Politecnico di Milano

AA 2018-2019

Software Engineering 2

DD

Design Document

version 2 - 16.12.2018

Matteo Acerbi

Table of Contents

1. Introduction

1.1 Purpose

This document is intended to be the next step in the analysis started with the RASD document, regarding the development of a new software-based service called DATA4Help , which is supposed to be offered by TrackMe app. In the RASD document we wanted to offer a more general description of the model on which the development of TrackMe should have been based, analyzing the world and the machine phenomena, designating the subjects involved in the system using class diagrams, identifying the functionalities that should have been performed and defining the goals to which this service would have due to fulfill and the various requirements to be respected. Now with the DD document we want to continue this study by concentrating the analysis in a more detailed description of the software architecture and the design style of TrackMe, defining the main components involved in it and the way in which they communicate with each other, thus going to found the basis for a future implementation of this software.
More precisely, the document presents:

- A first general view on the high level architecture of TrackMe
- A detailed description of the main components involved in the architecture of TrackMe and their corresponding interfaces (supporting the analysis by using component diagram, and deployment diagram)
- The runtime view (sequence diagrams)
- Architectural styles and design patterns
- Implementation, integration and test plan

1.2 Scope

TrackMe is designed to be an application that, thanks to its internal service DATA4Help, is able to collect and store personal data concerning individuals who register with it, working also as a data-sharing center which allows third parties request data relating to specific individuals or groups of individuals. In the first case, TrackMe must allow the single individual to express his will or not to share the requested data, while in the second case the application must be able to evaluate whether or not it is possible to anonymize requested data. The priority is that DATA4Help grants each user total control and protection of their data, in compliance with the regulations set out in the GDPR [1].

1.3 Acronyms and Abbreviations

- **Acronyms**

DD:	Design Document
RASD:	Requirements Analysis and Specifications Document
GDPR:	General Data Protection Regulation

DB: Database

DBMS: Database Management System

MVP: Model View Presenter is a design pattern used for GUIs

GUI: Graphical User Interface

API : Application Programming Interface

- **Abbreviations**

[Gn]: n-th goal

[Rn]: n-th functional requirement

1.4 Document Structure

Chapter 1 gives an introduction to the problem, identifying the purpose and the scope of the DD Document and also defining abbreviations and acronyms in order to give the reader a clear and complete vision of the problem that will be addressed in the following chapters.

Chapter 2 gives first an overview of how the architecture of TrackMe will be developed, and then analyzes the various components involved in this architecture and the corresponding interfaces, underlining the role that each of them has in the system and also the relationships between them. To improve understanding and support architectural analysis, component diagrams, deployment diagrams and sequence diagrams are used.

Chapter 3 deals with a topic already addressed in the RASD Document, which is the User Interface Design.

Chapter 4 shows the goals and requirements that had already been previously identified in the RASD Document, to define which design components are prepared for their achievement.

Chapter 5 identifies the order in which it is planned to implement the subcomponents of the system and the order in which it is planned to integrate such subcomponents and test the integration.

Chapter 6 lists all the reference documents used as a bibliographic support to develop the DD Document.

2. Architectural Design

2.1 Overview

Regarding a more general view of TrackMe architecture we choose to use a three-tier configuration, composed on three layers of logical computing . Three-tier architectures provide many benefits for production and development environments by modularizing the user interface, business logic, and data storage layers. Doing so gives greater flexibility to development teams by allowing them to update a specific part of an application independently of the other parts. This flexibility can improve overall time-to-market and decrease development cycle times by giving development teams the ability to replace or upgrade independent tiers without affecting the other parts of the system. Below we list and define in more detail the three tiers on which this type of architecture is based:

Presentation Tier: The presentation tier is the front end layer in the three-tier system and consists of the user interface. User interface is often a graphical one (GUI), accessible through a web browser or web-based application and which displays contents and informations useful to an end user. The idea here is to provide the user with an interface through which he can control the set of preferences defined by him and eventually update it, access his personal repository and check any notifications provided by the App Server concerning data access queries by third parties.

Application Tier: Also called the middle tier, logic tier or business logic, it is pulled from the presentation tier. It controls application functionalities by performing detailed processing and contains the functional business logic which drives an application's core capabilities.

Data Tier: This tier comprises of the database/data storage system and data access layer. Examples of such systems are MySQL, Oracle, PostgreSQL, Microsoft SQL Server, MongoDB, etc. Data is accessed by the application layer via API calls. The idea here is that the data tier will take care to store peregarding the user, such as his login credentialssistent informations or his set of prefencrees, but also covering the role and fulfilling the functionalities of DATA4Help, which is intended to be a cloud data storage service in which are stored personal data concerning the user, such as for example health data or location data, and to which TrackMe can access for eventual subsequent data transactions.

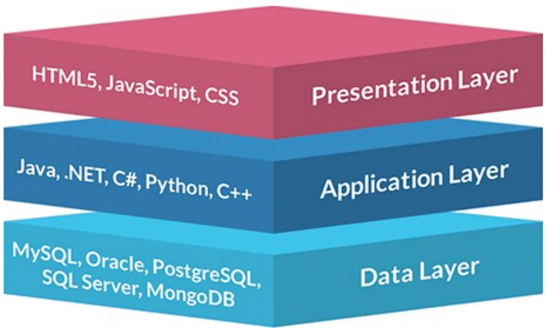

HTML5, JavaScript, CSS
Presentation Layer
Java, .NET, C#, Python, C++
Application Layer
MySQL, Oracle, PostgreSQL, SQL Server, MongoDB
Data Layer

