Personalised Health Monitoring and Decision Support Based on Artificial Intelligence and Holistic Health Records

D2.4 – Conceptual model and reference architecture

WP2 Requirements, State of the Art Analysis and User Scenarios in iHelp

Dissemination Level: Public
Document type: Report
Version: 1.0.0
Date: August 30, 2021

The project iHelp has received funding from the European Union's Horizon 2020 Programme for research, technological development, and demonstration under grant agreement no 101017441.

Document Details

Project Number	101017441
Project Title	iHelp - Personalised Health Monitoring and Decision Support Based on Artificial Intelligence and Holistic Health Records
Title of deliverable	Conceptual model and reference architecture
Work package	WP2
Due Date	31/08/2021
Submission Date	30/08/2021
Start Date of Project	January 1, 2021
Duration of project	36 months
Main Responsible Partner	Engineering Ingegneria Informatica (ENG)
Deliverable nature	Report
Author name(s)	Fabio Melillo (ENG) – Contributors: Ainhoa Azqueta (UPM), Miriam Cabrita (iSprint), Athanasios Dalianis (ATC), Antonio De Nigro (ENG), Krasimir Filipov (KOD), George Giotis (ATC), Maritini Kalogerini (ATC), George Manias (UPRC), Irida Manika (iSprint), Patricio Martinez (LXS), Nikolay Mehandjiev (KOD), Harm op den Akker (iSprint), Marta Patiño (UPM), Aristodemos Pnevmatikakis (iSprint), Jacob Roldan (LXS), Usman Wajid (ICE)
Reviewer name(s)	Jose María Zaragoza (LXS) Andrea Damiani, Nikola Dino Capocchiano, Calogero Casá, Livia Lilli (FPG)

Document Revision History

Version History			
Version	**Date**	**Author(s)**	**Changes made**
0.0.0	2021-02-16	Irida Manika (iSprint) Harm op den Akker (iSprint)	iHELP Deliverable Template – Initial version.
0.0.1	2021-03-10	Fabio Melillo (ENG)	First template for collecting the contributions
0.2.0	2021-03-15	Fabio Melillo (ENG)	Edit of the template structure
0.3.0	2021-03-20	Fabio Melillo (ENG)	Simplification of section 6
0.3.1	2021-03-23	Krasimir Filipov (KOD) Nikolay Mehandjiev (KOD)	KOD contribution
0.3.2	2021-03-30	George Manias (UPRC)	UPRC contribution

0.3.3	2021-04-02	Fabio Melillo (ENG)	Edit of the template structure. Added Conceptual Model section
0.3.4	2021-03-30	Usman Wajid (ICE)	Contribution in ICE related sections
0.3.5	2021-04-25	Marta Patiño (UPM) Ainhoa Azqueta (UPM)	UPM contribution
0.3.6	2021-04-30	George Giotis (ATC) Athanasios Dalianis (ATC) Maritini Kalogerini (ATC)	ATC contribution
0.3.7	2021-05-10	Aristodemos Pnevmatikakis (iSprint) Miriam Cabrita (iSprint) Harm op den Akker (iSprint)	iSprint contribution
0.3.8	2021-05-17	Jacob Roldan (LXS) Patricio Martinez (LXS)	LXS contribution
0.3.9	2021-06-16	Fabio Melillo (ENG)	Included the KI comments
0.3.10	2021-06-17	Jacob Roldan (LXS) Patricio Martinez (LXS) Harm op den Akker (iSprint)	LXS and iSprint addressed the KI comments. Documents merged.
0.3.11	2021-07-08	Fabio Melillo (ENG)	Edited the sections: 1, 2, 3, 4, 5, 6
0.3.12	2021-07-09	George Manias (UPRC)	Contribution to section 7
0.3.13	2021-07-12	Fabio Melillo (ENG)	Edited the sections 8, 9
0.3.14	2021-07-13	Antonio De Nigro (ENG) Fabio Melillo (ENG)	Edited the sections 1, 2, 3, 5, 7, 8, 9
0.3.15	2021-07-14	George Manias (UPRC)	Added the “Infrastructure discussion” document in the Annex
0.3.16	2021-07-21	Jose María Zaragoza (LXS)	Internal review
0.3.17	2021-07-26	Fabio Melillo (ENG)	Resolved feedbacks and comments
0.3.18	2021-07-31	Andrea Damiani (FPG)	Internal review
0.9	2021-08-03	Fabio Melillo (ENG)	Final Version
1.0	2021-08-30	Dimosthenis Kyriazis (UPRC)	Quality check and final version

Table of Contents

List of Figures

Executive summary

iHelp aims to early detect and mitigate the risks associated with Pancreatic Cancer applying advanced Artificial Intelligence (AI)-based techniques to support the actors of the system. Those techniques are performed on historic data of cancer patients gathered from existing cohorts and biobanks. The models developed, through the AI-based learning techniques will be useful to identify the risk earlier, and to elaborate a mitigation plan.

To this end, iHelp will use and extend the paradigm of Holistic Health Records (HHR) [1], in order to aggregate and re-use data needed by the Artificial Intelligence for developing their model. Several different sources will feed the common shared informative base, and for achieving this goal, a framework for the data ingestion will be provided for the pre-elaboration of different data in order to let it flow towards the shared data model.

The purpose of this deliverable is to provide the consortium with a common high-level view about the whole solution. The details of each software artefact are delegated to the respective leader and will be described in the related deliverables.

During the first months, the consortium had several virtual meetings to collect information and to agree about a common starting point for the architecture. After the first phase of coarse-grained data collection, some analysis was applied to design an efficient architecture based on the requirement emerged. Other refinement phases were performed regarding the design patterns to be implemented, provided and required interfaces, interconnections and protocols.

The common agreement was reached about the general view, the platform will be constituted by containerised services, both from the data ingestion and the data analysis, with slightly differences explained in this document.

The input of this deliverable was the *D2.1 – State of the art and Requirements Analysis [2]*, that was analysed, and from which the needed components and interactions are identified.

This deliverable is being released at M08 of the project, and it will be useful for the *T2.3 – Functional and Non-Functional specifications* for defining its purpose, and to all the technical partners as reference architecture for the whole iHelp solution.

This is the first release of reference architecture; the second and final one will be released at M18: *D2.5 - Conceptual model and reference architecture-II.*

1 Introduction

The purpose of this document is to provide an initial architecture that will drive the interactions among the different software artefacts developed by the iHelp partners. That is also a basis for the component names and will offers a preliminary analysis of the possible deployments.

The main purpose of this document is to define some guidelines among the consortium and to share a common view about the whole platform. The main topics that this version of the document addresses concern the kind of software artefacts that will be developed, the communication pattern used, the kind of graphical user interface provided, the infrastructure needed for testing and validation, and the envisaged deployment.

This design document is the product of the work carried out by the task *T2.2 - Reference Architecture Specifications*. Main goal of that task is to analyse the requirements and to mediate with the technical partners to reach an agreement on which to design a reference architecture able to satisfy the requirements presents in *D2.1* [2]. All the component leaders were involved in the definition, refinement and agreement on this common reference architecture.

This task ends in M20 and the refinement of this overall architecture is expected in M18 with the second version of this deliverable *D2.5 – Conceptual model and reference architecture-II* (M18).

This document does not address the internal design choices of the components, or the different nature of the data to be ingested or analysed, but it aims to clarify the impact of the gathered requirements onto the designed components and to highlight and describe some useful interactions.

The organisation of the document is the following:

Section 2 provides a global overview on the adopted architecture definition approach;

Section 3 aims to describe, both for technical and non-technical people, how iHelp fits into the existing environments, who and how will use the system;

Section 4 is focused to describe what the main functionalities of the system are;

Section 5 recalls the principles and the assumptions made about security and privacy aspects;

Section 6 collects the envisaged software artefacts that will compose the iHelp solution;

Section 7 due to the heterogeneous nature of the pilot partners, this section will depict the infrastructure needed by iHelp in order to perform its tasks and provide its features;

Section 8 based on the infrastructure identified in the previous section, this section will map the component envisaged on the identified infrastructure for the actual deploy.

2 Approach to the architecture definition

The main goal of the reference architecture of iHelp is to design a limited and well-defined set of component functionalities satisfying the user requirements scheduled for the first period of the project.

The architecture described in this document is the first release and it could be extended by designing new features according to the consolidation or emergence of new user requirements, which will occur optionally during the development of the project. The adopted methodology and techniques guarantee a good level of confidence in the architectural choices made so far; however, the choices made in the current version of the architecture can change as a consequence of a change of user requirements.

The approach adopted to describe the iHelp top-level architecture is based on the existing literature[3] and on previous experiences in other EU projects [4][5][6][7].

The architecture reported here has the following objectives:

- It serves as the blueprint both for the system and the project developing it.
- It defines the work assignments, in terms of component functionalities, that must be carried out by separate design and implementation teams.
- It is a vehicle for early analysis to make sure that the design approach will yield an acceptable system.
- It is the artifact that holds the key to post-deployment system understanding and/or mining efforts.

The definition of the iHelp architecture follows some goodness principles proposed in the scope of the NEXOF-RA project[8], which represents a quality model for the architectural design of software systems. The main quality attributes, considered in evaluating the alternative architectural choices, are the efficiency and the buildability (i.e. ease of realisation). This means that when multiple architectural alternatives were analysed, the solution considered most efficient and, in order of priority, the easiest to realise has been adopted. The most *efficient* solution for each component is the one that minimizes: (i) the time requested to perform its tasks, (ii) the amount of disk space requested to store internal data enabling its normal operation, and (iii) the overhead of communication with other components to exchange requested and provided data. When the simultaneous minimization of these three parameters has been not possible because of conflicting conditions, 'the best' trade-off among them has been chosen. The criteria to define the best trade-off assigns the highest priority to the minimization of the execution time requested to perform the tasks, secondly to the minimization of the overhead of communication with other components, and finally to the minimization of the amount of disk space required.

Accordingly, when high hardware requirements were expected, like CPU throughput and/or amount of memory, then it was decided to adopt a dedicated server in order to minimize the execution time even if such a choice does not optimize the efficiency in terms of communication overhead among distinct components. For the components designed to be replicable, in case of high load, the dynamic horizontal scaling has to be taken in account (i.e. for the Ingestion Pipeline Components §6.3). Naturally, when possible, it was agreed to have more components running on the same server and in the same execution environment (e.g. virtual machine, Kubernetes node) so to try to improve the communication as well (e.g. Data analysis components §6.4).

Other quality attributes have been taken in consideration, like the *modifiability* of the system, which is one of the most important quality attributes considered during the design. Indeed, the adopted incremental approach[9] implies continuous changes to the architecture and a highly modifiable system is strongly recommended. According to the adopted quality model, the modifiability is a complex attribute measured in terms of *extensibility* of capabilities, i.e. the ability to add new functionalities with less impact on the overall system, the deletion of unwanted capabilities, the *portability*, i.e. the ability of the system to run under different executing environment and the *restructuring*, i.e. the ability to support architectural configuration changes, such as rationalising system services, modularising, optimising or creating reusable components.

The *reusability* of some of the main components is a highly desired aspect for iHelp, even if it is not mandatory. Thus, one secondary goal of the iHelp architecture is to reduce as much as possible the coupling between components/micro-services, while keeping each component as cohesive as possible. The reusability of the components is an aspect that will be further investigated during the next steps of the project.

The possible constraints of the infrastructure that will host the iHelp trial platform have been tentatively evaluated in terms of the number of required servers and, for each of them, the amount of the required computational resources and disk space. Moreover, the technologies needed for the correct operation of the components have been considered; they are reported in Figure 17: Identified tools. The objective of such investigation is to verify the feasibility of the demonstrator and to gather any technical requirements from the infrastructure providers (i.e. the pilots' infrastructure for the on-premises deployments). The result of such investigation leads to a gross grained estimation of the hardware requirements to execute iHelp components, provided by the component owners and submitted to the infrastructure provider for an acceptance validation. The estimated infrastructure requirements are reported in §7 and in the Annex A - Infrastructure Discussion.

The whole design phase, instead of focusing on specific tool/language and techniques that development teams use, or micro-managing the internal architecture of the components/micro-services, concentrates on the protocols and interactions between the various software artefact and on the health and usefulness of the system as a whole.

3 Context

In the iHelp project, a software platform will be developed that will provide several features, interacting with different actors. From the first version of deliverable *D2.1* [2] these kinds of actors emerged:

- **`Health Care Professional`** (HCP): the physicians that will take care of the citizens' health and risk of developing the Pancreatic Cancer
- **`Individual`**: the citizen or patient that will be enrolled in the iHelp project and that can access its functionalities
- **`Model Builder`**: the data scientist that can train the models for better tuning the evaluation algorithms of the AI
- **`External Sources`**: other external system, such as: laboratories, social networks, data bases, etc.

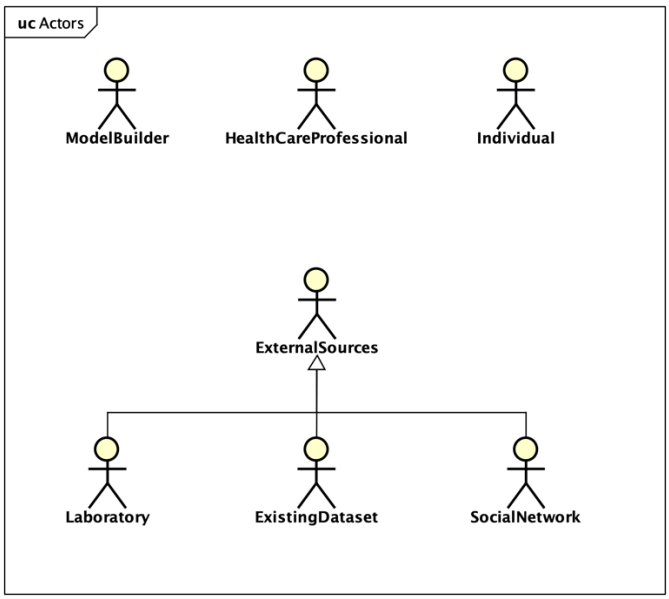

Figure 1: Actors of the system

The details of the target users and personas envisaged and analysed will be detailed in the scope of the deliverable produced by the *T2.4 – User centred design*.

The following diagram (Figure 2) places the actors in the context of the iHelp platform; the diagram will drill down into the details in the next section (§6.1).

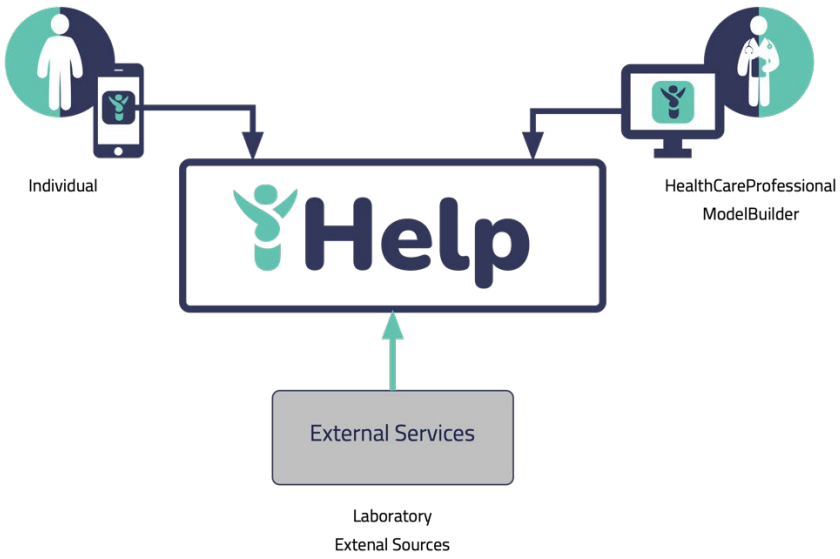

Figure 2: Context diagram

The big iHelp box, in the middle of the Figure 2, abstracts all the complexity of the platform, that will include: the ability of the system to accept and transform data from external sources to the common data model; the components that implement the AI of the system; the services that provide the presentation layer for rendering the final user applications, the components that will manage the authentication, infrastructure components, and so on. All those main components will be detailed in the following sections.

