Horizon 2020 Project LETHE
"A personalized prediction and intervention model for early detection and reduction of risk factors causing dementia, based on AI and distributed Machine Learning."

Research and Innovation Action
H2020-SC1-DTH-2020-1
GA 101017405

Duration: 48 months from 01/01/2021
Coordinator: Sten Hanke, FH JOANNEUM GESELLSCHAFT MBH

Deliverable ID.:	**D4.3**	
Deliverable title:	**LETHE sensing ecosystem**	
Planned delivery date:	30/04/2022 (M16)	
Actual delivery date:	08/07/2022 (M18)	
Editor:	Vasileios Loukas (FORTH), Lefteris Koumakis (FORTH), Thomas Kassiotis (FORTH), Georgia Karanasiou (FORTH)	
Contributing partners:	FORTH	
Internal reviewer:	Lelia Ataliani (INFO)	
Checked and released by:	Sten Hanke (FHJ)	
Dissemination Level:	**X**	PU = Public;
		CO = Confidential
Type:	Other	

This project has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 101017405.
This deliverable reflects only the authors' view and the Commission is not responsible for any use that may be made of the information it contains.

Document information and history

Deliverable description (from DoA)
This deliverable presents in detail the sensing ecosystem of LETHE along with supporting services.

Please refer to the Project Quality Handbook for guidance on the review process and the release numbering scheme to be used in the project.

Release number	Release date	Author [Person and Organisation]	Milestone*	Release description /changes made
V. 0.1	30/01/2022	Vasileios Loukas, Georgia Karanasiou (FORTH)	ToC	
V. 0.2	15/03/2022	Vasileios Loukas, Lefteris Koumakis, Thomas Kassiotis, Georgia Karanasiou (FORTH)	Intermediate	
V. 0.3	20/03/2022	Vasileios Loukas, Lefteris Koumakis, Thomas Kassiotis, Georgia Karanasiou (FORTH)	Proposed	
V. 0.4	15/04/2022	Vasileios Loukas, Georgia Karanasiou (FORTH)	Revised	Included review recommendations
V. 0.5	04/07/2022	Vasileios Loukas, Georgia Karanasiou (FORTH)	Approved	
V. 1.0	08/07/2022	Vasileios Loukas, Lefteris Koumakis, Thomas Kassiotis, Georgia Karanasiou (FORTH)	ER	Included review recommendations

** The project uses a multi-stage internal review and release process, with defined milestones. Milestone names include abbreviations/terms as follows:*

- *TOC = "Table of Contents" (describes planned contents of different sections)*
- *Intermediate: Document is approximately 50% complete – review checkpoint*
- *ER = "External Release" (i.e., to commission and reviewers)*
- *Proposed: document authors submit for internal review*
- *Revised: document authors produce new version in response to internal reviewer comments*
- *Approved: Internal project reviewers accept the document.*

Table of Contents

Table of Figures

List of Tables

1 Executive Summary

This document provides a detailed analysis of the design and implementation of the LETHE Sensing Ecosystem. The present deliverable D4.3 provides an overview, as well as relevant guidelines for the technical implementation of the sensing ecosystem. This framework constitutes one of the core components of the LETHE ecosystem as it provides the ability to monitor the status and abilities of the target population, thus introducing the digital biomarkers needed for the eFINGER interventional model. Initially, current trends in the relevant literature are presented as the state-of-the-art in sensing ecosystems. Consecutively, a market analysis that was conducted to pinpoint the most suitable wearable device for the scope of the LETHE project is presented for all the available smartwatch sensors. Following this, the requirements (user, functional, non-functional) of the middleware ecosystem are provided. The technical implementation is presented and the digital biomarkers harvested are thoroughly analysed. The methods for passive and active information collection developed are reported. The integration to the LETHE Data Lake is described. The RADAR-Kubernetes deployment of the LETHE sensing ecosystem is discussed and compared with the custom platform that was built. Finally, an overview is given along with the current status of the sensing ecosystem and its limitations.

2 About this Document

This document contains literature review, market analysis and all the technical characteristics of the design and implementation of the LETHE Sensing Ecosystem. The establishment and collection of the digital biomarkers will be used and will assist in the comprehensive assessment of cognitive status, functional abilities, behavioural and psychological status, and as a result this submodule of the LETHE project is crucial in the integration of the digital biomarkers to the retrospective data harvested from the patients. As a result, this document describes the process and guidelines for building the middleware sensing platform and introduces novel digital biomarkers that fall under the specific scope of the LETHE project.

2.1 Role of deliverable

This deliverable refers to the implementation scheme regarding the LETHE Sensing Ecosystem. The present deliverable builds the basis (from the interoperability aspect) for further technical implementation of the LETHE Platform architecture and modules regarding data collection. The role of this deliverable is to provide the guidelines for implementing and deploying an end-to-end streaming data platform for the uses of the clinical trial of the LETHE project. Furthermore, the role of this deliverable is to provide all the relevant information regarding the digital biomarkers which are collected from the LETHE sensing ecosystem, such as Measurements, Data Types, and Data values. Finally, Del. 4.3 provides a holistic approach for the sensing ecosystem not only from the internal perspective of the middleware platform but also at the project-level such as interoperability and interaction with the LETHE Data Lake, which acts as a data depository for all the components gathering data (LETHE sensing ecosystem, LETHE app, cCog, cTrain, etc) for the LETHE project.

2.2 Relationship to other LETHE deliverables

The following relationships to other LETHE deliverables and tasks were identified:

- Task 2.2: LETHE System Architecture: Decisions on Use Cases, Middleware, Platforms. As the middleware sensing ecosystem is a core component of the prospective data collection and decisions on this regard will affect the LETHE System Architecture.
- WP3: AI Models. As the sensing ecosystem provides digitally collected data as well as processed data in the form of biomarkers to the extended AI Models.
- WP4: Digital Biomarkers Data Set as basis for interoperability recommendations. As the data that get produced affect the knowledge gained regarding patients influencing the use cases for smart glasses and robot, as well as the formation of daily living activities profiling.
- WP5: Setup of LETHE big data infrastructure where all retrospective and prospective data will be inter-connected and commonly presented. As the abilities of the sensing ecosystem need to be accommodated appropriately by the subsequent hosting infrastructure, such as the LETHE Data Lake and LETHE Data Warehouse
- Task 6.1: Data Protection. As a core component in the data generation, appropriate data protection strategies need to be followed.
- Task 6.2: Security. As security guidelines need to take into consideration the choices made in the implementation of the sensing ecosystem.

2.3 Structure of the document

The document provides information on the following:

- State-of-the-Art in biomedical sensing ecosystems.
- Market Analysis for wearable devices for the provision of digital biomarkers.
- The user, functional and non-functional requirements for the LETHE middleware sensing ecosystem.

- Description of the basic components, architecture and functionality for RADAR-base, Kubernetes and their newly introduced tightly-coupled configuration.
- Measurements, Data Types, Data Values to be collected from the wearable devices and mobile sensors from the LETHE middleware platform.
- Integration of the LETHE middleware platform to the LETHE Data Lake.
- Deployment of the middleware platform on multi-nodal clustered EGI premises, based on the minimum deployment requirements of RADAR-base Kubernetes.

Therefore, Chapter 3 provides information and literature review regarding the state-of-the-art in implementation of sensing ecosystems, whereas Chapter 4 presents a market analysis regarding wearable devices, that was conducted by FORTH. Chapter 5 gives an overview of the identification of the basic user requirements and functional and non-functional specifications. Chapter 6 initially provides thorough information regarding RADAR-base, Kubernetes and an overview of the concepts and components of all the services provided within our architecture. Chapter 7 describes the measurements, data types and data values provided by the Fitbit smartwatches as well as the mobile sensors data that are to be collected by the subjects during the clinical trial. Chapter 8 gives information regarding the active and passive data collection. Chapter 9 provides information regarding the integration of the LETHE Sensing Ecosystem to the LETHE Data Lake. Chapter 10 describes the deployment of the LETHE Sensing Ecosystem on multi-nodal, clustered Virtual Machines provided by EGI. Finally, Chapter 11 discusses about the custom-built platform. Chapter 12 gives and overview discussion and the current status and limitations of the LETHE Sensing Ecosystem.

2.4 Relation to the DoA

Table 1 Actions taken in relation to the DOA.

Description in DoA	Actions
One of the core components in the LETHE ecosystem is the sensing framework for monitoring the status and abilities of the target population.	The LETHE sensing ecosystem for monitoring of sensor data in the project has been designed, implemented and deployed. Deliverable D4.3 provides details about the architecture and its functionality.
The LETHE sensing ecosystem (LES) will provide the necessary digital biomarkers necessary for the adopted extended eFINGER interventional model.	The LETHE sensing ecosystem can support and monitor all the digital biomarkers that have been identified by the clinical partners as part of the clinical trial, as described in Sections 7, 8.
During this task the necessary APIs will be developed for the collection of the passive and active information along with the metadata generated on the sensor platforms. LETHE partners will exploit the provided capability of the proposed sensing infrastructure for in-platform development of the LETHE apps.	The integration of the LETHE sensing ecosystem with the LETHE Data Lake and as a result with the final LETHE platform has been designed, as described in Section 9.
Moreover, LETHE will utilise RADAR-base open platform which its intended use is the generalised mHealth data collection. The acquisition of the sensors raw data will be utilised by the personalised self-management system (PSMS) of LETHE for the comprehensive assessment of cognitive status, functional abilities, behavioural and psychological symptoms.	LETHE sensing ecosystem is based on the RADAR-base open-source platform and specifically on the latest Kubernetes version of it. Due to delays in the stable version of RADAR-base, we decided to work in parallel on an alternative platform which is able to support the sensing ecosystem. Towards that direction, FORTH developed, deployed and tested an in-house platform able to gather information from wearables. Since RADAR-base has been successfully deployed in the development servers of LETHE, the consortium decided to follow the initial plan and use the extended RADAR-base platform as the LETHE sensing ecosystem, Section 6.

2.5 List of Abbreviations

Table 2 List of abbreviations

Abbreviation	Explanation
API	Application Programming Interface
aRMT	active Remote Monitoring Technology
DRF	Django Rest Framework
eCRF	electronic Case Report Form
ETCD	Etc Distributed
IoT	Internet of Things
JBOD	Just a Bunch of Disks or Just a Bunch of Drives
JBOF	Just a Bunch of Flash
HRV	Heart Rate Variability
kubectl	Kube command line tool
pRMT	passive Remote Monitoring Technology
PromQL	Prometheus Query Language
RADAR-base	Remote Assessment of Disease And Relapse
REST	Representational State Transfer
QR	Quick Response

3 State of the Art in sensing ecosystems

The majority of the elderly people suffer from chronic diseases [1], which dramatically decrease their standard of living and lead to social withdrawal or even death. The nature of these diseases is most likely cardiovascular [2],[3], respiratory [4], cancer [5], or cognitive disorders [6],[7]. Dementia constitutes the main cause of disability in elderly people and is estimated to affect 50 million people worldwide. A very important tool for tracking the progression of several diseases and making lifestyle interventions, including cognitive disorders, is patient monitoring.

Monitoring is defined as continuous measurement of patient parameters such as heart rate, blood pressure and oxygen saturation, among many others that manifest disease symptoms. As the digitization era has affected every aspect of contemporary life, a vast increase in non-invasive sensors has been exhibited, leading in increased amounts of digitally collected health related data. The technological progress in the field of wearable devices offers the possibility to create ecosystems, platforms, and applications with which important biomarkers for each subject can be monitored.

Digital remote monitoring of patients has become the intersection of biomedical engineering and computer science, leading to accurate tracking, analysing and modelling human behaviour through the establishment of digital biomarkers of acquired patient data. Several clinical trials have been conducted which required patient monitoring through the usage of wearable sensors. Respectively, several different platforms have been designed and implemented to accommodate the needs of these clinical trials depending on the constraints of the study, the characteristics of interest and the devices used. The aim of this section is to provide an overview of state-of-the-art methods, frameworks, and platforms that have been developed for remote patient monitoring.

In Cloud-based web services the Representational State Transfer technology (REST) is employed, to provide 3rd-party client application interoperability [8]. REST technology is heavily utilized to implement platform-wise intercommunication of resources. Nevertheless, this protocol communication does not procure for any kind of standardization of data format and transport. The only kind of standardization could be established in the development cycle between the developer teams, as a common way of integrating the communication protocol. Usually, platform development teams design proprietary data schemas for understandable and “standardized” data transfer. This is an open challenge while integrating either 3rd party client applications or devices into systems. Plethora of relevant studies in the field resort to custom-made platforms through REST which has become a very popular technology, such as the following.

The PRISM (Passive, Real-time Information for Sensing Mental Health) platform [9], leveraged passive sources of data for continuous monitoring of users’ health, to estimate possible correlation between walking outside and sun exposure with the progress of depression. The BEIWE platform [10], studied the relapse prediction in schizophrenia through digital phenotyping collecting smartphone sensor data including GPS, accelerometer, voice call logs, text message logs, and screen event data and usage pattern. Bhavsar et al. [11], also relied on a custom platform architecture for data collection from wrist-worn wearables to estimate various fitness and physiological parameters. Andrea et al. [12] proposed an integrated sensor network, which continuously monitors the surroundings of the user as well as his/her home to infer knowledge regarding the behavior and health status of the user. The platform interconnects and harvests data from different sensors that have been integrated such as wearables, biomedical sensors as well as unobtrusive sensors. The functionality of the system referred to data collectiong regarding breath analytics, blood pressure, ECG and other similar biomarkers. Abdelsalam et al. [13], realized another smart-home based architecture for continuous monitoring and assessment of diabetic patients. The proposed solution monitored activity, diet, and nutrition habits as well as evaluated the effect of different treatments’ response and compliance of the patients to the protocol followed. Different wearables were used to monitor all the different categories of digital data. The SMART Bear project [14], designed a smart-home platform solution to provide measurable improvements to the Quality of Life of the elderly and their ability to live independently.

On the other hand, health platforms provide a limited support functionality when it comes to smartphones and import of manually captured user data. Additionally, data-mining competences are rarely granted to such middleware platforms, as middleware acts mostly as a data aggregator. Usage of such functionalities could make it standardized and more secure for platforms to integrate further devices and user-captured data as well as data sharing with health care providers, without the use of 3rd party cloud-based services.
A mediator platform that has been widely used is Remote Assessment of Disease and Relapse - RADAR-base [15]. The RADAR-base platform is designed to meet the needs for collection of high-resolution data at scale and additionally facilitate a new generation of mHealth projects in this relevant field. Also, it provides a highly scalable end-to-end solution for event-driven messaging, which is able to satisfy a wide variety of use cases, for example, high throughput, low latency messaging, real-time data processing, and fault tolerance/robustness.
The most recent projects with relevant scientific context including a sensing ecosystem in the clinical trial setup, employing the RADAR-BASE technology [15], are among others the following: The TIMESPAN: ART-CARMA study [4], which monitors remotely adults with Attention-Deficit / Hyperactivity Disorder (ADHD), through smartwatch measurements and active and passive app data collection. The COVID-RED study [16], which evaluates the Ava wearable device with respect to usage and performance, through data collection from sensors measuring variables such as breath rate, pulse rate, skin temperature and heart rate variability. The explicit purpose of this study was the monitoring, with ultimate goal the early detection, of COVID-19 patients regarding general and high-risk populations. The UNFOLD study [17], targeted the “person in recovery” population, with primary aim the assessment of people recovering from mental and ill-health as well as the progression of their status and the processes involved as in-thereof. The DynaMORE study [18], which aims to promote stress resilience and improve mental health and well-being in the face of adversity. A similar study with the LETHE project, the RADAR AD project [19], aims to improve the assessment of the progression of the Alzheimer’s disease. Specifically, through the Radar-base platform in conjunction with smart-phones, wearables, and home-based sensors, the cognitive and functional decline is measured and directly correlated with the progression of the disease. The RADAR AD project has also been introduced to the LETHE Consortium, and has offered insight on critical issues regarding several aspects of the project. A plethora of studies are currently relying in the RADAR-base middleware platform architecture, as reported in [20], [21].

4 Market Analysis

Prior to the selection of the appropriate platform to be deployed for the explicit purpose of accommodating the needs of the LETHE project and the upcoming clinical trial of the project, the need for a market analysis has arisen regarding the suitability of possible wearable devices. This market analysis' purpose was two-fold: (i) to obtain timely-critical information on the market of wearables, thus reevaluating the technology stated in the proposal of the project, and (ii) to consult with the clinical partners regarding the functionality provided from every device.

The choice of the wearable devices to be used in the clinical trial affects not only the prospective data to be collected from the patients, but also the whole design and implementation of the middleware sensing ecosystem. FORTH conducted the respective research regarding various contemporary, commercially-available devices. Although several devices were identified as potential candidates, the most suitable ones – Fitbit Sense and Fitbit Charge 4 for the interventional and the control groups respectively – were selected as having the best rate in the trade-off between functionality, price and API availability. A further major characteristic in favor of Fitbit devices is the compatibility and integration to the RADAR-base platform, which advances even further with the novel deployment architectural strategy featuring Kubernetes.

The final decision on the choice of wearable devices is of paramount importance for all the aspects of this study, and as a multivariate constrained problem, both clinical and technical researchers were involved in the outcome. Thus, the final decision made came as a result of the collaboration of an established working group on behalf of both technical and clinical partners. Table 3 provides the potential candidates for wearable devices that were identified in the market analysis and their characteristics.

In case there are newer releases of these watches with improved functionality, in time for the clinical trial, and compatible with the services built, then this decision could be a subject of change. Nevertheless, this could only happen without critically affecting the underlying mechanisms of vital importance for the prospective data collection of the LETHE clinical trial.

Table 3 Market Analysis for contemporary wearable devices and the functionality provided.

			Sensors											Certification				Connectivity			
Name	Type	Cost	Accelerometer	Gyroscope	GPS	Steps count	Sleep (quality)	Heart Rate	ECG	Blood pressure	Galvanic Skin (GSR)	SpO2	Thermometer (Optical)	EC	Clinically Validated	Medical Device	Leads	API	Bluetooth	Cloud storage	Near Real time raw data
Fitbit Sense	Watch	300 €													App		1				
Withings Move ECG	Watch	130 €															3				
KardiaMobile	wearable	90 €															3				
Apple Watch 6	Watch	400 €													App						
Omron HeartGuide	Watch	450 €																			
Empatica E4	Wristband	1.690 €																			
Shimmer ECGmd Kit	wearable	-																			
Savvy	wearable	-															3				
ECG Check	wearable	80 €															3				
WIWE	wearable	290 €															3				

5 Requirements of the LETHE Sensing Ecosystem

The LETHE project follows the ISO 9241-210 (ergonomics of human-system interaction) with its user-centred design development. The international standard ISO 9241-210 provides requirements and recommendations for human-centred design principles and activities throughout the life cycle of computer-based interactive systems. It is concerned with ways in which both hardware and software components of interactive systems can enhance human system interaction. In LETHE, involvement of participants will be guaranteed. Clinical partners of the consortium will invite elderly users to participate in the first set up of the LETHE ICT solution to address human values, needs, emotions, behaviour, and usage of technology items.
The LETHE sensing ecosystem is based on the open-source platform RADAR-base [15] to leverage data from wearables and mobile technology, an open-source software that has been validated and evaluated in the clinical setting [15]. Nevertheless, for the purposes of the LETHE sensing ecosystem we had to assess if the RADAR-base provided an accepted level of core technical requirements the so called functional and non-functional requirements for software development. For the identification of the functional and non-functional requirements we followed the procedures of ISO/IEC 25000 series and specifically the ISO/IEC 25023. The extracted requirements have been our main criteria for the decision of the platform that will support the LETHE sensing ecosystem. Specifically, we followed the procedures of ISO/IEC 25023 and the extracted requirements have been our main criteria for the selection of the platform that will support the LETHE sensing ecosystem.

5.1 User Requirements

The stakeholders of LETHE have been defined in Deliverable D2.2. For the sensing ecosystem the main stakeholders are the patients along with the healthy population that will use the ecosystem for the collection of the prospective data. In terms of use cases, the LETHE sensing ecosystem will support the first use case (User collects and delivers prospective data to the RADAR-base) as it has been defined in D2.2. As a result, the target users of the LETHE sensing ecosystem would be the subjects of the LETHE clinical trial, and the respective user requirement would be that the digital biomarkers to be sent to the LETHE Data Lake for further processing, assessment and evaluation of the progression of their disease.
Essentially, regarding this ultimate requirement, the only thing required from the user would be to use the wearable device and follow the initial procedure for logging into the platform and pair the mobile phone, which will happen during the first visit of the subject at the clinic. Besides this, the subjects will remain oblivious of the middleware platform for the rest of the trial, having no interaction, since the middleware platform's purpose is just to collect the data from Fitbit servers and provide them to the Data Lake.

5.2 Functional Specifications

For the functional requirements we focus on the functional suitability, the performance efficiency and the compatibility of the sensing ecosystem modules. The functional specifications provide the means for a qualitative evaluation of our system. The main functional specifications that were identified can be found in Table 4.

