Data4Help

Design Document

V1.0

Davide Damato   Luciano Franchin

December 10, 2018

POLITECNICO MILANO 1863

Contents

1 Introduction

1.1 Purpose

This document represent the Design Document(DD) for Data4Help. The purposes of this document are: to give more details than the RASD, to analyze technical aspects about system architecture software desings, to guide the development team on the high level decisions that has been made for the project.

1.2 Scope

In order to address design concerns regarding software development, this document will provide an high level representation of the main components and the design pattern to be used. In the next section graphs and diagram will show components integration and data flow to visualize how the software should be developed. This document does not want to represent the final architecture of the system and/or the final implementation of the software. The concepts shown in this document are purposely intended to describe components generically, the specific aspects will be discussed with the technical team during development.

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Acronyms

PU:	Public User.
TPU:	Third Party User.
OS:	Operative System.
API:	Application Programming Interfaces, a set of functions and procedures that allow the creation of applications which access the features or data of an operating system, application, or other service.
MVC:	Model View Controller, design pattern described in *2.6 Selected architectural styles and patterns*
GPS:	Global Positioning System.
GSM:	Global System for Mobile communications.

1.3.2 Abbreviations

Gn:	n-Goal.
web-dev:	internet web pages development.

1.4 Revision History

[V1.0]: First release.

1.5 Reference Documents

- Lecture slides.
- Specification Document:
  "Mandatory Project Assignment AY 2018-2019.pdf".
- 1016-2009 - IEEE Standard for Information Technology–Systems Design–Software Design Descriptions

1.6 Document Structure

This Design Document is composed by 7 parts:

1. **Introduction:** First part explains what are the purposes of the document and about what it concerns. This section will also be useful to better comprehend the document, its definitions, its documentation and how it is conceived.
2. **Architectural design:** Second part describes how the system is designed. This section provide a description of the main components that make up the system and how the main activities are carried out.
3. **User interface design:** Third part shows mainly the existing mockups that were already defined in the RASD.
4. **Requirements traceability:** Fourth part maps RASD's defined goals with DD's defined components.
5. **Implementation, integration and test plan:** Fifth part explains in detail how the implementation of the system is performed, how much every component should be developed before integration and in which sequence the components should be developed and integrated.
6. **Effort spent:** Sixth part contains a detailed report of the hours spent to redact this document.
7. **References:** Seventh part lists all the tools used to redact this document.

2 Architectural Design

2.1 Overview

The system can be represented overall with a multitier architecture. This representation shows a three-tier architecture divided in: presentation layer, application layer and data layer.
Presentation layer, composed of blocks in red, is the system interface which interacts with the users. As specified in the RASD document, Data4Help refers mainly to three kind of users: PU, TPU and System administrators. For each user it is shown which is the best interface to use and to implement.
Application layer, composed of block in blue, acts as the core of the system, in the physical architecture the components of this layer will carry out most of the system functionalities. As it can be seen in the graph, this point represents the bottleneck of the architecture and it will require particular attention and maintenance.
Data layer, composed of block in green, is the main source of the data flow. In this layer there will be mainly the databases and the DBMS components, it will be sized in relation to the amount of user that will access the system.

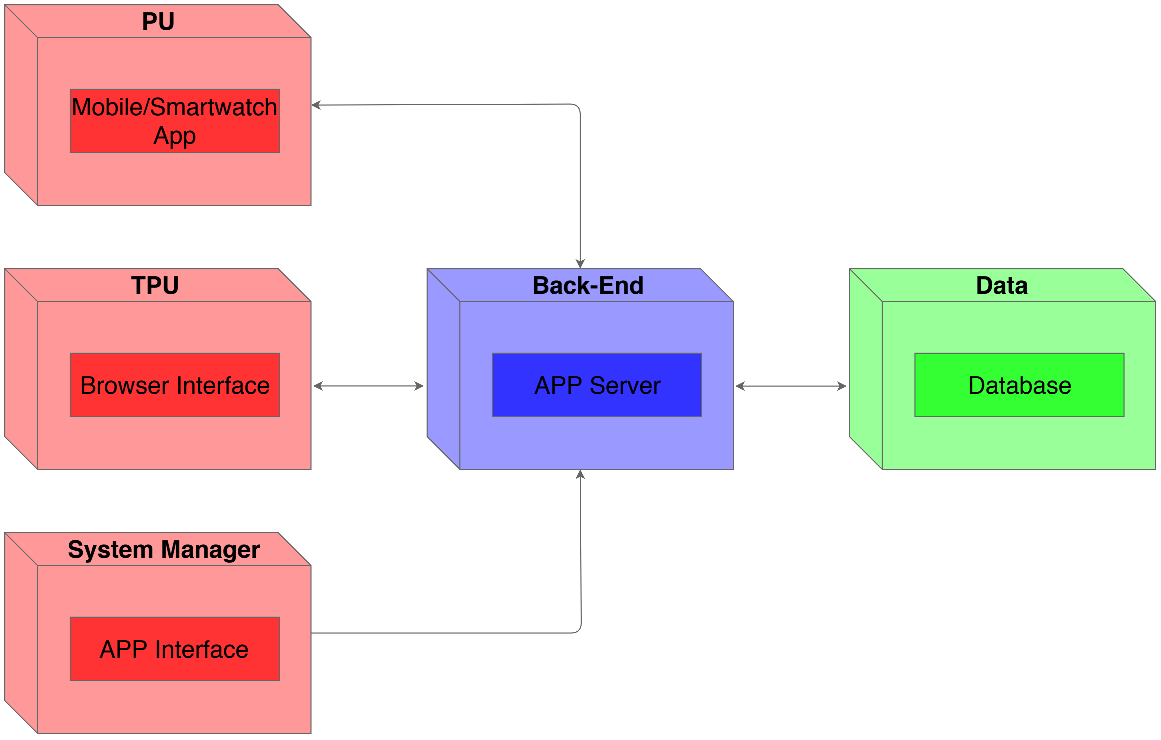


Figure 1: Logical layer architecture

High level architecture In this schema it is shown generically how the main system architecture should be intended, this does not represent a detailed or final physical component architecture. The colored boxes visualize how the three-tier architecture reflects on the physical architecture. User actors are outside of the boxes as they do not perform an active role on the system architecture to maintain its functioning.

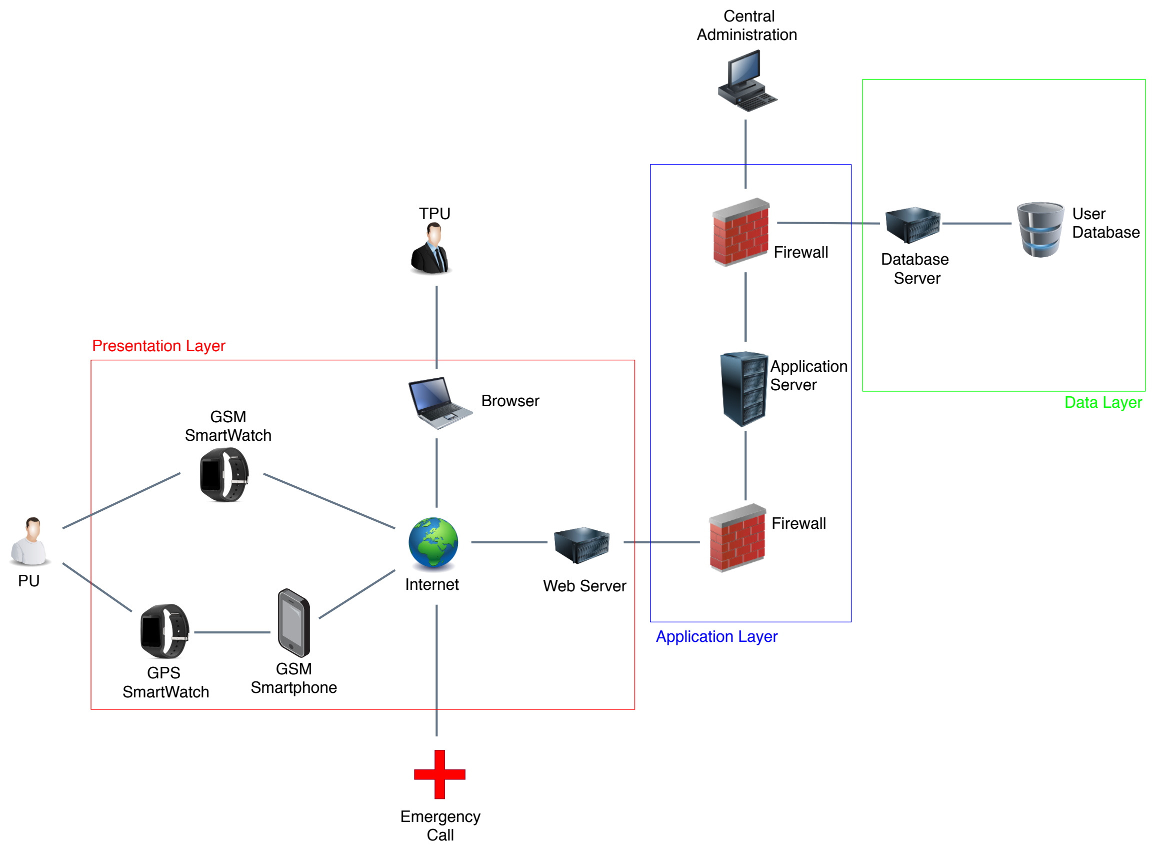
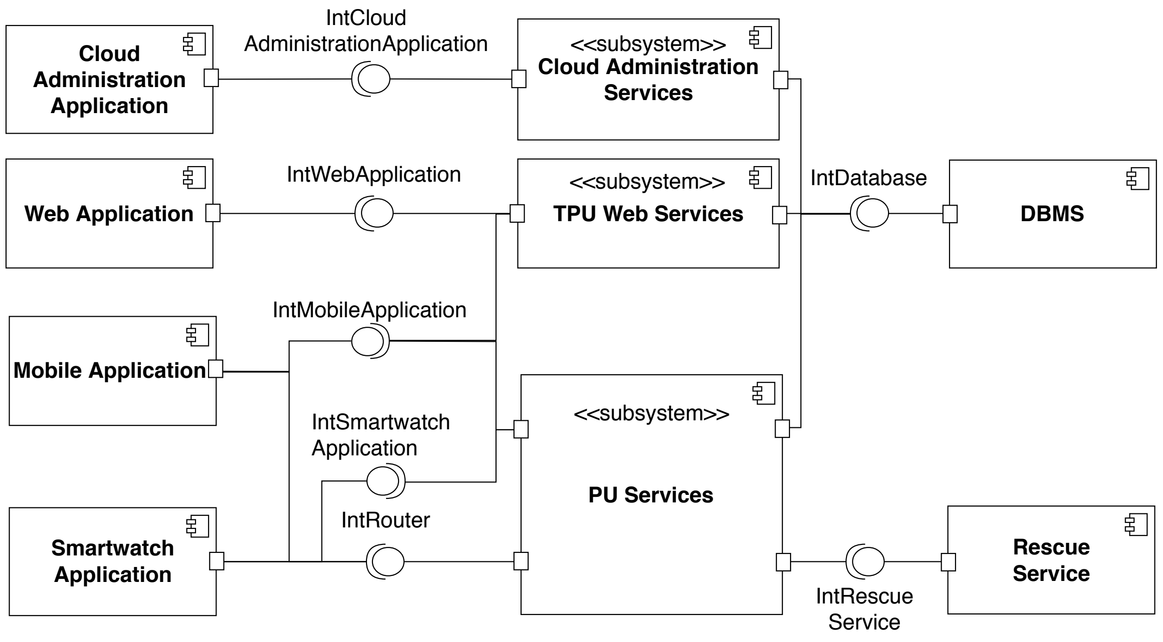


Figure 2: Graphical representation of the high level architecture

2.2 Component view

2.2.1 High level component view

The system can be divided in a nine high level logical component. These are meant to represent the main part of the system in which the functionalities will be divided. The component separation is useful to focus on how the procedures are assigned and to consider if some component can be bought from external developers.