4 Functional Overview

Starting from the deliverable *D2.1* [2] it is possible to enumerate the main use cases that drove the design phase; following their label with a short description of the envisaged behaviours. All details and specific interaction steps will be detailed in the scope of the task *T2.3 – Functional and Non-Functional specifications.*

- **Develop Risk Prediction Model**: allows the **`ModelBuilder`** to develop the prediction model
- **Risk Assessment**: allows the **`HCP`** to perform/trigger a risk assessment for a specific patient
- **Request Tests and Samples**: allows the **`HCP`** to generate a list of Tests that the patient has to perform
- **Elevated Risk Detected**: highlights to the **`HCP`** that an elevated risk of PC development was detected
- **Ingest Tests and Samples results**: allows the ingestion of data coming from a pilot laboratory test
- **Risk Mitigation/Treatment planning**: allows the **`HCP`** to plan a list of task(s) to be communicated to the patients in order to mitigate or treat the exacerbation of PC
- **Advice Review**: allows the **`HCP`** to endorse the suggestion proposed by the AI
- **Risk Mitigation Delivery**: provides delivery of messages for the mitigation plan
- **Monitoring**: allows the automatic monitoring of patient shared data, in order to estimate if their behaviour is in line with the mitigation plan
- **Advice Follow-up**: allows the **`HCP`** to look at the progress of the mitigation plan
- **Caption of profiling characteristics**: allows the **`HCP`** to input relevant patients' characteristics into the platform
- **Reporting**: shows to the **`HCP`** patients data in form of aggregate and elaborate report
- **Plan a visit or a remote contact**: allows the **`HCP`** to define a visit or a remote contact that will be shared with the patient
- **Data Visualisation**: is a view on the data present in the project storage, without the elaboration and aggregation computed by other use cases
- **Communication of risk**: allows to communicate to the patient the computed risk of developing the PC

From deliverable *D2.1* [2] and analysing this list of use cases, it is clear that the iHelp platform aims to support both clinicians and patients. The overview of the principal function produced a list of common requirements, that also impact the overall architecture.

The first need, clear for all the pilot, is the ability of the platform to ingest data, coming from different sources. Those data, managed following the ethics principles described in deliverable *D1.10* [10] are accepted, formatted, cleaned, mapped and transformed in a common data model; this pipeline will be better detailed in the scope of the *WP3*.

Those uniformed data are stored inside the iHelp storage repository, a big data platform, for further queries; the *WP4* is taking care of this process.

The data that comes into the repository, can also come from the **`Individual`**, through a mobile application and its dedicated backend. Those data will feed the repository as well, through the same ingestion pipeline.

A **ModelBuilder**, using its dedicated GUI, will train the AI algorithms on the data present in the repository, that will support the **HCP**s in their job.

The same data will be accessed by all the components that provide the AI algorithms, analyses and aggregations that will be shown to the **HCP**; those components, model and algorithms will be taken care of in the scope of *WP4* and *WP5*.

5 Security and Privacy

The system shall be compliant with the security and privacy requirements expressed in the deliverable *D2.1 [2]*, and all the components are responsible to comply with the Privacy-By-Design and Privacy-By-Default principles presents in the General Data Protection Regulation (GDPR) [11] and any other privacy constraint that could emerge from the use cases.

The GDPR is a regulation in EU law on data protection and privacy for all individuals within the European Union (EU) and the European Economic Area (EEA). It also addresses the export of personal data outside the EU and EEA areas. The GDPR aims primarily to give control to individuals over their personal data and to simplify the regulatory environment for international business by unifying the regulation within the EU. Superseding the Data Protection Directive 95/46/EC, the regulation contains provisions and requirements pertaining to the processing of personal data of individuals inside the EEA, and applies to an enterprise established in the EEA or - regardless of its location and the data subjects' citizenship—that is processing the personal information of data subjects inside the EEA.

The GDPR will apply when there is a “processing” of personal data. The activities that are considered to constitute a ‘processing’ have not gone through any significant change within the GDPR, which maintains a very broad scope of application.

Accordingly, the GDPR will apply in case of “any operation or set of operations which is performed on personal data or on sets of personal data, whether or not by automated means, such as collection, recording, organisation, structuring, storage, adaptation or alteration, retrieval, consultation, use, disclosure by transmission, dissemination or otherwise making available, alignment or combination, restriction, erasure or destruction”.

It goes without saying that iHelp involves processing of personal data within the meaning of the GDPR. Hence, the various principles and obligations set therein will need to be carefully assessed and complied with by the stakeholders involved.

The data protection principles are at the core of the processing of personal data. Many of them already existed under the Data Protection Directive and are now reinforced in the GDPR. Article 5(1) of the GDPR lists the key principles relating to the processing of personal data (examined in the following six sub-Sections). Article 5(2) provides for a general principle of "accountability", according to which the controller[1] shall be responsible for, and able to demonstrate compliance with, the other six principles.

[1] The data controller determines the purposes for which and the means by which personal data is processed. The data processor processes personal data only on behalf of the controller.[29]

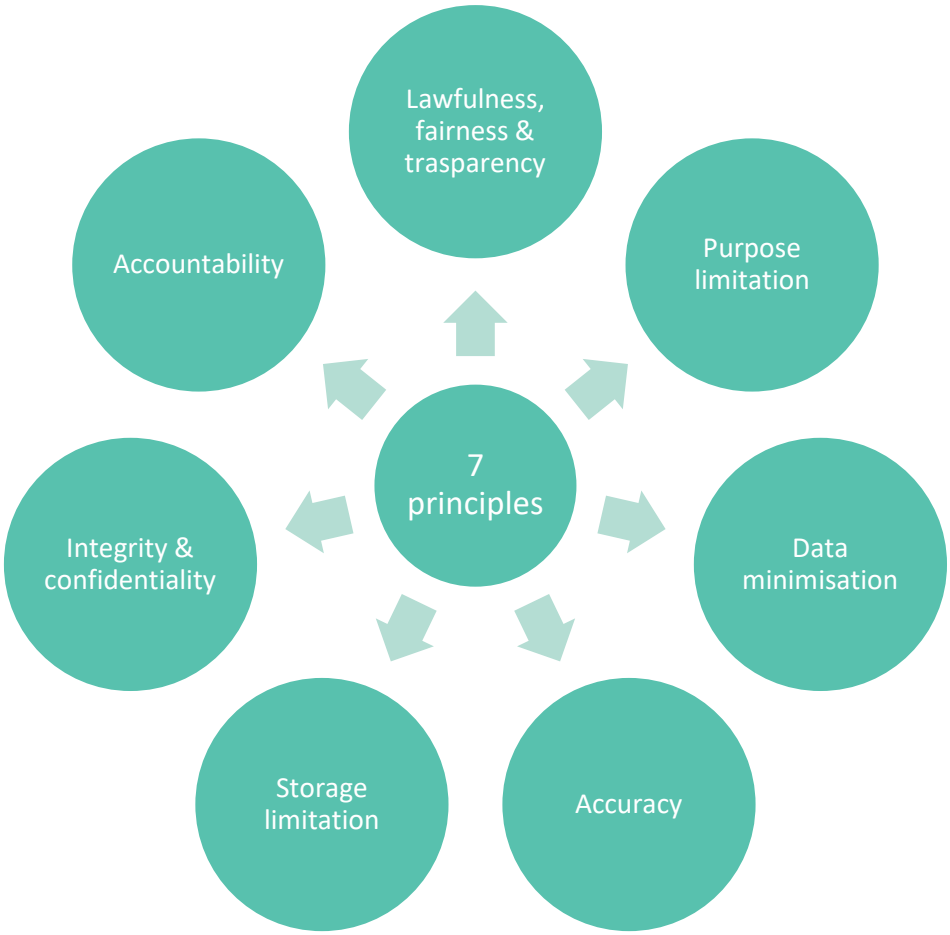

Figure 3: Overview of data protection principles

- **Lawfulness, fairness & transparency**
  - personal data must be processed lawfully, fairly, and in a transparent manner in relation to the data subject. The latter "transparency" requirement supplements what already existed in the Data Protection Directive.
- **Purpose limitation**
  - personal data must be collected for specified, explicit and legitimate purposes; and must not be further processed in a way incompatible with those purposes
- **Data minimisation**
  - personal data must be adequate, relevant and limited to what is necessary in relation to the purposes for which they are processed. Also, the period for which the data are stored should be limited to a strict minimum. Finally, personal data should only be processed if the purpose of the processing cannot be fulfilled by other means
- **Accuracy**
  - personal data must be accurate and, where necessary, kept up-to-date; every reasonable step must be taken to ensure that inaccurate personal data, having regard to the purposes for which they are processed, are erased or rectified without delay
- **Storage limitation**
  - personal data must be kept in a form which permits identification of data subjects for no longer than is necessary for the purposes for which the personal data are processed. Personal data may be stored for longer periods insofar as the data will be processed solely

for archiving purposes in the public interest, or scientific and historical research purposes or statistical purposes in accordance with Article 83(1) and subject to implementation of appropriate technical and organisational measures

- **Integrity and confidentiality**
  - personal data must be processed in a manner that ensures appropriate security of the personal data, including protection against unauthorised or unlawful processing and against accidental loss, destruction or damage, using appropriate technical or organisational measures.
- **Data governance and accountability**: the principle of accountability, which essentially refers to the various obligations organisations will have to follow in order to demonstrate compliance with data protection requirements, is not a new concept but is now expressly included in Article 5(2) of the GDPR. Hence, with the adoption of the GDPR, it now plays a significant role in the EU privacy regulatory framework

One of the ways to demonstrate accountability is through adopting the '**Privacy by design**' measures. This entails that every component or software artefact must implement appropriate technical and organisational measures (e.g. pseudonymisation techniques) designed to implement the data protection principles (e.g. data minimisation). Said measures must be implemented in an effective way so as to integrate the necessary safeguards into the data processing in order to meet the requirements of the GDPR and to protect the rights of data subjects. Said obligations must be respected both at the time of the determination of the means for processing and at the time of the processing itself. Some elements to consider while implementing the measures are: (i) the state of the art; (ii) the cost of implementation; (iii) the nature, scope, context and purposes of the processing; and (iv) the risks of varying likelihood and severity for rights and freedoms of individuals posed by the processing.

The privacy by design requirement should be carefully assessed by both the developers and the pilots (when initially collecting and processing personal data). The pilots must consider the implementation of appropriate technical and organisational measures in order to comply with the privacy by design requirement. As for the solution as such, it is recommended to carefully consider the privacy by design requirements in order to protect privacy by embedding it into the design specifications. This is done in particular through the application of pseudonymisation and anonymisation techniques. More detailed aspects were addressed in the *WP1* and reported in the deliverables about the ethical issues and data management plan [10], [12]–[16].

6 Software Architecture

In this section the main components that constitute the iHelp platform will be detailed, implementing the requirements emerged in the deliverable *D2.1* [2].

A simple overview of the platform is shown in Figure 4 in which are visible four big blocks as a placeholder for the more detailed description below.

The **`Final user applications`** are the apps (either mobile or web) that are dedicated to the human actors of the system (i.e. **`HCP`**, **`Individual`** and **`ModelBuilder`**).

The **`Data Ingestion`** represents all those components that are able to process data derived from different sources, also external to the iHelp project, and that can handle data in order to store them, compliant with the common data model, into the centrral **`Data Storage`**.

Symmetrically, the Data Analysis block, abstracts from the AI components that elaborates the data present in the **`Data Storage`** for feeding the **`Final User Application.`**

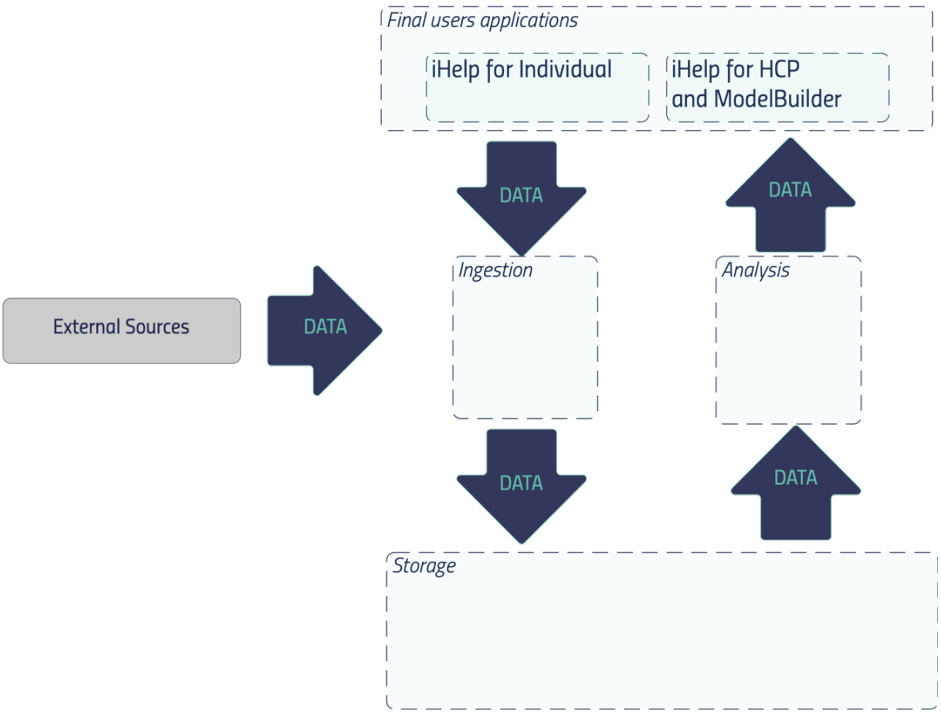

Figure 4: Simple Overview

6.1 Overall

In Figure 5 we can see a more detailed diagram of the overall architecture of the iHelp platform.

The formalism adopted in this diagram utilises some concepts derived from the UML v2.0 notation[17] (i.e. lollipops notation for the interfaces), but generally, a standard modelling language was not adopted for this kind of diagram, for better explaining also to the non-technical partners.

In particular, the shape of the software artefact mainly brings the concept of communication pattern, while a hexagon denotes the micro-services that communicate asynchronously, using a message broker.

A rectangle defines the components that offer or use an API for asynchronous communication.

The kind of interfaces used (identified with the "socket" glyph) or provided (identified with the "ball" glyph) are written in the attached label, for example, a **`REST API`** or a **`JDBC`**. For the component that offers a GUI we used the symbol for "providing an **`HTTP`** interface" (e.g.: the **`DSS Dashboard`**).

Other than the functional components, in the diagram some infrastructural components are also shown, needed for enabling the execution of the solution.

Generically, in microservice architectures, the UI usually connects with multiple microservices. If the microservices are finely grained, the client(s) may need to connect with lots of microservices, which becomes challenging. Also, the services, including their APIs, can evolve.

One possible way to solve these issues is to use API Gateway. API Gateway sits between the final user applications and the iHelp platform and acts as a facade. It can work as a reverse proxy, routing the client request to the appropriate service. It can also support the client request's fanning-out to multiple microservices and then return the aggregated responses to the client. It additionally supports essential cross-cutting concerns (such as authentication, load balancing, dependency resolution, data transformations and dynamic request dispatching can be handled in a convenient and generic way). In the iHelp reference architecture the block labelled **`API Gateway`**, will dispatch the external invocation to specific internal services; the consortium agreed with the adoption of Traefik tool [18].