Figura 1: Three-tier Architecture

2.2 High Level Architecture

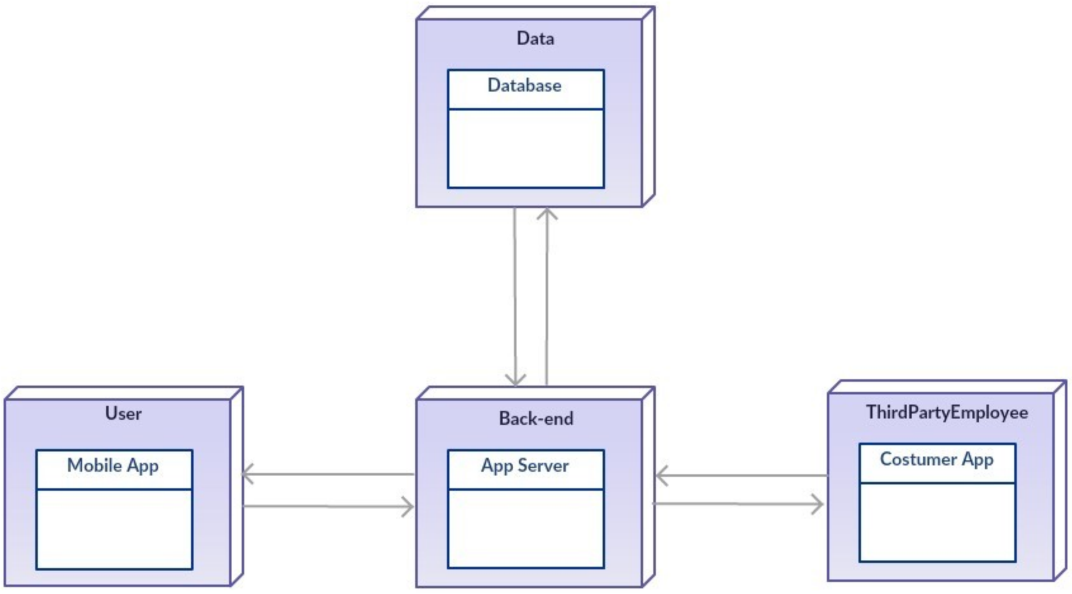


Figura 2: High Level Architecture

We remember from the RASD document that the ThirdPartyEmployee is a person employed by a specific third party in order to perform data queries on its behalf.

User App: running on a mobile computing device, e.g. a smartphone. It is able to connect with different wearable devices or other sensors, e.g. smart watches, via wireless communication, e.g. Bluetooth, to collect different type of data, like health-related data. Here the user can check his preferences set, his notifications space and also request the collection of some of his personal data. The collected raw data will then pass through a quality validation module, which will be introduced later, to get a quality score. After the validation, the validated data will be integrated with several identification labels, e.g. Title to briefly describe the data, DataType to indicate the type of the data, Size to indicate the size of the data, Quality to indicate the quality of the data etc. Some static personal data, e.g. gender, age and weight, could also be integrated if necessary. The integrated data will then be compressed and encrypted. The encrypted data will then be uploaded to DATA4Help that can be seen as a cloud store.

App Server: It represents the main component of the application, acting as an intermediary between the other present components. Here are computed the logical operations that allow the correct behavior of the application, such as:

- accessing to the database in order to insert new data or acquire some of them
- sending notifications to users in case of data access requests performed by third parties
- data sharing

The Server App then interacts directly with the user, the ThirdPartyEmployee and the database thus representing the central core of the application.

Customer App: running on a local device which is connected to the internet, or on a cloud server it is makes the ThirdPartyEmployee able to query for data access, concerning both specific users and groups of users, and to be notified about the outcome of each transaction.

Recalling what has already been stated in the RASD document, in this project DATA4Help is not designed to be a mobile application, but as an internal service to TrackMe, which instead is designed to be effectively an app. Therefore, the idea is to develop a TrackMe application that offers to users the possibility to save their personal data within DATA4Help, which operates as a cloud database service, and to third parties to use this application to gain access to some of the stored data.

2.3 Component View

Now that a general vision of the architecture of our model has been given, in this section we want to go more in deep in the architectural analysis, highlighting each single component that is involved in this model. We can observe how the analysis mainly concerns the AppServer and this is due to the fact that in this area the main logical operations are performed and there is a huge interaction between various components. Another aspect that is interesting to note is that DATA4HelpService is the only one service within the system to have direct access to the database and the idea that motivates this is that DATA4Help is a service provided by TrackMe in which it is possible to create and store repositories, each one associated to one single user, containing both system informations such as user credentials and preferences, and external personal data, such as health data and location data, associated to the same user. As a consequence, any other system service that requires access to the database will need to interface first with DATA4HelpService.

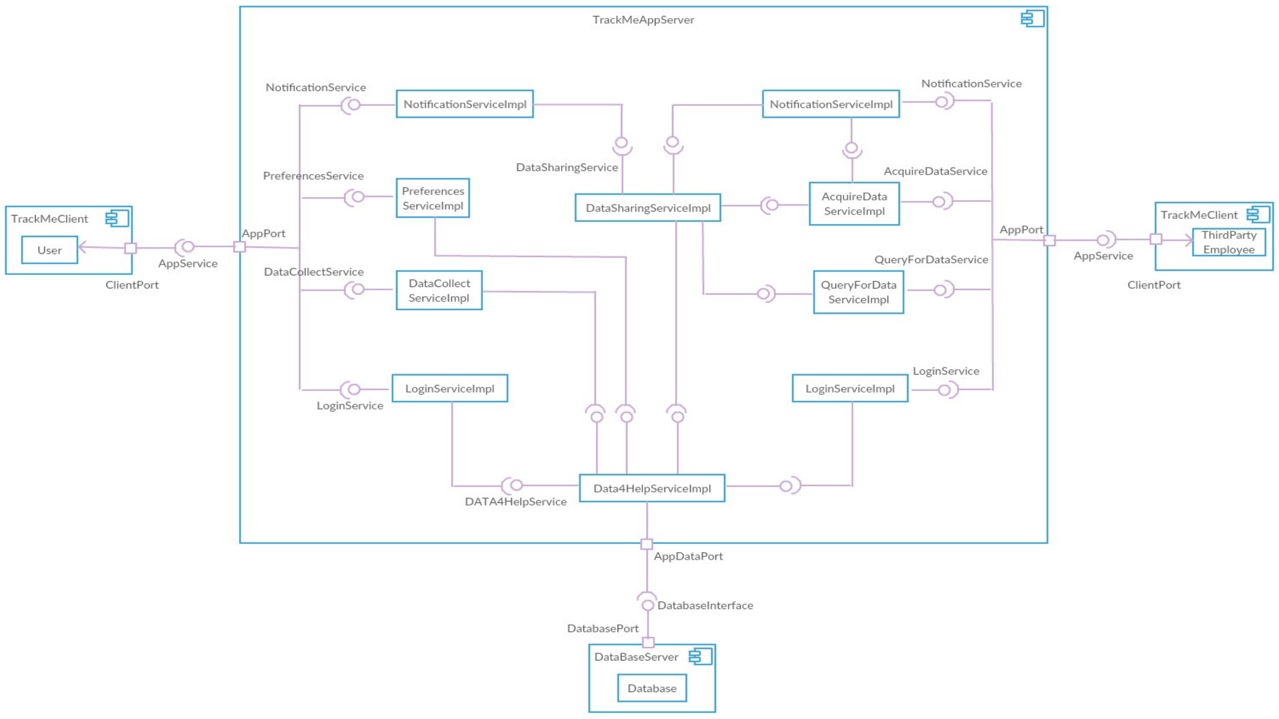


Figura 3: Component Diagram

Below we give a verbal description of the various components depicted in the Component Diagram and of the functionalities that each of them carries out:

- **LoginService:** allows both the registration of a client (which can be both a user and a ThirdPartyEmployee) in case of the first use of TrackMe app, and of its authentication in case of registration already successfully done. It is a basic and essential service for any application or web service that deals with private user data. If the authentication is successfully done, this service gives the user the possibility to access all the other features offered by TrackMe.

