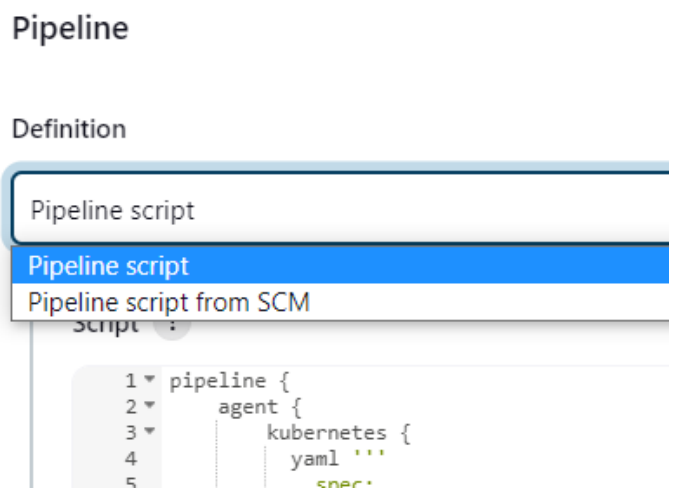
This script must be used in the Jenkins configuration of the pipeline as **Pipeline script** as shown in the Figure 2.9.

Figure 2.9: Pipeline script



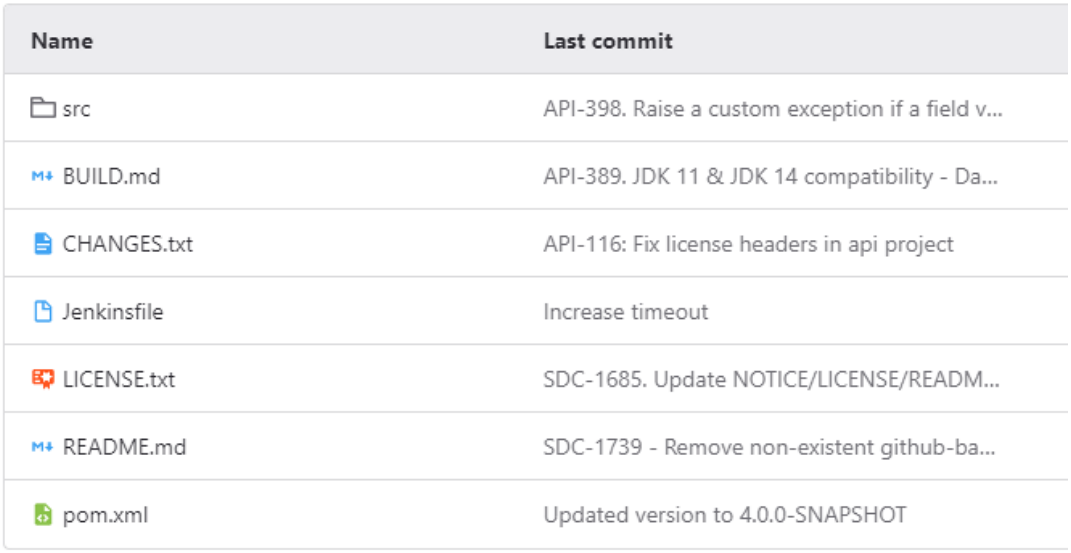
Moreover, there is another way to execute a Jenkins pipeline; instead of using the Pipeline script, it's possible to select **Pipeline script from SCM** as shown in Figure 2.10. In this case, Jenkins will look for the script in the repository providing the **endpoint** and **credentials** (if necessary) of the gitlab repository.

Figure 2.10: Pipeline script from SCM



The Jenkins server must find in the repository a Jenkinsfile file in the root of the source code which includes the same content as the Pipeline script. Figure 2.11 shows where the Jenkinsfile file is located.

Figure 2.11: Jenkinsfile in the source code



The two options above give free choice to developers on how to build the pipelines:

- In the first case, developers must create or update the pipeline in the Jenkins server; any changes must be performed using Jenkins login
- In the second case, the pipeline configuration is included directly in the repository through the Jenkinsfile file and all developers can edit the pipeline simply by editing this file

For both these two types of configurations, the developer can also start the pipeline after any commit in the repository (**auto build** of the pipeline). This is an important feature in a CI/CD process and it can be used, for example, to start the build process, to make available the artifacts, and to scan results after the developers merge the source code into the master branch. To do this, it's necessary to

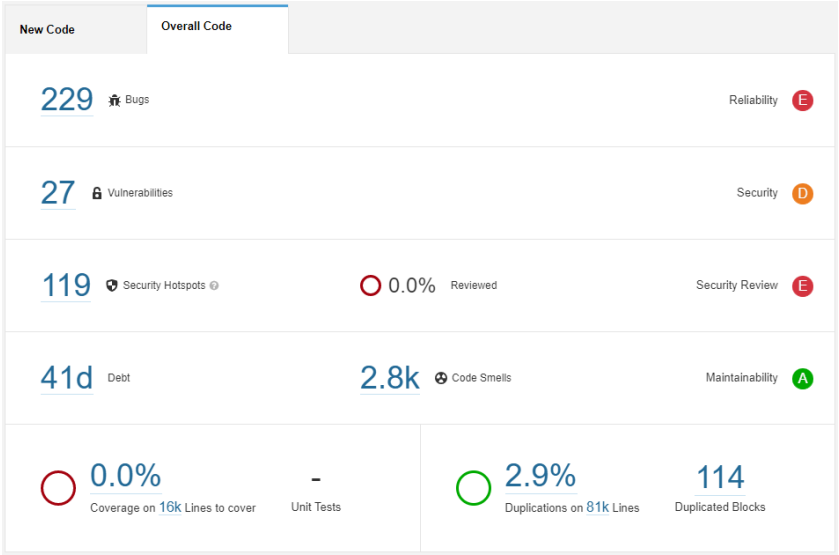
configure Jenkins and the repository (i.e., GitLab). An example of Jenkins and Gitlab configuration can be found in the *auto-build* section of the internal wiki[3].

2.2.3 Sonarqube

Sonarqube is used in the architecture for continuous inspection of code quality and vulnerabilities. The sonarqube server is used to display all results provided by scanning tools (i.e., Jenkins scanning stages provided by CI/CD pipeline - see section 2.2.2).

Sonarqube is already configured with Jenkins to communicate with each other. A **Sonarqube scanning code** is available in the internal wiki[4] to scan the source code. Figure 2.12 shows an example of the scanning analysis. These results represent feedback to improve the code removing potential bugs or issues.

Figure 2.12: Sonarqube interface



2.2.4 Harbor

Harbor is a secure repository of artifacts and used in the PHArA-ON infrastructure to store docker images similarly to **Docker Hub**.

Harbor is already configured with Jenkins for pushing the docker image. A **Push in Harbor** script is available in the internal wiki[5] to create and push a docker image in Harbor.

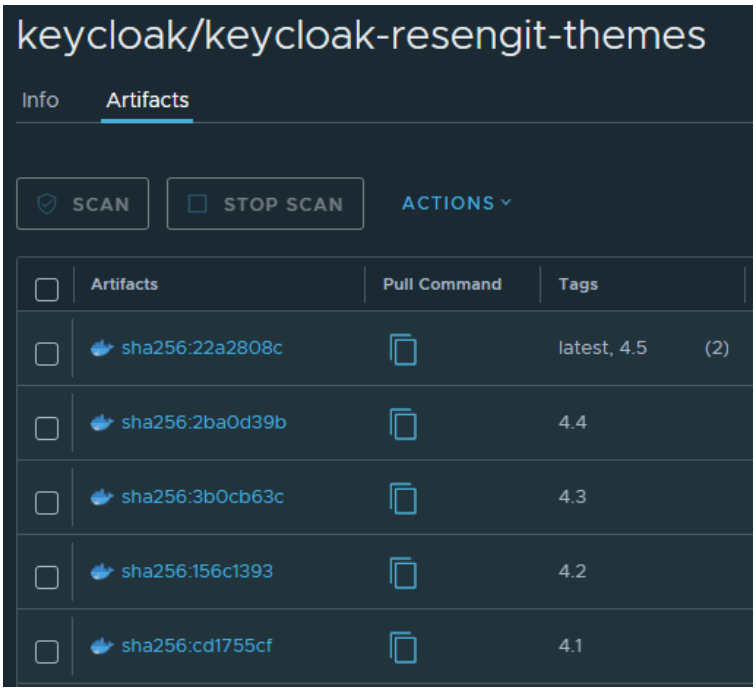
Figure 2.13 shows a list of artifacts (docker images) identified by a tag. The docker commands (i.e., `docker pull <image-name>` and `docker run <image-name>`) can be used to download the image from the repository and run it on a local machine.

[3] https://gitlab.com/pharaongroup/developers-handbook/-/wikis/Getting-started/Jenkins-CI#auto-build-to-any-change-code-on-gitlab

[4] https://gitlab.com/pharaongroup/developers-handbook/-/wikis/Getting-started/Jenkins-CI#sonarqube-scanning-code

[5] https://gitlab.com/pharaongroup/developers-handbook/-/wikis/Getting-started/Jenkins-CI#push-in-harbor-via-jenkinsfile

Figure 2.13: Harbor view



2.3 Testing Approach (Update)

The project-wide testing and Quality Assurance (QA) processes adopted in the PHArA-ON project has been presented in D5.1. The testing activities are based on multi-level testing considering both functional, non-functional, and emotional dimensions.

This section updates the approach defined in the previous deliverable giving more details on how to perform end-to-end testing (section 2.3.1) and how to assess emotional requirements (section 2.3.2).

2.3.1 End-to-end testing

The **end-to-end** (e2e) testing is a technique that tests the entire software product from beginning to end to ensure the application flow behaves as expected. In particular:

- It defines the product's system dependencies and ensures all integrated pieces work together as expected
- The main purpose of e2e testing is to test from the end user's experience by simulating the real user scenario and validating the system under test and its components for integration and data integrity

The e2e testing process follows the structure depicted in Figure 2.14 and has been taken into consideration by the pilots.

Figure 2.14: Testing process

To realize e2e testing, several tools have been analyzed (also considering the ones selected in the context of WP4, in particular under the section "Tools for every phase of SDLC"[6] of the PHArA-ON Developers Handbook) in order to verify where these tools can be used and which one better fits the pilot's needs.

Table 2.1 briefly presents these e2e testing technologies and reports in which pilots have been used to perform some preliminary e2e testing activities.

Table 2.1: End-to-end testing technologies

Technology	Description	Pilot
Selenium IDE	An end-to-end testing framework which is available in many popular programming languages including Java, Python, JavaScript, among others. It provides a set of tools that allows you to connect with an automated browser that interacts with the frontend.	Murcia pilot
Cypress	A purely JavaScript-based front-end testing tool built for the modern web. It aims to address the main points developers or QA engineers face while testing an application. Cypress is a more developer-friendly tool that uses a unique DOM manipulation technique and operates directly in the browser.	Slovenian and Italian pilots
Cucumber	This tool reads executable specifications written in plain text and validates that the software does what those specifications say. The specifications consist of multiple examples, or scenarios.	None
JMeter	The Apache JMeter application is open-source software, a 100% pure Java application designed to load test functional behavior and measure performance. It was originally designed for testing Web Applications but has since expanded to other test functions.	Slovenian and Italian pilots
Postman	Postman is an API (application programming interface) development tool which helps to build, test and modify APIs. Almost any functionality that could be needed by any developer is encapsulated in this tool.	Murcia, Slovenian and Italian pilots
Minsait Cilantrum	Minsait Cilantrum performs the test procedures with reduction of time and effort in the execution of the tests. Support in	Andalusian pilot

[6] https://gitlab.com/pharaongroup/developers-handbook/-/wikis/home#tools-for-every-phase-of-sdlc

	increasing the performance of test passes. It facilitates the detection of errors more quickly and makes it easy to implement continuous integration.	

During the end-to-end testing carried out by the Murcia pilot the **Selenium IDE** technology was applied as well as the **Postman** API Client for authorization requests in API REST. In addition, the pilot from Italy and Slovenia used **JMeter** technology for REST API and MQTT testing, as well as **Postman** API Client in some complex cases for REST API. In some applications **Cypress** was also used for UI testing. Regarding the Andalusian pilot, the **Minsait Cilantrum** (ad-hoc) tool developed by INDRA was used.

The Portuguese pilot carried out end to end testing tasks **manually**, due to timelines, resources and the fact that the system is already in the field (also in commercial use). In the future, the Portuguese pilot planned to use the Selenium IDE tool for tmhis kind of testing.

Also, for the Dutch pilot at the moment none of the tools described above are used for e2e testing (for the time being a manual step is included in the final testing phase). In the next period will be evaluated which of the presented e2e tools better fit the Dutch pilot needs.

2.3.2 Assessment of emotional requirements

This section describes the methods to be followed by each pilot to assess the satisfaction of their emotional requirements that have been identified in D2.1 and updated in D2.3.

1. **Assessment of emotional requirements in the Italian pilot**

Methodology

In addition to the KPI described in T7.4 - “Impact assessment”, the Italian pilot will rely also on the peculiar evaluation framework that will assess and investigate if the emotional requirements will be achieved or not.

Starting from the analysis of D2.1, we identified the list of the emotional goals of the Italian pilot. Then, we identified the quantitative and qualitative tools that will collect feedback on specific aspects of the identified emotional goal. As for the quantitative evaluation we decided to rely on domains assessed by the general PHArA-ON and the peculiar evaluation framework and on ad hoc questionnaires developed starting from the output of D2.1. Additionally, a focus group will be organized to evaluate the emotional goal with a representative of the target group. All these evaluations will be done at M6 and M12 (i.e., six and twelve months later the pilot started).

Description of the tools

Quantitative evaluation

- Standard questionnaire
- Ad-hoc questionnaire

Qualitative evaluation

- Focus group with a representative of the target group

Standard questionnaires

According to the Italian pilot evaluation framework, we will measure some domains that will give us hints of the emotional goal defined since they can be correlated with the emotional goals. Table 2.2 reports the link between the emotional goal and the domains of the peculiar Italian evaluation framework.

Table 2.2: Emotional Goal Domain - Italian Pilot

Emotion Goal Domain	Standard tool used that can be correlated with the emotional goal and it belongs to Italian Evaluation framework
Safe	Item 9 SUS scale
Stress	ANXIETY Domain from Almere model Questionnaire
In contact/Involved	UCLA Loneliness Scale
Confident/Conscious	Dependability and perspicuity domains from UEQ questionnaire

Ad-hoc questionnaire description

In order to assess/evaluate the identified emotional goal - and not a related domain - the Italian pilot will use a customized tool that was built on the results of D2.1. The facilitators at the two pilots' site revise the sentences, thus to be sure the correct understanding of the questions from PHArA-ON participants. Table A.1 in the Appendix A reports the ad-hoc questionnaire (English version). This ad-hoc questionnaire was presented to the other pilot leaders and the Portuguese pilot agreed to use it in their pilot sites to reinforce the impact of the results.

Focus group

In order to better understand the insight of the emotional goals, a dedicated focus group will be organized at pilot sites level (Tuscany and Apulia) to verify how the emotional dimension was perceived during the tests. The quantitative data can be used to promote the discussion among participants.

2. Assessment of emotional requirements in the Murcia pilot

Methodology

We started by analyzing the list of the emotional goals identified in the D2.1 - Murcia pilot, to determine the best method for assessing their satisfaction, i.e., whether a quantitative and/or qualitative method could be appropriate for collecting feedback from end-users to allow the aforementioned assessment. Accordingly, we have decided to use a quantitative tool relying on a questionnaire designed assessing emotional requirements of the Murcia pilot, along with some qualitative data perceived by the staff involved in the pilot.

Description of the tools

The emotional requirements questionnaire (available in Table A.2 in the Appendix A) was designed to collect feedback from end-users aiming at assessing the emotional requirements. We aim at collecting 10 questionnaires of Older Adults participating in the pilot.

3. Assessment of emotional requirements in the Portuguese pilot

Methodology

The Portuguese pilot is involved in 4 PHArA-ON Challenges (PCH4, PCH6, SCH7, and PCH8) that led to the scenarios and their specific interventions. Six services were created using four different technologies, with two additional services and technologies incoming from the OC1.

The emotional goals presented are directly related to the scenarios, technologies, and services already being implemented in the pilot, as well as the ones from the OC1.

First, the list of the emotional goals identified in the D2.1 - Portuguese Pilot was analyzed to determine the best method (whether quantitative and/or qualitative) for collecting feedback from end-users to allow the assessment.

After screening these emotional goals, it was decided to use qualitative and quantitative methods, such as standard questionnaires, ad-hoc questionnaires, and interviews at M0, M6 and M12 (i.e., at the start time of the pilot and after six and twelve months).

Description of the tools

- Quantitative evaluation
  - Standard questionnaire
  - Ad-hoc questionnaire
- Qualitative evaluation
  - Interviews

Standard questionnaires

From the standard questionnaires used in the Portuguese Evaluation framework, it will be possible to measure a group of emotional goals according to the table below.

Table 2.3: Emotional Goal Domain - Portuguese Pilot

Emotion Goal Domain	Standard tool used that can be correlated with the emotional goal and it belongs to Italian Evaluation framework
Safe	SUS scale[MC1]
Connected, Engaged, Enriched, Integrated, Stimulated	MSC (Most Significant Change)
Stress	PSS-10 (Perceived Stress Scale[MC2] [EP3])
Stress/Trust	ANXIETY Domain from Almere model Questionnaire
In contact/Involved	UCLA Loneliness Scale

Ad-hoc questionnaires

In the Portuguese Pilot, the Ad-hoc questionnaires (available in Table A.3 in the Appendix A) will be performed with Older Adults, Informal, and Formal Caregivers.

The questionnaires for Older Adults and Informal Caregivers were presented by the Italian pilot, and, after analyzing the goals and the method, the Portuguese pilot decided to use it as well in order to reinforce the impact of the results. However, considering the Italian and the Portuguese pilots have different emotional goals to assess, some inputs and slight adaptations were added to the Portuguese version.