- **Web application:** this component is meant to be a web page to allow TPU to perform queries. As shown on the mockups in section *3 User interface design*, TPU should be able to perform any possible query by connecting to Data4Help web page. The development of this application will be entrusted to an external web-dev company.
- **TPU Web services:** this is an application component that groups every TPU services functionality and interfaces it with the web application. Since this modules contains the main logic of the project it will developed in-house.
- **Mobile application:** this component represents the software application installed on mobile OS. Mobile Application provides all the PU functionalities to the user and plays as his/her main tool to administrate his/her profile. The development of this application will be entrusted to an external mobile applications company.
- **Smartwatch application:** this component represents the software application installed on smartwatch OS. Due to obvious physical limitation, this service may not provide all the functionalities offered by PU services. The development of this application will be entrusted to an external mobile applications company.
- **PU services:** this component plays as the principal back-end module which implements all the PU functionalities required by the RASD document. Since this modules contains some the main logic of the project it will developed in-house.

- **Cloud Administration Application:** this component is an externally developed application interface to manage main administration functions. This interface works on administration devices.
- **Cloud Administration Services:** this component plays the role as the core application of the administration system, it is installed on cloud devices and it offers all the required services to control application and data layer.
- **DBMS:** *DataBase Management System* this component manages every transaction between the components it interfaces and the real Database where all the data is stored. In this graph it is represented by only one module but it in the reality it may be composed by different computing structures.
- **Rescue Service:** this component groups every outside first aid service that will be linked with the AutomatedSOS system. Anything of this component won't be developed in-house, its presence on the graph is usefull just to show the flow of information. The development of this component will be entrusted to an external company that will equip every rescue service with the required tools to work on Automated SOS system.

2.2.2 Component view

After analyzing the main components of the system on the *2.2.1 High leve component view*, **TPU Web Services** and **PU Services** will be analyzed more in detail in this section. The graphs shown below illustrates what are the components inside both of the modules developed in-house. These two pictures should not represent a final composition of the modules of interest, they're meant to visualize the subdivision of the functionalities, which jobs are suitable for some components and how the system logic is implemented. The first component to be analyzed is the PU Services; its main role is to manage every operation regarding PU and AutomatedSOS functionalities, it is composed of 6 internal components and all of them are in-house developed.

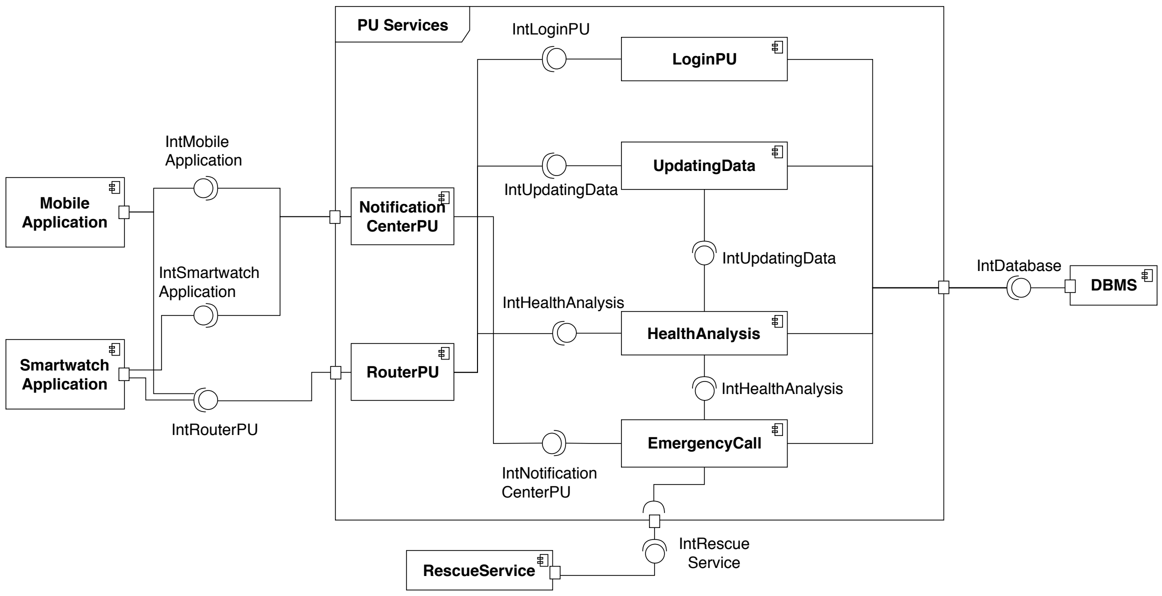


Figure 3: **PU Services** component view

- **Router:** receives all the request from outside the module and directs them to the right destination.
- **LoginPU:** offers the common login functionalities: registration, password change and user profile settings.
- **UpdatingData:** main component that manages the data flow between user and the database. It can be triggered automatically by regular body measurements performed by the measuring devices or manually by the user.
- **HealthAnalysis:** crucial component that checks user's health condition and interacts with EmergencyCall.
- **EmergencyCall:** this component receives the emergency signal and critical user's health parameters, subsequently it makes the emergency call.
- **NotificationCenter:** this component receives messages from PU Services' components and delivers it to Mobile Application and Smartwatch Application.

The second component to be analyzed will be TPU Web Services; its main role is to manage every operation regarding TPU and Data4Help functionalities, it is composed of 5 internal components and all of them are in-house developed.

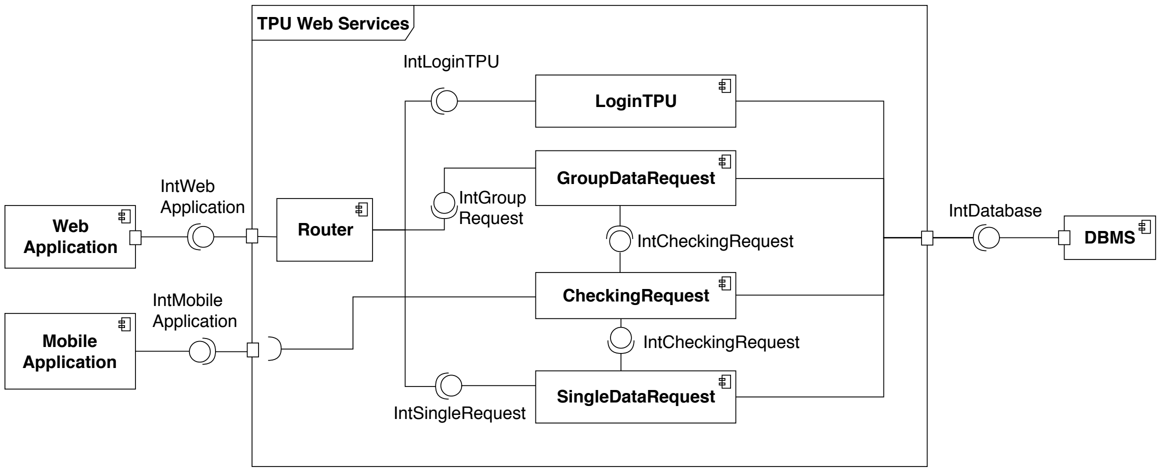


Figure 4: **TPU Web Services** component view

- **Router:** receives all the request from outside the module and directs them to the right destination.
- **LoginTPU:** offers the common login functionalities: registration, password change and user profile settings.
- **CheckingDataRequest:** receives all the elaborated queries and checks if they respect the *Term of use*. This module represents the core of the query functionalities and constitute the Data Protection Supervisor of the entire system.
- **GroupDataRequest:** Data4Help module that performs all the required group queries. This modules relies on CheckingDataRequest to validate all the processed query.
- **SingleDataRequest:** Data4Help module that performs all the required single person queries. This modules relies on CheckingDataRequest to validate all the processed query.

2.3 Deployment view

In this section we analyze how the services should be deployed on the system architecture. This graph shows the devices component that should implement the required functionalities. The components are grouped by layers to reflect the three-tier architecture of the *2.1.1 High level architecture*.

- **Presentation tier** composed by: Smartphone, Smartwatch, Personal computer and GSMSmartwatch. This layer interacts directly with the user and represents his/her direct connection with the application layer. Biosensors are crucial to implement AutomatedSOS service and all the devices that carry them are required to implement the rescue functionalities. A-GPS represents another core component of this architecture as they're crucial to develop AutomatedSOS.
- **Application tier** composed by: PU Application Server and TPU Application Server. This layer can comprehend Firewalls and Web Server but these components execute merely an architectural role. PU Application is required to process all the data of the PU functionalities; the data flow through this component can be considered to be quite heavy and the component has to be adequately clustered to support all the traffic. TPU Application Server manages the flow of the query operation. This component is useful to lighten the PU Application Server connection to the DBMS.
- **Data layer** composed by: Database Server. This component compose one of the heaviest and expensive structure of the system and may be rented from an external web service provider.