Table 4 The functional specifications of the middleware platform.

Functional Specifications	A.1	**FUNCTIONAL SUITABILITY**
	A.1.1	Functional completeness
	A.1.2	Functional correctness
	A.1.3	Functional appropriateness
	A.2	**PERFORMANCE-EFFICIENCY**
	A.2.1	Time behaviour
	A.2.2	Resource utilisation
	A.2.3	Capacity
	A.2.4	Scalability
	A.2.5	Performance
	A.3	**COMPATIBILITY**
	A.3.1	Co-existence
	A.3.2	Interoperability

5.3 Non-Functional Specifications

For the non-functional requirements specify criteria that can be used to judge the operation of a system. In contrast, functional requirements define specific behaviour or functions. For the non-functional specifications we followed the ISO/IEC 25023 and Table 5 provides the main specifications that were identified.

Table 5 Non-functional specifications of the middleware platform.

Non-Functional	A.4	**USABILITY**
	A.4.1	Appropriateness recognisability
	A.4.2	Learnability
	A.4.3	Operability
	A.4.4	User error protection
	A.4.5	User interface aesthetics
	A.4.6	Accessibility
	A.5	**RELIABILITY**
	A.5.1	Maturity
	A.5.2	Availability
	A.5.3	Fault tolerance
	A.5.4	Recoverability
	A.6	**SECURITY**
	A.6.1	Confidentiality
	A.6.2	Integrity
	A.6.3	Non-repudiation
	A.6.4	Accountability
	A.6.5	Authenticity
	A.7	**MAINTAINABILITY**

	A.7.1	Modularity
	A.7.2	Reusability
	A.7.3	Analysability
	A.7.4	Modifiability
	A.7.5	Testability
	A.8	**PORTABILITY**
	A.8.1	Adaptability
	A.8.2	Installability
	A.8.3	Replaceability

5.4 Verification & Evaluation

Verification and evaluation methodologies aim to test the software during and at the end of the development process and to assure the quality of the desired outcome. Trying to answer the question if the system does the right things, an integration of software life cycle management and risk management activities is recommended. Software validation is accomplished through a series of activities and tasks which are planned and executed at various stages of the software development life cycle.

The LETHE Sensing ecosystem is based on the open-source RADAR-base platform that has been tested (verification) and evaluated. Nevertheless, for the purposes of LETHE we had to re-evaluate the RADAR-base platform if we wanted to reuse it as the backbone of our sensing ecosystem. We did not perform verification to the existing parts of the platform since the system is already functional. Nevertheless, evaluation will provide indications if the platform is the right one for our purposes. Functional and non-functional requirements are part of the software validation procedure that has been delivered by FORTH. A series of technical tests using automated scripts and dedicated software have been used for the technical validation based on the identified functional and non-functional requirements. The results indicate that the RADAR-base can and will be used for the purposes of LETHE.

The evaluation of the LETHE sensing ecosystem and the LETHE platform will be based on the outcome of the clinical trial since evaluation is a procedure that tries to answer the question if the software/tool/platform has an added value to the specific clinical setting.

6 Middleware ecosystem Implementation

The scope of this section is to provide all necessary information on the deployment and usage of the RADAR-base (Remote Assessment of Disease And Relapses) platform and the functionality and communication of its components. RADAR-base is an open-source platform to leverage data from wearables and mobile technologies [15]. The main focus of RADAR-base is seamless integration of data streams from various wearables to collect sensor data (passive and active) in near real-time and store, manage, and share the collected data with researchers for retrospective analysis [1]. Prior versions of RADAR-base were built within the context of containerization, robustness and ease of deployment with the assistance of Docker. Nevertheless, this version of RADAR-base was recently deprecated and would be no longer supported or further developed.
A vast update on the RADAR-base end-to-end platform by The Hyve [22], defined a contemporary update for RADAR-base in terms of the architectural standpoint. One of the most important technologies that the novel RADAR-base platform employs is Kubernetes, which allows automated application deployment, scaling, and management. Besides Kubernetes, RADAR-base has integrated a variety of technologies (Kafka, MinIO, Prometheus, Grafana, Graylog, etc.) essentially providing smooth and secure operation of the platform and monitoring of its core component, with the result of providing an end-to-end solution for remote monitoring use cases.
Before we delve into the core components of the Radar Base platform, the most important features of Kubernetes are introduced. Furthermore, the rationale behind Kubernetes adoption is analysed, being the most suitable technological investment, with respect to container orchestration.

6.1 Kubernetes (K8s)

One of the main problems that developers face is the compatibility issue between the various technologies that may exist in a project. For instance, in a project that contains several technologies that need to work cooperatively in order to ensure the smooth operation of the software, there is a very high probability that some of the following problems might occur.

- Compatibility problems of a component with the OS.
- Compatibility problems between services, depending on the choice of libraries and respective dependencies conflicts.
- Difficulty in migrating the working environment on a new computer.
- Difficulty in deploying the software in production.
- Difficulty of cooperation between developers and operations team.
- Difficulty in maintaining and updating each service separately.

Figure 1 illustrates a typical custom-made platform for a project that contains several technologies. It is apparent that the problems described above are very likely to occur since each technology has its own libraries, dependencies, and an integration of these technologies could become challenging and labour-intensive with respect to the aforementioned issues.

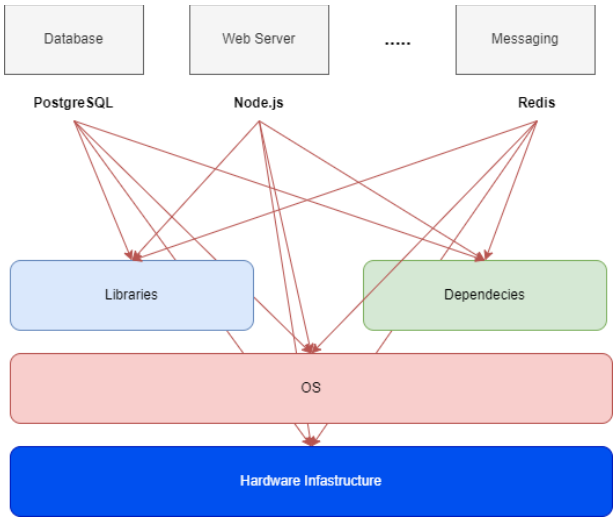

Figure 1 The architecture of typical custom general-purpose platform.

The preceding problem is addressed by the use of containers. Containers are isolated environments, which have their own processes, services, network interfaces, mounts, etc. There are now several technologies with which we can create the containers we need for our application. One of the most well-known technologies for creating containers is Docker, which is also used by the RADAR-base platform.
The process to automatically deploy and manage containers is called Container Orchestration and one of the most widespread technologies in this area is Kubernetes. Kubernetes is a container orchestration technology and is supported by all cloud platforms.

6.1.1 Kubernetes Architecture

Before discussing the core components of Kubernetes [23], it would be useful to thoroughly analyse the notion of the basic concepts that Kubernetes is built upon. The concepts are described below, and Figure 2 illustrates a typical Kubernetes cluster configuration.

- **Node**: A node is a machine (physical or virtual) on which Kubernetes is installed. A node can be a worker node or a master node.
- **Master Node**: It is a node that is configured as a master. The master oversees the other nodes and is responsible for organizing the containers that run on the worker nodes.
- **Worker Node**: The worker nodes launch and run the containers, which contain our applications. If a worker node crashes, then the apps that were running on that node also crash. Note that there are mechanisms in Kubernetes that discover when a node or container has crashed, and perform the necessary actions so that the application that stopped working is available again.
- **Cluster**: A set of grouped nodes that cooperate to perform operations and run programs that have been requested.

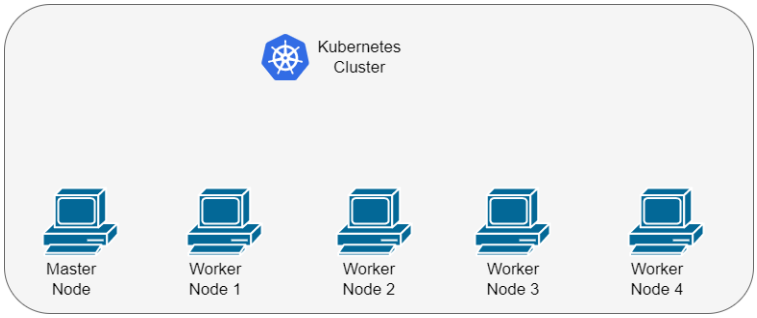

Figure 2 Typical Kubernetes cluster configuration.

6.1.2 Kubernetes Components

The basic components of Kubernetes are the following:

- **API Server.** It is the front end of Kubernetes. Users, management, devices, cmd interfaces all communicate with the API Server so that they can interact with the Kubernetes Cluster.
- **ETCD Service**. This service is a key value store, it is a distributed key value store that Kubernetes uses to store all the data used for cluster management. It is an open-source, distributed, consistent key-value store for shared configuration, service discovery, and scheduler coordination of distributed systems or clusters of machines. ETCD facilitates safer automatic updates, coordinates work being scheduled to hosts, and assists in the setup of overlay networking for containers.
- **Kubelet.** Constitutes the primary "node agent" that is executed on every node. A kubelet is an agent that runs on every node in the cluster. The agent is responsible for ensuring that the containers are running on their respective nodes.
- **Controllers.** Controllers constitute the logic behind orchestration. They are responsible for discovering when various nodes, containers or endpoints are down. They also make decisions to create new containers in such cases.
- **Schedulers.** The schedulers are responsible for allocating the workload or containers to all nodes. Practically, the schedulers assign containers to nodes.
- **Container Runtime.** It is the software that containers execute on. RADAR-base employs the Docker container runtime.

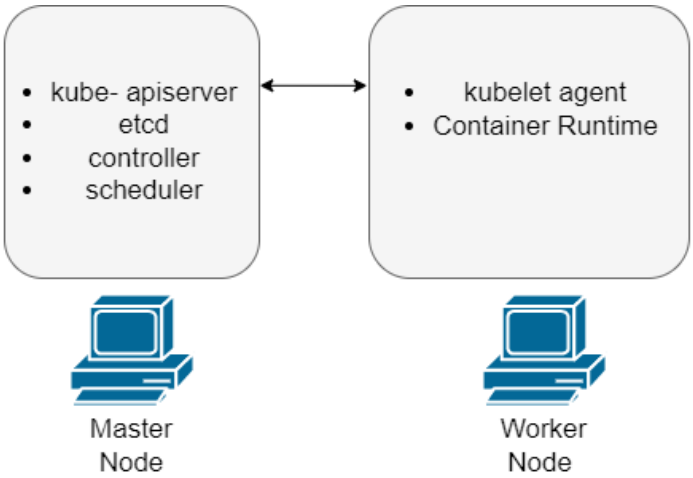

Figure 3 Communication channel between master and worker nodes within Kubernetes architecture.

Figure 3 displays the communication channel between a master and a worker node. The kubelet agent contained in the worker node is responsible for interacting with the master node in order to provide the master node with information about the health of the worker node or to perform operations requested by the master. All the information collected by the master node is stored in a key value store. The key value store that Kubernetes uses, is based on the Etc Distributed - ETCD framework. Finally, the Kube command line tool exists – kubectl, which is used to deploy and manage applications that are governed by a Kubernetes cluster.

6.1.3 Kubernetes Concepts

This section provides an overview and breakdown of the basic Kubernetes concepts that one has to familiarize himself with to gain a thorough understanding of the prevalent technology. Specifically, the most fundamental concepts employed by Kubernetes are the following.

- **Pod**: The most primitive object that can be created in Kubernetes is called a Pod. A pod is an instance of one or more applications. Kubernetes does not deploy containers immediately to worker nodes,

but places them inside an object known as a Pod. When Kubernetes is instructed to create a container, it creates a Pod and subsequently places the container with our application inside the pod. Figure 4 depicts the configuration of a Pod.

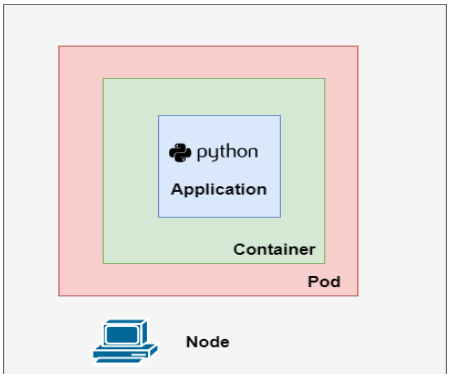

Figure 4 Architecture of a Pod object.

- **Controllers**: Kubernetes objects are monitored by processes and respond accordingly as a result of the controlling processes. A controller is able to track at least one Kubernetes resource type. These objects have a spec field that represents the desired state. The controller(s) for that resource are responsible for making the current state to advance to that desired state. Figure 5 illustrates the architecture for a Replication Controller. This controller extends to all nodes of the cluster. The Replication Controller's task is to ensure that the predefined number of Pods that have been declared are actively executing at any given time. Kubernetes contains several kinds of controllers such as replication controller, endpoints controller, namespace controller etc., but it also grants the possibility to implement custom-purpose controllers.

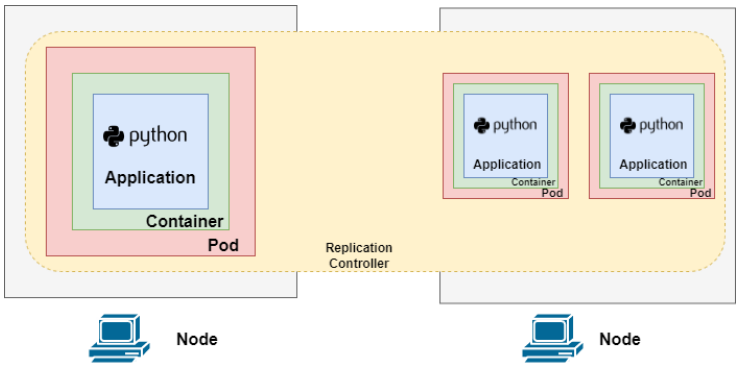

Figure 5 Replication type Controller configuration.

- **Deployments:** A Kubernetes Deployment instructs Kubernetes on the specific way of creation or modification of instances of the pods that hold a containerized application. The number of replica pods can be scaled, updates can be deployed, and roll-backs to previous versions through versioning-control, through the Deployment object. Figure 6 exhibits the configuration of a typical deployment of a Kubernetes cluster.

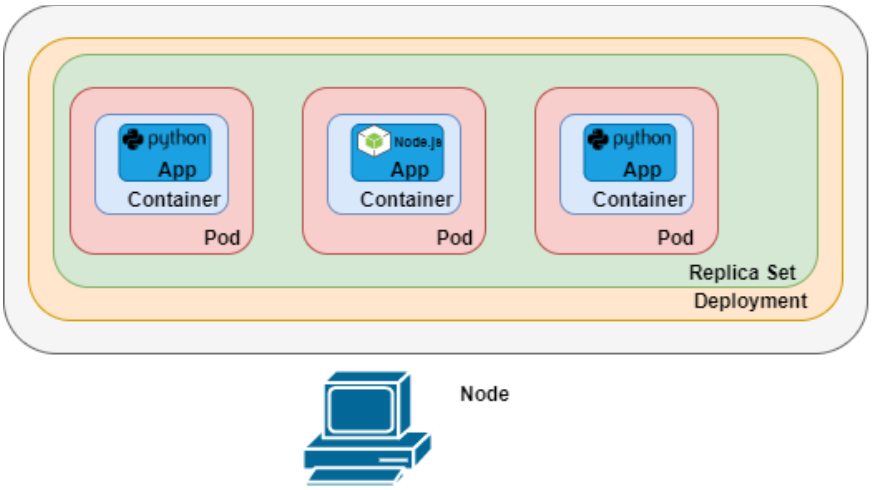

Figure 6 Typical deployment in a Kubernetes cluster.

- **Services:** Within the Kubernetes context, services constitute pure abstractions that define: (i) a collection of logical Pods, and (ii) respective policies for interaction with the Pods. With the Pods being impermanent by their definition, a service is responsible for enabling a collection of pods, that perform specific objectives, to be assigned with a ClusterIP (a name and a unique IP address). The IP address will remail the same as long as the service executes on an IP address. Policies for accessing the services are also delineated. The most prevalent services used in conjunction with Kubernetes are the NodePort Service, ClusterIP Service and the Load Balancer Service, and they are described in the following section.
  - **NodePort Service**: This service creates an internal Pod accessible on a Node port. In this way the Pod object is accessible from any webspace outside the Node. Figure 7 displays the NodePort service configuration.

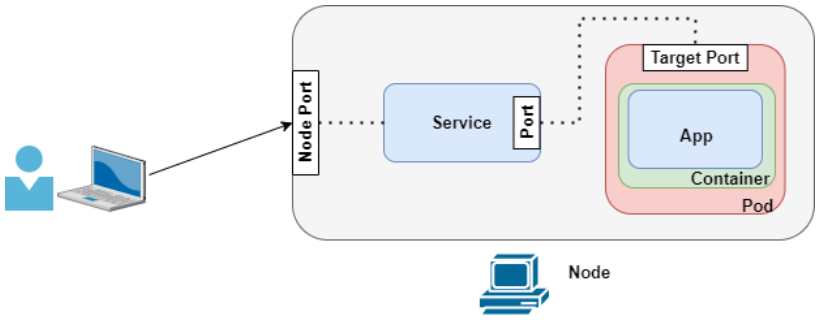

Figure 7 NodePort service configuration.

  - **ClusterIP Service**: Through this service, we have the ability to group Pods and provide a single interface so that we can access those Pods. Each service is given a name and IP within the cluster. The name of the service can be used by other Pods to access the service. This type of service is called ClusterIP. Figure 8 displays the ClusterIP service concept.

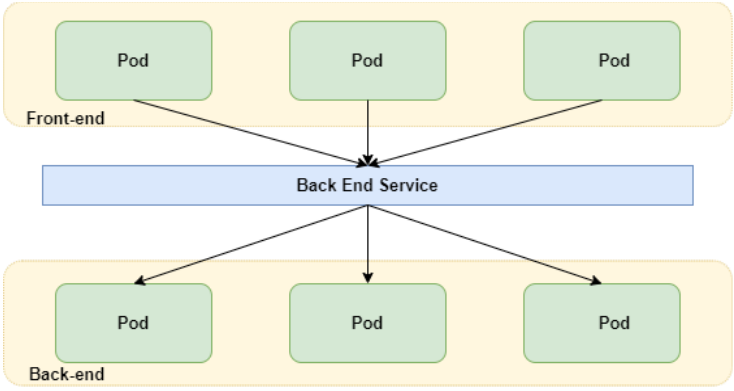

Figure 8 ClusterIP service concept.

- **Load Balancer Service**: This service type creates load balancers in various Cloud providers such as AWS, GCP, Azure, and are responsible to expose the deployed application to the Internet. Cloud providers procure for mechanisms that route the traffic to the services. Figure 9 displays the Load-Balancing service concept.

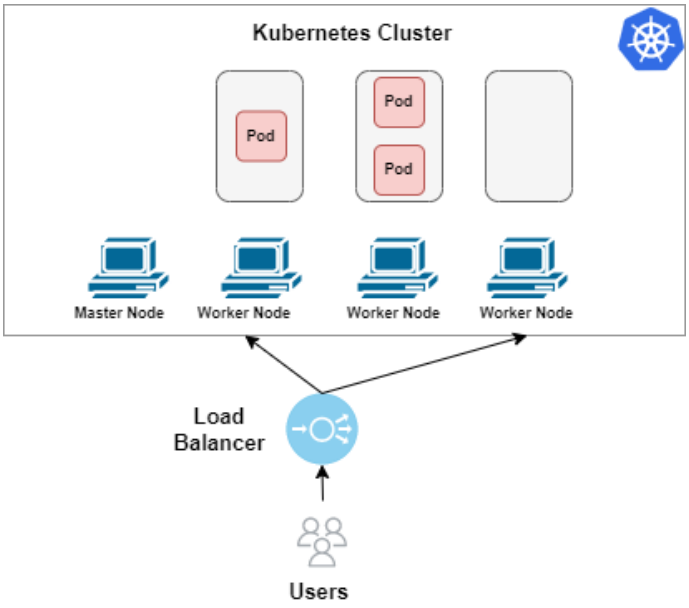

Figure 9 Load-Balancing service concept.

6.2 Apache Kafka

Another technology that RADAR-base employs is Apache Kafka [24]. Apache Kafka is an advanced message broker. This technology is responsible for storing data with high availability for processing of other applications, such as long-term storage or real-time analysis of data. Essentially, Apache Kafka is messaging system for gathering and processing of big data in real-time. As a result, Kafka renders itself a fundamental component for applications that execute in the scope of a Kubernetes cluster. The main benefits of Kafka are summarized below:

- **Kafka is highly scalable.** Kafka has the ability to scale on-the-fly without any overhead or interruptions of services.
- **Kafka is highly durable.** Kafka exhibits persistence in memory which allows intra-cluster replication. This makes for a highly durable messaging system.

