CrowdHEALTH

Collective Wisdom Driving Public Health Policies

D3.7 Data Sources & Gateways: Software Prototype v1

Project Deliverable

This project has received funding from the European Union's Horizon 2020 Programme (H2020-SC1-2016-CNECT) under Grant Agreement No. 727560

D3.7 Data Sources & Gateways: Software Prototype v1

Work Package:		WP3
Due Date:		31/12/2017
Submission Date:		31/12/2017
Start Date of Project:		01/03/2017
Duration of Project:		36 Months
Partner Responsible of Deliverable:		SiLo
Version:		1.1
Status:	☒ Final ☐ Draft ☐ Ready for internal Review ☐ Task Leader Accepted ☐ WP leader accepted ☒ Project Coordinator accepted	
Author name(s):	Konstantinos Perakis (SiLo)	Dimitris Miltiadou (SiLo)
Reviewer(s):	Vegard Engen (IT Innovation)	Tomas Pariente Lobo (ATOS)
Nature:	☐ R – Report ☒ D – Demonstrator	
Dissemination level:	☒ PU – Public ☐ CO – Confidential ☐ RE – Restricted	

REVISION HISTORY			
Version	Date	Author(s)	Changes made
0.1	01/12/2017	Konstaninos Perakis (SiLo)	Draft – Index
0.2	04/12/2017	Konstantinos Perakis & Dimitris Miltiadou (SiLo)	Contributions to section 2
0.3	07/12/2017	Konstantinos Perakis & Dimitris Miltiadou (SiLo)	Contributions to section 2
0.4	13/12/2017	Konstantinos Perakis & Dimitris Miltiadou (SiLo)	Contributions to section 2
0.5	19/12/2017	Konstantinos Perakis & Dimitris Miltiadou (SiLo)	Contributions to section 3
0.6	20/12/2017	Dimitris Miltiadou (SiLo)	Minor modifications in sections 2 and 3
0.7	21/12/2017	Dimitris Miltiadou (SiLo)	Review ready Version
0.7_ITInnov	22/12/2017	Vegard Engen (IT Innovation)	IT-INN Internal Review
0.7_ATOS	22/12/2017	Tomas Pariente Lobo (ATOS)	IT-INN Internal Review
1.0	22/12/2017	Konstaninos Perakis (SiLo)	Final Version
1.1	31/12/2017	ATOS	Quality Review. Submission to EC

List of acronyms

API	Application Programming Interface
CSV	Comma-separated values
DB	Database
IoT	Internet of Things
FHIR	Fast Healthcare Interoperability Resources
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
JWT	JSON Web Token
RFC	Request For Comments
UML	Unified Modeling Language
URL	Uniform Resource Locator
XML	Extensible Markup Language

Contents

List of figures

List of tables

Executive Summary

The scope of deliverable D3.7 is to document the preliminary software development efforts undertaken within the context of Task 3.2 - Data Sources and Gateways. D3.7 is a demonstrator deliverable and the current document is the accompanying report documenting the implementation details of the Data Sources and Gateways component delivered within the context of Task 3.2. This report documents the service prototype overview by providing information about the main components of the prototype. More specifically, the document reports on the processes and internal interfaces of each of the services, the external interfaces exposed by the Data Sources and Gateways component, as well as the technologies and tools used in the implementation of the prototype. In addition to the implementation details, the information concerning the source code availability and exploitation has been provided.

It should be noted that the development of the Data Sources and Gateways is a living process that will last until M22, when the Data Sources and Gateways: Software Prototype v2 (corresponding to D3.8) will be delivered. Additionally, the design and specifications of the Data Sources and Gateways component is subject to updates and refinements based on the initial evaluations of the service and any new requirements that may arise as the project evolves. These changes will be documented in Data Sources and Gateways: Design and Open Specification v2 (corresponding to D3.6) in M20 and will drive the implementation of the second version of this deliverable which will be documented in D3.8 as mentioned above.