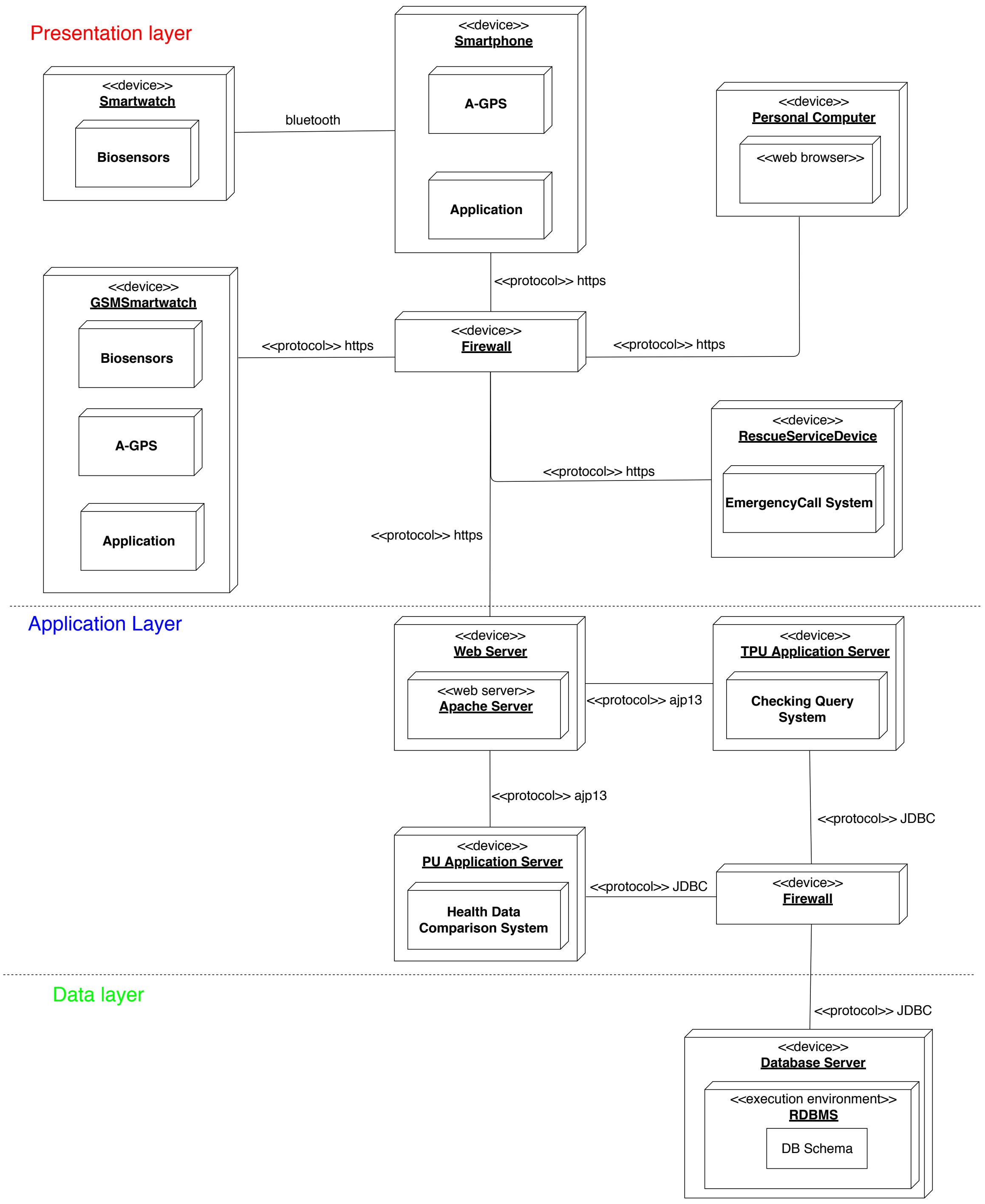


Figure 5: Deployment Diagram

2.4 Runtime View

Software developers must know how most of the component interacts with each other and in which order every actions is executed. In this section there are shown a series of sequence diagrams to explain how the Data4Help components work.

2.4.1 PU's login

This section shows the initial PU's login operation. Three components of the system are used in addition to MobileApplication to execute this functionality. Router forwards the login request with PU's credentials to LoginPU that after sending them to the database for a check, based on the outcome of the check, will allow the user to log in.

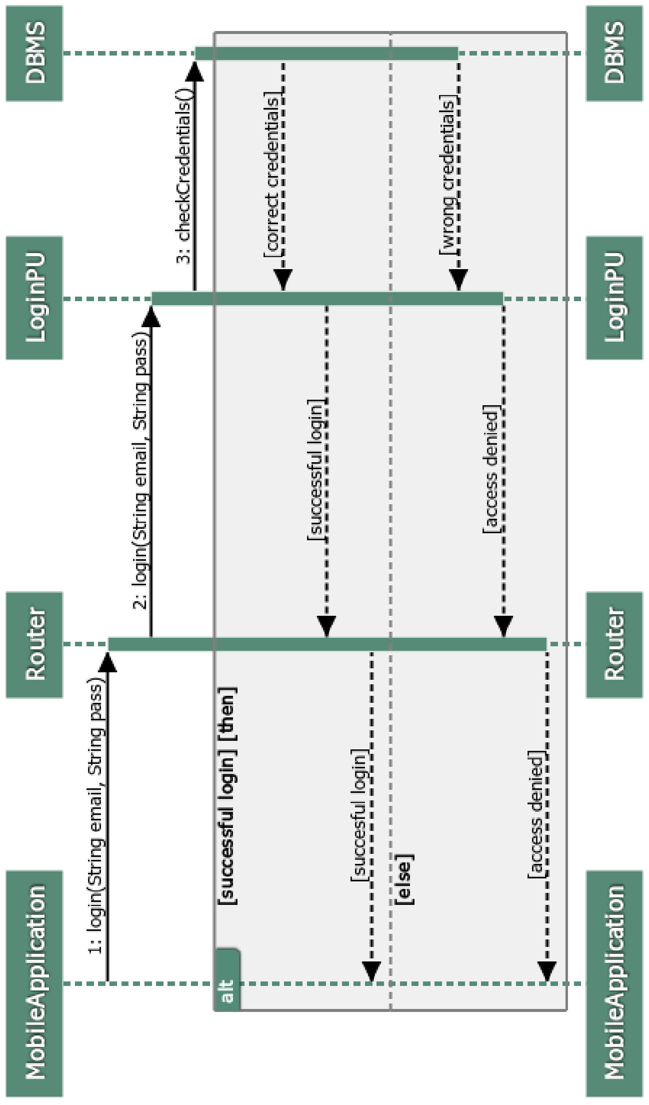


Figure 6: PU's login

2.4.2 TPU's login

TPU's login operations sequence and components mostly identical to PU's login.

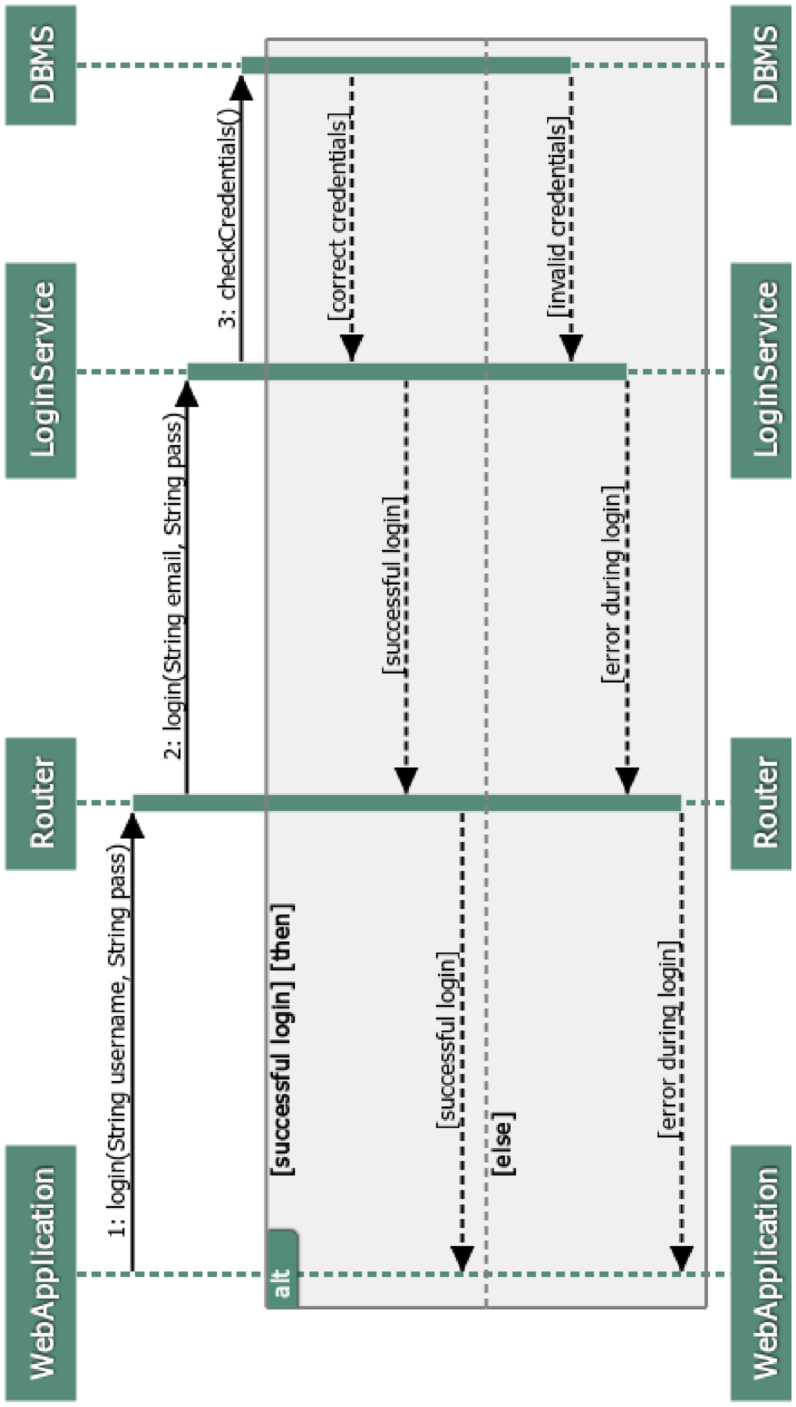


Figure 7: TPU's login

2.4.3 PU's health data update

One of the most recurrent and important operation in the Data4Help system is the transfer of PU's data from his/her personal device to the Data4Help system. This transaction involves three different main components: Mobile Application or Smartwatch Application, PUServices and DBMS. Inside PUServices, Router and UpdatingData are involved to process the data flow. In the graph shown below it appears only the Mobile Application component, this should not limit the development to this component because it refers both to Mobile Application and Smartwatch Application. Another graph is not included to avoid excessive redundancy.

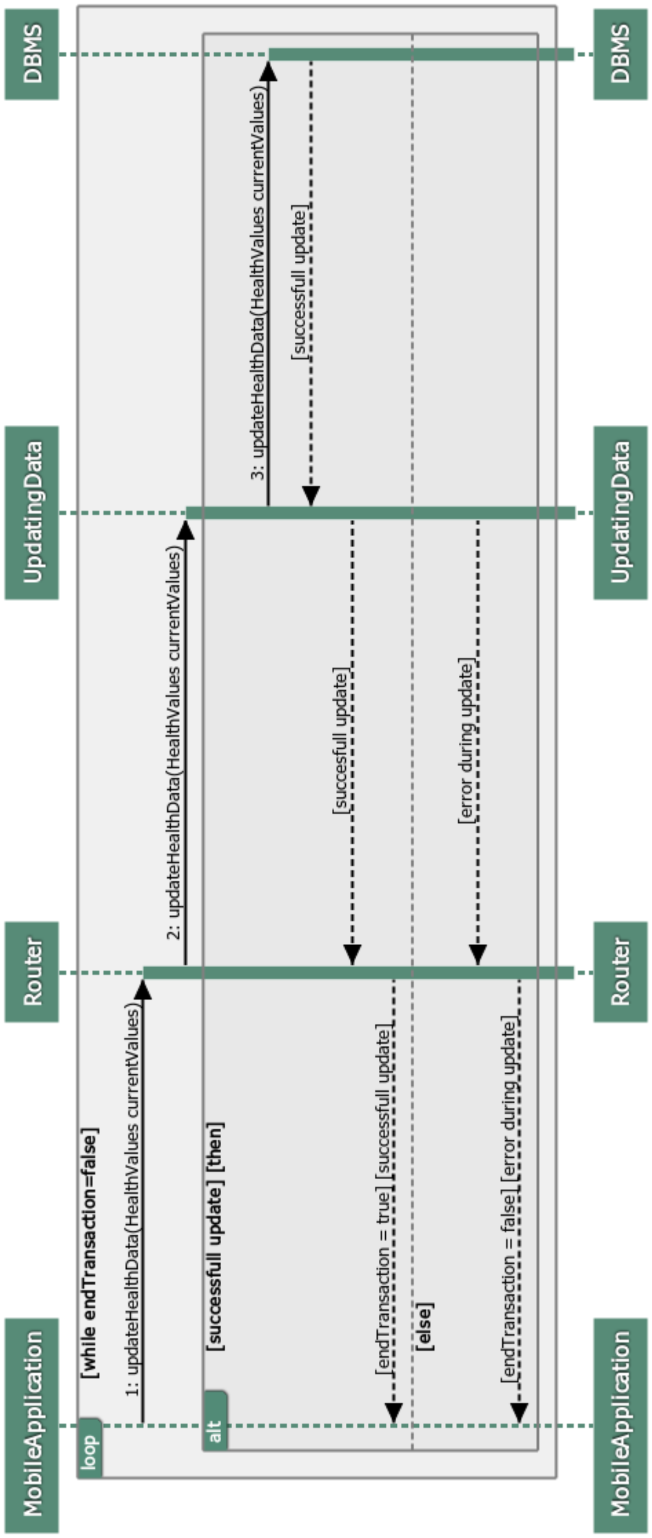

MobileApplication
Router
UpdatingData
DBMS
loop
[while endTransaction=false]
1: updateHealthData(HealthValues currentValues)
2: updateHealthData(HealthValues currentValues)
3: updateHealthData(HealthValues currentValues)
[successfull update]
alt
[successfull update] [then]
[succesfull update]
[endTransaction = true] [successfull update]
[else]
[error during update]
[endTransaction = false] [error during update]

Figure 8: Updating and automatic saving of the user's health data

2.4.4 PU's personal information update

Once the PU is registered to Data4Help he/she may need to change his/her information about body and health measurement. To allow this functionality Mobile Application interacts with PUServices which interacts in turn with DBMS. Router and Login components are involved inside PUServices.