- **Kafka is highly reliable.** Kafka supports multiple subscribers and has the ability to replicate data. Employs automatic load balancing with respect to consumer entities, when in case of failure. As a result, high reliability for the service and the users is exerted.
- **Kafka offers High Performance.** Kafka offers high throughput transactions in terms of publishing and subscribing. Essentially, constant levels of IO performance are kept, despite possible scaling of workloads and traffic.

A short description is provided of how Apache Kafka works. Apache Kafka is based on the publish-subscribe model and the following protocol is adopted to orchestrate events.

- Producers produce messages and publish them to topics.
- Kafka categorizes the messages into topics and stores them so that they are immutable.
- Consumers subscribe to a specific topic and absorb the messages provided by the producers.

As Producers and Consumers are illustrated applications that generate event-driven messages and applications that consume the messages produced. All these messages are stored on Kafka brokers and have the ability to be sorted by topics established by the user. One component of vital importance that executes in a tightly-coupled manner with Kafka is Zookeeper, as seen in Figure 10. In the case that Zookeeper is not actively running, the respective Kafka service loops into a restart state until a Zookeeper instance is encountered. Zookeeper's responsibility is to coordinate all the Kafka subcomponents, such as Producers, Consumers, Brokers, as well as cluster memberships.

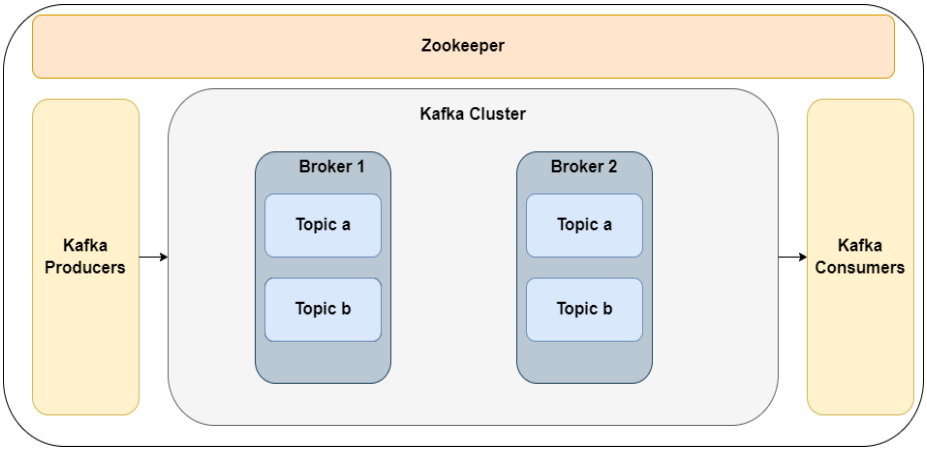

Figure 10 The Apache Kafka and Zookeeper component architecture.

6.3 Graylog

Graylog is an open-source log management tool [25]. Graylog is responsible for collection, indexing and analysis of data from any source regardless of structuring of the data. The majority of contemporary applications are implemented in the context of microservices, which are usually executed from different machines, making the collection of logs from these different machines a manual and labour-intensive task for the user. Graylog's installation is centralized and configured at one machine, while gathering all available logging data from the distributed environment of the microservices deployed. Graylog also employs a search engine (Elasticsearch) to store log data. Figure 11 depicts the Graylog architecture. Some of the advantages that Graylog provides are the following.

- Centralized log data.
- Improved system performance.

- Time-efficient monitoring.
- Automated issue troubleshooting.

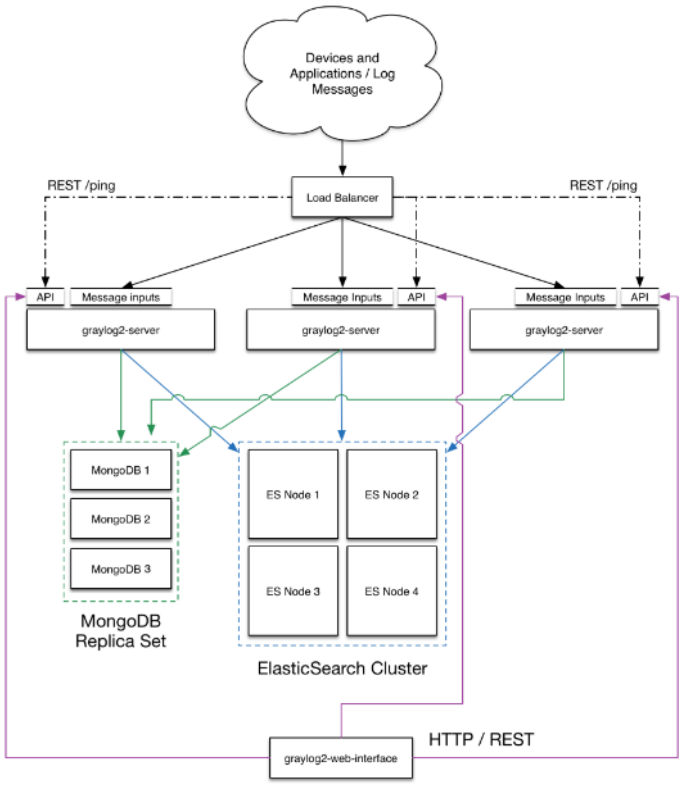

Figure 11 The Graylog architecture [25].

6.4 Prometheus - Grafana

Prometheus is a monitoring solution for recording and processing any purely numeric time-series [26]. This component is responsible for gathering, organization and storage of metrics with unique identifiers and timestamps. Prometheus is an open-source software that scrapes metrics from HTTP endpoints and collects them while identifying the target each time with an identifier and the respective timestamp. As a target can be supported applications, services such as a database management system or even infrastructure platforms such as a Kubernetes deployment. This kind of architecture further assists and expedites the discovery and scraping of defined targets that generate the data under collection. Prometheus also stores the scraped data, which can be later assessed and analysed through the Prometheus Query Language – PromQL.
Grafana is an open-source analytics and monitoring solution for any database [27]. Also, Grafana supports querying Prometheus and the Grafana data source for Prometheus is already included in the Prometheus package. As a result, the Grafana component integration facilitates and accelerates qualitative and quantitative visualization and rendering of scraped data. Figure 12 illustrates the components, services and overall architecture of Prometheus, as well as the Grafana component integration.

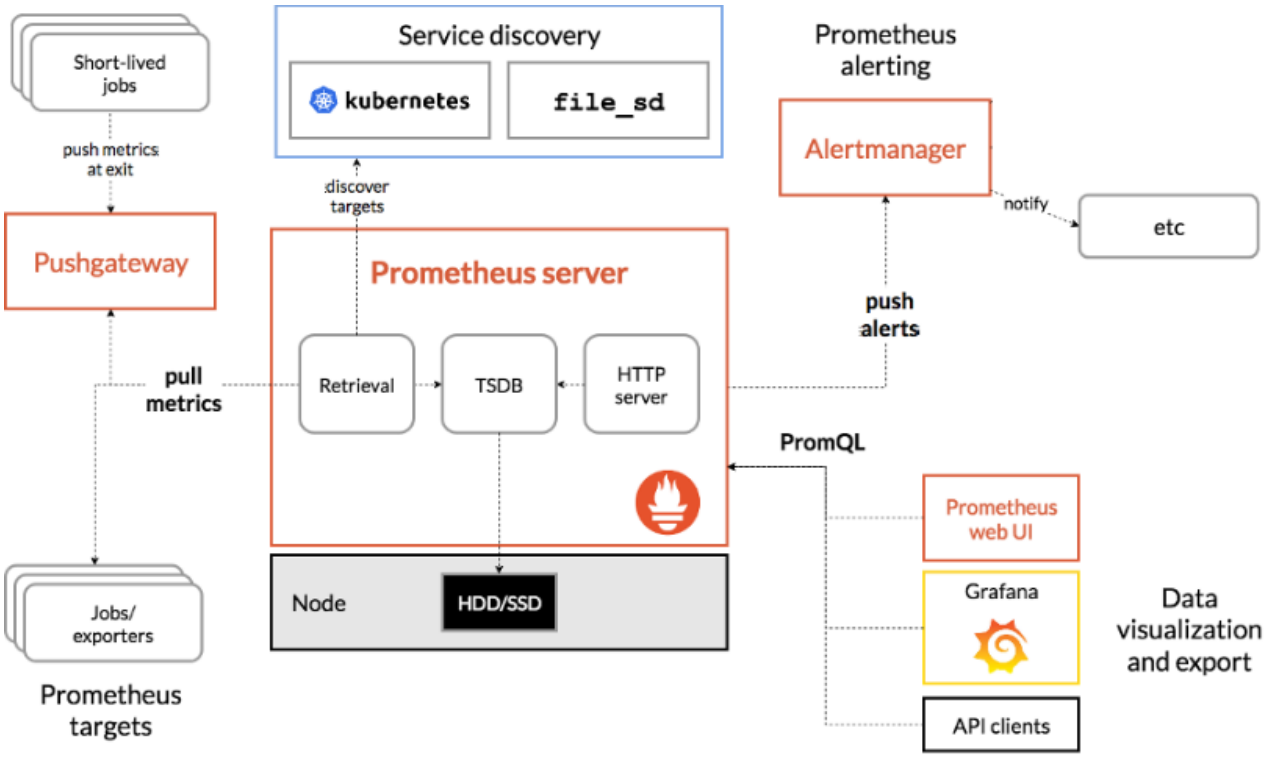

Figure 12 Prometheus and Grafana Architecture [28].

6.5 MinIO

MinIO is a high-performance distributed object storage server, designed for large-scale data infrastructure [29]. MinIO is native to Kubernetes and is a high-performance, S3 compatible object storage. This essentially means that MinIO is the only object storage suite available for usage on every cloud, either public or private, any Kubernetes distribution, as well as on the edge. MinIO is software-defined and is fully open-source under GNU AGPL v3 [2].

MinIO is capable of operating on any server with any kind of storage drives (Just a Bunch of Disks or Just a Bunch of Drives – JBOD, Just a Bunch of Flash – JBOF). Since clustered servers always contain computationally equal nodes in terms of resources (fully symmetrical architecture), there do not exist any name or metadata servers.

MinIO stores data alongside metadata in the form of objects, disposing of the need for any further databases to be created and managed, e.g., metadata databases. Additionally, MinIO's functions are performed in an inline manner with explicitly consistent operations, such as erasure code, bit rot check, encryption, leading to highly resilient results.

A MinIO cluster consists of a group of single-process, distributed MinIO servers, per each node. MinIO is an individual process executed in the user space, employing ultra-lightweight functions as tasks with the ultimate purpose being high concurrency and thus high performance.

A deterministic hashing algorithm is employed to place objects into drives, which are *a priori* grouped into sets. Overall, MinIO provides scalable and secure object storage, since by design it has been granted with the ability to handle large-scale, multi-data center cloud storage services. Every MinIO cluster is fully isolated by other clusters that might be executing at the same space, ensuring no disruption with respect to updates, upgrades or security incidents.

Figure 13 illustrates the basic architecture of a distributed MinIO in N distributed clusters.

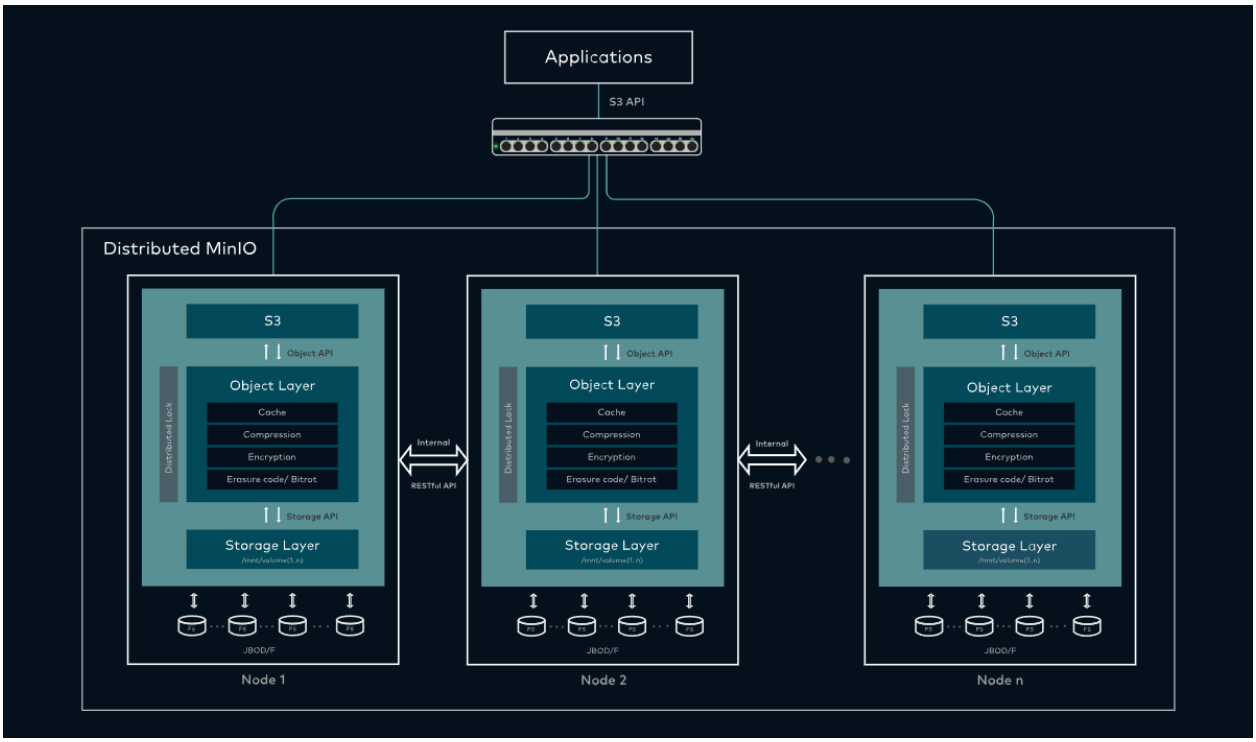

Figure 13 Distributed MinIO architecture on multiple service nodes [16].

6.6 RADAR-base component overview and component breakdown

This section presents and describes thoroughly the components of RADAR-base. Once the installation of RADAR-base has been successfully completed, the Kubernetes Cluster objects that are running for the RADAR-base operation are visible on the Services Logs. Since RADAR-base provides the option to enable and disable Deployments and Services depending on the needs of each project, the RADAR-base components are divided into the following categories.

- Essential Components
- Output Components
- Support Components
- Fitbit Components
- Grafana Components
- Other Components

6.6.1 Essential Components

These components constitute core components of RADAR-base and should always be active. The essential components contain the following elements, as depicted in Table 6.

Table 6 The essential component services of RADAR-base.

Name	Description
Cert Manager	Creates SSL/TLS certificates to access the web applications. Uses Letsencrypt certificates by default.
Catalog Server	Creates RADAR-base topics in Kafka, registers schemas in Schema Registry and keeps a catalogue of available source types.
Kafka	Advance message broker for long – termdata storage
Management Portal	Portal to manage projects and participants throughout RADAR-base
Nginx Ingress	Web server used for accessing web applications
PostgreSQL	Database used to store Management Portal Data
Radar Output	Reads data from an intermediate object storage and converts it to a RADAR-base oriented output directory, by organizing it by project, user and collection date.
Radar S3 connector	Reads raw participants data from Kafka and puts them into an intermediate object storage like AWS S3 or MinIO.
Redis	Used by Radar Output to coordinate data processing in a parallel way
Radar Gateway	REST Gateway to Kafka, for incoming participant data. It performs authentication, authorization, content validation and decompression.
SMTP	Email server for sending registration email to researchers and participants.
Kafka Schema Registry	Registry of Avro Schema's used in a RADAR-base installation. Validation layer on top of Kafka to structurally verify incoming data.
Zookeeper	Use for Kafka cluster management and coordination.

6.6.2 Storage and Output Components

With Essential components it is only possible to send the intermediate and output data to an already deployed object storage like AWS S3, these components expand this to allow more options, as seen in Table 7.

Table 7 Output components of RADAR-base.

Name	Description
MinIO	Storage system that behaves like an AWS S3 object storage and allows the data to be hosted on private cloud and local machines.
S3-Proxy	Allows using Google Cloud, Microsoft Azure and a few other storage providers to store RADAR-base data by providing an AWS S3 compatible API.

6.6.3 Support Components

These components enable logging and monitoring of application to make sure they are always accessible and healthy. Installation is optional, but advised and these services are actively utilised by our deployment. These components are exhibited in Table 8.

Table 8 Support Components of RADAR-base.

Name	Description
Graylog	Log management system that allows receiving and viewing application logs with search and long-term storage capabilities.
Mongodb	Database system used by Graylog to store its configuration.
Elasticsearch	Search engine used by Graylog to store log data.
Fluent Bit	Data collection software that sends applications logs to Graylog.
Kafka Manager	For monitoring and managing Kafka cluster via a web dashboard.
Kube Prometheus Stack	Monitoring bundle that includes Prometheus for checking health of the applications, Alertmanager to send alerts in case an application is failing and Grafana as a dashboard to check health of the cluster via graphs.
Velero	Used for backing up the cluster and its data to an external object storage.

6.6.4 Fitbit Components

These components enable data collection from Fitbit devices. These are core components for our data collection from the wearable devices and can be seen in Table 9.

Table 9 Fitbit Components of RADAR-base.

Name	Description
Radar Fitbit Connector	Collects data from participants via the Fitbit Web API.
Radar REST Source Authorizer	Portal to authorize the Fitbit connector to read data from Fitbit accounts.
Radar REST Sources Backend	Backend for the RADAR Rest sources authorizer

6.6.5 Grafana Components

These components enable a dashboard mainly to visualise data collection compliance. All these services are also used, as seen in Table 10.

Table 10 Visualization Components of RADAR-base.

Name	Description
RADAR JDBC Connector	Stores data from selected Kafka topics into a TimescaleDB database.
RADAR Grafana	Shows a dashboard of collected data, usually by showing a heatmap.
TimescaleDB	Database to store time-series data from Kafka topics. TimescaleDB is an extension of PostgreSQL.

6.6.6 Other Components

These components are used situationally to enable more integration or customization features. They are employed in our architecture and are depicted in Table 11.

Table 11 Miscellaneous Components of RADAR-base.

Name	Description
RADAR Integration	Used for integrating REDCap surveys with RADAR-base.
cc-schema-registry-proxy	Used to redirect incoming requests related to Schema registry to Confluent cloud Kafka cluster.
App Config	Mobile app configuration engine with per-project and per-user configuration.
App Config Frontend	Frontend web UI for App Config
RADAR Backend Monitor	Provides a layer to monitor and analyse streams of wearable data and write data to storage.
RADAR Backend Stream	Provides a layer to monitor and analyse streams of wearable data and write data to storage.

6.7 RADAR-base Concepts

Before giving a detailed description of the RADAR-base workflow, it would be useful to describe some of the basic concepts of Radar Base.

ManagementPortal (MP) is a core component to manage subjects of remote monitoring studies. It has been built by the RADAR-base community and aids the default security protocol followed in RADAR-base data workflow. It facilitates the management of studies, subjects, and data-sources. Furthermore, the Management Portal handles authentication and authorization of entities through role-based access control. The authentication and authorization process employs the OAuth2 protocol workflow. The main page of MP is provided in Figure 14.

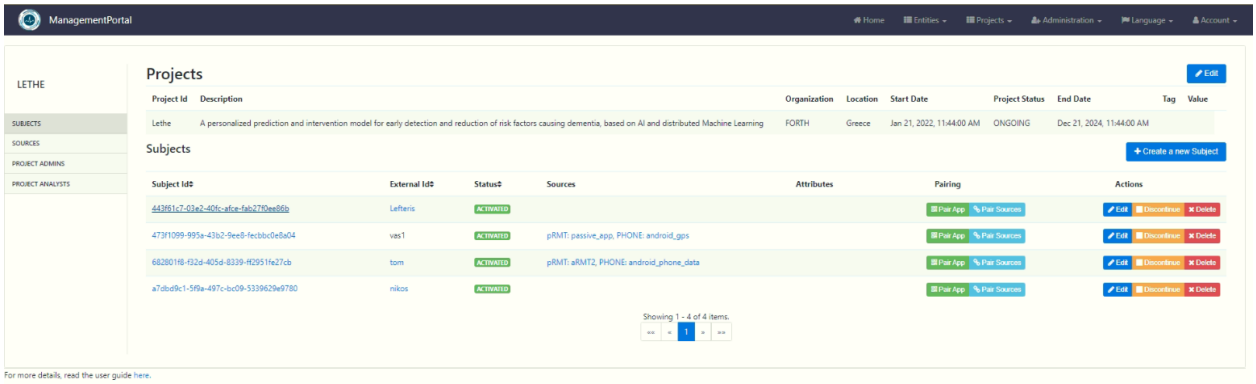

Figure 14 The Management Portal Component.