In oreder to allow access only to authorised users, the iHelp solution will adopt an internal **`IdentityManger`**, based on the open source product Keycloak [19], that will take care of the resources access. iHelp can be configured to support OpenID Connect. OpenID Connect (OIDC) [20] is an authentication protocol that is an extension of OAuth 2.0. While OAuth 2.0 is only a framework for building authorization protocols and is mainly incomplete, OIDC is a full-fledged authentication and authorization protocol. OIDC also makes heavy use of the Json Web Token (JWT) set of standards. These standards define an identity token JSON format and ways to digitally sign and encrypt that data in a compact and web-friendly way.

About the internal communication, the possible choices were mainly split into two different branches, synchronous or asynchronous communication. Although synchronous communication is not a problem in a monolithic application, due to the simple context (i.e. a client sends a request and waits for a response), in a scenario where multiple services call each other in a non-trivial way, this can produce a waterfall of failing calls. Due to that, and for the sake of flexibility toward any further change or improvement of the requests flow, for the envisaged high demanding process an asynchronous communication pattern was designed.

Specifically, the constituent components of the data ingestion building block communicate through a message broker. Nevertheless, to decentralise the coordination, instead of a static orchestration, a choreography transaction was designed, specifically applied to the SAGA pattern for transactions. With this choice, each software artefact has a “brain” that can decide, listening to all the other components, if an action should be taken or not. This approach simplifies the possible refinement of the pipeline without impacting the single development of the components in the separated teams. The specific flow and the detailed interactions will be explained better in the scope of the WP3.

Available for all the components, but specifically used inside the pipeline ingestion, iHelp will also use the Apache Kafka [21], a message broker and stream processor which allows to publish, subscribe, archive and process streams of records in real time. It is designed to manage data streams from multiple sources by distributing them to multiple consumers.

Coupled with Kafka, iHelp consortium agreed with the adoption of Kubernetes [22]. Kubernetes (k8s) is an open source platform that automates Linux container operations. It eliminates many of the manual processes involved in deploying and scaling containerized applications and allows to manage host clusters running containers easily and efficiently.

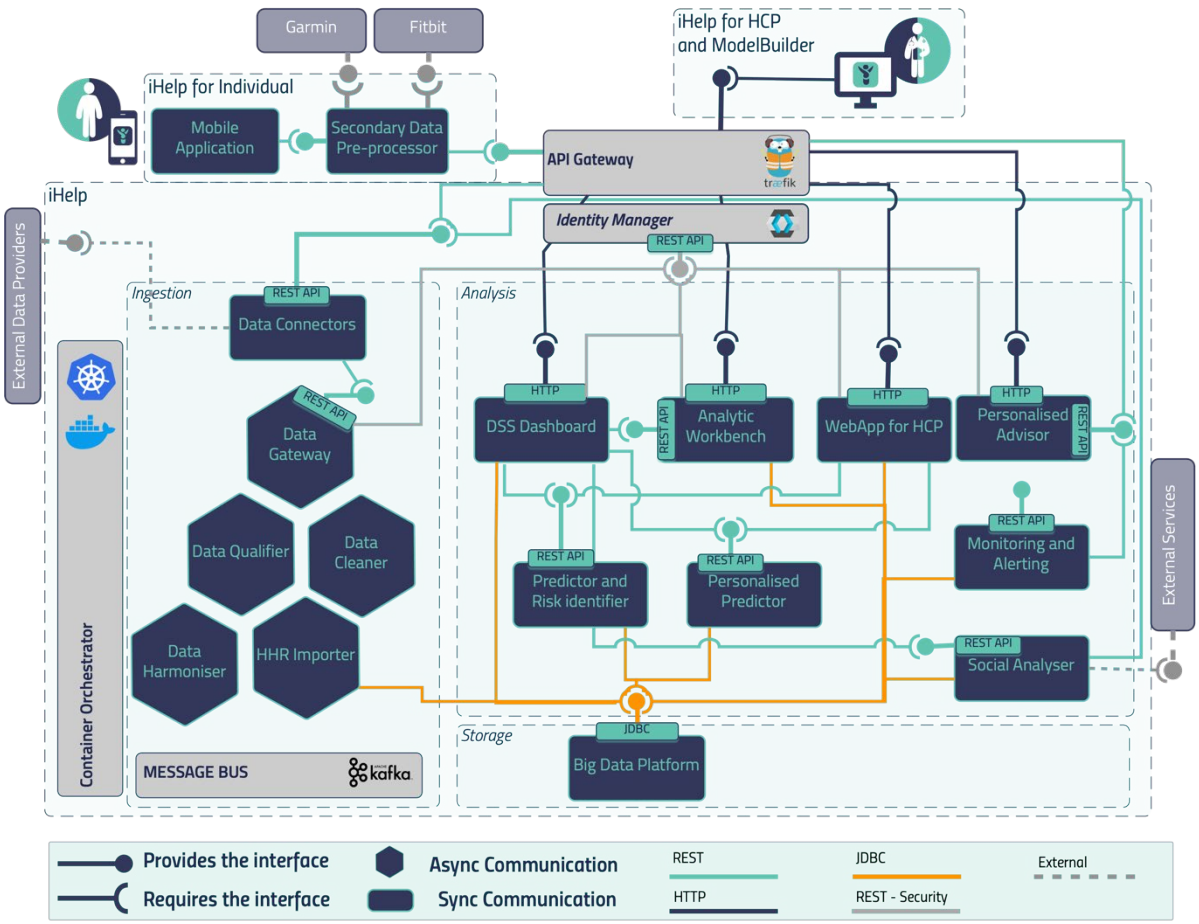

Figure 5: Overall Architecture

The goal and the main envisaged interactions of the components are described below.

6.2 Final User applications

6.2.1 DSS Dashboard

6.2.1.1 Objective of the software artefact

The Decision Support system will allow physicians and policy makers to analyse data and make decisions for both individuals and groups of patients. The DSS will be able to analyse data from a relational database through the execution of SQL statements and also to design more complex analytics through workflows. The DSS will provide visualization capabilities to present the analytics results. Different types of users will be considered from data scientists who will define the analytics and their visualization to nurses that may monitor a patient data.

6.2.1.2 Interactions with other components

Expected interfaces

The DSS interacts with databases and executes analytics on different types of data. The DSS needs JDBC and Python interfaces to interact with data stores and the different analytical models.

Offered Interfaces

The DSS will provide different interfaces for data scientist and physicians to log into the system, create federated queries and analytic workflows, and display the results from federated queries and analytics on different types of charts and dashboards.

Figure 6 shows an example of the login interface to access the different functionalities of the DSS.

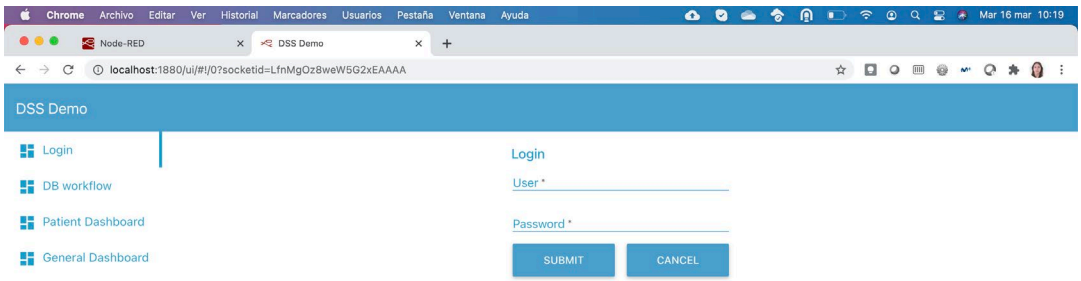

Figure 6: Login interface

Figure 7 shows an example of a dashboard that displays the rows of a table from a database selected from the *Database* drop-down menu. The first 50 rows stored in the selected table are displayed at the bottom of the figure.

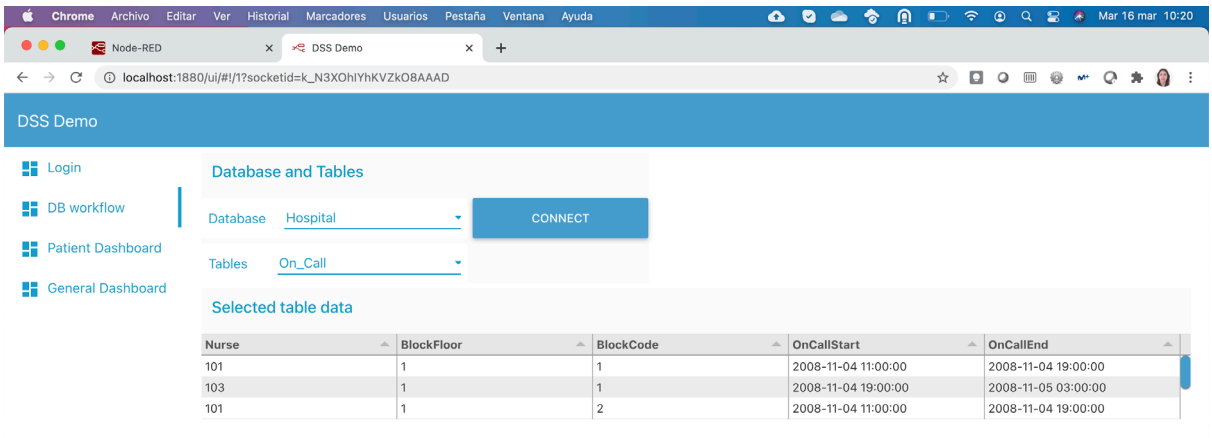

Figure 7: Database's data visualization

Figure 8 and Figure 9 depict two different dashboards where the data returned from SQL queries is visualized. Figure 8 shows the data of a given patient in text fields and in a bar chart. Figure 9 shows the results in a pie chart, horizontal bar chart, radar chart, and lines chart.

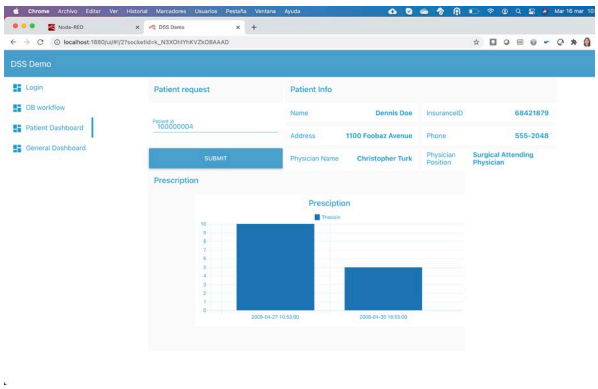

Figure 8: Individual patient data visualization

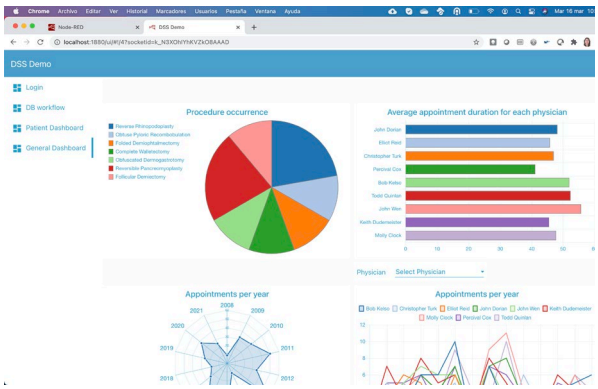

Figure 9: Different chart types

Figure 10 shows an example workflow where the data scientist creates federated queries and analytic workflows and defines the associated dashboards.

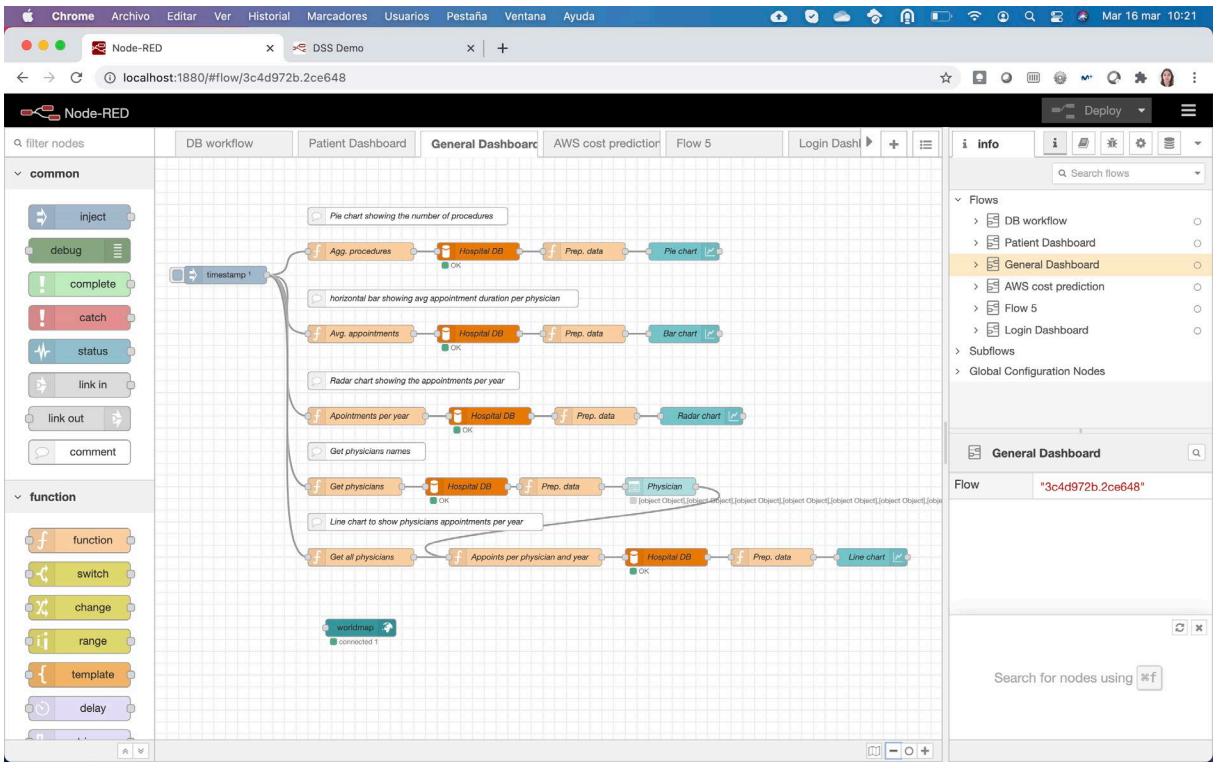

Figure 10: Workflow UI

Technological stack requirements

The DSS will use a core technology Node-Red[23], a flow-based development tool that allows creating programs by dragging and dropping different nodes and connecting the nodes to create the expected code flow. Python libraries for analytics like Pandas for data analysis and scikit-learn, Theano and TensorFlow for machine learning purposes may be included in the DSS.

6.2.2 WebApp for HCP

This component, conceptually separated from the **`DSSDashboard`**, will be concretely implemented in the same **`DSSDashboard`** component, but offering different views in order to allow the physicians to better follow the patients. This conceptual component at this time, is in the progress of definition in the scope of task *T2.4 – User centred design* and will be better explained and detailed in the related deliverable.

6.2.3 Mobile App

6.2.3.1 Objective of the software artefact

The Mobile App component is the main user interface for patients/citizens in the various iHelp pilots. As the name suggests, this component is a mobile application that is available for download from the Google Play Store[2] (for Android users) and the Apple App Store[3] (for iOS users). The application (Healthentia) is already available to download (see links in footnotes).

