pharaon

PILOTS FOR HEALTHY AND ACTIVE AGEING

Grant Agreement: 857188

D5.1

Pharaon ecosystem - Initial release

Document Information

Deliverable number:	**D5.1**
Deliverable title:	Pharaon ecosystem - Initial release
Deliverable version:	1.0
Work Package number:	WP5
Work Package title:	Technology Ecosystem Integration
Due Date of delivery:	31/01/2022
Actual date of delivery:	11/03/2022
Dissemination level:	Public (PU)
Type	Demonstrator (DEM)
Editor(s):	Stefania D'Agostini, Gabriele Giammatteo (ENG)
Contributor(s):	Stefania D'Agostini, Gabriele Giammatteo, Pasquale Vitale, Paolo Fabriani (ENG), Michael Mrissa (IR CoE), Andrej Grgurić, Miran Mošmondor (ENT), An Stevens (MAIN), Cesar Marin (ICE), Christiane Grünloh, Vera Bulsink (RRD), Ramón Martínez Carreras (UPCT), Miguel Ángel Beteta Medina, Rafael Maestre Ferriz (CETEM), Antonio Sanchez, Ricardo Perez de Zabalza (MIWENERGIA), Mónica Álvarez, José Barriga (INDRA), Janet van den Boer (UT), Margarida Nery, Ana Canaveira, Flavio Teixeira, Paulo Correia, Pedro Pinto (GLINTT HS), Jure Lampe (SENLAB), Guillem Garí (ROB), Nicolas Mayer (ASC), Tarmo Pihl (SENTAB)
Reviewer(s):	Rafael Maestre Ferriz (CETEM)
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	22.11.2021	ENG	Initial Table of Contents
0.2	01.12.2021	ENG	Update of the Table of Content
0.3	09.12.2021	ENG	Added content on Section 2 and Section 4
0.4	10.12.2021	ENT	Input on ENT technology components
0.5	17.12.2021	ENG, CETEM, RRD, GLINTT, MAIN, ADS, UPCT, ENT, MIW	Input on technology components, Updated content on sections
0.6	16.01.2022	IR CoE	Added HATEOAS client description
0.7	18.01.2022	ENG	Updated content on Section 2 Updated content on Section 4 to include ENT and ICE contributions
0.8	26.01.2022	ENG	Added content on Section 3 and updated content on Section 4.
0.9	27.01.2022	SENLAB	Added content on Section 2 and 4 (IoChat and IoTool platforms, SeniorsPhone and Daisy clients description).
0.10	28.01.2022	ROB	Added content on Section 2 and 4 (rb1-base component)
0.11	04.02.2022	SENTAB	Added content on Section 2 and 4 (Sentab Tablet component)
0.12	09.02.2022	ENG	Improvement of Section 3
0.13	14.02.2022	ASC	Added content on Section 2 and 4 (Discovery component)
0.14	21.02.2022	ENG	Improved content on Section 3
0.15	25.02.2022	UTARTU	Revision of the deliverable wrt connections to WP2
0.16	28.02.2022	ENG	Updated images on Section 2 and improved content on Section 3
1.0	11.03.2022	ENG, CETEM	Revised and finalised document

PM Efforts per Beneficiary having contributed to the Deliverable

#	Partner	PM effort in D5.1
7	CETEM	5
10	MIWENERGIA	0.2
15	INDRA	0.25
16	SCMA	0.1
20	MAIN	0.41
21	RRD	0.2
23	UT	0.1
28	ENT	5.5
29	ASC	1
32	ICE	0.3
36	GLINTT HS	2
37	SENLAB	0.5
38	SENTAB	0.1
42	UTARTU	0.5
45	ENG	6.5

TOTAL	**Pharaon Consortium**	**22.66**

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*. In particular, the document reports the work performed under T5.1 - *Pharaon release coordination* in order to set-up a continuous integration and continuous delivery (CI/CD) process and infrastructure to guide the integration, release and testing of the various PHArA-ON components developed in WP3 and WP4.

The document presents an overview of the PHArA-ON ecosystem, identifies the integration points between all the technologies developed within the project, reports their current implementation status and describes the current deployment status in the staging environment.

Moreover, the document presents the initial activities carried out also under T5.2 - *Pharaon platform integration* and under T5.3 - *Pharaon platform testing* in order to integrate and validate the PHArA-ON ecosystem components implemented within the WP3 and WP4.

Progress Beyond the State of the Art and Play

PHArA-ON adopts a Software Development LifeCycle (SDLC) based on the DevSecOps principles and guidelines. This ensures an integration, test and release process that is at the state-of-the-art, modern, robust, reliable, secure and efficient. This also allows us to support the most modern programming languages and development and testing tools.

The adaptation of DevSecOps to the PHArA-ON project required considerable effort and resulted in the definition of customised and novel processes and tools unique for the PHArA-ON project. In fact, the two most challenging parts were:

- Adapt DevSecOps to a project with the size of PHArA-ON in terms of the number of components integrated, organisations (45) and developers involved. Also, the pure research nature of the project and the needed synchronisation of the development activities with the execution of pilots and open calls required careful customization and planning;
- Build a process and use tools flexible enough to meet the different requirements in terms of privacy, licence, availability of source code, testing and deployment of the different components.

This document describes the Continuous Integration and Continuous Delivery (CI/CD) process and the Quality Assurance (QA) approach and process defined for the PHArA-ON project.

PHArA-ON uses several phases of the Software Development LifeCycleprocess (i.e., plan, code, build, test, release and deploy). The *Plan* and *Code* phases are managed in WP3 and WP4, while WP5 takes care of the *Build*, *Test* and *Release* activities before the deployment of the pilots in production, which is a phase carried out within WP7.

In the deliverable several tools and technologies have been reviewed in order to select the more suitable ones to support W5 activities in a "project-wide" context. Indeed, no project-level solutions were available during the first phases of the project due to the fact that (i) the code base cannot be shared for the components that are not open-source and (ii) some component's licence prohibits sharing binaries. Consequently, the technology providers have organised the release of their

components autonomously using on-premises continuous integration servers to build the code and storing artifacts in private repositories. Also the testing activities have been performed following approaches specific to each component.

Therefore, the focus of this document is the definition of a CI/CD and testing approach guided by WP5 and offered at project level to manage and automate the PHArA-ON release process. Since some components (i.e., technologies and services) are not open source and/or cannot share binary artifacts, a “hybrid” approach is also considered in order to realise CI/CD pipelines using both project-level and private tools.

Contents

Figures

Tables

Acronyms & Abbreviations

Table 1: Acronyms

Term	Description
AmI	Ambient Intelligence
CI	Continuous integration
CD	Continuous Deployment or Continuous Delivery
CF	Continuous Feedback
CS	Continuous Security
DevOps	Development, and Operations
DevSecOps	Development, Security, and Operations
HCI	Human-Computer Interaction
HTTP	HyperText Transfer Protocol
JSON	JavaScript Object Notation
JWT	JSON Web Tokens
MQTT	Message Queuing Telemetry Transport
OHC	Onesait HealtCare
QA	Quality assurance
RA	Reference Architecture
REST	REpresentational State Transfer
SAAS	Software as a Service
SAST	Static Application Security Testing
SDLC	Software Development LifeCycle
SQA	Software Quality Assurance
WebRTC	Web Real-Time Communication

1 Introduction

1.1 Overview

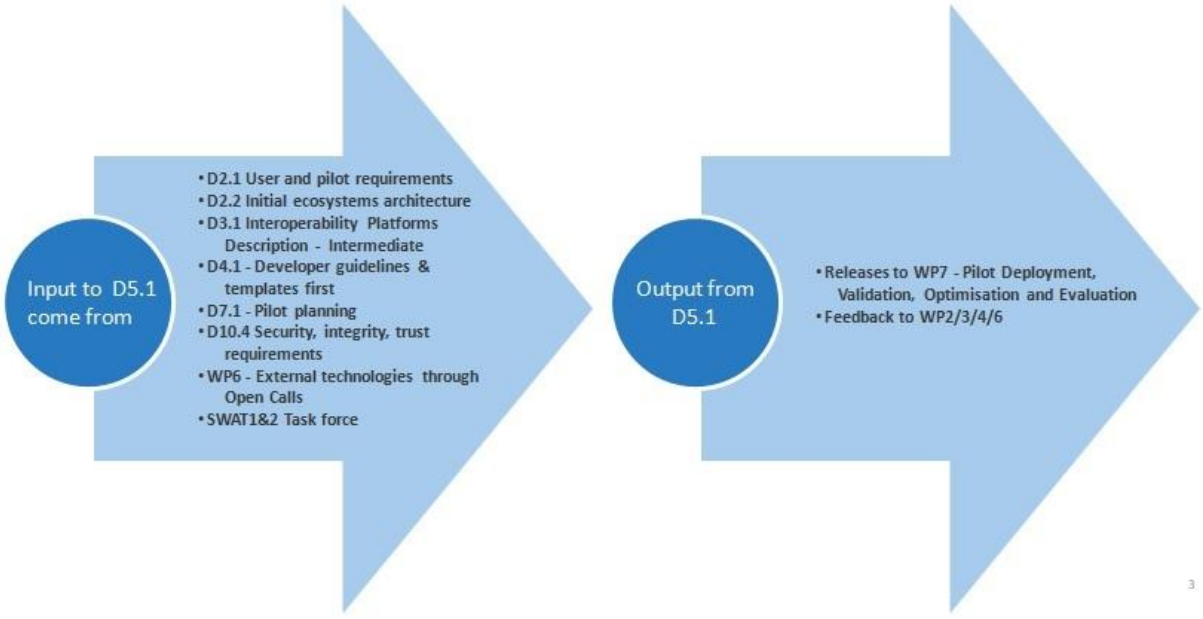
The main purpose of this deliverable is to outline the Initial Release of the PHArA-ON ecosystem at M26 of the project describing the functionalities provided by the individual components and how these components have been integrated (following the architecture defined in D2.2) to achieve the user and pilot requirements identified in deliverable D2.1. In particular, the document focuses on the progress of task T5.1 within the WP5 where the target consists of the setup of a **solid continuous integration process** in order to integrate the PHAr-A-ON services developed in WP3 and WP4. Within this task, a common release process has been set-up and managed following the main principles and guidelines of DevOps approach. In addition, several CI/CD tools have been reviewed and some of these tools have been selected to support WP5 activities. Moreover, the document presents the initial activities performed under T5.2 to support the different activities and phases of the PHArA-ON integration and release cycle.This document also identifies and describes all the integration points between all the PHArA-ON components providing an overview of their implementation status. It describes the staging environment set-up for the Initial Release, where the technologies (components and services) are deployed to test their integration and functionalities before the deployment in production. The testing activities performed within the T5.3 are based on a multi-level testing considering both functional and not-functional dimensions, in order to release high-quality software.

Finally, the Initial Release of the PHArA-ON ecosystem is described specifying for each component released basic information such as release notes, licence, support contacts and links to source code and/or binary packages.

1.2 Relation to other tasks and deliverables

Input to this deliverable comes from WP2, WP3, WP4, WP6, WP7 and WP10 activities. Figure 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: Input and output for D5.1 in relation to other WPs



1.3 Structure of the deliverable

This document describes the Initial Release of the PHArA-ON ecosystem and it is organised in the following sections:

- **Chapter 2** gives an overview of the PHArA-ON software ecosystem describing the involved components, their interactions and how they fit the PHArA-ON reference architecture. Moreover, this chapter reports the high level pilot architectures to depict the sets of PHAr-A-ON components used within each of the six pilots;
- **Chapter 3** describes the continuous integration process, techniques and tools adopted within the PHArA-ON project; the release process and the release plan along with the testing process and tools used to validate the released functionalities;
- **Chapter 4** describes the Initial Release of the PHArA-ON ecosystem at M26 providing a history and evolution of the functionalities implemented by each individual component and the preliminary testing activities that have been performed;
- **Chapter 5** draws the conclusions and the next steps.

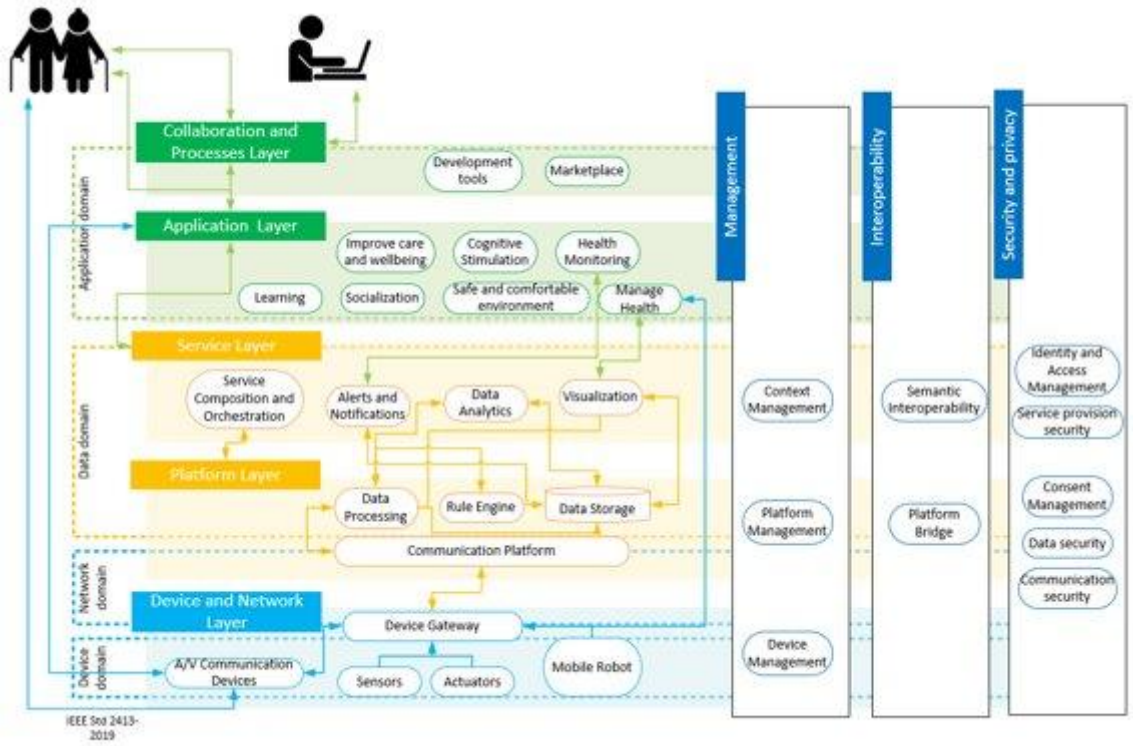
2 PHArA-ON software ecosystem overview

The initial release of the PHArA-ON ecosystem provides a set of components developed in the context of WP3 and WP4 activities that have been integrated in order to achieve the user and pilot requirements described in D2.1. This initial release represents the reference release for the pilot deployment at M28.

As stated in the deliverable D2.2, each PHArA-ON component (i.e., technology, service, module) is mapped to one or more functional boxes of the PHArA-ON Reference Architecture (RA). In particular, the PHArA-ON functional view, which is concerned with the functionalities that the system provides to the end-users, is based on several layers (i.e., Device and Network, Platform, Service, Application and Collaboration and Processes layers) according to the kind of capabilities provided by the components involved in the project (e.g., edge devices, IoT and robotics components, middleware and data platforms, application services).

All the six pilots of PHArA-ON comply with the reference architecture depicted in Figure 2 while adopting different components according to their needs. Indeed, a pilot can use a component provided by the PHArA-ON project in order to achieve a requirement not yet covered by its pre-existing solution or products.

Figure 2: Decomposition of PHArA-ON RA functional components (from D2.2)



The concepts of *intra* and *inter* platform interoperability have been introduced in the deliverables D2.2 and D3.1. The intra-platform interoperability aims to enable the communication between the different modules (e.g, technologies, services) involved in a complete system deployed on a particular platform (i.e., a pilot) while the inter-platform interoperability refers to the ability of

different complete systems deployed on different platforms (i.e., the pilots) to work together and exchange information.

In order to achieve intra-platform interoperability, the PHArA-ON components have been integrated using several open standards and protocols (such as JSON, HTTP, MQTT, etc.) along with security and privacy mechanisms needed to protect the exchanged data.

Even though the initial release of the PHArA-ON ecosystem does not cover the implementation of the inter-platform interoperability, as at the time of writing this deliverable no cross-platform use case has been defined, the work to achieve this aspect has started. As stated in the deliverable D3.1, in the PHArA-ON project, a federated approach to the architecture will be most likely adopted. With this approach, the pilots and their platforms will be developed independently but at the same time, tools and mechanisms will be put in place to interconnect and establish interoperability between different platforms. Moreover, the federated approach will provide the foundation to integrate within the PHArA-ON ecosystem new technologies coming from the open calls. At the time of writing, we are investigating and discussing the possible inter-platform interoperability solutions in the WP5 and SWAT1 synchronisation meetings. This work will be reported in the deliverables describing the next release of the PHArA-ON ecosystem (i.e., D5.2).

2.1 PHArA-ON components and interactions

Table 2 lists the components involved in the PHArA-ON ecosystem providing a brief description for each of them. Figure 3 describes how they fit the reference architecture providing an overview of the functional RA layer(s) to which they belong. More details on these software components can be found in the D3.1 deliverable of WP3.

Table 2: Overview of the PHArA-ON components

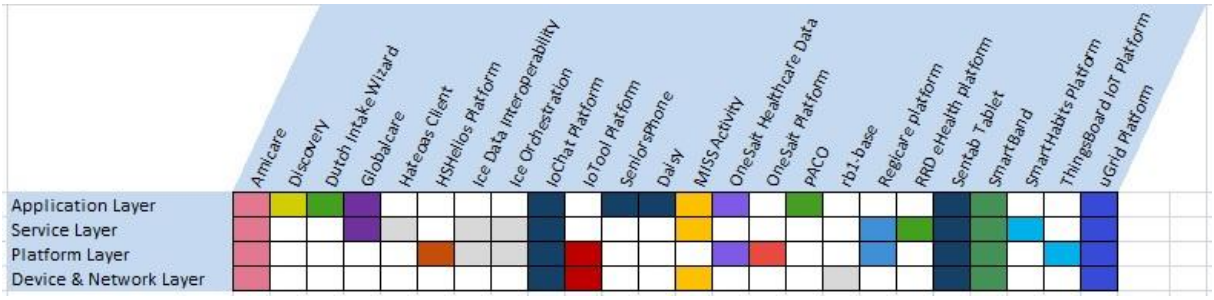
Component	Description
Amicare (CETEM)	IoT device to non-intrusively monitor and send occupancy, movement and environmental data such as temperature, humidity and brightness to the PHArA-ON Amicare server. This data can be visualised via a web or mobile app.
Discovery (ASC)	Discovery is a web application and service, which is the interface to caregivers and informal caregivers. It visualises data from sensors for specific persons and summarises sensor values like heart rate, calories burned, temperature, humidity etc. It further handles the authorization, authentication and care profiles.
Dutch Intake Wizard (RRD)	Web application that guides the user through the intake by means of a dialogue.
Globalcare (GLINTT)	The GLOBALCARE hospital management software consists of 4 product lines: HMS - Hospital Management System; Clinical; Pharma & Logistics; and BI&A – Business Intelligence & Analytics. These products cover virtually all hospital activities and meet the needs of all hospital teams. Able to be integrated and communicate with other systems, GLOBALCARE can be provided as a complete software package or as a modular solution, according to business

	needs.
Hateoas Client (IR CoE)	The HATEOAS client allows to build client-side processes by using hyperlinks to drive interactions with RESTful APIs. The client will sit between the actual services and the user interface. A proxy will enrich messages if hyperlinks are not included in legacy services.
HSHelios Platform (GLINTT)	HS.HELIOS is an integration system that promotes interoperability, by regulating and analysing the exchange of messages between different health information systems that are currently deployed at a Health institution. It is used for database semantic integrity checking and monitoring data quality in real-time.
Ice Data Interoperability (ICE)	ICE Data Interoperability is intended to provide a tool to allow the user to use a graphical interface to generate code to transform data between different formats. Generally, these transformations are short-lived and change frequently. In the PHArA-ON case, however, the transformations appear to be long-term and fixed. ICE will, therefore, use the tool to generate individual transformations (one per pilot) which will convert the pilot output to a common format once these formats have been defined. The data in common format can be stored on a central repository for use by any central analytic and/or visualisation tools
Ice Orchestration (ICE)	ICE Orchestration is a tool for designing and executing business processes. Such processes can involve tasks performed by a human (a user) such as reviewing and completing forms, manual tasks performed in an external system, and service tasks performed automatically by a software service. In the PHArA-ON case, the tool supports both 1) the service composition activities in designing processes (i.e. service compositions) made of software services only; and 2) the service orchestration activities in (providing support for) executing service compositions so that services are called one after another in an automatic manner.
IoChat Platform (SENLAB)	IoChat is the Secure WebRTC Communication Platform, Multi-platform Client / API / Self-hosted Server / Private Cloud solution used for user-to-user or multiuser text chat, file sharing, audio and video conferencing systems. In the PHArA-ON case, IoChat is used as an Android SmartTV API for the simplified TV video conference for seniors and as a standalone Android, iOS or web browser solution for caregivers. Also, IoChat is used as a SAAS solution (Cloud) to enable conferencing (presence, signalling and text chat, user database and STUN/TURN server).
IoTool Platform (SENLAB)	IoTool platform helps connect IoT devices (with more than 150 sensors and 50 actuators already supported through flexible and open extensions system) via any interface to a smartphone, microcontroller or directly to the IoTool Cloud. The collected data is encrypted, stored, displayed, processed, and synchronised to the Cloud (IoTool servers) and other solutions (ThingsBoard, IBM Watson, Amazon IoT, Microsoft Azure). In the PHArA-ON case, IoTool is used as a smartphone API for the SenLab's SeniorsPhone to collect smartphone internal and external sensor data. Also IoTool is used as a SAAS solution (Cloud) to connect smartphone clients and other environmental sensors and call buttons. All data is stored internally and has been sent to the