Concepts within ManagementPortal

Management Portal has been designed to support remote monitoring of subjects. RADAR-base follows a few terms and concepts across the platform which are further defined. Figure 15 displays the login page of MP.

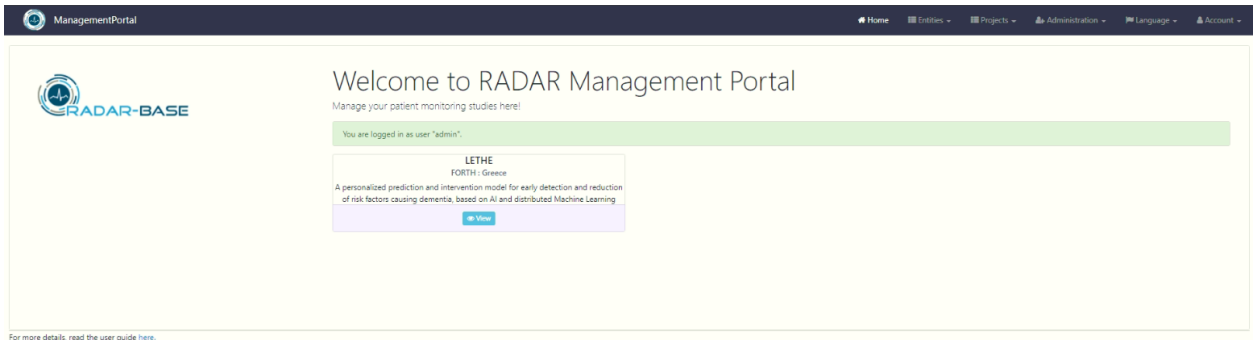

Figure 15 RADAR-base Management Portal login page.

Project: A project is set up regarding a study or an epoch of a study. A project setup is imperative in order for the data collection to happen. All users are classified based on the project, besides the master entity, SYSTEM_ADMIN. Users can co-exist within different projects. One restriction taking place is that a project name should be unique, and no projects can exist without a name. Projects are created by the superuser SYSTEM_ADMIN. Figure 16 displays the project creation page.

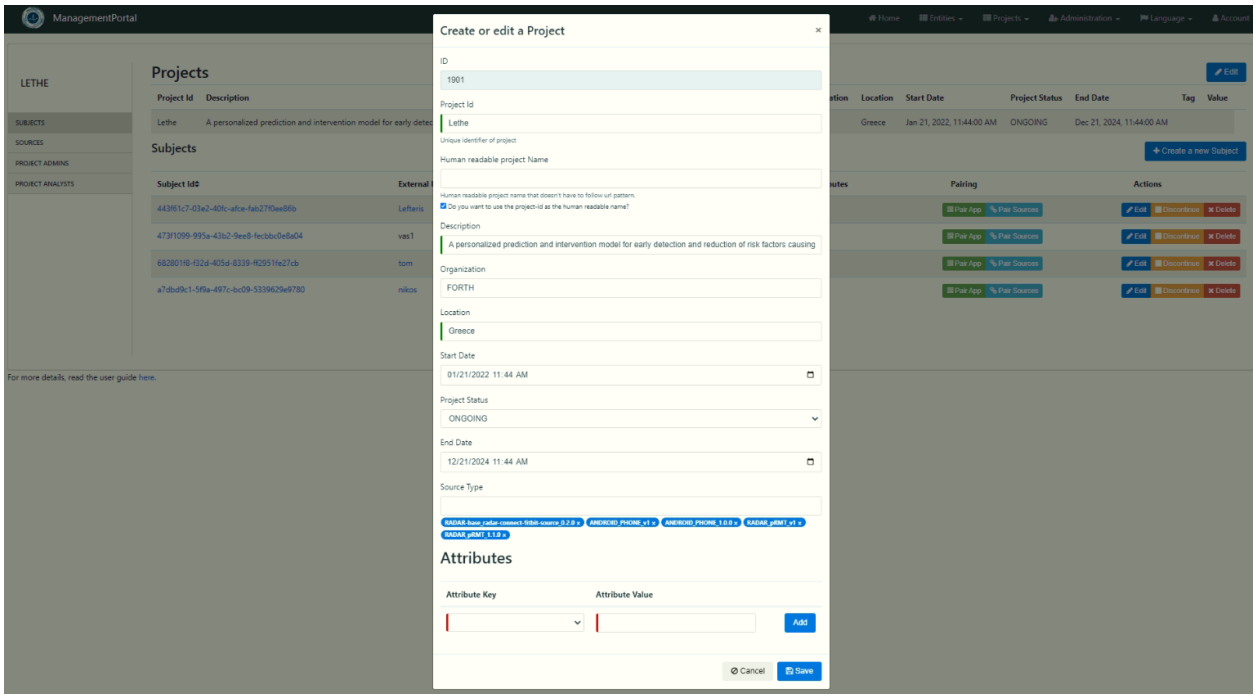

Figure 16 Project Creation in Management Portal.

Subject: A subject essentially refers to a person/object of interest in terms of data collection. Subjects are appointed to a project and are described by a unique identifier, which is created by Management Portal exclusively for each subject. These initial identifiers are not human-readable (Universal Unique Identifier format – UUID), but there has been procured additional metadata storage for providing more human-friendly identifiers to the subjects, if needed. Subjects re

While a study participant is the most obvious real-world counterpart of the subject, it is not limited to that use case. Subject can refer to anything in the environment you want to assign a set of sensors to e.g., subjects of a study. Figure 17 exhibits the subject creation page.

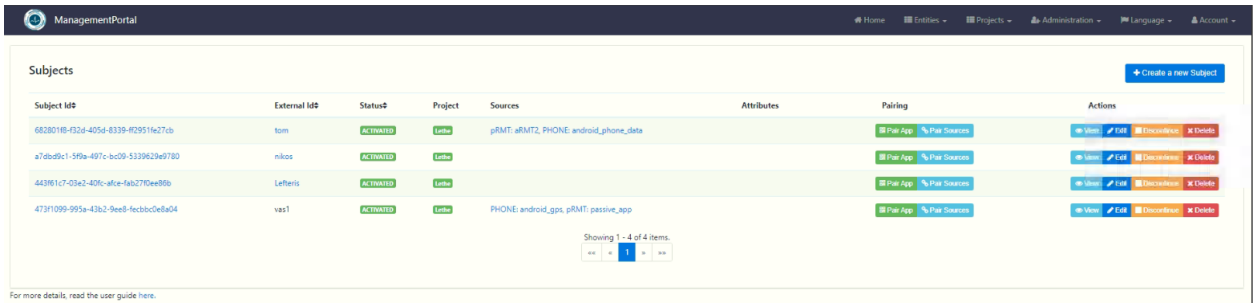

Figure 17 Subject creation in Management Portal.

Source Type: The source type constitutes the type or the category of a source, e.g., Fitbit Charge, Fitbit Sense, questionnaires or manually input variables such as glucometer, body mass etc. The source type is perceived by the uniquely combining the producer, model and catalog version. The catalog concept outlines the way that data is collected as well as the way it should be versioned from the source, thus providing maximum flexibility for custom-made source types. For a project created in Management Portal, various source types can be defined and will be intra-related with respect to data generation and fetching, since source types contain collections of source data. Figure 18 displays the Source Type selection page.

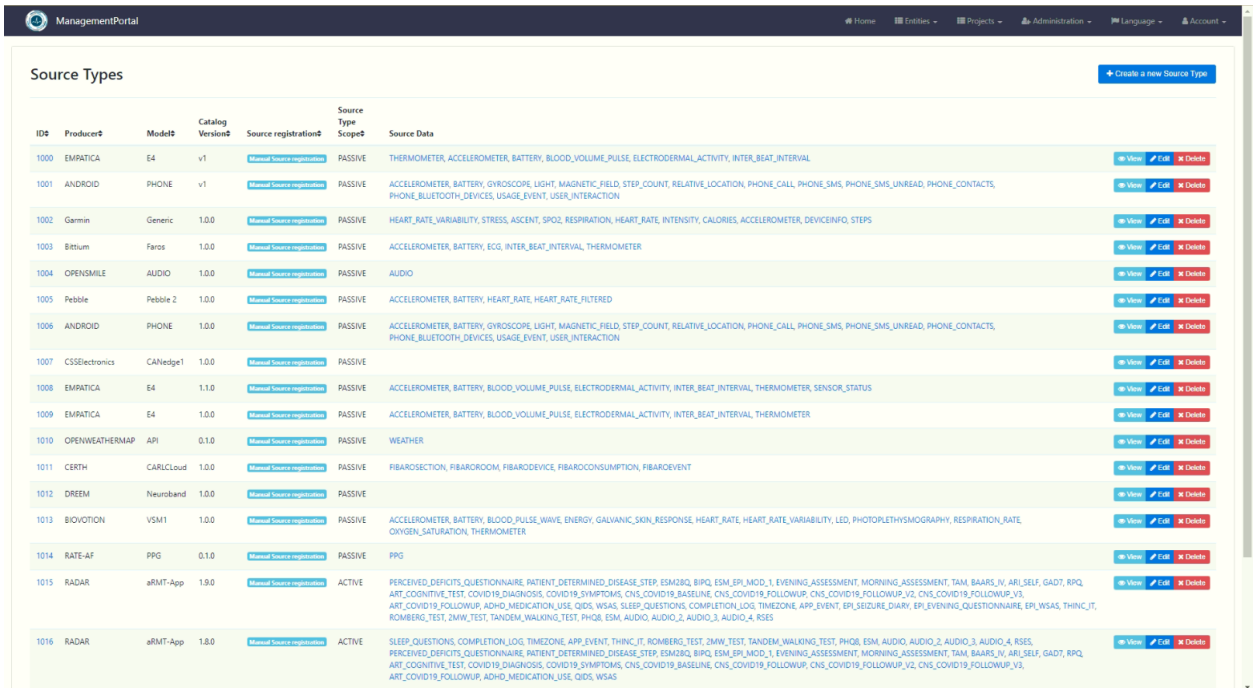

Figure 18 Source Type selection in Management Portal.

Source Data: The Source Data defines the under-collection data from a source type. Source data can be used in a source type and a source type may contain multiple source data. The source data contain metadata with respect to the source of collection, i.e., type and format of the data, schemas, topics that receive data, data units, polling rates for data fetching from Fitbit servers. Figure 19 displays the Source Data selection page.

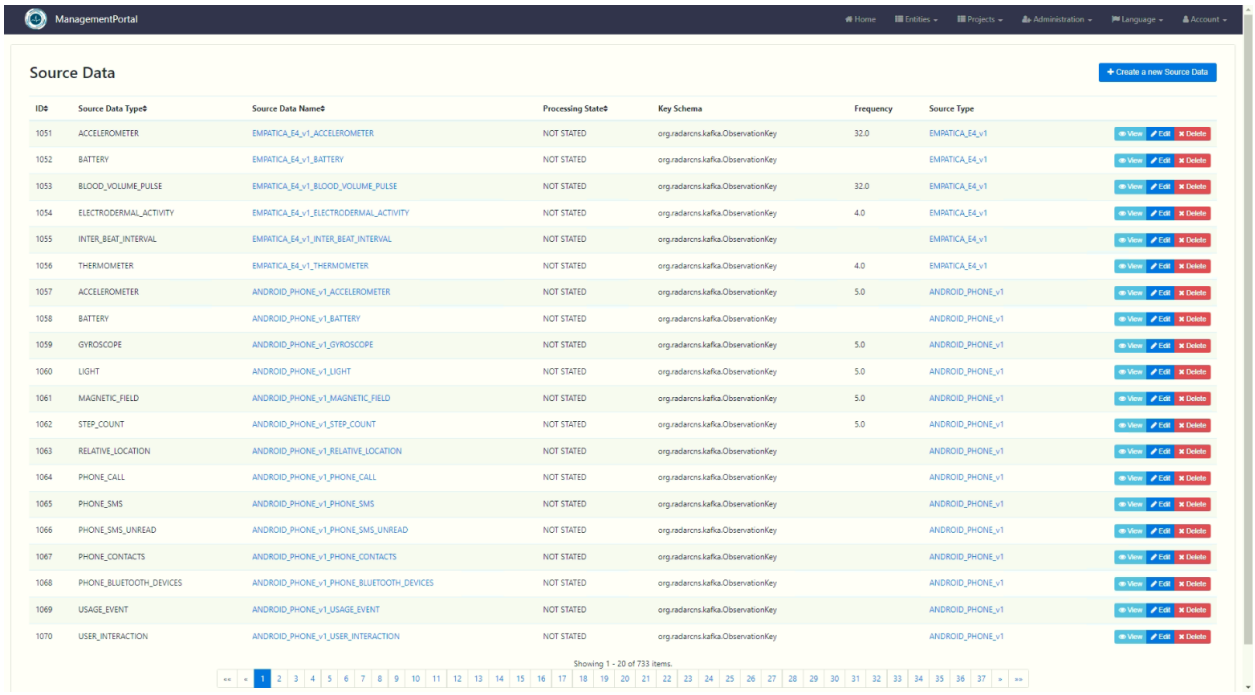

Figure 19 Source Data selection in Management Portal.

Source: A source constitutes the producer of the data, that is the source that generates the data, i.e., a device as Fitbit Sense or a mobile phone. A source is assigned to a project that is uniquely identifiable through the unique identifier provided by the MP. The source represents an instance of a Source Type, i.e., Fitbit Sense or Charge or Mobile Phone. A source can be created dynamically through the source type within MP. The security measures, currently taken by RADAR-base validations, do not allow subjects outside Management Portal to send data from sources assigned to them. Figure 20 displays the Source selection page.

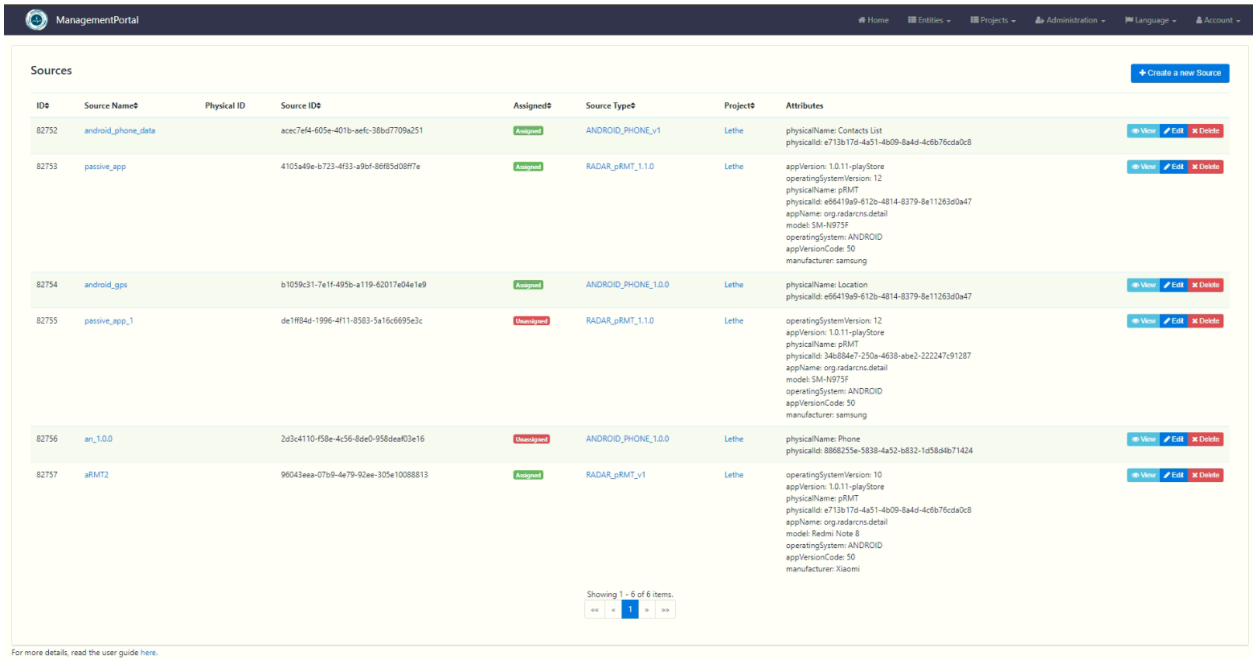

Figure 20 Source Selection in Management Portal.

Security and Role Based Access Control.

The MP utilizes the paradigm of the Role Based Access Control (RBAC) as well as the OAuth2 protocol for secure identification. Every user that exists in the MP can be assigned with one or more roles.

The concept of roles in MP blends together two different notions; the project and the authority. Through authority, privileges can be assigned and a user has his abilities fully defined with respect to his course of action. Some of the most characteristic examples could be the following: a SYSTEM_ADMIN is able to create a new project, a PROJECT_ADMIN can add subjects or sources, source_types and so on. A role can subsume capabilities of other roles, ultimately ending up to the subjects that have no access to the MP.
Concurrently, the concept of authority contains permissions arranged in sets. A permission is defined as the minimal action executed on the RADAR-base platform. An authority is essentially a collection of permissions with respect to the requirements. This tightly coupled configuration between roles and authority distinguishes the ability of roles to execute sets of actions based on permissions explicitly assigned.

6.8 RADAR-base & RADAR-Kubernetes

Since the basic technologies that are utilized by RADAR-base have been described, it is time to delve into the functionality and components of the RADAR-base platform. As mentioned before, RADAR-base is an open-source platform for remote research studies and virtual clinical trials. RADAR-base is a modular solution for active and passive data collection, real-time streaming, harmonized data schemas, distributed storage and data exploration capabilities. RADAR-Kubernetes is the most novel architectural approach of RADAR-base and will be the long term supported form of deploying the platform. Even though, RADAR-Kubernetes is being used in few production environments, it is still in its early stage of development.

6.9 RADAR-Kubernetes Architecture

Figure 21 represents the architecture and components that Radar Base uses. In the diagram provided, the main components and services of RADAR-base Kubernetes Architecture have been included.
From the illustration in Figure 21, it is apparent that RADAR-base combines many technologies to provide an end-to-end solution for remote monitoring use cases. The components used will be described, as well as the workflow from data collection, storage and finally the retrieval and sharing of the data that is being collected.

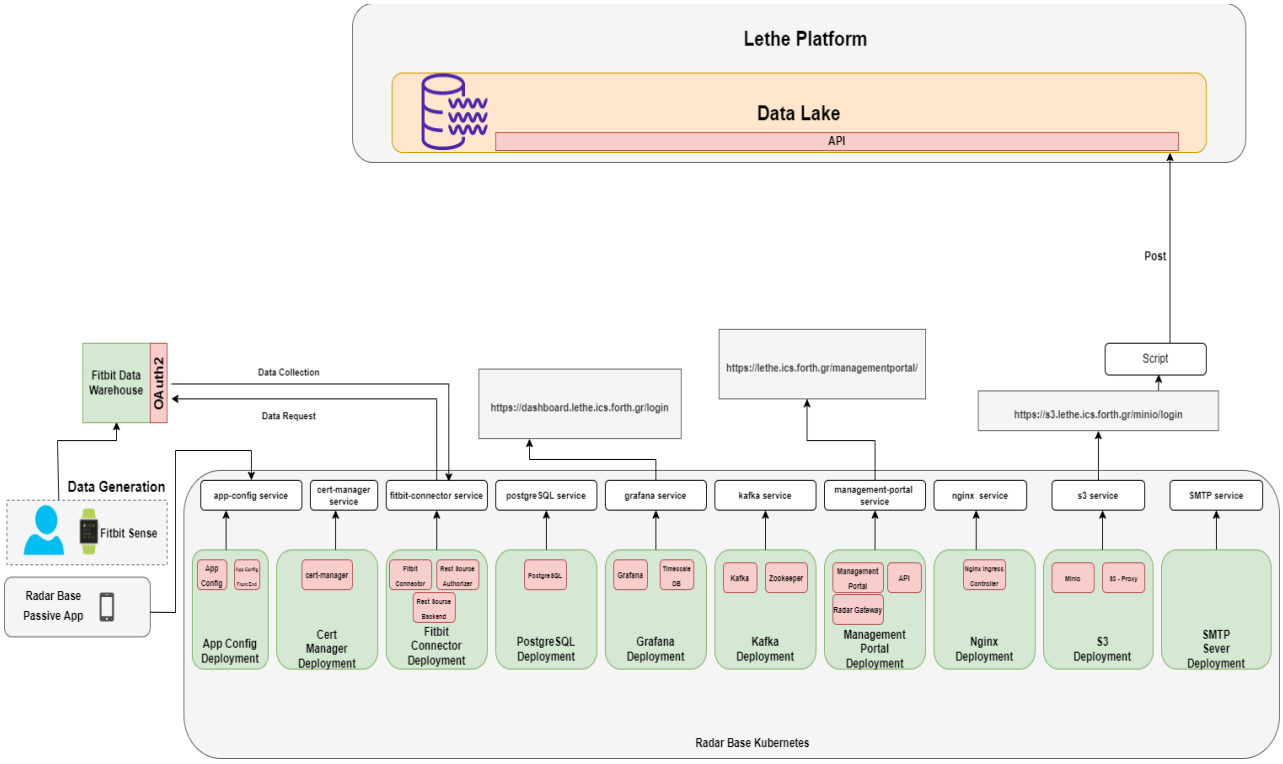

Figure 21 The architecture of the RADAR-base Kubernetes middleware platform for the LETHE project.