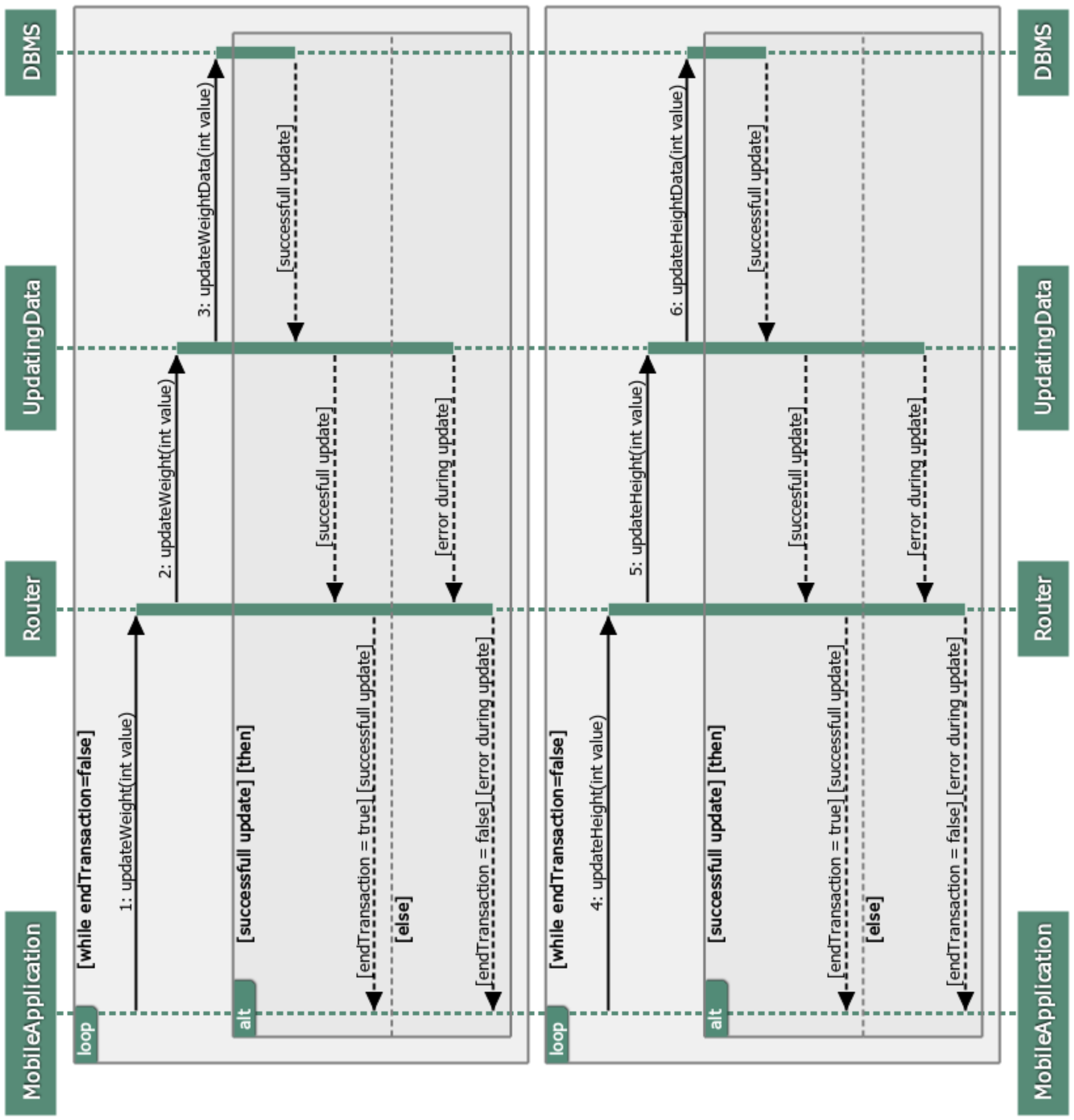

MobileApplication
Router
UpdatingData
DBMS
loop
[while endTransaction=false]
1: updateWeight(int value)
2: updateWeight(int value)
3: updateWeightData(int value)
[successfull update]
alt
[successfull update] [then]
[succesfull update]
[error during update]
[endTransaction = true] [successfull update]
[else]
[endTransaction = false] [error during update]
4: updateHeight(int value)
5: updateHeight(int value)
6: updateHeightData(int value)

Figure 9: Updating of data by the user

2.4.5 Emergency call

This sections analyzes one of the most crucial and delicate operation of the system. Emergency call describes the situation in which a PU health status is below threshold and requires an immediate intervention of first aid service. This function has to be accurately developed since it requires high reliability. In the sequence there are involved three main components: Mobile Application, PUServices and Rescue Service. Inside PUServices four others components are involved: Router, UpdatingData, HealtAnalysisService, EmergencyCall and Notification Center.

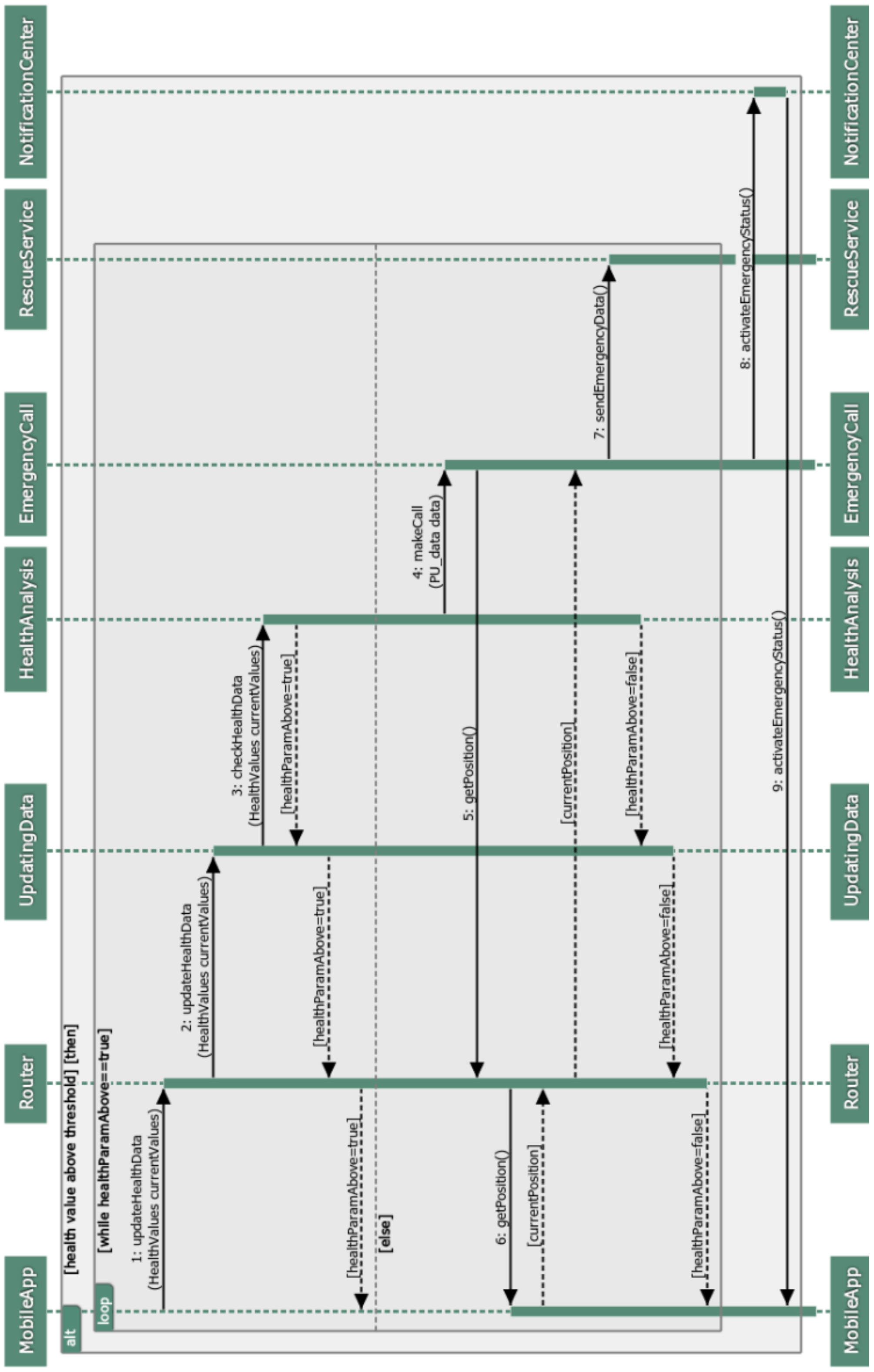


Figure 10: Health data analysis and management of the emergency situation

2.4.6 TPU single person query

One of the two possible query available for TPUs is the single person query. This kind of query shows to the TPU every data about a specific PU, to do this TPU must have the SSN of the concerned PU. As every operation that deals with PU's data, it must respect the PU privacy and the terms of use. To guarantee privacy protection, this operation requires data access confirmation directly from the interested PU. To develop this functionality Web Application interacts with Router inside TPUwebServices; Router sends the request to SingleDataRequest which analyzes it and sends it to CheckingRequestService.