Based on this first questionnaire and according to the same domains, a questionnaire was also created for formal caregivers to assure the assessment of the emotional goals for all the intervention groups.

4. Assessment of emotional requirements in the Andalusian pilot

Methodology

We started by analyzing the list of the emotional goals identified in the D2.1 - Andalusian pilot, to determine the best method for assessing their satisfaction, i.e., whether a quantitative and/or qualitative method could be appropriate for collecting feedback from end-users to allow the aforementioned assessment.

At the beginning of the pilot deployment, we decided to include standardized questionnaires about emotion regulation (*Emotion Regulation Questionnaire,* Spanish version (ERQ)). This instrument assesses the dispositional use of two emotion regulation strategies: reappraisal and emotional suppression. This questionnaire has shown adequate validity and internal consistency (Cronbach's α=0.75; 0.71, respectively). In this line, the questionnaire is quite useful to assess what are the emotional strategies that people are using in their life to address emotional stimuli, but we realized that it is not adapted to our actual sample. In the first assessment using this questionnaire we realize that most older people have problems identifying their emotional status and understanding words that we use to describe emotions (e.g., anxiety, stress, empowerment, etc.). For this reason, we remove it from the evaluation protocol. In this sense, the methodology that we propose is a qualitative methodology aligned with this issue.

Description of the tools

We decided to use mainly qualitative methods since there are no standardized questionnaires to assess this kind of construct in older adults. We design a semi-structured interview with open questions for participants about how they feel when they interact with the technology in the different use cases**.** Once we finish the semi-structured interview, we will identify the words that older people use to describe emotions, and we will design a 5 point Likert-like scale with the words that they use to describe the emotional states so that they can assess in which way they feel that emotion. The expected sample for this emotional assessment will be 30 older adults in total.

Example of the Semi-structured Interview:

- How do you feel today?
- How did you feel in the last month in your normal routine?

- How did you feel while using the technology?
- How did you feel while participating in the community?
- How did you feel while you were writing a post in Sentab?
- How did you feel while you were meeting new people in Sentab?
- How did you feel while you were using the sensors?

Example of the 5 point Likert-like “scale”:

As we mentioned above this is only an example due to the final list of “emotions” that will be designed once we finish the semi-structured interviews,

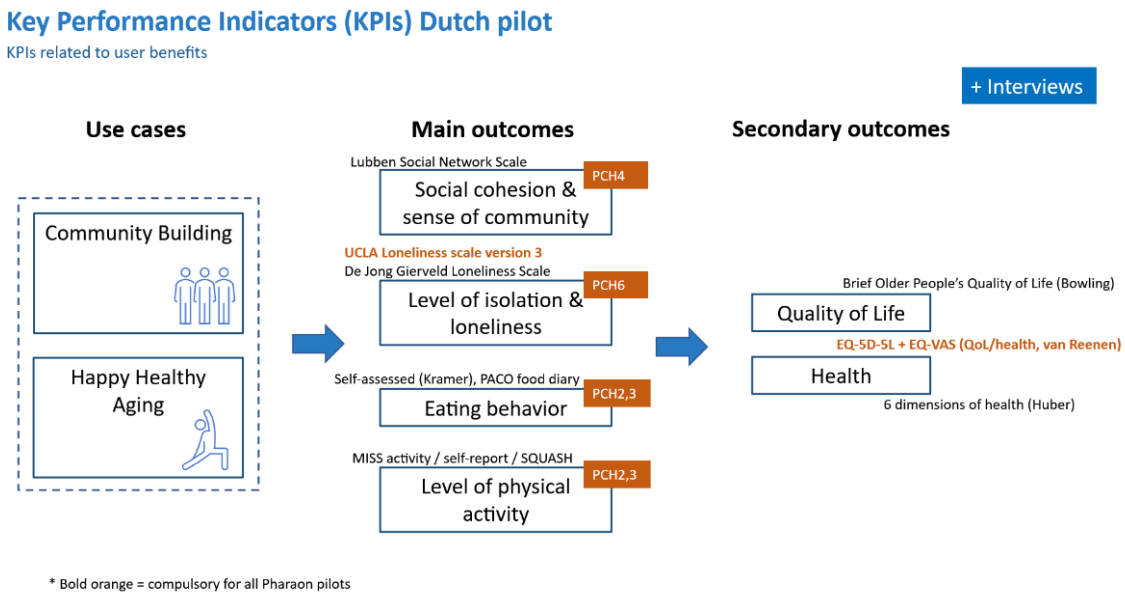
From the following list of emotions, you can indicate to what extent while using sentab/miss activity you felt...:					
	(1) Strongly disagree	(2) Disagree	(3) Neither agree nor disagree	(4) Agree	(5) Strongly agree.
Cared					
Listened					
Included					
Joined					
Motivated					

5. Assessment of emotional requirements in the Dutch pilot

Methodology

The first step was to analyze the list of the emotional goals identified in D2.1 - Dutch pilot, to determine suitable methods for assessing satisfaction. Quantitative and/or qualitative methods in order to collect feedback from end-users were both considered. In D2.1 emotional goals for both older adults and for coordinators were established. Regarding the emotional goals for older adults the level of isolation and loneliness is being measured using the UCLA Loneliness Scale (20 questions). Additionally, social cohesion and sense of community is being measured through the Lubben Social Network Scale (6 questions). Moreover, eating behavior and level of physical activity is being measured using questionnaires and activity data. These measured constructs relate to quality of life and health, which are also being measured using questionnaires, including the EQ-5D-3L and the EQ-VAS, and interviews. An overview of these constructs, the measurements, and how they relate to the Dutch pilot’s use cases is provided in the figure below.

Figure 2.15: KPI of Dutch Pilot



Regarding the emotional goals for the coordinators, interviews are employed to gain insight into their workload (i.e., required efforts and perceived benefits related to the intervention). We aim at collecting these data for a sufficient number of older adults and coordinators involved in the Dutch pilot.

6. Assessment of emotional requirements in the Slovenian pilot

Methodology

We re-analysed the list of the emotional goals identified in D2.1 for the Slovenian pilot. In our view, the appropriate means to assess the emotional goals are to combine existing questionnaires common to the overall PHArA-ON Evaluation protocol with a psychological needs satisfaction approach [6]. We will combine the results of the needs satisfaction questionnaire with the SUS and UCLA Loneliness scale to provide a more complete picture.

Accordingly, we have decided to use a quantitative tool relying on the **Basic Psychological Needs Satisfaction Scale** [7]**.** The questionnaire has been in use for many years and is considered reliable. A translation to Slovenian is needed and will be completed following the recommended process of: translation from English to Slovene, translation back to English, then making adjustments before doing a small test of understanding with respondents with similar demographics to our planned sample.

Description of the tools

The tool is short and relatively simple, making it a sound choice to reduce the length of the overall assessment protocols and to limit the burden on our target audience, Older Adults and their caregivers. The scale has 21 items measured on a 7-point Likert-like scale from "not at all true" to "very true". All questions measure "Feelings I have" and scoring is done by combining items on the scale into three items, Relatedness, Competence, and Autonomy, by average the score of the appropriate individual questions. Some items on the scale are reverse scored. The first 5 questions are quoted below to demonstrate the type of questions on the scale. Those with (R) marked before the question are reverse scored.

1. I feel like I am free to decide for myself how to live my life.
2. I really like the people I interact with.
3. (R) Often, I do not feel very competent.
4. (R) I feel pressured in my life.
5. People I know tell me I am good at what I do.

We plan to assess the needs of the older adults in our pilot and compare between the control and intervention group. We aim to include 70 older adults in the intervention group and 40 in the control group. We plan to administer the questionnaire to participants at least twice.

3 PHArA-ON Intermediate Release

This chapter describes the Intermediate Release of the PHArA-ON ecosystem providing a snapshot of the functionalities provided by each individual component involved in the PHArA-ON project at M36 and the status of their integration. In particular, section 3.1 provides an update on the interactions between the PHARA-ON components, section 3.2 an update on how the PHArA-ON components are involved in the pilots and section 3.3 an update (from D5.1) of the extensions/enhancements applied to the PHArA-ON's components (e.g., versioning info) in the context of the Intermediate Release.

3.1 Integration Matrix (Update)

Figure 3.1 reports the integration matrix related to the PHArA-ON Intermediate Release (M36). This matrix depicts the interactions between the PHArA-ON components, developed within WP3 and WP4, providing an update respect the one reported in the previous deliverable (D5.1).

The matrix depicts the current status of the interactions (i.e., "Implemented", "Ongoing", "Planned", "Proposed" and "Pre-existing". The "N" symbol (in the cells of the matrix) points out the new connections between the PHArA-ON technologies. Some of these new connections have been implemented during the integration activities, others have been proposed or planned, while other interactions are in the "Ongoing" status. The matrix also shows the introduction of the new technologies from the Open Call 1 (OC1) and their interactions. Table 3.1 provides a brief description of the winning applications (as reported in the deliverable D6.3) and in which pilots they are involved. Next section details the ongoing work to integrate the new technologies in order to fill the technological gaps in the pilot's platforms.

Figure 3.1: Interactions between PHArA-ON components (Update from D5.1)

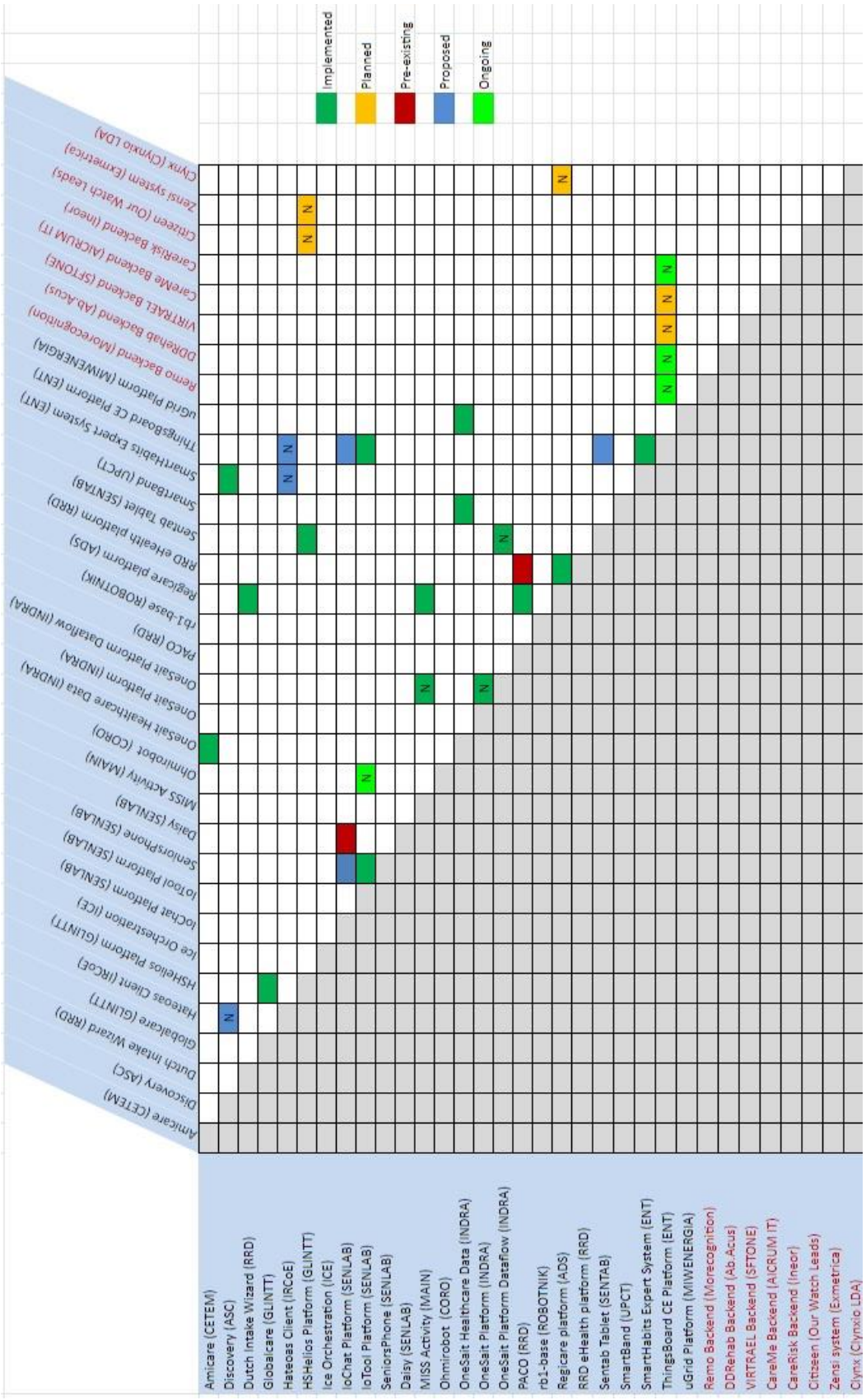


Table 3.1: Overview of the new technologies from OC1

Component	Description	Pilot
CareMe Backend (AICRUM IT)	A solution to provide personalised coaching on physical activity to improve the health and wellbeing of older adults.	Italian, Slovenian
CareRisk (Ineor)	A solution that gathers data from wearables (smart bands that record vital signs, activity, etc.), indoor environmental sensors (temperature, CO2, etc.) or other accessible data to create personalized models for elderly to keep them in good shape.	Slovenian
Citizeen (Our Watch Leads)	An app that promotes the engagement in nature preservation within cities, and the mental and physical activity of older citizens.	Portuguese
Clynx (Clynxio LDA)	A digital solution that enables physical and cognitive simulation to improve the quality of life. This solution provides digital and motivating therapy sessions that take place in an engaging game-like environment.	Dutch
DDRehab (Ab.Acus)	An innovative smartphone-based system for physical and cognitive telerehabilitation.	Italian
Remo (Morecognition)	A solution powered by a single sensor (muscle + inertial), an Android App and a back end with AI based algorithms which gives the possibility of performing physical and rehabilitation exercises delivered both at home and at private and public rehabilitation services.	Italian
VIRTRAEL (SFTONE)	A disruptive and innovative solution specially designed for cognitive stimulation and training for adults above 65 years old, based on the combination of cognitive training and serious games through a dedicated app for touchscreens and virtual reality, focusing on attention, memory, planning, and reasoning.	Italian
Zensi system (Exmetrica)	A smart system to monitor and prevent falls as well as detect early signs of illness.	Portuguese

3.2 Pilot hight level architectures (Update)

This section provides an update, respecting the previous deliverable (D5.1), of (i) the high-level architectures of the PHArA-ON pilots (i.e., the involved components and their interactions) and (ii) the technology protocols and security aspects used to achieve the integration. This section highlights the impact on the pilot's high-level architectures after the introduction of the new technologies from the OC1 (for the pilot that have stated this activity).

3.2.1 Italian pilot

Figure 3.2 depicts the high-level architecture of the Italian pilot, while Table 3.2 presents the status of interactions between the components and adopted technologies. The updates made from the previous release (highlighted with the red colour) refer both to some changes made to previous interactions and to the introduction of new ones for the integration of the technologies from OC1 (at the time being this is an ongoing work).

Figure 3.2: Italian pilot high-level architecture (Update)

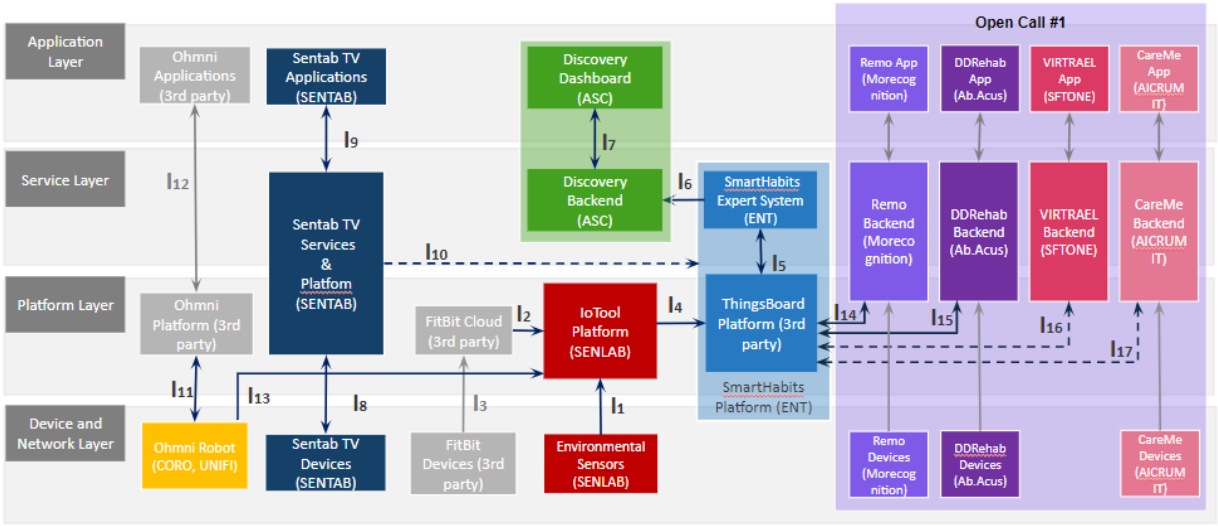


Table 3.2: Italian pilot - intermediate release of interactions between the components (Update)

Interaction	Status	Technologies and Protocols	Security
I_1	Implemented	HTTP	Various, depends on the sensor model, mostly TLS, API tokens, IP filtering. Shelly environmental sensors (N/A) - only http.
I_2	Implemented	HTTPS REST/JSON	TLS, API tokens
I_4	Implemented	MQTTS/JSON	TLS, API tokens, IP filtering

I_5	Implemented	AMQP/JSON	Secure zone/Private network, login
I_6	Implemented	HTTPS REST/JSON	TLS, API tokens, IP filtering
I_7	Implemented	HTTPS REST/JSON (OpenAPI)	TLS/API tokens/IP filtering
I_8	Pre-existing	HTTPS/JSON	SSL / login / tokens
I_9	Pre-existing	HTTPS/JSON	SSL / login / tokens
I_{10}	Proposed	HTTPS REST/JSON	TLS, API tokens, IP filtering
I_{11}	Implemented	HTTPS	SSL / login / tokens
I_{13}	Ongoing	HTTPS/JSON	TLS
I_{14}	Ongoing	HTTPS REST/JSON	TLS, API tokens, IP filtering
I_{15}	Ongoing	HTTPS REST/JSON	TLS, API tokens, IP filtering
I_{16}	Planned	HTTPS REST/JSON	TLS, API tokens, IP filtering
I_{17}	Planned	HTTPS REST/JSON	TLS, API tokens, IP filtering
I_3, I_{12}	Pre-existing (3rd party)	N/A	N/A

3.2.2 Slovenian pilot

Figure 3.3 depicts the high-level architecture of the Slovenian pilot, while Table 3.3 presents the status of interactions between the components and adopted technologies. The updates made from the previous release (highlighted with the red colour) refer both to some changes made to previous interactions and to the introduction of new ones for the integration of the technologies from OC1 (at the time being this is an ongoing work).