- **PreferencesService:** is in charge of allowing the user to define a personal set of preferences, in particular concerning the management of his personal data by DATA4HelpService. In particular, with this service each user can express the desire to make all or part of his personal data collected in DATA4Help inaccessible to any third party. Through this service, the user also has the option to designate in advance certain third parties as unauthorized to access any data or some particular data. All directives defined by the user through this service must be available in any time for the AppServer, so that, when a third party makes a query addressing a specific user, it is possible for the AppServere to consult what was specified by the user in his own set of preferences and eventually deny the transaction. This fact gives advantages, because in this way it is no necessary to communicate the query to the user and therefore to wait for his response, allowing the system to work faster and avoid unnecessary communications to registered users.

- **DataCollectService:** it receives each user request concerning storing of datasets containing some specific personal data and takes care of collecting them, checking their goodness, defining thei type, and finally compressing and encrypting them and then transferring data packages to DATA4HelpService.

- **NotificationService:** this is the interface for communicating informations to the client, who can be both a user and a third party's employee. In the first case, this service aims to communicate to the user any query for data access coming from a specific third party which has passed the user's preferences check, allowing the same user to communicate his consent or not to the access. At the same time, in the second case, the notification service communicates to the third party the responses obtained to queries previously carried out. It is therefore an important service, as it is the foundation of the interaction between users and third parties.

- **QueryForDataService:** it is ready to receive any data access query performed by a third party. As known, these queries may concern specific individual's data or anonymous data about groups of individuals, so this service is designed in order to allow third parties to perform both types of queries, wich are then sent to the DataSharingService component.

- **DataSharingService:** it is the central core in which the main operations for the transfer of personal data are performed. Below there is a list of the main operations performed by DatSharingService:

    - It receives queries from the QueryForDataService module .

    - In case of query concerning a specific user, this service examines user's preferneces set, which is stored in the database, available at any time. If there are no foreclosures in the analysis of the preferences, DataSharingService sends a signal to the NotificationService module, which in turn will notify the user of the addition of a query. Otherwise, DataSharingService requests to send a warning to the query mittent, in order to notify the impossibility to access the requested data.

    - In case of query concerning a group of individuals, this module has to check whether or not it is possible to anonymize the requested data. For simplicity we assume that if the cardinality of this group is less than 1000, DataSharingService denies data access, asking the NoitificationService to send a warning to the third party.

    - If any impediment to data access has been checked, both in the case of queries concering specific individual’s data and queries concerning data about group of individuals, DataSharingService must deal with the data transfer, by sending a request to the DATA4HelpService module for the effective acquisition of data from the database and transferring obtained data to the third party which had carried out the query.

- **DATA4HelpService:** it is a cloud datastore service that has direct access to the application database. It takes care of storing data packages and extracting them within the database.

- **AcquireDataService:** thanks to this service the ThirdPartyEmployee receives the link to the encrypted requested data and also key shares in order to decrypt them. At this point the ThirdPatyEmployee is able to download data, which now are ready to be used and the workflow ends.

2.4 Deployment View

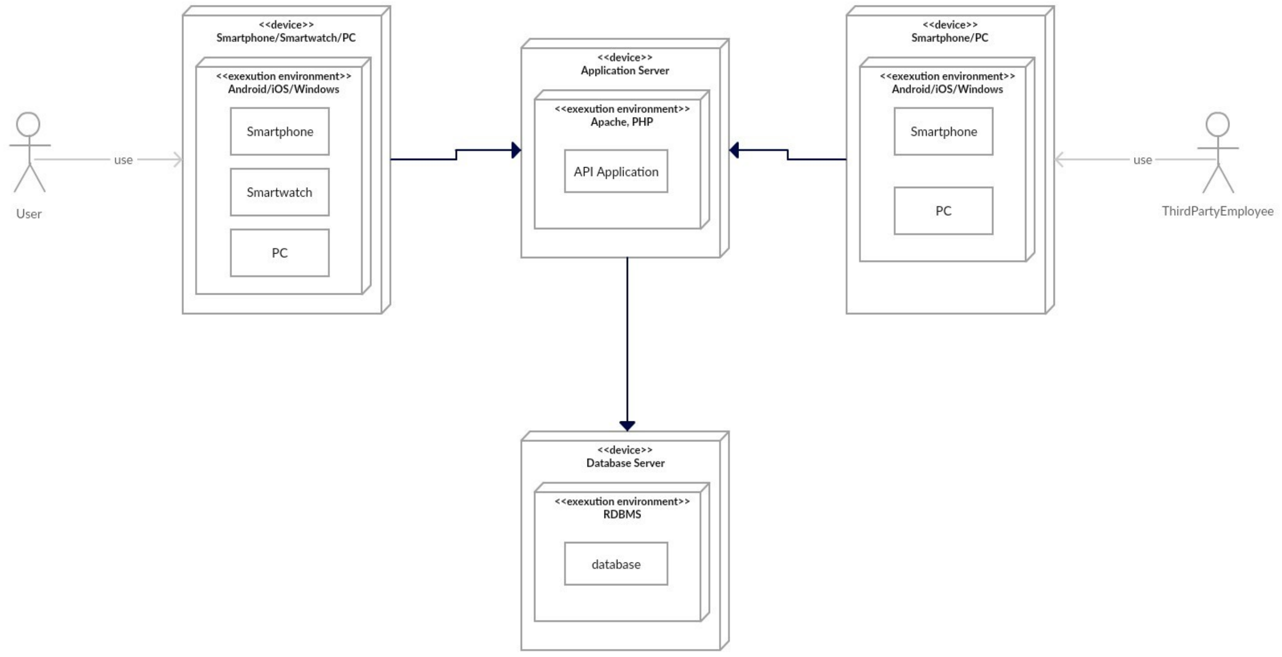


Figura 3: Deployment Diagram

The devices shown in the Deployment diagram are described below:

1. **Smartphone / PC:** these are devices that allow both the normal user and the ThirdPartyEmployee to use respectively the TrackMe app or the TrackMe web site.Through these devices, the generic client can receive informations from the Application Server and take advantage of the features of TrackMe.