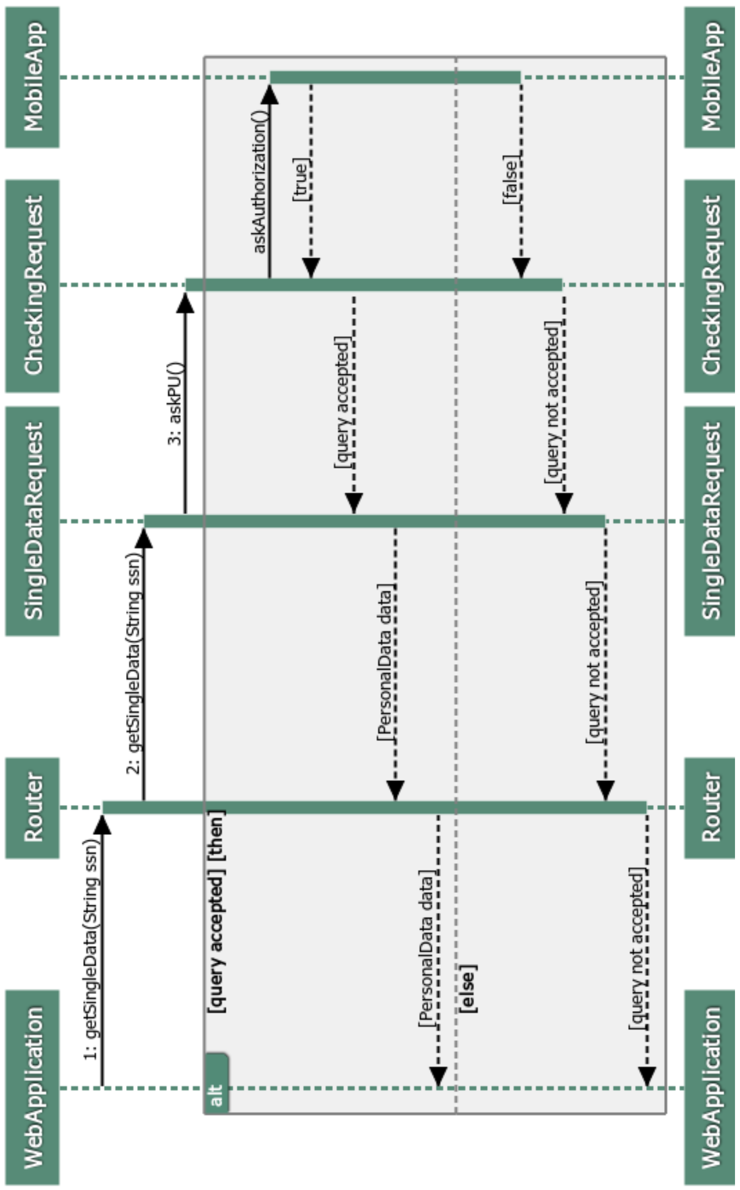


Figure 11: Single data query by TPU

2.4.7 TPU group data query

In this section there will be explained an operation which is thought to be one of the most used and useful. Group data query can be the most productive functionality to TPU since it can assemble huge amounts of data and it can show refined information. Group queries requires to respect privacy and term of use too, to guarantee this requirement every query must be accepted and analyzed by a specific module. In this operation there are involved three main component: Web Application, TPUwebServices and DBMS. To develop the data flow inside TPUwebServices, three component are used: Router, GroupDataRequest and CheckingRequestService.

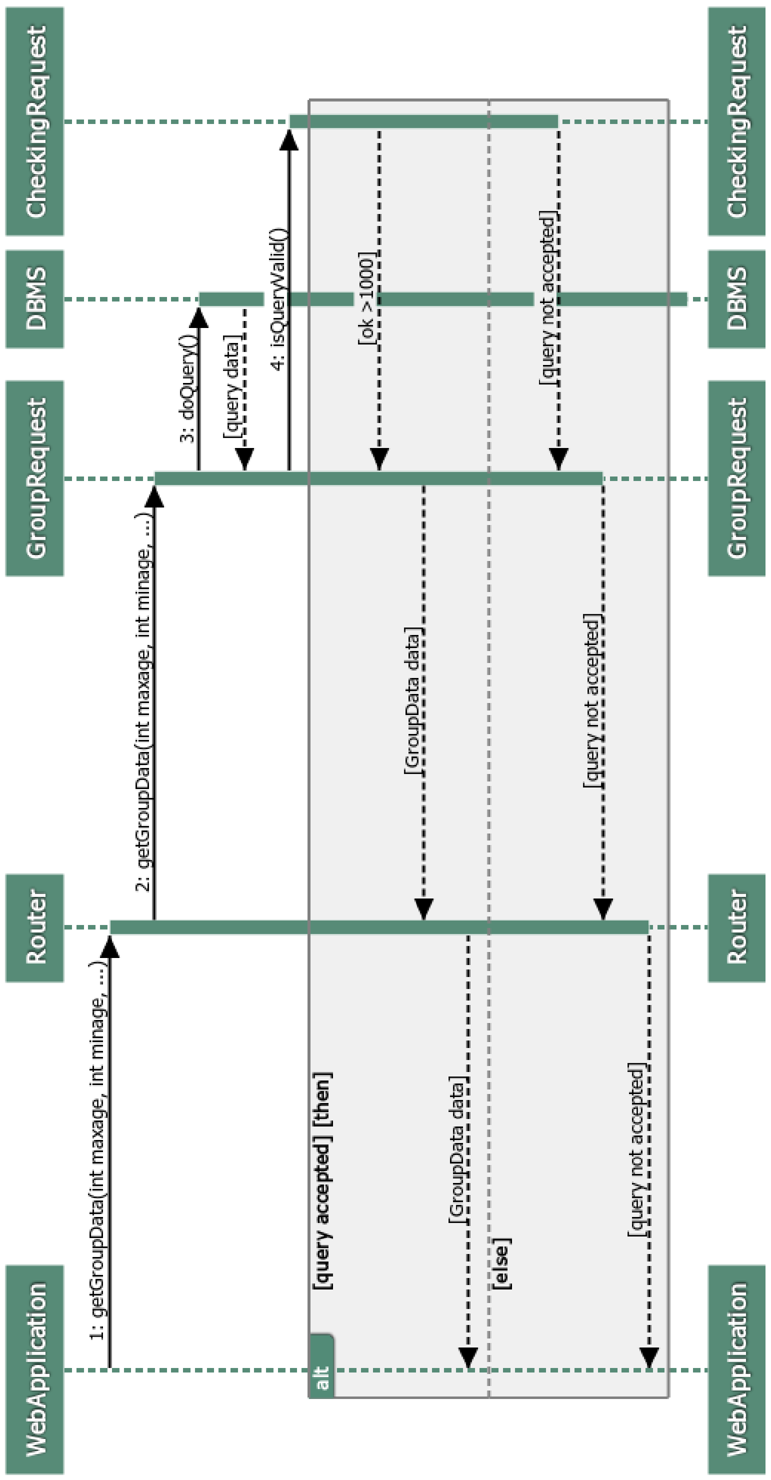


Figure 12: Group data query by TPU

2.5 Component interfaces

2.5.1 PU component interfaces

- **IntLoginPU** interface, which represents the way components can interact with the LoginPU component, exposes the following methods:
  - login(String username, String password): analyzes the username and password inserted by PU and controls if they match with the data stored on the database.
  - modUsername(String new): checks if the new username is available with a query on the DBMS, if so changes the username field on the database.
  - modPass(String newp): checks if the password respects the security standards, if so proceeds to change the password field on the database.
- **IntUpdatingData** interface, which represents the way components can interact with the UpdatingData component, exposes the following methods:
  - updateWeight(int value): allows to change the personal weight on the database, the method checks if the value fits in minimum and maximum value range then proceeds on updating the weight database field.
  - updateHeight(int value): allows to change the personal height on the database, the method checks if the value fits in minimum and maximum value range then proceeds on updating the weight database field.
  - updateHealthData(HealthValues currentValues): this method is called whenever the PU Services receives updated health measurement from the Biosensors attached to the user body. First of all these data have to be sent to the HealthAnalysis module, then they can be transferred to the DBMS to store them.
- **IntHealthAnalysis** interface, which represents the way components can interact with the HealthAnalysis component, exposes the following methods:
  - showParameters(): this is used to elaborate the PU health data and delivers them when requested.
  - checkHealthData(HealthValues currentValues): this method checks if the current health values are below or above the threshold. If the health values are below the threshold EmergencyCall is triggered and the emergency procedure starts.
- **IntEmergencyCall** interface, which represents the way components can interact with the EmergencyCall component, exposes the following methods:
  - sendEmergencyData(): this method immediately send every important PU information to the first aid service in order to dispatch the rescue team.

  IntNotificationCenter interface, which represents the way components can interact with the NotificationCenter component, exposes the following methods:
  - getPosition(): returns the current position of the PU from the GSMSmartwatch or from the GSMSmartphone.
  - activateEmergencyStatus(): this method is used to trigger the emergency status in the PU's devices and abilitate every function to make facilitate the PU finding.

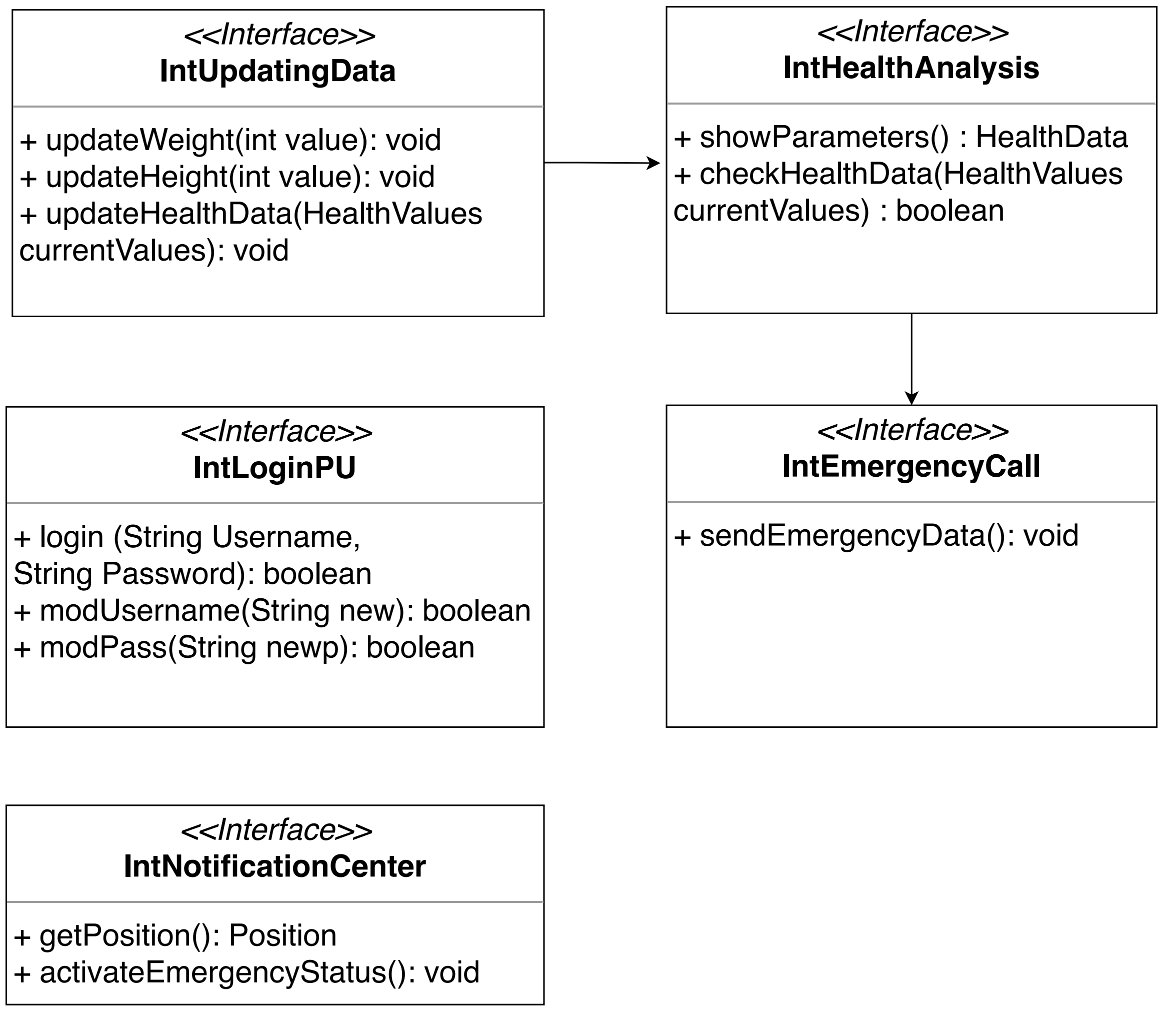

<<Interface>>
IntUpdatingData
+ updateWeight(int value): void
+ updateHeight(int value): void
+ updateHealthData(HealthValues currentValues): void
<<Interface>>
IntHealthAnalysis
+ showParameters() : HealthData
+ checkHealthData(HealthValues currentValues) : boolean
<<Interface>>
IntLoginPU
+ login (String Username, String Password): boolean
+ modUsername(String new): boolean
+ modPass(String newp): boolean
<<Interface>>
IntEmergencyCall
+ sendEmergencyData(): void
<<Interface>>
IntNotificationCenter
+ getPosition(): Position
+ activateEmergencyStatus(): void