The Healthentia application is an app that can be used in various trial/study configurations, so the exact functionality of the mobile app depends on the user who is logged into the application, and the trial/study

[2] https://play.google.com/store/apps/details?id=com.innovationsprint.healthentia&hl=en&gl=US
[3] https://apps.apple.com/nl/app/healthentia/id1491757656?l=en#?platform=iphone

in which he/she participates. The type of functionalities offered by the mobile app in iHelp can broadly be described as:

- Monitor the user's lifestyle and symptoms through connected sensors, self-reported events and answering questionnaires from within the app.
- Communicate with the patient's/citizen's healthcare professional, either through messages, teleconferencing, or by receiving notifications directly from the professional.
- Provide virtual coaching to the patient/citizen – offering health education, advice, through visualisations of measured data, simple feedback messages or in-depth conversations with a digital virtual coach.

The main objective for this component is to design and develop the functionalities needed to support up to 5 different application configurations linked to the 5 iHelp Pilots. It may be possible to use the same application configuration for 2 different pilots, or it may be necessary to have 5 different configurations – this will depend on the details of the pilots as they are being specified. Details of the exact design and development progress of the mobile application may be found in *D5.6* and *D5.7*.

As background technology, the existing Healthentia application supports a number of functionalities mostly related to monitoring patients/citizens, as listed here:

- Questionnaires – A set of questions with various supported types (e.g. multiple choice, Visual Analogue Scales, Likert Scale) that are triggered to be answered by the user at defined time intervals.
- Physical Activity Monitoring – By connecting an external tracking device, the application can monitor the user's daily levels of physical activity.
- Sleep Monitoring – By connecting external tracking devices that support tracking sleep, the application can monitor the user's daily sleep patterns.
- Liquid Intake – Through self-reporting, the application allows the user to measure their own liquid intake during the day.
- Symptom and Event Reporting – Through self-reporting, the application allows the user to report any number of predefined events or health related symptoms.

In order to satisfy the core objectives of the Mobile App component in supporting the various iHelp Pilots, the focus will be on adding various types of functionality related to providing feedback and coaching to the end-users, as further defined in Deliverable 2.1.

6.2.3.2 Interactions with other components

The iHelp Mobile App component is a series of configurations of the Healthentia Mobile App which is embedded into the Healthentia platform. From a technical perspective, the mobile app is a client to the Healthentia platform, which means that it is the only component it directly communicates to. In Figure 11 below, we depict the iHelp Mobile App and its direct interactions with the Healthentia ***Data Service*** and the Healthentia ***Virtual Coach Service***. The connections with the Data Service are mostly already in place, and are used to support the monitoring functionalities of the application (e.g. retrieving data from external tracking devices, or storing questionnaire data). The connections with the Virtual Coach Service are to be implemented in the iHelp project, and support the delivery of feedback and personalized healthcare from other iHelp components (e.g. the - Personalised Advisor see §6.4.4).

In summary, there are three types of interactions with the mobile app:

- The end-user that will use the Mobile App as their primary GUI.
- The data services that enable collection of data through the app (see §6.3.7 - Secondary Data Pre-processor).
- The coaching services that enable the delivery of personalized feedback and coaching (see §6.4.4 - Personalised Advisor).

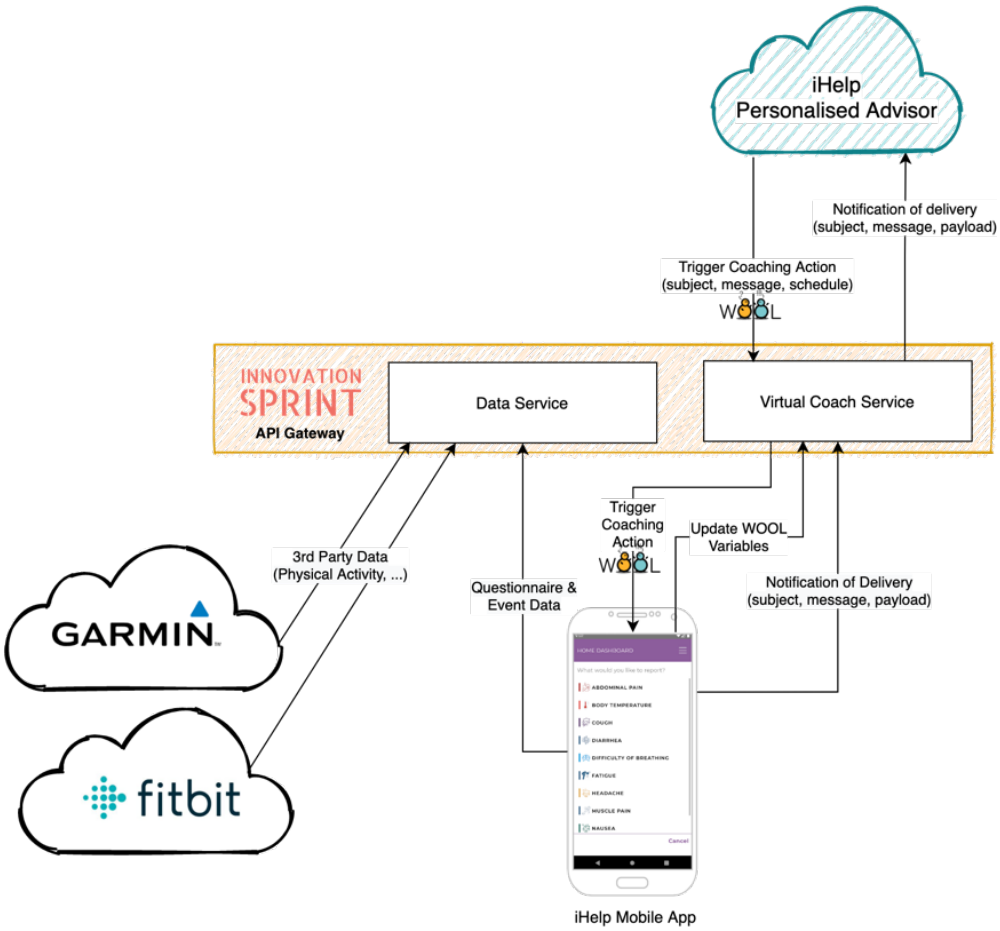

Figure 11: iHelp Mobile App and its direct interactions with other components. The mobile app collects data (Reports and Questionnaires) which is stored in the Data Service. At the same time, the app visualizes data from 3rd party sensor providers (e.g. Garmin, Fitbit) through the same Data Service. The digital virtual coach is powered by a separate service (Virtual Coach Service), which in turn is controlled through the iHelp Personalised Advisor.

6.3 Data Ingestion

6.3.1 HHR Importer

6.3.1.1 Objective of the software artefact

The objective of the HHR Importer is to store data received under the HHR format into the Big Data Platform component of the iHelp platform. During the ingestion process, raw (primary) and aggregated (secondary) data are mapped and transformed to the common HHR model. At the final stage of the data ingestion pipeline, the HHR importer takes the transformed data and persistently stores them in the datastore. The latter has defined a relational schema that is compatible with the E-R conceptual model of HHR and this software artefact will implement the following two functionalities:

- Transform the HHR entities to various data items that will be stored into data tables of the Big Data Platform.
- Establish the data connectivity with the datastore and perform all relevant operations in order to persistently store the produced data items.

The functionality of this component will be packaged as a Java archive and will be either deployed as a static standalone microservice, or it will be provided as a *function* to be dynamically deployed via the serverless platform of the project.

6.3.1.2 Interactions with other components

This component interacts with the other software artefacts involved in the data ingestion pipeline and the Big Data platform. It will provide an interface that will allow any other software artefact to push any HHR data compliant with the FHIR format. As a result, the component will be waiting for any other component to send data that need to be stored in the Big Data Platform. No GUI will be provided for this component. At this phase of the project, it is not clear whether the interface will be exposed as a REST API that will allow other components to invoke it, or a much simpler interface will be provided to allow the serverless platform to pass meta-information parameters, such as data queue specific information. In case of the latter option, this information will be used internally to connect to a Kafka queue and data will be retrieved by the specified topic.

The HHR Importer does not produce any output data. The result of its invocation will be a transition of the state of the Big Data Platform. This transition is the result of a data modification operation (i.e. data insertion) that is successfully committed in the database layer. The HHR imported will interact with the Big Data Platform either via the standard JDBC connectivity mechanism, or using one of its provided data connectors.

6.3.2 Data Gateway

6.3.2.1 Objective of the software artefact

The Data Gateway microservice exposes a REST API through which the iHelp connectors can send their data to the iHelp platform. The communication is done through HTTPS and the Data Gateway performs two main actions:

- Message authentication, where the Data Gateway checks the provided security keys in the message against the ones defined for a specific connector.

- Message transformation, where the message is sent to the Message Streaming Platform (Kafka) in the appropriate format.

6.3.2.2 Interactions with other components

The Data Gateway microservice is consumed by the iHelp Data connectors, that is, the connectors send the primary and secondary data that they gather to the Data Gateway, in order to initiate the ingestion pipeline. The Data Gateway transforms the REST request received to a Kafka message and relays the data to the next component without making any other transformations.

6.3.3 Data Harmoniser

6.3.3.1 Objective of the software artefact

Data Harmoniser will be utilized as an integrated microservice in the overall iHelp platform. It seeks to support and harmonise data coming from heterogeneous sources into a common format. To this end, its main aim is to provide annotated / correlated health data & harmonize them with the widely used HL7 FHIR format. The microservice will use its own internal sub-mechanism in order to correlate data resources with HL7 FHIR resources to be compliant with the HHR model that will be defined in the scope of the task 3.1.

The Primary Mapper sub-component will enable the mapping of primary data, that is the clinical data from the electronic health records, into the common data format that will be used in the iHelp platform (HHR). The component will provide the necessary transformation functions that are required to map the primary data to the holistic health records stored in the iHelp platform.

The Secondary Mapper sub-component will enable the mapping of secondary data (e.g. from mobile, wearable and social-media platforms) into the data format (schema, model) used in the iHelp platform (HHR). The component will provide necessary transformation functions that are required to map the secondary data from heterogeneous sources to the holistic health records stored in the iHelp platform. The Secondary Mapper will be an integral part of the iHelp platform as it supports the enrichment of typical health records with the secondary (e.g. lifestyle, social etc) data of the individuals.

6.3.3.2 Interactions with other components

The Data Harmoniser microservice will be utilized dynamically whenever it is selected, or when it is considered necessary and mandatory by the dataset provider. This microservice will be integrated with the provided message bus mechanism and it will consume and produce corresponding messages to this. The message to be consumed should be cleaned data, which will be further harmonised and transformed, hence an annotated, transformed and HL7 FHIR format aligned message will be the output of this specific microservice. To this end, the Data Harmoniser microservice has been determined to integrate closely with the Data Cleaner microservice.

In deeper details, the Data Harmoniser microservice incorporates the use of three (3) integrated sub-mechanisms, as also presented in the below figure (Figure 12).

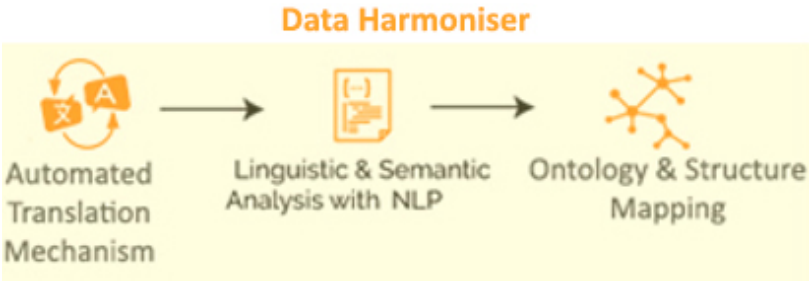

Figure 12: Data Harmoniser

- Automated Translation Mechanism: State-of-the-art techniques and approaches from the field of Neural Machine Translation (NMT), such as Transformers[24] and Recurrent Neural Networks (RNNs) based on the seq2seq architecture[25], will be utilized for handling data from different languages in order to provide multilingual and language-independent solutions.
- Linguistic & Semantic Analysis with NLP: In this second sub mechanism translated data will be analysed, transformed, and annotated with appropriate metadata and controlled vocabularies will be identified and designed through the utilization of Semantic Web technologies coupled and enhanced by the utilization of NLP techniques, such as Named Entity Recognition (NER), Part-of-Speech Tagging etc.
- Ontology & Structure Mapping: This sub mechanism seeks to interlink annotated and transformed data with proper identified ontologies from the healthcare domain and in the HL7 FHIR format, and to correlate datasets among them. Successful annotation, transformation and mapping of data and corresponding ontologies in terms of semantic and syntactic interoperability of data is one of the key elements of the Data Harmoniser mechanism.

6.3.4 Data Cleaner

6.3.4.1 Objective of the software artefact

Data Cleaner will be utilized as an integrated microservice in the overall iHelp project and its main objective is to deliver the software implementation that will provide the assurance that the provided data coming from several heterogeneous data sources will be clean and complete, to the extent possible. This microservice will be designed to minimize and filter the non-important data, thus improving the data quality and importance.

6.3.4.2 Interactions with other components

The Data Cleaner microservice will be utilized for every new incoming dataset in the platform since it seeks to detect and correct (or remove) inaccurate or corrupted data from the datasets. The input to this microservice will be provided by the shared message bus which will be utilized in the scopes of the iHelp project (e.g. Kafka). The topic from which data in the format of messages will be consumed can be set either dynamically, as a parameter whenever the microservice is called, or statically, based on an agreement in case that it is preferred. The input message will be the whole dataset in order to allow microservice to provide all the necessary cleaning actions towards the aim to produce consistent and cleaned data and datasets.

On top of this, the Data Cleaner service will be comprised of three (3) discrete, but integrated with each other, sub mechanisms which are listed below.

- Data Validation: This service will ensure that the other iHelp microservices will operate on clean, correct and useful data. Therefore, the Data Validation service will perform data validation of the incoming information data with the purpose of identifying errors based on conformance to a specific set of constraints.
- Data Cleaning: Entails the main sub mechanism of the Data Cleaner microservice. Its main goal is to correct or remove all the data elements for which validation errors were raised, considering missing, irregular, unnecessary, and inconsistent data. Thus, the Data Cleaning sub mechanism will perform the necessary corrections or removal of errors identified by the Data Validation Service.
- Data Verification: The main objective of this sub mechanism is to check the data elements of a dataset for accuracy and inconsistencies after the steps of data validation and cleaning are performed. To this end, it will ensure that all the corrective actions performed by the Data Cleaning service will be executed in compliance with the data models design of the iHelp platform. To this end, this service seeks to ensure that data will accurately be corrected or completed, and that the dataset will eventually be error free.

6.3.5 Data Qualifier

6.3.5.1 Objective of the software artefact

The scope of the Data Qualifier microservice is to automatically categorize both known and unknown data sources to specific “levels of trustfulness” (i.e. data reliability, provided data type, data source availability) according to a given threshold, resulting into the adaptive selection of all the available reliable data sources in order to be connected into the iHelp project. To this end, this microservice seeks to provide a predictive selection mechanism for achieving data sources reliability during runtime and for providing the decision whether a connected data source will be considered as reliable or not as a result of the execution.

6.3.5.2 Interactions with other components

The Data Qualifier microservice will be utilized for every new registered data source and every incoming dataset in the platform because its goal is to enhance and adapt the selection of reliable sources. This specific microservice will also be integrated with the message bus mechanism provided in the scope of iHelp project. To this end, the whole dataset and information about the data source should be the input to this microservice in order to provide the reliability levels of the connected data sources into a message (i.e. String) format, which will be annotated to the dataset as metadata. This specific microservice has been identified to integrate with the Data Cleaner microservice in a dynamically modifiable workflow, since these two microservices may be utilized in different order depending on the type of data, sources, and the preferences of the data provider. The latter indicates that in terms of safeguarding the good quality of the incoming data and the reliability of the data sources, there may be the need in one scenario to utilize the Data Cleaner before the Data Qualifier, in order to provide cleaned data before the qualification of the dataset and the data, while in another scenario it is preferred to have at first the qualification of the data source and the dataset and then to perform the Data Cleaner microservice.