	SmartHabits (ENT) platform.
SeniorsPhone (SENLAB)	SeniorsPhone features a simple and easy interface for the Android smartphone with big buttons for easy calling, texting, SOS, location functions and other functions (weather, magnifier, music, torch...). In the Pharaon case, SeniorsPhone also incorporates a smart wristband or smartwatch to collect biosensor readings and send them to the IoTool platform.
Daisy (SENLAB)	Daisy is the simplest possible video conferencing via TV that seniors would be willing and able to use. It allows video conferencing with presence and it is connected to IoChat platform and from there to IoChat clients (Android, iOS, web browser).
MISS Activity (MAIN)	MISS Activity is an e-health tool that motivates elderly to be more active by enabling them to set goals and providing them insight in their activity. It consists of MOX2, a mobile app and dedicated algorithms developed in-house.
OneSait Healthcare Data (INDRA)	Onesait Healthcare is a versatile and adjustable suite of products designed for covering all management and care processes of a health system. Onesait Healthcare Data integrates, normalises and stores any health data of the organisation, facilitating the use of resources through REST API, providing integration and interoperability capabilities, adaptation to different scenarios and great versatility in exchange protocols, being an effective element in the transformation towards highly interoperable global systems based on standards. For PHArA-ON, the Homecare module will be available for healthcare professionals, and MyHealth for patients and their informal carers. Homecare allows remote monitoring of patients with CHF by healthcare professionals and MyHealth provides features for the inclusion of data by patients and their informal caregivers including integration with biometric devices, questionnaires and devices from other technology providers.
OneSait Platform (INDRA)	Onesait Platform provides the flexibility so that developers can build their own solutions in a solid and agile way using Open Source technologies. Onesait Platform covers the entire life cycle of information (from ingest to visualisation through its process and analysis). It offers a unified web console for the development and operation profiles of the solutions. Thanks to different assistants and the encapsulation of best practises, it is possible to develop systems with complex architectures in a simple way, reducing cost and time2market.
PACO (RRD)	**P**ersuasive E-health **A**gents for **C**oaching **O**lder adults: Coaching tool to help adults toward a dietary and healthier behaviour change.
rb1-base (ROBOTNIK)	Robotnik's RB-1 is an autonomous and configurable robot, focused on the field of research in indoor applications. It can carry different loads or materials and can integrate other components or platforms such as telepresence devices.
Regicare platform	RegiCare is a user-friendly information system built around the primary

(ADS)	processes of Welfare organisations.
RRD eHealth platform (RRD)	The RRD eHealth Platform is a backend service and consists of three main software modules: Android Application Framework, database management system, configurable web portal for user management and data visualisation.
Sentab Tablet (SENTAB)	Sentab is a digital platform for older adults to communicate and socially interact between themselves, caregivers and family members. The application allows sharing media, posts, getting enrolled into communities (groups), having video calls with connections, real-time broadcasts, using Android apps, etc.
SmartBand (UPCT)	IoT application to monitor vital signs using Mi Smart Band 5 and Android devices, sending heart rate and daily steps to the PHArA-ON Smartband server.
SmartHabits Platform (ENT)	Intelligent privacy-aware home care assistance system. Uses machine-learning technology to provide peace of mind to informal caregivers caring for persons living alone. It does so by learning the user's typical daily activity patterns and automatically issuing warnings if an unusual situation is detected.
ThingsBoard IoT Platform (ENT)	3rd party open-source IoT platform for data collection, processing, visualisation, and device management, acting as an IoT platform layer for the SmartHabits Platform.
uGrid Platform (MIWENERGIA)	uGRID is an energy management software developed by MIWenergia with simple solutions that helps you control and monitor your energy consumption and achieve the maximum possible sustainability. In order to know the data in real-time, a series of equipment and sensors are deployed in the internal network of the facility.

Figure 3: Overview of the RA layers of the PHArA-ON components for the Initial Release (M26)



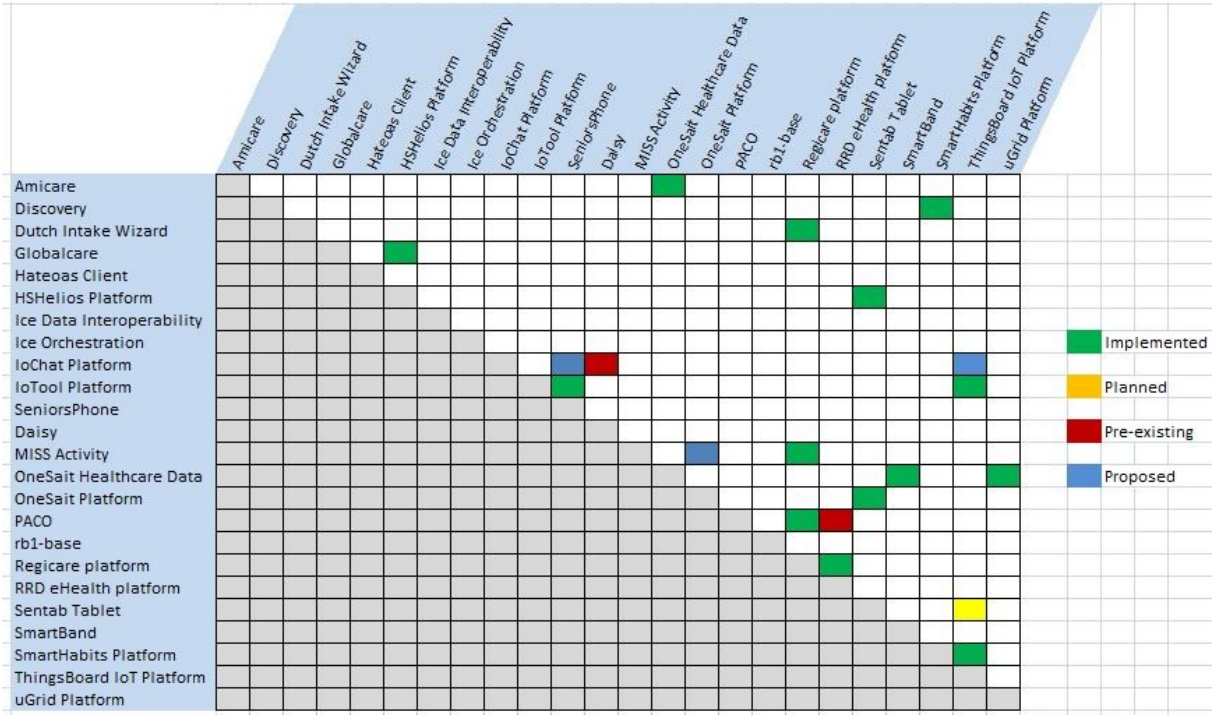
The focus of this deliverable is to highlight how the PHArA-ON components, developed within WP3 and WP4, have been integrated in order to achieve the user and pilot requirements described in the deliverable D2.1.

The matrix shown in Figure 4 provides an overview of the interactions between the PHArA-ON components. A connection point - a filled square - is indicated for the pairs of components that interact with each other. Moreover, for each of these interactions, Figure 4 reports its current

status, denoting whether the interaction has been “Implemented”, “Planned”, “Proposed” or is a “Pre-existing” one.

Section 2.2 describes the set of PHAr-A-ON components used within each of the six pilots on the respective pilot platforms to meet the user requirements of the respective pilots stated in the deliverable D2.1. Section 2.2. also includes the technologies, protocols and security mechanisms adopted to achieve the interoperability.

Figure 4: Overview of the interactions between PHArA-ON components



2.2 Pilot high-level architectures

Subsections 2.2.1-2.2.6 provide an overview of the high level architectures of the Pharaon pilots indicating for each pilot with its respective platform the involved components at different architectural layers and interactions between them. In addition, the status of these interactions, the technologies, protocols and security aspects are described in the context of the PHArA-ON initial release.

2.2.1 Italian pilot

Figure 5 depicts the high-level architecture of the Italian pilot, while Table 3 presents the status of interactions between the components and adopted technologies.

Figure 5: Italian pilot high-level architecture

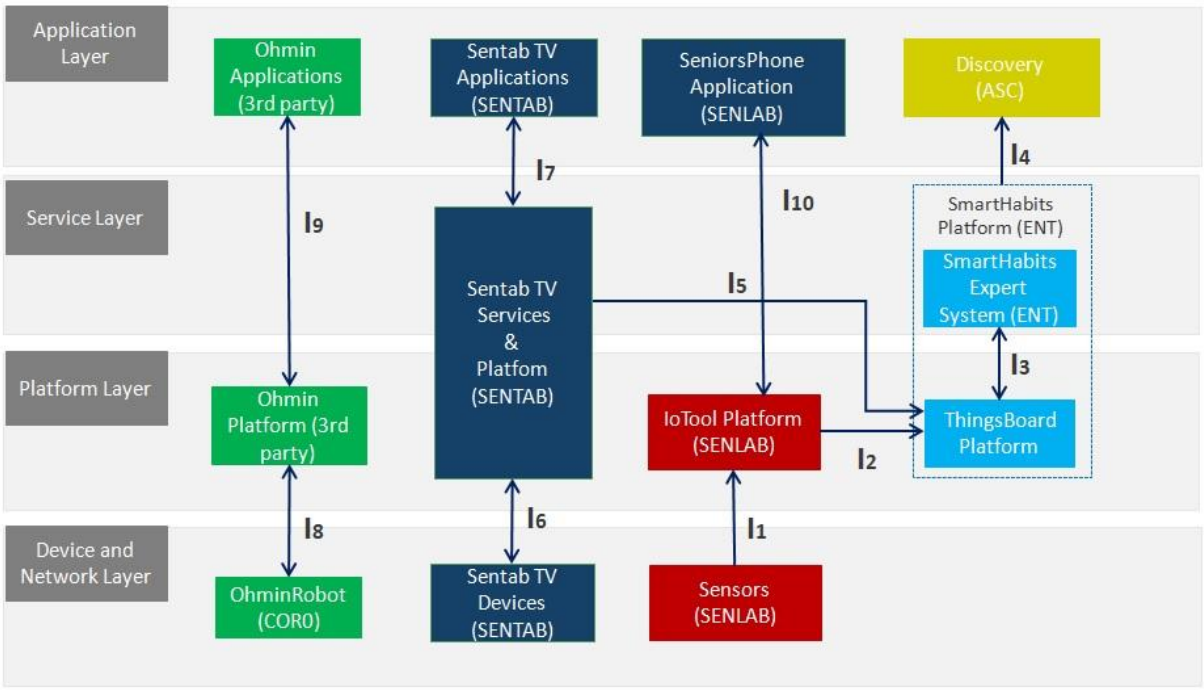


Table 3: Italian pilot - initial release of interactions between the components

Interaction	Status	Technologies and Protocols	Security
I_1	Implemented	bluetooth, wifi	BLE encryption
I_2	Implemented	http rest, mqtt	TLS, api tokens, IP filtering
I_3	Implemented	http rest, mqtt, amqp	Token, secure zoned, private network
I_4	Implemented	http rest	TLS, jwt tokens, IP filtering, rate limiting
I_5	To be integrated	HTTPS and JSON	SSL and API key
I_6	Pre-existing	HTTPS and JSON	SSL / login / tokens
I_7	Pre-existing	HTTPS and JSON	SSL / login / tokens
I_8	Pre-existing	HTTPS	SSL / login / tokens
I_9	Pre-existing	3rd party	3rd party
I_{10}	Implemented	https	Authentication (SSL), login, IP filtering

2.2.2 Slovenian pilot

Figure 6 depicts the high-level architecture of the Slovenian pilot, while Table 4 presents the status of interactions between the components and adopted technologies.

Figure 6: Slovenian pilot high-level architecture

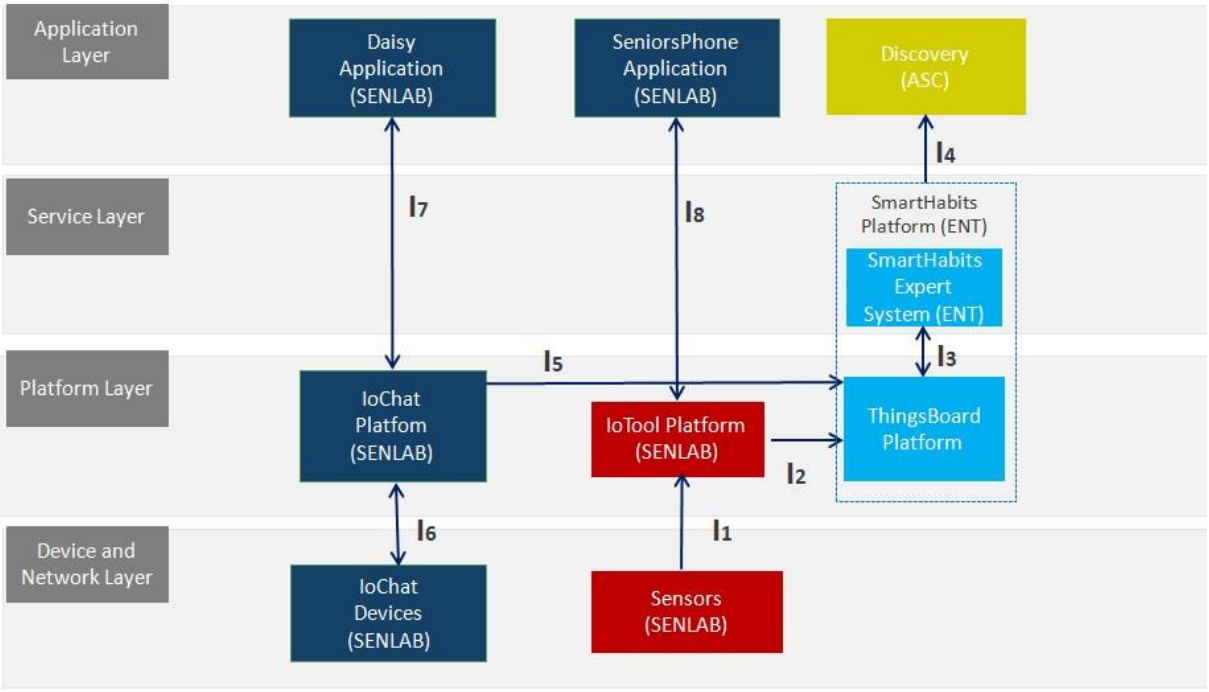


Table 4: Slovenian pilot - initial release of interactions between the components

Interaction	Status	Technologies and Protocols	Security
I_1	Implemented	bluetooth, wifi	BLE encryption
I_2	Implemented	http rest, mqtt	TLS, api tokens, IP filtering
I_3	Implemented	http rest, mqtt, amqp	Token, secure zoned, private network
I_4	Implemented	http rest	TLS, jwt tokens, IP filtering, rate limiting
I_5	Proposed	N/A	N/A
I_6	Pre-existing	N/A	N/A (NFC reader, microphone, camera on USB)
I_7	Pre-existing	https	Authentication (SSL), login, IP filtering
I_8	Implemented	https	Authentication (SSL), login, IP filtering

2.2.3 Portuguese pilot

Figure 7 depicts the high-level architecture of the Portuguese pilot, while Table 5 presents the status of interactions between the components and adopted technologies.

Figure 7: Portuguese pilot high-level architecture

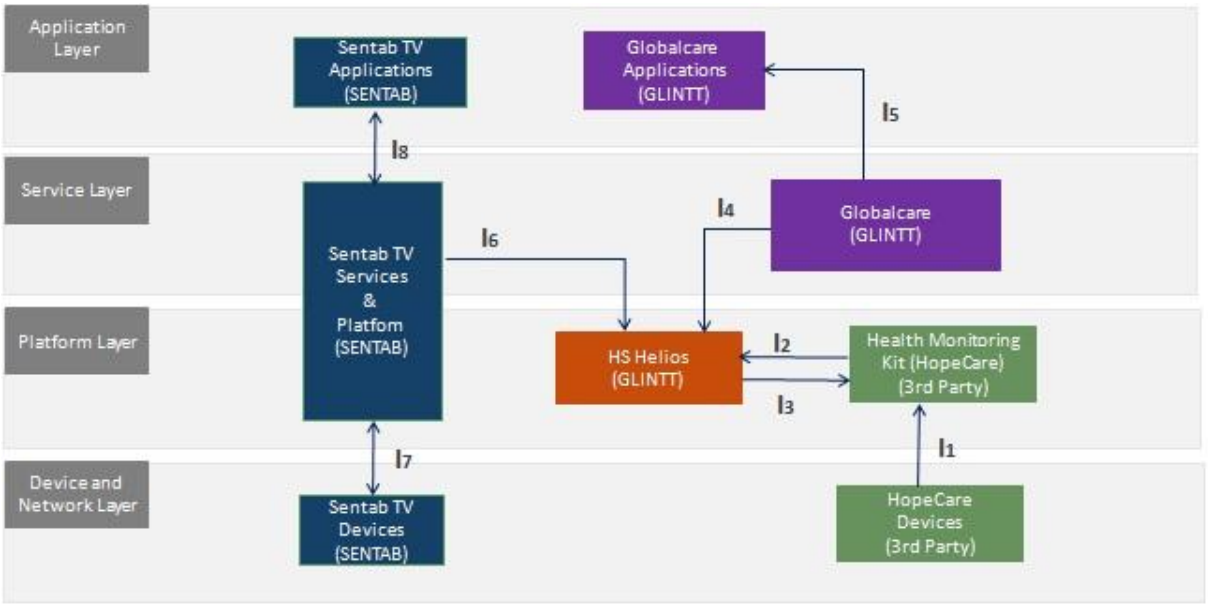


Table 5: Portuguese pilot - initial release of interactions between the components

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	Integrated	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

2.2.4 Andalusian pilot

Figure 8 depicts the high-level architecture of the Andalusian Pilot, while Table 6 presents the status of interactions between the components and adopted technologies.

Figure 8: Andalusian pilot high-level architecture

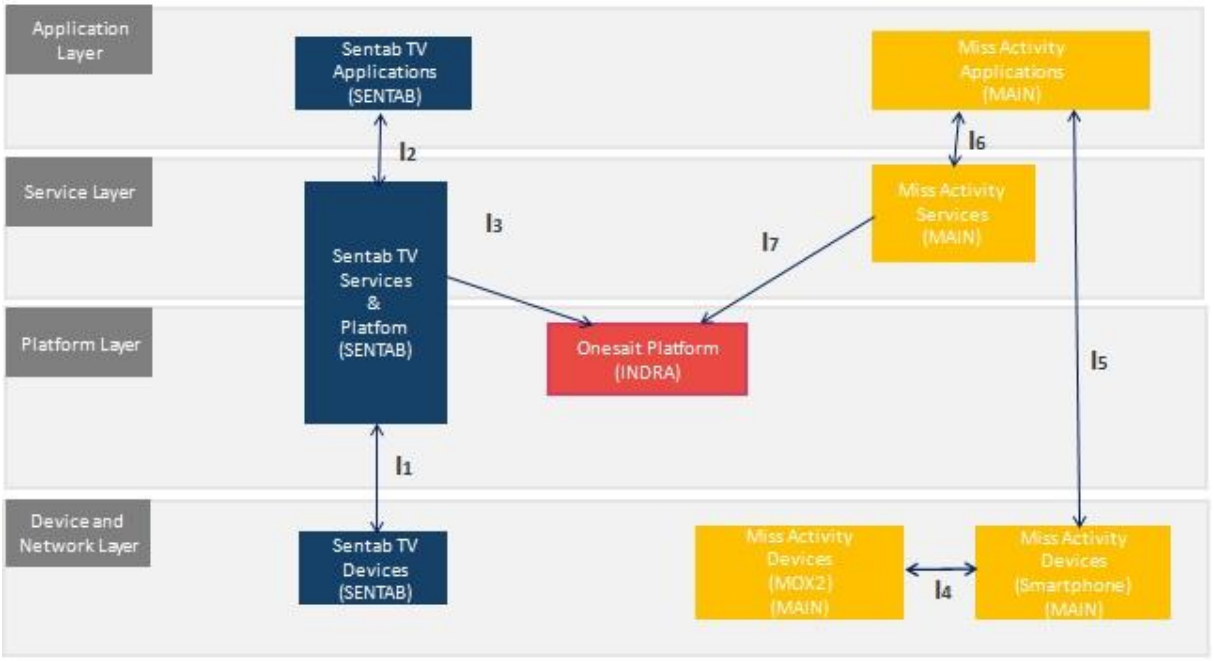


Table 6: Andalusian pilot - initial release of interactions between the components

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

2.2.5 Murcia pilot

Figure 9 depicts the high-level architecture of the Murcia pilot, while Table 7 presents the status of interactions between the components and adopted technologies.