Figure 13: PU component interfaces

2.5.2 TPU component interfaces

- **IntLoginTPU** interface, which represents the way components can interact with the LoginTPU component, exposes the following methods:
  - login(String username, String password): analyzes the username and password inserted by TPU and controls if they match with the data stored on the database.
  - modUsername(String new): checks if the new username is available with a query on the DBMS, if so changes the username field on the database.
  - modPass(String newp): checks if the password respects the security standards, if so proceeds to change the password field on the database.
- **IntGroupRequest** interface, which represents the way components can interact with the GroupDataRequest component, exposes the following methods:
  - getGroupData(int maxAge, int minAge, int maxWeight, int minWeight, boolean gender, String region, String city, String street): receives the query paramaters from the TPU

Web Application and delivers them to the DBMS. If the query is accepted returns the results to the TPU.

– getGroupData(int maxAge, int minAge): receives the query paramaters from the TPU Web Application and delivers them to the DBMS. If the query is accepted returns the results to the TPU.

– getGroupData(int maxWeight, int minWeight)receives the query paramaters from the TPU Web Application and delivers them to the DBMS. If the query is accepted returns the results to the TPU.

- **IntSingleRequest** interface, which represents the way components can interact with the SingleDataRequest component, exposes the following methods:
  - getSingleData(String SSN): receiver the SSN from the TPU Web Application and delivers it to the CheckingRequest module. If the query is accepted returns the results to the TPU.
- **IntCheckingRequest** interface, which represents the way components can interact with the CheckingRequest component, exposes the following methods:
  - isQueryValid(GroupData queryResults): this method delivers the query results to the CheckingRequest component in order to control if the query respects the *Terms of use.*
  - askPU(String SSN): this method notifies the interested PU to know if he/she consents on his data usage.

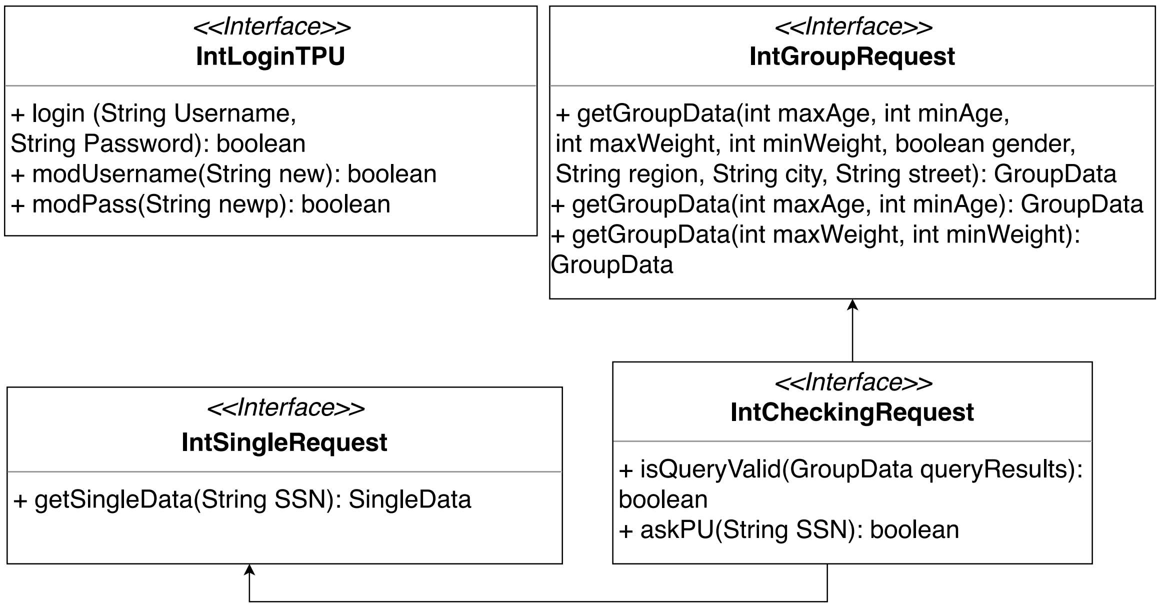

<<Interface>>
IntLoginTPU
+ login (String Username, String Password): boolean
+ modUsername(String new): boolean
+ modPass(String newp): boolean
<<Interface>>
IntGroupRequest
+ getGroupData(int maxAge, int minAge, int maxWeight, int minWeight, boolean gender, String region, String city, String street): GroupData
+ getGroupData(int maxAge, int minAge): GroupData
+ getGroupData(int maxWeight, int minWeight): GroupData
<<Interface>>
IntSingleRequest
+ getSingleData(String SSN): SingleData
<<Interface>>
IntCheckingRequest
+ isQueryValid(GroupData queryResults): boolean
+ askPU(String SSN): boolean

Figure 14: TPU component interfaces

2.6 Selected architectural styles and patterns

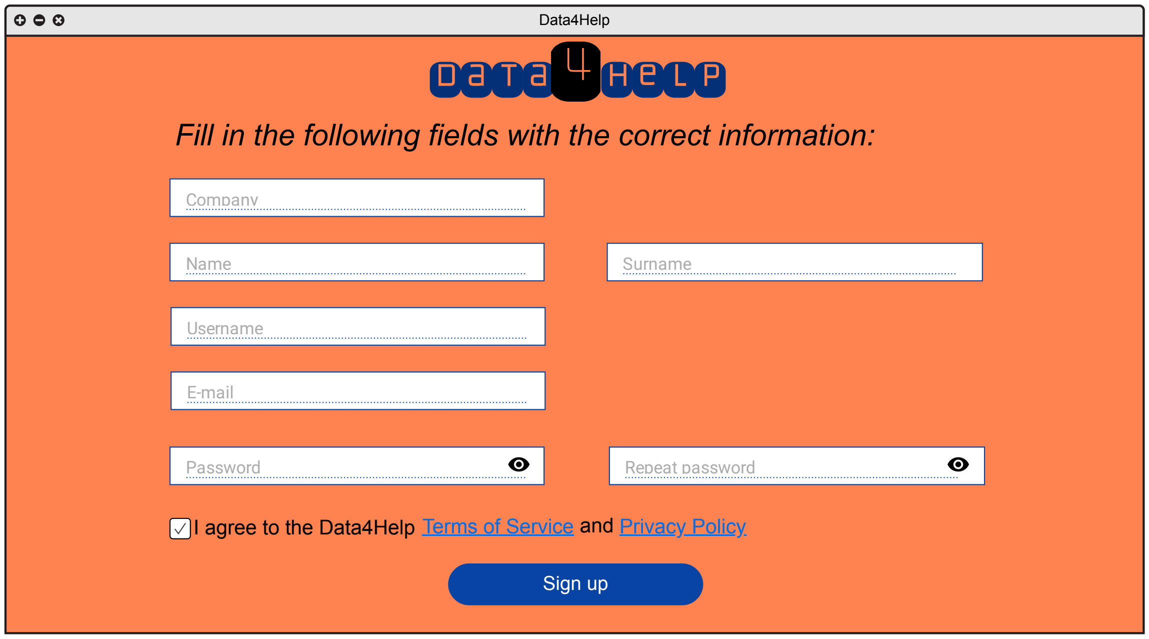
- Selected design pattern:
  - Adapter: converts the interface of a class to another interface that customers expect. This pattern allows classes to work together, otherwise it would not be possible due to incompatible interfaces.

- Architectural pattern for implementation:
  - Model View Controller (MVC): an architecture for building applications that separates the logic and data (model), the user interface (view) and processing (controller). The MVC design pattern decouples these major components allowing for efficient code reuse and parallel development.
- Design pattern for implementation:
  - Factory: it defines an interface for creating an object, but leaves the choice of its type to the subclasses, creation being deferred at run-time.
  - Observer: after the modification of a data, it notifies the objects or classes that depend on this object.

2.7 Other design decisions

AutomatedSOS is an application that requires perfect coordination with the rescue service. For this reason to each structure that accepts to integrate automatedsos in its service, a device suitable for receiving the emergency signal will be installed.

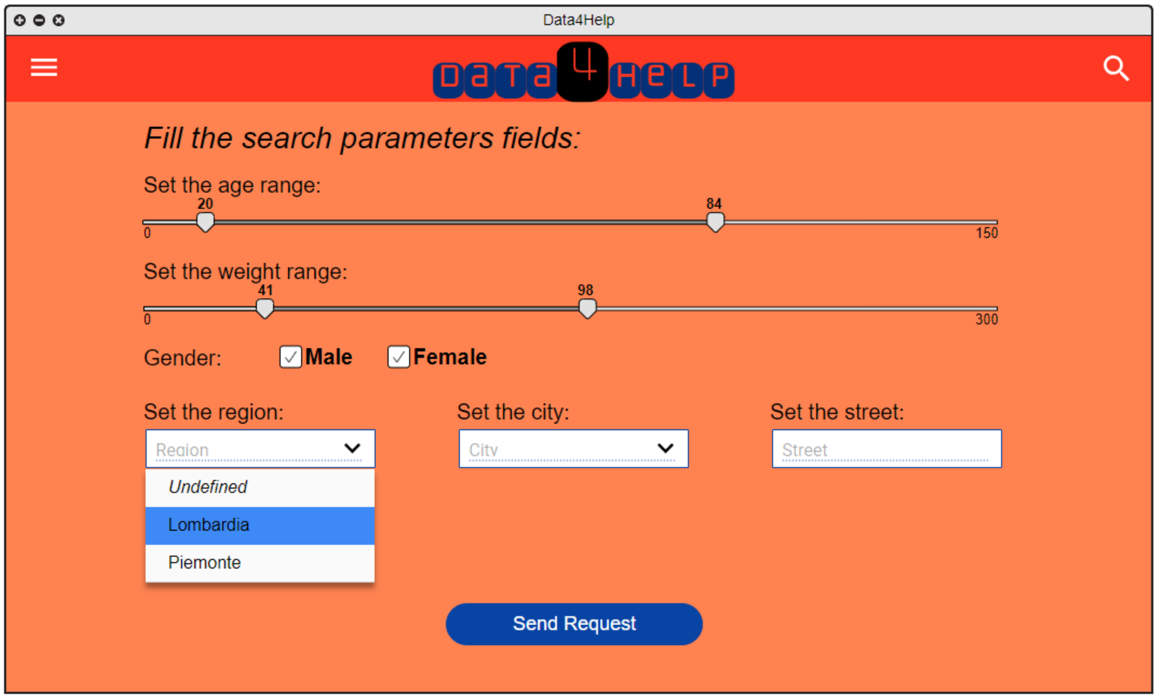
3 User interface design



Fill in the following fields with the correct information:

[✓] I agree to the Data4Help Terms of Service and Privacy Policy

Figure 15: Mock up - Data4Help registration page.