1. Introduction

Deliverable D3.7 undertakes the documentation of the preliminary efforts carried out within the context of Task 3.2 - Data Sources and Gateways. Task 3.2 aims at the development of the suitable processes and support tools to enable the acquisition of multimodal data from a variety of sources as provided by various data providers. The scope of these processes and tools includes solving the connectivity and communication challenges that arise due to the diversity of the sources and by providing a level of abstraction of the different data formats of the incoming information into the common specification format of Holistic Health Records. The implementation of the Data Sources and Gateways component was driven by the architecture and the design presented in deliverable D3.5 of the project (Perakis K., Miltiadou D., De Nigro A., Torelli F., 2017).

Since D3.7 is a demonstrator deliverable, the current document is the accompanying report documenting the information concerning the implementation of the Data Sources and Gateways component delivered within the context of Task 3.2, based on the information provided until the time of writing by the project pilot partners. Within this context, an overview of the main components of the prototype is documented and for each of the services of the prototype the list of implemented processes and internal interfaces is presented. In addition to this, the external exposed interface of the Data Sources and Gateways component is documented along with the technologies and tools used in the implementation of the services. In conjunction the implementation details, the information concerning the source code availability and exploitation has been documented.

The deliverable is structured as follows:

1. Section 1 introduces the deliverable and documents its scope, providing also a brief description of the document structure. It also documents the positioning of the deliverable within the project and the relation of the current deliverable with the other deliverables.
2. Section 2 describes the main components of the prototype by presenting all the services implementing supplemented by the list of functions and internal interfaces implemented. In addition to the implemented services, the exposed external interface is documented along with the technologies and tools used in the implementation.
3. Section 3 provide the necessary information about the source code availability and exploitation.
4. Section 4 concludes the current deliverable and discusses future work.

Deliverable D3.7 is the first version of the Data Sources and Gateways component. However, the development of the Data Sources and Gateway will last until M22 when the second version will be delivered with deliverable D3.8 Data Sources and Gateways: Software Prototype v2. The implementation of the second version will be driven by the second version of the architecture and design of the component which will be documented in deliverable D3.6

Data Sources and Gateways: Design and Open Specification v2 in M20. Deliverable D3.6 will contain additional end user requirements as identified within the context of the project and any refinements deriving from the initial evaluations of the service.

2. Prototype overview

2.1. Main components of the prototype

The scope of Data Sources and Gateways component is to undertake the process of data acquisition from various data sources and various data providers, available usually in heterogeneous data formats and through different connection and communication options. The Data Sources and Gateways component delivers the abstracted and unified API, as designed and documented in deliverable D3.5 of the project (Perakis K., Miltiadou D., De Nigro A., Torelli F., 2017), with a two-fold purpose.

Firstly, the Data Sources and Gateways component will address the connectivity and communication challenges that the various data providers introduce by enabling the acquisition capabilities of CrowdHEALTH platform from sources like healthcare organizations, IoT devices or sensors, web platforms and more. Secondly, it will offer a level of abstraction of the various data formats of the incoming information from the rest of the components of the CrowdHEALTH platform by interpreting, harmonizing and cleaning (in terms of erroneous data correction and missing data handling) all incoming information into the common specification format of Holistic Health Records, through interactions with the rest of internal components of CrowdHEALTH.

In deliverable D3.5 the architecture and the design of the Data Sources and Gateways component was analysed and documented. The component design as documented in D3.5 is illustrated in Figure 2-1 for reference.