Figure 9: Murcia pilot high-level architecture

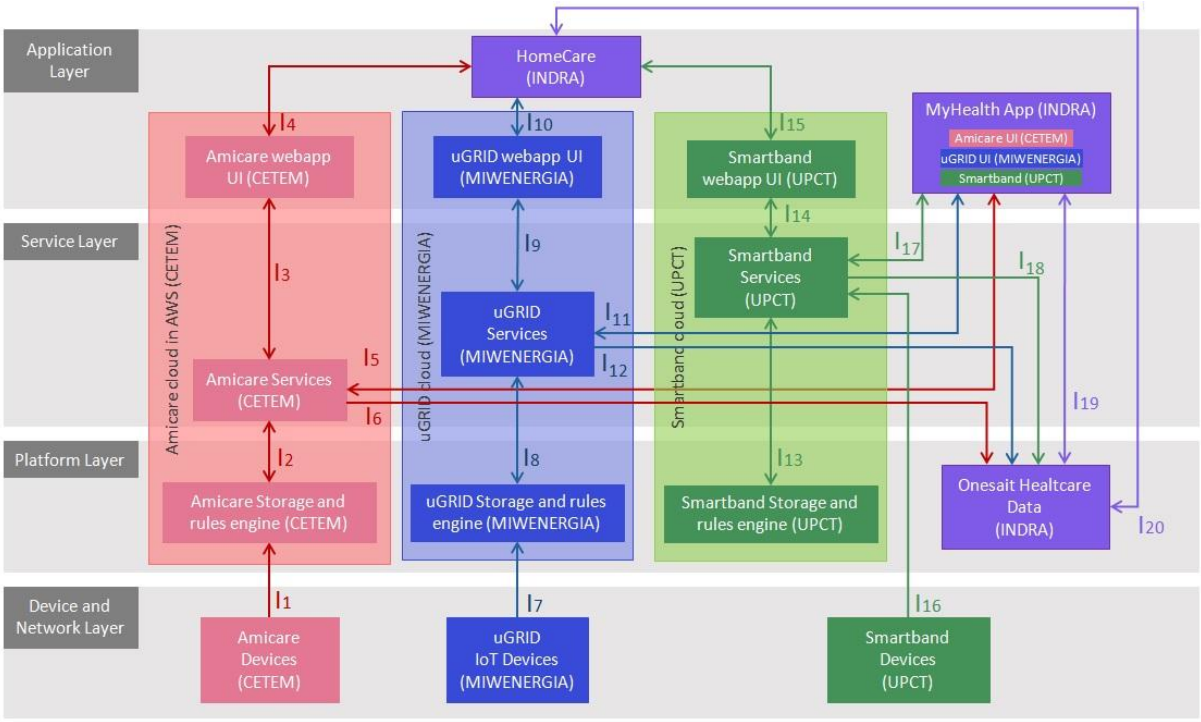


Table 7: Murcia pilot - initial release of interactions between the components

	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

2.2.6 Dutch pilot

Figure 10 depicts the high-level architecture of the Dutch Pilot, while Table 8 presents the status of interactions between the components and adopted technologies

Figure 10: Dutch pilot high-level architecture

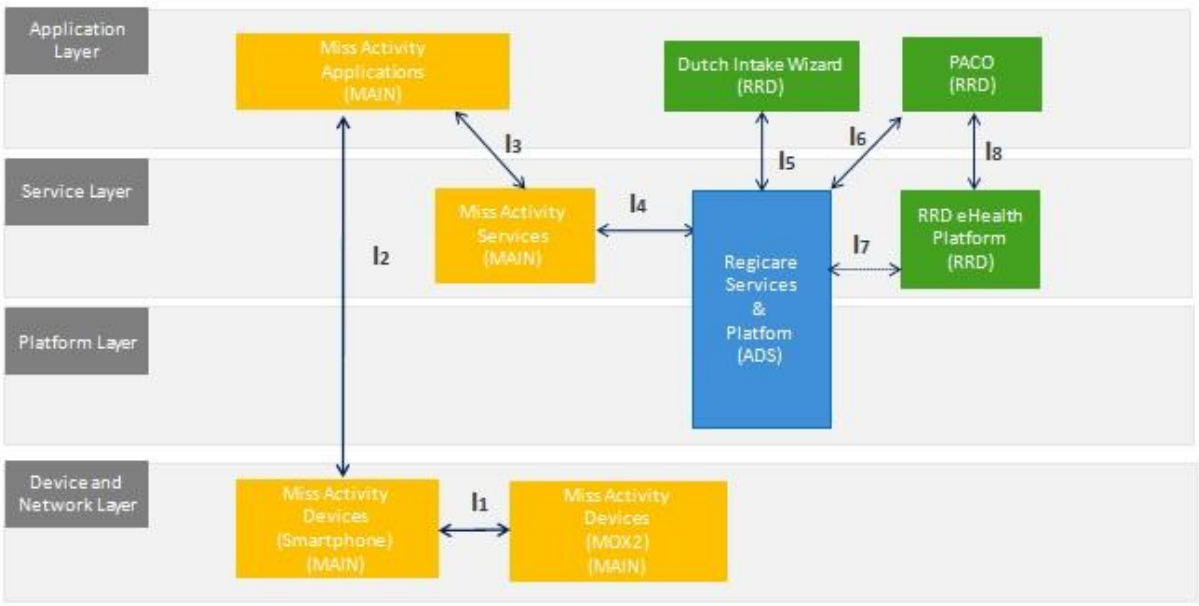


Table 8: Dutch pilot - Initial release of interactions between the components

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 callback		
I_8	Pre-existing	HTTP REST/JSON	Uses SSL, email/password login, JWT authentication

3 Infrastructure and Process for Integration and Test

3.1 Overview

As already introduced in D4.1, PHArA-ON adopts a Software Development LifeCycle (SDLC) methodology inspired by DevSecOps with a strong focus on automation and CI/CD pipelines. This has an influence on the WP5 processes, tools and activities that have been defined taking into consideration the DevSecOps principles.

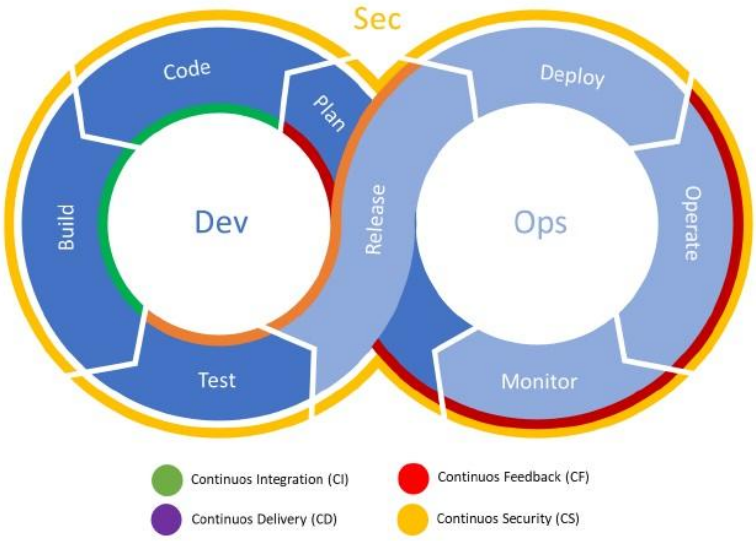
The main objectives of the DevOps methodology are (i) to save time on incident resolution, (ii) launch new features faster, (iii) reduce risks through process automation, (iv) increase the satisfaction of customers and developers at the same time. These objectives can be achieved taking into consideration the following key principles:

- *Automation*: a means to accomplish the work efficiently by automating large parts of the development operations and deployment process;
- *Iteration*: a means to rapid collect user feedbacks to accelerate the development process;
- *Self-Service*: a means to enable developers to deploy applications on demand by themselves in order to accelerate releases;
- *Continuous Improvement*: a means to make ongoing iterative improvements to the development process, workflow and toolset;
- *Continuous Testing*: a means to evaluate the quality of software at every stage of the software development life cycle by testing early and often;
- *Collaboration*: a means to increase the communication between teams in order to make development operations more rapid and effective.

DevSecOps (Development Security Operations) is an extension of the DevOps model aiming to establish a continuous integration and delivery cycle that combines application development with security and operations considerations. In particular, it ensures that a DevOps environment incorporates security best practises through testing, staging and deployment.

Figure 11 depicts the full SDLC used within the PHArA-ON project. The adoption of *Continuous Integration* (CI) and *Continuous Delivery* (CD) have changed how developers and testers ship software and provided a solution to the problems associated with constantly integrating new code. **Continuous Integration** focuses on blending the software work products of individual developers together into a repository. This can be done several times a day, with the primary purpose being to enable early detection of integration bugs while also allowing for tighter cohesion and more development collaboration. The aim of **Continuous Delivery** is to minimise the friction points that are inherent in the deployment or release processes. Typically, a team's implementation involves automating each of the steps for build deployments so that a safe code release can be done at any moment in time. It involves frequent, automated deployment of the master branch to a production environment following automated testing.

Figure 11: PHArA-ON DevSecOps lifecycle (from D4.1)



A good CI/CD pipeline will provide several benefits to the organisations: **reliability**, **compatibility**, **speed**, and **automation**. The overall strategy is to "**do more with less**" while also increasing output accuracy and efficiency. Moreover, teams can leverage their improved release cycles to refine their bug-fix workflows and improve the code quality. In short, the adoption of the CI/CD principle provided the following advantages:

- better code quality;
- faster bug fixing and release rate;
- traceability and reproducibility of changes and what's deployed in production;
- measurable progress;
- smaller changes, less disruptive deployments, reduced risk;
- standardisation of the process;
- fault isolation;
- cost reduction;,
- increased client satisfaction.

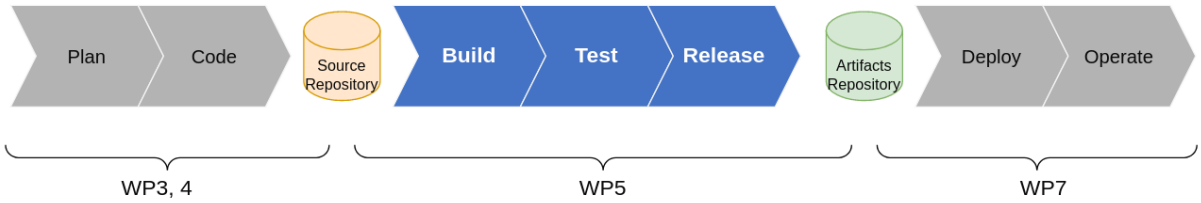
An important aspect of the CI/CD pipeline workflow is also the feedback loop it provides to make the **continuous** part work properly. The **Continuous Feedback** (CF) phase of the SDLC is necessary to improve the DevOps processes and future releases. In particular, WP5's activities will report feedback back to the development teams (mainly to WP3 and WP4).

Another important aspect of DevSecOps is the **Continuous Security** (CS). Security tools can be implemented in the **Build**, **Test** and **Release** phases of the CI/CD pipeline and the ones adopted in PHArA-ON are reported in section 3.6.

The activities of the SDLC are distributed across the project work packages. As shown in Figure 12, the *Plan* and *Code* phases are carried out in WP3 and WP4, where the actual development of the component is done. Work Package 5 is responsible for the *Build, Test* and *Release* activities and, finally, the deployment and the operation for the pilot execution is carried out within Work Package

7. The figure also highlights how WP5 receives as input the code ready to be built and tested and produces in output installable releases stored in a repository where they will be taken to deploy the software for pilot execution. The process established in WP5 to guide and control the *Build, Test* and *Release* phases are detailed in sections 3.2 and 3.6.

Figure 12: Plan and Code phases



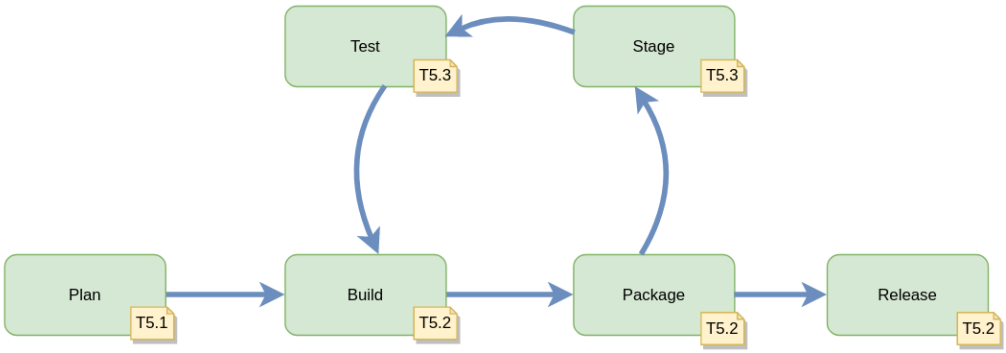
3.2 Pharaon CI/CD Process

This section focuses on the definition of a common release process that will be set-up and managed in the context of WP5 following the main principles and guidelines described in section 3.1.

We define the PHArA-ON release process as composed by six different phases:

1. **Plan**: in this phase the releases are planned in terms of time (when a release integration will start, when it should finish), content (which components and which functionalities will be included) and quality plans (tests to execute, validations to run, quality thresholds, acceptance criteria). Sections 3.4 and 3.6 describe in the detail this step and the work done and the decisions made so far;
2. **Build**: in this phase all the components are built from source code, source-code quality validations are executed (e.g., source code static analysis, unit testing) and binary artifacts are produced;
3. **Package**: in this phase the binary artifacts are packaged in deployable units (e.g., Docker Images, Helm charts, etc.,) and additional quality validations are executed on packages produced (e.g., Docker vulnerability scanning, Helm linting);
4. **Stage**: in this phase the packages from the previous step are deployed in the staging environment (see section 3.5) to verify their deployability and to prepare the environment for the testing phase. This phase includes also the deployment/update of all runtime dependencies and data needed to run the components successfully;
5. **Test**: in this phase the candidate release is tested executing the tests defined and agreed in the *plan* phase. The testing process, tools and templates are discussed in section 3.6;
6. **Release**: this phase includes all the finalisation activities to publish the new release and make it available for deployment in production environments. It includes update of software versions, update of change logs, publication of artifacts, production of release notes, update of documentation. Additional checks and validation of produced packages and documentation will be carried out (e.g., release checklists, documentation review).

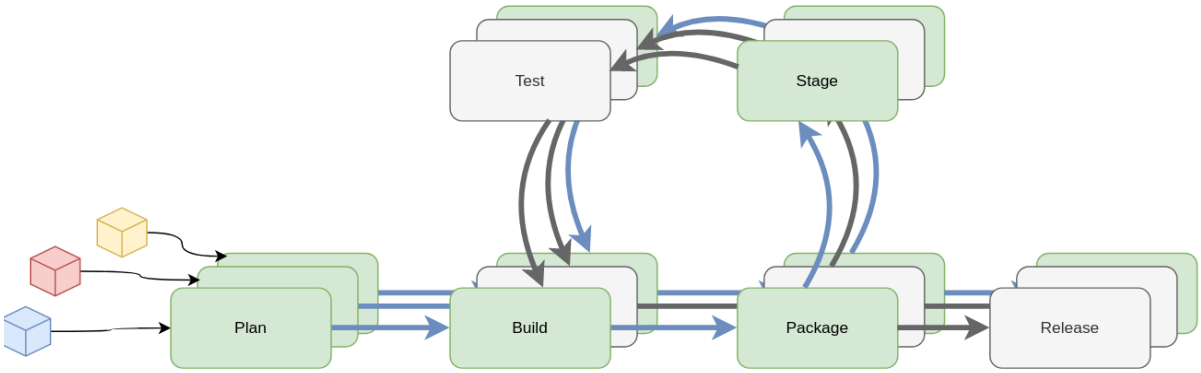
Figure 13: CI/CD process in PHArA-ON



The *build*, *package*, *stage* and *test* phases can form a loop that can be repeated multiple times during the same release cycle. Each *build* phase produces a release candidate version that is packaged, deployed in the staging environment and tested. If tests are not successful, any validation check (executed either in *build*, *package* or *stage* phases) fails or the software is below the quality thresholds, changes are requested to the software developers and the new version goes back to the *build* phase where a new release candidate version is produced restarting the loop until all the acceptance criteria are met. As stated in section 3.1 software quality is one of the main aspects that we want to guarantee in the project and, therefore, it is considered in each phase of the release process: quality is planned in the *plan* phase and continuously checked in all the other phases where each artifact produced is validated and tested.

The release process can be partially parallelized to reduce the overall time to create a release. PHArA-ON components are independent projects, with distinct codebases and without major interdependencies at build-time. This makes it possible, if needed, to run multiple instances of the release cycle process simultaneously, one for each component to release allowing a parallelization of the activities and a reduction of the overall release times. As a result, we can have during the preparation of a PHArA-ON release, multiple components that are at different stages of release and are ready at different moments (see picture Figure 14). Nevertheless, all of them will be part of the same PHArA-ON release.

Figure 14: Multiple release cycles



A limitation to the parallelization of the release might be posed by the testing phase. In fact, if two (or more) components need to be tested together (e.g., are used in the same test definition), the

release candidate version of all the components needs to be deployed in the staging environment at the same time before the test can be executed. As a consequence, the release of a component might be delayed until all the other components needed by the tests planned for that component are also in the staging environment.

3.3 CI/CD Tools

In the appendix section (A, B, C, and D) are described in detail the most important CI/CD tools, Continuous Inspection tools, Vulnerability Scanning tools and Artifact Repositories tools.

Considering the tools reviewed and the ones previously analysed in deliverable D4.1, we selected a set of tools that will be made available to the project to support the WP5 activities. The main criteria that guided the selection of tools were:

- Open source tools have been preferred over proprietary tools; tools free to use for research purposes have been preferred;
- Possibility to install the tools on premises to meet the privacy requirement expressed by some technologies in the PHArA-ON project;
- Active community behind the tool in order to easily find documentation, tutorial, support and extensions;
- Possible and seamless integration with other tools already in use in the project to support the PHArA-On SDLC (in particular in WP3, WP4) or other tools selected in WP5;
- Tools that cover more needs and are easily customizable and adaptable to the project needs have been preferred;
- Preference of a tool over other alternatives given by previous experiences.

These tools will be used in conjunction with the tools already set-up and used by WP3 and WP4 (e.g., Gitlab.com for source code, JUnit for unit testing).

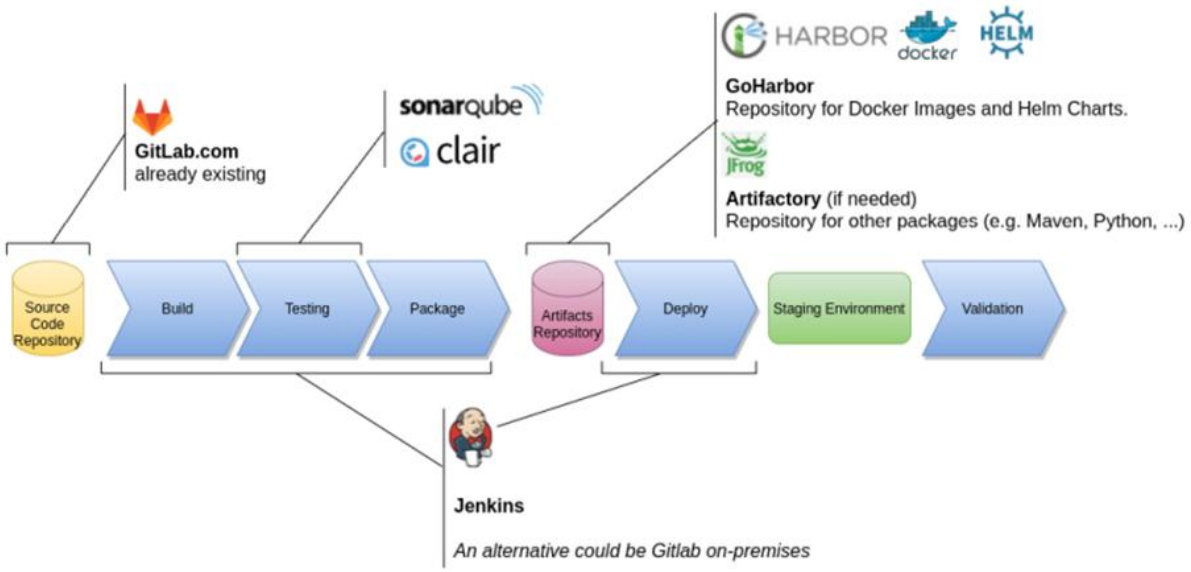
As shown in Figure 15, different tools have been selected to support different activities/phases of the PHArA-ON integration and release cycle. They are:

- **Jenkins**: for the automation of CI/CD pipelines. This tool was selected since it meet all the criteria we used: it is open-source, can be installed on premises (privacy), has very active community, is plugin-based and easily extensible, several plugin exists for multiple type of jobs (e.g., quality reports, deploy on docker). It is also already being used by some of the technologies of the PHArA-ON project. This will reduce the effort needed to migrate to the new project-wide solution.
- **SonarQube Community:** for the continuous inspection of code quality. There are multiple reasons why this tool has been selected over the analysed alternatives. The tool offers several types of analysis of the source code and several dashboards to control the quality of the software components, make aggregations, comparisons and historical analysis. It supports several programming languages and can be easily extended through plugins. It has a seamless integration with Jenkins and requires a minimal set-up and configuration effort. It offers functionalities (e.g., quality gates) that integrate well and can enhance the testing and release processes in PHArA-ON.
- **Clair**: for vulnerability scanning of Docker images. Clair is an open source project that analyses Docker Images based on vulnerability definitions from various sources (e.g., CVE

database[1]). With respect to other alternatives, it has a larger list of known vulnerabilities, updated more frequently and seamlessly integrates with GoHarbor. Clair, however, has not a ready-to-use integration with Jenkins. For this reason, for the vulnerability scanning at build time we could use another tool named Grype[2] (from Anchore) that has a better integration with Jenkins.

- **GoHarbor**: as a repository for deployable packages. It is the only solution analysed that is open source, supports both Docker Images and Helm charts and has a fine grained role-based policies and access control. It also integrates seamlessly with Clair for vulnerability scanning. Finally, It also is a project in the CNCF[3] and very popular in the Kubernetes community.
- **Artifactory OSS**: as a repository for generic binary artifacts. Since GoHarbor only supports Docker Images and Helm charts, in case additional binary types needs to be stored for some technologies (e.g. jar/war files, PyPI packages, deb/rpm packages), the open source version of Artifactory can be taken into consideration.