6.3.6 Data Connectors

6.3.6.1 Objective of the software artefact

The objective of this type of software artefact is to allow connection with the data providers in order to retrieve data from the source and send it to the beginning of the data ingestion pipeline. There will be

different types of data connectors, each one of those will provide different means for connectivity. At this phase of the project, we envision a connector that can get data stored in static files in an FTP server, another type of connector that can execute a query statement in a JDBC compatible database and an additional one that will provide a REST API to allow for secondary data to be pushed. It will be further investigated during the progress of the project if other types of primary data connectors will be required.

Each of the connectors will be released as Java archives and will be available to the *Data Gateway,* so that the latter can make use of the appropriate one to fetch data from the source and start ingesting it to the data pipeline.

6.3.6.2 Interactions with other components

This set of software artefacts interact with the *Data Gateway* and the external data sources, that can be of different types. According to its type, it can read data autonomously or allow for external components to periodically push data to the connector. For instance, in case of a static or periodic data ingestion (batch ingestion) the component will read data from the external source, using the corresponding connector. In case of the secondary data ingestion, the corresponding component (i.e. the Secondary Data Pre-processor) will expose a REST API to allow the secondary data collector (i.e. a DataConnetor) to periodically retrieve pre-processed secondary data and send those in a predefined format that needs to be further defined.

Regarding the input data, in case of a batch ingestion processing the input data is the meta-information regarding the parameters that the corresponding connectors need to know in order to establish the data connectivity (i.e. URI of the ftp server and the path to read the data). In case of the secondary data ingestion, the input data will be the aggregated data that the secondary data connector will send, in a format that will be defined in the forthcoming period of the project.

All primary data connectors will give as output byte buffers that are generic and enclose the data received, along with the type of the dataset. These connectors are dataset agnostic and therefore they treat all received information as byte buffers, as their purpose is only to establish the data connectivity, and not to perform any data processing. At a latter phase, it might be decided that the output of the primary data connectors can additionally send the schema definition of the dataset in the output.

6.3.7 Secondary Data Pre-processor

6.3.7.1 Objective of the software artefact

The iHelp secondary data is about dynamic data arriving online from sensors or questionnaires, obtained outside a clinical setting, attempting to describe various aspects of everyday life. These aspects are expected to be physiological (exercise, nutrition, body signals), psychological, social and environmental, to be finalised at the requirements collection phase of T2.1. The scope of the Secondary Data Pre-processor component is to collect this dynamic everyday-life raw data and process it into higher-level information.

Raw secondary data are objective measurements from IoT devices, or subjecting reports and self-assessments collected using the mobile application (see section 6.2.3). Devices will be integrated utilizing either APIs or SDKs from their manufacturers for automated measurements. In the API case (most probable), the integration will be carried out at the Secondary Data Pre-processor component, by interfacing with the device manufacturers' servers. In the rarer case where an SDK is provided by the device manufacturer, the integration functionality of the component has to be moved into the mobile application

one. Questionnaires will be built in the mobile app. The Secondary Data Pre-processor component will be collecting all the patient data from the mobile application.

Raw secondary data are also to be processed to yield higher-level information including relationships, interactions, experiences, habits and behaviours that can influence people's health, design of preventative measures or risk of developing (chronic) conditions. To do so signal processing and/or machine learning algorithms will be employed to understand secondary data semantics. These algorithms are also implemented in the Secondary Data Pre-processor component.

6.3.7.2 Interactions with other components

The Secondary Data Pre-processor component has two types of data input to autonomously get the raw patient secondary data:

- The interaction with the mobile application is done by offering an API to be used by the mobile application to push the available data.
- The interaction with the device manufactures' servers is done via their API. The exact mode of interaction differs across device manufacturers, e.g. Fitbit provides endpoints to be called ad hoc to pull data, while Garmin allows the component to register itself and push new data, whenever available, to the server.

The Secondary Data Pre-processor component serves the raw and the processed higher-level secondary data to the iHelp platform by interfacing with the Data Connectors component. To this aim it offers an API to receive requests for data originating from the Data Connectors component.

6.4 Data analysis

6.4.1 Analytic Workbench

6.4.1.1 Objective of the software artefact

The Analytic Workbench component in the iHelp platform enables the "ingestion" of multiple data analytics functions / tasks to enable the development of adaptive preventive and intervention models for different risks and contributing factors associated with Pancreatic Cancer. The Analytic Workbench component provides a complete information processing framework that will incorporate declarative methods for the specification of targeted analysis tasks, as well as declarative analytics to include predictive data services (especially from generic methods such as time series modelling).

The objectives of this component are:

- To define a workbench for analytic functions that can extract meaningful information from integrated big-datasets
- To facilitate scalable real-time big data management and analytics (including outcomes of analytics such as models) through an analytics workbench that supports processing across multiple data analytic functions

6.4.1.2 Interactions with other components

The analytic workbench facilitates openness and usability by allowing any actor (e.g., researcher, healthcare professional, data provider, etc) to develop on-demand adaptive learning models for different risks based on the application of advanced AI analytic techniques.

The analytic workbench component uses the following sub-components to materialize the desired objectives:

- Apache Druid is an open-source, distributed data store designed to quickly ingest massive quantities of event data, and provide low-latency queries on top of the data. Druid streams data from message buses such as Kafka and supports the design of workflows that provide fast ad-hoc analytics, instant data visibility and concurrency to power UIs where an interactive, consistent user experience is desired
- Apache Superset is an open-source software cloud-native application for data exploration and data visualization capable of handling data at petabyte scale (big data). Superset is used as a fast, lightweight and intuitive solution that makes it easy for users of all skill sets to explore and visualize their data, from simple line charts to highly detailed geospatial charts
- Apache Kafka is a framework implementation of a software bus using stream-processing. Apache Kafka is used as a distributed event streaming platform that is used to support high-performance data pipelines, streaming analytics and data integrations

The relationship of Analytic Workbench with other components in the iHelp platform is shown in the following diagram (Figure 13):

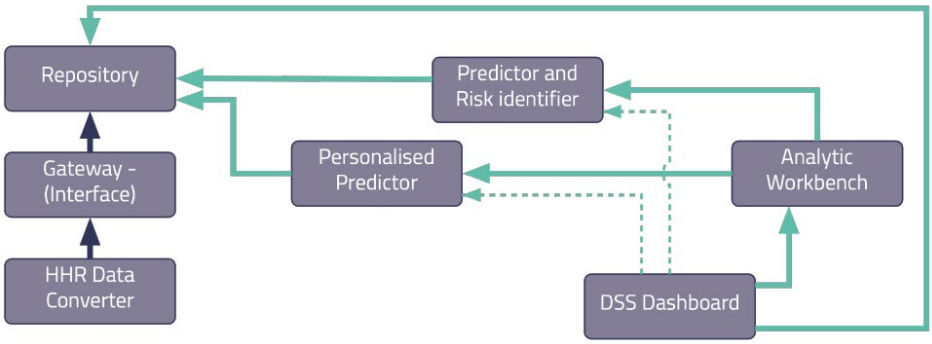

Figure 13: Interactions of Analytic Workbench component with other iHelp components

Based on the interactions depicted in Figure 13, the Analytic Workbench will provide the necessary infrastructure (software libraries, scripts, orchestration) that allows the deployment and execution of analytic applications/services (related to risk identification, predictions etc) in the iHelp platform. In this respect, the users (i.e. developers of analytic services in the iHelp platform) will interact with the Analytic Workbench to deploy and execute their services. To support the user engagement, the Analytic Workbench will provide an administrative dashboard interface that will also allow the monitoring of different analytic services that are using the workbench.

6.4.2 Personalised Predictor

6.4.2.1 Objective of the software artefact

The Personalised Predictor is responsible to deliver the necessary toolset that enables the usage of advanced AI learning techniques towards the development of a personalized health model. This tool will assist in the identification of specific diseases, along with their contributing factors and it will support the scalability and security needs of personalized health data. For the implementation of the Personalized Predictor, Deep Neural Networks will be trained, with the goal of modeling associations between the patient data and the available disease related data (e.g.: disease onset, exacerbation, adverse effects of therapy, patient compliance, patient risky behaviours, patient lifestyle). Hidden patterns could be discovered and the extracted knowledge would help the study of the disease.

6.4.2.2 Interactions with other components

The Personalised Predictor will use the HHR records stored in the iHelp repository in order to analyse the dataset and train the health models that will be developed in accordance with the user needs. The output will be the trained health models that will be used to identify associations and perform predictions about the disease. The Personalised Predictor will make use of the infrastructure provided by the Analytics Workbench and potentially expose an API to the DSS Dashboard depending on the project's needs.

6.4.3 Predictor and Risk identifier

6.4.3.1 Objective of the software artefact

The goals of the Predictor and Risk Identifier component are to predict Pancreatic Cancer risks in individuals, as well as perform assessment on the identified risks. The P.R.I component will develop feature extraction models in order to identify key attributes in the patients' health records and prediction models that will take into account the full range of attributes in the HHR records and the output of the feature extraction models, so that in the end it will identify the risks and their evolution in time.

6.4.3.2 Interactions with other components

The Predictor and Risk Identifier uses the HHR records for its analysis and training of the necessary models. These operations can be triggered by an admin user or be schedule-based, considering that the HHR records will be enriched in the course of time. The P.R.I. component also performs predictions and risk assessments, so an API will be provided for a user to enter the proper parameters and receive the relevant results, potentially through the DSS Dashboard.

6.4.4 Personalised Advisor

6.4.4.1 Objective of the software artefact

The Personalised Advisor module is a microservice that makes use of a small set of different microservices (i.e. a microservice cluster). The main objective of the Personalised Advisor module is to control the advice messages and dialogues that are presented to the end-users of the Mobile App (see §6.2.3). The component facilitates this in two different ways:

- Acting as a gateway service for other components to communicate to the mobile app.
- Acting as a smart component that makes decisions on when and what to say to the mobile app user.

6.4.4.2 Interactions with other components

In Figure 14 below we show the Personalised Advisor module and all related/surrounding components. The three “cloud” modules depicted are:

- **iHelp Personalised Advisor** – the core microservice of the cluster – this component serves 2 main purposes (1) to pass-through recommendations/advice generated by other iHelp components to the Mobile App and end-user, and (2) to make decisions on which advice to provide to the user at which point in time.
- **WOOL Repository Service** – A service that stores WOOL Projects, containing collections of WOOL Dialogues, which are handcrafted scripted definitions of coaching dialogues (to be developed as iHelp service and released as an open-source WOOL module).
- **WOOL Variable Store Service** – A service that stores WOOL Variable values for specific end-users. WOOL Variables are simple, primitively typed variables that can be used within the definition of WOOL Dialogues (to be developed as iHelp service and released as open-source WOOL module).

Then, as stated in the objectives of this module, the Personalised Advisor communicates with the Healthentia Edge Component that is depicted by the following set of components:

- **Mobile App** – the main user interface for the primary end-users of the iHelp platform (patients or citizens) – see §6.2.3
- **Healthentia Web Portal** – a web-based user interface for clinicians, or other (healthcare) professionals that allows configuration of studies, managing users, viewing detailed data visualisations, and scheduling coaching actions to primary end-users.
- **Healthentia Data Service** – an internal back-end platform component dealing with capturing and providing access to measured data (e.g., sensor data, questionnaires, events, or log data). For additional details see §6.3.7 - Secondary Data Pre-processor.
- **Healthentia Virtual Coach Service** – an internal back-end platform component that controls the delivery of “coaching actions” (e.g., feedback messages, or dialogues) to users of the mobile application, and possibly professional users of the Web Portal.

These four components of the Healthentia Edge Component are depicted in Figure 5 in the centre/bottom of the high-level architecture overview.

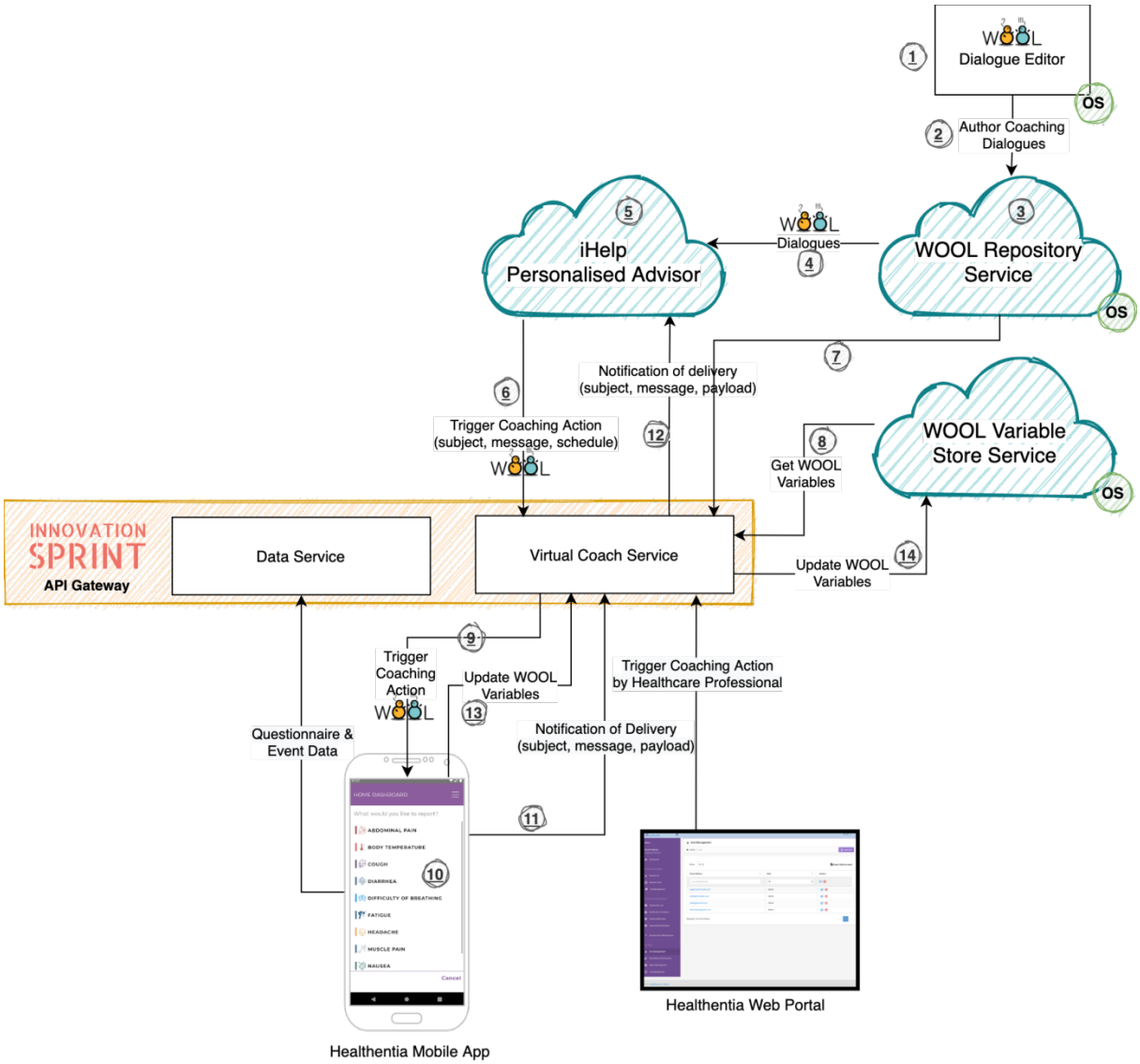

Figure 14: Personalised Advisor module and all related/surrounding components. The iHelp Personalised Advisor module is the core component here, that uses the various WOOL Services to send notifications/messages/dialogue to the Mobile App through the Virtual Coach Service.