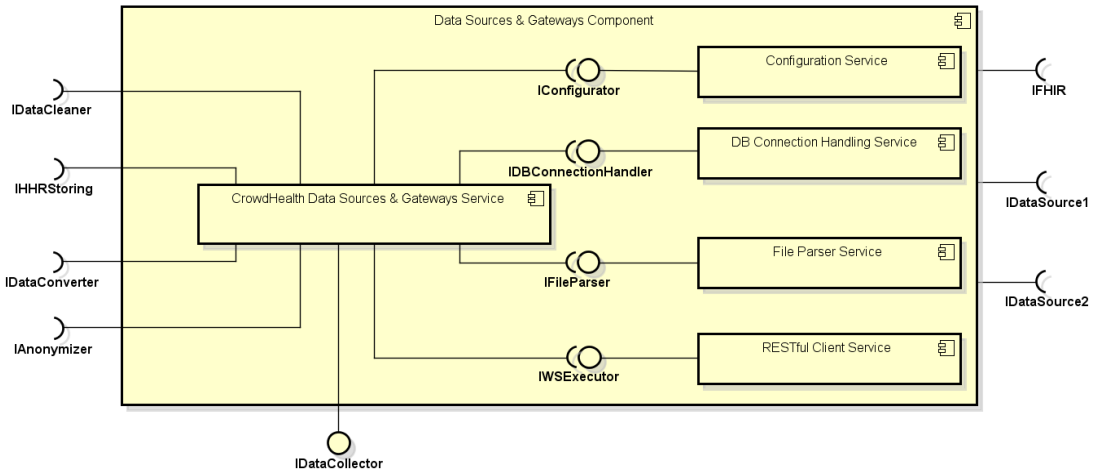

Figure 2-1: Data Sources and Gateways component design

The implementation of the component was driven by this specification and the implementation model of the Data Sources and Gateways component is illustrated using a UML diagram in Figure 2-2 (an extract of the class diagram for readability purposes).