Fill the search parameters fields:

Set the age range: 0 – 150, 20 – 84

Set the weight range: 0 – 300, 41 – 98

Gender: [✓] Male [✓] Female

Set the region:

Set the city:

Set the street:

Figure 16: Mock up - Group query page.

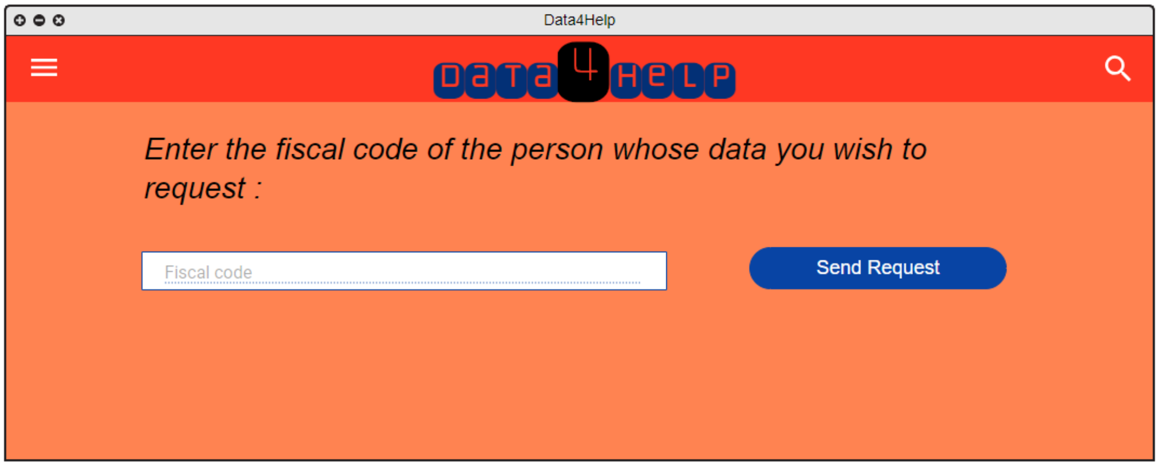

Data4Help
Data 4 Help
Enter the fiscal code of the person whose data you wish to request :
Fiscal code
Send Request

Figure 17: Mock up - Personal query page.

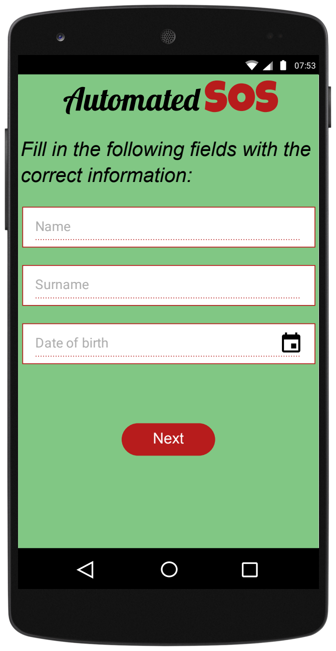


Figure 18: Mock up - AutomatedSOS first registration page.

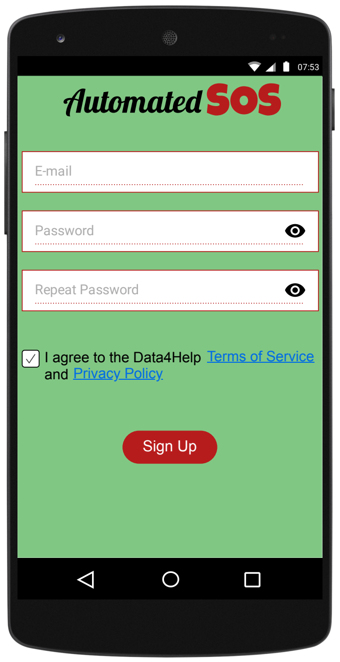

07:53
Automated SOS
E-mail
Password
Repeat Password
I agree to the Data4Help Terms of Service and Privacy Policy
Sign Up

Figure 19: Mock up - AutomatedSOS second registration page.

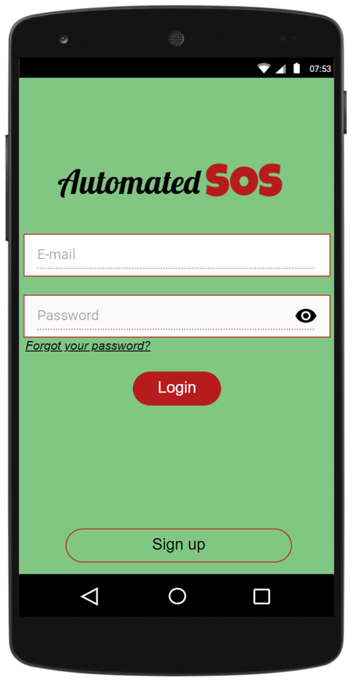


Figure 20: Mock up - AutomatedSOS login page.

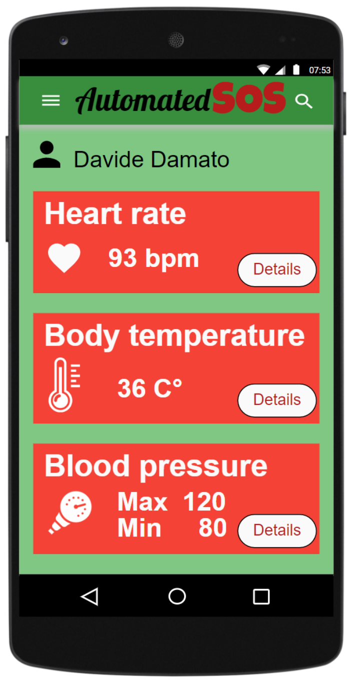


Figure 21: Mock up - AutomatedSOS health status on smartphone.

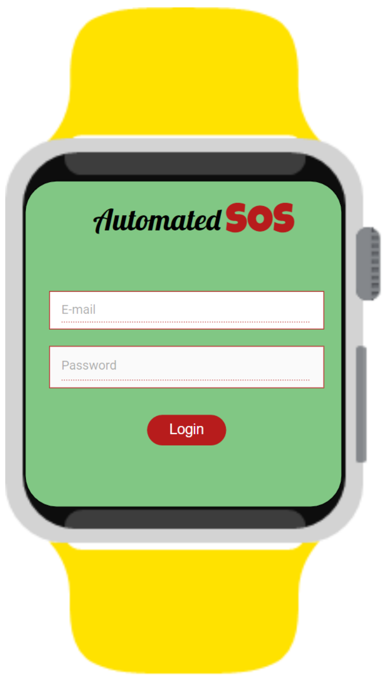


Figure 22: Mock up - Login for Smartwatch.

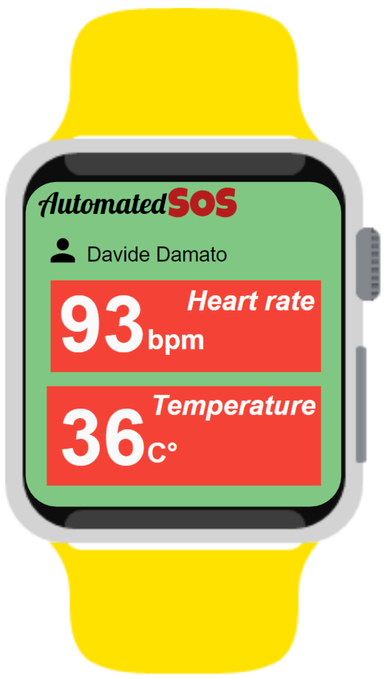


Figure 23: Mock up - Health status.

4 Requirements traceability

In this section the goals and requirements defined in the RASD document are mapped with the components defined in this DD document. This correlation of elements from the two documents allows to maintain a concrete and consistent project description between RASD document and DD document.

[**G1**]**:** Collect user data. (Requirements from [R1] to [R3])
PU Services:

- LoginPU
- UpdatingData

[**G2**]**:** Allow Third Parties to receive data collection. (Requirements from [R4] to [R5])
TPU web Services:

- LoginTPU
- GroupDataRequest
- SingleDataRequest
- CheckingRequest

[**G3**]**:** Allow a user to monitor his own parameters. (Requirements from [R6] to [R7])
PU Services:

- LoginPU
- HealthAnalysis

[**G4**]**:** Allow a user receive first aid in emergency situation. (Requirements from [R8] to [R9])
PU Services:

- LoginPU
- HealthAnalysis
- EmergencyCall

5 Implementation, integration and test plan

5.1 Implementation plan

In this section there will be explained the reasons and motivations behind the order of components implementation. This sequence of implementation represents one of the most important working decision since it will lay down the structure of the project development. The component are developed in an order such that to avoid any risk of failure propagation and to prevent work hours loss due to unwanted maintenance.

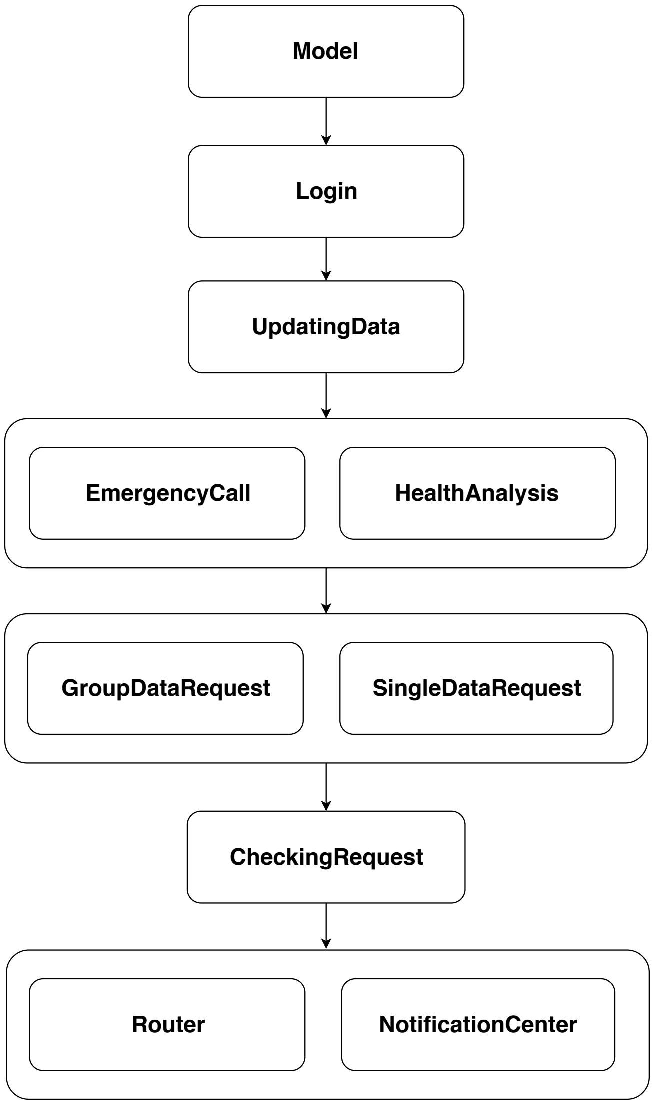


Figure 24: Implementation sequence

The sequence above represents only an high level implementation order, during real development it may slightly change and some components can be changed while developing others due to necessity of new functionalities; this sequence is important to focus on some component before others and to analyze which ones are the milestones of the system.