Figure 3.3: Slovenian pilot high-level architecture (Update)

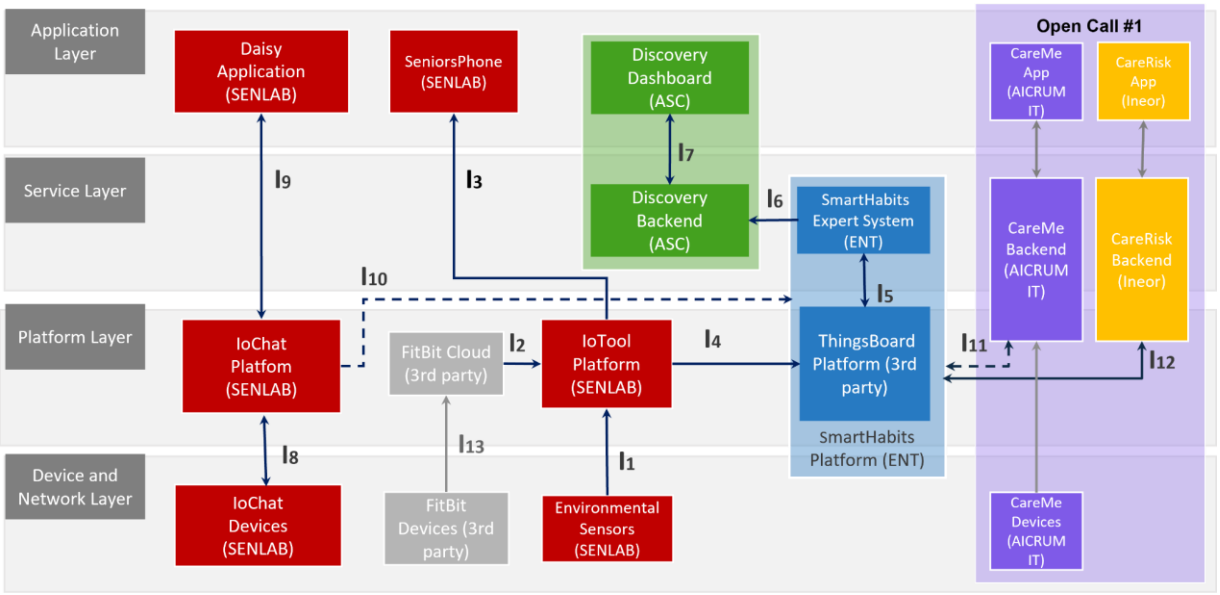


Table 3.3: Slovenian pilot - intermediate release of interactions between the components (Update)

Interaction	Status	Technologies and Protocols	Security
I_1	Implemented	HTTP	Various, depends on the sensor model, mostly TLS, API tokens, IP filtering. Shelly environmental sensors (N/A) - only http.
I_2	Implemented	HTTPS REST/JSON	TLS, API tokens
I_3	Implemented	HTTPS	Authentication (SSL), login, IP filtering
I_4	Implemented	MQTTS/JSON	TLS, API tokens, IP filtering
I_5	Implemented	AMQP/JSON	Secure zone/Private network, login

I_6	Implemented	HTTPS REST/JSON (OpenAPI)	TLS, API tokens, IP filtering
I_7	Implemented	HTTPS REST/JSON (OpenAPI)	TLS, API tokens, IP filtering
I_8	Pre-existing	N/A	N/A (NFC reader, microphone, camera on USB)
I_9	Pre-existing	HTTPS	Authentication (SSL), login, IP filtering
I_{10}	Proposed	HTTPS REST/JSON	TLS, API tokens, IP filtering
I_{11}	Planned	HTTPS REST/JSON	TLS, API tokens, IP filtering
I_{12}	Ongoing	HTTPS REST/JSON	TLS, API tokens, IP filtering
I_{13}	Pre-existing (3rd party)	N/A	N/A

3.2.3 Portuguese pilot

Figure 3.4 depicts the high-level architecture of the Portuguese pilot, while Table 3.4 presents the status of interactions between the components and adopted technologies. The updates made from the previous release (highlighted with the red colour) refer to the introduction of new interactions for the integration of the technologies from OC1 (at the time being this is an ongoing work).

Figure 3.4: Portuguese pilot high-level architecture (Update)

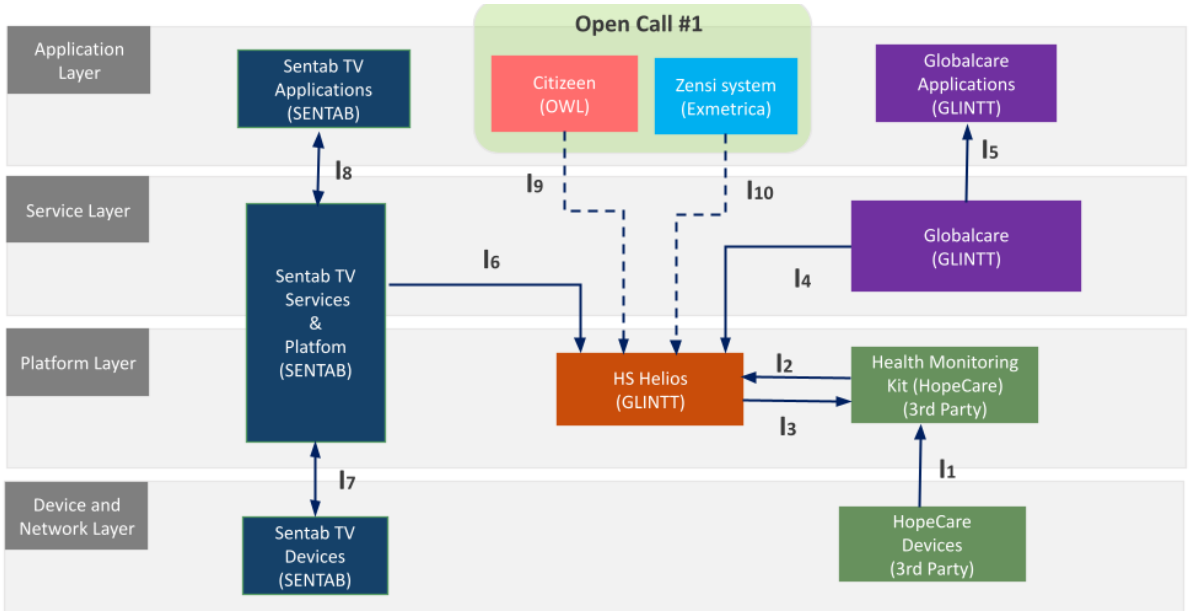


Table 3.4: Portuguese pilot - intermediate release of interactions between the components (Update)

Interaction	Status	Technologies and Protocols	Security
I_1	Implemented	REST API / HTTPS	SSL e OAuth 2.0 Encryption: RSA Key, with 512 bits
I_2	Implemented	JSON (Response)	N/A
I_3	Implemented	REST API	API KEY
I_4	Implemented	FHIR	OAuth2.0 Authentication
I_5	Implemented	REST API	OAuth2.0 Authentication SSL
I_6	Implemented	HTTPS and JSON	SSL and API Key

I_7	Pre-existing	HTTPS and JSON	SSL / login / tokens
I_8	Pre-existing	HTTPS and JSON	SSL / login / tokens
I_9	Planned	To be defined	To be defined
I_{10}	Planned	To be defined	To be defined

3.2.4 Andalusian pilot

Figure 3.5 depicts the high-level architecture of the Andalusian pilot, while Table 3.5 presents the status of interactions between the components and adopted technologies. For the time being, there are not particular updates in the high-level architecture and technology interactions due to some administrative issues related to the winning company from OC1 that is delaying the work. The pilot team is ready to start the integration of the new technologies as soon as these administrative issues are solved.

Figure 3.5: Andalusian pilot high-level architecture (D5.1)

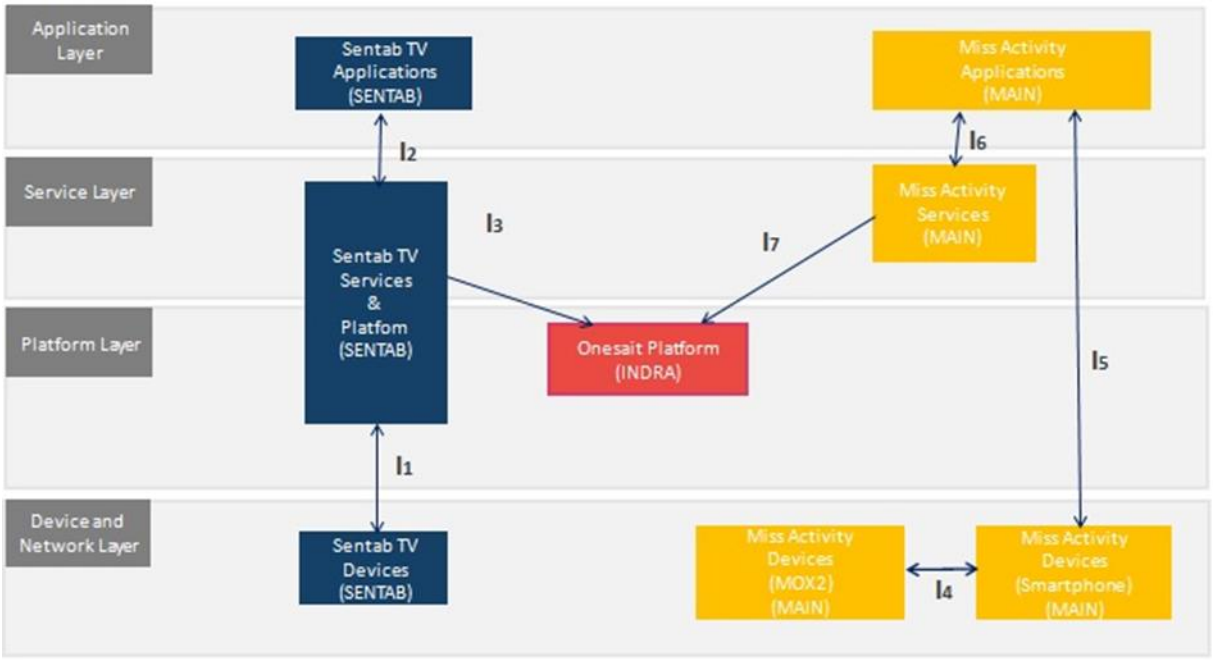


Table 3.5: Andalusian pilot - intermediate release of interactions between the components (D5.1)

Interaction	Status	Technologies and Protocols	Security
I_1	Pre-existing	HTTPS and JSON	SSL / login / tokens
I_2	Pre-existing	HTTPS and JSON	SSL / login / tokens
I_3	Implemented	HTTPS and JSON	SSL and API Key
I_4	Pre-existing	Bluetooth LE 4.0	
I_5	Implemented Planned	HTTPS and JSON	SSL API key

I_6	Planned	HTTPS and JSON	SSL and API key
I_7	Planned	HTTPS and JSON	SSL and API key

3.2.5 Murcia pilot

Figure 3.6 depicts the high-level architecture of the Murcia pilot, while Table 3.6 presents the status of interactions between the components and adopted technologies. At the moment of the release of this document the high-level diagram of the Murcia pilot has no update to report. The integration of the OC1 technologies is not moving as expected due to some administrative issues of the platform provider. Consequently, PHArA-ON management is looking for a solution. Simultaneously, pilot members and PHArA-ON management are also considering and assessing other options as a contingency plan.

Figure 3.6: Murcia pilot high-level architecture (D5.1)

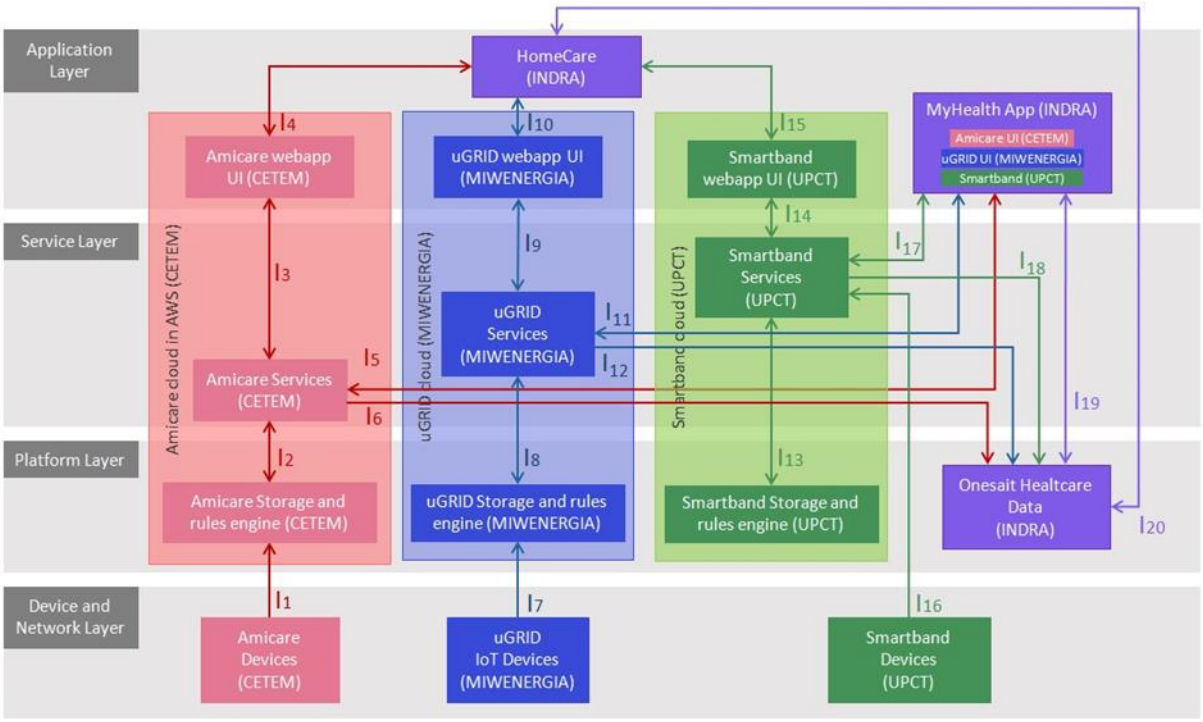


Table 3.6: Murcia pilot - intermediate release of interactions between the components (D5.1)

	Interaction	Status	Technologies and Protocols	Security
Amicare (1-6)	I_1	Pre-existing	MQTT over WIFI	Authentication
	I_2	Pre-existing	VNP	Authentication

	I_3	Pre-existing	REST API	TLS, AWS STS (Security Token Service)
	I_4	Implemented	HTTPS	JWT, SSL
	I_5	Implemented	REST API	TLS, AWS STS (Security Token Service)
	I_6	Implemented	REST API	Authentication in Rest services is through a JWT token signed with its private key
uGrid (7-12)	I_7	Implemented	MQTT over WIFI	Login
	I_8	Implemented	REST API	SSL, API key
	I_9	Pre-existing	REST API	SSL, API key
	I_{10}	Implemented	.NET Core (Blazor Server)	SSL, Login
	I_{11}	Implemented	Ionic + Angular + Cordova	SSL, Login
	I_{12}	Implemented	API REST/REST FHIR	Authentication in Rest services is through a JWT token signed with its private key
Smartband (13-18)	I_{13}	Implemented	Local host socket	Authentication
	I_{14}	Implemented	HTTPS	Authentication (SSL)
	I_{15}	Implemented	HTTPS	JWT, SSL
	I_{16}	Implemented	Bluetooth	Authentication
	I_{17}	Implemented	HTTPS REST API	JWT, SSL

	I_{18}	Implemented	API REST/REST FHIR	Authentication in Rest services is through a JWT token signed with its private key
OHC (19-20)	I_{19}	Pre-existing	API REST/REST FHIR	OAuth2
	I_{20}	Pre-existing	API REST/REST FHIR	OAuth2

3.2.6 Dutch pilot

Figure 3.7 depicts the high-level architecture of the Dutch pilot, while Table 3.7 presents the status of interactions between the components and adopted technologies. The updates made from the previous release (highlighted with the red colour) refer to the introduction of new interactions for the integration of the technologies from OC1 (at the time being this is an ongoing work).