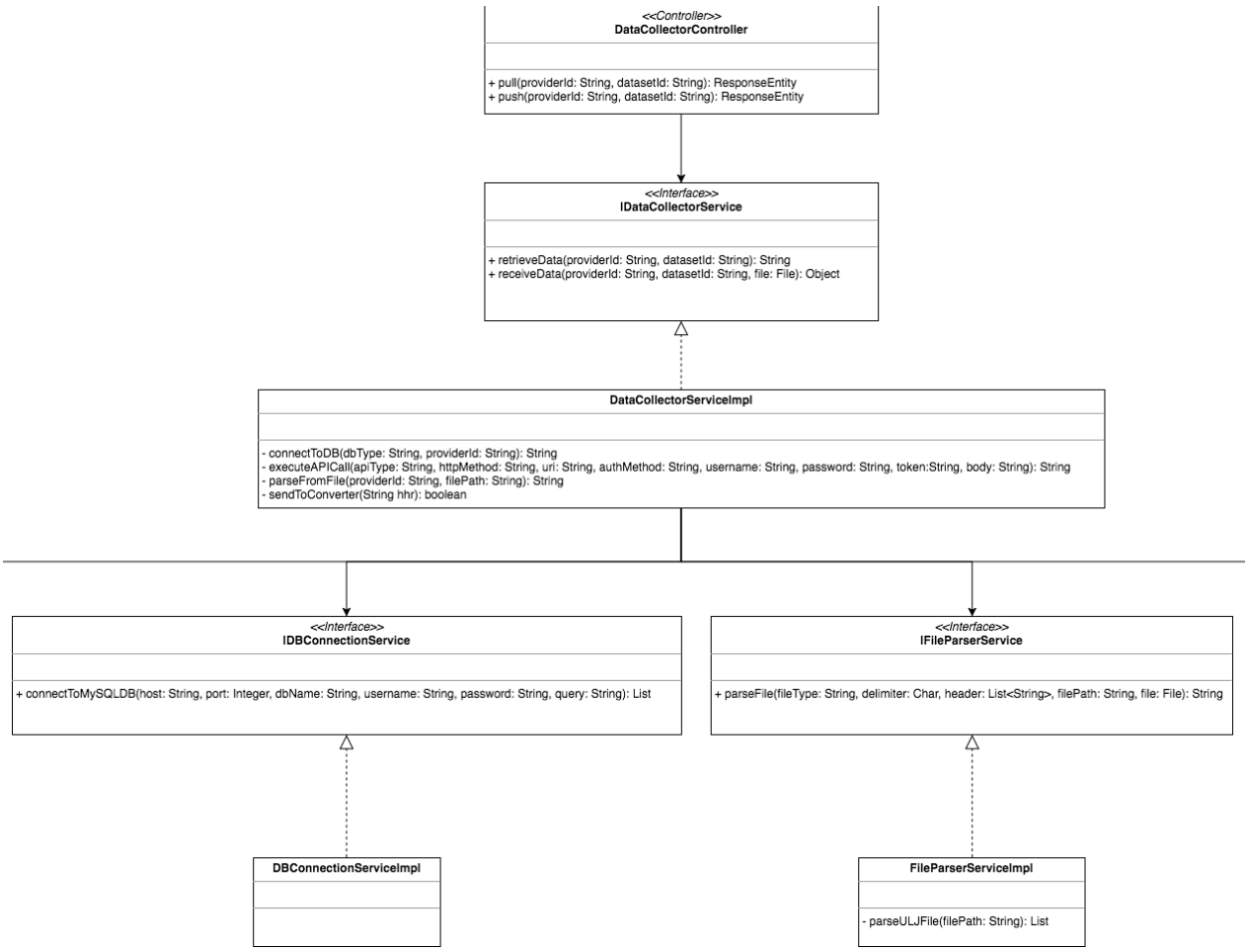

Figure 2-2: Data Sources and Gateways model

As displayed in the UML diagram the Data Sources and Gateways component is composed by one main service namely the DataCollectorService which is responsible for all incoming and outgoing traffic of the component. This main service comprises of four internal services, the ConfigurationService, the DBConnectionService, the FileParserService and the WSExecutorService, which are not exposed to the rest of the platform components and the interaction with these internal services is realised through their exposed internal interfaces. The main service implements also the single interface for data acquisition from external data sources of the CrowdHEALTH platform, the IDataCollectorService interface.

In addition to the communication with the internal services, the DataCollectorService undertakes the necessary communication actions with other internal CrowdHEALTH platform components, such as the Data Cleaner and the Data Converter, through their corresponding interfaces exposed in the course of the incoming information processing.

2.1.1. DataCollectorService

The DataCollectorService is the main service responsible for orchestrating the internal services while also being responsible for the required interactions with the rest of the CrowdHEALTH components towards the execution of the data acquisition flow. Additionally, the service is responsible for the implementation of the single interface for handling data acquisition from external data sources, namely the IDataCollectorService.

The DataCollectorService implements the following four main functions:

- *connectToDB(dbType: String, providerId: String): String* : This function is executing the necessary actions in order to retrieve information from databases. The function requires the database type and the provider identifier as input and interacts with DBConnectionService internal service.
- *executeAPICall(apiType: String, httpMethod: String, uri: String, authMethod: String, username: String, password: String, token:String, body: String): String* : The function is undertaking the necessary actions in order to retrieve information via the exposed external APIs. This function requires various connection and request parameters as input and interacts with WSExecutorService internal service.
- *parseFromFile(providerId: String, filePath: String): String* : This function is executing the necessary actions for the retrieval of information from files (e.g. csv files). This function requires provider identifier and the file path as input and interacts with FileParserService internal service.
- *sendToConverter(String hhr): Boolean* : This function is undertaking the necessary actions to provide the information received to the Data Converter component of the CrowdHEALTH platform in the course of information transformation from Holistic Health Record information to FHIR compliant information.