6.10 RADAR-base Kubernetes Workflow

This section presents the workflow of RADAR-base for data collection from the Fitbit Charge/Sense smartwatch as well as from the passive Remote App provided by RADAR-base. The smartwatches and the smartphones are the main devices that generate the subjects' digital biomarkers that are received during the clinical trial, as depicted in Section 7 of the current deliverable. Additionally, mobile sensor data such as relative location, nearby Bluetooth devices, and others will be harvested, and as a result the Passive Remote Monitoring Technology App (pRMT App) will be employed as well. All the digital biomarkers and mobile sensor data that will be harvested are thoroughly described in Section 7.

Creating a Project

Initially, a project should be created through the MP. Each project can be assigned subjects, administrators and analysts. Additionally, dates of activity, managerial organizations and relational metadata regarding the project can be inserted for thorough description of the project. Figure 22 displays the Project Creation page.

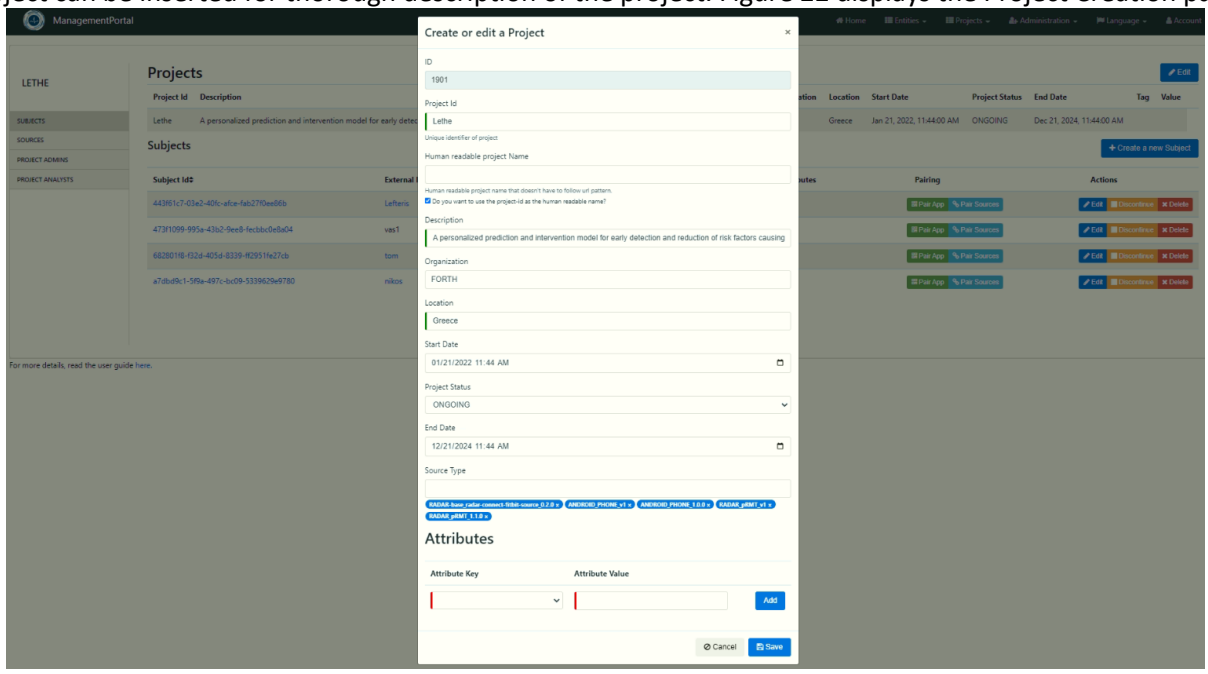

Figure 22 LETHE Project Creation.

Creating Subjects

The next step is to create subjects and assign them to the project. A subject should be assigned to a project and should be uniquely identifiable. Being a prerequisite for the subject's existence, the Management Portal automatically creates a unique identifier for every subject. Figure 23 displays the Subject creation page.

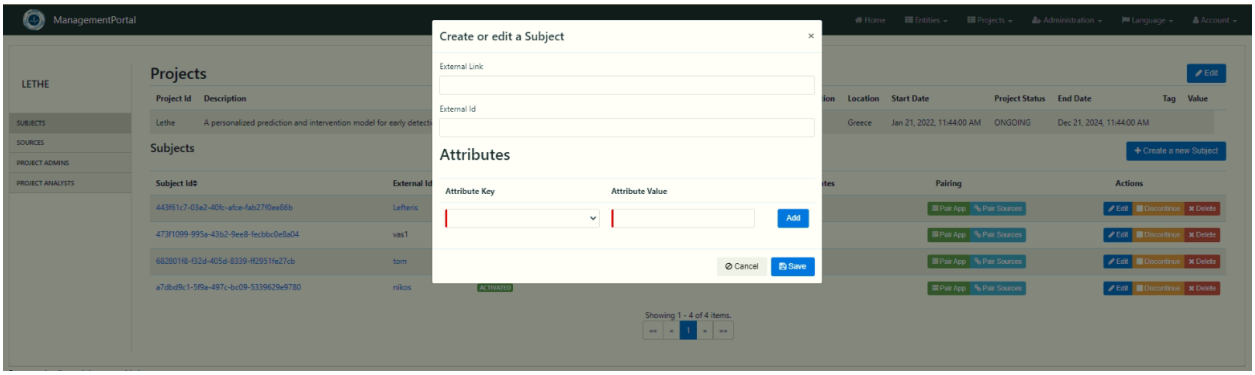

Figure 23 LETHE Subject Creation.

Collecting Data from Fitbit Sense Smartwatch

For each subject, a Fitbit App is registered via https://dev.Fitbit.com/apps and subsequently the subject is authorized via the rest-source authorizer service provided by RADAR-base. The following screen, Figure 24, will come up upon visiting https://LETHE.ics.forth.gr/rest-sources/authorizer/login (initial development environment) and logging in with the credentials that were given from Fitbit.

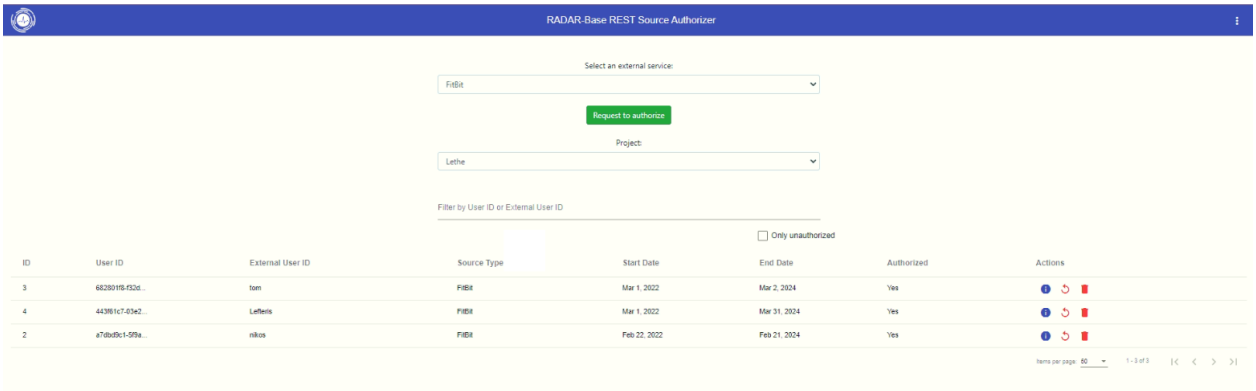

Figure 24 Subject authorization through the REST Source Authorizer.

Once the user has been authorized, through the python-Fitbit-connector service, data from the Fitbit Data Warehouse are being received. The data is stored in PostgreSQL and is accessible to the other services (Grafana dashboards, MinIO, etc.) via the Kafka cluster.

Visualization Dashboards for Data

The data collected from every subject in accessible and can be visualized ad hoc, through the respective dashboards created by the Grafana component. Currently, the LETHE dashboard is hosted under https://dashboard.LETHE.ics.forth.gr/ (initial development environment) and is accessible upon provision of credentials obtained from RADAR-base. The dashboard makes the graphical representations of the subjects' data at our disposal. Figure 25 is an example of heart rate data measurements received, for a user, from Fitbit Data Warehouse, in the desired date and time range.

Figure 25 Data visualization dashboard with collected heart rate data.

Application (passive RMT app) Configuration

Radar Base provides a service with which we can configure the passive RMT app, as to which data we want to receive from the mobile sensors. It allows us to configure the data we receive at the Project level and at

the user level. For example, we may want for a particular subject to receive only relative location, while for someone else we may want to receive accelerometer or gyroscope data as well. If we visit https://LETHE.ics.forth.gr/appconfig/login (initial development environment) and log in with the credentials we have, we can configure the Radar Base passive RMT app. Figure 26 andFigure 27 present the app config service at the project level and at the subject level respectively.

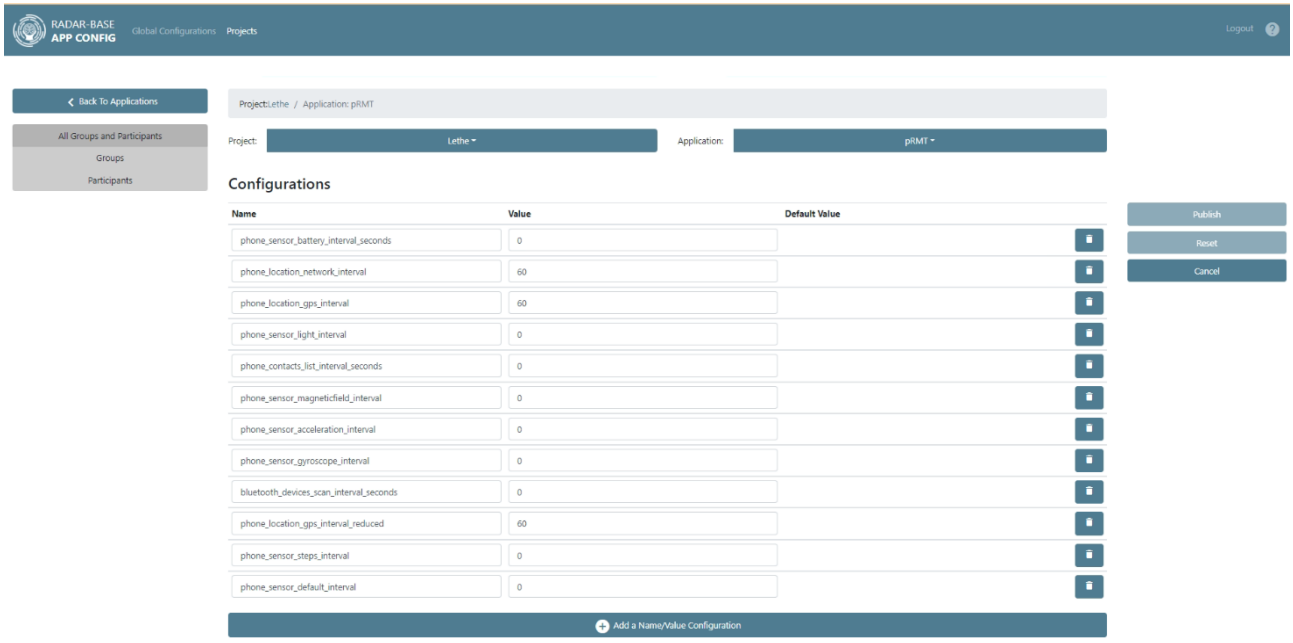

Figure 26 Parameterization of the mobile device sensors at the project-level.

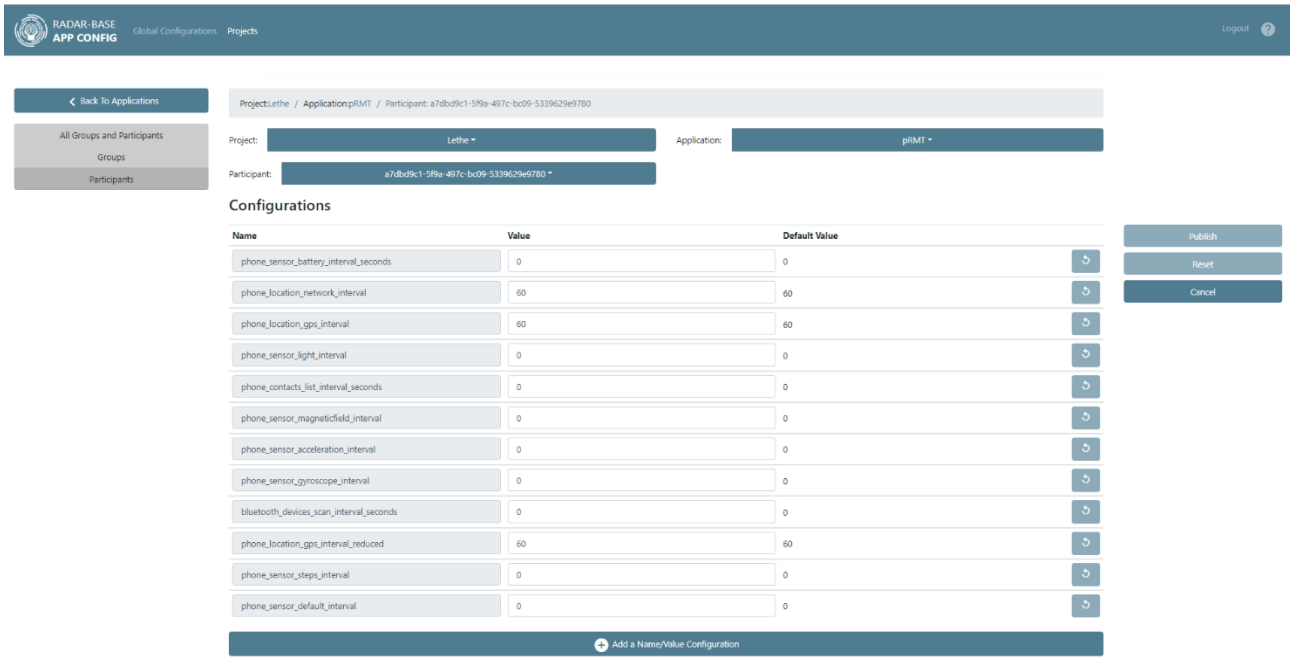

Figure 27 Parameterization of the mobile device sensors at the subject-level.

Collecting Data from RADAR-base pRMT application

For every subject the RADAR-base pRMT app will be downloaded, installed, and paired with each respective Fitbit account from the Android play store. Consecutively, the app needs to be paired with the corresponding user. Since the user has been assigned to a project, the pairing app screen will come up with a Quick Response code – QR code which needs to be scanned through the RADAR-base pRMT app.

Upon completion of this task, the subject has successfully paired his pRMT app with the RADAR-base project at the subject-level. At this stage, the pRMT app continues executing in the background and sends the data with the platform, while the subject remains oblivious of its existence. Figure 28 displays an example of a generated QR code. Figure 29 shows the user end of the configured application. Besides stopping the services nothing else can be configured from the user and the application continues in background execution as a service.

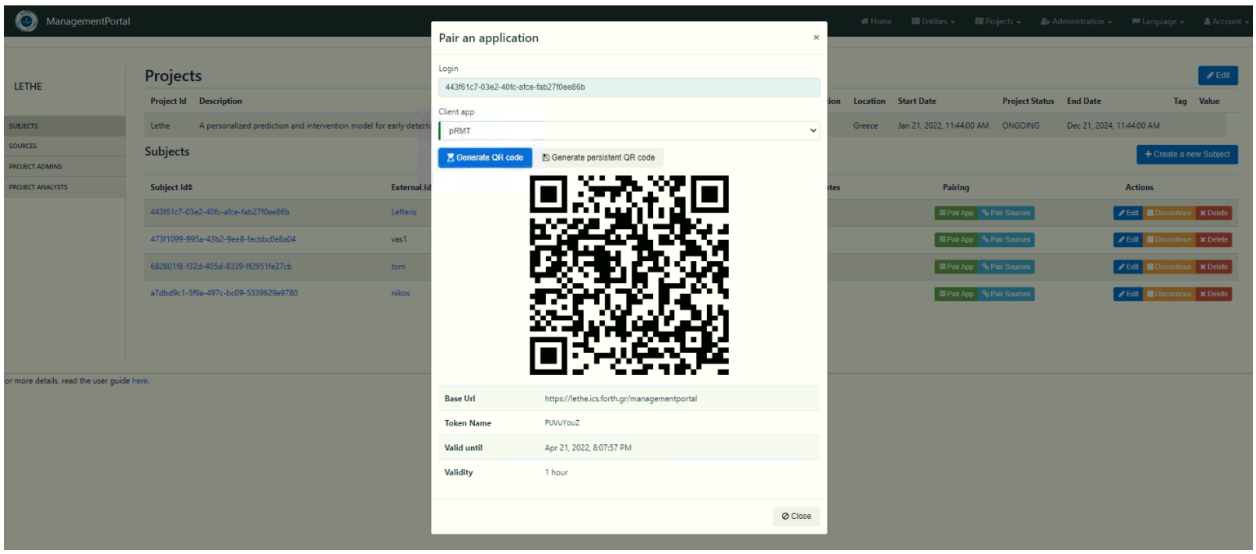

Figure 28 QR code generated for pairing with the subjects' devices.

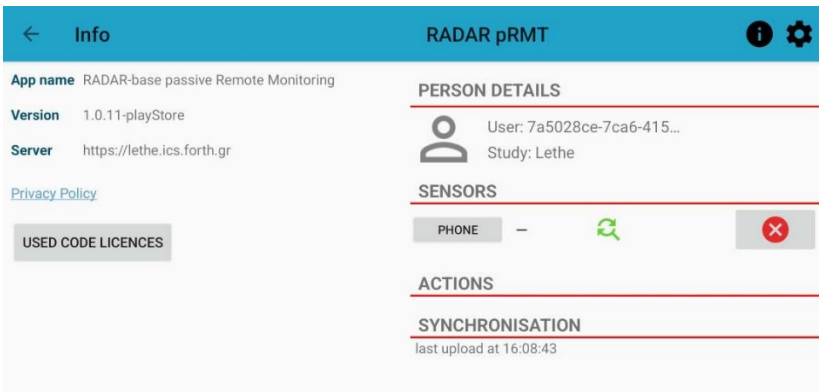

Figure 29 pRMT application on the client side.

Storage services

From the service provided by RADAR-base for MinIO, we can see all the data we have stored in our database and we can download individual data. Below are the data we have collected from a user via Fitbit sense. Figure 30 displays the MinIO storage service page, for the topic collection for a particular subject.

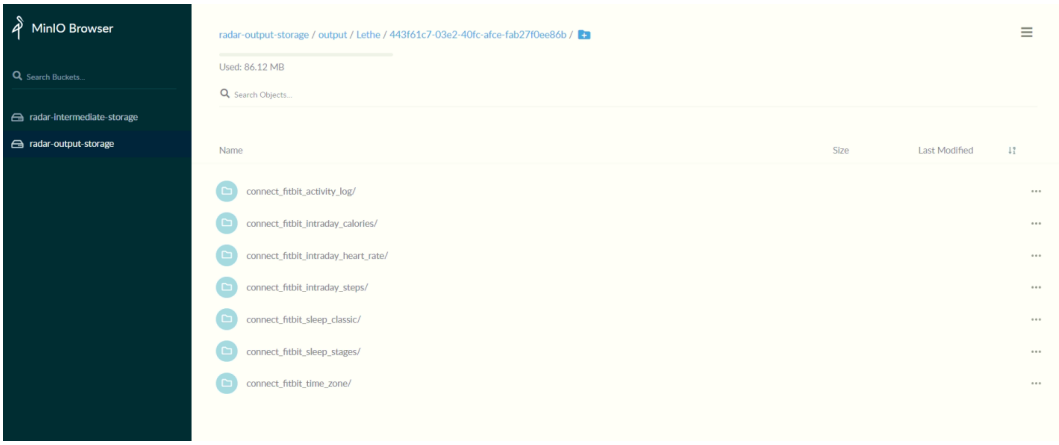

Figure 30 MinIO storage service.

Summarizing the above steps, Figure 31 presents the workflow of the RADAR-base platform.