Figure 3.7: Dutch pilot high-level architecture (Update)

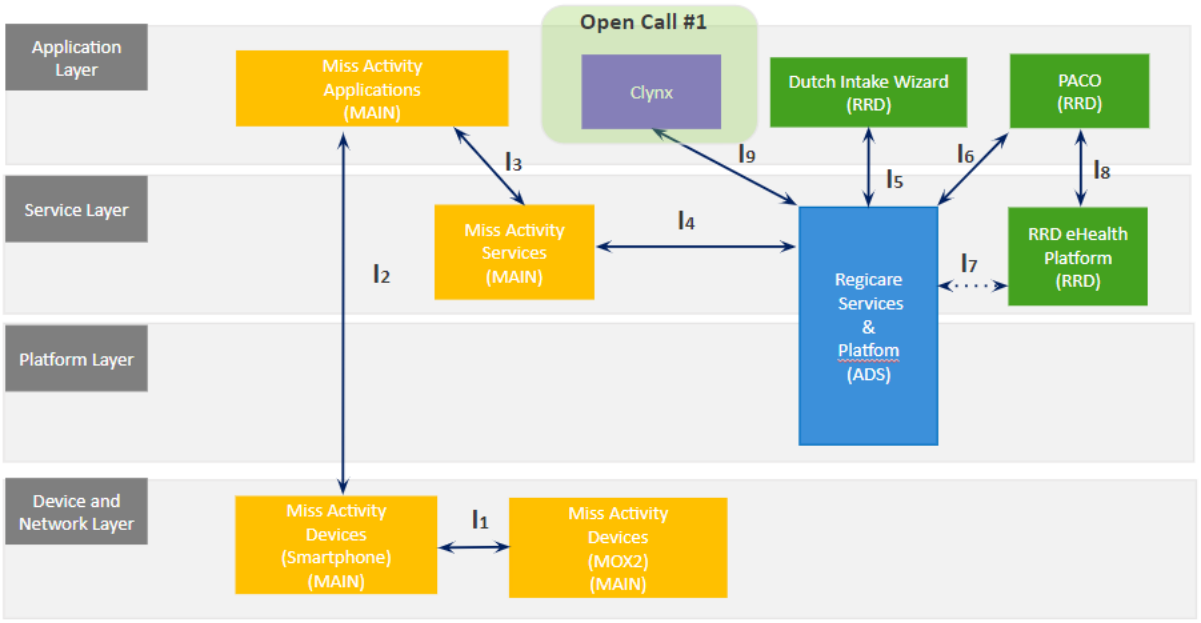


Table 3.7: Dutch pilot - intermediate release of interactions between the components (Update)

Interaction	Status	Technologies and Protocols	Security
I_1	Pre-existing	Bluetooth LE 4.0	

I_2	Implemented Planned	HTTPS and JSON	SSL and API key
I_3	Planned	HTTPS and JSON	SSL and API key
I_4	Implemented	HTTP REST/JSON	Uses SSL, OAuth 2.0
I_5	Implemented	HTTPS	Uses SSL, no authentication (the wizard is a static application)
I_6	Implemented	HTTP REST/JSON	Uses SSL, OAuth 2.0
I_7	There's no real direct interaction between Regicare and the RRD eHealth Platform; Only a simple redirect to PACO for the OAuth call-back		
I_8	Pre-existing	HTTP REST/JSON	Uses SSL, email/password login, JWT authentication
I_9	Planned	HTTP REST/JSON	Uses SSL, OAuth 2.0

3.3 Release notes (Update)

This section provides, for each technology involved in the project, an update of the release notes (i.e, the released versions with the new and/or updated functionalities, the license information, the links to the relevant resources and the main contact points) to give a snapshot of the PHArA-ON intermediate release at M36.

3.3.1 Amicare

Release notes	**Version V3.0**, released in March 2022, provides the following new functionalities: ● Notifications ● Back-end (alerts log)
License	Proprietary
Resources	● Development status and requirements are available at https://gitlab.com/pharaongroup/amicare ● User guide is available at https://gitlab.com/pharaongroup/amicare ● Images for this technology are hosted at https://gitlab.com/pharaongroup/amicare ● The service is accessible at https://master.d3bvci0sdxtp32.amplifyapp.com/ ○ User: medical01 ○ Pass: amica301
Support and contact points	● Rafael Maestre Ferriz (CETEM) - r.maestre@cetem.es ● Miguel Ángel Beteta Medina (CETEM) - ma.beteta@cetem.es

3.3.2 Discovery

Release notes	**Version 2.1** ● Setup ELK Stack ● ELK Stack integration (monitoring) ● Separated instances, so that data is not shared between instances ● Setup of test environment ● Added different Keycloak instances ● Newest MongoDB used ● Deployment of Kubernetes cluster ● Integrated health check ● Integration into secondary alarm group in Ascora ecosystem

License	Proprietary (Discovery), everything else is free
Resources	Source code available at ● https://gitlab.com/pharaongroup/carer-dashboard ● https://gitlab.com/pharaongroup/carer-dashboard-backend ● https://gitlab.com/pharaongroup/discovery ● https://gitlab.com/pharaongroup/discovery-dataleecher Images for this technology are hosted at Ascora GitLab https://registry.ascora.eu/ The service is accessible at: ● https://dashboard2-pharaon.ascora.eu/ ● https://dashboard1-pharaon.ascora.eu/ ● https://dashboard-pharaon.ascora.eu/ ● https://dashboard-dev-pharaon.ascora.eu/ ○ Test environment ○ User: admin ○ Password: MoreQuality4PharaOn
Support and contact points	● Nicolas Mayer (ASC) - mayer@ascora.de

3.3.3 Dutch Intake Wizard

Release notes	A second instance of the wizard was deployed with an alternative intro text, for locations that do not have a plusbus. The software itself was not updated and remains at **version 1.11.1**.
License	MIT
Resources	● Wool source code: https://github.com/woolplatform/wool ● PHArA-ON specific content: https://gitlab.com/pharaongroup/dutch-intake-wizard
Support and contact points	● Boris van Schooten (RRD) - B.vanschooten@rrd.nl

3.3.4 Globalcare

Release notes	**Version GCX2211** released in November 2022 brings a new functionality: ● Ability to import house care interventions between visits, without the need of validation by the social worker ● Additionally, minor fixes from the previous phase were performed
License	Globalcare is released under an end-user license agreement (EULA)
Resources	● Development status and requirements are available at https://gitlab.com/pharaongroup/Globalcare ● User guide is available at https://drive.google.com/drive/folders/1LNFUnyBIkCix_FruMcusXK2c09-NqSM9 ● Globalcare application is available at https://epr.pharaon.pt/EPR_MOBILE/Glintths.Shell and https://gc.pharaon.pt/forms/frmservlet?config=PHARAON_JNLPS
Support and contact points	● Telma Dantas (GLINTT) – telma.dantas@glintt.com

3.3.5 Hateoas Client

Release notes	The last version from 4.5.2022 includes the latest version of the code with a proxy to generate links to add in the HTTP responses. It has been tested and included in the CD/CI pipeline.
License	Open source
Resources	● https://gitlab.com/pharaongroup/hateoas-client
Support and contact points	● Michael Mrissa - michael.mrissa@innorenew.eu

3.3.6 HSHelios Platform

Release notes	HSHelios Platform is still in stable **version 2.5**, released in December 2021. From the last report in D5.1 up to now, no functionalities have been added or bugs have been reported in the context of the PHArA-ON project.

License	HSHelios is released under an end-user license agreement (EULA)
Resources	• Information about HSHelios and the specification made for the PHArA-ON project is available at https://gitlab.com/pharaongroup/hshelios-platform
Support and contact points	• Alexandre Augusto (HealthySystems) - integracoes@hltsys.pt

3.3.7 ICE Orchestration

Release notes	**Version 1.1** • Service Registry implementation and integration in Process Designer • Task Manager implementation for managing user tasks • API Gateway implementation for secured service to service connections • Process Designer bug fixes related to parameter handling and secured services
License	Proprietary
Resources	• Service Registry source code in https://gitlab.com/pharaongroup/service-registry
Support and contact points	• Jose Luis Alfonso, Information Catalyst for Enterprise (ICE) - jose.alfonso@informationcatalyst.com

3.3.8 IoTool Platform

Release notes	**v1.1. Prevalidation** • PHArA-ON platform integration, namely IoTool platform adaptation to connect with ThingsBoard/SmartHabits and further with Discovery • Shelly, Awair, QingPing sensors, mHealth band integration • Based on the IT and SI pilot requirements (needed sensor data, timeframes, frequency, privacy and security) IoTool modules part has been adapted to aggregate some data and sent it using https or MQTTs to ThingsBoard

	• IoTool module for triggering (once per minute or once per 5 minutes) of aggregated data for sending to ThingsBoard / SmartHabits • Various integration tasks **v1.2.** The production version used in IT and SI pilots. • FitBit Cloud integration • Some additional IoTool views (last_readings) have been developed based on pre-validation results to better support pilots • Integration tests between IoTool platform and SmartHabits and forward to Discovery in the test and production environments for pilots in Italy and Slovenia • Finalizing MQTT IoTool to ThingsBoard integration and monitoring tools for IoTool changes • Improved http connectors • Minor bugs resolved and improvements based on the pre-validation • Various integration tasks **V1.3**. Kubernetes containerization and overall platform improvements (Work in progress) • OmniRobot sensors integration • Improved Grafana support • Preparation to decompose IoTool as a distributed application using containers (PostgreSQL or other SQL, MQTT client, MQTT broker, frontend) which can be combined based for the specific project needs • IoTool platform to microservices and containers for the emerging Pharaon-in-a-box. This would vastly improve scalability, performance and reliability. In this period, we have selected a new hosting partner and done some initial tests on Docker and Kubernetes • Added Load balancers • Added Portainer - the Container Management (https://portainer.iotool.io/) • Added ArgoCD - Continious Deployment for all IoTool Kubernetes clusters (https://argocd.iotool.io/) • Added KeyCloak OAuth - Identity and Access Management for all IoTool Kubernetes clusters (https://oauth.senlab.io/) - Work in progress • Added Prometheus / AlertManager / Grafana - Monitoring and alerting for all IoTool servers - migration from CloudRadar solution

	(CloudRadar suspended all operations from 31.10.2022) - Work in progress ● Added GitHub Actions for continuous deployment - Work in progress ● Upgraded Mosquitto broker ● Added Swagger - API calls documentation - Work in progress ● Added node.js and node-red for event-driven routing - Work in progress ● The newest PostgreSQL v15 used ● Minor bugs resolved and improvements ● Source code changes ● Currently (Nov.2022), two Kubernetes clusters are deployed for IoTool: ○ Pharaon-demo.iotool.io ○ Pharaon-test.iotool.io ○ and Oauth server (oauth.senlab.io)
License	● Core platform: Proprietary (with some open-source components, services and modules) ● Sensor connectors for Shelly, FitBit: MIT ● Data aggregators: MIT
Resources	● IoTool web page: https://iotool.io/ ● Information about IoTool platform and changes made for the PHArA-ON project is available at: https://gitlab.com/pharaongroup/iotool-platform
Support and contact points	● Jure Lampe, CEO (SenLab) - jure.lampe@senlab.io

3.3.9 IoChat Platform

Release notes	**v1.1**. Prevalidation ● Minor bugs and improvements based on the pre-validation **v1.2**. The production version used in SI pilots. ● Added NFC support for IoChat client ● Added NFC support for platform admin

	• Minor bugs resolved and improvements based on the pre-validation **V1.3**. Kubernetes containerization and overall platform improvements (Work in progress) • Added Load balancers • Added Portainer - the Container Management • Added ArgoCD - Continuous Deployment for all IoChat Kubernetes clusters • Added KeyCloak OAuth - Identity and Access Management for all IoChat Kubernetes clusters (https://oauth.senlab.io/) • Added Prometheus / AlertManager / Grafana - Monitoring and alerting for all IoChat servers - migration from CloudRadar solution (CloudRadar suspended all operations from 31.10.2022) • Added GitHub Actions for continuous deployment • Added node-red for event-driven routing • IoChat / IoTool integration - Connection to IoTool servers, IoChat events as a sensor readings to IoTool • The newest Prosody IM 0.12.1 used • The newest PostgreSQL v15 used • Minor bugs resolved and improvements • Source code changes
License	Proprietary (with some open-source components, services and modules)
Resources	• IoChat web site: https://www.iochat.io • Information about IoChat platform and IoChat browser, Android and iOS clients and changes made for the PHArA-ON project is available at: https://gitlab.com/pharaongroup/iochat-platform
Support and contact points	• Jure Lampe, CEO (SenLab) - jure.lampe@senlab.io

3.3.10 SeniorsPhone

Release notes	**v1.1**. Pre-validation • Minor bugs and improvements based on the prevalidation • Added IoTool integration

	v1.2. The production version used in SI pilots. ● Minor bugs resolved and improvements based on the pre-validation ● New Android OS supported **V1.3**. Kubernetes containerization and overall platform improvements (Work in progress) ● New IoTool API v 1.3 used
Licence	Proprietary
Resources	● SeniorsPhone web site: https://seniorsphone.mobi/ ● Information about SeniorsPhone Launcher and changes made for the PHArA-ON project is available at: https://gitlab.com/pharaongroup/seniorsphone
Support and contact points	● Jure Lampe, CEO (SenLab) - jure.lampe@senlab.io

3.3.11 Daisy

Release notes	**v1.1**. Pre-validation ● Minor bugs and improvements based on the prevalidation **v1.2**. The production version used in SI pilots. ● Added NFC support ● Minor bugs resolved and improvements based on the pre-validation **V1.3**. Kubernetes containerization and overall platform improvements (Work in progress) ● Kubernetes back-end related changes ● Minor bugs resolved and improvements
License	Proprietary (with some open-source components, and modules)

Resources	• Daisy web site: https://daisy.iochat.io/ • Information about Daisy client and changes made for the PHArA-ON project is available at: https://gitlab.com/pharaongroup/iochat-platform
Support and contact points	• Jure Lampe, CEO (SenLab) - jure.lampe@senlab.io

3.3.12 MISS Activity

Release notes	Miss Activity is an easy-to-use activity monitor for the elderly. The current app versions are: **Android - Version 4.8** (update May 2nd, 2022) and **iOS - Version 4.8** (update May 3rd, 2022). Miss Activity is implemented in 2 PHArA-ON pilots, the Dutch and Andalusian pilots. The new features developed for PHArA-ON platform + integration are: • Designed authentication flow using OAuth between Miss Activity and Adsysco. See link: Architecture and data flow and PartnerViewModel and OAuth code snippet • Redesigned app frontend to allow authentication according to OAuth. See link: Wireframe • Included measurement ID in app. See link: Wireframe • Extended local data storage to database storage. See link: Architecture and data flow • Included privacy policies. See link: Wireframe • Created a backend database. See link: Architecture and data flow • Created web service. See link: Architecture and data flow • Integrated data calculation module in database to share matchmaking data to Adsysco. See link: Matchmaking algorithms • Investigated Data Processing Agreement (DPA) • Performed a Data Protection Impact Assessment (DPIA), in collaboration with the Data Protection Officer (DPO) from Maastricht University. • Tested the app, backend and first implementation of the Adsysco API • Transformed MISS Activity from Dutch only to a multilingual app. See link: Language selection screen • Translated the MISS Activity app to English. See link: Main screen English • Translated the MISS Activity app to Spanish. See link: Main screen Spanish • Developed MISS Activity API to connect to other platforms, e.g. Spain: Andalusia

	● Connected MISS Activity to the Andalusian Ecosystem via the MISS Activity API **Dutch Pilot Connectivity**: Miss Activity is connected via OAuth to the Adsysco Regicare portal, partner in PHArA-ON. By means of matchmaking algorithms, high-level information of physical activity of participating persons is sent to the Adsysco Regicare API. **Andalusian Pilot Connectivity**: MISS Activity is in the Andalusian Pilot connected to Minsait via the MISS Activity API. The Miss Activity API allows partners to retrieve MISS Activity data. **App download links**: ● Google Play Store: https://play.google.com/store/apps/details?id=com.MI.MissActivity&hl=en_US&gl=US ● Apple App Store: https://apps.apple.com/us/app/miss-activity/id1438242305
License	Proprietary
Resources	Public website of current app versions: ● Miss Activity on accelerometry.eu: https://www.accelerometry.eu/products/ehealth-solutions/miss-activity/ The GitLab repository contains all information on the upgrade of the current Miss Activity system to facilitate interoperability with the PHArA-ON ecosystem: ● https://gitlab.com/pharaongroup/maastricht-instruments-miss-activity
Support and contact points	● Marike Hendriks (MAIN) - marike.hendriks@maastrichtuniversity.nl ● Freek Boesten (MAIN) - freek.boesten@maastrichtinstruments.com

3.3.13 OneSait Healthcare Data

Release notes	Last version of Homecare **is v2.0** (October 2022) includes a new functionality that has been included in Homecare regarding alerts triggering due to weight variation. This is an improvement considered from the pre-validation with end-users (cardiologists).