Finally, Figure 14 shows, in the top right, one additional component:

- **WOOL Dialogue Editor** – A GUI Tool that allows non-technical domain experts to author coaching dialogues using the WOOL Dialogue language. A working version of the dialogue editor exists as open source, and improvements will be made to match the iHelp requirements (see §6.4.4.3).

The delivery of personalised healthcare and real-time feedback in iHelp is an interplay between the **`Personalised Advisor`** (a back-end service component) and the **`Mobile App`** (the main user interface component for the end-user). The Personalised Advisor is a smart service to be developed in the iHelp project, while the Mobile App will be based on the existing Healthentia Mobile application to be extended to support the various iHelp use cases.

When we are talking about personalised healthcare delivery, we are talking about three key concepts: (1) visualising measured data, (2) providing simple feedback messages, and (3) engaging the user in a dialogue

with the digital virtual coach (see also *D2.1*[2]). As such, these three coaching actions ("*show data visualisation*", "*give feedback message*", and "*trigger coaching dialogue*") are the three possible outputs of the **`Personalised Advisor`** component. In short: the personalised advisor decides for every iHelp user *when* to deliver *which* one or more of the coaching actions.

The process of delivering such coaching actions is proposed to work in the following way. We explain the process step-by-step, following the encircled numbers in Figure 14.

1. A health expert authors a coaching dialogue using the WOOL Dialogue specification language, using the WOOL Dialogue Editor (a graphical user interface tool – Figure 15). In our example, the health expert authors an interactive dialogue that lets the user agree on a physical activity goal in cooperation with the iHelp virtual coach.
2. The health expert is logged in with their iHelp credentials, identifying them as the owner of a specific WOOL Project (e.g., related to Pilot #1). Logged in with the proper authentication, the health expert can store their WOOL dialogue in the WOOL Repository Service.
3. The WOOL Repository service contains a number of WOOL Projects (collections of WOOL dialogues), and knows which users have read/write access to those projects.
4. The iHelp Personalised Advisor can retrieve the latest set of WOOL Dialogues relevant to a specific pilot.
5. The Personalised Advisor makes a decision that a subject should set a physical activity coaching goal now.
6. The trigger to start the "physical_activity_goal" dialogue for the specified user as soon as possible (now), is sent to the Healthentia Edge Component.
7. The Healthentia Edge Component retrieves the latest Dialogue definition for the "physical_activity_goal" dialogue from the WOOL Repository Service.
8. The Healthentia Edge Component retrieves the latest set of WOOL Variables needed to execute the current dialogue (e.g., $userAge, $userGender, etc..).
9. The Healthentia Edge Component sends a trigger to execute this dialogue to the subject's mobile application.
10. When the user sees the notification that the virtual coach has something to say, the mobile application will execute the dialogue – engaging with the user in the dialogue about physical activity goal setting.
11. When the dialogue is started (or finished), a notification of delivery is sent back to the Healthentia Edge Component.
12. The notification of delivery is passed back to the source of the coaching action trigger (i.e., the Personalised Advisor). In this case, the personalised advisor can know that its triggered advice has reached the end-user.
13. After the user has finished interacting with the dialogue, any updated or newly set WOOL Variables are passed back to the Healthentia Edge Component. In this example, the WOOL dialogue led to a new variable being set: $userActivityGoal = 10000".
14. The WOOL Variables that need updating are sent to the WOOL Variable Store Service for storage.

6.4.4.3 WOOL Dialogue Editor

WOOL is an Open Source (MIT Licensed) platform for authoring and executing scripted dialogues that was designed and developed specifically for the purpose of integrated personalized health coaching dialogue into eHealth applications. WOOL was developed in the context of a previous EU H2020 project called "Council of Coaches" and is currently maintained by partners of that project Roessingh Research and Development and Innovation Sprint. WOOL is and for ever will be fully open source and has support for continuous development from various EU R&D projects for the coming years. For more information see www.woolplatform.eu.

The WOOL Platform will be used in iHelp to support advice delivery in the form of conversations between a virtual coach and the end-user. As part of the Personalised Advisor workflow, the WOOL Editor is the tool that allows professionals to design their own scripted dialogue (see Figure 15 below).

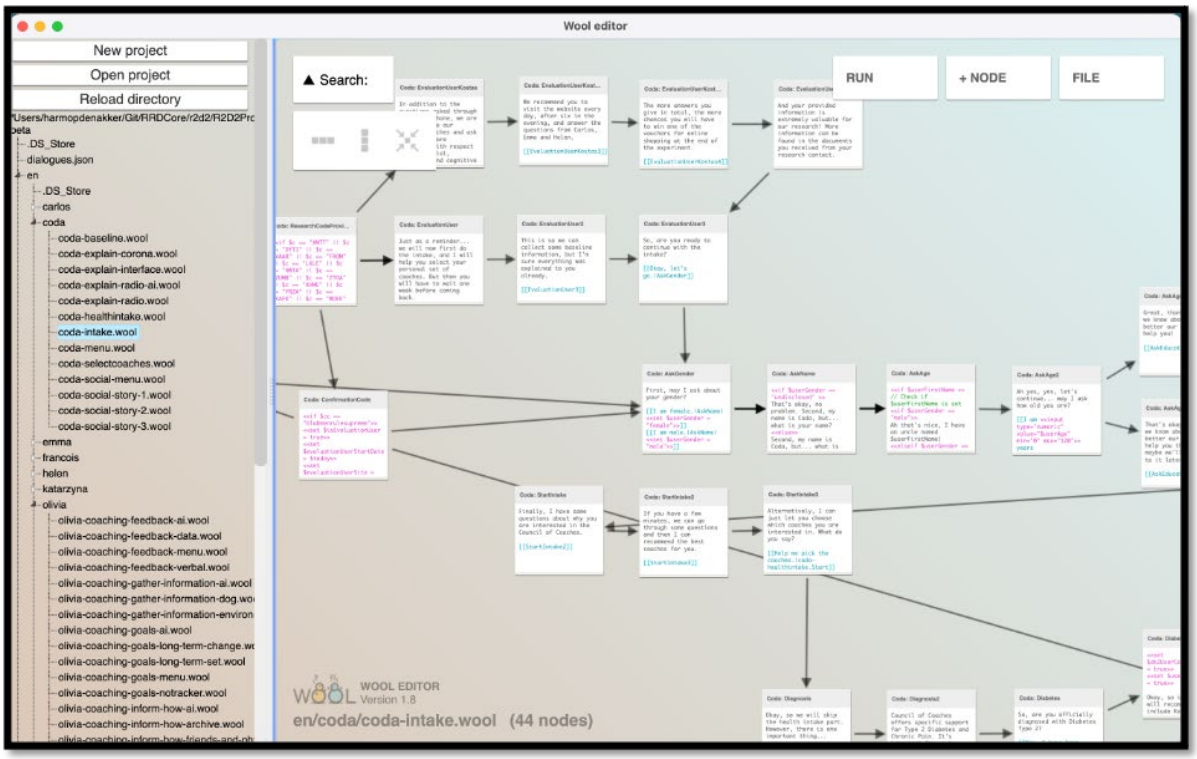

Figure 15: Screenshot of the WOOL Editor Tool, a work in progress graphical user interface for authoring WOOL Dialogues. The screenshot shows the structure of a collection of WOOL dialogues (left), and the main Dialogue Editing window on the right. Every box in the dialogue editor window is a step in the conversation, which are linked to other steps as shown by the arrows.

The WOOL Editor will be updated to reflect the needs arising from the various iHelp use cases.

6.4.5 Social Analyser

6.4.5.1 Objective of the software artefact

The Social Analyser component is responsible for the analysis of discussion taking place on the social media platforms in order to provide insight into the lifestyle trends, mental models, emotions, social interactions and societal influences on the individuals who either have developed the Cancer or are in the early stage of risk identification. This type of information about social aspects is important to analyse/evaluate the impact of the risks, the personalised healthcare and/or the targeted recommendations by the clinical experts.

The Social Analyser component will use Complex Event Processing (CEP) techniques, in addition to the state-of-the art Natural Language Processing (NLP), Sentiment Network Analysis (SNA) and AI technologies to provide a complete toolkit for clinicians and policy makers; allowing them to collect, process and visualize Cancer related information exchanged on the social media platforms. In this respect, the objectives of the Social Analyser component (as described in the iHelp DoA are) are:

- To develop mechanisms for real-time analysis of social data and contribution to policy making activities

- To define mechanisms for efficient monitoring, alerts and feedback gathering regarding the personalised recommendations and user-centric healthcare solutions developed in the project
- To evaluate the benefits of AI and personalised healthcare approach developed in the project

In addition to the above, the following objectives are defined for the Social Analyser component:

- To extract iHelp relevant information (e.g., discussions on risk identification, targeted recommendations etc) from social media discussions and present it to policy makers in an easy to interpret and easy to understand way
- To integrate the analysis of social media data with relevant HHRs in the iHelp platform
- To present the social media analysis in such a way that policy makers will be able to analyse the general healthcare trends as well as impact of specific recommendations/policies

The outcome of the Social Analyser engine will be presented to the relevant decision makers as the feedback loop to the clinicians/policy makers, allowing them to make more informed decisions regarding targeted recommendations and policies

The Social Analyser component will utilise a number of sub-components, each with a specific purpose that contributes towards delivering a comprehensive social media analysis and decision support system. The following sub-components are used by the Social Analyser:

- **CEP Engine** to develop event patterns and event trends based on expert and machine learning methodologies. The engine will be extended to enable the inference of probabilistic events suitable for early risk identification and evaluation of targeted recommendations.
- **NLP** to analyse, decipher, understand and make sense of the human languages in a manner that is useful for the objectives of the Social Analyser component
- **SNA** to analyze social networks on the social media platforms and to extract relevant information about the structure of the networks, identify clusters or communities and predict new links between users
- **Administrative Dashboard** to create alerts by specifying which topics (keywords and phrases) they are interested in monitoring on specific social media platforms. These topics can relate to the lifestyle trends, mental models, emotional intelligence, social interactions and societal influences on the individuals
- **Monitoring Dashboard** to provide real-time monitoring updates on the pre-specified alerts through a set of intuitive visualisations. The alerts will show e.g., how many social media posts are being posted on specific topic(s), from whom, when and the pattern of posts
- **MySQL** is an open-source relational database management system (RDBMS). It organizes data into one or more data tables in which data types may be related to each other. MySQL is used to store data about users of the social media analysis solution
- **MongoDB** is a document-oriented NoSQL database used for high volume data storage. MongoDB is used to store the social media data
- **Apache Kafka** is a system to manage the events' logs. It is a durable messaging system that send messages between processes, applications, and servers. Apache Kafka is used as a message brokering system in the social media analysis component. It is where the messages will be published for processing by different components in the social media analysis solution

- **Apache ZooKeeper** is a centralized service for maintaining configuration information, naming, providing distributed synchronization, and providing group services. ZooKeeper is used to coordinate and manage different components in the social media analysis solution

6.4.5.2 Interactions with other components

The Social Analyser component will provide intuitive interfaces (as Administrative and Monitoring Dashboards) to the users, which are perceived to be clinicians and other healthcare related policymakers. The component doesn't require input from any other component within the iHelp platform as it gets input directly from the users (i.e., alerts on specific topics) and social media platforms (i.e., data concerning discussions taking place related to the alert-topics).

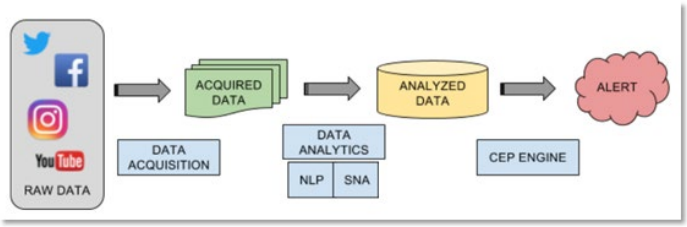

Figure 16: Overview of the Social Analysis component

However, the output of the Social Analyser component will be stored in the iHelp platform e.g., as additional data type in the HHR structure. In this respect, the **`Social Analyser`** will interact with the following component in the iHelp platform:

- **`DataConnector`** - to ingest the social analytic data within the relevant HHRs in the iHelp platform

6.4.6 Monitoring and Alerting

6.4.6.1 Objective of the software artefact

The main goal of the Monitoring and Alerting module is to track and/or monitor the effect of the recommendations that are prescribed to the individuals by the clinician/medical experts. The module (and its sub-modules) will run as a separate service and will observe, check, and aggregate data coming from secondary data sources (mobile and/or wearable devices), and compare them with personalized targets issued by the clinicians/medical experts.

The monitoring mechanism will be executed on assigned configurable intervals (daily, weekly, monthly etc.), will make a quick assessment of progress towards goals, and will decide whether to activate the alerting submodule. The decisions will be based on comparing certain monitored parameter target values defined in the personalized advices against real values. Different escalation policies and threshold values will be considered when generating the proper alert and alert recipient (monitored individual or the advisor). The alerting functionality may be embedded in the delivery mechanism e.g. mobile or wearable apps.

6.4.6.2 Interactions with other components

The Monitoring and Alerting module will be packed up in a docker container and will be part of the iHelp infrastructure. It can be started on certain configurable intervals and work with respect to the FaaS (*Function-as-a-Service*) paradigm or it can run continuously depending on the level of granularity of

achieving the targets, set by the advices, we want to track, the time of reaction and the most appropriate thresholds and value tolerances we have set.

The Monitoring and Alerting module will use the following simple logic:

1. The service will be started [on certain configurable intervals]
2. It will obtain a list of currently active personal advices from the personal adviser.
3. In the context of each rule: aggregate row data if rule uses comparison of aggregated data against target
4. It will obtain a list of aggregated data regarding the individuals from the advices
5. It will evaluate actual values vs targets
6. Based on various parameters to be monitored, target value, escalation policy etc. the system will generate alerts, to be sent to the individual or the clinician
7. Generated alerts are sent to the appropriate person using the delivery mechanism e.g. mobile or wearable apps.

In order to work properly, the module requires the following interfaces:

- Raw data for evaluation – we need an interface that will provide specified data type for a given individual in a given period
- Personalised advices - (individual, alert title and content, parameters to be monitored , target value , escalation policy) from personal adviser will be obtained on certain intervals (daily, weekly , etc)
- Strategy list – obtain a list of strategies defining what actions should be performed if certain target values have been reached – e.g. notify user twice, if still in warning notify clinician, escalate one week later.
- Alert distribution / delivery – mechanism to send certain alert to certain individual (achieved through the Personalized Advisor component, see §6.4.4).