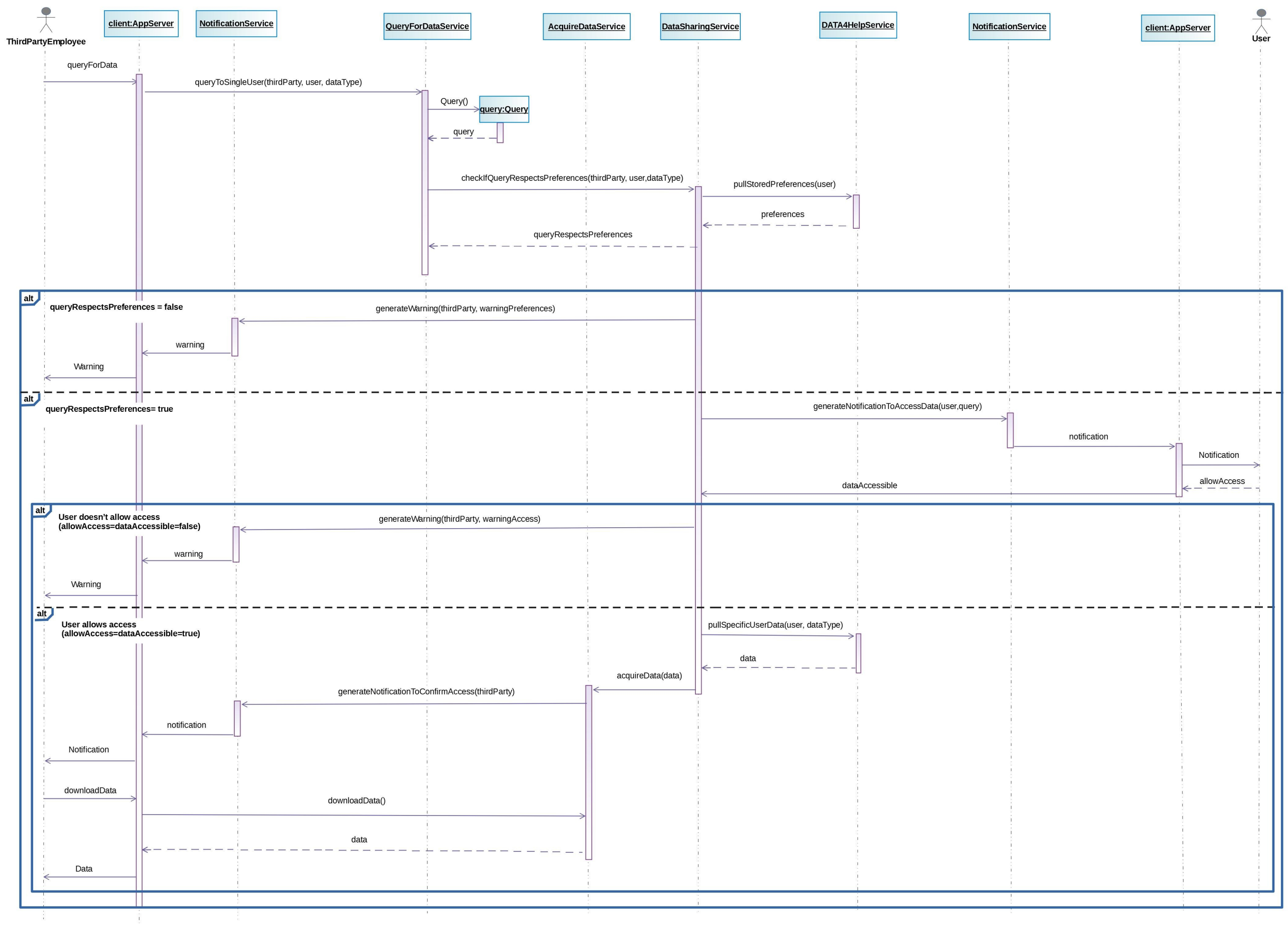
2. **SmartWatch:** is a device reserved exclusively for the normal user use and through which to take advantage of the features of TrackMe application.

3. **Application Server:** here will be computed the main logic and the main functionalities of TrackMe. This server will be used to manage client requests from one side and on the other hand to store all data in the database.

4. **Database Server:** this device has the indispensable role of storing persistent data concerning users, such as login credentials and preferences, as well as other personal informations such as health data and location data.

2.5 Runtime View

2.5.1 Query for data concerning a specific user



ThirdPartyEmployee
client:AppServer
NotificationService
QueryForDataService
AcquireDataService
DataSharingService
DATA4HelpService
NotificationService
client:AppServer
User
queryForData
queryToSingleUser(thirdParty, user, dataType)
Query()
query:Query
query
checkIfQueryRespectsPreferences(thirdParty, user,dataType)
pullStoredPreferences(user)
preferences
queryRespectsPreferences
alt
queryRespectsPreferences = false
generateWarning(thirdParty, warningPreferences)
warning
Warning
alt
queryRespectsPreferences= true
generateNotificationToAccessData(user,query)
notification
Notification
allowAccess
dataAccessible
alt
User doesn't allow access
(allowAccess=dataAccessible=false)
generateWarning(thirdParty, warningAccess)
warning
Warning
alt
User allows access
(allowAccess=dataAccessible=true)
pullSpecificUserData(user, dataType)
data
acquireData(data)
generateNotificationToConfirmAccess(thirdParty)
notification
Notification
downloadData
downloadData()
data
Data

Figura 4: Sequence Diagram - Query for data concerning a specific user

This sequence diagram describes the process that triggers when a third party makes a query to access the data of a specific individual. The first component that operates in this context is the QueryForDataService, which, as already analyzed in the previous chapter, allows a third party to formulate a request for data access, specifying the type of data requested and providing a valid ID of the user from which to obtain data, allowing the application to identify the user within the system. The QueryForDataService then creates the query "object" and sends it to the DataSharingService component, which plays a central role in this process. In fact, when it receives the query, the DataSharingService requires access to the database, which is done through the DATA4HelpService component, in order to check the stored set of preferences expressed by the owner of the requested data. If the preferences consulted reveal the impossibility of transferring the data, the DataSharingService prepares the sending of a warning, which will be carried out through the NotificationService component, to the requesting third party, thus blocking the transaction. Otherwise the DataSharingService prepares the sending of a notification, again through the NotificationService, to the owner of the requested data and awaits its response. If the user allows access to his personal data, the DataSharingService again requests access to the database through the DATA4HelpService, to acquire and decrypt the user data and then send them to the AcquireDataService component, which will allow the third party to download the available data. Otherwise the DataSharingService prepares the sending of a warning to the third party, signaling the problem encountered.