	Last version of MyHealth (**v3.5**) includes improvements to allow third parties (technologies AMicare and Smartband) to push notifications through MyHealth App. Both applications had Go-live early in November 2022. No bug fixing in the period.
License	Proprietary
Resources	• https://gitlab.com/pharaongroup/Homecare • https://gitlab.com/pharaongroup/onesait-healthcare-data-myhealth
Support and contact points	• Mónica Álvarez (INDRA) – malvarez@minsait.com • Carlos Gutiérrez (INDRA) – cgutierrez@minsait.com

3.3.14 OneSait Platform/OneSait Platform Dataflow

Release notes	Onesait Platform is the technological platform used in the Andalusian pilot to gather and exploit the data obtained from all the other technologies used in the pilot. Currently, data from two different partners is integrated: Miss Activity and Sentab. The main component used by Onesait Platform to integrate data from different sources is Onesait Platform Dataflow. Onesait Platform Dataflow is built over Streamsets Data Collector. A tool developed by Streamsets that was an open-source project until August 2021. Since that date, the open-source project is not maintained by Streamsets. As a consequence, Minsait is maintaining a fork created from the original Streamsets Dataflow open-source repository to allow the evolution and to fix any potential error or vulnerability detected. In the context of this deliverable, Minsait has performed the following actions allow the integration of new releases (from 3.23.0 version) of Onesait Platform Dataflow: • Develop and implement all the scripts and deployment artifacts to create Docker images from the source code. All these artifacts were not present on the open-source repository and were created to integrate the new versions of Dataflow in Kubernetes and Docker environments, that are the technologies used in the Andalusian Pilot. This not only allows Minsait to build Onesait Platform Dataflow docker images, but also it allows any other organization to do it. • Generation of Docker images for each new release created. The images are available in a public Docker registry, which makes the

	usage of Onesait Platform Dataflow and its integration with Kubernetes easier. ● Develop the Jenkins pipelines to integrate the Onesait Platform Dataflow build process in the centralized Jenkins server deployed in WP5. Currently the pipelines are created in a Minsait private environment (see section 4.1.14) and the migration to the WP5 Jenkins server is an on-going task. ● Onesait Platform Dataflow uses a market place where additional libraries can be downloaded. Previously, Streamsets facilitated such a market place. Now, to keep the full project open-source, Minsait has created all the artifacts and Jenkins pipelines to allow the deployment of new library marketplaces. All the marketplace internal structure and descriptors were created to be compatible with the interface used by Onesait Platform Dataflow. This task facilitates the usage of Onesait Platform Dataflow for any organization without the need to use Repositories from other companies, nor to create and maintain custom docker images. ● In other work packages, Minsait has created guidelines for users, developers and administrators using Docusaurus. In WP5, Minsait has automated the build process of such a site using Jenkins and the creation of a docker image to allow any Onesait Platform Dataflow user to be able to deploy its own documentation site. This feature makes your own documentation site possible without using the web site of other companies. ● We are currently improving the capabilities for deploying Onesait Platform Dataflow in Kubernetes environments, specifically, to allow the creation of Dataflows clusters to provide high-availability capabilities when deployed in Kubernetes. This facilitates the integration of Onesait Platform Dataflow in High Availability deployments.
License	● Onesait Platform: Apache 2 License ● Onesait Platform Dataflow: Apache 2 License
Resources	● Onesait Platform Dataflow Repository: https://gitlab.com/pharaongroup/onesait-platform-dataflow ● Onesait Platform Repository: https://github.com/onesaitplatform/onesaitplatform-cloud ● Dataflow Docker registry: https://registry.onesaitplatform.com/repository/onesaitplatform/dataflow

	• Onesait Platform full Docker registry: https://registry.onesaitplatform.com/home
Support and contact points	• Jose María Barriga García (INDRA) - jmbarriga@minsait.com • Carlos Fernández Sánchez (INDRA) - cfsanchez@minsait.com

3.3.15 PACO

Release notes	No functional changes in this period.
License	Proprietary
Resources	• Development version (result of CI/CD): https://www.rrdhost.nl/paco/ • Production version: https://portals.rrdweb.nl/paco/
Support and contact points	• Dennis Hofs (RRD) - d.hofs@rrd.nl

3.3.16 rb1-base

Release notes	The features included in this intermediate release are: • Robot base software deployed including autonomous and manual navigations, remote screen tilt • Teleconference software deployed • Video Conference room selection development • Robot navigation improvements • Video Conference HMI improvements
License	Proprietary (with some open-source components)
Resources	Minimal open-source code of the robot: • https://github.com/RobotnikAutomation/rb1_base_common
Support and contact points	• Guillem Gari (ROB) - ggari@robotnik.es

3.3.17 Regicare Platform

Release notes	The features included in this intermediate release are: ● API Proxy to have one endpoint for connecting third party applications and still support multiple instances on a client/organization level The latest version numbers of the two Regicare platform main components are: ● **RegiWeb v1.29.0** (b2238) ● **RegiExtranet v1.36.0**
License	Proprietary
Resources	https://gitlab.com/pharaongroup/regicare-platform
Support and contact points	● Danny Jansen (ADS) - djansen@adsysco.nl

3.3.18 RRD eHealth platform

Release notes	No functional changes in this period.
License	Proprietary
Resources	● Development version (result of CI/CD): https://www.rrdhost.nl/servlets/r2d2/ ● Production version: https://portals.rrdweb.nl/servlets/r2d2/
Support and contact points	Add information on main contact persons ● Dennis Hofs (RRD) - d.hofs@rrd.nl

3.3.19 Sentab Tablet

Release notes	The features included in the latest release **25112022** are: ● Customisation to support languages requested by the consortium (Italian, Spanish, Portuguese)

	• On tablet branch, adding new features compared to TV interface such as:
	  ○ Commenting option
	  ○ Menu navigation and menu items
	  ○ Infinite scroll
	  ○ Posting text and media
	  ○ Basic chat between two users
	  ○ Reporting content / blocking users
	  ○ Visualization enhancements
	  ○ Call window improvements
	• ToS and PP links exposed
	• Bugfixes

License	Proprietary
Resources	Binary code for Sentab Tablet can be found here: • https://gitlab.com/pharaongroup/sentabtablet
Support and contact points	• Tarmo Pihl (SENTAB) - tarmo.pihl@sentab.com

3.3.20 Smartband

Release notes	**Smartband-cordova-plugin version 1.0.7 / Smartband-app version 1.2 / Smartband-backend version 1.0** Smartband solution is the software solution developed by UPCT and used in the Murcia Pilot to gather and exploit vital signs and daily activity data of the participants in the pilot. The data is monitored and collected by a commercial smartband (Myband 5) which sends the data to an Android smartphone/tablet running the piece of Smartband software developed. The smartphone/tablet works as a bridge to upload the data to the PHArA-ON Smartband server, Smartband backend software developed to collect, manage and deliver the data to third-party solutions. The Smartband solution was primarily designed to work isolated for the pre-validation phase (Smartband-app solution + Smartband-backend). In the second pre-validation phase it was integrated in the Ionic project developed by the platform provider in Murica Pilot (Indra-Minsait). It was developed as a Cordova Plugin for the Ionic integration in the Myhealth app, developed by Indra-Minsait, named Smartband-cordova-plugin. The backend did not suffer from modifications for the integration.

	The smartband-cordova-plugin encapsulates all the smartband-app functionalities for being used on a Ionic project. And it contains the code from the Smartband-app repository with the proper modifications to become a Cordova plugin. Note that the smartband-app/cordova-plugin only works with linked MiBand5 bands, previously registered on the PHArA-ON Smartband server. Then, PHArA-ON Smartband server (Smartband backend) is, in any case, a mandatory tool for the Smartband solution deployment. Note that all new/updated functionalities with respect to the initial release and bugs fixed are included in changelog in GitLab). In the context of this deliverable, UPCT has performed, with the support of the WP5 leaders, the development of the Jenkins pipelines to integrate the Smartband Dataflow build process in the Jenkins server deployed in WP5: Pipeline smartband-app and Pipeline Smartband-backend and Artifacts (see section 4.1.20).
License	● Smartband-app - LGPL (GNU Lesser General Public License) ● Smartband-cordova-plugin - LGPL (GNU Lesser General Public License) ● Smartband Backend - LGPL (GNU Lesser General Public License)
Resources	Source code repository-binary packages, website, etc., are available in the following links: ● Smartband-backend server website: https://pharaon.upct.es/ ● Smartband-app link for app download: https://pharaon.upct.es/storage/apk/pharaon_1_1_release_4.apk Gitlab repository: ● Smartband-cordova-plugin: https://gitlab.com/pharaongroup/cordova-plugin-smartband ● Smartband-app: https://gitlab.com/pharaongroup/smartband_app ● Smartband-backend:https://gitlab.com/pharaongroup/smartband_backend
Support and contact points	● María Victoria Bueno Delgado (UPCT) - mvictoria.bueno@upct.es ● Ramón Martínez Carreras (UPCT) - ramon.martinez@upct.es

3.3.21 SmartHabits Platform (SHP)

Release notes	**Version 2.1** Next major release of SmartHabits Platform (SmartHabits-v2.1) developed in scope of the PHArA-ON project from Jan 2022 to Nov 2022. The new version is focused on improvements in interoperability, deployability, scalability and maintainability of SmartHabits Platform and in several new improved data processing features that are used in pilots. Main novelties include: ● improved deployment capabilities with containerization of SmartHabits Platform components (docker support) ● integration of new load balancer and reverse proxy (HAProxy) in SmartHabits Platform solution ● improved communication protocol between IoT layer (ThingsBoard) and SHP Expert System (added AMQP support) ● new major framework upgrade ● adaptations to anomaly detection capabilities to support several new sensor types ● added support for new Q4H value dimension ● improved processing capabilities of several sensor types ● new version of ThingsBoard rule chain for improved interoperability with IoTool ● replaced open api implementation (springfox replaced with springdoc) ● improved monitoring and logging support (alertmanager, prometheus, grafana)
License	Proprietary, copyright Ericsson Nikola Tesla d.d.
Resources	Source code and binaries are hosted on private GitLab in Ericsson Nikola Tesla d.d., PHArA-ON Gitlab dedicated repository for documentation, issue reporting, and site can be found on following link: ● https://gitlab.com/pharaongroup/smarthabits-platform Installation and maintenance for PHArA-ON is completely covered by ENT but basic instructions are also reported in Developer Handbook to be transparent to the inquiring parties:

	• https://gitlab.com/pharaongroup/developers-handbook/-/wikis/Installation/SmartHabits-installation The SmartHabits Platform uses alongside the privately hosted IoT platform layer based on the open source (Apache License 2.0) Community Edition of the ThingsBoard platform. Public links are https://thingsboard.io/ https://github.com/thingsboard/thingsboard, while PHArA-ON -specific link is at https://gitlab.com/pharaongroup/thingsboard-platform. PHArA-ON specific link includes updates done to ThingsBoard rule chain configuration that includes data filters, model transformers and REST gateway. The ThingsBoard offers a vast amount of up-to-date documentation https://thingsboard.io/docs/ that facilitates onboarding of new use cases and possible future technical integrations. Installation and maintenance for PHArA-ON is covered by ENT, but basic instructions are also reported in Developer Handbook to be transparent to the inquiring parties: • https://gitlab.com/pharaongroup/developers-handbook/-/wikis/Installation/ThingsBoard-installation
Support and contact points	Add information on main contact persons • Miran Mošmondor (ENT) - miran.mosmondor@ericsson.com • Andrej Grgurić (ENT) - andrej.grguric@ericsson.com

3.3.22 uGRID

Release notes	**uGRID release v2**
	• **V2** release notes
	  ○ No changes were necessary from v1.1
	• **V1.1** release notes
	  ○ Detected bugs in pre validation phase have been fixed
	  ○ Backend changes in order to improve data collection
	  ○ Backend changes in alerts engine
	  ○ Backend changes in database engine

License	Proprietary
Resources	• Repository https://gitlab.com/pharaongroup/ugrid-platform

	• Platform (production) https://ugrid-pharaon.miwenergia.com/login
Support and contact points	• Antonio Sánchez Ibernón (MIW)- antonio.sanchez@miwenergia.com • Ana Garcia Garre (MIW) - ana.garcia@miwenergia.com • Ricardo Pérez de Zabalza (MIW) - ricardo.pz@miwenergia.com

3.3.23 Ohmirobot / telepresence service

Release notes	The features included in this intermediate release are: • HMI (Human Machine Interface) : Implementation of a customised HMI to interact with the robot. The HMI is WEB hosted and allows the user to activate the PHArA-ON remote telepresence service • Integration of ROS (Robotic Operative System) robot embedded services to exploit future robotic platforms integration in the PHArA-ON ecosystem • Bug fixing • Laboratory tests • Pilot site integration test • Pilot experimentation support
License	Open source BSD and GPL license types
Resources	• https://gitlab.com/pharaongroup/telepresence-robot-ohmni
Support and contact points	• Dr. Manuele Bonaccorsi (CORO) -management@corobotics.eu • Eng. Antonio Ferritto(CORO) - antonio.ferritto@corobotics.eu

4 Component/pilot pipelines

This chapter describes the per-technology pipelines and the work done to realize the per-pilot pipelines, according to the approach described in section 2.1. Section 4.1 reports the definition and/or the implementation of the CI/CD pipelines related to the PHArA-ON's components. In particular, for each technology a description of the DevOps phases covered by the pipelines (manually or in an automated way) are described, along with information such as used tools (private or project-wide), the implementation status, the kind of performed tests and results. Section 4.2, instead, reports the per-pilot pipeline example provided for the Duch pilot.

4.1 Per-technology pipelines

4.1.1 Amicare

CI/CD pipelines	
Description	Build, downloads the source code, executes unit tests, compiles code. Commit to GitLab, the server detects the changes, builds, packages, and orchestrates the web app.
Tools	Tools used: • Private Tools: GitLab and AWS CI/CD. • Tools used for testing: Selenium.
Implementation Status	Completed
Performed Tests	• Functional: unit tests, UI tests, API tests. Private Amicare tests. • Non-functional: Data monitoring, security (AWS).
Results	All tests executed successfully without errors.

4.1.2 Discovery

CI/CD pipelines	
Description	Automatic build, and integration testing are carried out by private Gitlab. Builds get published as docker images to the Gitlab docker registry. After successful building, deployment must be manually triggered with an internal

	tool. After that, docker containers are automatically updated and restarted in a docker swarm infrastructure with 2 instances each. In this, the instances are restarted separately and traffic is sent to the not restarting container, so that no downtime is happening.
Tools	Private Tools ● GitLab ○ Code repository with complete DevOps chain till docker registry ● Dockman ○ UI to manually restart docker containers in docker swarm and increase the running instances of it. Authorisation integrated, so that only authorized people can restart specific containers ● Docker swarm ○ Like Kubernetes. Redundancy, reduced downtime, possibility to use a load balancer ● Traefik ○ Load balancer ● Watchdog ○ Alarm service for DevOps, if hardware or a service fails, which escalates alarm on smartphone for ASC admins. Also has some automatic resolutions integrated, like restarting containers, increasing the number of services etc. Tools used for testing ● Cypress
Implementation Status	Completed
Performed Tests	● Functional tests ○ Integration test/UI test via Cypress ● Non-functional tests ○ ELK ○ Prometheus
Results	All tests executed successfully without errors.

4.1.3 Dutch Intake Wizard

CI/CD pipelines	
Description	Phases: build, unit test, deploy to staging.
Tools	Tools used: ● Private Jenkins server. Pipeline is defined in a Jenkinsfile on Git, and triggered via Gitlab push. ● Tools used for testing: Jest (JavaScript). Includes code coverage.
Implementation Status	Completed
Performed Tests	● Unit test of Wool engine by running all example dialogues through the parser.
Results	Pipeline execution success and tests passed. Code coverage: 79% of the core engine; 62% of all engine-related classes. This does not include user interface code.

4.1.4 Globalcare

CI/CD pipelines	
Description	Our pipeline is composed of 5 stages: Checkout, Build, Test, Publish and Deploy. At Checkout all source-code is free and checked for good practices and static checks and submitted into the source-code repository. At Build, dependencies are resolved and source-code verified to run against target frameworks. Test will run all unit, integration, and regression tests. Publish will generate distribution packages and at Deploy all installations processes are executed at the target.
Tools	● Sonarqube for static code check ● Jest and JUnit for unit tests ● CodeceptJS for integration tests ● X-Ray for test result repository ● JIRA for test result notification ● Bitbucket for source-code repository. ● Jenkins for automation of build, test and deploy stages

Implementation Status	Completed

Performed Tests	• Functional tests:
	  ○ Unit tests / Integration tests
	• Non-functional tests:
	  ○ Performance tests / Interoperability tests / Regression tests

Results	• Source-code static check results • Test results on X-Ray • Test reports on JIRA • Distribution packages on Bitbucket

4.1.5 Hateoas Client

CI/CD pipelines	
Description	All the phases are covered by the pipeline (the code is a Python script)
Tools	• Project-wide tools: the pipeline is part of the ENG running Jenkins instance and Sonarque is used for quality check
Implementation Status	Completed
Performed Tests	• Build and quality test
Results	CI/CD pipeline execution successful, test execution results with grade A and 0 bugs 0 vulnerabilities, 0 security hotspots, 0 code smells

4.1.6 HSHelios Platform

CI/CD pipelines	
Description	The Healthy Systems' private GitLab pipeline covers the build, unit tests, performance tests and package of all the components of the HS.HELIOS. The build stage fetches all dependencies, compiles and installs the software component. The unit and performance tests stages execute functional and performance experiments to validate the component. The package stage

	builds the rpm package of the software. Finally, we also use the GitLab container registry to update the docker image in our private repository.
Tools	All the CI/CD pipeline is done using the Healthy Systems' private GitLab.
Implementation Status	Completed
Performed Tests	During the development of the HSHelios Platform, the HealthSystems' team have performed unit and integration tests to ensure integrity and validity of the system, including module inter-connection tests. So far, we have performed the end-to-end integration test getting data collected by the HopeCare sensors and delivering it to the Globalcare solution.
Results	All tests executed successfully. Pipeline executed with no errors.

4.1.7 ICE Orchestration

CI/CD pipelines	
Description	ICE Orchestration is composed of several modules, but all of them share the same CI/CD configuration, which contains build and deploy. ● Build, downloads the source code, executes unit tests, compiles code ● Deploy, creates a new docker image and pushes it to the docker hub
Tools	Tools used: ● Since ICE Orchestration is proprietary code, we are using our own repository which is composed of Gitlab CI/CD ● For Service Registry and API Gateway modules, we use Project-wide tools (e.g., ENG's Jenkins instance)
Implementation Status	Completed
Performed Tests	We have unit and integration tests to ensure integrity and validity of the system, including module inter-connection tests.
Results	All tests executed successfully. Pipeline executed with no errors.