Figure 15: Pipeline and tools in PHArA-ON



The tools described above will be managed by WP5 and offered at the project level to automate execution of the release process activities. However, some PHArA-ON teams are already using alternatives to the ones selected for multiple reasons:

- The components are not open-source and the code base cannot be shared neither publicly nor within the PHArA-ON consortium. In this case, private, on-premises continuous integration servers are used to build the code;
- The components have a licence that prohibits sharing binaries. In this case artifacts are stored in private repositories;

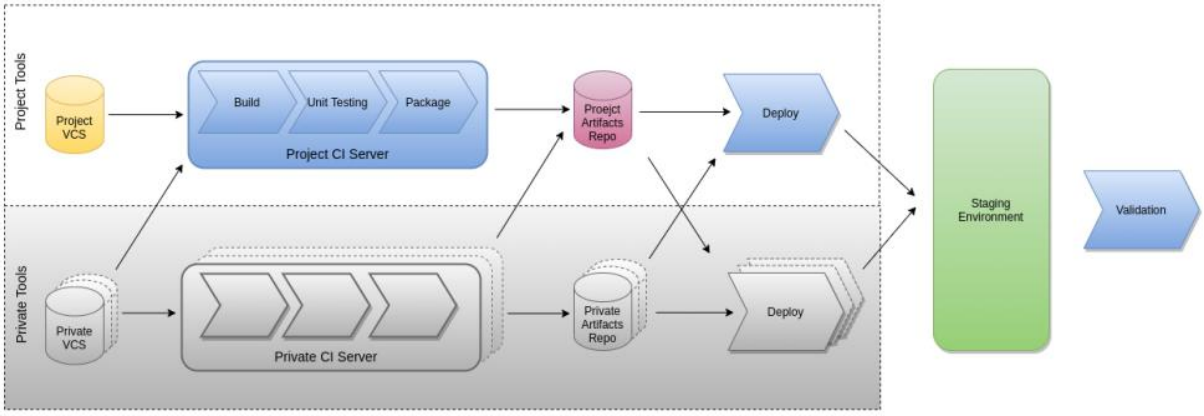
[1] https://cve.mitre.org/cve/
[2] https://github.com/anchore/grype/
[3] Cloud Native Computing Foundation https://www.cncf.io/

- Mature technologies already used their own CI/CD tools before PHArA-ON project and continued to use them also in PHArA-ON;
- No project-level solutions were available during the first phases of the project. This forced all technology providers to organise the release of their components autonomously.

The last two points will be gradually removed by migrating all the components to use the new project-level tools in order to streamline as much as possible the pipelines to simplify the management of the release and to share common practices and knowledge.

There are cases, however, of components that cannot be integrated using project-level tools, for instance because they are not open source and/or cannot share binary artifacts. Private integration environments will be allowed for these components, however they will need to follow the PHArA-ON release process to guarantee the same level of quality of the other components. It will be possible also to have "hybrid" pipelines that use private and project-level tools. Figure 16 depicts the different possible pipelines built using partially project-level tools and partially private tools. For instance, a closed-source component could store its source code in a private repository (not using the project-wide GitLab) and use a private Jenkins instance to build and execute source code analysis. At this point, the produced artifact is published in the Harbor repository (project-wide tool) and from there it can continue the release process using the project-wide tools.

Figure 16: "Hybrid" PHArA-ON CI/CD approach



3.4 Release Plan

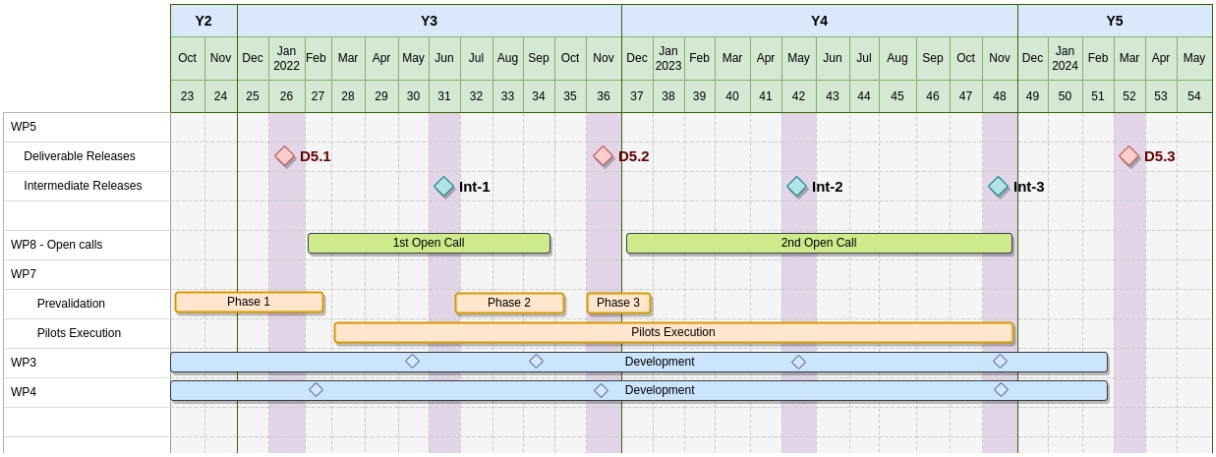
Figure 17 reports the estimated dates for the next releases of the PHArA-ON ecosystem. As described in the DoA, three main PHArA-ON releases are expected. The first one at **M26** (described in this document). An Intermediate Release is expected at **M36** and the Final Release is expected at **M52**.

Besides these main releases, some intermediate releases are expected for bug fixing but also to provide new features (if needed).

The intermediate release at M31 (**Int1**) will include the technologies available from the 1st Open Call. The Intermediate release at M36 will include bug fixing and new functionality (if any) considering the results from the pre-validation Phase 2. The intermediate releases at M42 (**Int2**) and M48 (**Int3**) will include bug fixing and the new versions of components released in the WP3/WP4 milestones.

According to development activities the release plan will be updated (if needed) in the next WP5 deliverable.

Figure 17: Release Plan



3.5 Staging Environment

A *staging environment* is a nearly replica of a production environment for software testing in order to ensure that the software works as designed, i.e., the features and functionalities have been developed according to the specifications. In particular, the staging environments are made to test codes, builds, and updates to ensure quality under a production-like environment before application deployment.

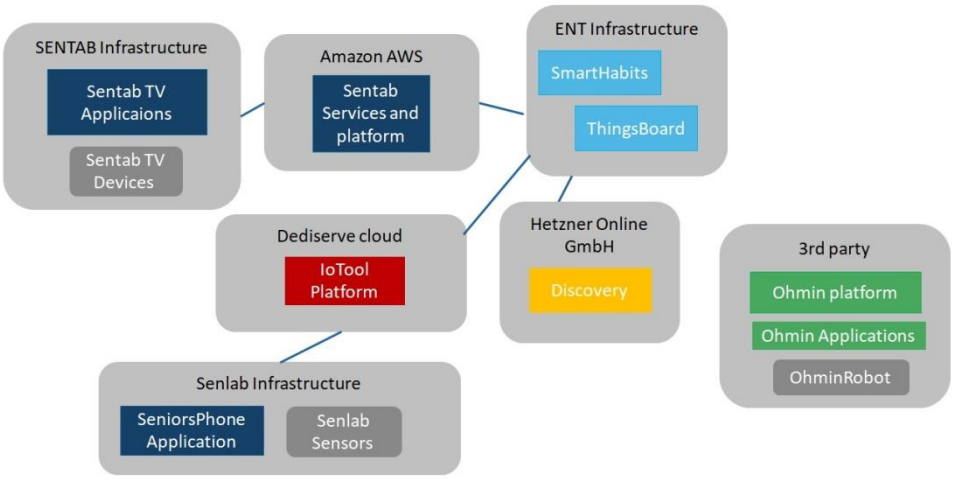
In this Initial Release of the PHArA-ON ecosystem, the staging environment is characterised by a pilot environment supported by multiple physical infrastructures where the technologies (components and services) are deployed in order to test their integration and functionalities before the deployment in production. In some cases the hosting infrastructures are owned by the technology providers, in other cases the services are hosted by 3rd party providers (e.g., AWS). 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.

3.5.1 Italian pilot

Figure 18 provides an overview of the physical hosting infrastructures of the staging environment for the deployment of the technologies involved in the Italian pilot. SENTAB, ENT and SENLAB are the owners of the infrastructures to deploy and test the Sentab TV applications, the SmartHabits/ThingsBoard component and the SeniorsPhone application, respectively. The Sentab services are hosted in AWS. The IoTool Platform is hosted in the *DediSERVE*[4] cloud, while the Discovery component is hosted in *Hetzner Online GmbH*, a company and data centre operator based in Gunzenhausen (Germany). Also the OHMNI robot is a 3rd party solution. The equipments for the collection of the data (i.e., Sentab TV Devices and Senlab Sensors) are part of the technology providers infrastractures.

[4] https://www.dediserve.com/

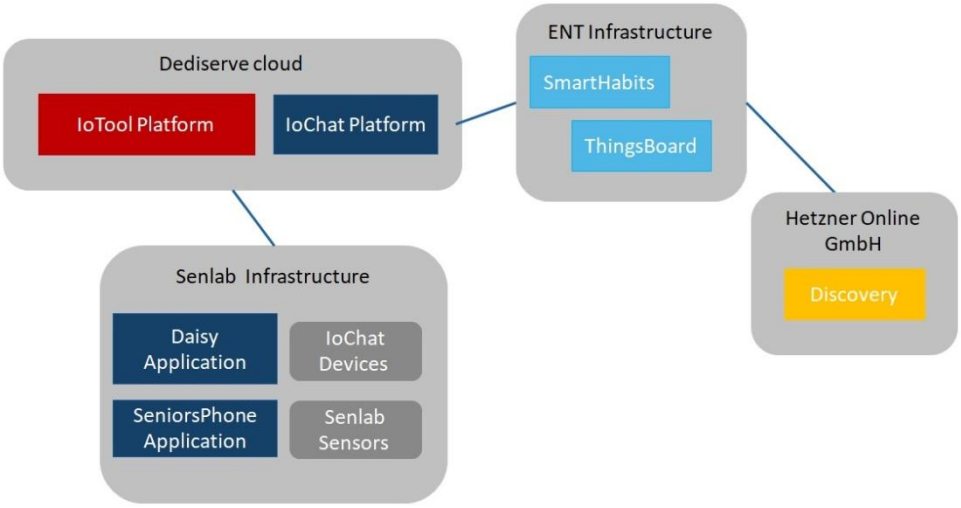
Figure 18: Overview of the Staging Environment for the Italian Pilot



3.5.2 Slovenian pilot

Figure 19 provides an overview of the physical hosting infrastructures of the staging environment for the deployment of the technologies involved in the Slovenian pilot. The hosting infrastructures are the same of the ones involved in the Italian pilot. Compared to the latter pilot, the *DediSERVE* cloud provider also hosts the IoChat Platform component, while the SENLAB infrastructure hosts the Daisy Application along with the IoChat Devices.

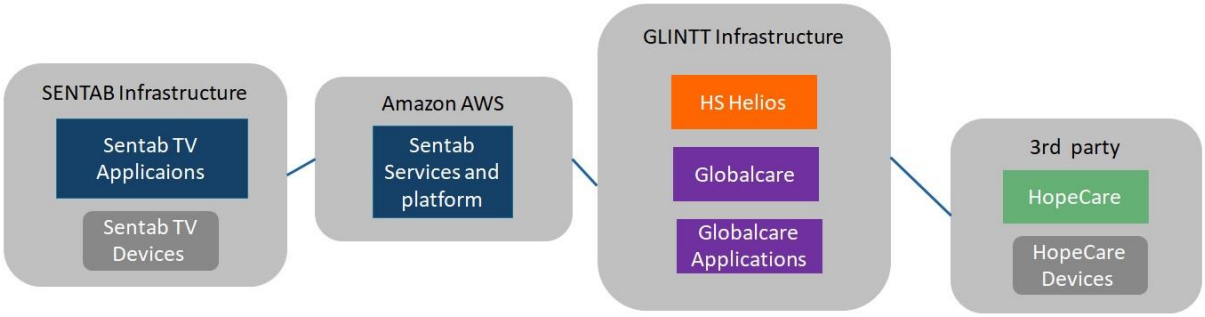
Figure 19: Overview of the Staging Environment for the Slovenian Pilot



3.5.3 Portuguese pilot

Figure 20 provides an overview of the physical hosting infrastructures of the staging environment for the deployment of the technologies involved in the Portuguese pilot. SENTAB and GLINTT are the owners of the infrastructures to deploy and test the Sentab TV applications and the Globalcare and HS Helios services, respectively. The Sentab services are hosted in AWS, while the HopeCare component and equipments are provided by a 3rd party.

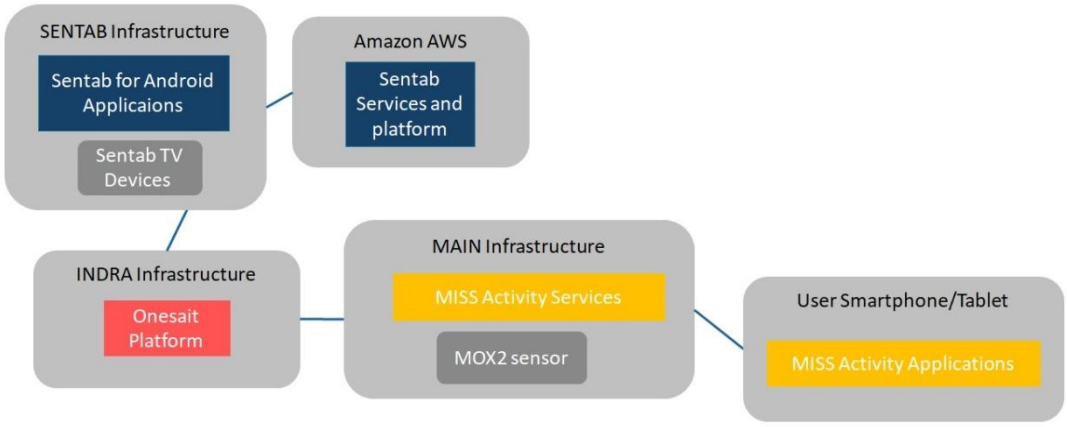
Figure 20: Overview of the Staging Environment for the Portuguese Pilot



3.5.4 Andalusian pilot

Figure 21 provides an overview of the physical hosting infrastructures of the staging environment for the deployment of the technologies involved in the Andalusian pilot. SENTAB and INDRA are the owners of the infrastructures hosting the Sentab for Android Applications and the Onesait Platform, respectively. The Senbtab services are hosted in AWS as for the Italian and Portuguese pilots. The MISS Activity Services are hosted in the infrastructure provided by MAIN along with the MOX2 sensor. The MISS Activity Application, instead, is installed on the user's smartphone, but could also be installed on the user's tablet.

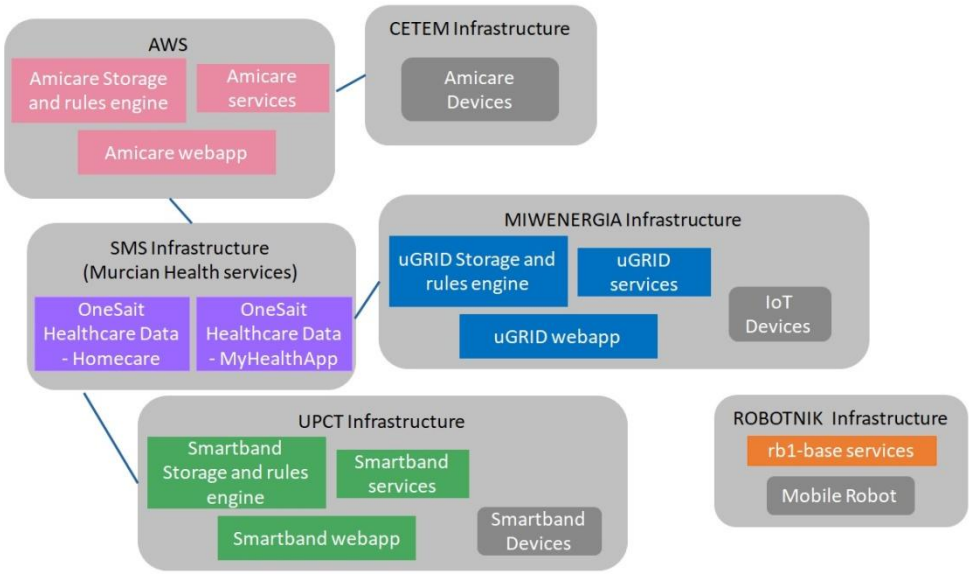
Figure 21: Overview of the Staging Environment for the Andalusian Pilot



3.5.5 Murcia pilot

Figure 22 provides an overview of the physical hosting infrastructures of the staging environment for the deployment of the technologies involved in the Murcia pilot. MIWENERGIA, UPCT, and ROBOTNIK are the owners of the infrastractures provided to host the uGRID, Smartband and rb1-base services, respectively, along with the need equipments (i.e., IoT devices, Smartband devices and the mobile robot). The CETEM infrastructure hosts the Amicare devices, while the Amicare services and backend are hosted in AWS. The OneSait Healthcare Data - Homecare and MyHealth Application are hosted in the Murcia Health services (SMS) infrastructure.

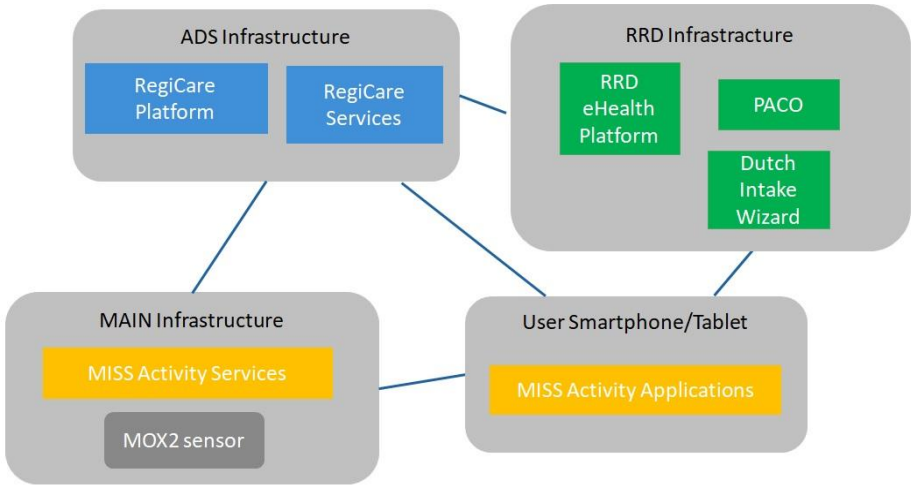
Figure 22: Overview of the Staging Environment for the Murcia Pilot



3.5.6 Dutch pilot

Figure 23 provides an overview of the physical hosting infrastructures of the staging environment for the deployment of the technologies involved in the Dutch pilot. ADS, RRD, and MAIN are the owners of the infrastractures provided to host the RegiCare platform and services, the RRD eHealth Platform/PACO/Dutch Intake Wizard and MISS Activity services, respectively. The MOX2 sensor is part of the MAIN infrastructure, while the MISS Activity Application is installed on the user's smartphone or tablet.

Figure 23: Overview of the Staging Environment for the Dutch Pilot



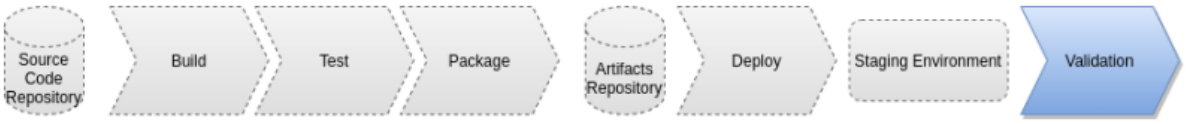
3.6 Testing process and tools

This section presents the project-wide testing and Quality Assurance (QA) processes adopted in the PHArA-ON project and executed in WP5. It is worth mentioning that the following processes are still not fully implemented in the project and the test results reported in Chapter 4 are based on test suites and tools owned by the development teams of the single components and executed independently for each component without the coordination of a testing process at project level.

QA is a systematic process of determining whether a product or service meets expected quality. It helps an organisation to create products and services that meet the needs, expectations, and requirements of customers, by reducing the number of errors and faults in the product and maintaining the product quality to a given level. Therefore, implementing QA in the CI/CD pipelines is the final step in creating a system that truly provides the maximum amount of benefits in the minimum amount of time for the DevOps teams. Software Quality Assurance (SQA) systematically finds patterns and the actions needed to improve development cycles. SQA has become important for developers as a means of solving potential errors before they occur in the production environment, saving development time and expenses.

The main objective of Task 5.3 of WP5 is to ensure the quality of the PHArA-ON components by performing testing activities to validate the PHArA-ON ecosystem components and basic building blocks made available by Task 5.2. The Validation phase of the PHArA-ON release cycle is where most of the validation activities of the PHArA-ON ecosystem are executed (see Figure 24). However, quality is ensured in all phases of the cycle running multiple validation and testing tools against the artifacts of each phase (e.g., linting of binaries generated by Build phase, scanning of packages generated in the Package phase) In particular, the testing activities are based on multi-level testing and consider both functional and not-functional dimensions.

Figure 24: Validation activities



Indeed, there are various testing levels as well as techniques based on the desired scope of testing that can be applied to a CI/CD pipeline to ensure more coverage. However, well-functioning unit and integration tests form the foundation of a CI/CD pipeline, which can be complemented by functional tests of the system as a whole to test the product from the end user's point of view.

In the PHArA-ON project, the following kinds of tests will be performed:

- **Functional testing**: to validate the software system against the functional requirements and specifications. In particular, the purpose of this kind of testing is to validate the functionalities of a component by providing appropriate inputs and verifying the outputs according to the functional requirements that are specified in D2.1 of Pharaon;
- **Non-functional testing**: a kind of testing to validate the non-functional requirements of the system, such as performance and stability, which are termed as **quality requirements** in

PHArA-ON. In particular, the quality requirements are specified in D2.1, and the quality requirements concerned with the security aspects are separately in the deliverable D10.4;

- **Testing of emotional requirements:** as a novelty, in D2.1 emotional requirements were specified for all the pilots of PHArA-ON. Testing against emotional requirements will be specified in the deliverables D7.3 and D7.4.