Moreover, the DataCollectorService implementing the external interface, namely the IDataCollectorService, the only exposed external interface by the component. This interface is

provided with the aim of facilitating the data acquisition from external sources with the following two main functions:

- *retrieveData(providerId: String, datasetId: String): String :* This function is responsible for pulling information from a data provider. This functionality is only triggered internally when new information will be pulled into the CrowdHEALTH platform.
- *receiveData(providerId: String, datasetId: String, file: File): Object* : This function is responsible for receiving information being pushed from a data provider. This functionality is only triggered externally when new information will be into the CrowdHEALTH platform.

In addition to the DataCollectorService, the DataCollectorController is responsible for handling the HTTP requests of the exposed external interface and implements the following two main functions:

- *pull(providerId: String, datasetId: String): ResponseEntity* : This function is responsible for handling HTTP requests in the case of pulling information from a data provider.
- *push(providerId: String, datasetId: String): ResponseEntity* : This function is responsible for handling HTTP requests in the case of information being pushed from a data provider.

Please also refer to Section 2.2 for more details on the exposed external interface.

2.1.2. ConfigurationService

The ConfigurationService is the internal service responsible for maintaining the data provider configuration files. These configuration files contain all the necessary connection details for each data provider in JSON format files. The service can retrieve these configuration details upon request.

The ConfigurationService implements the internal interface namely IConfigurationService with the following main function:

- *fetchConfiguration(providerId: String, datasetId: String): String* : This function is responsible for fetching the appropriate configuration file. Upon receiving as input the data provider identifier and the dataset identifier the service returns the corresponding configuration file.

2.1.3. DBConnectionService

The DBConnectionService is the internal service responsible for retrieving new information from databases with the appropriate connection details (e.g. host, port, credentials, query, etc.). These connection details are maintained and received from the ConfigurationService.

The DBConnectionService implements the internal interface namely IDBConnectionService with the following main function:

- *connectToMySQLDB(host: String, port: Integer, dbName: String, username: String, password: String, query: String): List* : This function is responsible for pulling information from a database of a data provider (i.e. connection to MySQL). The expected input is the corresponding connection details along with the query to be executed.

It should be noted that the DBConnectionService will be extended to support additional databases in the forthcoming versions of the Data Sources and Gateways component, such as for example MongoDB, as described in D3.5, based upon the requirements of the project Data Providers.

2.1.4. FileParserService

The FileParserService is the internal service responsible for retrieving new information from files (e.g. csv files) with the appropriate configuration details (e.g. file type, delimiter, file path, etc.). These configuration details are maintained and received from the ConfigurationService.

The FileParserService implements the internal interface namely IFileParserService with the following main function:

- *parseFile(fileType: String, delimiter: Char, header: List<String>, filePath: String, file: File): String* : This function is responsible for extracting information from files as provided by the data provider. The expected input is the configuration details required for the effective file parsing execution (e.g. file type, delimiter, file path, etc.) as obtained by the ConfigurationService.

The FileParseService also implements the following function:

- *parseULJFile(filePath: String): List* : This function is a more specific implementation of the generic parseFile function for the file parsing of excel files currently provided by ULJ/SloFit.

2.1.5. WSExecutorService

The WSExecutorService is the internal service responsible for retrieving new information from external exposed APIs with the appropriate connection details e.g. web service type, http method, URI, etc.). These connection details are maintained and received from the ConfigurationService.

The WSExecutorService implements the internal interface namely IWSExecutorService with the following three main functions:

- *executeGETCall(authMethod: String, token: String, username: String, password: String, uri: String, body: String): String* : This function is responsible for supporting any incoming HTTP GET request for information from a data provider. The function requires as input all the necessary information as obtained by the corresponding configuration file.
- *executePOSTCall(authMethod: String, token: String, username: String, password: String, uri: String, body: String): String* : This function is responsible for supporting any incoming HTTP POST request for information from a data provider. The function requires as input all the necessary information as obtained by the corresponding configuration file.
- *executePUTCall(authMethod: String, token: String, username: String, password: String, uri: String, body: String): String* : This function is responsible for supporting any incoming HTTP PUT request for information from a data provider. The function requires as input all the necessary information as obtained by the corresponding configuration file.