4.1.8 IoTool Platform

CI/CD pipelines	
Description	*IoTool backend*: build various services, manual test; *IoTool mobile client*: build in Android Studio, upload the source code to GitHub, test on various physical Android phones, merge with master, additional tests on various physical Android devices, publish to Google Play (internal test), pre-launch report from Firebase Test Lab, get feedback from beta testers, repeat or publish to production. *IoTool administrator*: PHP code testing and peer reviewing.
Tools	Tools used: ● Private Tools: Mercurial and GitHub; Issue tracking: ClickUp and GitLab, Wiki: Confluence and ClickUp, IoTool mobile client - pre launch report from Firebase Test Lab (stability, performance, accessibility, security and trust); ● IoTool mobile client: build with Android studio;
Implementation Status	Completed
Performed Tests	IoTool backend: ● Manual integration tests, services work, database connectors ● Monitoring: Prometheus/Grafana/AlertManager IoTool administrator: ● Static Analysis - lint ● Functional, integration test - connection with clients IoTool mobile client: ● Static Analysis - lint ● Functional, integration test - connection with IoTool backend ● Non-functional: monitoring Google analytic, Crashlytics
Results	No major issues.

4.1.9 IoChat Platform

CI/CD pipelines	
Description	*IoChatl backend*: build various services, manual test *IoChat backend*: build various services and scripts, manual test *IoChat client*: Build, upload the source code to GitHub, test on various physical Android and iOS phones, merge with master, additional tests on various physical Android and iOS devices, publish to Google Play (internal test) or Apple Market, Android pre-launch report from Firebase Test Lab, get feedback from beta testers, repeat or publish to production
Tools	● Private Tools: Mercurial and GitHub; Issue tracking: ClickUp and GitLab, Wiki: Confluence and ClickUp, IoChat mobile client - pre launch report from Firebase Test Lab (stability, performance, accessibility, security and trust) ● Build with Android studio
Implementation Status	Completed
Performed Tests	IoChat backend: ● Manual integration tests, services work, database connectors ● Monitoring: Prometheus/Grafana/AlertManager IoChat administrator: ● Static Analysis - lint ● Functional, integration test - connection with clients IoChat mobile client: ● Static Analysis - lint ● Functional, integration test - connection with IoChat backend ● Non-functional: monitoring Google analytic, Crashlytics
Results	No major issues.

4.1.10 SeniorsPhone

CI/CD pipelines	
Description	Build in Android Studio, upload the source code to GitHub, test on various physical Android phones, merge with master, additional tests on various physical Android devices, publish to Google Play (internal test), pre-launch report from Firebase Test Lab, get feedback from beta testers, repeat or publish to production.
Tools	• Private Tools: Mercurial and GitHub; Issue tracking: ClickUp and GitLab, Wiki: Confluence and ClickUp, pre-launch report from Firebase Test Lab (stability, performance, accessibility, security and trust) • Build with Android studio
Implementation Status	Completed.
Performed Tests	• Static Analysis - lint • Functional, integration test - connection with IoTool platform • Non-functional: monitoring Google analytic, Crashlytics
Results	No major issues.

4.1.11 Daisy

CI/CD pipelines	
Description	Browser version preparation, upload the source code to GitHub, manual test on browser, application build (Cordova) for Android, test on various physical Android TV boxes, merge with master, additional tests on various physical Android TV boxes, get feedback from beta testers, repeat or publish to production.
Tools	• Private Tools: SCM: GitHub, Issue tracking: ClickUp and GitLab, Wiki: Confluence and ClickUp • Build with Cordova

Implementation Status	Completed.
Performed Tests	• Static Analysis - lint • Functional, integration test - connection with IoChat platform, NFC readers, various TV boxes • Non-functional: monitoring Google analytic, Crashlytics
Results	No major issues.

4.1.12 MISS Activity

CI/CD pipelines	
Description	Development process MAIN Manual (CI) • 1. Coding • 2. Test on developer's PC • 3. Test pass: commit code Automatic (CI) • 4. BuildServer notices commit • 5. BuildServer builds software • 5. Build pass: Build Server does Unit Test • 6A. Unit Test pass: Build complete • 6B. Build/ Unit Test fail: Broken build then Back to step 1 Manual (CD) • 7. Manual testing, many tests up until User Tests • 8. Manual release Automatic (CD) • 9. Monitoring
Tools	Tools used: • Buildserver: Teamcity • Building with: Iar compiler • Testing with: Ceedling
Implementation Status	Implemented at local infrastructure

Performed Tests	• Functional (unit tests, API tests, system tests, acceptance tests, UI tests, User tests) • Non-functional (performance tests)
Results	Successfully passed

4.1.13 OneSait Healthcare Data

CI/CD pipelines	
Description	Automatic build, unit testing, and deployment are carried by Jenkins jobs. After successful building, JUnit testing, and Deployment in a Jboss Server, Artifacts are stored in Minsait's Sonatype Nexus repository.
Tools	Tools used: • Jenkins is used for continuous delivery • Artifacts are stored in a Sonatype Nexus repository • JUnit is used for unit testing
Implementation Status	Completed. Pilot running.

Performed Tests	• Functional
	  ○ Functional tests are done by specific team based on the acceptance criteria from the defined user stories
	  ○ UI tests: internal test platform and testing by UX team
	• Unit tests are done automatically with JMeter by Jenkins.

Results	• Coverage of unit tests • Test plan report

4.1.14 OneSait Platform/OneSait Platform Dataflow

CI/CD pipelines	
Description	The Jenkins pipelines cover the build, package and test of all the components. Additionally, the pipelines cover the Docker image creation and the publication of the image in a public Docker registry.

Tools	● Private Tools: Minsait private Gitlab because it is the corporate source code repository. Minsait private Jenkins, because it is the corporate CI/CD tool used. Onesait Platform public Docker Registry, because it is the corporate docker registry used for this project ● Project-wide tools: we are migrating to the project Jenkins instance ● Repositories where artifacts are stored: Docker registry (https://registry.onesaitplatform.com/home) ● Tools used for testing: JUnit and Mockito
Implementation Status	Completed (new versions of the pipelines could be implemented over the project execution).
Performed Tests	Onesait Platform: ● Functional: ○ unit tests: junit ○ API tests: Minsait private test platform and testing team ○ UI tests: Minsait private test platform and testing team ● Non-functional: ○ Monitoring: Prometheus + Grafana ○ Performance tests: JMeter and JProfile for specific core functionalities in Minsait private development environments ○ Logging analysis: Using Greylog Onesait Platform Dataflow: ● Functional: ○ unit tests: junit ○ API tests: junit¡ ● Non-functional ○ Monitoring: Prometheus + Grafana ○ Performance tests: JMeter and Jprofile for specific core functionalities in Minsait private development environments
Results	Onesait Platform ● Jenkins pipeline execution result (successful or details of the errors). The build process executes the unit tests ● Each release candidate has statistics about functional tests executed and the result: success, fail, block by other error. Before any new version is released, the functional errors are fixed ● Performance, monitoring, and log analysis are done in each deployment. We have not found any issue in the PHArA-ON environment Onesait Platform Dataflow:

	• Jenkins pipeline execution result (success or details of the errors). The build process executes all the unit tests • Performance, monitoring, and log analysis are done in each deployment. We have not found any issue in the PHArA-ON environment

4.1.15 PACO

CI/CD pipelines	
Description	One CI/CD pipeline has been set up for both PACO and the RRD eHealth Platform, which is the backend for the PACO application. We have set up a Jenkins pipeline that retrieves source code from a private git repository, builds Docker images for which Java code is compiled and packaged, and web applications are built. The pipeline then deploys the Docker images using Docker Compose to the RRD development server. Finally two integration tests for the database library and for the backend API are executed.
Tools	Add information on the used tools. For instance: • Private Tools: GitLab repository and Jenkins server. They are private because the source code is proprietary • Tools used for testing: JUnit
Implementation Status	Completed
Performed Tests	• Functional: integration test for the database library and for the backend API • Non-functional: none
Results	CI/CD pipeline execution and test execution: successful

4.1.16 rb1-base

For the rb1-base technology no CI/CD is in place at the moment since the software needs to run on the robot and, sometimes compiled on it.

4.1.17 Regicare Platform

CI/CD pipelines	
Description	Feature-branches are automatically deployed to a staging server for further testing. At the end of development cycle branches will be merged and manually deployed on a testing server.
Tools	Tools used: ● Private Gitlab as the code is proprietary ● Artifacts are stored in GitLab ● PHPUnit
Implementation Status	Completed
Performed Tests	● Functional: PHPUnit ● Monitoring: Prometheus, Grafana, Rollbar ● Performance: Loganalyzer
Results	CI/CD pipeline execution and test execution: successful

4.1.18 RRD eHealth platform

The RRD eHealth platform is the backend for the PACO application. All CI/CD work has been integrated for these two components and is described in the PACO section (i.e., section 4.1.5)

4.1.19 Sentab Tablet

CI/CD pipelines	
Description	Our CI/CD pipeline contains: ● Dev, test, stage and live environment ● Private Jenkins server. Pipeline is defined in a Jenkinsfile on Git, and triggered via Gitlab push
Tools	Tools used: ● Gitlab with Issue Tracking ● CI servers: Jenkins ● Wiki: Sentab wiki (Atlassian)

	• Tools used for testing: JIRA for creating testing scenarios & tickets, different plugins for Jenkins; different plugins used on developer machines (e.g. as VSCode plugins)
Implementation Status	Completed
Performed Tests	Tests performed: • Functional (unit tests, unit tests, API tests, system tests, acceptance tests, UI tests) • Non-functional (load tests, performance tests)
Results	Successfully passed

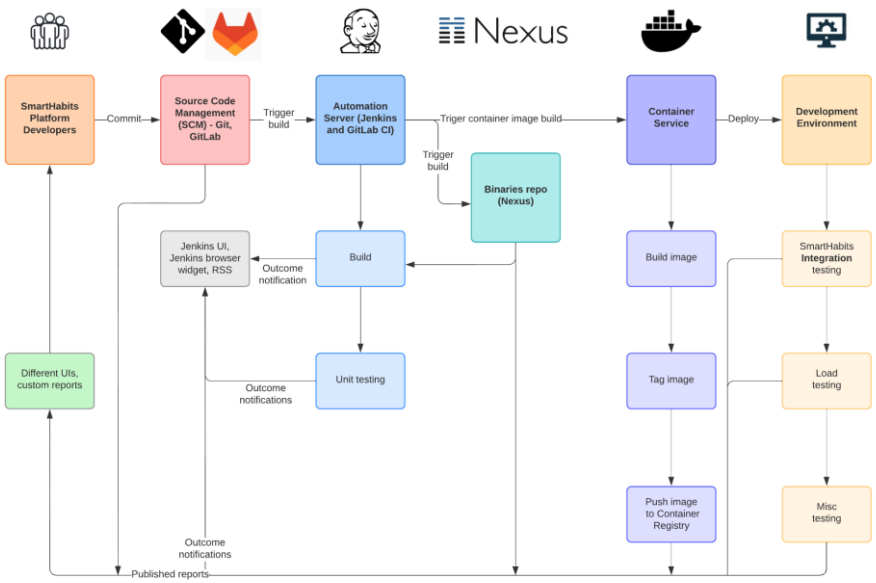
4.1.20 SmartBand

CI/CD pipelines	
Description	In the context of this deliverable, UPCT has performed, with the support of the WP5 leaders, the development of the Jenkins pipelines to integrate the Smartband Dataflow build process in the Jenkins server deployed in WP5. Two pipelines have been created in the Jenkins server: • smartband_app (android app) • smartband_backend Pipelines have been developed following the script sections guide (see section 2.2.2) with agent, stages, and post actions (optional). The agent lists set docker images to use in the stages (if necessary, in the stage section). For smartband_app two containers (android-sdk and sonar-scanner) have been used within some stages; the stages list represents the tasks needed to perform the pipeline; the post actions are those to be done (e.g. send an email). The following steps have been performed for building the pipelines: 1. Checkout - to get the source code from GitLab 2. Build Android - to build the app via Gradle (only for smartband_app) 3. Archive Files - to create APK package (artifacts, only for smartband_app) 4. Static Analysis - to get bugs, optimization code, etc... (lint analysis, only for smartband_app) 5. Sonarqube Analysis - to check the quality, security, vulnerability, etc., of the code

	6. Quality Gate - to define requirements, quality policy before releasing the code 7. Declarative: Post Actions - to be notified by email of pipeline result (successful or failed) (only for smartband_app)
Tools	Information on the used tools: Private tools: no private tools of UPCT have been used, just GitLab as the PHArA-ON tool for source code repository and Jenkins as CI/CD for creating pipelines. - Smartband-cordova-plugin https://gitlab.com/pharaongroup/cordova-plugin-smartband - Smartband-app https://gitlab.com/pharaongroup/smartband_app - Smartband-backend https://gitlab.com/pharaongroup/smartband_backend - UPCT project with Smartband solution in Jenkins: https://jenkins.res.eng.it/job/PHArA-ON/job/Partners/job/UPCT/ Project-wide tools: ENG's Jenkins and Sonarque. Jenkins repository: - Smartband-app Jenkins: https://jenkins.res.eng.it/job/PHArA-ON/job/Partners/job/UPCT/job/smartband_app/ - Smartband-backend Jenkins: https://jenkins.res.eng.it/job/PHArA-ON/job/Partners/job/UPCT/job/smartband_backend/ - Smartband-app Sonarque https://sonarqube.res.eng.it/dashboard?id=smartband_app Repositories where artifacts are stored in Jenkins repository: - Smartband-app artifact (apk): https://jenkins.res.eng.it/job/PHArA-ON/job/Partners/job/UPCT/job/smartband_app/lastSuccessfulBuild/artifact/ Tool used for testing is Sonarque (includes static analysis i.e., filter under Language as Android Lint) : - https://sonarqube.res.eng.it/project/issues?id=smartband_app&resolved=false - https://sonarqube.res.eng.it/project/issues?id=smartband_app&languages=android&resolved=false
Implementation Status	In progress/Completed (new actions could be performed in next months for creating new pipelines Docker images, etc.,)

Performed Tests	As it stated in the description, the test performed have been: ● Static Analysis - to get bugs, optimization code, etc... (lint analysis, only for smartband_app) ● Sonarqube Analysis - to check the quality, security, vulnerability, etc., of the code
Results	All CI/CD pipeline tests/execution results using Sonarcube have been Successful (see section "Tools", above)

4.1.21 SmartHabits Platform (SHP)

CI/CD pipelines	
Description	The ENT CI/CD internal pipeline and toolchain for the SmartHabits Platform specific development is summarized in the following figure: 
Tools	ENT private deployment of mostly open-source tools: ● SCM: Gitlab with Issue Tracking ● Binaries repository (mostly for maven generated files): Nexus ● Container Registry: GitLab Container repo ● CI servers: Jenkins, GitlabCI ● Wiki: Gitlab wiki

	• Tools used for testing: JUnit, JMeter, Postman, different plugins for Jenkins; different plugins used on developer machines (e.g. as VSCode plugins)
Implementation Status	Completed
Performed Tests	Complete description regarding SmartHabits and ThingsBoard deployment in test environment with instructions for testing and validation is described in Developers' Handbook, here: • https://gitlab.com/pharaongroup/developers-handbook/-/wikis/Testing/Slovenian-pilots-test-environment During September and October 2021 various functional, integration and performance tests were performed using open-source tools for automated testing tools: Postman and Apache JMeter (mentioned in D4.1). Postman can be used to automate many types of tests including unit tests, functional tests, integration tests, end-to-end tests, regression tests, mock tests, etc. Postman was primarily used to perform functional and integration testing of SmartHabits Platform. It was used to generate samples of all supported sensor data types as input to ThingsBoard platform and validating proper processed context event output through SHP REST API, on the other side. Apache JMeter is an open-source tool used for load testing to analyse and measure the efficiency and performance of the services especially if the services are web applications. It was used for performance testing of SmartHabits Platform HTTP REST and MQTT input interfaces. For the purpose of preparing for the extensive loads within PHArA-ON pilots where data ingress is planned from 3 main sites (SLO pilot site, IT Apulia pilot site and IT Tuscany pilot site) an additional performance and load test was performed. For our tests Gatling open source (https://gatling.io/open-source/) performance and testing tool was used. Kind of tests that have been performed: • Functional (including unit tests, API test, UI tests) • Non-functional (including monitoring, load, performance tests)
Results	During the above mentioned tests, no major issues have been found. More details not available in this report because of the confidentiality and security reasons

4.1.22 uGRID

CI/CD pipelines	
Description	Work has been done in building, packaging and staging different phases. Once they are completed, we will overtake the testing phase.
Tools	• Private tools: We are not using MIW private tools, just GitLab as the PHArA-ON tool for source code repository and the project-wide Jenkins instance as the CI/CD tool for creating pipelines.
Implementation Status	Not completed yet / In progress
Performed Tests	• CI/D flow implementation in progress
Results	Evaluation pending as tests must be performed

4.1.23 Ohmirobot / telepresence service

For the Ohmirobot technology there is no CI/CD pipeline in place. This is due to the fact that the software backbone in use is 3rd party, and the custom implementation focused only on the interfaces. Such interfaces are remotely hosted on a web-server and accessed using the system provided by the 3rd party software.

4.2 Per-pilot pipelines

In the Dutch pilot, RRD pioneered federated CI/CD via the methods described in section 2.1 We created a pilot-wide CI/CD pipeline that coordinates partner pipelines. We also tried integration testing with multiple Dockerized modules as an alternative. The Dutch pilot components are:

- Regicare platform (ADS)
- MISS Activity (MAIN)
- PACO (RRD)
- eHealth platform (RRD)
- Intake wizard (RRD)

Because the other Dutch pilot partners did not implement CI/CD of the relevant online services to a level that is useful for federation, we could only test the federation schemes with our own modules (the three RRD modules). This means we could only evaluate the technical aspect of CI/CD federation, not the organizational aspect.

We created a pilot-wide pipeline as a Jenkins pipeline with task and module parameters, enabling fine-grained control over federation. Communication between pipelines is performed using the generic webhook and webhook step plugins. A component pipeline calls the pilot pipeline through a webhook,

and the pilot pipeline calls it back through a callback webhook when finished. The callback URL is passed as an additional parameter to the pilot pipeline.