Generally, functional testing is defined with the following workflow (steps):

- Determine what functionalities need to be tested;
- Create input data for functionalities to be tested according to specified requirements;
- Check output parameters according to specified requirements;
- Execute test cases;
- Compare the actual output from the test with the foreseen output values to check if the software is working as expected.

Unlike functional testing, non-functional testing is not defined through steps of a workflow but is based on quality requirements. These requirements include the performance output that is expected from the software under test. Quality requirements ensure high satisfaction from potential customers.

Table 9 summarises the differences between functional testing and non-functional testing and lists the ones relevant (that will be implemented) for the PHArA-ON project.

Table 9: Comparison between functional testing and non-functional testing

	Functional testing	Non-functional testing
Objective	Verify software actions	Verify performance of the software
Target	On functional requirement	On quality requirement
Functionality	What the product does	How the product works
Execution	Before non-functional testing	After functional testing
Types	• Unit testing • Integration testing • End-to-end testing	• Performance testing • Reliability testing • Security testing • Scalability testing

Another important aspect to consider is to automate the testing process as much possible wherever appropriate with the help of appropriate tools. For the automation of both functional and non-functional testing, software that already includes testing procedures or specific tools could be used, such as *JMeter*, *Postman*, and *Coverage.py*. The "*Appendix A.0.3 SDLC tools - Test phase*" of D4.1 includes a full list of software tools for testing in different programming languages.

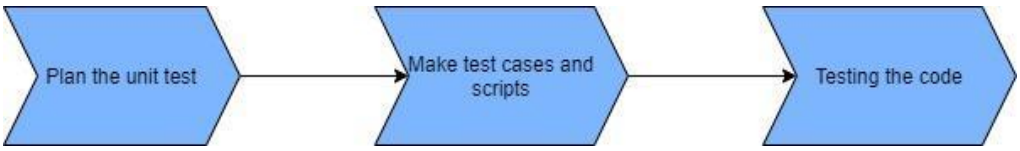
3.6.1 Functional testing

Unit testing is a type of functional testing and the first level of testing, followed closely by the integration tests. Having robust unit tests paves the way to implement quality code throughout the entire development life cycle and the product itself because they catch bugs earlier in the process, where it's cheaper and easier to fix.

Unit and integration tests reside at the core of a CI/CD pipeline, these types of testing are focused on validating that small units of code execute as expected so a good suite of unit tests is important, its coverage should be kept as high as possible and accurate. New tests need to be developed and/or updated for each new feature or unit of code implemented or modified, and also, failing tests should fail for the correct reasons, meaning a bug was found.

In PHArA-ON unit testing will be used to validate the functionalities provided by the PHArA-ON components implemented in the technical work packages, according to the user and pilot requirements. In particular, as depicted in Figure 25, the unit testing will be performed in three steps: in the first step the unit test is prepared and reviewed, in the next step the test cases and scripts are defined and, finally, the code is tested.

Figure 25: Unit testing steps



The unit tests will be executed as frequently as possible (e.g., as part of a build process) and, even if the tests can be made manually, automated testing is the preferred approach since it saves the developer a lot of time and effort. To this aim, several testing frameworks to develop test cases are available. Each of the PHArA-ON's components involved in the testing activities will adopt the more appropriate one according to the different programming languages.

In addition to the unit testing, the *integration testing* is used to validate how the software modules are combined as a group aiming to ensure that the components properly interact and correctly exchange data. To this objective, the integration tests will be executed considering pairs of interacting components. The pairs of components involved in the integration tests are identified by examining the interactions depicted in the pilot architecture diagrams in 2.2.

Table 10 defines the template that will be used to document the integration tests.

Table 10: Integration testing Template

Integration Testing Scenario ID	
Description	Test case description
Prerequisite	Test case prerequisite
Postrequisite	Test case postrequisite
Test Execution Steps	Description of test execution steps
Test Results	Description of the test results
Test Comments	Comments on the performed test

Finally, the e*nd-to-end (e2e) testing* is used to test the functionalities of a system in the situation similar to that of using a real product to simulate the user scenario from the beginning to the end. Therefore, this kind of testing will be conducted based on the use case scenarios defined in WP2.

3.6.2 Non-functional testing

As already mentioned *Non-functional testing* checks all the aspects not covered in functional tests. It includes the performance, usability, scalability, and reliability of the software.

Performance testing is a process to test the speed and the response time of services in order to eliminate bottlenecks in the software application. Typically this test is performed through multiple requests per second and to observe the final results (i.e., if the application responds quickly, and is up and running). The same process is also used for *reliability testing* and *scalability testing* (i.e., to determine the maximum load handled by the software application).

The *security testing*, as described in D4.1, is another important aspect of DevSecOps to manage the **Continuous Security** (CS). DevSecOps have introduced a new set of tools to automate security and incident response in the process and help us to implement the following features:

- **Dependency Scanning**: to find security vulnerabilities automatically in the software dependencies while you're developing and testing applications;
- **Container Scanning**: to analyzer container images for known vulnerabilities, secrets keys (private keys), compliance checklist, and malware;
- **Static Application Security Testing or (SAST)**: to allow developers to find vulnerabilities in the application source code earlier in the SDLC (also known as a "white box testing");
- **Dynamic Application Security Testing or (DAST)**: to find security vulnerabilities and weaknesses in typically web apps (up and running) by feeding malicious data to the software, without a check into the internal source code (also known as "black box testing");
- **Secret Detection**: to check an unintentionally commit of secrets and credentials data to repositories;
- **Licence Scanning**: to detect the licences in use in the application.

The types of not-functional tests listed above are the main ones that we plan to implement in the project to verify not only that the software works as expected but also the way (and how well) it works. However, depending on the specificity of the components in the next releases and needs of the project new or different tests could be taken into consideration.

To conclude, for both functional and non-functional testing is true that in a CI/CD pipeline most of the tasks should be automated, however, it is a human being or a team of people the ones in charge of deciding which tests should be part of the CI/CD pipeline based on the business rules/knowledge, the use of the features provided by the app under test, the knowledge of the system and last but not least the expertise of the engineers.

Although automated testing is very useful, it’s not enough to guarantee the quality of a software product, there are some features that cannot be covered by automated testing, such as exploratory testing, usability testing, and user experience testing, this should also be done by a trained engineer, as well as other testing types, so both manual testing and automated testing should be set in place to help guarantee the good quality of a software product.

4 Pharaon Initial Release

This section describes the Initial Release of the PHArA-ON ecosystem providing a snapshot of the functionalities implemented by each individual component involved in the PHArA-ON project at M26. In particular, for each of the component reported in Table 2, the next subsection describes (i) the release notes (i.e., the released versions with the provided functionalities), (ii) the licence information, (iii) the links to relevant resources (e.g., source code repository, binary packages, documentation, web site), (iv) the information of main contact points and (v) the test reports (if available).

4.1 Amicare

AmiCare is a non-invasive system that monitors older-adults' daily habits and greatly contributes to the peace of mind of relatives and carers thanks to the power of IoT. Out-of-sight textile sensors are wirelessly and safely connected through the cloud to any smartphone app with a pre-registered user.

Release notes	At the time of writing two versions of the Amicare technology have been released. Version V1.0, released in September 2021, provides the following new functionalities: ● WebApp interface ● Android app interface ● Improved function to occupancy detection ● Improved function to movement detection Version V2.0, released in December 2021, provides the following new functionalities: ● WebApp interface for the integration with OHC ● Android app interface for the integration with MyHealth ● Amicare FW for remote device update Moreover, the following issues have been solved: ● Change in tab name "Usuarios" by "Actividad" ● Updated Amicare light feature so it's not annoying
Licence	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
Testing reports	Amicare has been evaluated in the functional test of the first phase of the pre-validation.

4.2 Discovery

Discovery is a web application and service, which is the interface to caregivers and informal caregivers.

Release notes	The new functionalities provided by Discovery in the context of the PHArA-ON initial release are:
	● Integrating SmartHabits and IoTool (services from other project partners) by creating specific Leecher microservices
	● Configuring dashboards for new data points
	  ○ Environmental sensors (temperature, humidity)
	  ○ Static sensors (motion)
	  ○ Dynamic sensors (heart rate, calories, steps)
	● Integrating new kinds of dashboards
	● Tailoring web application to the PHArA-ON project
	The new provided content is:
	● User friendly dashboard for caregivers
	  ○ Creating design/mock up for the dashboard
	  ○ Implementing design/mock up
	  ○ Profile management component
	  ○ User management, authorization and authentication via KeyCloak added to dashboard and combined with Discovery
	  ○ Caregiver configuration
	  ○ Informal caregiver configuration
	  ○ Care receiver configuration
	  ○ Sensor configuration
	Moreover, the following issues have been solved:
	● Integration issues with SmartHabits and IoTool (unconstant and unexpected sensor values, general integration issues)
	● Reducing user errors from prevalidation (e.g. removing trailing and leading whitespace etc)

Licence	Proprietary, sensor Leechers (Apache 2)
Resources	● Source code of the Leecher microservice is hosted at *https://gitlab.com/pharaongroup/discovery-dataleecher* ● Development status and requirements of Discovery is available at *https://gitlab.com/pharaongroup/discovery* ● Source code of the dashboard is hosted at https://gitlab.com/pharaongroup/carer-dashboard ● The service is accessible at ○ https://dashboard-pharaon.ascora.eu/ (ITA pilot UP) ○ https://dashboard1-pharaon.ascora.eu/ (SLO pilot) ○ https://dashboard2-pharaon.ascora.eu/ (ITA pilot CSS)
Support and contact points	● Nicolas Mayer (ASC) - mayer@ascora.de ● Support requests and issues can be reported in the prevalidation at https://gitlab.com/pharaongroup/pre-validation/-/issues ● Support requests and issues can be reported in general at https://gitlab.com/pharaongroup/discovery/-/issues
Testing reports	Discovery has been evaluated in the functional test of the first phase of the pre-validation.

4.3 Dutch Intake Wizard

Web application that guides the user through the intake by means of a dialogue. The application consists of a front-end web application and a back-end Java Tomcat web service. The web service makes use of the WOOL platform, an open source dialogue platform co-maintained by RRD.

Release notes	The new functionalities provided by the Dutch Intake Wizard in the context of the PHArA-ON initial release are: ● Tailoring Wizard web application to the PHArA-ON project: ○ Per-node avatar images can be defined through a CSV file ○ Dialogue file names can be defined via a GET parameter; between-dialogue jumps are supported in the browser ○ Initial variables can be passed via GET parameter ○ After dialogue is finished, it is possible to redirect to an URL, adding the final values of the variables as a GET parameter ○ Restyled layout to be more compact ○ Custom CSS can be defined

	The new provided content is: ● Developing WOOL dialogues for conversational agent for the intake questions ● Creating design for the virtual agent ● Implementing designs for the virtual agent with different facial expressions Moreover, the following issues have been solved: ● Some of the text was not visible and it was not possible to scroll it up and down ● On the dialogue end node the image of the virtual agent suddenly changes into the default character
Licence	The Wool platform is released under the MIT licence
Resources	● Source code of this component is hosted at https://github.com/woolplatform/wool
Support and contact points	● Boris van Schooten (RRD) - B.vanschooten@rrd.nl ● Support requests and issues can be reported at https://gitlab.com/pharaongroup/dutch-intake-wizard/-/issues
Testing reports	The integration has been manually tested and successfully validated during the first and second pre-validation phases.

4.4 Globalcare

Globalcare is a Hospital software designed and developed by Glintt to cover virtually all hospital activities and meet the needs of all hospital teams.

Release notes	At the time of writing two versions of the Globalcare technology have been released. Version 21R09, released in September 2021, brings initial evaluation interviews and weekly organisation planning. In particular, the new provided functionalities are: ● Initial evaluation interview ● Scheduling the workplans ● Intervention Plan (create, update and execute)

	• Changes and adaptations to make GC suitable for Social Care workers and Homecare assistants
	• Ability to consult the historic information registered during the homecare visits
	
	Version 21R12, released in December 2021, brings integration with health monitoring devices.
	In particular, the new provided functionalities are:
	• Updated layout to display integrated data
	• Ability to open a 3rd party App (Hopecare) and automatically collect data from 5 health monitoring devices:
	  ○ Glucometer
	  ○ Weight Balance
	  ○ Oximeter
	  ○ Thermometer
	  ○ Blood pressure measurement device
	• The integration between Hopecare App, to import the monitored data collected, and Globalcare was done using platform HSHelios
	• Minor fixes from pre pilot phase 1
	
	Moreover, the following issues have been solved:
	• Add field "Estado Civil" to "Avaliação Inicial" Questionnaire
	• Icon does not turn green when intervention is performed

Licence	Globalcare is released under an end-user licence 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	• Pedro Pinto (GLINTT) - ppinto@glintt.com • Support requests and issues can be reported at https://gitlab.com/pharaongroup/Globalcare/-/issues
Testing reports	The integration with vital signs monitoring devices were tested with up to 10 end users in the pre-pilot phase (Social Workers and Homecare Assistants). All measurements were retrieved correctly. Globalcare was running on a Samsung

	Galaxy Tab A7 and was able to retrieve all the measurements done. Although, all devices integrated correctly with Globalcare during the testing phase, the devices chosen to monitor vital signs in the pre-pilot phase were: ● Oximeter - Pulse Oximeter (Nonin) ● Thermometer - Fora IR20b ● Blood pressure measurement device - AND (A&D Medical) Additionally, a real case scenario was tested, where Globalcare and Helios integration with the health biomonitoring kit was done, at an older adult's house. The measurements were correctly obtained and imported to Globalcare by the homecare assistant. The Globalcare setup was also validated by the end-user.

4.5 Hateoas Client

The HATEOAS client is a software component that allows full usage of the HATEOAS (Hypermedia As The Engine Of Application State) principle when using REST services. In a nutshell, the principle is that, together with the response message a REST service provides, it also provides affordances for further requests, to the same or other services. The clients are free to follow the affordances that suit their needs. The developed component includes the client itself, but also a proxy server that extends the responses received by the original services and extends them with links, thus creating affordances. Ultimately, this prototype allows client-side to drive REST service composition, as an alternative to business process-based approaches that can be quite complex to create and manage.

Release notes	A first prototype has been developed.
Licence	Open source
Resources	Information about the HATEOAS client and the specification made for the PHArA-ON project is available at https://gitlab.com/pharaongroup/hateoas-client
Support and contact points	● Michael Mrissa - michael.mrissa@innorenew.eu ● Sidra Aslam - sidra.aslam@innorenew.eu
Testing reports	We tested our prototype with the key to the beta version of the API of the Smarthabits platform. A new version will follow with the new API.

4.6 HSHelios Platform

HS.Helios is a proprietary integration system that promotes interoperability, by regulating and analysing the exchange of messages between different health information systems that are currently deployed at a Health institution.

Release notes	The version 2.5 of HSHelios Platform was released in December 2021. The provided functionalities are listed below: ● HSHelios can both receive and provide data to Pharaon project by: ○ Publishing or connect to REST or SOAP services ○ Establish HL7 MLLP channels to receive and provide data ○ Database pooling Depending how data are exchanged authorization process can be different: ● Publish/Subscribe: HSHelios has implemented a publish and subscribe model based on subscription. HS.Helios can deliver subscribed data to a REST/SOAP/MLLP endpoint. ● Web Service: HSHelios can publish REST or SOAP APIs to share integrated data. In this scenario HSHelios relies on OAuth to authorise API access.
Licence	HSHelios is released under an end-user licence 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
Testing reports	The integration provided by HSHelios between Globalcare and Hopecare, to import the monitored data collected, was tested in order to verify if the specification and mapping carried out for this project was within the project's objectives. The tests were successfully completed, having achieved the goal of integration between Globalcare and Hopecare through HSHelios.

4.7 ICE Data Interoperability

The ICE Data Interoperability tool will be used by ICE to generate mapping from the outputs of each pilot to some common format for central use. At the moment there is no obvious requirement for mappings going in the opposite direction but if the need arises, these could be generated. The mappings would operate as microservices running as either standalone docker containers or as part of a Kubernetes cluster.

At the time of writing, there is no definition of either the output format from each of the pilots nor the common format. Other things which have an impact on the production of the transformations include whether they are to be run on a central server or on the individual pilots; how often they will be expected to be run i.e., running continuously with streaming data or occasional manually triggered batch runs as the need arises; method used to transfer the data from pilot to central server. It is possible that some of the above options will require changes to ICE Data Interoperability design time tool and/or the runtime environment.

Release notes	It is unlikely that these transformations will be produced in the period covered by this delivery so the following describes what will be done beyond that. The items to be released will include a transformation for each pilot, and either the ICE Data Interoperability runtime or possibly a PHArA-ON specific runtime. Each of the transformations would have a common core so it may be appropriate to have a single release note for all the transformations with pilot specific sections if required. The release note for the runtime would be common.
Licence	The transformations produced by ICE Data Interoperability will be released under the Apache 2.0 licence. The ICE Data Interoperability Run-time is ICE proprietary. At time of writing, it is not expected that the ICE Data Interoperability Design-time will be released.
Resources	● Source code for the transformation will be hosted at, or near, https://gitlab.com/pharaongroup/ice-data-interoperability. It is likely that a new subgroup will be created with a project for each pilot. ● Binary files will be available at https://gitlab.com/pharaongroup/ice-data-interoperability/artifacts/… ● Docker Images for this technology will not be hosted anywhere for PHArA-ON since it is not a technology for integrating with the rest of PHArA-ON technologies. It will rather be used to produce transformation only, to work on each pilot's data. ● User guide instructions will be available https://gitlab.com/pharaongroup/….
Support and contact points	● John Beattie, Information Catalyst of Enterprise (ICE) - John.beattie@informationcatalyst.com ● Support requests and issues will be able to be reported at, or near, https://gitlab.com/pharaongroup/ ice-data-interoperability/-/issues
Testing reports	TBD

4.8 ICE Orchestration

ICE Orchestration is a tool for designing and executing business processes. Such processes can involve tasks performed by a human (a user) such as reviewing and completing forms, manual tasks performed in an external system, and service tasks performed automatically by a software service. In the PHArA-ON case, the tool supports both: (i) the service composition activities in designing processes (i.e., service compositions) made of software services only and (ii) the service orchestration activities in (providing support for) executing service compositions so that services are called one after another in an automatic manner. At the time of this writing it is still unclear who will use this tool. A possibility is to use the tool to connect services from different pilots helping to realise the PHArA-ON ecosystem.

Release notes	The current available version of ICE Orchestration is accessible at https://pharaon.icelab.cloud/ and contains the following functionalities: ● *Designer*: this functionality is for designing processes (e.g. service compositions) in a graphical way using the BPMN standard; ● *Forms editor*: this functionality allows the user to design forms to capture data from users when a use task is being executed as part of a running process; ● *Marketplace*: this functionality allows the user to register software services, so that then they can be used by the Designer to compose services; ● *Control panel*: this functionality allows the user to execute processes (i.e., service compositions). Many instances can be created for each designed process. They can be paused or even inspected in real-time; ● *My tasks*: this functionality allows the user to complete forms as indicated by user tasks; ● *Process analytics*: with this functionality the user can visualise statistics of their processes such as total execution time, number of instances running, etc.
Licence	ICE Orchestration is proprietary software, but a version of it is planned for open source under Apache 2.0. There is no exact date when this release will be made in the future.
Resources	There is no open-source repository yet. Docker images for this technology will be available in an ICE site once it is clear when they will be used. User guide instructions will be available at https://gitlab.com/pharaongroup/ice-orchestration/ The live version is accessible at https://pharaon.icelab.cloud/
Support and contact points	● Cesar Marin, Information Catalyst for Enterprise (ICE) - cesar.marin@informationcatalyst.com ● Support requests and issues can be reported at

	https://gitlab.com/pharaongroup/ice-orchestration/-/issues
Testing reports	TBD

4.9 IoTool Platform

The IoTool platform helps connect IoT devices via any interface to a smartphone, microcontroller or directly to the IoTool Cloud.

Release notes	The current available version of IoTool (1.0/220403) is accessible for testing purposes on https://stest.iotool.io/. The same version is installed on the pre-validation pilot server (IT and SI). IoTool standalone client can be downloaded from Google Play. From the start of the project, some changes has been done to the IoTool platform to better suit PHArA-ON pilots and integration part needs: ● connection to ThingsBoard (ENT) -> SmartHabits (ENT) -> Discovery (ASC) with selective data preparation (only the part of received data is sent further, some as raw data, some as average (calculated) data); ● added additional reports, like last sensor readings based on feedback from the pilot partners; ● connection to ThingsBoard (ENT) has been changed from http post to MQTT messages; ● added support to all Shelly home and environmental sensors (22) with 133 different sensor readings; ● added support to three AirQuality monitoring devices (AWAIR and QingPing) with multiple parameters; ● added support to PineTime smartwatch (firmware changes (HR, PPG, Steps, SOS) and IoTool platform support); ● added support to FitBit devices (WIP); ● added support to WearOS devices (WIP); ● added Grafana support; ● changed registration procedure to allow faster deployment for pilots.
Licence	IoTool is a proprietary software. Some libraries with different licences has been used: ● Adminer - Free for commercial and non-commercial use (Apache Licence or GPL 2), ● InfiniTime - GNU General Public Licence version 3. InfiniTime modifications for the PHArA-ON project - GNU General Public License version 3.
Resources	● IoTool web page

	• Source code is hosted in the private GitHub repositories, PHArA-ON Gitlab dedicated repository for documentation, issue reporting, and the site can be found here.
Support and contact points	• Jure Lampe, CEO (SenLab) - jure.lampe@senlab.io • Support requests and issues can be reported at https://gitlab.com/pharaongroup/iotool-platform/-/issues
Testing reports	TBD

4.10 IoChat Platform

IoChat is the Secure WebRTC Communication Platform, Multi-platform Client/API/Self-hosted Server/Private Cloud solution used for user-to-user or multiuser text chat, file sharing, audio and video conferencing systems.