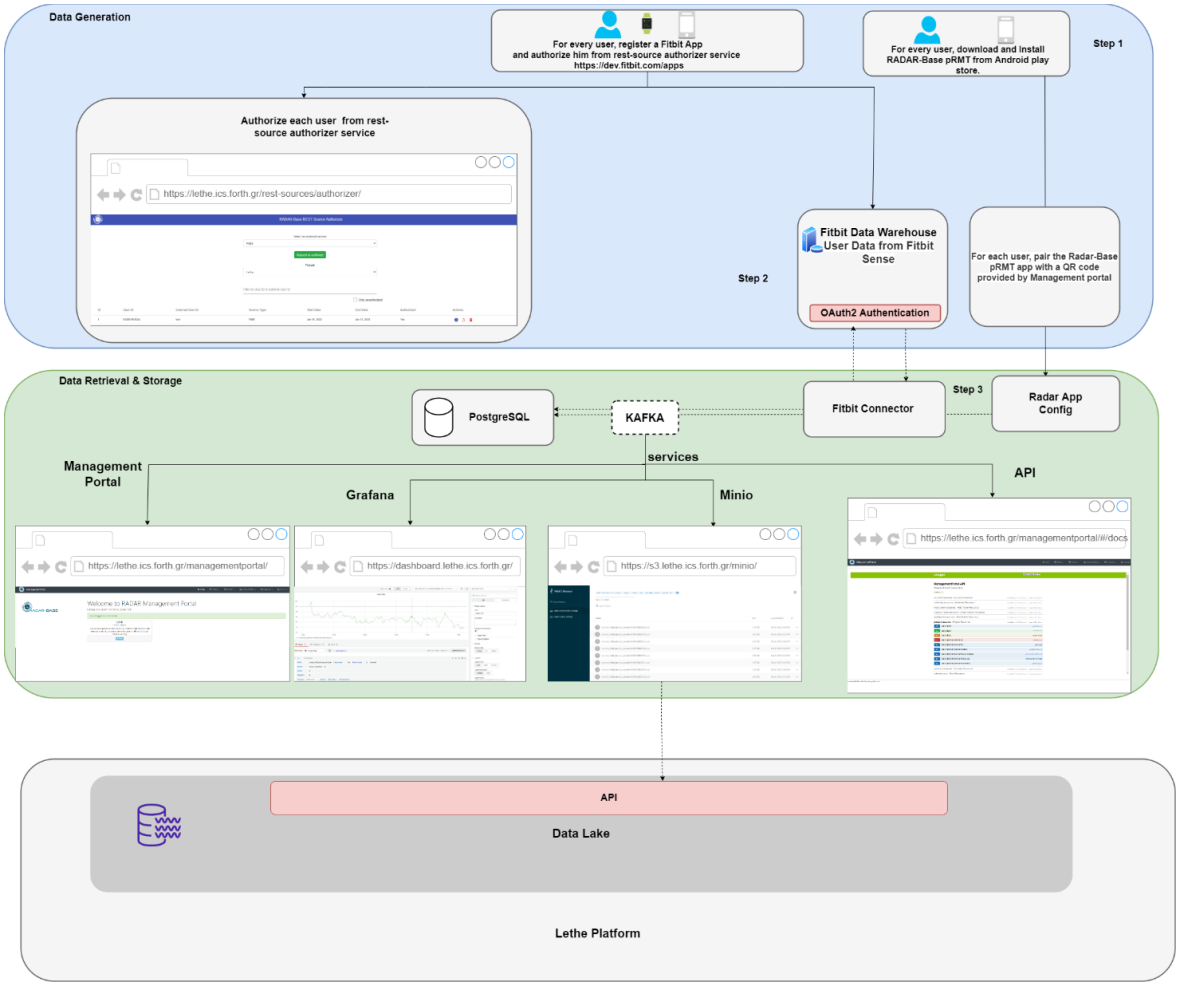

Figure 31 Workflow of the RADAR-base platform.

6.11 Sequence Diagrams

This section provides the Sequence Diagrams for the prominent functions of the RADAR-base platform. Specifically, the ones provided refer to: (i) Subject Registration, as seen in Figure 32; (ii) Pairing a subject with Fitbit and requesting for data from the Fitbit Data Warehouse, as seen in Figure 33; (iii) Posting data to the Data Lake following the prior data acquisition, as seen in Figure 34.

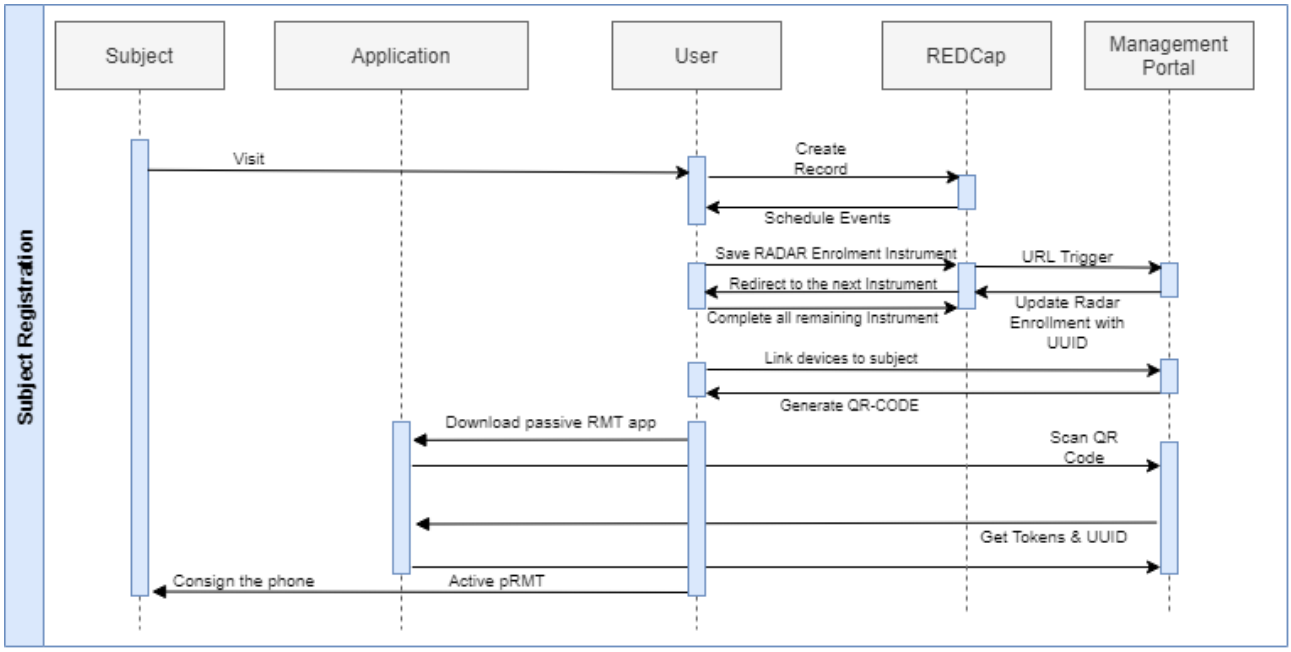

Figure 32 Subject registration sequence diagram.

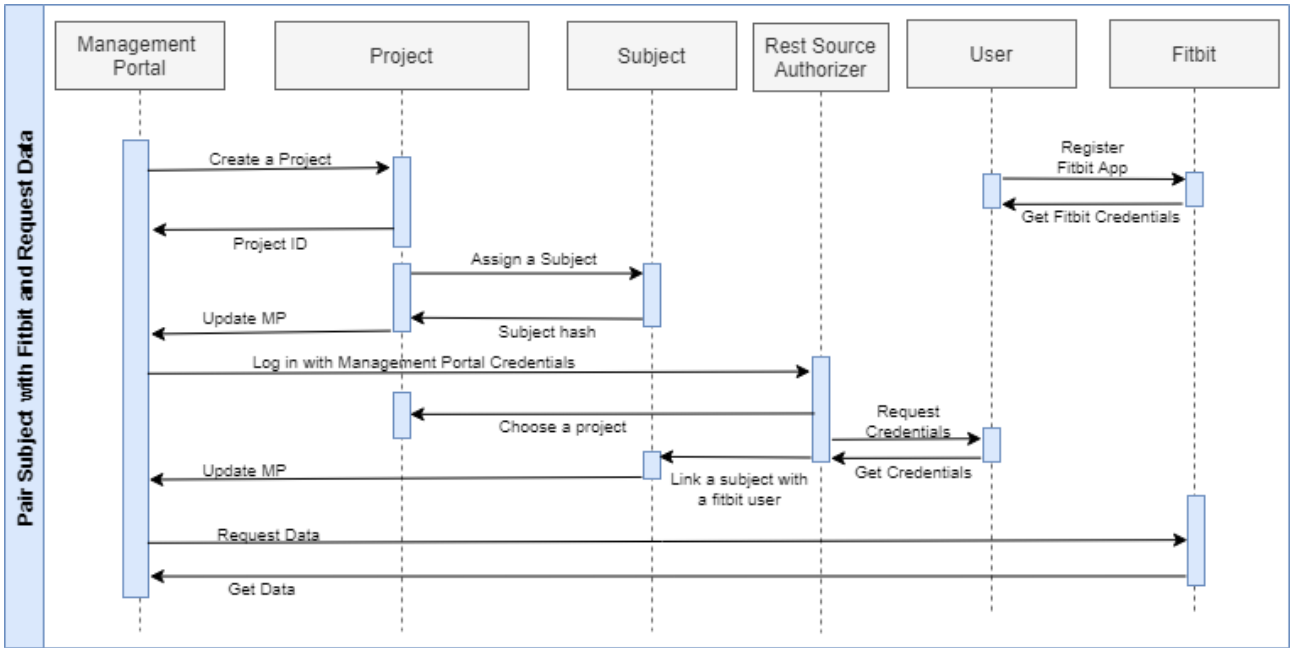

Figure 33 Pair Subject with Fitbit and Data request sequence diagram.

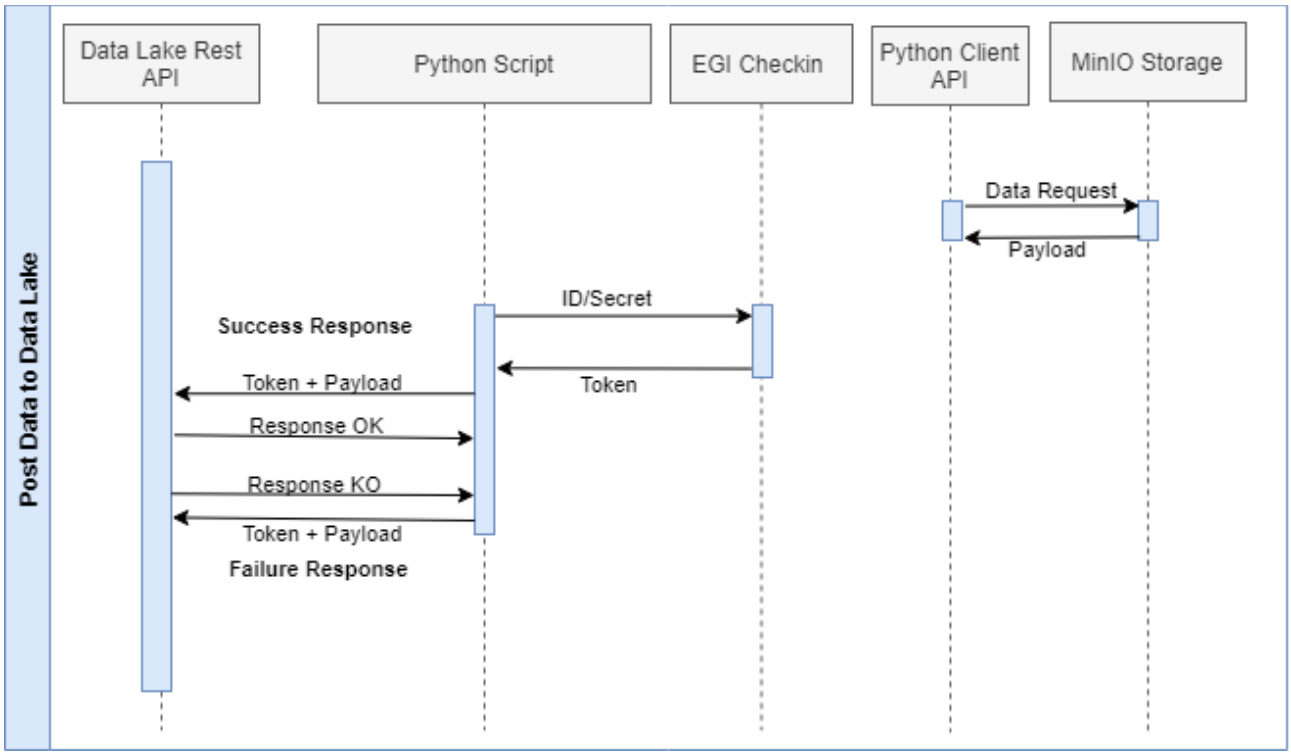

Figure 34 Posting Data to LETHE Data Lake sequence diagram, as developed by ExtraRed.

6.12 Authentication

The RADAR-base platform [21], has been developed under the scope of handling sensitive data that have been collected while being in charge of dealing with further emerging risks associated with the Internet of Things – IoT. As a result, the concepts regarding encryption, authentication, deidentification and authorization are not only of vital importance, but also imperative. These issues are addressed through a set of counter-measures that have been designed and implemented as described.

Quick Response Code Authentication

QR codes or Tokens are provided from the MP to subject registration for the respective data source. This QR code contains all the necessary information regarding the role, scope, user ID and sources of the subject. This code comes also in the form of Token which can be manually entered, instead of being scanned by the built-in QR code scanner.

Open Authentication 2

Authentication and authorization are provided by the adopted OAuth2 protocol, which is also the industry standard. The mobile apps the Refresh Token grant type [12], and the rest of the clients make use of the client credentials grant type. Characteristic examples of the two OAuth2 are the mobile apps with the with the Refresh Token grant type and the gateway of the REST-API which use the client credentials grant type.

The mobiles apps are requested to provide an access token to authorize themselves on the platform. If the access token is not provided or is faulty, then no further transaction is allowed between the platform and the app. A QR code is generated as a Refresh token from the MP for the mobile apps. This code is to be scanned by the mobile apps for the provision of a JSON Web Token, that in turn provides the authorization endpoint as a Bearer Token to an HTTPS request for a new Refresh-token and Access token pair. The latter is used for data access and post, always with respect to scopes, resources and roles assigned. Automatic refresh of the Access token is also enabled, upon expiration of the valid one, through the Refresh Token acquired.

The RADAR-base platform provides utilities for clients to easily manage the OAuth2 authorization flow [13]. The pRMT Android app uses OpenID Connect for authorization and authentication [15].

Deidentification

Strongly identifiable information is completely isolated from the RADAR-base platform and is separately stored in an independent eCRF system. The only data that the platform ever contains is purely non-identifiable data. Those two databases are associated with each other as pseudonymized data through the RADAR-base Universally Unique ID.

Reverse Proxy

Traffic is proxied to the platform through nginx web server, which can be configured in such a way to be considered as a countermeasure against Distributed Denial-of-Service attacks or can be used as a load balancer.

7 Measurements, Data Types, Data Values

This section contains all the relevant information regarding the measurement, data types and data values with respect to the under-collection data from the subjects of the clinical trial. In this section follows a thorough overview of all the digital biomarkers that are offered by the Fitbit through the wearables as well as the mobile sensors, both in terms of active and passive monitoring.

7.1 Fitbit Sense

Through the Fitbit-connector service the following data from the Fitbit Data Warehouse are received. Fitbit enables the collection of Intraday data. Intraday data is available through the Activities and Heart Rate Time Series endpoints. Intraday support can extend the detail-level response to induce a polling rate of 1 minute and 15 minutes for Activity, and 1 second and 1 minute for Heart Rate [4]. Table 12 -Table 19 thoroughly explain the data collection methods from the wearable devices.

Digital Biomarker: HeartRate
Outline: Intraday heart rate data from Fitbit sense device.

Table 12 Heart Rate Digital Biomarker Data.

Name	Type	Description
time	double	Device timestamp in UTC (s)
timeReceived	double	Time that the data was received from the Fitbit API (seconds since the Unix Epoch)
timeInterval	int	Chronological window size (s)
heartRate	int	Heart rate value (bpm)

Digital Biomarker: Calories
Outline: Intraday calories and activity level data from Fitbit sense device.

Table 13 Calories Digital Biomarker Data.

Name	Type	Description
time	double	Device timestamp in UTC (s)
timeReceived	double	Time that the data was received from the Fitbit API (seconds since the Unix Epoch)
timeInterval	int	Chronological window size (s)
calories	double	Calories in this period
level	int	Level of activity: 0 - sedentary; 1 - lightly active; 2 - fairly active; 3 - very active
mets	double	The metabolic equivalent of task (MET)

Digital Biomarker: Activity Log
Outline: Fitbit Activity Log record, containing the aggregate data of a single activity. This data is pre-processed by Fitbit, using algorithms based on the intraday data.

Table 14 Activity Log Digital Biomarker Data.

Name	Type	Description
time	double	Start time of the activity, time since the Unix Epoch (s)
timeReceived	double	Time that this record was collected by a connector. Start time of the activity, seconds since the Unix Epoch (s)
timeZoneOffset	int	Time zone offset compared to UTC (s)
timeLastModified	double	Time when the record was last modified, time since the Unix Epoch (s)
duration	float	Duration of the activity (s)
durationActive	float	Duration of the part of the activity where the participant was active (s)
id	long	Activity log ID
name	null, string	Activity name. Null if none or unknown
logType	null, string	Type of log, e.g., manual. Null if unknown
type	null, long	Fitbit-generated ID of the type of activity. Null if unknown
source	null, FitbitSource	Source of the data. Null if unknown
manualDataEntry	null, FitbitManualDataEntry	Indicates what values of this record are entered manually. Null if unknown
energy	null, float	Estimated of energy expended during the activity (kJ). Null if unknown
levels	null, FitbitActivityLevels	Levels of activity. Null if unknown
heartRate	null, FitbitActivityHeartRate	Heart rate aggregate information collected during the activity. Null if unknown
steps	null, int	Number of steps made during the activity. Null if unknown or if the type of activity is not step-based
distance	null, float	Distance covered during the activity (km). Null if unknown or if the type of activity is not distance-based
speed	null, double	Mean speed during the activity (km/h). Null if unknown or if the type of activity is not distance-based

Digital Biomarker: Steps
Outline: Intraday steps data from Fitbit sense device.

Table 15 Steps Digital Biomarker Data.

Name	Type	Description
time	double	Device timestamp in UTC (s)
timeReceived	double	Time that the data was received from the Fitbit API (seconds since the Unix Epoch)
timeInterval	int	Chronological window size (s)
steps	int	Steps taken in this period

Digital Biomarker: Sleep Classic
Fitbit classic sleep data.

Table 16 Sleep Classic Biomarker Data.

Name	Type	Description
dateTime	string	ISO 6801 formatted local date time of device (yyyy-MM-ddTHH:mm:ss)
timeReceived	double	Time that the data was received from the Fitbit API (seconds since the Unix Epoch)
duration	int	Duration at this sleep characteristic in seconds
level	enum	Level of sleep as computed by Fitbit. ["AWAKE", "RESTLESS", "ASLEEP", "UNKNOWN"]
efficiency	null, int	Calculated sleep efficiency score (in percentage). It is a percentage of the amount of time the user was asleep (and not restless) divided by the time they spent in the bed after initially falling asleep

Digital Biomarker: Sleep Stages
Outline: Fitbit stages sleep data.

Table 17 Sleep Stages Digital Biomarker Data.

Name	Type	Description
dateTime	string	ISO 6801 formatted local date time of device (yyyy-MM-ddTHH:mm:ss)
timeReceived	double	Time that the data was received from the Fitbit API (seconds since the Unix Epoch)
duration	int	Duration at this sleep characteristic in seconds
level	enum	Level of sleep as computed by Fitbit. ["DEEP", "LIGHT", "REM", "AWAKE", "UNKNOWN"]
efficiency	null, int	Calculated sleep efficiency score (in percentage). It is a percentage of the amount of time the user was asleep (and not restless) divided by the time they spent in the bed after initially falling asleep

Digital Biomarker: Source
Outline: A single Fitbit source.

Table 18 Source Digital Biomarker Data.

Name	Type	Description
id	string	Source id
name	null, string	Source name. Null if unknown
type	null, string	Source type. Null if unknown
url	null, string	Source URL. Null if unknown

Digital Biomarker: TimeZone
Outline: Fitbit profile time zone at the current time. This can be used to estimate the UTC time for the Sleep API.

Table 19 TimeZone Digital Biomarker Data.

Name	Type	Description
timeReceived	string	Time that the data was received from the Fitbit API (seconds since the Unix Epoch)
timeZoneOffset	null, int	Offset from UTC (s). If unknown, this is null

7.2 Mobile Sensors Data

Through the RADAR-base passive RMT app we have the ability to collect data from the phone sensors and also the capability to collect data from a number of wearable devices (Fitbit, Empatica, Garmin, etc.). In addition, RADAR-base passive RMT app provides the ability to add new wearable devices that can be readily connected to the passive app as plugins using a vendor SDK. Below we present the phone sensors from which we can obtain data and group them according to the type of information that is collected [30]. Table 20 Table 22 thoroughly explain the data collection methods from the mobile device sensors.