2.5.2 Query for data concerning a group of users

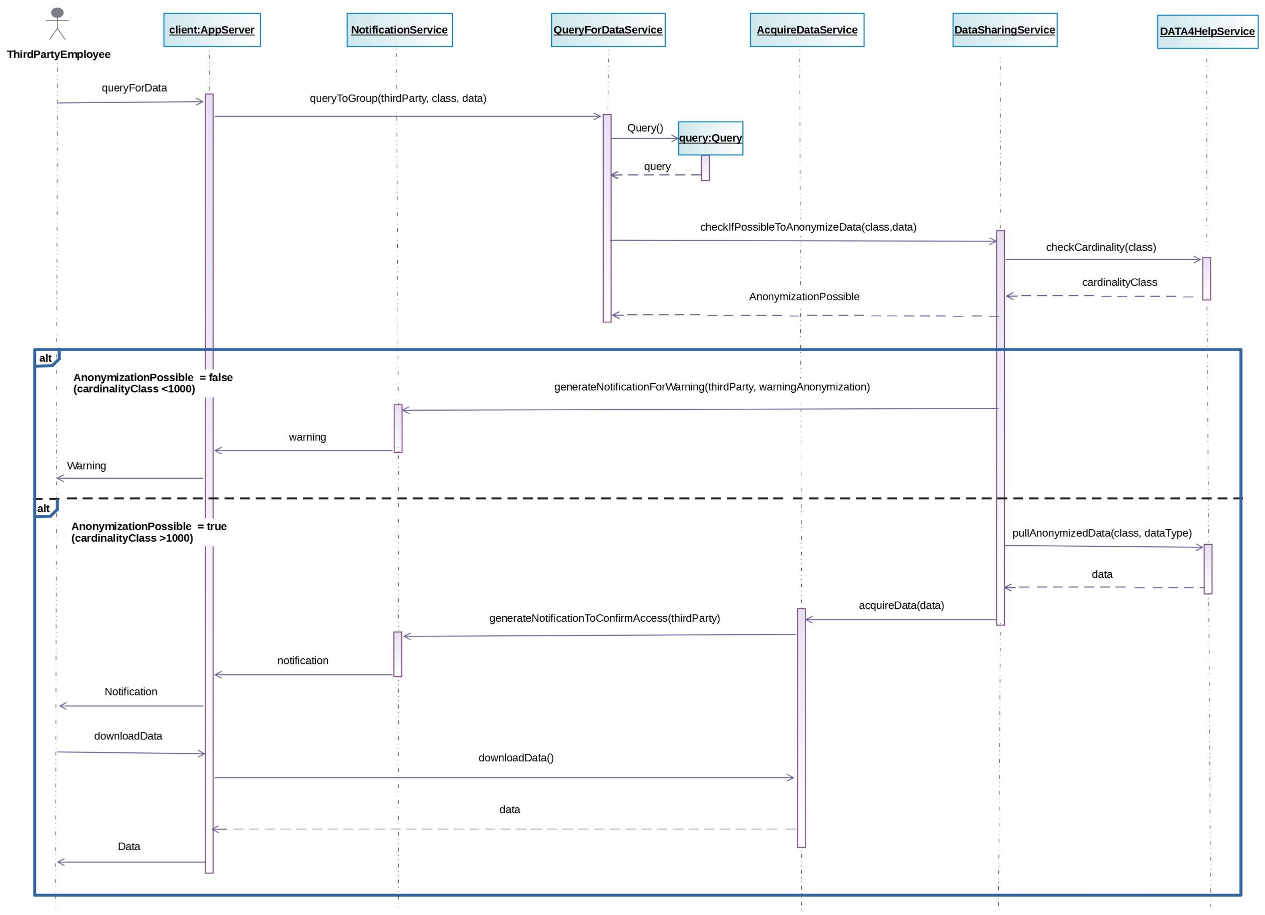

ThirdPartyEmployee
client:AppServer
NotificationService
QueryForDataService
AcquireDataService
DataSharingService
DATA4HelpService
queryForData
queryToGroup(thirdParty, class, data)
Query()
query:Query
query
checkIfPossibleToAnonymizeData(class,data)
checkCardinality(class)
cardinalityClass
AnonymizationPossible
alt
AnonymizationPossible = false (cardinalityClass <1000)
generateNotificationForWarning(thirdParty, warningAnonymization)
warning
Warning
alt
AnonymizationPossible = true (cardinalityClass >1000)
pullAnonymizedData(class, dataType)
data
acquireData(data)
generateNotificationToConfirmAccess(thirdParty)
notification
Notification
downloadData
downloadData()
data
Data

Figura 5: Sequence Diagram - Query for data concerning a group of users

This sequence diagram describes the process deriving from the query of access to anonymous data concerning a group (class) of individuals. Also in this case clearly the first component to operate is the QueryForDataService, which allows a third party also to request data concerning several individuals, specifying the type of data required and defining the class of individuals whose data are to be obtained, as for example, people residing in a specific geographical area or belonging to a specific working class, or to a specific age group, in such a way as to allow the application to identify data owners within the system. The QueryForDataService then creates the query "object" and sends it to the DataSharingService component, which also in this context plays a central role. In fact, when received the query, the DataSharingService requires access to the database, which is done through the DATA4HelpService component, to examine whether or not it is possible to anonymize requested data. For simplicity, it is assumed that this system allows access to dataonly if the number of members belonging to the class specified in the query is not less than 1000. If it emerges from the database that this constraint is respected, the DataSharingService requires access to the data, and then it sends them to the AcquireDataService component, which will allow the third party to download data. Otherwise the DataSharingService prepares the sending of a warning to the third party, signaling the problem encountered.