2.2. Interfaces

As described also in the Section 2.1.1, the Data Sources and Gateways component is providing the implementation of the interface responsible for data acquisition through the DataCollectorService, namely the IDataCollectorService and the HTTP requests are handled by the DataCollectorController.

The interface offers the following three endpoints:

1. Authentication/Login endpoint
2. Push data endpoint
3. Pull data endpoint

2.2.1. Authentication/Login endpoint

This endpoint is responsible for the authentication process of the interface. The authentication process is implemented using token based authentication, more specifically, JSON Web Token (JWT)[1], an open standard (RFC 7519) that defines a compact and self-contained way for securely transmitting information between parties as a JSON object. Once received, each subsequent request will include the JWT to enable the user to access the rest of the endpoints and resources. It should be noted at this point that in the current implementation a predefined user is configured on the Data Sources and Gateways component so as to guarantee communication and information exchange security.

The Authentication/Login endpoint is documented in Table 1:

[1] JSON Web Tokens, https://jwt.io/

Authentication/Login Endpoint	
Description	Creates a JWT for the user, this JWT can be fetched by the response headers and should be included the forthcoming requests to access the rest of the endpoints.
Endpoint URL	http://hostname[:port]/login
HTTP method	GET
Parameters	N/A

Request Body	Expects a request body in the following format:
	{
	“username”: “user”,
	“password”: “pass”
	}

Table 1- Authentication/Login Endpoint

2.2.2. Push data endpoint

This endpoint enables data providers to push data to the CrowdHEALTHplatform. The JWT obtained from the Authentication/Login Endpoint is mandatory as an Authorization parameter in the request. The Push data endpoint is documented in Table 2:

Push Data Endpoint			
Description	Enables data providers to push data to the CrowdHEALTH platform.		
Endpoint URL	http://hostname[:port]/gateway/receive/{providerId}/{datasetId}		
HTTP method	POST		
Parameters	Authorization: Valid JWT as received by Authentication/Login Endpoint. The combination of the providerId and datasetId parameters is predefined according to the project use cases identified:		
	▪ bio ◦ allergies ◦ biosignals ◦ medication ◦ phr ▪ cra ◦ patient ◦ diagnosis ◦ treatment ◦ comorbidity ◦ behaviour ◦ coaching ◦ sideeffect	▪ dfki ◦ activity ◦ allergen ◦ allergy ◦ annotation ◦ biodata ◦ datasource ◦ diet ◦ diettype ◦ dish ◦ ingredient ◦ patient ◦ recipe ◦ recipestep	▪ hulafe ◦ emergency ◦ hah ◦ hospitalization ◦ labtest ◦ morbidity ◦ outpatient ◦ patient

Table 2 - Push Data Endpoint

2.2.3. Pull data endpoint

This endpoint enables pulling data from a data provider. This endpoint will be utilized in the cases where data will be fetched from a data provider in order to be uploaded to the CrowdHEALTH platform. The JWT obtained from the Authentication/Login Endpoint is mandatory as an Authorization parameter in the request.

The Pull data endpoint is documented in Table 3:

Pull Data Endpoint	
Description	
Endpoint URL	http://hostname[:port]/gateway/{providerId}/{datasetId}
HTTP method	POST
Parameters	Authorization: Valid JWT as received by Authentication/Login Endpoint providerId: the provider identifier. Supported values are: • ulj • ki datasetId: the dataset identifier. This is parameter is optional.

Table 3 - Pull Data Endpoint

2.3. Baseline technologies and tools

Data Sources and Gateways is developed as a standalone SpringBoot application[2] with Java 1.8. One of the biggest advantages about Springboot applications is the ability to package the application as jar and using an embedded HTTP server. However, several technologies and tools have been also used in the context of Data Sources and Gateways in order to address specific requirements. In the file parsing functionality, parsing of Excel files had to be handled correctly. For this specific functionality, Apache POI[3] has been utilized. In order to enable build automation and address any build dependencies the Apache Maven[4] tool has been used.