The Monitor and Alerting module will offer the following interfaces:

- **Data Aggregator** – if required - aggregate raw data for a period – how many steps user has done in a week
- **Alert Evaluation** - do we have a match between certain advice target and user results
- **Alert Generator** – replace placeholders in advice with actual values and send them to alert distribution interface according to strategy and personal preferences

6.5 Data storage

6.5.1 Big Data Platform

6.5.1.1 Objective of the software artefact

This software artefact provides the persistent data management layer of the iHelp platform. It will rely on LXS datastore and its main objectives are the following:

- **To allow for data ingestion in very high rates**. Based on its internal ultra-scalable transactional engine of LXS and the sophisticated internal data structures for data indexing, the Big Data Platform can allow for highly rated data ingestion, avoiding the high contention of traditional relational datastores, and providing transactional semantics that the NoSQL world sacrifices for the sake of scalability. This will be used in the data ingestion pipelines that will be established for migrating data into the iHelp platform, either by batch ingestion (periodic or static), or by sending data over a stream.
- **To allow for efficient query execution for analytical processing over the operational data**. The Big Data Platform provides Hybrid Transactional and Analytical Processing (HTAP) that allows to perform analytics over the operation data, as data is being ingested, without the need for migrating historical data into a data warehouse and perform the analytic operations there. This means that the analytics will always access fresh and not obsolete data and can provide insights over the real data set. By providing the rich query processing capabilities of the relational datastores, data pre-processing can be efficiently executed near the data storage, avoiding the need to send the entire datasets to the analytical processing layer, while the intra-operation parallelism of the query engine allows for parallel execution of analytical and aggregated operations.
- **To allow for federated query processing, combining data that are stored externally to the Big Data Platform**. Using the polyglot capabilities of its query engine, the Big Data Platform can open connections and push down queries to be processed in other instances that might be deployed externally. This can be the case where each organization has deployed its own instance of the iHelp platform, and might need to correlate data that are stored internally, with data that are stored in another organization. The Big Data Platform can accept such queries over federated datastores, open connections to the external ones to access data, push the query execution there, retrieve only result of such execution (without having to move data between the different instances) and combine those with the data stored internally.

6.5.1.2 Interactions with other components

The Big Data Platform is a central component of the iHelp platform and it interacts with almost all other software artefacts of the platform. It will be provided as an internal component with a static deployment, whith scaling capabilities at runtime in order to cope with increased workloads. This component is mainly used for others to read or to store data and it will only read data autonomously in case of a federated query execution via its polyglot engine. We categorize its interactions with the other components in two different types: One used for storing data and one used for reading data.

Inserts:

The main software artefact that will interact with the Big Data Platform in order to store data is the *HHR Importer*. The latter will communicate with the Big Data Platform either by its standard JDBC connectivity mechanism, or by making use of it is dual SQL/NoSQL interface and its direct API that allows data ingestion in very high rates. Another option that can be investigated in a latter phase of the project, is the communication of the *HHR Importer* with the Big Data Platform, via an intermediate Kafka data queue. In such a case, the LXS Kafka connector will be used, which inserts data transparently to the datastore. In this case, the *HHR Importer* will have to serialize the data to be ingested via an Avro schema registry.

Another category of software artefact that needs to interact with the Big Data Platform to insert data are the analytical functions that will need to persistently store intermediate of final results, or even store the training results of their algorithms. They will communicate using the standard JDBC data connectivity mechanism of the datastore.

Reads:

The first category of software artefacts that need to interact with the Big Data Platform in order to perform read operations will be the analytical functions. Those functions need to read raw data stored in the data repository in the common data model coming from either primary or secondary resources. They will either make use of the standard JDBC data connectivity mechanism of the datastore, or they can also rely on their own analytical frameworks to perform the pre-processing for their analysis. In case they make use of the Spark analytical framework, then they can also benefit from the LXS Spark connector.

A second category of software artefacts are the ones that are being involved in the data ingestion process. There are envisioned cases where some of the intermediate functions of the data ingestion pipeline might need to read historical data to perform some analysis. They will also make use of the JDBC data connection mechanism of the Big Data Platform.

Finally, this component will interact with the visual analytic tools in order to visualize the result of the analysis. Once again, the most appropriate way is to make use of the JDBC standard connectivity mechanism of the datastore.

7 Infrastructure

In the context of the iHelp project, UPRC will provide the needed computing and storage capabilities to set-up the cloud-based infrastructure to serve the project needs and in order for all the needed technologies and tools to be deployed and managed by project's partners. Overall, this cloud infrastructure aims to help data scientists, healthcare professionals and different stakeholders, to analyse a wide plethora of datasets from different data sources, and facilitate the study of the clinical, physiological, personal and societal aspect of Pancreatic Cancer conditions and treatments.

Furthermore, the project's Consortium has identified a list of needed tools that must be provided and be available to all partners in terms of the overall proposed architecture and the integration and deployment/management of their solutions/components. On top of this, the CI/CD approach represented below, that should be followed during the whole lifecycle of iHelp project and the corresponding technologies/tools have been identified and are shown in Figure 17.

Figure 17: Identified tools.

Moreover, the provided infrastructure will help project's partners to run computation- or data- intensive tasks and host online services in virtual machines or Docker containers on IT resources. To facilitate the cloud provisioning of resources for the project and to collect all the technical needs, UPRC has invited project members to report technical needs in a shared document. This document, which was introduced during one of the iHelp consortium meetings organized by the Project Coordinator, is a live document, and at the date was attached in the Annex A - Infrastructure Discussion.

The aforementioned document is composed of two main sections. In Section 2 (Technical Groups), the appropriate technical partners should be indicated, who will be responsible for the above presented tools/technologies of the CI/CD approach. Different technical groups of partners should be organised based on their expertise, experience, and level of familiarity with these technologies/tools, in order to discuss activities, approaches, and guidelines and to drive the deployment and maintenance of these technologies/tools. The latter will be facilitated based on the role assignment shown in Table 4 (see Appendix), where specific (groups of) partners will be responsible for providing, deploying, and maintaining

the corresponding technologies/tools. Moreover, in this specific table the needed resources for each tool should be recorded in order everyone to be aware of the needs and the overall architecture that will be followed. On top of this, in Section 3 of this shared document (Infrastructure Resources) Table 5 is made available allowing each partner to provide their needs on Infrastructure Resources for components and tools deployment.

The envisaged infrastructure will be provided for the technical partners as a testing environment, accessible remotely and in which every component leader is responsible to test the component and the integration with the other software artefacts present on the platform.

An identical infrastructure will be provided for the validation of the solution. On this instance all the components will be deployed, and the pilots will validate the actual operation of the whole platform.

It is important to note that both the testing and the validation infrastructure will not handle real data, but a set of synthetic data [10] useful only for testing and validation phases.

The pilots, on their premises, will have to provide the defined infrastructure in order to allow the deployment of the needed iHelp components, otherwise, if needed, they could use the centralised solution, adopting the Software as a Service approach, in this case, nothing will be deployed on their premises.

The status of the iHelp infrastructure will be further updated in the upcoming deliverable *D2.5 – Conceptual model and reference architecture II*, which will be delivered during M18. The total amount of the resources provisioned by the UPRC will be also presented for supporting the iHelp pilot partners.

8 Deployment

In this section a description is given of what components (§6) will be deployed on which nodes of the infrastructure (§7).

Due to some ethical and legal aspects, it could be possible that in some cases data to be ingested cannot be shared and transferred across the countries. Another aspect to take in account is that not all the pilots have the same goal or expressed the same use cases, hence not all pilots will need the same iHelp features.

To overcome the former limitation and addressing the latter need, iHelp propose, for the first deployment, a modular, static and tailored deploy on the hospital premises. As an overview, each pilot holds its own raw data, and can use iHelp on the ingested data compliant to the common HHR data model

In case it becomes possible to transfer the data, and to use the platform as SaaS, the designed architecture could be deployed in a remote cloud, and be accessed from the final users.

Table 1: Traceability matrix (Use Cases / Components)

		HHR Importer	Data Gateway	Data Harmoniser	Data Cleaner	Data Qualifier	Data Connectors	Analytic Workbench	DSS Dashboard	Big Data Platform	Personalised Predictor	Predictor and Risk identifier	Mobile App	Personalised Advisor	Social Analyser	Monitoring and Alerting	Secondary Data Pre-processor	WebApp for HCP
Develop Risk Prediction Model	UNIMAN, HDM, MUP, TMU	x						?	x	?	?				?			
Risk Assessment	UNIMAN, HDM, MUP, TMU								x	x	x	x						x
Request Tests and Samples	UNIMAN, HDM, MUP, TMU	?	?	?	?	?	?		x	?				?		?		x
Elevated Risk Detected	UNIMAN, HDM, MUP, TMU								x		x	x						x
Ingest Tests and Samples results	UNIMAN, HDM, MUP, TMU	x	x	x	x	x	x			x								
Risk Mitigation/Treatment planning	UNIMAN, HDM, MUP, TMU	?	?	?	?	?	?		x	?								x
Advice Review	UNIMAN, FPG, HDM, MUP, TMU								x									x
Risk Mitigation Delivery	UNIMAN, FPG, HDM, MUP, TMU												x	x				
Monitoring	UNIMAN, FPG, HDM, MUP, TMU	?	?	?	?	?	?			x			x	x		x	x	
Advice Follow-up	UNIMAN, HDM, MUP, TMU								x									x
Caption of profiling characteristics	UNIMAN, HDM, MUP, TMU	x	x	x	x	x	x			x			x				x	
Reporting	FPG								x									
Plan a visit or a remote contact	FPG	?	?	?	?	?	?		x	?				x				x

Table 1 shows a traceability matrix that has as main goal to summarise the impact of each use case, collected and available in the *D2.1*[2], on the main components defined in this document.

For each use case, the components required for the standard flow are marked with a **x**, and with a **?** the components possibly involved by optional flows.

The specific flow and invocations will be better detailed in the scope of the task *T2.3 – Functional and Non-Functional Specification*.

Table 2: Traceability matrix (Pilots/Components)

	HHR Importer	Data Gateway	Data Harmoniser	Data Cleaner	Data Qualifier	Data Connectors	Analytic Workbench	DSS Dashboard	Big Data Platform	Personalised Predictor	Predictor and Risk identifier	Mobile App	Personalised Advisor	Social Analyser	Monitoring and Alerting	Secondary Data Pre-processor	WebApp for HCP
UNIMAN	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
FPG	•	•	•	•	•	•		•	•			•	•		•	•	•
HDM	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
MUP	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
TMU	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•

Table 2 shows the functional dependencies of each use case, and related components, specified per pilot. That matrix, and its further version, can drive the deployment process in the next phases of the project. In that table, are summarised the needed components for each pilot. For every pilot a • means that the component in that column is needed.

The process of requirement refinement is still in progress, but at this time, after the first round discussions and analysis, the result is that most of the pilots (4/5) needed the full solution deployed on their premises. For the moment, seems that one pilot will use a subset of iHelp features.

Due to the modular approach, iHelp will be able to customise the final solution to only the needed components, do to optimise the actual resources. In Figure 18 and Figure 19 two possible on-premises deployments, the first one with all the identified components, the second with a reduced number of deployed components.

As mentioned in section 7, the test and validation phases need the deployment of the whole set of features offered by iHelp, following that, the Figure 18 can also represent the deployment diagram for the testing and validation phases, with the only exception of that the *device* in which is available the VM is not on the hospital premises, but on the iHelp centralised server.

As listed in the section 7, all the backend components are containerised, so they need an execution environment, like Kubernetes, to be operative.

A different approach was adopted for the **`AnalyticWorkbench`**. It uses a container orchestration platform, developed by ICE and known as DryICE.[26] It provides a data processing pipeline and CEP capabilities, and offers the possibility for a serverless deployment of the components.

The consortium agreed for the first release of the platform, to perform a tailored, customized, static deployment of the services and functions involved in the data ingestion data pipeline. Once validated the features of the system and satisfied the pilots' requirements, the technical partners will investigate the possibility to have a "serverless"[27] deployment of such components. That approach will allow to dynamically instantiate the needed component only on demand. This approach could be useful in case of components invoked very rarely, that can be suspended for saving resources, and only if needed, instantiated, and invoked. This approach, and the DryICE solution, will be better detailed in the *WP4*, and could impact some components developed in the scope of the *WP3*.

Another special mention in the deployment diagram is reserved to the **`BigDataPlatform`**. This software artefact, as already explained in §6.5.1, and further detailed in the related deliverable *D4.9*[28], can provide polyglot capabilities built in its query engine. This capability allows the final user (e.g.: a **`ModelBuilder`**) of the Pilot A to correlate some data, among a federated network of datastores, with a selection of data from Pilot B. In this example, we are assuming that both the pilots, i.e. A and B, have the **`BigDataPlatform`** deployed on their premises, and they established a federated datastore. The power of this feature is that the AI component does not have to know the infrastructural or network detail of the iHelp platform, but everything is delegated to the **`BigDataPlatform`**, invoked using the JDBC API in the same execution environment. It can be also used in scenarios when pilot A should make use of data coming from pilot B, but this data cannot live the premises of pilot B. As the **`BigDataPlatform`** can access the data remotely, there is no need to move this data from pilot B to be stored in the infrastructure of pilot A.