For example, the Jenkins code on the component side to start the Dutch intake wizard looks like the one provided in Listing 2:

Listing 2: Component pipeline - Script to call the pilot pipeline

```
stage('deploy') {
      // deploy functions
      hook = registerWebhook()
      encodedurl = java.net.URLEncoder.encode(hook.url, "UTF-8")
      result = httpRequest url:
"https://[JENKINSHOST]/generic-webhook-
trigger/invoke?token=[PilotPipelineToken]&task=start&module=DutchIntakeW
izard&returnurl=${encodedurl}"
      timeout(30) {
            data = waitForWebhook hook
      }
      // process data returned by pilot pipeline (success or fail)
}
```

So, the deploy stage of the local pipeline sends a request to the pilot pipeline to start the DutchIntakeWizard module. The callback is performed by dynamically registering a webhook, passing it to the pilot pipeline, then waiting for it to call back. The data returned is currently a simple success or fail string. Note that it will not wait forever for the pilot pipeline to finish, and in this case we chose to time out after 30 seconds.

On the pilot pipeline side, the script that handles the call to start the DutchIntakeWizard looks like the one provided in Listing 3:

Listing 3: Per-pilot pipeline - Script to handle the call from the component pipeline

```
switch (params.module) {
      case "DutchIntakeWizard":
            hook = registerWebhook()
            encodedurl = java.net.URLEncoder.encode(hook.url, "UTF-8")
            result = httpRequest url:
"https://[JENKINSHOST]/generic-webhook-
trigger/invoke?token=IntakeWizardToken&task=${params.task}&returnurl=${enc
odedurl}"
            timeout(30) {
                  data = waitForWebhook hook
            }
            // handle data
      break;
}
```

The code structure is the same. Basically, for each module, it calls a particular webhook, passing the task as a parameter. Alternatively, if the component has a separate pipeline for each task, an inner switch statement can take care of that. Or, if the component pipeline is to be called in a different way, custom code can be added. This way, the pilot pipeline can be easily customized for different setups on the components' side.

We also tried Dockerization as an alternative approach to federated CI/CD. We Dockerized both PACO and the eHealth platform. The eHealth platform serves as a backend to PACO, while the Wizard is not connected to either, so this combination made the most sense. Because both components are Dockerized, they can be easily started in tandem with an appropriate docker-compose configuration file. We used this to perform automated integration testing on the local build server. We found this an attractive approach to integration testing, even though no third-party components were involved. The testing can be done on basically any machine, is always done from a reliable start state, and all configuration is automatic and in one place. Being able to test with third-party components would provide even more added value to partners who bother to Dockerize their components.

This work on federated CI/CD done on the Dutch pilot will be taken as an example for the other pilots, in order to implement (where possible) similar per-pilot pipelines scripts. Feedback and experience from other pilots can be used to further improve the developed strategies.

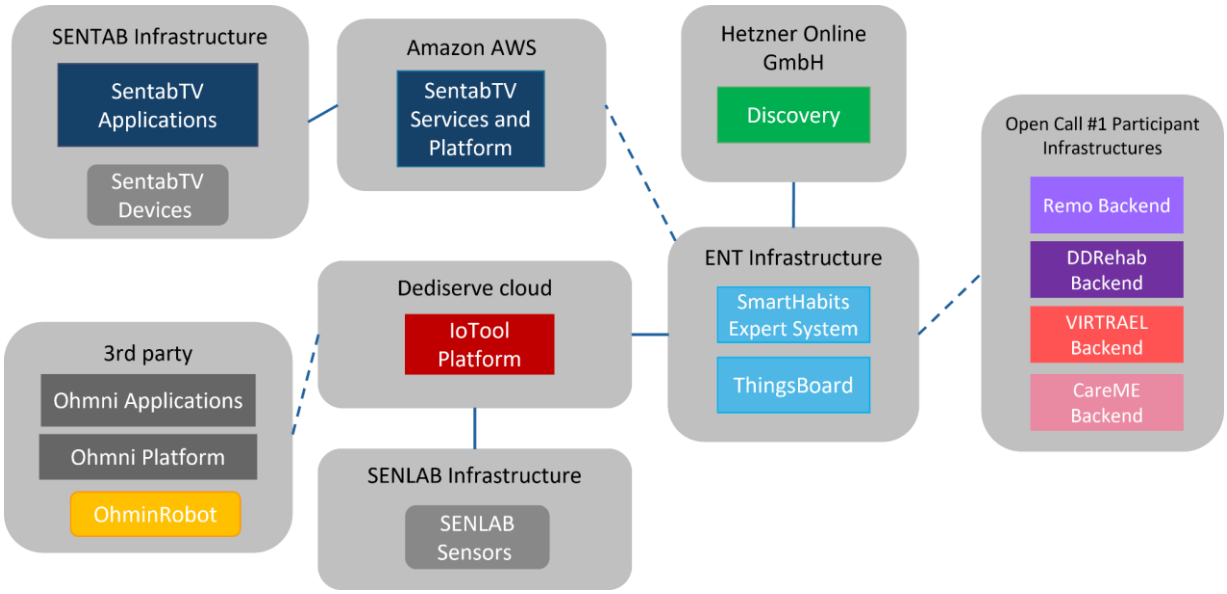
5 Staging environment (Update)

This section provides an update (from D5.1) of the staging environment where the technologies (components and services) are deployed in order to test their integration and functionalities before the deployment in production. In particular, as mentioned in the previous deliverable, the staging environment is characterised by pilot environments supported by multiple physical infrastructures (e.g., hosting infrastructures owned by the technology providers, services hosted by 3rd party providers etc.). The physical hosting infrastructures involved in each pilot, the owners of these infrastructures and the technologies deployed on them (along with the needed equipment) are detailed below.

5.1 Italian pilot

Figure 5.1 provides an update of the physical hosting infrastructures involved in the staging environment for the Italian pilot. The new technologies from the OC1 (i.e., Remo Backend, DDRehab Backend, VIRTRAELL Backend and CareMe Backend) will be hosted in the infrastructures owned by the new applicants. Moreover, a new dashed connection has been added between the 3rd party infrastructure for the OhminRobot (hosting the Ohmni Applications and Platform) and the DediSERVE4 cloud (hosting the IoTool platform). This is because the integration of the OhmniRobot and IoTool Platform technologies is ongoing (see section 3.2.1).

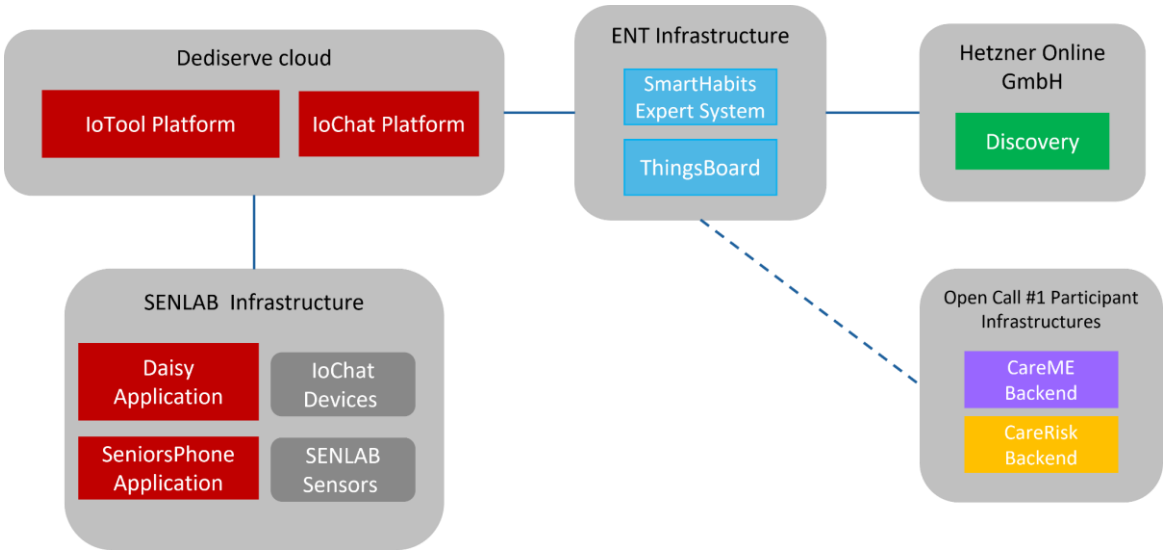
Figure 5.1: Overview of the Staging Environment for the Italian Pilot (Update)



5.2 Slovenian pilot

Figure 5.2 provides an update of the physical hosting infrastructures involved in the staging environment for the Slovenian pilot. The new technologies from the OC1 (i.e., CareMe Backend and CareRisk Backend) will be hosted in the infrastructures owned by the new applicants.

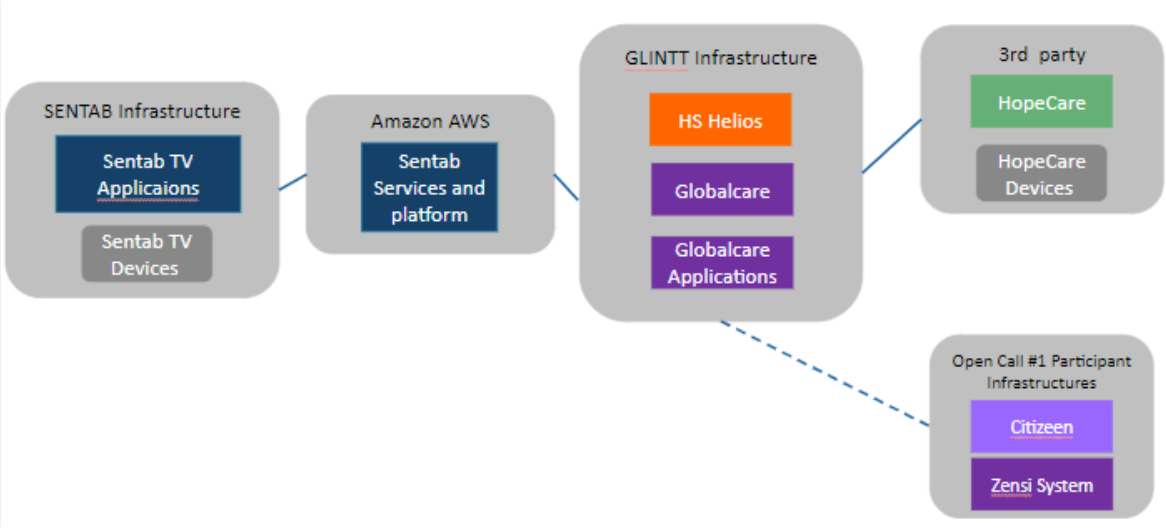
Figure 5.2: Overview of the Staging Environment for the Slovenian Pilot (Update)



5.3 Portuguese pilot

Figure 5.3 provides an update of the physical hosting infrastructures involved in the staging environment for the Portuguese pilot. The new technologies from the OC1 (i.e., Citizeen and Zensi System) will be hosted in the infrastructures owned by the new applicants.

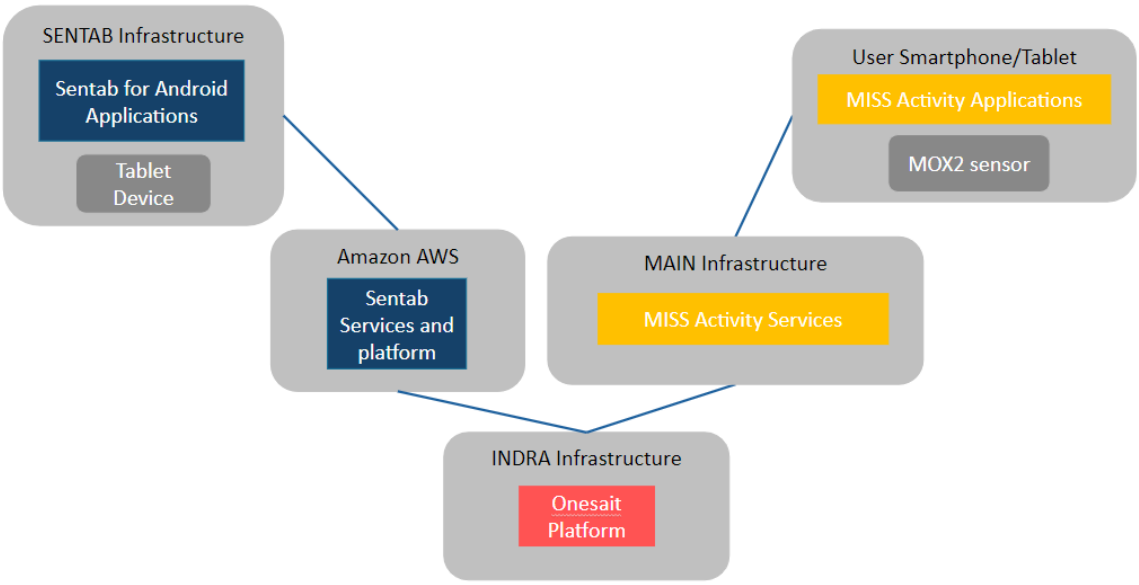
Figure 5.3: Overview of the Staging Environment for the Portuguese Pilot (Update)



5.4 Andalusian pilot

Figure 5.4 provides an update of the physical hosting infrastructures involved in the staging environment for the Andalusian pilot. Respect the previous deliverable (D5.1), the connection between the AWS infrastructure (hosting the Sentab services and platform) and the INDRA infrastructure (hosting the Onesait Platform) has been added. For the time being, no updates are made for the new technologies from the OC1 since their integration has not started yet (as mentioned in section 3.2.4).

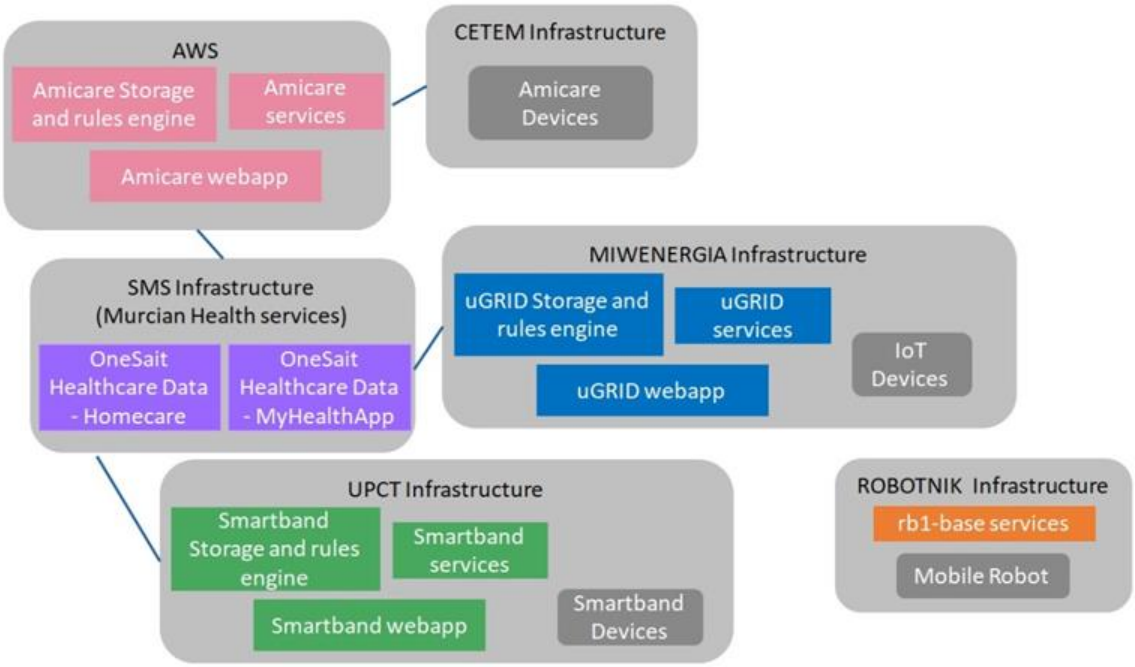
Figure 5.4: Overview of the Staging Environment for the Andalusian Pilot (Updated)



5.5 Murcia pilot

Figure 5.5 reports the physical hosting infrastructures involved in the staging environment for the Murcia pilot. For the time being, no updates are reported with respect the initial release (D5.1) since the integration of the new technologies from the OC1 has not started yet (as mentioned in section 3.2.5).

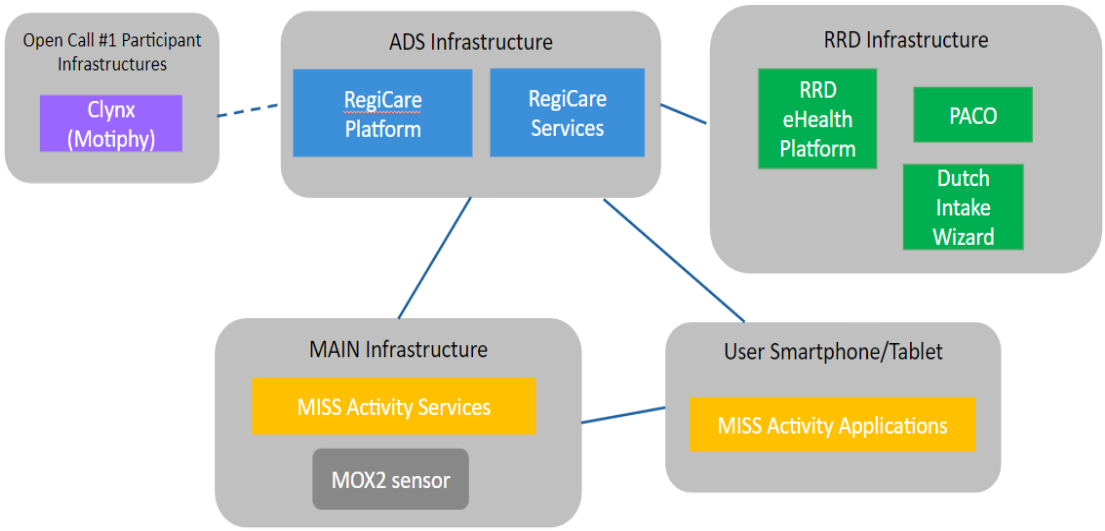
Figure 5.5: Overview of the Staging Environment for the Murcia Pilot (D5.1)



5.6 Dutch pilot

Figure 5.6 reports the physical hosting infrastructures involved in the staging environment for the Dutch pilot. The new technologies from the OC1 (i.e., Clynx) will be hosted in the infrastructures owned by the new applicants.