Positional Sensors
Collects the relative degrees of latitude and longitude data from GPS and/or the Android Network Location Provider (using cell tower and Wi-Fi signals). To ensure participants privacy, the absolute location is converted to a location relative to an unspecified reference offset. No absolute location data is streamed to or stored on the server. Besides the coordinates, the module also collects altitude (relative), speed, bearing and the accuracy of the location.
Location data can be used to derive information about the mobility of participants and determine commonly visited locations.

Table 20 Positional Sensors Data

Name	Type	Description
latitude	null, double	Relative latitude from an initial random reference latitude (degrees with range [-90, 90])
longitude	null, double	Relative longitude from an initial reference longitude (degrees with range [-180, 180])
altitude	null, float	Height above the WGS 84 reference ellipsoid (m)
accuracy	null, float	The estimated horizontal accuracy of this location, radial (m). We define horizontal accuracy as the radius of 68% confidence. In other words, if you draw a circle centered at this location's latitude and longitude, and with a radius equal to the accuracy, then there is a 68% probability that the true location is inside the circle.
speed	null, float	Speed over ground (m/s)
bearing	null, float	The horizontal direction of travel of this device (degrees with range (0, 360])

Other Sensors

- **Battery Level**

Collects the phone battery level and charging status upon change. It can for instance be used to help determine when the participant is travelling and when he/she is home, depending on the charging status. The battery level is only reported when it changes.

Table 21 Battery Level Data

Name	Type	Description
time	double	Device timestamp in UTC (s)
timeReceived	double	Device receiver timestamp in UTC (s)
batteryLevel	float	Battery level from 0 to 1
isPlugged	boolean	Whether the phone is connected to a power source
status	enum	Android device battery states ["CHARGING", "DISCHARGING", "NOT_CHARGING", "FULL", "UNKNOWN"]

- **Nearby Bluetooth devices**

Collects the number of paired and nearby Bluetooth devices. By default, the data is collected every hour. As most phones have Bluetooth, the number of nearby Bluetooth is a proxy for the number of people that are in the vicinity of the participant. This should be used with care however, because also most other non-person bound devices have Bluetooth capabilities. For instance, nearby printers and laptops are also counted.

Table 22 Nearby Bluetooth devices Data.

Name	Type	Description
time	double	Device timestamp in UTC (s)
timeReceived	double	Device receiver timestamp in UTC (s)
pairedDevices	null, int	Number of paired devices
nearbyDevices	null, int	Number of nearby devices
bluetoothEnabled	boolean	Whether Bluetooth is enabled

Additional to these measurements, the LETHE project will exploit further the Radar-base Passive RMT application, gathering data regarding phone and application usage. These variables refer to android phone user interaction, android phone usage event, application record counts, and application server uptime. These metrics will allow us to gather information and knowledge regarding the engagement of the user with the phone from a general perspective. Furthermore, these variables will grant the ability to compare and contrast the overall phone usage with respect to the LETHE app usage and possibly assist in offering insight and classifying the adherence or preference of the user in using the LETHE app.

Movement Sensors (such as acceleration, gyration, magnetic field) have been omitted from the collection, with the reason being two-fold. First and foremost, the measurements for such category types are fine-grained (e.g., 200μs comparatively to others which can be from every 1 minute up to 10 every minutes). This would severely affect the battery consumption from the patients' mobile phones. On the other hand, the most useful purpose movement sensor data would be to compute and evaluate fine and gross motor skills. Nevertheless, the patients would neither be instructed to carry the phones on them at all times, nor to perform their motor skills in specific manners, so as to be recognized and evaluated properly from respective system functionality. As a result, these measurements have been omitted as most probably the data would be inconclusive and not offer additional knowledge to the project, while on the other hand, would severely affect the patients' main tool of the trial, the mobile phone.

As the digitally collected data are to be used in the extended AI models and thus provide information and knowledge regarding the condition of the subjects and the progression of their lifestyle and health status, all the aforementioned measurements have been subject of discussion with the clinical team of LETHE in order to assess the suitability and overall inclusion of digital data/biomarkers.

Regarding the current status of the available measurements, all the above information still holds. Fitbit is continuously updating their API's revealing to public access new measurement. These new API endpoints are integrated into the Radar-base platform. Upon release of new updates on the Radar-base middleware platform, FORTH continuously investigates the updated packages, assesses their usefulness or criticality and works in integrating them to the LETHE project. This is especially true regarding the measurements of HRV and SpO2 which have just been made available for public access from Fitbit. FORTH is currently working on deploying the newest version since these variables are of great significance to the added knowledge on the subjects' condition.

A primary data estimation has been conducted with the available measurements, up to this point. On behalf of Radar-base, and excluding HRV and SpO2 variables, a gross estimate of 3.5MB per person/per day has been calculated that will be produced from the subjects' measurements (both Fitbit and smartphone). Of course, this part is a subject of change depending on the number of measurements harvested as well as the exact number of files to be received by the subjects and availability of their devices (e.g. actual usage of devices).

8 Middleware Platform Development

During the data collection process, data can be divided into two main categories (active data & passive data). The separation is based on how we collect the data and more specifically, what actions the user has to perform in order to collect the data of interest.
In RADAR-base, passive data sources are collected via the pRMT app, active data sources via the aRMT app. The data that we collect from the Fitbit sense smartwatch belongs to the passive data category and this is because the only requirement from the user is to wear the Fitbit sense. But, there are also active data for which the user has to perform some specific actions in order to start the data collection process.

8.1 Active Information Collection

The only data for which the user is required to perform an action so that we can capture it, is the Fitbit activity log. When the user wants to log an activity, he must enable it via Fitbit sense. Fitbit provides a plethora of actions which we can record. Most of them are presented in Figure 35.

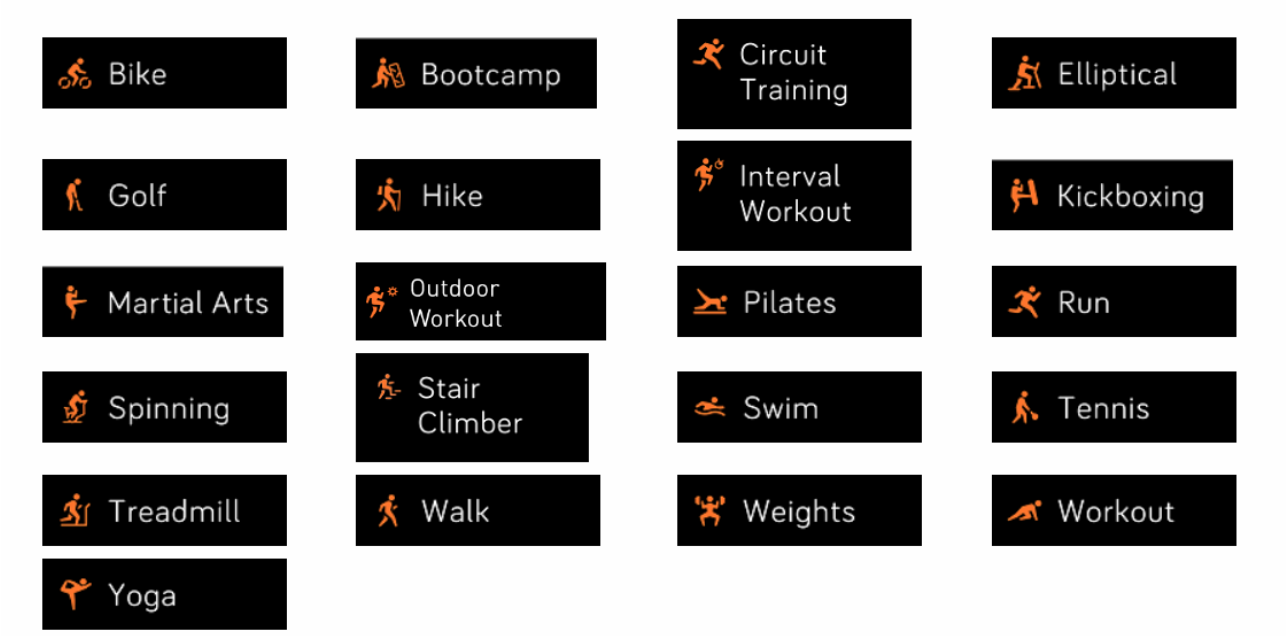

Figure 35 Activities that can be actively recorded in Fitbit smartwatches [31].

Once the user selects one of the above activities, the Fitbit sense starts recording the activity. When the process is complete, the data is sent to the platform.

8.2 Passive Information Collection

The rest of the data belong to the category of passive data because no further action is required from the user in order to be recorded. The user only needs to wear the Fitbit sense and have his mobile phone with him so that the data gets collected to the platform. Table 23 provides a data classification with respect to the type of collection of data.

Table 23 Data classification per type of collection.

Active Data	Passive Data
Activity Log	Heart Rate
	Steps
	Calories
	Sleep Stages
	Sleep Classic
	Heart Rate Variability (if timely integrated)
	Oxygen Saturation (if timely integrated)
	Relative Location
	Battery Level
	Nearby Bluetooth Devices
	Phone usage event
	Phone user interaction
	Application record counts
	Application server status
	Activity Log (semi-auto-detect)

9 Integration with the Data Lake

The ultimate goal is data to be sent to the LETHE Data Lake. This is accomplished with a Kubernetes Cronjob Service which has been implemented. Through the Cronjob Service implemented, the LETHE system is granted with the ability to periodically execute scheduled actions. Additionally, since the Cronjob Service is yet another Kubernetes object, essentially, information can be retrieved regarding the current status and the status of completion of pending and submitted jobs (GET or POST). The functionality of the Cronjob Service is split into two distinct actions: (a) Data Retrieval from MinIO Object Storage, which is a python script to download data for each subject, and (b) Data POST to LETHE Data Lake, which is a python script that posts retrieved data to the LETHE Data Lake.

Close collaboration with partners from ExtraRed (ER) has been established, since they are responsible for providing the resources as well as the respective API for writing into the Data Lake. For this explicit purpose, and since the LETHE Sensing Ecosystem is the first LETHE component to explore and utilize the storage space of the Data Lake, several integration tests have taken place to assure the high-quality development of the communication mechanisms, the registration of the data, and the authentication of the RADAR-base client application through the EGI check-in process [32].

In particular to the integration of the LETHE sensing ecosystem to the LETHE Data Lake, the EGI check-in mechanism will be used to trust the communication between the LETHE sensing ecosystem and the Data Lake. This will be assured by means of the OAuth2 protocol adopted by the EGI check-in mechanism. Upon the periodic collection of the subjects' data from the Fitbit Data Warehouse, the data must be provided to the LETHE Data Warehouse. The sensing ecosystem has been granted a Unique ID and a secret token by the EGI check-in mechanism. This Unique ID and the secret will be used to obtain an Authentication Token from EGI check-in, using a specific API call. This ID will be provided to the LETHE Data Lake, which will respond with an Authentication Token. This Authentication Token will be subsequently handed over while provisioning the data, in order for the transaction to follow through. Figure 36 depicts the architecture of the communication channel between the LETHE sensing ecosystem and the LETHE Data Lake.

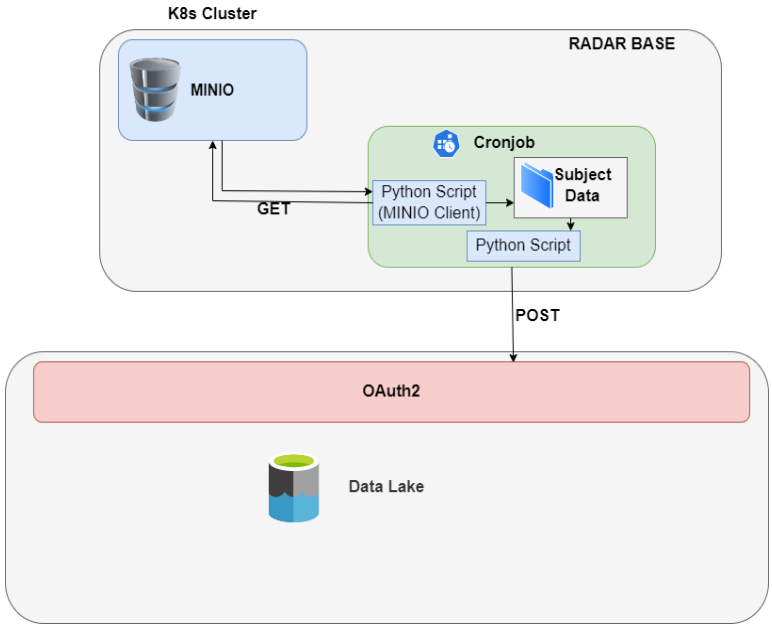

Figure 36 Architecture of data communication to the Data Lake mechanism.

To download the data from MinIO, we use the python client API provided by MinIO. MinIO Python SDK is Simple Storage Service client to perform bucket and object operations to any compatible object storage service [33]. Table 24 provides the parameters for creating a client object. The time interval data provisioning to the LETHE Data Lake is also predefined and parameterized in the script.

Table 24 Parameters for the script that periodically collects the data from MinIO.

Param	Type	Description
endpoint	str	Hostname
access_key	str	*minio.credentials.Provider*
secret_key	str	(Optional) Secret Key (aka password) of your account in S3 service
session_token	str	(Optional) Session token of your account in S3 service
secure	bool	(Optional) Flag to indicate to use secure (TLS) connection to S3 service or not
region	*str*	(Optional) Region name of buckets in S3 service.
https_client	*urllib3.poolmanager.PoolManager*	(Optional) Customized HTTPS client
credentials	*minio.credentials.Provider*	(Optional) Credentials provider of your account in S3 service

The next step involves provision of the data to the Data Lake. This is possible through a POST call, according to the API developed and provided by ER. This explicit purpose of this POST call is to identify the data to be sent, as well as the characterization of the client, the device, the time of the transaction as well as the schema of the data. The development process of the POST mechanism is currently ongoing.

10 LETHE Sensing Ecosystem Deployment and Minimum Requirements

The minimum requirements for the development of the Radar Base platform are presented in Table 25, which is also available on the RADAR-base GitHub page [21].

Table 25 RADAR-Kubernetes minimum deployment requirements.

Component	Description	Required
Kubernetes Cluster	An infrastructure with working installation of Kubernetes services. Minimum requirements for a single node: 8 vCPU's, 32 GB memory, 200 GB storage. Minimum requirements for a cluster: 3 nodes with 3 vCPUs, 16 GB memory, 100 GB storage each and 200 GB shared storage	Required
DNS Server	Some applications are only accessible via HTTPS and it's essential to have a DNS server	Required
SMTP Server	RADAR-base needs an SMTP server to send registration email to researchers and participants.	Required
Object storage	An external object storage allows RADAR-Kubernetes to backup cluster data such as manifests, application configuration and data via Velero to a backup site. You can also send the RADAR-base output data to this object storage, which can provider easier management and access compared to bundled MinIO server inside RADAR-Kubernetes.	Optional
Managed services	RADAR-Kubernetes includes all necessary components to run the platform as a standalone application. However, you can also opt to use managed services such as with the platform, e.g., Confluent cloud for Kafka and schema registry, Postgres DB for storage, Azure blob storage or AWS S3 instead of MinIO.	Optional

The RADAR-Kubernetes is currently being installed on the official testing environment as resources have just been provided by EGI. No major differentiations are expected and currently there are no major concerns regarding this official trial deployment. The different parts that need to be established refer to the Domain Name System assigned to the LETHE project as a whole, and in particular the security certificates that need to be present. These issues have been realized and addressed by the technical partners and are currently under implementation. Relevant documentation will take place in Deliverable D2.4.

10.1 Positioning in the LETHE architecture

Figure 37 highlights the positioning of the LETHE sensing ecosystem within the overall LETHE architectural approach. As illustrated, the LETHE sensing ecosystem is responsible for the collection of data from the Fitbit smartwatches and the associated mobile devices sensors, and transport them to the LETHE Data Lake. This collection and transportation of data will happen in a periodic manner, as agreed upon with the LETHE Data Lake, taking into account all aspects of the rest of the LETHE architecture components, such as use cases for the smart-glasses and/or Robot, AI/ML modules. The rest of the components receive the required data/information from the structured data repository, the LETHE Data Warehouse. Any collaboration between the other modules will happen with data communicated from the LETHE Data Warehouse.

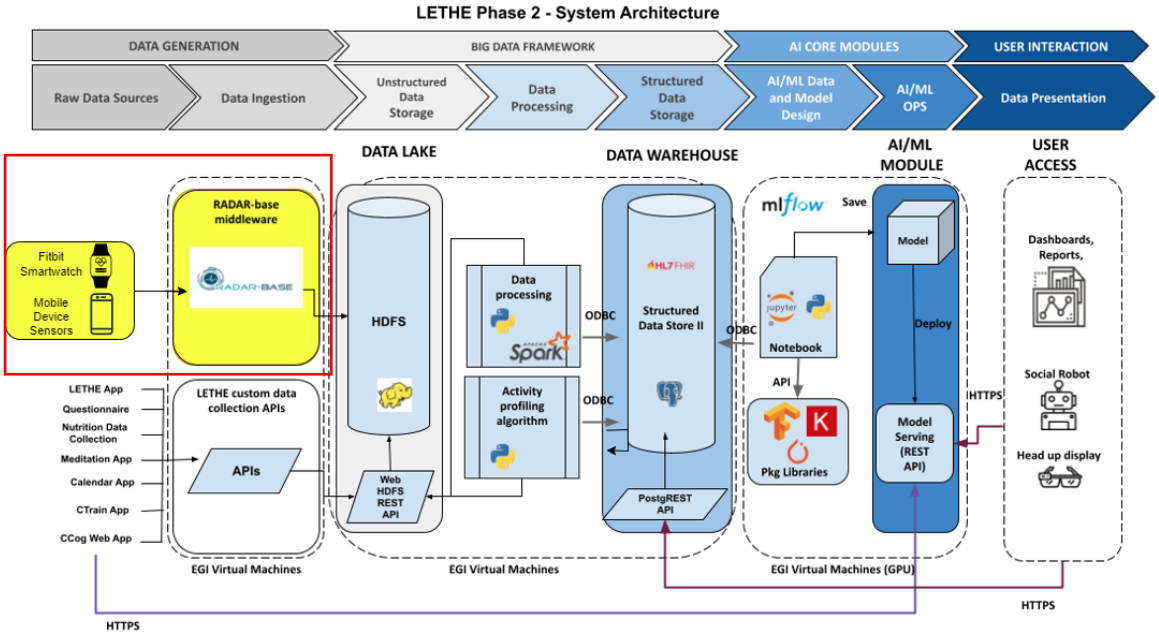

Figure 37 The LETHE sensing ecosystem within the context of the LETHE System Architecture.

11 Custom platform

Upon deprecation of the Docker version of RADAR-base, The Hyve released the new version of RADAR-base, based on Kubernetes technology. This project is in its early releases and is still under development by the company. In order to mitigate any possible risk by deploying a middleware platform in the early stages of development FORTH has put in place an alternative, simplified approach for serving the explicit purpose of collecting data from the subjects' wearable devices. The custom platform is based on the Django-Python framework [34].
Django is a Model - Template - View (MTV) framework and supports all contemporary and industry standard services like OAuth2 and REST. The MTV design pattern is illustrated in Figure 38.

- **Model:** Provides interface to the Database. The Database contains the data of the application we want to implement.
- **Template:** Provides the logic and interface between the user and the Django application.
- **View:** Manages the business logic for the application and acts as a broker between Model and View.

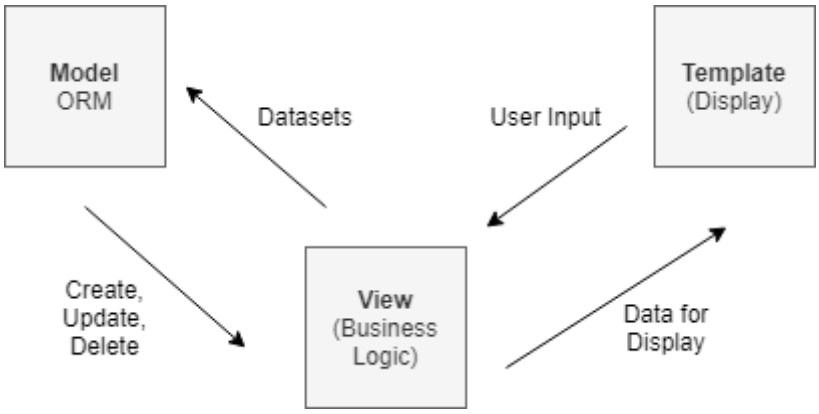

Figure 38 Model-Template-View design pattern, the backbone of the Django architecture.