Release notes	The current available version of IoChat (1.0/220105)) is accessible for testing purposes on https://eu.iochat.io/. Most of the IoChat platform changes are related to Daisy new functionalities, like NFC support and similar (described in Daisy section).
Licence	IoChat is a proprietary software. Some libraries with different licence have been used: • Framework7 - MIT
Resources	• IoChat web page • Source code is hosted in the private GitHub repositories, PHArA-ON Gitlab dedicated repository for documentation, issue reporting, and the site can be found here.
Support and contact points	• Jure Lampe, CEO (SenLab) - jure.lampe@senlab.io • Support requests and issues can be reported at https://gitlab.com/pharaongroup/iochat-platform/-/issues
Testing reports	TBD

4.11 SeniorsPhone

SeniorsPhone features provide a simple and easy interface for the Android smartphone with big buttons for easy calling, texting, SOS, location functions and other functions (weather, magnifier, music, torch...).

Release notes	The current available version of SeniorsPhone is accessible for testing purposes on Google Drive. After the first installation Over-The-Air updates are available on https://ota.senlab.io. From the start of the project, some changes has been done to SeniorsPhone to better suit PHArA-ON pilots and integration part needs: ● application changes based on Google new privacy and security requirements; ● IoTool sensors, cloud connection, settings library integration; ● Over-The-Air scripts preparation for easy SeniorsPhone Application updates (SenLab OTA server); ● changes on UX/UI based on the prevalidation feedback.
Licence	SeniorsPhone is a proprietary software.
Resources	● SeniorsPhone web page ● Source code is hosted in the private GitHub repositories, PHArA-ON Gitlab dedicated repository for documentation, issue reporting, and the site can be found here.
Support and contact points	● Jure Lampe, CEO (SenLab) - jure.lampe@senlab.io ● Support requests and issues can be reported at https://gitlab.com/pharaongroup/seniorsphone/-/issues
Testing reports	TBD

4.12 Daisy

Daisy is the simplest possible video conferencing via TV that seniors would be willing and able to use.

Release notes	Daisy is an Android TV IoChat client. The current available Daisy version (1.0/22035) is accessible for testing purposes on Google Drive. From the start of the project, some changes has been done to Daisy to better suit Fast-pilots (installed in DU Izola to help seniors during Covid-19) and PHArA-ON pilots needs: ● changes on the FrontEnd to be suited in sharing multi-user environments - added NFC cards support for easy login/logout with a personal card; ● added a NFC reader support; ● added centralised multi-user administration (users' operations, NFC

	cards, logging) with different roles; ● changes on UX/UI based on the prevalidation feedback.
Licence	Daisy is a proprietary software. Some libraries with different licence have been used: ● Framework7 - MIT
Resources	● Daisy web page ● Source code is hosted in the private GitHub repositories, PHArA-ON Gitlab dedicated repository for documentation, issue reporting, and the site can be found here.
Support and contact points	● Jure Lampe, CEO (SenLab) - jure.lampe@senlab.io ● Support requests and issues can be reported at https://gitlab.com/pharaongroup/iochat-platform/-/issues
Testing reports	TBD

4.13 MISS Activity

The MISS Activity is a development of the MOX Activity Monitor from Maastricht Instruments. It is an easy-to-use activity monitor for the elderly. Within PHArA-ON, the embedded step algorithm, the user interface and the interoperability are adapted, while the hardware (MOX sensor) is not changed.

Release notes	At the time of writing several versions of the MISS Activity component have been released. **Version 0** was released in 2019 and is the original MISS Activity (pre- PHArA-ON). This version was used in *pre-validation Phase I* and provides a standalone application with sensor and smartphone (see link: https://www.accelerometry.eu/products/ehealth-solutions/miss-activity/) **Version 1** was released for internal testing (MAIN and Adsysco) in September 2021. This release brings the new functionalities of data flow to database and the possibility of data sharing with third party Adsysco. This version was used in *pre-validation Phase II*. The full list of changes can be found below: ● Dashboard components: ○ Redesign app frontend to allow authentication according to OAuth ○ Inclusion of privacy policies ● For Backend components: ○ Designed authentication flow using OAuth between Miss

	Activity and Adsysco ○ Include measurement ID in app ○ Extend local data storage to database storage ○ Create backend database ○ Create webservice ○ Develop data calculation mode to translate physical activity data into physical activity categories for matchmaking (Adsysco) ● Other: ○ Investigated Data Processing Agreement (DPA) and Data Protection Impact Assessment (DPIA) **Version 2** was released for internal testing (MAIN) in November 2021 and is a reaction to the feedback from pre-validation Phase I and some reported bugs: ● User interface: improved visualisation by adapting font size and colour contrast ● Resolved issues: connection problem between sensor and mobile, MOX device name representation in the setting screen, adaptation to mobile phone screen size **Version 3** is work in progress (foreseen for Q1 2022)**.** In this version, a general API description is defined, for connecting with the Andalucian pilot. Furthermore, a new functionality is added in the translations. ● API description ● All fields translatable to english and spanish
Licence	● "Acr.UserDialogs" Version="7.1.0.483" – MIT ● "OxyPlot.Xamarin.Forms" Version="1.0.0" – MIT ● "Plugin.BluetoothLE" Version="6.3.0.19" – MIT ● "Raygun4Xamarin.Forms" Version="1.0.8" – Terms of Service (raygun.com) ● "RestSharp" Version="106.11.7" – Apache-2.0 ● "RestSharp.Serializers.NewtonsoftJson" Version="106.11.7" /> – Apache-2.0 ● "SkiaSharp" Version="2.80.2" /> – MIT ● "SkiaSharp.Views.Forms" Version="2.80.2" /> – MIT ● "Splat" Version="11.0.1" /> – MIT ● "sqlite-net-pcl" Version="1.7.335" /> – NuGet Gallery | License of sqlite-net-pcl 1.7.335 ● "Xamarin.Forms" Version="5.0.0.2012" /> – MIT ● "Xamarin.Essentials" Version="1.6.1" /> – MIT
Resources	● General presentation: https://www.accelerometry.eu/products/ehealth-solutions/miss-

	activity/ ● Technical documents, images, manual: https://gitlab.com/pharaongroup/maastricht-instruments-miss-activity ● Version 0 of the MISS Activity can be downloaded from Google Play Store download link: https://play.google.com/store/apps/details?id=com.MI.MissActivity&hl=en_US&gl=US Apple App Store download link: https://apps.apple.com/us/app/miss-activity/id1438242305
Support and contact points	● An Stevens (MAIN) - an.stevens@maastrichtuniversity.nl ● Freek Boesten (MAIN) - freek.boesten@maastrichtinstruments.com ● Support requests and issues can be reported at https://gitlab.com/pharaongroup/maastricht-instruments-miss-activity/-/issues
Testing reports	Version 0 was tested thoroughly and has taken part in a field pilot study as a standalone monitoring device. Version 1 and 2 have undergone functional tests using a fixed protocol with different mobile devices and free user tests. The results have been documented and actions have been described in Jira. As soon as version 3 will be released, this will also be tested in the 2 new languages.

4.14 OneSait Healthcare Data

Onesait Healthcare Data is the core platform of the Murcian Pilot that will allow providing clinicians and older adults and their caregivers with new tools for the management of chronic diseases and provide healthcare organisations with a proactive and integrated care approach for the management of chronic patients. Onesait Healthcare Data facilitates communication and tools for the self-management of chronic patients.

Release notes	Main modules of OHC adapted for the Murcian pilot are the Homecare and the MyHealth App applications that allow the remote monitoring of Chronic Heart Failure (CHF) patients by healthcare professionals, and self-monitoring by older adults suffering from CHF and their caregivers. Both applications comes from previous works and have been adapted for PHArA-ON needs, main new features include their adaptation to Heart Chronic diseases such as inclusion of related vital sign measurements (Oxygen saturation as heart rate frequency and Blood pressure come from previous versions) , the inclusion of the Personalised Care Plan and the Initial Clinical assessment. Homecare has been adapted to allow receiving alerts from other devices (integration with other Pharaon technologies: uGRID, Smartband and AMICARE). The solution has developed the role of the informal caregiver and associated features. In addition other structural components for the correct performance of the solution need to be released for the correct performance of the integrated solution (Background) such as the modules that deal with users management, resources and structure management of the organisation, access and authentication , catalogues management, forms management and Professional desktop. At the moment, the Onesait Healthcare solution from Minsait is being deployed in the pre-production environment of the Murcian Health Service (SMS). Set –up and configuration tasks are ongoing, some configuration issues have arisen and are being fixed by Minsait and SMS teams. In parallel Minsait is progressing in the parametrization of the different modules : access rights and permissions, catalogues, physical and functional structure of the organisation, users, forms etc. Details of Onesait Healthcare version release being deployed in the pre – production environment at SMS infrastructure are as follows: - Homecare : 3.0.0 - MyHeath App : 3.0.0 Background (structural modules required for the performing of Hoemcare and MyHealth) - Professional desktop (clinicians dashboards) : v.3.6.1 - MPI- Users management: 3.7.4 - Alerts: v3.0.1 - Authentication: 3.8.3 - Catalogues management: 3.7.2 - Configuration: 3.8.3 - Forms Management: 3.5.3

Licence	Onesite Healthcare Data is a proprietary software and is released under an end-user licence agreement (EULA) . The terms and conditions of the EULA are being signed among Minsait and the Murcian Health Service (SMS) for the use of Onesait Healthcare Data within the PHArA-ON timeframe.
Resources	Source code is hosted in the private GitHub repositories of MInsait , PHArA-ON Gitlab dedicated repository for documentation, issue reporting, and the site can be found here: https://gitlab.com/pharaongroup/Homecare; https://gitlab.com/pharaongroup/onesait-healthcare-data-myhealth
Support and contact points	• Mónica Álvarez – malvarez@minsait.com • Carlos Gutiérrez – cgutierrez@minsait.com
Testing reports	Homecare 3.0.0 unit and functional tests were performed before integration tests with other Murcian technologies (we use Jira for testing reporting and follow an Agile methodology). Integrated version of Homecare (3.0.0) (application for clinicians) including the interoperability with the technologies UGRID, AMICARE and Smartband was thoroughly tested before the Pre-validation with end users conducted in December. The testing has been performed in testing environments of Minsait as the Infrastructure of the Murcian Health Service (SMS) was not set up in that period , so Minsait facilitated their testing environments to minimise this risk. For Homecare no incidences were reported during pre-validation with end –users although feedback from end-users was collected and some improvements have been done (such as improvements on user interface, usability) . Other improvements are new functionalities suggested by users out of the scope of PHArA-ON, but they have been included in the backlog. The integrated solution of Homecare is being set up and configured in the Pre-production environments of the SMS , and in the next week tests will be performed (as soon as configuration issues are fixed as commented above) . MyHealth version (3.0.0) , the mobile application for patients and caregivers has been tested during the second week of February. Prior to the user validation, thorough technical and functional tests have been

	performed and managed in our Jira. Again for the pre-validation, test environments have been facilitated by Minsait for the Murcian Pilot as the pre-production environments are being set-up at the moment. In addition the SMS environment does not allow access to the internet so that the APK version of MyHealth cannot be tested by Patients from their mobiles in Pre-production (it is important to note that we are dealing with healthcare organisations managing sensible data of level 3 of security so security policies are consistent) . During Pre-validation tests with end –users no incidences have been reported and minor improvements comments from end users (patients and informal caregivers) were received, which have been included in the backlog. In the next week as soon as the Pre-production environment problems are fixed, tests in this environment will be performed, especially interoperability tests with technologies and Homecare. For MyHealth App interoperability tests need to be performed in production environments, as the PRE-production environment of SMS does not allow access from external networks (internet).

4.15 OneSait Platform

Onesait Platform provides the flexibility so that developers can build their own solutions in a solid and agile way using Open Source technologies, a flexible architecture and an innovative approach. Based on Open Source components, Onesait Platform covers the entire life cycle of information (from ingest to visualisation through its process and analysis). It offers a unified web console for the development and operation profiles of the solutions. Onesait Platform is used in production in many different domains: IoT, energy, smart cities, industry, etc.

Release notes	The version used in the Andalusian pilot is 3.2.0-legend. The full release notes can be read in https://onesaitplatform.refined.site/space/ROAD/2217869548/List+of+releases, and the whole release history in https://onesaitplatform.refined.site/space/ROAD/7897561/Releases+History
Licence	Onesait Platform has an Apache Licence v2

Resources	You can find source code at: https://github.com/onesaitplatform/onesaitplatform-cloud Onesait Platform Community edition is a free, open-source Digital Platform that anyone can download and use to build a complete solution over it. This repo contains the Cloud Side of the Platform. The documentation is located at https://dev.onesaitplatform.com.
Support and contact points	In the context of the Pharaon Project the main contacts are: ● Jose María Barriga García - jmbarriga@minsait.com) ● Carlos Fernández Sánchez (cfsanchez@minsait.com) The official link for Onesait Platform Support is https://servicedesk.onesait.com/servicedesk/customer/portal/85/
Testing reports	Onesait Platform is developed following a methodology that enforces the quality of the source code created. Unit tests and Integration tests are included in the devops workflow. In addition, each new version of Onesait Platform has to pass a QA process before its release. The main issue tracking system used by Onesait Platform is not public and each user can only see its own issues. This facilitates the sharing of confidential information between users and the support team. Anyway, within each release the list of issues solved is published (https://onesaitplatform.refined.site/space/ROAD/7897561/Releases+History).

4.16 PACO

The Persuasive E-health Agents for Coaching Older adults (PACO) is a coaching tool to help adults toward a dietary and healthier behaviour change.

Release notes	The new features provided by the PACO component in this PHArA-ON initial release are: ● Designed authentication flow using OAuth between PACO and Adsysco. ● Tested the first implementation of the Adsysco API. In particular, these are the functionalities implemented in the version released in October 2021: ● Implemented integration with OAuth from Regicare (Adsysco):

	Regicare can open PACO with a parameter to indicate that a Regicare user wants to use PACO. PACO then always initiates the OAuth process. This allows PACO to authenticate and identify the user and gain authorization to access the Regicare API for this user. ● Implemented data integration with Regicare: when the user defines goals in PACO, they will be shared with Regicare ● Updated the design of the buttons at the top right (menu and logout) because they were not clear to users. Moreover, these are the functionalities implemented in the version released in December 2021 ● Removed the week counter and total number of weeks that were used in a previous PACO study. ● Removed reference to the number of weeks from the dialogues. ● Removed request for a research code from the dialogue.
Licence	Proprietary
Resources	● Source code of this component is hosted at https://gitlab.com/pharaongroup/paco ● The service is accessible at https://portals.rrdweb.nl/paco/
Support and contact points	● Dennis Hofs (RRD) - d.hofs@rrd.nl ● Support requests and issues can be reported at https://gitlab.com/pharaongroup/paco/-/issues
Testing reports	PACO has been evaluated in a previous evaluation study. The integration has been tested and successfully validated during the first and second pre-validation phases.

4.17 rb1-base

Robotnik's RB-1 is an autonomous and configurable robot, focused on the field of research in indoor applications. It can carry different loads or materials and can integrate other components or platforms such as telepresence devices.

Within the context of PHArA-ON, it participates in the Murcia Pilot. This robot will be placed on Hospital Morales Meseguer (part of SMS) in the infectious zone, in order to help healthcare professionals in their daily tasks.

Release notes	The robot has been delivered and is used by the hospital personnel. The features included in this release are: ● Remote and Local movement ● Remote and local camera tilt position ● Video Conference capabilities with zero effort to the patient ● Mapping generation capabilities

	• Safe Autonomous navigation
Licence	Proprietary
Resources	Public available source code for this robot can be found: https://github.com/RobotnikAutomation/rb1_base_common https://github.com/RobotnikAutomation/rb1_base_sim https://github.com/RobotnikAutomation/rb1_base_common-release https://github.com/RobotnikAutomation/rb1_base_sim-release
Support and contact points	• Guillem Gari - ggari@robotnik.es • Support requests and issues can be reported at https://gitlab.com/pharaongroup/rb1-base/-/issues
Testing reports	Robotnik's RB-1 have been already delivered to the hospital and personnel have been properly trained. It has been manually tested during the pre validation phase.

4.18 Regicare Platform

RegiCare is a user-friendly information system built around the primary processes of Welfare organisations. It's the user platform of the Dutch pilot that stores and exposes user profiles, authentication and authorisation schemes and provides the user with summarised information from PACO (Persuasive eHealth Agents for Coaching Older adults) and MISS Activity.

Within the context of PHArA-ON the older adult can enrol in activities (for example the BoodschappenPlusBus), update his/her profile, link to PACO, start the Wizard and participate in a user community related to the users' activities.

Release notes	In the context of the Initial PHArA-ON release the Regicare Platform provides the following functionalities: • OAuth implementation for Dutch pilot • API for access to RegiCare platform • Create a user community for elderly adults and volunteers related to the activities they participated in
Licence	RegiCare is released under an end-user licence agreement (EULA).
Resources	RegiCare is hosted in a private datacenter. Application programming interfaces are available for third parties to connect to the platform. More information is available at https://gitlab.com/pharaongroup/regicare-platform.
Support and contact points	• Danny Jansen - djansen@adsysco.nl • Support requests and issues can be reported at

	https://gitlab.com/pharaongroup/regicare-platform/-/issues
Testing reports	RegiCare has been manually tested and successfully validated during the first and second pre-validation phases.

4.19 RRD eHealth platform

The RRD eHealth Platform serves as the backend for the PACO application. It handles authentication, data storage and OAuth connections with Regicare.

Release notes	A new PHArA-ON feature released in the October 2021 version is the OAuth connection with Regicare.
Licence	Proprietary
Resources	• Documentation of this component is hosted at https://gitlab.com/pharaongroup/rrd-ehealth-platform • The service is accessible at https://servlets.rrdweb.nl/r2d2/
Support and contact points	• Dennis Hofs (RRD) - d.hofs@rrd.nl • Support requests and issues can be reported at https://gitlab.com/pharaongroup/rrd-ehealth-platform/-/issues
Testing reports	See PACO.

4.20 Sentab Tablet

Sentab is a digital platform for older adults to communicate and socially interact between themselves, caregivers and family members.

Release notes	The main development activities for the Sentab Tablet component are reported below: **V1** - the following features extended to tablet interface: • upgrading the entire application to Android 11 and javax • changes in UI compared to TV interface • migration to touch screen interface • adding language supports (PT, IT, SP) • submitting media uploads to the Feed, Events and Communities • submitting text posts to the Feed, Events and Communities • adding comments • video call lib update

	• editing/deleting posts on tablet interface • iHealth watch integration • bug fixes (Android permissions, clock etc) **V1 - TV app fixes**: • upgrading the entire application to Android 10 and javax • updating ROM and OTA updater • adding language supports (PT, IT, SP) • Youtube links work • iHealth watch lib update • video call lib update • label names **V1 - family app**: • adding language supports (PT, IT, SP)
Licence	Proprietary
Resources	There are 2 repositories, the first one for tablet interface, another one for the cognitive games that should be included in the tablet interface, but are separate apps: • https://gitlab.com/pharaongroup/sentabtablet • https://gitlab.com/pharaongroup/sentabgames
Support and contact points	• Tarmo Pihl - tarmo.pihl@sentab.com
Testing reports	Sentab has been manually tested and validated during the first and second pre-validation phases. Some bugs are being addressed in the newer release of the software. These bugs are not related to the core software, but critical third party libraries, which are not backwards compatible.

4.21 SmartBand

The SmartBand component consists of 2 software modules: IoT application and PHArA-ON SmartBand server. During the first pre-validation phase, a version of the application composed of 2 Gitlab projects called *smartband-app* and *smartband-backend* has been used. In the second phase of pre-validation, the latest version of the component can be found in 3 GitLab projects called *Pharaon Kit 12 V2 Smartband Integration*, *Cordova Plugin Smartband* and *smartband-backend*. The following sections provide all the information requested for each of these developments.