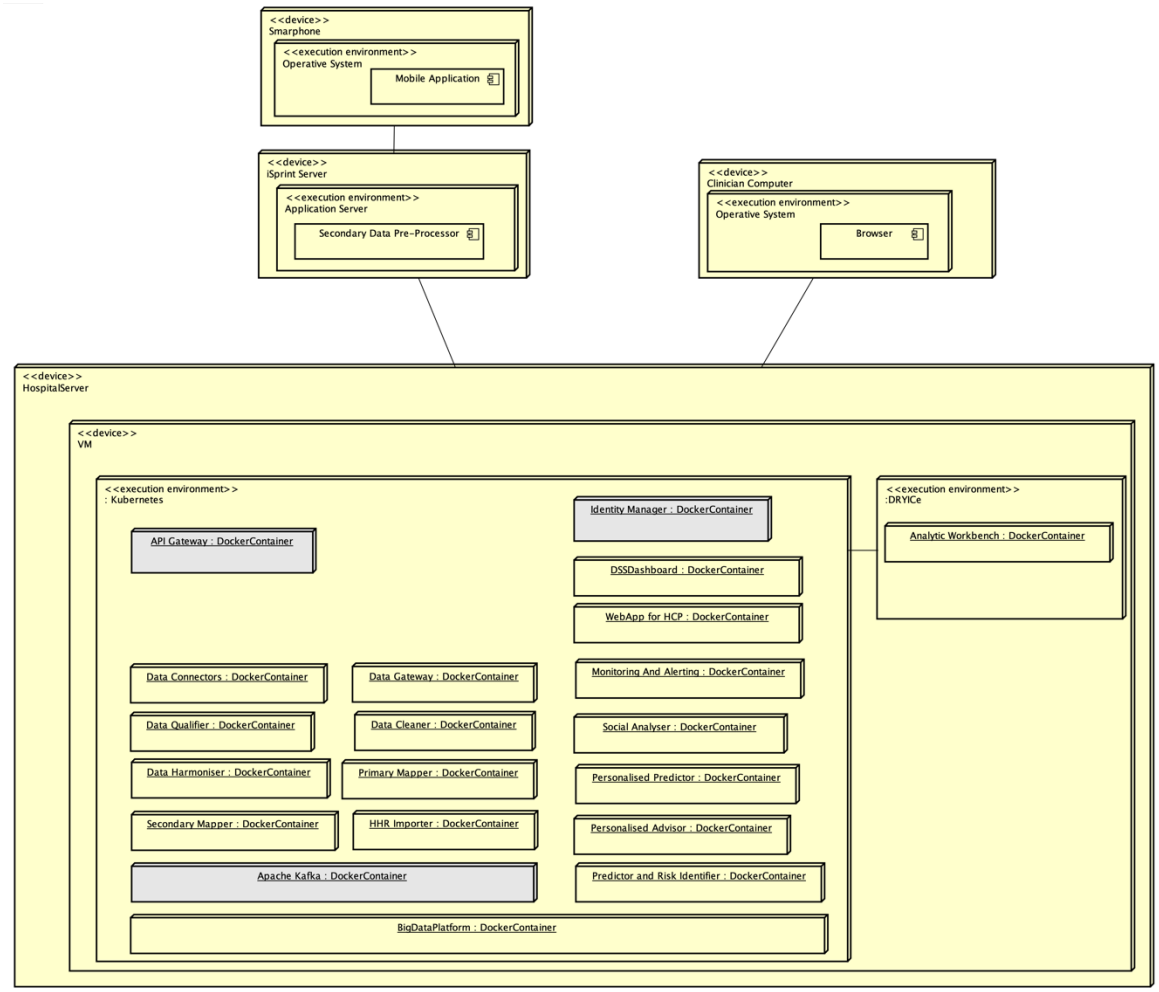

Figure 18: Deployment diagram - whole solution

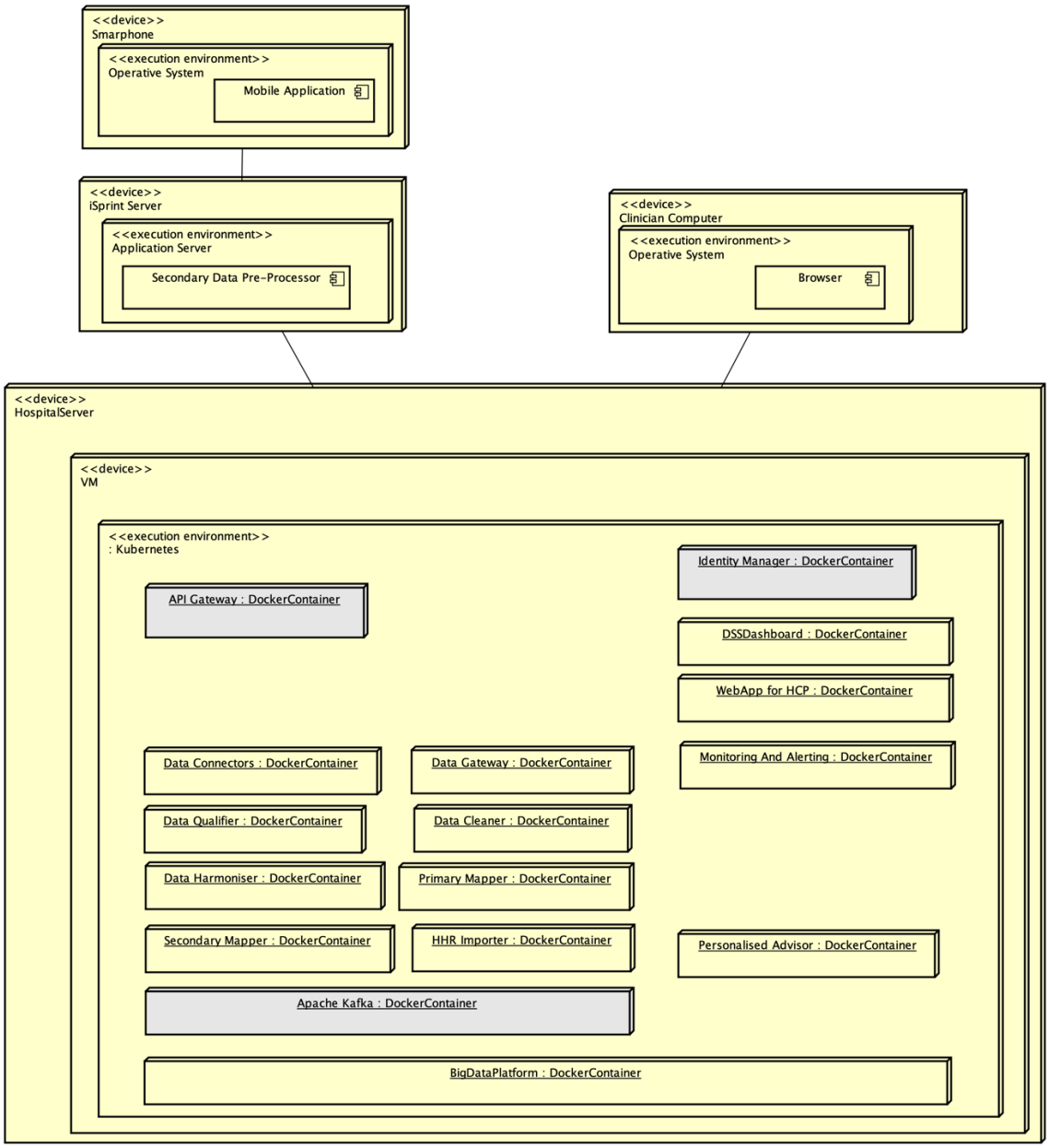

Figure 19: Partial deployment

9 Conclusions

This document provides the initial view of the conceptual overall architecture and model of the iHelp platform. It highlights the main components, their interconnections, key capabilities and offered some details about the infrastructure and the approach that will be followed for the deployment of the platform.

All the details, and the internal design of each component, are demanded to the related tasks; based on this document, instead, is the higher-level image, and how the iHelp solution will be provided to the final users.

During this period the consortium agreed on some architectural decisions, first of all, the software will be provided in form of containerised microservices; some of those will use an asynchronous communication (i.e. Data Ingestion) some other will use the synchronous communication (i.e. Data analysis). The connection with the iHelp repository will be performed using the various data connectivity mechanisms, some of such following well known standards, like the JDBC.

Three kinds of GUI will be provided to the application reserved to the clinicians; specifically, a GUI for the **`HCP`** for looking at the specific patient's data, a GUI for the **`ModelBuilder`**, that can allow to train and tune the AI algorithm implemented in the platform, a GUI for both to elaborate statistics and compute aggregation on the data present in the platform.

The **`Individual`**, will interact with the system using only a mobile application, which can act as an aggregator of the user data that will be shared within iHelp.

Moreover, a centralised infrastructure will be provided, which can receive the deployment of all the components for testing and validation phases. This infrastructure will not handle real data, but only synthetic data. The minimum set of resources needed by the components, that will be provided by the infrastructure, is still in definition progress, the snapshot of the live document on which the technical partners are working on is in the annex.

The deployment, for the first phase, will be static for many of the identified components, but customised on the needs of the pilot, for a more efficient resources allocation.

For the next version of this deliverable, taking into account that the use cases will be more mature and stable, it is reasonable to expect a further detailed GUI and goal of components. The interconnections could be updated to reflect the new needs.

The *serverless* deployment approach will be better analysed to understand if, for the specific pilots' scenarios, it could be feasible and efficient, for the second release, to move some components toward such kind of development. The infrastructure discussion will continue, so the live document in the annex will be updated.

Bibliography

[1] D. Kyriazis *et al.*, "CrowdHEALTH: Holistic Health Records and Big Data Analytics for Health Policy Making and Personalized Health.," *Stud. Health Technol. Inform.*, vol. 238, pp. 19–23, 2017.

[2] iHelp, "D2.1 - State of the Art and Requirement Analysis-I," 2021.

[3] S. Brown, "Software architecture for developers," *Coding Archit.*, 2013.

[4] "CrowdHEALTH." [Online]. Available: https://www.crowdhealth.eu/. [Accessed: 14-Jul-2021].

[5] "MARKOS." [Online]. Available: https://cordis.europa.eu/article/id/117369-markos-global-integrated-and-searchable-opensource-software. [Accessed: 14-Jul-2021].

[6] "SkinCancer." [Online]. Available: http://viewderma.com/. [Accessed: 14-Jul-2021].

[7] "PATHway." [Online]. Available: https://cordis.europa.eu/project/id/643491. [Accessed: 14-Jul-2021].

[8] NEXOF-RA project, "Quality Model for NEXOF-RA Pattern Design."

[9] IHelp, "D1.1 - Project Management Handbook," 2021.

[10] IHelp, "D1.10 - Ethical Issues Related to the Protection of Personal Data," 2021.

[11] The European Parliament and the council of the European Union, "GDPR." [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32016R0679&from=EN. [Accessed: 07-Jul-2021].

[12] IHelp, "D1.2 - Data Management Plan," 2021.

[13] IHelp, "D1.9 - Ethical Issues Related to the Involvement of Humans in iHELP," 2021.

[14] IHelp, "D1.11 - Ethical Issues Related to the Involvement of Non-European Countries," 2021.

[15] IHelp, "D1.12 - Ethical Issues Related to the Human Cells-Tissues," 2021.

[16] IHelp, "D1.13 - Ethics Guidelines for Trustworthy AI in iHELP," 2021.

[17] OMG, "UML." [Online]. Available: https://www.omg.org/spec/UML/2.0/About-UML/. [Accessed: 07-Jul-2021].

[18] "Traefik." [Online]. Available: https://traefik.io/solutions/api-gateway/. [Accessed: 07-Jul-2021].

[19] "Keycloak." [Online]. Available: https://www.keycloak.org/. [Accessed: 07-Jul-2021].

[20] "OpenId Connect." [Online]. Available: https://openid.net/connect/. [Accessed: 07-Jul-2021].

[21] "Apache Kafka." [Online]. Available: https://kafka.apache.org/. [Accessed: 07-Jul-2021].

[22] "Kubernetes." [Online]. Available: https://kubernetes.io/. [Accessed: 07-Jul-2021].

[23] "Node Red." .

[24] G. Tiwari, A. Sharma, A. Sahotra, and R. Kapoor, "English-Hindi neural machine translation-LSTM seq2seq and ConvS2S," in *2020 International Conference on Communication and Signal Processing (ICCSP)*, 2020, pp. 871–875.

[25] P. Bahar, N. Makarov, and H. Ney, "Investigation of transformer-based latent attention models for neural machine translation," in *Proceedings of the 14th Conference of the Association for Machine Translation in the Americas (AMTA 2020)*, 2020, pp. 7–20.

[26] I. Catalyst, "ICE Knowledge Discovery." [Online]. Available: https://informationcatalyst.com/ice-knowledge-discovery/. [Accessed: 16-Jul-2021].

[27] I. Baldini *et al.*, "Serverless Computing: Current Trends and Open Problems," in *Research Advances in Cloud Computing*, S. Chaudhary, G. Somani, and R. Buyya, Eds. Singapore: Springer Singapore, 2017, pp. 1–20.

[28] iHelp, "D4.9 - Big data platform and knowledge management system-I," 2021.

[29] European Commission, "What is a data controller or a data processor?" [Online]. Available: https://ec.europa.eu/info/law/law-topic/data-protection/reform/rules-business-and-organisations/obligations/controller-processor/what-data-controller-or-data-processor_en. [Accessed: 03-Aug-2021].

List of Acronyms

AI	Artificial Intelligence
API	Application Programming Language
CA	Consortium Agreement
CEP	Complex event processing
D	Deliverable
DoA	Description of Action
EEA	European Economic Area
EU	European Union
GDPR	General Data Protection Regulation
GUI	Graphical User Interface
HCP	Health Care Professional
HHR	Holistic Health Records
K8S	Kubernetes
NLP	Natural Language Processing
PC	Pancreatic Cancer
REST	Representational state transfer
SaaS	Software As A Service
SNA	Sentiment Network Analysis
T	Task
UC	Use Case
UML	Unified Modelling Language
VM	Virtual Machine
WP	Work Package

Annex A - Infrastructure Discussion

Infrastructure & Architecture Discussions

As discussed and agreed during the 2nd & 3rd Virtual Consortium Meeting call of iHelp project, UPRC will provide the server and appropriate infrastructures in order all the needed technologies and tools to be able to be deployed and be managed by partners. UPRC can provide the wanted VMs for the project.

Moreover, a list of tools was discussed that need to be provided and be available to all partners in plans of integration and deployment/management of their solutions/components. On top of this, the below CI/CD approach that can be followed during the whole lifecycle of iHelp project, and the corresponding technologies/tools were proposed by Usman (ICE) during the 2nd Virtual Meeting Call of the iHelp Project.

To this end, the below table summarises and indicates the Open Issues and Discussions that have been raised according to Architecture and Infrastructures during our latest Virtual Meeting.

Table 3: Infrastrucutral open points

Open Discussion Items	Description	Technical Partners Involved
Use of a container-orchestration system	UPRC can provide VMs over OpenStack using their infrastructure. It is recommended the use of a container-orchestrator system (i.e. Kubernetes) so that everything can be easy deployed when needed. The latter will require: 1. All platform components/tools to be containerized. 2. Native and locally deployed applications, also from pilots, might need to be containerized (that might require the help of a technical partners) but there	All

	might be always the option for them to deploy in a separate VM.	
Use of a Serverless vs a Static Microservices Deployment	Most of the technical partners raised the issue of having a serverless deployment and utilization of our microservices, in order these microservices can be triggered and utilized in a dynamic way. May we have scenarios which do not need all functions/microservices to be utilized, thus it is proposed to have a more dynamic deployment. - During the 3rd Consortium Meeting it was proposed from Pavlos (LXS) & Sakis (ATC) for the WP3 components to have static deployments at least until the 1st Review. It is strongly recommended to have our deployments on top of Kubernetes. Hence, there is no need for extra VM for each component. WP4 components will be based on serverless implementation based on DryICE tool provided from ICE.	WP3 components static implementation, WP4 components serverless implementation based on DryICE.
(Any other discussion that should be raised)		

Technical Groups

As also proposed, the appropriate technical partners should be named/indicated, who will be responsible for the above presented tools/technologies of the CI/CD approach. Different technical groups of partners should be organised based on their expertise, experience, and level of familiarity with these technologies/tools, in order to discuss activities, approaches, and guidelines and to drive the deployment and maintenance of these technologies/tools. The latter will be facilitated based on the below table, where specific (group of) partners will be responsible for providing, deploying, and maintaining the corresponding technologies/tools. Moreover, in the below table the needed resources for each technology/tool to be deployed should be recorded in order everyone to be aware of the needs and the overall architecture that will be followed.

For example, Kafka must be deployed on a project's single VM with a Public IP in order every partner to be aware and able to connect, produce and consume messages from the appropriate Kafka Topics under this specific IP. For this deployment, a VM of 4vcores, 8GB RAM and 50GB storage is needed.

Table 4: Infrastructural tools

Technology/Tool	Description	Responsible Partner(s)	Needed Resources
Gitlab	The project's private GitLab repository, so that everyone be able to push and share code. Each organization must have at least one account with admin/maintenance rights, so that people can administer the components that their organization is responsible for. Each member of the consortium should have one account there.	UPRC	1 VM with 8 vcores, 32GB RAM, 1 Public IP and 100GB storage.
Central Gitlab Container Registry	Gitlab container registry, so that it is feasible to build, upload and store Docker Images in project's own and private repo. Docker images will be later used by Kubernetes to create the containers and orchestrate the overall deployment.	UPRC	Deployed on top of Gitlab, thus o extra resources are needed.
SonarQube	Automatic code quality check is performed constantly by this tool.	ICE	Deployed on top of Kubernetes, yaml file. Specify in the pipelines through the Gitlab pipelines. Components have to have their own libraries. E.g. Jacoco for Java.

Technology/Tool	Description	Responsible Partner(s)	Needed Resources
Kubernetes	Kubernetes will be utilized for the orchestration of the deployments of the containerized components.	ICE	TBD. Minimum requirements. 2 Kubernetes instances were proposed during the 3rd Consortium Meeting.
Docker Compose/Helm Chart registry	Will be used for defining and running multi-container Docker applications.	UPRC	Deployed on top of Docker. In general docker compose can easily be transposed to Kubernetes docker.
Rancher	Cluster Manager for Kubernetes	ICE	Deployed on top of Kubernetes
Kafka	The Message Bus/Broker of the project. Run time communication between different components of the overall pipeline.	LXS	Will be deployed on top of Kubernetes. Needed resources should be calculated along with Kubernetes resources.
Serverless Orchestrator (e.g. DryICE)		ICE	TDB
(Any other tool that should be highlighted)			

Infrastructure Resources

Based on the below table each partner can provide his needs on Infrastructure Resources, in order to deploy his own components and tools.

Table 5: Infrastructural resources

Partner	Description of Actions	Needed Resources
UPRC	T3.4 Standardization and Quality Assurance of Heterogeneous Data	1VM with 8vcores, 32GB RAM, 50GB storage
ICE	T4.2 Analytic Workbench (DryICE)	TBD. Minimum resources needed.
(Please all partners fill up the table according to your needs.)		