2.5.3 Store personal data

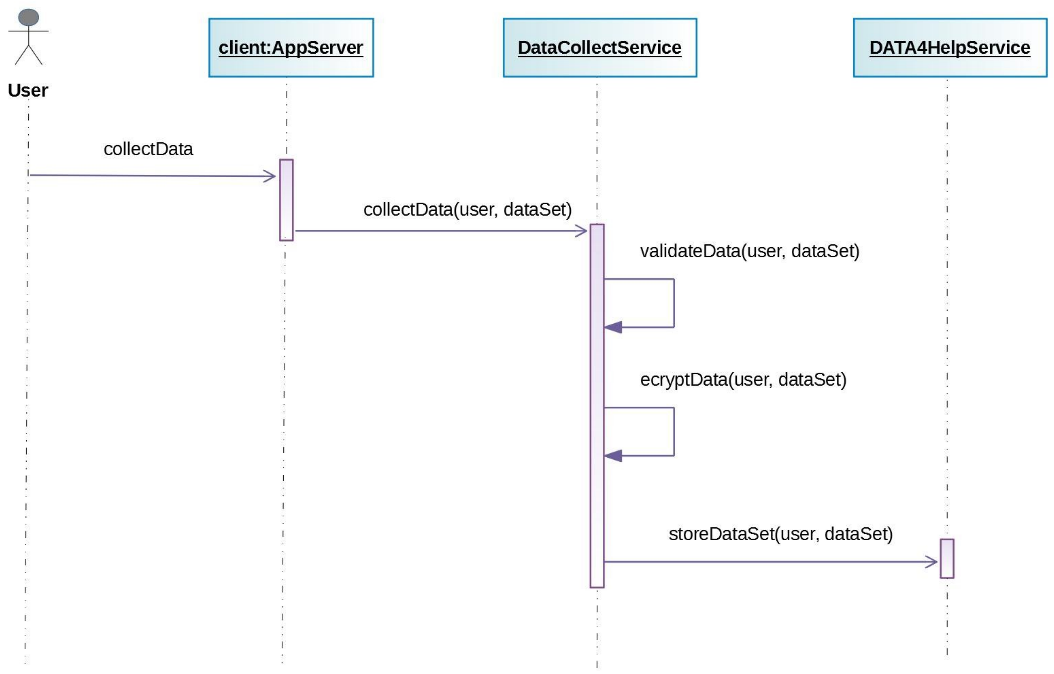

User
client:AppServer
DataCollectService
DATA4HelpService
collectData
collectData(user, dataSet)
validateData(user, dataSet)
ecryptData(user, dataSet)
storeDataSet(user, dataSet)

Figura 6: Sequence Diagram – Store personal data

This sequence diagram describes the process of collecting and storing user's personal data into the database, operation that allow each user to create his own repository containing personal information about him such as health data or location data. In our view this process requires the intervention of two components, namely the DataCollectService and the DATA4HelpService. The first one is designed to acquire user data, collect them in a single dataset, validate them and then encrypt them, while the latter has the task of storing each dataset within the database.

2.5.4 Set Preferences

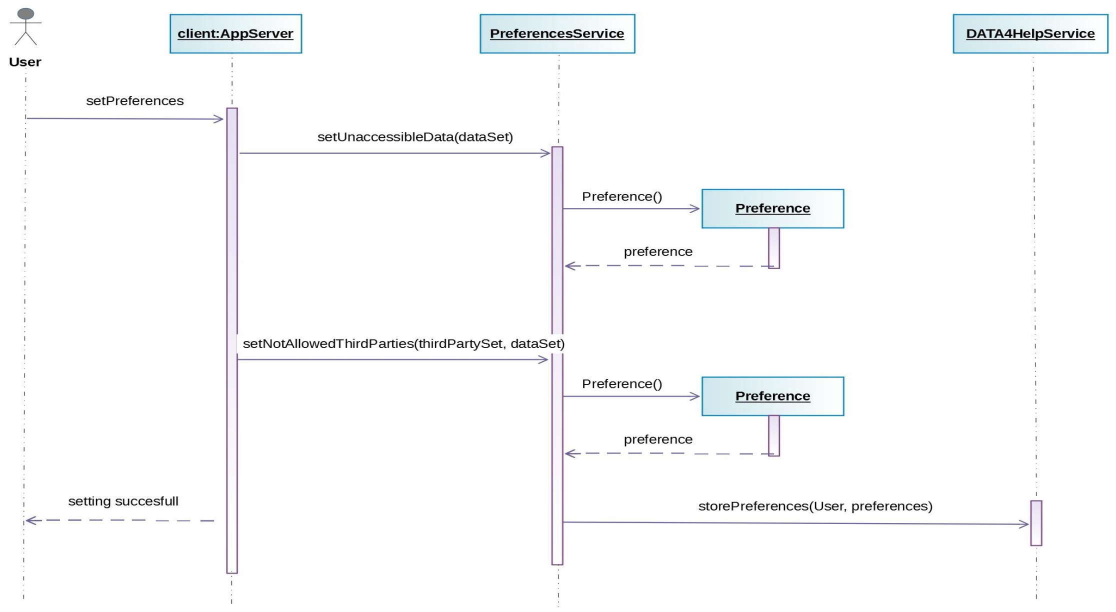


Figura 7: Sequence Diagram – Set Preferences

This sequence diagram represents the flow of events that characterize the process that allows each user to define his own set of preferences. Clearly the component involved in this case is strictly the PreferencesService, which, as seen from the diagram, allows the user to define some of his personal data as inaccessible and to designate certain third parties as not permitted to acquire certain data. Then the DATA4HelpService intervenes and assumes the task of saving in the database the user's set of preferences, in order to ensure that these preferences once expressed will be always present and available from the system.

2.6 Component interfaces

Below are shown the component interfaces involved in the TrackMe architecture. They had already been seen and described in section 2.3, but while in this case we had a more architectural vision of the relationships between them, here we want to focus on the analysis of the methods defined within each interface and of the way in which interfaces communicate with each other through these methods.