[2] Springboot, https://projects.spring.io/spring-boot/
[3] POI, https://poi.apache.org/
[4] Maven, https://maven.apache.org/

3. Source code

Data Sources and Gateways is organized following the standard directory layout of the Java projects and more specific common directory layout proposed by Maven. At the top level pom.xml file exists containing all the necessary information about the project and configuration details used by Maven to build the project. Additionally, the README.md file exists containing a short description of the project, the necessary information on how to build and run the project. The source files of the prototype reside under the src directory. More specific the src directory contains all the source material for building the project, including the main function and the resources of the project. It should be noted that the current version of the source code available in the repository mentioned below, corresponds to the first, fully functional version of the Data Sources and Gateways prototype, based upon the specifications that have been provided by the project Data Providers (documented in Deliverable D3.5). This source code is subject to changes as per future revised requirements that may come up during the project from the data providers. These requirements may be associated with connection details, with data schemas, or with other features as identified in the previous sections.

3.1. Availability

Data Sources and Gateways is provided as open source repository on GitHub and can be found (currently, until the full transition in the project repository) in the following URL:

https://github.com/SiLo-CrowdHEALTH/gateway

3.2. Exploitation

As already described the prototype is developed as a standalone SpringBoot application with Java 1.8 and Maven tools has been used to handle build automation and address the dependencies. As a result, the only prerequisite for building and running the prototype to have Maven tool preinstalled on the system.

With Maven tool installed on the system, on the top-level directory of the cloned repository run the following command: "*mvn clean install*". This command will download the necessary libraries and build the project.

To run the prototype run the following command: "*mvn spring-boot:run*". This start the start the standalone SpringBoot application and the exposed external interface will be available at *localhost:8080*.

The information described above is also available in the README.md file at the top level of the repository.

4. Conclusions

The scope of D3.7 was to document the preliminary efforts undertaken within the context of Task 3.2 - Data Sources and Gateways. The implementation of the version of the Data Sources and Gateways component was based on the outcomes of the deliverable D3.5 where the architecture and design of the component were presented.

In this first version, the Data Sources and Gateways component includes the main service, namely the DataCollectorService, and the four internal services, namely the ConfigurationService, the DBConnectionService, the FileParserService and the WSExecutorService. The DataCollectorService is responsible for the orchestration of the internal services and the handling of the necessary interactions with the rest of the CrowdHEALTH components, towards the data acquisition flow execution. Moreover, the DataCollectorService is implementing the exposed external interface of the Data Sources and Gateways component. The ConfigurationService is responsible for maintaining and providing the configuration files containing connection details for each data provider. The DBConnectionService undertakes the retrieval of new information from databases based on the connection details obtained by the ConfigurationService. The FileParserService is responsible for retrieving new information from files based on the configuration details provided by ConfigurationService. The WSExecutorService undertakes the retrieval of new information from external exposed APIs with the appropriate connection details as obtained by the ConfigurationService. In addition to the services implemented, the exposed external interface is documented along with the technologies and tools used for the implementation.

D3.7 is a demonstrator deliverable and the current document is the accompanying report documenting the implementation details of the Data Sources and Gateways component by describing the services and the interfaces implemented, the source code availability and exploitation. It should be noted that the development of the Data Sources and Gateways is a living process that will last until M22, when the Data Sources and Gateways: Software Prototype v2 (corresponding to D3.8) will be delivered. The second version of the Data Sources and Gateways will contain refinements and enhancements towards the aim of the providing additional functionalities but also addressing additional end user requirements identified as the project evolves.

5. References

1. Perakis K., Miltiadou D., De Nigro A., Torelli F., Velert S. (2017). D3.5 Data Sources & Gateways: Design & Open Specification v.1. EC H2020 CrowdHEALTH Project