Release notes	At the time of writing several versions of the SmartBand component have been released. The main development activities for each version are reported below:

	smartband-app ● **Version 1.2 Release** ○ MyHealth style guide applied ○ Improved connection messages ○ Improved sync info connection status ○ API traffic vía HTTPS ○ Better performance for Invalid AUTH KEY or SECRET pairing error ○ Added bad secret error alert ○ Removed unused permissions ○ Removed unused drawables ○ Removed several unused resources and strings ○ Operations for avoid conflicts for becoming a cordova plugin ■ Removed data binding library ■ Changed the Application class to a singleton pattern. ■ All strings renamed with a prefix ■ Modifications in base sync provider ● **Version 1.1 Release** ○ Added User statistics screens ● **Version 1.0 Release** ○ Some synchronisation bugs solved ○ Some unused code removed ● **Version 1.0** ○ Unbound process notifies the server ○ Changed trigger alarm params ○ Device tries to reconnect to the band if connection doesn't send data in 2 minutes ○ Force bluetooth enable if not in the main activity ○ Difference between realtime data and fetch data in database. ○ Sync philosophy sends data each seconds to the server ***smartband-backend*** ● **Version 1.0 Release** ○ Bug fix: "wake" command excludes any band which mobile device is not bound to. ○ Added 2nd external code field for patient profile. ○ External view authorization process. ○ Bug fix: authorization process changed for controllers. ○ API: added patient pre-filtering for available bands. ○ Bug fix: UPCT platform user detail's view corrected: now is showing it's properties, and the edit button brings you correctly to 'edit' view. ○ Bug fix: Responsive content for patient details view: size it's modified according to screen size. ○ Bug fix: Alarm controls now can only be driven by UPCT

	administrators and health professionals. ○ The behaviour for wake_up fcm message has been changed to affect any band which hasn't reported any activity in the last hour. ○ It's been added a new Rest API op., api/device/unbound, used to unbind a band from a mobile. ○ Mobiles: model attribute it's been provided through the pairing process. ○ Bands: when defining a new band through the web interface, it's been added the function of providing patient external code. ○ In the patient details view, it's been added the function of downloading a PDF report. ○ The data of the graphs is loaded asynchronously via Ajax / JQuery. ○ Added the possibility of setting the interval in minutes to define the granularity of the data. ○ In the patient detail view, a graph is shown with the steps of the last day, grouped by hours. ○ In the patient detail view, a graph is shown with the heart rate for the last day, grouped by hour, obtaining the maximum value for each interval. ○ Added a new activity type: "today steps",that gathers the total steps of the day. ○ Added graphical assets implemented by the library ApexCharts.
Licence	SmartBand is released under the LGPL (GNU Lesser General Public Licence)
Resources	Source code of SmartBand component is hosted at 4 GitLab projects: ● Pharaon Kit 12 V2 Smartband Integration: https://gitlab.com/pharaongroup/pharaon-kit-12-v2-smartband-integration ● Cordova Plugin Smartband: https://gitlab.com/pharaongroup/cordova-plugin-smartband ● smartband-app: https://gitlab.com/pharaongroup/smartband_app ● smartband-backend: https://gitlab.com/pharaongroup/smartband_backend The above repositories include the user and installation manual. The service is accessible at: http://pharaon.upct.es

Support and contact points	● Ramón Martínez Carreras (UPCT) - ramon.martinez@upct.es ● María Victoria Bueno Delgado (UPCT) - mvictoria.bueno@upct.es ● Support requests and issues can be reported at: ○ Pharaon Kit 12 V2 Smartband Integration: https://gitlab.com/pharaongroup/pharaon-kit-12-v2-smartband-integration/-/issues ○ Cordova Plugin Smartband: https://gitlab.com/pharaongroup/cordova-plugin-smartband/-/issues ○ smartband-app: https://gitlab.com/pharaongroup/smartband_app/-/issues ○ smartband-backend: https://gitlab.com/pharaongroup/smartband_backend/-/issues
Testing reports	The component has already been evaluated in the functional test of the first pre-validation phase using *smartband-app* and *smartband-backend* projects. It has now been successfully validated at a high level in the second phase of the pre-validation using *Pharaon Kit 12 V2 Smartband Integration* and *Cordova Plugin Smartband* projects instead of *smartband-app*. However, the final integration tests with the platform are still pending.

4.22 SmartHabits Platform (SHP)

The SHP is based on microservices architecture providing great flexibility for responding to the needs of different scenarios. The platform utilises Artificial Intelligence (AI) concepts of reasoning, pattern detection, decision making, and relies upon advances in Ambient Intelligence (AmI), sensor networks, and Human-Computer Interaction (HCI). It uses non-vision-based sensor data and data-driven pattern recognition (based on probabilistic reasoning and machine learning). By using affordable, unobtrusive environmental sensors in the home, the system learns about a person's typical behavioural patterns (recurrent way of acting) without invading their privacy and alerts her family member(s) or (informal) caregiver(s) if an anomalous situation is detected. The system does not focus on health monitoring nor life-threatening emergencies but rather on pattern recognition and anomaly detection providing better care assistance, reassurance, and peace of mind.

The SHP exposes its functionalities primarily via REST (REpresentational State Transfer) APIs using lightweight JSON (JavaScript Object Notation) as a data-interchange format. All APIs are secured by JWT (JSON Web Tokens).

Release notes	At the time of writing several versions of The SHP have been released. The functionalities provided for each version are reported below: **Version 1.0**

	First version of SmartHabits Platform (SmartHabits-v1.0) was released in 2018 and it is considered as one of the input platform components to the PHArA-ON project. First version of SmartHabits Platform used its own implementation of the IoT platform layer based on the open source eWall platform and supported pattern recognition and anomaly detection for data from environmental sensors. The first version was successfully validated in the real user environment within a pilot that was set up in the city of Zagreb in cooperation with the Foundation taking care of the older people living alone. The pilot had run for six months after which the feedback from all involved users was collected and analysed. **Version 2.0** The second version has been done in scope of the PHArA-ON project and includes many major upgrades. One of the main improvements was changing the IoT layer and achieving interoperability and integration with 3rd party open source ThingsBoard IoT platform. This open source IoT platform is considered as a main syntactic integration point for all platforms and technologies that are planned to be used for Italian and Slovenian pilots. The first development of SmartHabits Platform version 2 was completed in September 2021, validated in staging/integration testing environment during September and October 2021, and released as stable 2.0.0-RC1 version in November 2021. ● Functionalities: ○ Interoperability and integration of devices management with ThingsBoard IoT platform ○ Interoperability and integration of telemetry (sensor) data with ThingsBoard IoT platform ○ SmartHabits security layer update to support new roles ○ Underlying framework upgrade and migration to latest releases ○ Data model and processing updates to accommodate for PHArA-ONdevices and new sensor types ○ Implementation of PHArA-ON API Gateway adaptations for new use cases ○ Interoperability with Discovery notification management system ○ New token support ○ Various updates in relation to PHArA-ON pilot setup (including modification in database to be compliant with changes in DB usage and licence rules) ○ Adoption of OpenAPI specification and exposing interactive always-up-to-date REST API documentation based on swagger2.0 (as described in PHArA-ON deliverable *D4.4 Service orchestration support tools – first* in chapters 3.3. and

	3.4 where SmartHabits platform was used as a lighthouse implementation for the rest of PHArA-ON) ○ Adoption of newest OpenAPI 3.x specification and exposing REST API interactive documentation for PHArA-ON consumers based on swagger 3 ○ Upgrade of underlying framework and 3rd party libraries based on the cybersecurity and vulnerability analysis done within the cybersecurity task T3.4. ○ TLS 1.3 and HTTPS setup to support GDPR compliance from the technical perspective (in parallel with data processing agreements preparation for all three pilot sites where SmartHabits is being used namely IT: Apulia, IT: Tuscany, SLO: Izola) ○ New external notification system support
Licence	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 the 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	● Miran Mošmondor (ENT) - miran.mosmondor@ericsson.com ● Andrej Grgurić (ENT) - andrej.grguric@ericsson.com ● Support requests and issues can be reported at https://gitlab.com/pharaongroup/smarthabits-platform/-/issues
Testing reports	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. A detailed description of the testing activities performed to validate the SmartHabits and ThingsBoard deployments can be found in Appendix A.

4.23 uGRID

uGRID is an energy management software developed by MIWenergia with simple solutions that helps you control and monitor your energy consumption and achieve the maximum possible sustainability. In order to know the data in real time, a series of equipment and sensors are deployed in the internal network of the facility. The measurement collects accurate information on consumption in real time and is displayed through the online platform.

Release notes	In order to adapt the uGRID tool into PHArA-ON requirements, many developments were made in this technology as described below: ***uGRID back-end*** ● Until now, uGRID only recovered data from electrical meters and analyzers, and we had to make many modifications in the backend

	software to be able to recover the data in real-time of different IoT devices; ● We developed a new alarm generation system to detect certain behaviours and events; ● We developed a new API with all the data, functions and information related to Pharaon requirements. ***uGRID front-end*** We adapted our UI in order to make it simpler, and easy to use for the user; We also had to integrate uGRID into other technologies, so we had to adapt the interface to the partners visual and style guides requirements; ***New mobile app*** We developed a new mobile application, following style guidelines and recommendations from the partners in order to integrate it into their technology.
Licence	● uGRID : Proprietary license ● uGRID Mobile : Open source ● uGRID Pharaon API : Open source
Resources	● uGRID is accessible at https://ugrid-pharaon.miwenergia.com/ ● uGRID Pharaon API is accessible at https://api-pharaon.miwenergia.com/swagger/index.html ● Source code of uGRID Pharaon API is accessible at https://gitlab.com/pharaongroup/ugrid-platform ● Source code of uGRID Mobile app is accessible at https://gitlab.com/pharaongroup/ugrid-platform
Support and contact points	● Antonio Sánchez Ibernón (MIW) - antonio.sanchez@miwenergia.com ● Support requests and issues can be reported at https://gitlab.com/pharaongroup/ugrid-platform/-/issues
Testing reports	uGRID tools have already been evaluated in the functional test of the first phase of the pre-validation. However, the final integration tests with the platform are still pending.

5 Conclusions

This deliverable describes the Initial Release of the PHArA-ON ecosystem at M26 of the project, which represents the reference release for the pilot deployment at M28.

The components involved in the PHArA-ON ecosystem have been presented highlighting how they fit in the reference architecture described in D2.1 and how they have been integrated in order to achieve the user and pilot's needs. In particular, the document details the set of the PHArA-ON components involved in each pilot and how they interact with each other. The inter-platform interoperability, aiming to enable the communication between the different modules (e.g, technologies, services) has been achieved thanks to the adoption of several open standards and protocols (such as JSON, HTTP, MQTT, etc.) along with security and privacy mechanisms to protect the exchanged data.

One aspect not covered yet in this deliverable is the inter-platform interoperability, namely the ability of different complete systems (i.e., the pilots) to work together and exchange information even if the work to achieve it has been started. In PHArA-ON a federated approach will be adopted to interconnect and establish interoperability between different platforms and at the time of writing investigation and discussion on the possible inter-platform interoperability solutions have been put in place during the project's synchronisation meetings. This work will be reported in the next WP5 deliverables.

According to the Software Development LifeCycle (SDLC), this document also reports the CI/CD process established in WP5 to guide and control the Build, Test and Release phases. In particular a common release process has been identified following the main principles and guidelines of the DevOps approach, along with a set of project-wide tools that will be used to support and automate different activities/phases of the PHArA-ON integration and release cycle.

Finally, this deliverable has introduced the testing approach adopted within the project to achieve the QA, highlighting how the testing activities are based on a multi-level testing that takes into consideration both functional and not-functional dimensions.

6 References

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

[2] PHArA-ON Consortium, Deliverable 2.2 - Pharaon initial ecosystem architecture, 2021.

[3] PHArA-ON Consortium, Deliverable 3.1 - Interoperability Platforms Descriptions, 2021.

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

[5] PHArA-ON Consortium, Deliverable 5.2 - Pharaon ecosystem - Intermediate release, 2022 (expected).

[6] PHArA-ON Consortium, Deliverable 10.4 - Security, integrity, trust requirement, 2020.

[7] PHArA-ON Consortium, Deliverable 7.3 - Guidelines for action research, 2024 (expected).

[8] PHArA-ON Consortium, Deliverable 7.4 - Impact assessment methodology, 2022 (expected).

7 Appendix A: CI/CD tools

Table A.1 shows a list of the most important CI/CD tools.

Table A.1: CI/CD tools

Jenkins	
Jenkins is an open-source automation server where the central build and continuous integration process takes place. The program is a Java-based self-containing program with packages for Windows, macOS, etc. Jenkins supports the construction, deployment and automation of software development projects by providing hundreds of plugins.	
Features	• Easy installation and upgrade to various operating systems • Simple and easy to use interface • Extensible with a huge community-based plugin resource • Easy configuration of the environment in the user interface • Supports distributed master-slave architecture builds • Build schedules based on phrases • Supports execution of Windows shells and commands in pre-build steps • Supports notification of build status
Licence	Free. Jenkins is an open-source tool with an active community.
Homepage	https://jenkins.io/
CircleCI	
CircleCI is a CI / CD tool which supports the rapid development and release of software. CircleCI enables automation across the user's pipeline, from code building, testing to deployment. You can integrate CircleCI with GitHub, GitHub Enterprise, and Bitbucket to create builds when you commit new code lines. CircleCI also hosts continuous cloud-managed integration or runs a private infrastructure firewall.	
Features	• Fits into Bitbucket, GitHub, and Cloud Enterprise • Uses a container or a virtual machine to build • Straightforward debugging • Automated parallelising • Fast Testing • Custom text, and IM updates • Deployment is continuous and branch-specific • Quite customisable • Automated fusion and custom packet upload commands • Rapid setup and unlimited construction
Licence	Linux plans begin with the option to run one job at no charge without any parallelism. Open-source projects are getting three more free containers. You can see the pricing for determining which plan(s) you need during sign-up.
Homepage	https://circleci.com/

Bamboo	
Bamboo is a continuous integration server which automates software application release management and thus creates a continuous delivery pipeline. Bamboo includes the development and functional testing, assigning models, marking updates, deploying and enabling new output models.	
Features	● Provides support for up to 100 remote agents ● Run test batches in parallel and get quick feedback ● Creates images and pushes to a record ● Per-environment permissions which allow developers and testers to deploy on demand in their environments while the output remains locked ● Detects new branches in Git, Mercurial, SVN Repos and automatically applies the main line CI scheme to them ● Triggers build based on modifications found in the repository. Pushes Bitbucket notifications, a set schedule, completing another build or any combination thereof.
Licence	Bamboo price ranges are based on agents rather than users, or “build slaves.” The more agents it can run at the same time, the more processes it can run-either in the same build or different builds.
Homepage	https://www.atlassian.com/software/bamboo

TeamCity	
TeamCity is the build management and continuous integration server for JetBrains. TeamCity is a continuous integration tool which helps to develop and deploy various types of projects. TeamCity runs in a Java environment and integrates Visual Studio and IDEs. The tool can be installed on both Windows and Linux servers, and supports projects like. NET and open-stack. TeamCity 2020.1 provides conditional construction steps, allows build agents to be launched into a Kubernetes cluster, and integrates with Azure DevOps and Jira Software Cloud. It introduces more functionality in a multi-node system to secondary servers, comes with a new Slack notifier, and has several major enhancements to the experimental UI.	
Features	● Provides several ways for the subproject to reuse parent project settings and configurations ● The parallel runs works on various environments simultaneously ● Allows to build history, view test history reports, pin, tag and add favourites ● Easy to customise, interact and server extension ● Keeps the CI Server stable and functional ● Flexible user management, assignment of user roles, grouping of users, different user authentication methods and a log with all user actions to ensure transparency of all server activities
Licence	TeamCity is a commercial tool with both free and proprietary licences.
Homepage	https://www.jetbrains.com/teamcity/

GitLab	
GitLab is a suite of tools designed to handle different aspects of the life cycle of software creation. The core product is a Git repository manager focused on the web, with features such as issue monitoring, reporting, and a wiki. GitLab allows each commit or push to trigger builds, run tests and deploy code. You can build jobs on a virtual machine, Docker container, or on a specific server.	
Features	● Use branching tools to view, create and manage codes and project data ● Plan, build and manage codes and project data from a single distributed version control framework so that business values can be easily iterated and delivered ● Provides a single source of truth and scalability for project and code collaborations ● Helps delivery teams to fully embrace CI by automating source code development, integration, and verification ● Provides container scanning, static application security testing (SAST), dynamic application security testing (DAST) and dependency scanning to deliver safe applications along with compliance with licences ● Helps the automation and shortening of releases and application
Licence	GitLab is a free, commercial, package tool. It offers on-site and/or public cloud hosting of SaaS on GitLab or in your instance.
Homepage	https://about.gitlab.com/
Travis CI	
Travis CI is a CI service which is used to construct and test projects. Travis CI automatically detects new commitments that were made and pushed to a repository in GitHub. And after each commit of new code, Travis CI will build the project and execute tests accordingly. The tool supports multiple construct configurations and languages such as Node, PHP , Python, Java , Perl, etc.	
Features	● Quick Installation ● Live views build for monitoring GitHub projects ● Pull Assistance order ● Multiple Cloud services deployed ● Pre-installed apps on servers ● Auto deployments on builds which pass ● Clean VMs for all builds ● Supports Linux, macOS, and iOS ● Supports a variety of languages, including Android, C, C#, C++, Java, JavaScript (with Node.js), Perl, PHP , Python, R, Ruby, etc.
Licence	Travis CI is a CI / CD service which is hosted. Private ventures can be checked on a fee basis at travis-ci.com. Open-source projects can be applied on travis-ci.org at no charge.
Homepage	https://travis-ci.com

Buddy	
Buddy is a CI/CD software designed by GitHub, Bitbucket, and GitLab to build, test, deploy websites and applications with code. It employs Docker containers with pre-installed languages and frameworks to build on and monitor and notify actions along with DevOps.	
Features	● Fast to customise Docker based images as an environment for testing ● Detection of intelligent improvement, state-of-the-art caching, parallelism and all-round optimisation ● Develop and test environments, build, customise and reuse ● The scopes are simple and encrypted, fixed and settable: workspace, mission, pipeline, acts ● Services available with Elastic, MariaDB, Memcached, Mongo, PostgreSQL, RabbitMQ, Redis, Chrome Selenium, and Firefox ● Monitor the progress and the logs in real time, unlimited history ● Managing workflows with models for cloning, exporting and import pipelines ● Support for and integration of first class Git
Licence	Buddy is a commercial free tool.
Homepage	https://buddy.works/

8 Appendix B: Continuous Inspection

Table B.1 lists the most important tools for Continuous Inspection to test the quality of code.

Table B.1: Continuous Inspection

SonarQube	
SonarQube is an open-source platform developed by SonarSource for continuous inspection of code quality to perform automatic reviews with static analysis of code to detect bugs, code smells on 20+ programming languages. SonarQube offers reports on duplicated code, coding standards, unit tests, code coverage, code complexity, comments, bugs, and security recommendations.	
Platforms	Windows, macOS, Linux, Browser
Licence	Free
Homepage	https://www.sonarqube.org/
PyCharm	
PyCharm is an IDE with a rich set of tools for Python developers. The software was developed by JetBrains, and it is available for Windows, Mac, and Linux. PyCharm will analyze, test, and debug code. It also integrates with the Django web development platform. An open-source version of the IDE is available for free, and paid plans are available which offer more features.	
Platforms	Windows, macOS, Linux, Browser
Licence	Commercial
Homepage	https://www.jetbrains.com/pycharm/
Codacy	
Codacy is a software which allows for the automated code testing and reviewing of a piece of programming.	
Platforms	Browser
Licence	Freemium
Homepage	https://www.codacy.com/
Klocwork	
Klocwork is a static code analysis and SAST tool for C, C++, C#, and Java that identifies software security, quality, and reliability issues helping to enforce compliance with standards. This has made Klocwork the preferred static analyzer that keeps development velocity high while enforcing continuous compliance for security and quality.	
Platforms	Browser

Licence	Commercial
Homepage	https://www.perforce.com/products/klocwork
DeepSource	
DeepSource continuously analyzes source code changes to find and fix issues classified as security, performance, anti-patterns and bug-risks.	
Platforms	Browser
Licence	Freemium
Homepage	https://deepsource.io/
WhiteSource	
WhiteSource offers an open source license management and security solution. WhiteSource automates the entire process of open source selection, approval, detection of vulnerable or problematic components and remediation. Recognized by Forrester SCA Wave 2017 as best current offering.	
Platforms	Browser
Licence	Commercial
Homepage	https://www.whitesourcesoftware.com/

9 Appendix C: Vulnerability Scanning

Table C.1 shows a list of the most important vulnerability scanning tools.

Table C.1: Vulnerability Scanning

Anchore	
Anchore is an open-source project for deep analysis of docker images.	
Features	• Provides deep inspection of container images, OS packages, software artifacts such as jar files • Integrates with your CI/CD pipeline seamlessly to find security breaches • Defines and applies policies to prevent building and deploying dangerous images • Check for only certified and secured images before deploying them on an orchestration platform. • Customise checks for vulnerabilities, configuration files, image secrets, exposed ports, etc. • It also offer a standalon tool for analysis called Grype
Licence	Apache License 2.0
Homepage	https://github.com/anchore/anchore-engine https://github.com/anchore/grype
Clair	
Clair is an open-source project which offers static security and vulnerability scanning for docker and application (appc) containers.	
Features	• Scans for existing vulnerabilities and prevents them from being introduced in the future. • Provides REST API for integration with other tools • Sends a notification when it identifies any vulnerability • Provides report in HTML format with all the details of the scan • Updates metadata at regular intervals
Licence	Apache License 2.0
Homepage	https://github.com/quay/clair
Dagda	
Dagda is an open-source tool for static analysis of known vulnerabilities such as trojans, malware, viruses, etc. in docker images and containers. It uses the ClamAV antivirus engine to detect such vulnerabilities.	
Features	• Supports multiple Linux images (CentOS, Ubuntu, OpenSUSE, Alpine, etc.) • Analyses dependencies from java, python node js, javascript, ruby, PHP • Integrates with Falco for monitoring the running containers

	● Stores each analysis report in MongoDB to maintain the history of each docker image or container
Licence	Apache License 2.0
Homepage	https://github.com/eliasgranderubio/dagda
Harbor	
Harbor is an open-source and trusted cloud native registry that provides security policies and role-based access control (RBAC). It stores, signs, and scans docker images for vulnerabilities. It can be installed on a Kubernetes cluster or any other system which supports Docker.	
Features	● Easily deployable using Docker Compose ● Provides security and vulnerability analysis ● Multi-tenant content signing and validation ● Identity integration and role-based access control ● Extensible API and User Interface ● Image replication between instances ● Supports LDAP/AD and OIDC for user management and user authentication
Licence	Apache License 2.0
Homepage	https://goharbor.io/

10 Appendix D: Artifact Repositories

Table D.1 shows a list of artifact repositories.