Figure 5.6: Overview of the Staging Environment for the Dutch Pilot (Update)



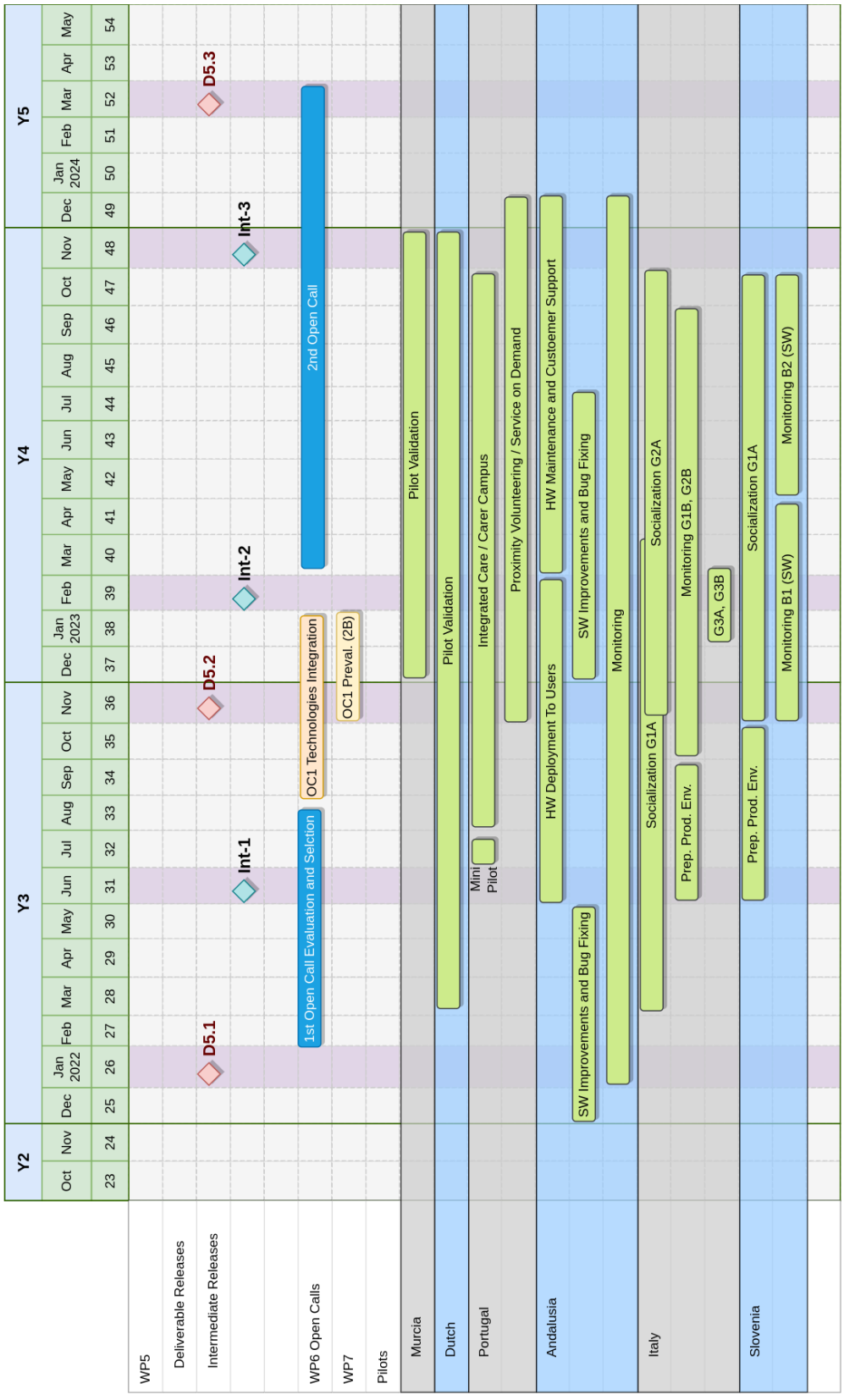
6 Release Plan (Update)

Figure 6.1 provides an update of the release plan reported in the previous deliverable (D5.1). The major releases (**M26/M36/M52**) include bug fixing and (if needed) new versions of the components that will be updated (in the context of **WP3/WP4**) to improve and/or to add new requested functionalities according to the feedbacks received during the pilot validation phases. Moreover, some internal releases have been considered in the plan to better monitor the integration and testing activities and to provide in advance (if needed) bug fixing and/or improvements of the components (**Int1, Int2 and Int3**). Respecting the previous release plan the internal release **Int2** has been moved to M39 due to some updates made in the OC1 plan (within the **WP6**). We expect that the Int2 release will include bug fixing (e.g., considering the result coming from the pre-validation Phase 2B) and the improvements made in the CI/CD pipelines of the PHArA-ON technologies. Moreover, moving this internal release makes it possible to include in it (where feasible) the work performed to test the integration of the new technologies brought by the OC1's partners, via the CI/CD process and infrastructure implemented in the project.

The internal release **Int3** (M48), instead, will focus on bug fixing, improvements and adaptations made to the PHArA-ON components to support the OC2 new pilots.

Figure 6.1 also includes the detailed operation activities of each pilot. They have been considered in the planning of WP5 internal releases Int2 and Int3 trying to make it possible to introduce newly released software in the pilot executions as soon as possible. However, as mentioned before, in PHArA-ON the production environments are considered "frozen" during the evaluation periods and the updates are avoided as much as possible. Therefore, in some cases there is not a perfect alignment between the releases and the deployment activities within WP7. Each pilot, according to their needs (hardware availability and installation, user recruitments etc.) verifies which is the best time to deploy the new software releases and restart the pilot validation.

Figure 6.1: Release Plan (Update)



7 Conclusions

This deliverable describes the Intermediate Release of the PHArA-ON ecosystem at M36 of the project, which represents the reference release for the pilot deployment from this moment on.

According to the CI/CD and testing process reported in the previous deliverable (D5.1) we monitored and coordinated the activities performed by the partners to integrate and test the PHARA-ON technologies to achieve the users and pilots requirements (defined in D2.1 and D2.3). Indeed, this document provides an update on the interactions between the technologies and on their integration status under different perspective (i.e., improvements and/or enhancements of PHArA-ON technologies, updates and/or addition of integration points, updates on the pilot high-level architectures and on the deployment diagrams on staging environment).

The work was also focused on the set-up and deployment of a "project-wide" CI/CD and testing infrastructure hosted on the ENG premises (i.e., installation and configuration of Jenkins, SonarQube, Clair and GoHarbor repository). This infrastructure was made available to the project consortium to achieve the WP5 activities and some pipeline templates were provided to facilitate their definition.

Thanks to a technical assessment made with the partners we better identified the status of CI/CD testing activities and the technology constraints (e.g., availability of the source code and binaries, licenses) in order to provide some guidelines on how to implement CI/CD pipelines (i.e., using private tools, project-wide ones or a hybrid solution). Moreover, we defined a technical implementation of the CI/CD process where we considered pipelines per technical partner and per pilot.

Concerning testing activities, we worked to improve the definition and implementation of the end-to-end testing in the context of the PHArA-ON's pilots also analyzing the available tools to perform this kind of tests. Moreover, this document reports the progress made on the assessment of the emotional requirements.

Finally, we worked to refine the release plan (respect the one presented in the D5.1) in order to align it with the pilot and open calls plans.

Concerning next steps, in the last period of the project within WP5 we will work to improve the CI/CD pipelines, supporting the migration to project-wide tools where possible; we will support the integration and adaptation activities for the OC1 and OC2 and we will improve testing work (mainly focusing on e2e testing, non-functional testing and on the validation of emotional requirements).

8 References

[1] PHArA-ON Consortium, Deliverable 5.1 - Pharaon ecosystem - Initial release, 2022.

[2] PHArA-ON Consortium, Deliverable 4.1 - Developer guidelines & templates – first, 2021.

[3] PHArA-ON Consortium, Deliverable 6.3 - Report on winning applications 1, 2022.

[4] PHArA-ON Consortium, Deliverable 2.1 - User and pilot requirements, 2020.

[5] PHArA-ON Consortium, Deliverable 2.3 - Update of Requirements and Architecture, 2022.

[6] Deci, E.L. and Ryan, R.M. (2000) 'The "What" and "Why" of Goal Pursuits: Human Needs and the Self-Determination of Behavior', Psychological Inquiry, 11(4), pp. 227–268. Available at: https://doi.org/10.1207/S15327965PLI1104_01.

[7] Gagné, M. The Role of Autonomy Support and Autonomy Orientation in Prosocial Behavior Engagement. Motivation and Emotion 27, 199–223 (2003). https://doi.org/10.1023/A:1025007614869

Appendix A: Questionnaires for emotional requirements

Table A.1: Italian Pilot - Ad-hoc questionnaire template

Domain	**User Code ----------**					
Open Question	How do you feel using the Pharaon System?					
	Can you indicate the most important change you had in your life in the past 6 months?					
	After using the Pharaon System, do you feel (answer from 1 to 5)**:**					
		Strongly Disagree	**Disagree**	**Neutral**	**Agree**	**Strongly Agree**
Less stressed	Less stressed by the usage of the Technology?	1	2	3	4	5
Informed	That you have more information on your OA? (Referred to the IC) Or in general for your health? (referred to the OA)	1	2	3	4	5
Empowerment	That you have increased the opportunities of socialization?	1	2	3	4	5
Involved	Involved in using the technology?	1	2	3	4	5

In contact	More in contact with your relatives?	1	2	3	4	5
Confident/ Conscious	More confident in using the technology?	1	2	3	4	5
Safe	Safe in using the technology?	1	2	3	4	5

Table A.2: Murcia Pilot - questionnaire templates

	Related PUCs								
Emotional requirement	**PUCS_M01**	**PUCS_M01.01**	**PUCS_M_01.02**	**PUCS_M_02**	**PUCS_M_02.01**	**PUCS_M_02.02**	**PUCS_M_03**	**How will we measure it?**	**When will we measure it**
Motivated	■	■	■	■	■	■		**-How frequently users access to Myhealth app, register their health data (weight, blood pressure), answer the health questionnaires that HPs request.**	**Intermediate (M06 of the pilot Deployment)**
Involved		■							
Informed			■		■				
Capable	■	■	■					**-How frequently users access to Myhealth app, register their health data (weight, blood pressure), answer the health questionnaires that HPs request.** **-Users feedback provided to technicians regarding difficulties they found using the system.**	**Final (M12 of the pilot deployment)**

Cared		■		■	■	■	■	**Emotional Requirements Questionnaire Q1**
Sense of belonging			■					**Emotional Requirements Questionnaire Q2**
In Control				■	■	■	■	**Emotional Requirements Questionnaire Q3**
Empowered							■	
Cautious					■			**Emotional Requirements Questionnaire Q4**

User code: _____

Domain										
Open Question	**How do you feel using the Pharaon System?**									
	After using the Pharaon System, do you feel (answer from 1 (lowest) to 10 (highest)):									
Q1	More **cared** by myself and my caregivers/health professionals	1	2	3	4	5	6	7	8	9	10
Q2	That you form part of an active healthcare community?	1	2	3	4	5	6	7	8	9	10
Q3	That you can **control** your CHF better?	1	2	3	4	5	6	7	8	9	10
Q4	More **cautious** with your health status & chronic disease	1	2	3	4	5	6	7	8	9	10

Table A.3: Portuguese Pilot - questionnaire templates

Emotional Goals

(Older Adults and Informal Caregivers)

User code: _____

Domain	Questions					
Open Question	How do you feel using the Pharaon System?					
The Most Significant Change (MSC)	Looking back over the last year, what do you think was the most significant change?					
Instructions: For each question, please rate how you feel after using the Pharaon System by placing a check in the space provided for each statement (scale 1-5)		strongly disagree	disagree	neutral	agree	strongly agree
Less stressed*	Are you stressed by the usage of Technology?	1	2	3	4	5
Informed	Do you have more information on your OA? (Referred to the IC) Or in general for your health? (referred to the OA)	1	2	3	4	5

Empowerment	Do you have increased opportunities for socialization?	1	2	3	4	5
Involved	Are you involved in using the technology?	1	2	3	4	5
In contact	Do you have more contact with your relatives?	1	2	3	4	5
Confident/ Conscious	Are you more confident in using the technology?	1	2	3	4	5
Safe	Do you feel safe using the technology?	1	2	3	4	5
Trust in the system (Almere)	I would trust the ***Pharaon*** system if it could give me advice	1	2	3	4	5
Trust in the system (Almere)	I would follow the advice given to me by the ***Pharaon*** system	1	2	3	4	5
Perceived safety	I think ***Pharaon*** is acceptably safe	1	2	3	4	5

*reverse score

Emotional Goals

(Formal Caregivers)

User code: _____

Domain	Questions					
Open Question	How do you feel using the Pharaon System?					
The Most Significant Change (MSC)	Looking back over the last year, what do you think was the most significant change?					
Instructions: For each question, please rate how you feel after using the Pharaon System by placing a check in the space provided for each statement (scale 1-5)		strongly disagree	disagree	neutral	agree	strongly agree
Less stressed*	Are you stressed by the usage of Technology?	1	2	3	4	5
Informed	Do you have more information on the OA that you take care?	1	2	3	4	5

Empowerment	Do you have increased the opportunities of socialization?	1	2	3	4	5
Involved	Are you involved in using the technology?	1	2	3	4	5
In contact	Do you have more contact with the OA that you take care?	1	2	3	4	5
Confident/ Conscious	Are you more confident in using the technology?	1	2	3	4	5
Safe	Do you feel safe using the technology?	1	2	3	4	5
Trust in the system (Almere)	I would trust the ***Pharaon*** system if it could give me advice	1	2	3	4	5
Trust in the system (Almere)	I would follow the advice given to me by the ***Pharaon*** system	1	2	3	4	5
Perceived safety	I think ***Pharaon*** is acceptably safe	1	2	3	4	5

*reverse score

Emotional Goals

(All Citizeen users)

User code: _____

Questions						
Open Question	How do you feel using the CITIZEEN System?					
The Most Significant Change (MSC)	Looking back over the last year, what do you think it was the most significant change?					
Domain	**Using the *CITIZEEN* System, do you feel**					
	Instructions: For each question, please rate how you feel after using the Pharaon System by placing a check in the space provided for each statement (scale 1-5)					
Be away Satisfied	This is a place that is far from the demands of day to day and where I can relax and think about what interests me.	I totally disagree (1)	Discordo (2)	I'm not determined (3	Agree (4)	I totally agree (5)
Fascination Curious	This place is fascinating, it's big enough for me to find out and get curious about things	I totally disagree (1)	Discordo (2)	I'm not determined (3	Agree (4)	I totally agree (5)

site extension	This is a place that is very large, without restrictions to movements, it is a world of its own apart from day to day	I totally disagree (1)	Discordo (2)	I'm not determined (3	Agree (4)	I totally agree (5)
time extension, Feeling of refuge	This is a place is a world of its own apart from day to day	I totally disagree (1)	Discordo (2)	I'm not determined (3	Agree (4)	I totally agree (5)
compatibility	In this place it is very easy to guide me and move around so I can do what I want	I totally disagree (1)	Discordo (2)	I'm not determined (3	Agree (4)	I totally agree (5)
how it affects, Accountable	How does your favorite natural area make you feel?	Very Sad (1)	Sad (2)	I don't know (neutral) (3)	Happy (4)	Very Happy (5)
		Very Calm (1)	Calm (2)	I don't know (neutral) (3)	Cheerful (4)	Very Excited (5)
preference	How attractive you consider the natural areas you visit	Not attractive (1)	Unattractive (2)	I don't know (3)	Attractive (4)	Very attractive (5)
preference	How attractive considers the photos of natural areas the local	Not attractive (1)	Unattractive (2)	I don't know (3)	Attractive (4)	Very attractive (5)

Behavioral, Sense of belonging	Feel like visiting other green sites (an approach prevention measure)	I don't feel like it. (1)	I feel little desire (2)	I don't know (neutral) (3)	I feel like it (4)	I feel very much like (5)
frequency of approximation	If it were possible, how many times a week would you like to visit a green place?	Never (1)	Once (2)	Twice (3)	Thrice (4)	Every day (5)
Preference, Connected	Would you like to live or be near a place with lots of green and a brook like the places you visit or the photos of the application?	I wouldn't like (1)	I would like little (2)	It's indifferent to me (3)	I would like to (5)	I would very much like to (6)
General life preferences - ideal scenario	If you could choose where you'd like to live?	In the countryside near a stream and green vegetation (Village, countryside)	In the countryside near other houses, but mostly green (Village, countryside)	On the outskirts of the city (around urban areas with some green areas, periurban zone	In the city (in the urban area overlooking green spaces)	In the city (in the urban area, without views of the green spaces)
System Confidence (Almere)	I would trust the CITIZEEN system if it could give me advice	I totally disagree (1)	Disagree (2)	I'm not determined (3	Agree (4)	I totally agree (5)

Trust in the system (Almere)	I would follow the advice given to me by the CITIZEEN system	I totally disagree (1)	Disagree (2)	I'm not determined (3	Agree (4)	I totally agree (5)
perceived safety	I think CITIZEEN is acceptably safe	I totally disagree (1)	Disagree (2)	I'm not determined (3	Agree (4)	I totally agree (5)