11.1 Django Custom Platform Architecture

The core components of the custom platform are presented below. Figure 39 illustrates the architecture of the Django custom platform.

- **Python Fitbit Connector:** This python function is designed to retrieve data from Fitbit Web API (Authentication policies OAuth2) and post data to PostgreSQL.
- **Django Scheduler:** Python utility that schedules the Python Fitbit Connector function to run automatically at specified time and date or intervals.
- **Django Scheduler Credentials:** Python utility that schedules the renewal of credentials for each user in order to collect data from the Fitbit Data Warehouse. Renewing credentials is necessary because Fitbit uses the OAuth2 Authentication protocol.
- **PostgreSQL:** PostgreSQL is used as the primary data store for Django.
- **Django Rest Framework:** We used Django Rest Framework - DRF to create our web API (Authentication policies OAuth2, Response JSON)

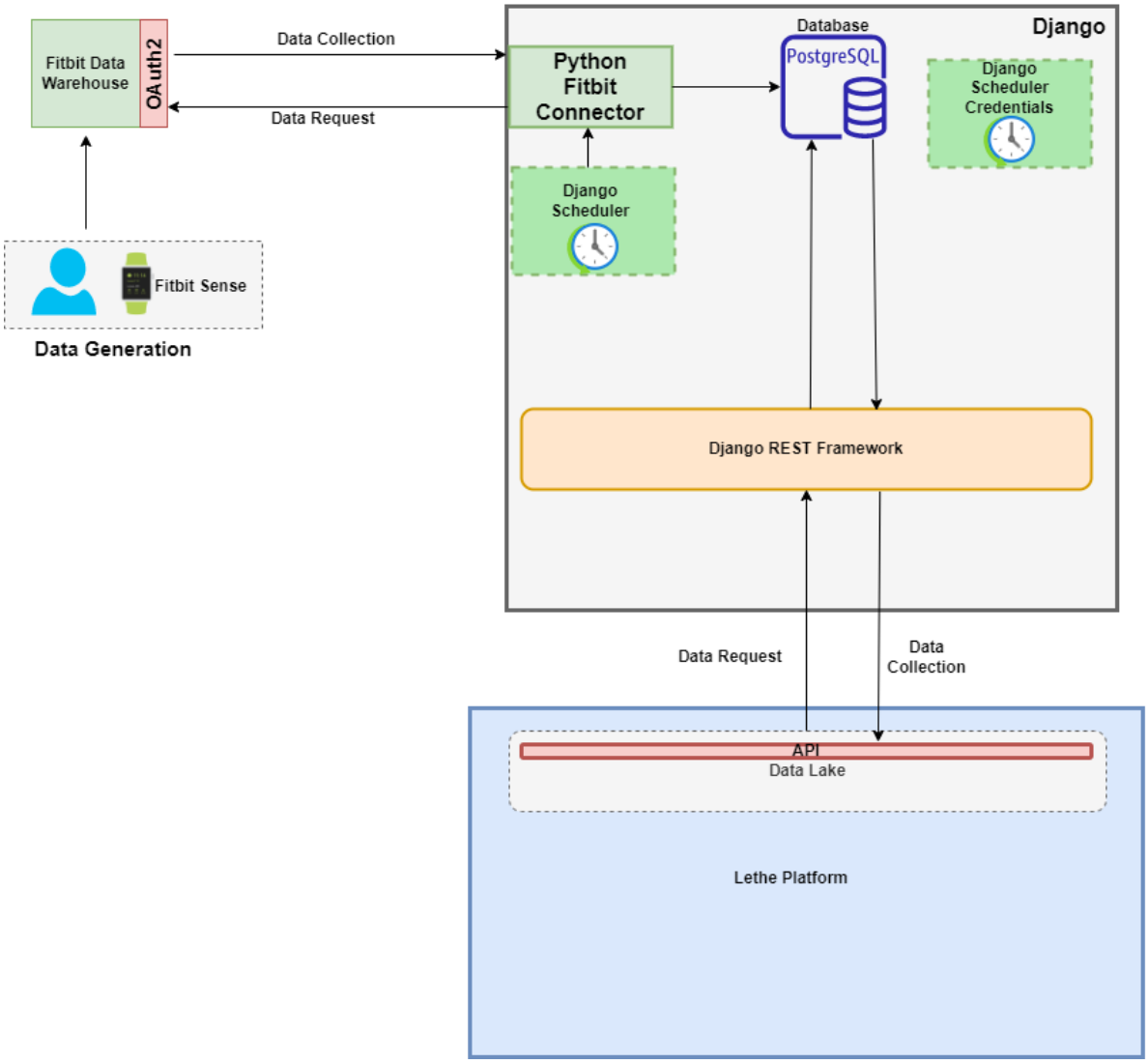

Figure 39 Custom platform architecture.

11.2 Custom platform workflow

This section presents the workflow of the custom platform, which in terms of functionality is similar to the workflow of RADAR-base, nevertheless the mechanisms are completely different. Comparatively to the RADAR-base this platform is much more simplified and not quite as complex, but without losing the usability and functionality to collect the subjects' data. For every user a Fitbit App is registered through Fitbit and their credentials are stored in our Django database.
At predefined periodic intervals, at the Django scheduler, requests are made through the python Fitbit connector (Python script) and data is collected from the Fitbit Data Warehouse. Afterwards, the data is sent to the LETHE Data Lake as described in Section 9. Figure 40, depicts the workflow of the custom-built platform.

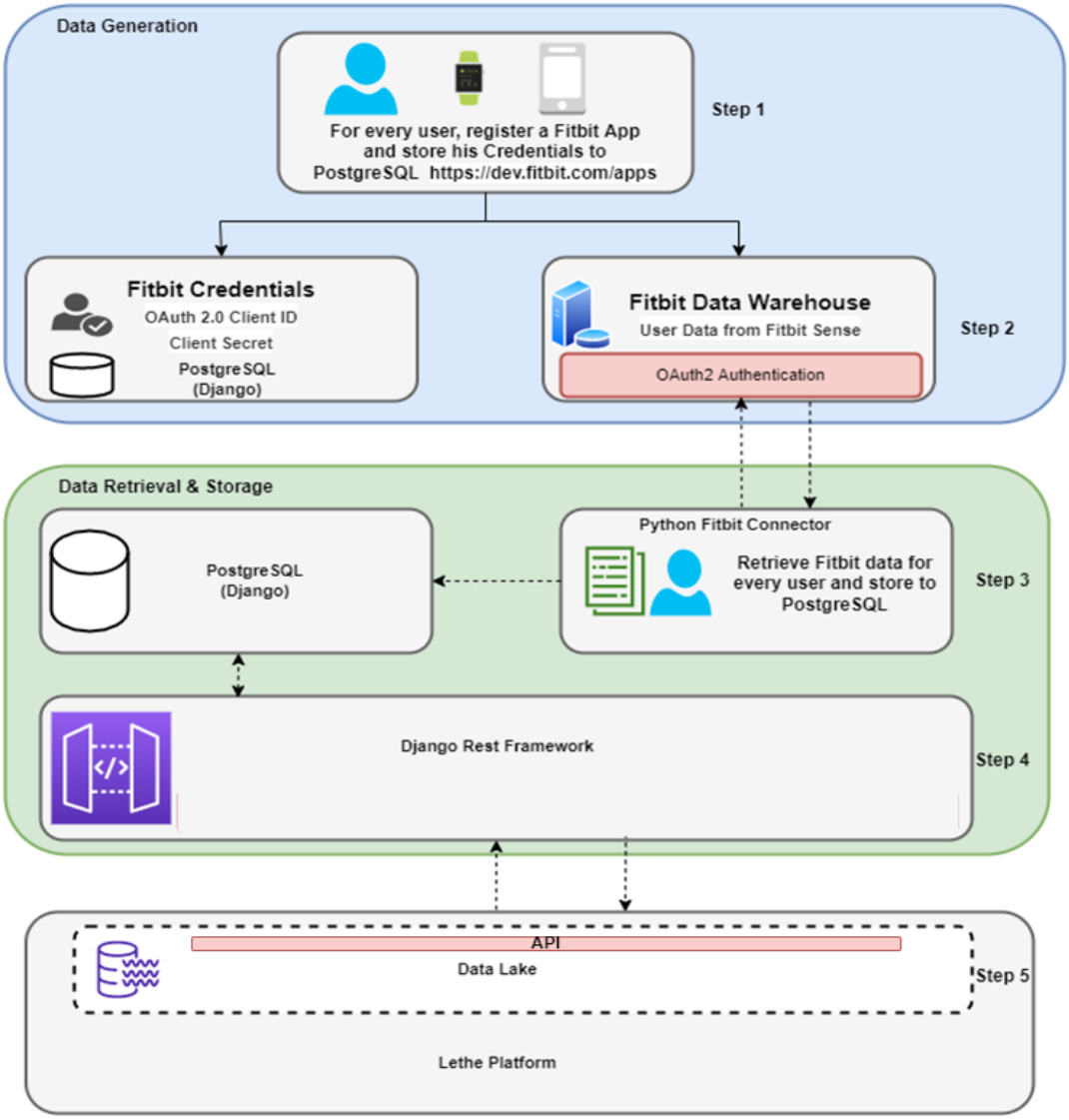

Figure 40 Custom platform workflow.

Although this custom platform has been built, deployed and tested, it has always been considered as a contingency plan in case there were difficulties deploying the RADAR-Kubernetes platform, or functionality was not as expected. It is explicitly a simplified version, but at the same time it is a reliable means of data collection from the Fitbit smartwatches that will be used in the clinical trial. Nevertheless, we have obtained a stable deployment of the RADAR-Kubernetes architecture, that is fully functional, and this plan is expected to be followed, as aforementioned.

12 Summary/Discussion

In this deliverable the LETHE sensing ecosystem is described. As a preparatory phase in advance of the final LETHE platform deployment, we gathered and analysed the current development level for sensors, techniques, and scientific fields, which were important for the development and deployment of the platform. The software components, procedures and architectural designs that should be followed for the realization of the LETHE sensing ecosystem have been described.

For the purposes of LETHE, our primary goal was to use and extend the RADAR-base platform as the backbone of the sensing ecosystem. Due to strategic decisions from the developers of the RADAR-base we had to withdraw the Docker version of RADAR-base which was the only stable version until recently. The developers of RADAR-base stated that from 2022 will support only the Kubernetes version of the product so we focused on that version. Unfortunately, until recently there was no stable version and we had to collaborate with the Hyve team for development and debugging until we reach a stable version. LETHE partners have been active in the integration efforts of RADAR-base Kubernetes to the LETHE System. FORTH has been contributing to the open-source RADAR-BASE community by identifying bugs and providing features not yet available on the platform.

Due to delays in the stable version of RADAR-base, we decided to work in parallel on an alternative platform able to support the sensing ecosystem. Towards that direction, FORTH developed, deployed and tested an in-house platform able to gather information from wearables (Fitbit) based on the open-source technology Django. Since RADAR-base has been successfully deployed in the development servers of LETHE, the consortium decided to follow the initial plan and use the extended RADAR-base platform as the LETHE sensing ecosystem.

Integration of the LETHE sensing ecosystem with the LETHE architecture is based on two main directions. Initially, the communication and data flow to the Data Lake based on well-defined REST APIs and Json format for the data, as they are developed by ER. Secondly, the integration of the LETHE security mechanism that is provided by the EGI check-in along with the pseudonymization of the data.

This document is a snapshot of the evolving LETHE sensing ecosystem as the project progresses. It should be noted that LETHE takes an iterative approach towards the design of the architecture, so this first iteration does not by any means offer the final solution. Although, the under-deployment system has already reached its final form by means of RADAR-Kubernetes, a most recent device may be procured for the patients that may offer further data, or in case Fitbit's API gets updated releasing additional measurements to the public, some alteration may occur. In this case, and if required the deliverable D4.3 will be updated accordingly.

12.1 Current Status and Limitations

The current status of the LETHE sensing ecosystem is that it has been deployed in a private development environment provided by FORTH. For this deployment, major debugging has taken place in collaboration with The Hyve. The deployment was based on the formal minimum required for a Kubernetes cluster with 3 nodes, 1 master node and 2 worker nodes. We have been successful in registering projects, subjects, roles, pairing mobile applications with the respective subject's account, consistently collecting data from the Fitbit Data Warehouse and providing them to the LETHE Data Lake. The next step is to migrate/re-deploy the LETHE sensing ecosystem to the clustered nodes that have just been provided by EGI.

As RADAR-Kubernetes is the newest release of RADAR-base, there have been identified and mitigated several bugs that did not allow its deployment. It is possible that during the next months, critical updates will be rolled out from the Hyve team will be carefully inspected and upon testing might be integrated to our platform as well. Furthermore, more API endpoints might be integrated by the Hyve, which will make more digital biomarkers publicly available, which due to the modularity of the platform might also be integrated to LETHE as well.

One limitation is that the mobile sensors measurements, such as accelerometer or gyroscope and magnetic field, are energy intensive services and drain the mobile battery. A characteristic example of this is the

Accelerometer measurement which is executed at 200 millisecond intervals. This not only drains the mobiles batteries, but also accumulates huge amounts of information in the LETHE storage structures, which could be obtained more efficiently, such as step count or activity information.
On the other hand, if measurements are not fine-grained no useful information can be extracted. For example, if the polling rate of the GPS location is set to one hour, we cannot track if the subject has taken a quick walk. On the contrary, if the polling rate is set to 10 minutes, then the phones need to be recharged more than once per day. A countermeasure for using the GPS data, could possibly be the Nearby Bluetooth Devices count and its derivative as an indication of mobility and socialization based on the surrounding located objects.
Another limitation could be that raw data cannot be extracted from the Fitbit devices. For example, raw data in heart rate measurements could be beneficial in terms of calculating the Heart Rate Variability – HRV. Nevertheless, the data that we receive is averaged through every minute. Devices that could provide raw sensor data were well above the budget allocated for the devices, such as the Empatica E4.
The LETHE sensing ecosystem, at its current state, is a stable deployment of the RADAR-Kubernetes and has been granted with full functionality that is up to this released. FORTH will integrate any possible updates that might be of interest or critical in any aspect of security, stability or modularity. FORTH will continue to investigate every recent event regarding the RADAR-Kubernetes, Fitbit API, and respective functionalities in order to provide the best result with respect to the upcoming LETHE clinical trial. The collection of prospective data in terms of digital biomarkers is of great importance and sets the basis for the rest of the LETHE modules and the novelty of possible research outcomes of the clinical trial.
During June 2022, Fitbit have updated their API endpoints, providing the measurements of Heart Rate Variability (HRV) as well as the Oxygen Saturation (SpO2), as publicly available hooks of data collection. Currently, The Hyve is working on integrating these API endpoints to Radar-Kubernetes. Furthermore, The Hyve has planned and already started a major update on the Radar-Kubernetes API. This essentially changes the way that data are communicated from Fitbit Warehouse to the Radar-Kubernetes. These two changes overlap in terms of implementation time plan. FORTH will continuously work on debugging, automating, and deploying in a stable manner the Radar-Kubernetes component with all updated functionality, such as measurements of interest (e.g., HRV, SpO2).

13 References

[1] B. K. Kennedy *et al.*, "Geroscience: linking aging to chronic disease," *Cell*, vol. 159, no. 4, pp. 709–713, Nov. 2014, doi: 10.1016/j.cell.2014.10.039.
[2] "The Intersection Between Aging and Cardiovascular Disease | Circulation Research." https://www.ahajournals.org/doi/10.1161/circresaha.111.246876 (accessed May 02, 2022).
[3] "Archives of Cardiovascular Diseases - Journal - Elsevier." https://www.journals.elsevier.com/journals.elsevier.com/archives-of-cardiovascular-diseases (accessed May 02, 2022).
[4] G. Viegi, S. Maio, S. Fasola, and S. Baldacci, "Global Burden of Chronic Respiratory Diseases," *J. Aerosol Med. Pulm. Drug Deliv.*, vol. 33, no. 4, pp. 171–177, Aug. 2020, doi: 10.1089/jamp.2019.1576.
[5] T. Estapé, "Cancer in the Elderly: Challenges and Barriers," *Asia-Pac. J. Oncol. Nurs.*, vol. 5, no. 1, pp. 40–42, 2018, doi: 10.4103/apjon.apjon_52_17.
[6] L. McWhirter, C. Ritchie, J. Stone, and A. Carson, "Functional cognitive disorders: a systematic review," *Lancet Psychiatry*, vol. 7, no. 2, pp. 191–207, Feb. 2020, doi: 10.1016/S2215-0366(19)30405-5.
[7] M. Kivipelto, F. Mangialasche, and T. Ngandu, "Lifestyle interventions to prevent cognitive impairment, dementia and Alzheimer disease," *Nat. Rev. Neurol.*, vol. 14, no. 11, Art. no. 11, Nov. 2018, doi: 10.1038/s41582-018-0070-3.
[8] "What is REST," *REST API Tutorial*. https://restfulapi.net/ (accessed May 02, 2022).
[9] M. R. KAMDAR and M. J. WU, "PRISM: A DATA-DRIVEN PLATFORM FOR MONITORING MENTAL HEALTH," *Pac. Symp. Biocomput. Pac. Symp. Biocomput.*, vol. 21, pp. 333–344, 2016.
[10] "Home," *Beiwe / Forest*. https://www.beiwe.org/ (accessed May 02, 2022).
[11] K. Bhavsar *et al.*, "Digital Biomarkers: Using Smartwatch Data For Clinically Relevant Outcomes," in *2021 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*, Mar. 2021, pp. 630–635. doi: 10.1109/PerComWorkshops51409.2021.9431000.
[12] A. Monteriù *et al.*, "A Smart Sensing Architecture for Domestic Monitoring: Methodological Approach and Experimental Validation," *Sensors*, vol. 18, no. 7, Art. no. 7, Jul. 2018, doi: 10.3390/s18072310.
[13] A. Helal, D. J. Cook, and M. Schmalz, "Smart Home-Based Health Platform for Behavioral Monitoring and Alteration of Diabetes Patients," *J. Diabetes Sci. Technol. Online*, vol. 3, no. 1, pp. 141–148, Jan. 2009.
[14] "Smart Bear Project." https://www.smart-bear.eu/ (accessed May 02, 2022).
[15] Y. Ranjan *et al.*, "RADAR-Base: Open Source Mobile Health Platform for Collecting, Monitoring, and Analyzing Data Using Sensors, Wearables, and Mobile Devices," *JMIR MHealth UHealth*, vol. 7, no. 8, p. e11734, Aug. 2019, doi: 10.2196/11734.
[16] "COVID-Red." https://www.covid-red.eu/en/ (accessed May 02, 2022).
[17] "UNFOLD," *Research Into Recovery*. https://www.researchintorecovery.com/research/unfold/ (accessed May 02, 2022).
[18] "Home page," *DynaMORE*. https://dynamore-project.eu/ (accessed May 02, 2022).
[19] "Welcome to RADAR-AD | Radar-AD." https://www.radar-ad.org/welcome-radar-ad (accessed Jul. 08, 2022).
[20] "Projects Archive," *RADAR-base*. https://radar-base.org/index.php/project/ (accessed May 02, 2022).
[21] *RADAR-Kubernetes*. RADAR-base, 2022. Accessed: May 02, 2022. [Online]. Available: https://github.com/RADAR-base/RADAR-Kubernetes
[22] "Home | The Hyve." https://www.thehyve.nl/ (accessed May 02, 2022).
[23] "Kubernetes." https://kubernetes.io/ (accessed May 02, 2022).
[24] "Apache Kafka," *Apache Kafka*. https://kafka.apache.org/ (accessed May 02, 2022).
[25] "Graylog Docs." https://docs.graylog.org/ (accessed May 02, 2022).
[26] Prometheus, "Prometheus - Monitoring system & time series database." https://prometheus.io/ (accessed May 02, 2022).

[27] "Grafana: The open observability platform," *Grafana Labs*. https://grafana.com/ (accessed May 01, 2022).
[28] "Prometheus and Grafana: The Perfect Combo," *Epsagon*, Mar. 23, 2020. https://epsagon.com/tools/prometheus-and-grafana-the-perfect-combo/ (accessed May 02, 2022).
[29] M. Inc, "MinIO | High Performance, Kubernetes Native Object Storage," *MinIO*. https://min.io (accessed May 02, 2022).
[30] "Phone Data Sensors," *RADAR-base*. https://radar-base.org/index.php/data-sensors/ (accessed May 02, 2022).
[31] "How do I track my activity with my Fitbit device?" https://help.fitbit.com/articles/en_US/Help_article/1785.htm (accessed May 02, 2022).
[32] "EGI | Check-in." https://www.egi.eu/services/check-in/ (accessed May 02, 2022).
[33] "MinIO | Python Client Quickstart Guide." https://docs.min.io/docs/python-client-quickstart-guide.html (accessed May 02, 2022).
[34] "The web framework for perfectionists with deadlines | Django." https://www.djangoproject.com/ (accessed May 02, 2022).

14 Annex