Table D.1: Artifact Repositories

JFrog Artifactory	
Artifactory is the enterprise-ready repository manager available today, supporting secure, clustered, High Availability Docker registries.	
Licence	Commercial
Homepage	https://jfrog.com/
Nexus Repository Manager	
Nexus Repository Manager manages components, builds artifacts, and releases candidates in one central location.	
Licence	Commercial
Homepage	https://www.sonatype.com/products/repository-pro
npm	
npms paid products and services offer teams and companies ways to organise, share, and secure code, integrate npm with testing and deployment tools, and bring code reuse into the enterprise.	
Licence	Commercial
Homepage	https://www.npmjs.com/
Github Package Registry	
GitHub Package Registry is a software package hosting service, similar to npmjs.org, rubygems.org, or hub.docker.com, that allows customers to host packages and code in one place. Customers can host software packages privately or publicly and use them as dependencies in their projects.	
Licence	Commercial
Homepage	https://docs.github.com/en/packages

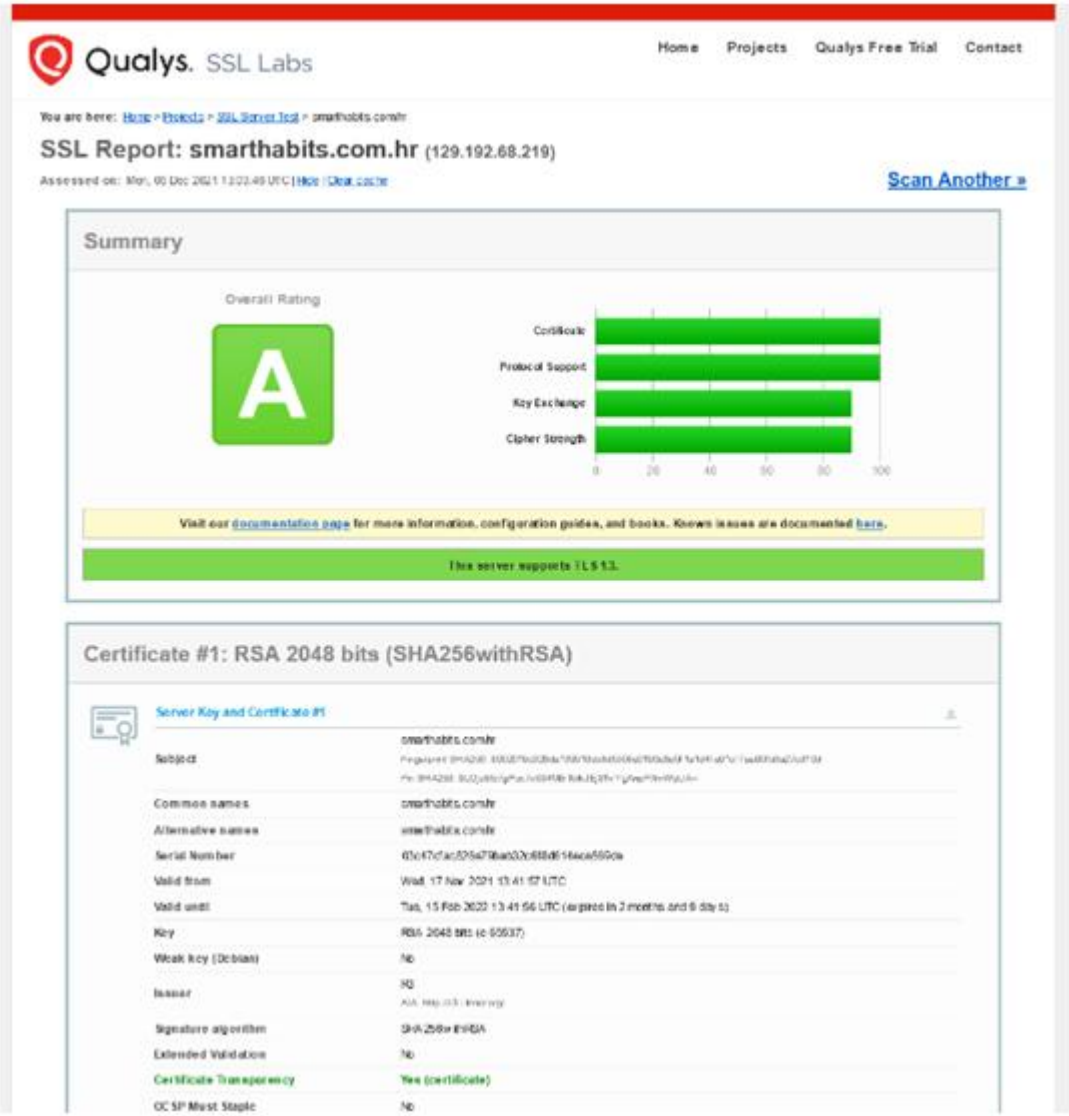
11 Appendix E - Testing activities

This appendix reports some details concerning the testing activities performed to validate the development of the SmartHabits and ThingsBoard components of the PHArA-ON project.

Apache JMeter is the open-source tool used for load testing to analyse and measure the efficiency and performance of the services especially the services are web applications. It was used for performance testing of SmartHabits Platform HTTP REST and MQTT input interfaces.

Figure A provides an excerpt from the SSL test to ensure data privacy and security.

Figure E.1: Excerpt from the SSL test to ensure data privacy and security



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[5] open source performance and testing tool was used.

For the selection of the tools the tools survey[6] for PHArA-ON was consulted and Gatling was selected from the "Load test" section in the "Test" tab, while Zabbix was selected from "IT infrastructure Monitoring" section from the "Monitor" tab.

Important thing that is not noted on the Gatling website is that a Proxy server is needed, so the recorder can be able to capture packets. Proxy server needs to be set on a local host and port Gatling is listening (port that is entered in the Gatling GUI field). In that way, all traffic goes through this proxy server (after leaving on default HTTP port 80) and can be recorded. How to set a proxy server is explained here[7]. If the proxy server refuses connection, try restarting the browser, or computer.

Performance tests were done with both cloud and local lab deployment of the ThingsBoard CE platform serving as an IoT layer of the SmartHabits platform (SHP).

On one computer ThingsBoard service was started, and on the other, ThingsBoard was visited on a local IP address and specified port (where ThingsBoard service is active) from the browser. With described actions, the scenario needs to be recorded and then modified. Once again the Proxy server needs to be configured. It can be problematic to make it work, because sometimes proxy will refuse connection, restarting computer can do the trick. Scala code that was used is represented on the code snippet reported in Figure B (creating devices). Then this simulation was run on two computers (name of device must be modified so there wouldn't be two same requests for creating devices) at the same time. It took about 30 minutes for it to be over. On the computer where the ThingsBoard service was active, Zabbix agent was configured, and connected to Zabbix server configured on another machine. It is important to emphasise that server and agent versions must be compatible. Zabbix is open-source software designed for real-time monitoring of server performance (CPU or memory utilisation, packet loss rate). Link to the used manual is available here[8]. Figure C shows the testing set-up.

[5] Gatling - https://gatling.io/open-source/

[6] PHArA-ON tools survey - https://gitlab.com/pharaongroup/developers-handbook/-/wikis/Tools/tools-survey

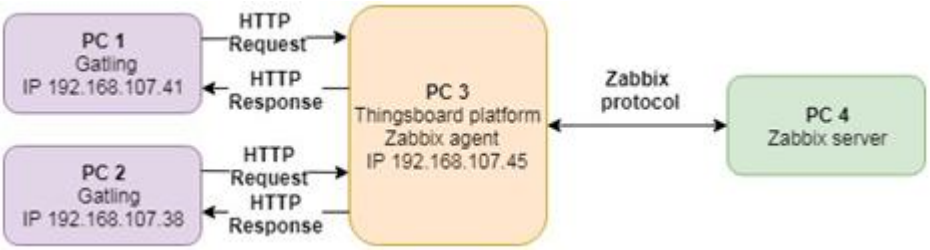
[7] https://www.digitalcitizen.life/how-set-proxy-server-all-major-internet-browsers-windows

[8] https://www.zabbix.com/documentation/3.4/manual

Figure E.2: Scala modified code for simulation on Local CE ThingsBoard

```
package createdevicepc2

import scala.concurrent.duration._

import io.gatling.core.Predef._
import io.gatling.http.Predef._
import io.gatling.jdbc.Predef._

class PC2CreateDevice extends Simulation {

    val httpProtocol = http
        .baseUrl("http://192.168.107.45:8080")
        .inferHtmlResources()
        .acceptHeader("application/json, text/plain, */*")
        .acceptEncodingHeader("gzip, deflate")
        .acceptLanguageHeader("en-US,en;q=0.5")
        .userAgentHeader("Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:68.0) Gecko/20100101
Firefox/68.0")

    val headers_0 = Map(
        "Accept" -> "text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8",
        "Upgrade-Insecure-Requests" -> "1")

    val headers_1 = Map("Accept" -> "text/css,*/*;q=0.1")

    val headers_2 = Map("Accept" -> "*/*")

    val headers_4 = Map("X-Authorization" -> "Bearer
eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJ0ZW5hbnRAdGhpbmdzYm9hcmQub3JnIiwic2NvcGVzIjpbIlRFTkFOVF9BRE1JTiJdLCJ1c2
VySWQiOiJhYjVkODNiMC1hN2MxLTExZTktODA4Ni05ZjZmN2FmNzNmZWIiLCJlbmFibGVkIjp0cnVlLCJpc1B1YmxpYyI6ZmFsc2UsI
nRlbmFudElkIjoiYWIyZjIwYjAtYTdjMS0xMWU5LTgwODYtOWY2ZjdhZjczZmViIiwiY3VzdG9tZXJJZCI6IjEzODE0MDAwLTFkZDIt
MTFiMi04MDgwLTgwODA4MDgwODA4MCIsImlzcyI6InRoaW5nc2JvYXJkLmlvIiwiaWF0IjoxNTY3NDk3NTk2LCJleHAiOjE1Njc1MDY
1OTZ9.VqEKXlyTr297nxyiFQThGrLNPMnedvaBVM4dKv4_crkR0-jSld1lUJ-t5P-fU4tH-X5CDsdliVFHbhBbJ-LrAQ")

    val headers_15 = Map(
        "Content-Type" -> "application/json;charset=utf-8",
        "X-Authorization" -> "Bearer
eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJ0ZW5hbnRAdGhpbmdzYm9hcmQub3JnIiwic2NvcGVzIjpbIlRFTkFOVF9BRE1JTiJdLCJ1c2
VySWQiOiJhYjVkODNiMC1hN2MxLTExZTktODA4Ni05ZjZmN2FmNzNmZWIiLCJlbmFibGVkIjp0cnVlLCJpc1B1YmxpYyI6ZmFsc2UsI
nRlbmFudElkIjoiYWIyZjIwYjAtYTdjMS0xMWU5LTgwODYtOWY2ZjdhZjczZmViIiwiY3VzdG9tZXJJZCI6IjEzODE0MDAwLTFkZDIt
MTFiMi04MDgwLTgwODA4MDgwODA4MCIsImlzcyI6InRoaW5nc2JvYXJkLmlvIiwiaWF0IjoxNTY3NDk3NTk2LCJleHAiOjE1Njc1MDY
1OTZ9.VqEKXlyTr297nxyiFQThGrLNPMnedvaBVM4dKv4_crkR0-jSld1lUJ-t5P-fU4tH-X5CDsdliVFHbhBbJ-LrAQ")

    object DevCreate {
    val devicecreate = exec(http("devicecreate")

            .get("/devices")
            .headers(headers_0)
            .resources(http("request_1")
```

Figure E.3: Testing setup



Stress test duration was 36 minutes after which Gatling self-generated reports containing graphs such as total executions, successful executions, failed executions, response times, active users along simulation, etc.

Figure D shows the number of requests per second. The graph was generated after the script was done running. A steady decline in a number of requests is noticeable during simulation.

Figure E.4: Number of requests per second

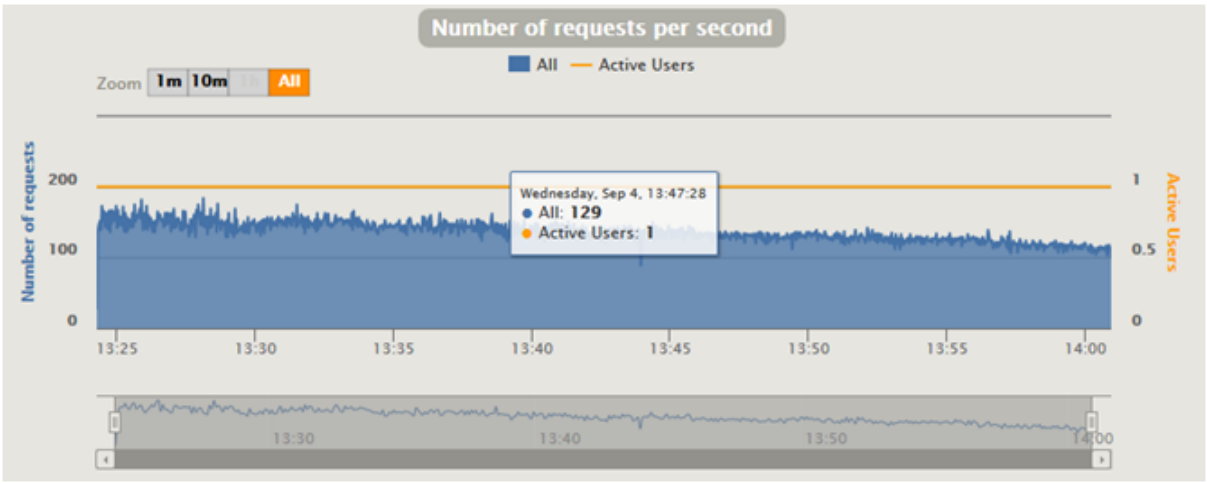


Figure E shows the number of responses per second. Gatling generates the graph upon script completion. As with a number of requests per second, a steady decline in the number of responses per second can be observed.

Figure E.5: Number of responses per second

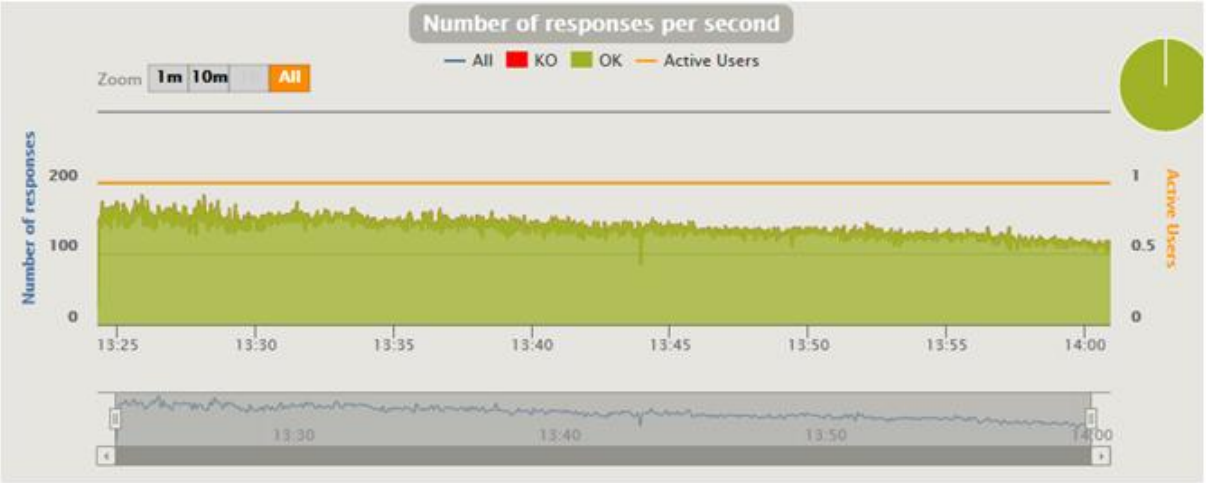


Figure F shows statistics regarding execution and response times. Under “execution”, four important columns are “OK”, “KO”, “%KO” and “Cnt/s”. Column “OK” represents successfully executed requests, “KO” represents unsuccessfully executed requests, “%KO” represents percentage of unsuccessful requests, and “Cnt/s” column represents count of events per second. “Response time” section contains data about response times of executed requests. Column “99th pct” shows response time of requests that were faster than 99% of other requests.

Once both scripts were done executing commands, Zabbix server was checked to see if there was any load increase on the computer that was stress tested. Zabbix server offers CPU, memory, network, and storage tracking.

Figure E.6: Execution statistics and response times

STATISTICS — Expand all groups | Collapse all groups

Requests	Executions					Response Time (ms)							
	Total	OK	KO	% KO	Cnt/s	Min	50th pct	75th pct	95th pct	99th pct	Max	Mean	Std Dev
Global Information	300001	300001	0	0%	136.302	4	18	28	41	54	550	21	11
devicecreate	15000	15000	0	0%	6.815	4	5	6	7	9	105	5	1
thingsboard.ico	1	1	0	0%	0	63	63	63	63	63	63	63	0
style.9e...2948.css	1	1	0	0%	0	80	80	80	80	80	80	80	0
bundle.4...03ebf.js	1	1	0	0%	0	255	255	255	255	255	255	255	0
vendor.4...03ebf.js	1	1	0	0%	0	42	42	42	42	42	42	42	0
request_5	15000	15000	0	0%	6.815	8	21	26	35	46	105	22	7
request_4	15000	15000	0	0%	6.815	7	17	22	31	41	97	20	6
request_6	15000	15000	0	0%	6.815	9	28	31	38	55	128	28	7
request_7	15000	15000	0	0%	6.815	7	17	20	43	54	219	21	10
request_8	15000	15000	0	0%	6.815	6	16	19	42	53	105	20	10
request_9	15000	15000	0	0%	6.815	15	30	33	42	53	108	31	6
request_11	15000	15000	0	0%	6.815	6	15	17	22	40	319	15	6
request_12	15000	15000	0	0%	6.815	5	11	14	20	37	103	12	6
request_13	15000	15000	0	0%	6.815	5	7	10	15	33	89	9	5
request_14	15000	15000	0	0%	6.815	15	21	24	27	41	113	21	4
request_15	15000	15000	0	0%	6.815	18	26	29	34	53	201	27	6
request_16	15000	15000	0	0%	6.815	17	36	45	53	66	550	38	11
request_17	15000	15000	0	0%	6.815	8	28	34	54	62	167	29	12
request_18	15000	15000	0	0%	6.815	7	22	31	38	55	166	25	9
request_19	15000	15000	0	0%	6.815	8	24	31	46	59	176	26	10
request_20	15000	15000	0	0%	6.815	6	20	30	35	46	318	24	8
request_1	14999	14999	0	0%	6.815	5	16	18	26	32	85	17	4
request_2	14999	14999	0	0%	6.815	5	16	18	27	32	89	17	4
request_3	14999	14999	0	0%	6.815	6	16	19	27	32	84	18	4

Figure G shows CPU load during stress test. A steady increase in CPU load can be observed throughout the whole stress test. About twenty minutes into the stress test, a spike in CPU load occurred.

Figure H displays also CPU load, but over a longer time frame. At 14.20 the test started, which is clearly shown by the increase in CPU usage. Around 14.56 both Gatling scripts ended, which is also visible by the decrease in CPU usage.

Figure E.7: CPU load (1)

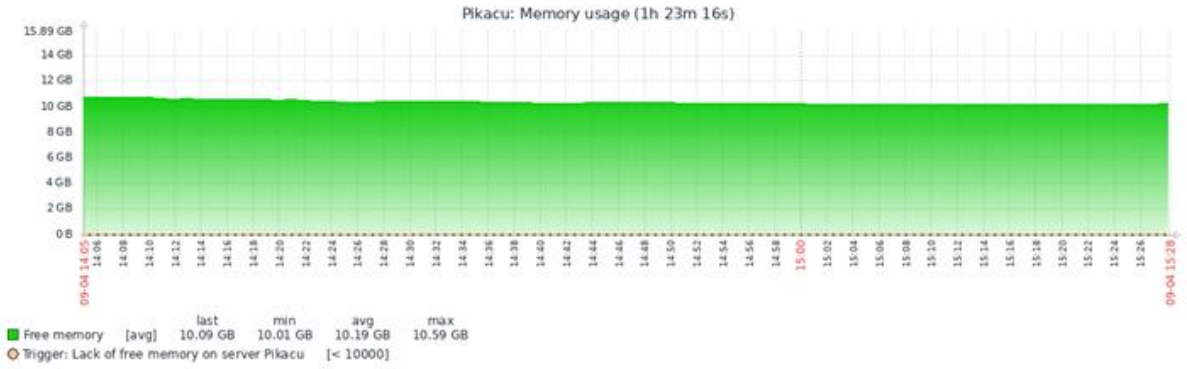
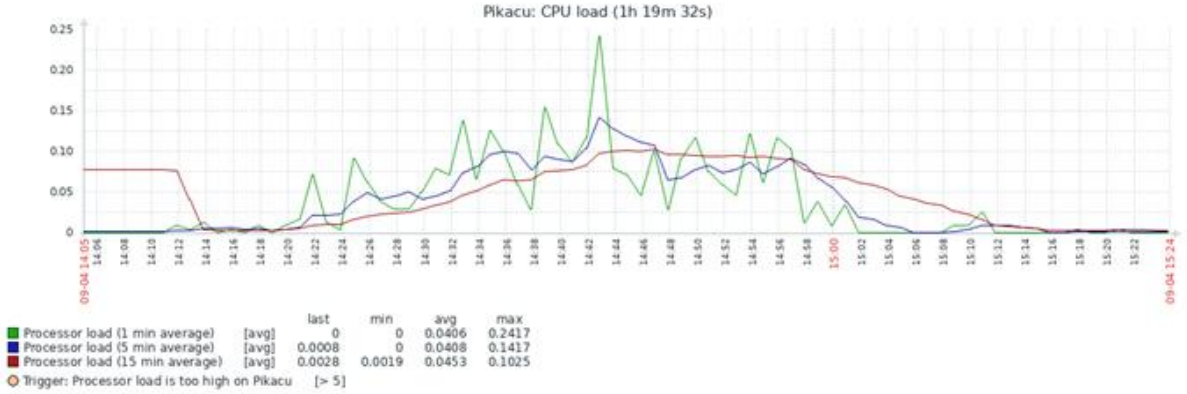


Figure E.8: CPU load (2)



Memory usage showed no major difference before, during, or after stress testing.