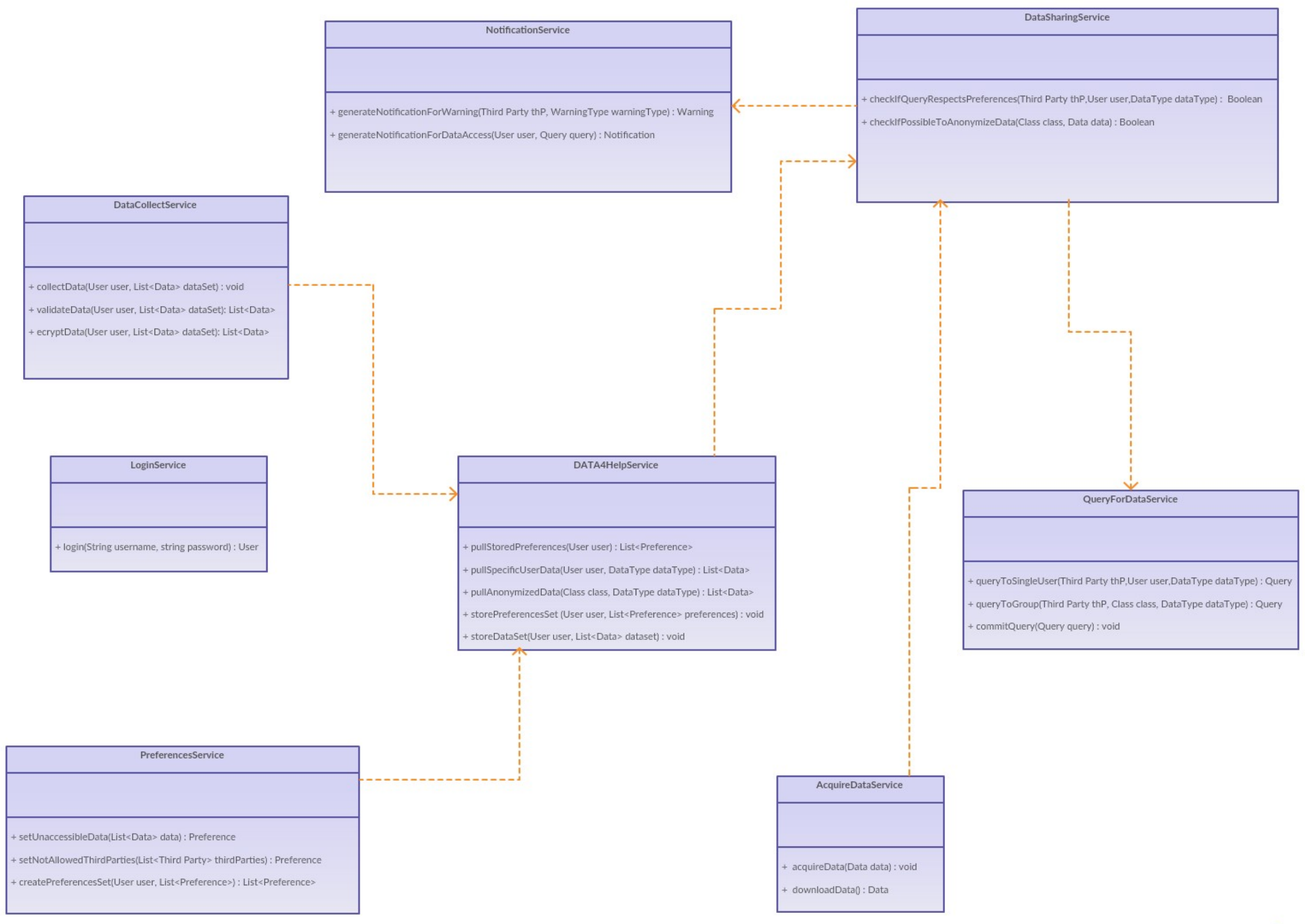

NotificationService
+ generateNotificationForWarning(Third Party thP, WarningType warningType) : Warning
+ generateNotificationForDataAccess(User user, Query query) : Notification
DataSharingService
+ checkIfQueryRespectsPreferences(Third Party thP,User user,DataType dataType) : Boolean
+ checkIfPossibleToAnonymizeData(Class class, Data data) : Boolean
DataCollectService
+ collectData(User user, List<Data> dataSet) : void
+ validateData(User user, List<Data> dataSet): List<Data>
+ ecryptData(User user, List<Data> dataSet): List<Data>
LoginService
+ login(String username, string password) : User
DATA4HelpService
+ pullStoredPreferences(User user) : List<Preference>
+ pullSpecificUserData(User user, DataType dataType) : List<Data>
+ pullAnonymizedData(Class class, DataType dataType) : List<Data>
+ storePreferencesSet (User user, List<Preference> preferences) : void
+ storeDataSet(User user, List<Data> dataset) : void
QueryForDataService
+ queryToSingleUser(Third Party thP,User user,DataType dataType) : Query
+ queryToGroup(Third Party thP, Class class, DataType dataType) : Query
+ commitQuery(Query query) : void
PreferencesService
+ setUnaccessibleData(List<Data> data) : Preference
+ setNotAllowedThirdParties(List<Third Party> thirdParties) : Preference
+ createPreferencesSet(User user, List<Preference>) : List<Preference>
AcquireDataService
+ acquireData(Data data) : void
+ downloadData() : Data

Figura 8: Component Interfaces

2.7 Selected architectural styles and patterns

2.7.1 Architectural Style

As already mentioned in section 2.1, a three tier architecture has been chosen for TrackMe. At the highest level there is the Presentation Tier, which aims to interface directly with the client, providing him with information and results related to services such as PreferencesService, DataCollectService, QueryForDataService, etc. At an intermediate level there is the Application Tier, in which the main operations and functionalities of TrackMe are performed, thus playing a major role within the application. Finally there is the Data Tier, prepared for storing permanent data and all the informations that will then be requested by the App tier. In the implementation, integration and testing phase we will take into account this style of architecture and we will adopt a bottom up approach, by starting from the lowest level (Data Tier), then moving to the App Tier, until getting to the Presentation Tier.

2.7.2 Design Patterns

- **MVP (Model View Presenter):**

We decide to use, in order to architect our software, the MVP pattern. MVP is a user interface architectural pattern that can be seen as a derivative from the known MVC (Model View Controller). This pattern allows separating the presentation layer from the logic and this means that everything from how the interface works to how it represents on the screen. MVP eases automated unit testing and it is responsible to provide clean code. It is composed by three layers with three different roles:

The Model:

- It represents the data layer, which holds the business logic as well as controls how data is created, stored and modified. In Android, it is a data access layer, for example, database API or Remote server API.
- The Model consists of components that are responsible for functionalities like for generating, storing, exposing and fetching the data.
- All these functionalities usually perform in the background thread, because they could be time-consuming, so they can potentially block the main thread UI.

The View:

- It is a passive interface, which displays data, and the routes user actions to Presenter.
- It contains a visual part of the application.
- It does not contain any logic or knowledge of the displayed data.

The Presenter

- The Presenter is the layer that stands in between the Model and the View. Its principale role is to triggers the business logic, and to let to know the View when to update.
- It recovers data received from the Model and shows it in the View.
- It interacts with the Model, then fetches and transforms the data from the Model to update the view.

There are few important things to understand about this pattern, which are:

1. The View never communicates with the Model directly.
2. It's the Presenter's responsibility to communicate with the Model and update the View accordingly.
3. View only accepts input from the user and sends it to the Presenter.

Clearly there is separation of responsibilities, and since the View and the Presenter refer each other through interfaces, the code is easily testable.
It is also interesting to say that MVP is one of the patterns which Android community prefers at this time. In Android, the application should be easily extensible and maintainable. Therefore, in order to maintain the level, it is important to define separated layers well. MVP makes things easier for developers and it makes the views independent of the data source.

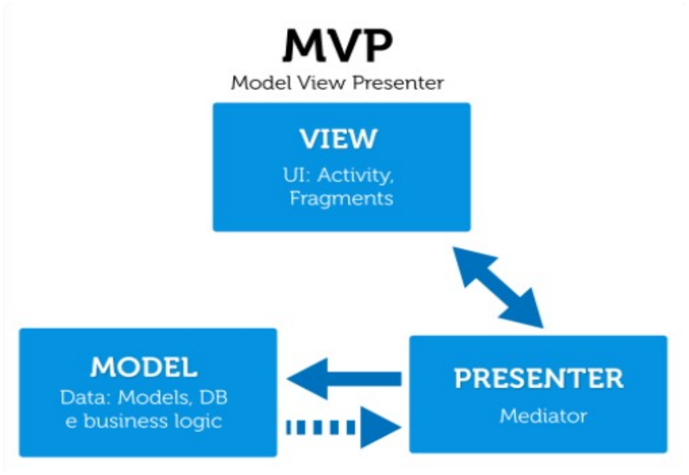


Figura 9: MVP Architectural Pattern

3. User Interface Design

This topic has already been discussed within the RASD document and more precisely in section 3.1.1, in which, although no real user interface for TrackMe App have been really developed and reported, the latter have been described in detail, therefore it is not necessary now to report other characteristics or other interesting aspects concerning this topic.

4. Requirements Traceability

The purpose of the DD document is to define and analyze in detail an architecture for TrackMe app that has to respect the study of the model proposed in the RASD document. In this sense all the goals and the requirements previously defined in the RASD must necessarely be accomplished by the architectural asset. Therefore, in order to ensure that this happens and that there will be coherence between the two documents, below are reported the goals and requirements already identified in the RASD document, specifying which design components are in our vision prepared for their achievement.

[G1] TrackMe allows users and third parties'employees to be recognized by providing a method of identification (**[R1]**)

- LoginServiceImpl

[G2] DATA4Help has to work as a cloud datastore service, in which each user can be associated with a private repository prepared to collect user's personal data, in the total respect of the GDPR. (**[R2]**)

- DataStorageServiceImpl
- DATA4HelpServiceImpl

[G3] The user is guaranteed to have access to his stored data at any time, in order to monitor and possibly modify them. (**[R3]-[R4]**)

- DataStorageServiceImpl
- DATA4HelpServiceImpl

[G4] Each user can express preferences that will be associated with him and stored by DATA4Help in his personal repository. This preferencs may concern the user's will to make his personal data not accessible or only partially accessible or to define certain third parties as not allowed or only partially allowed to access some data. (**[R5]-[R6]**)

- PreferencesServiceImpl

[G5] TrackMe allows third parties registration (**[R7]**)

- LoginServiceImpl

[G6] Third parties can ask for data relating to a specific individual (**[R8]-[R9]**)

- QueryForDataServiceImpl
- DataSharingServiceImpl
- DATA4HelpServiceImpl
- NotificationServiceImpl

[G7] Third parties can ask for anonymous data relating to a group (class) of individuals (**[R10]-[R11]-[R12]-[R13]**)

- QueryForDataServiceImpl
- DataSharingServiceImpl
- DATA4HelpServiceImpl
- NotificationServiceImpl

5. Implementation, integration and test plan

5.1 Implementation plan

At this point the architecture of the Trackme system has been defined and analyzed, and also themes concerning its design have been addressed, so we want to give a perspective view of the implementation of this application. Even if actually TrackMe will not be implemented by us, this section aims to define an implementation plan, specifying in particular in which order the components involved in the TrackMe architecture will be implemented, studying this process module by module.
The order of implementation of the components is a choice that depends on the functionalities of these components and on the importance they cover in the system. Clearly, each component defined in a software architecture has its relevance, but it will always be possible to identify some particularly crucial ones, from which others depend and which must therefore take precedence during implementation.
Below are listed the components involved in TrackMe, grouped and reported according to the order in which we want to implement them:

1. DATA4HelpService

2. DataSharingService and DataCollectService

3. LoginService, PreferencesService, NotificationService, QueryForDataService and AcquireDataService

4. Client

In our idea DATA4HelpService is the first component that has to be implemented, precisely for the role it covers, i.e. a service that totally manages the application database, allowing saving data related to users both inherent to the system, as login credentials and preferences, both concerning the life of the users themselves, for example health data and location data, and also allowing access to the data stored in the database, for their subsequent transmission to third parties interested in their acquisition. It is a service that covers the roles of a DBMS and is therefore a primary component,

involved in all the main operations performed within the application and to which services such as LoginService, PreferencesService, DataCollectService and DataSharingService communicate directly.
The second group consists of DataSharingService and DataCollectService. These components offer the two main services provided byTrackMe. The fisrt one allows the user to request the storage of information concerning his person, while the latter deals with the total management of third prties's requests concerning data access. Both of these components have the peculiarity of communicating directly with DATA4Help, to request write access (data entry) and read (data acquisition) respectively. DataSharingService in particular is one of the most complex components within the application, due to the many features that are entrusted to it, such as handling queries coming from third parties, accessing to the database not only to acquire data but also to check both the preferences expressed by users in case of queries to a specific individual and the cardinality of a class in case of requests to groups of individuals. It also has the task of preparing the notification to both third parties and registered users. It is therefore a risky component from the point of view of implementation time and for this reason it must be implemented as soon as possible.
The third group includes services that are closer to the client's part, allowing the latter to take advantage of all the features offered by TrackMe. The LoginService allows both a normal user and a ThirdPartyEmployee to access the application, while the NotificationService sends warnings or notifications regarding data sharing to the generic client; the PreferencesService allows the normal users to define their own set of personal preferences and the DataCollectService to request the saving of their personal data; the QueryForDataService allows a third party (through its employee) to perform a query for access to certain data, while the AcquireDataService allows data download. These are all simpler operations compared to those performed by the components seen before and therefore also faster and less risky to implement.
The fourth group consists of components that are the closest to the client side and that are used to redirect his requests.

It can be noted that the implementation plan has a bottom-up approach, starting from the lowest level in charge of managing data and permanent informations, up to the highest level that interfaces directly with the real client.

5.2 Integration and test plan

The order to be followed in the process of integrating the components will necessarily have to respect the order of implementation and what was said in the previous section.
Based on this we define the following integration order:

1. Integration of components with DATA4HelpService (DBMS)

2. Integration of the components of the application server

3. Integration of the client (mobile application) and the application server

As far as the test plan is concerned, we first propose that the Unit Test is performed for each TrackMe component, in order to validate that each unit of the software performs as designed and once this step is completed we propose to perform an Integration Test, in order to expose defects in the interfaces and in the interactions between integrated components. Regarding this test, we propose a bottom-up approach that is well suited to the TrackMe architecture.

6. Reference Documents

- Xiaochen Zheng, Raghava Rao Mukkamala, Ravi Vatrapu1, Joaqun Ordieres-Meré. *Blockchain-based Personal Health Data Sharing System Using Cloud Storage*, 2018 IEEE 20th International Conference on e-Health Networking, Applications and Services (Healthcom)

- Chuan Fu, Jun Yang, Zheli Liu and Chunfu Jia. *A Secure Architecture for Data Storage in the Cloud Environments*, 2015 9th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing

- Alatishe A.A, Adegbola M.A, Dike U. Ike. *DESIGN AND IMPLEMENTATION OF A FILE SHARING APPLICATION FOR ANDROID*, International Journal of Computer Science Engineering (IJCSE)

- Slides – “Software Design & Software Architecture.pdf”

- Slides – “Tips and Tricks for Supporting Architecture Description with UML.pdf”