The first component to be implemented should be the **Model** of the MVC, this module is intended to compose the main structure of the system and to support every other component. Due to the key role of the Model it can not be implemented along the way while other components are being developed. Once the Model is finished every other component that interacts directly with it can be implemented, this procedure avoids that different members of the development team will develop parts of the Model by their own causing inconsistency and integration problems.

When the main data structure for the system has been implemented, the development can move on to the **registration procedure** in order to test if the system works. The registration procedure will be performed by the **Login** modules both in **PUServices and TPUServices**. When both these module are implemented the simplest and most basic data flow of the structure is defined and developers can move on to more complicated components.

UpdatingData represents one of the most used component in the PUServices structure and it has to respect strict performance requirements to meet the stakeholders requests. This component constitute the key step of the continuous **data update** from the PU's Applications and the DBMS. UpdatingData executes the role of sending updated health values to HealthAnalysis so this component requires high reliability and performance, it must be accurately developed and tested to respect the requirements stated in RASD document.

HealthAnalysis and EmergencyCall both work on the same task of monitoring PU's health status in order to guarantee a prompt response to emergencies. Specifically **HealthAnalysis** monitors every 5 seconds if PU's health parameters meet the thresholds and **EmergencyCall** is responsible of communicating users informations to first aid services. Since these components perform an elaborated functionality, they require a well defined system structure on which to base their methods.

When the primary system structure and the most important functionality are implemented, the developers can continue implementing the surrounding processes. Among these, the most important are **GroupDataRequest and SingleDataRequest**, these two component execute the main role to implement requirements about Data4Help application. **GroupDataRequest** performs every procedure about multiple individuals query requested by TPU, from elaborating the parameters of the query to communicating with the DBMS. **SingleDataRequest** performs every procedure about single individuals query request by TPU, it communicates with the interested PU and then unrolls the request on the DBMS. As shown on the graph above **HealthAnalysis and EmergencyCall** should be developed before **GroupDataRequest and SingleDataRequest**, since these last modules requires a complete functioning data structure to work on.

CheckingRequest is a component required by GroupDataRequest and SingleDataRequest to validate the query results. This modules does not perform any elaborated functionality but needs to be implemented separately to ensure maintainability and meet stakeholders requirements anytime during the deployment process.

The last components to be implemented are the **Router and NotificationCenter**, this modules play the role of the reception in the communication between Applications and internal services. **Router** has to redirect every external request to the designated destination, indeed, to understand how to route every package in the data flow, the destination components must already be defined. **NotificationCenter** has to send every message from internal components to external components and, in order to do this, the senders module have to be defined.

5.2 Integration and testing

5.2.1 Entry criteria

The implementation of the system components must be gradual and tested step by step. When the modules are integrated and connected, they are individually tested appropriately and must be able to execute their operations correctly.

In order to perform the tests effectively, below are the minimum implementation percentages for each module:

- Login - 50%
- UpdatingData - 70%
- HealthAnalysis - 70%
- EmergencyCall - 90% (this module contains only the emergency call operation, the main operation that must be implemented as completely as possible immediately to better integrate the other modules)
- GroupDataRequest - 80%
- SingleDataRequest - 70%
- CheckingRequest - 90% (the check operation must be precise and complete even at the beginning of the implementation)
- Router - 50%
- Notification Center - 50%

5.2.2 Elements to be integrated

As shown in the introduction, Data4Help system architecture can be divided in three layers. Thanks to this partition the components integration can be divided in three groups in order to develop the structure from the ground and then rising to the integration with high level components.
To clarify the integration procedure it can be divided in three macro groups as shown on the list below.

- **Data Layer:** integration with DBMS and components interfacing with it.
- **Application Layer:** integration of components inside PUServices and TPUwebServices.
- **Presentation Layer:** integration of external components.

Data Layer: on this level the components have to integrate with the DBMS and interface every data exchange procedure. This must be the first level to integrate because the system will be built with a bottom-up strategy, the interaction with the database represents one of the simplest and lowest level interaction inside the system.

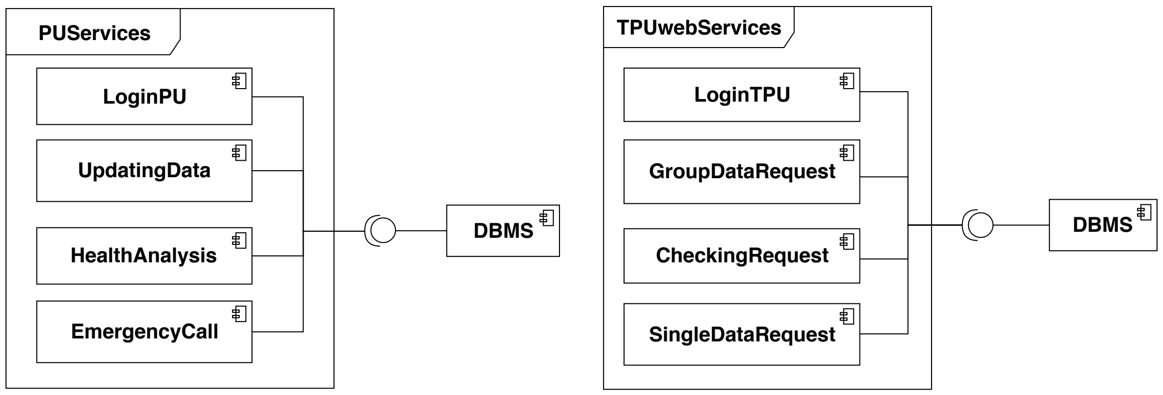


Figure 25: Data layer elements integration

Application Layer: the goal on this is to implement every component inside PUServices and TPUwebServices. The component on this level represents the core of the system and require a certain effort to be integrated properly, once they have been interfaced between each other the system should have most of its functionalities developed.

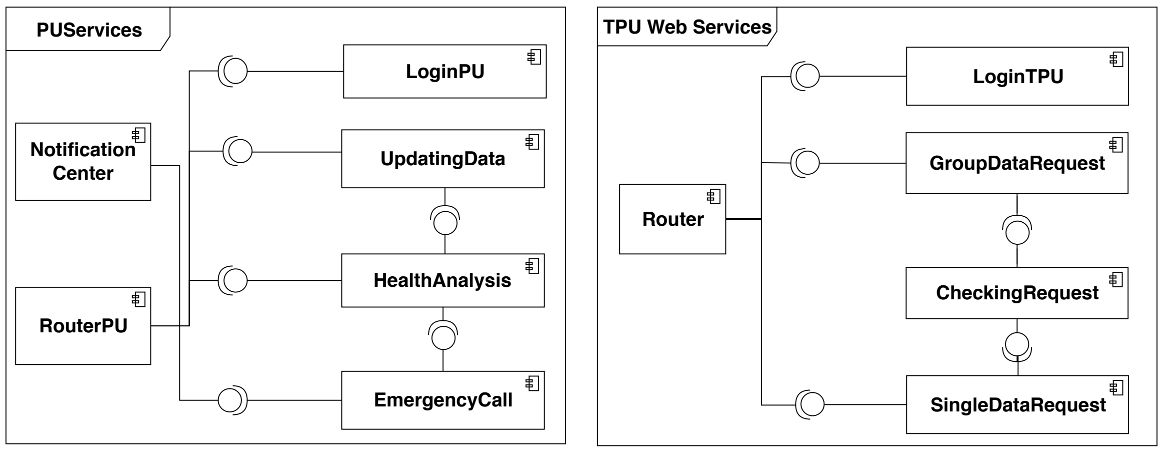


Figure 26: Application layer elements integration

Presentation Layer: this is the last level to be integrated since it requires the integration with some high level components. When this level is integrated the system integration can be considered finished. Anyway, some new external components may be added during development and this structure shouldn't be considered final.

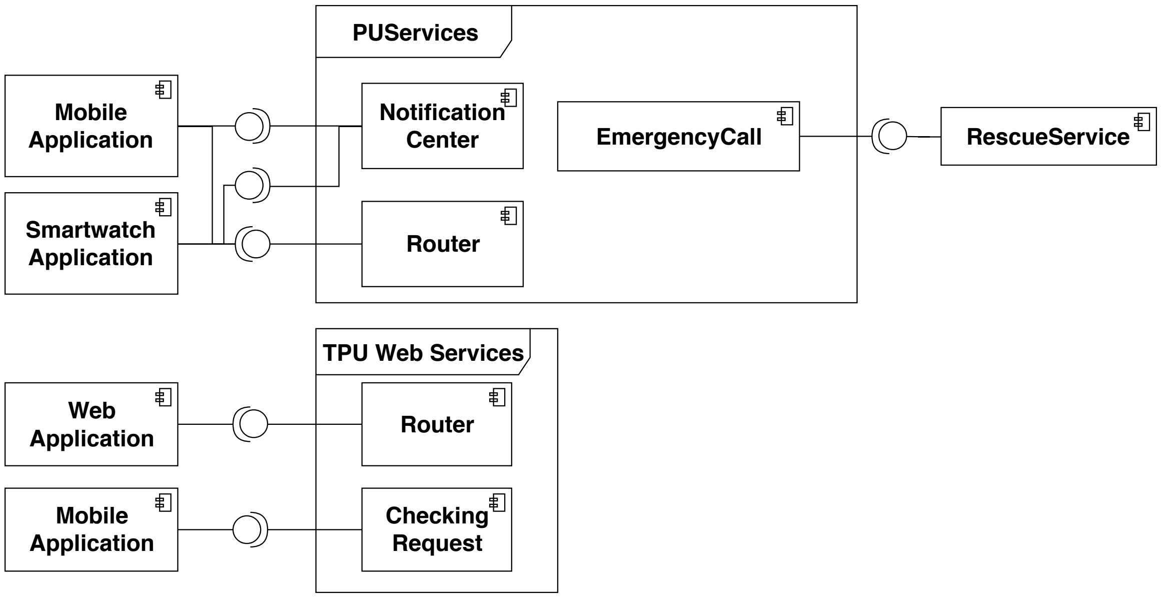


Figure 27: Presentation layer components integration

5.2.3 Integration testing strategy

As described in the previous sections, the implementation of the system components follows a tree structure. Bottom-up strategy allows to perform better tests, very precise for each function. The goal is analyze and check the components individually from the beginning of the implementation, in order to have correct and functioning modules when integrating with the rest of the system. The same procedure must be carried out to EmergencyCall and CheckingRequest modules, but with more attention, being sensitive components that could seriuosly compromise the operation of the application in the presence of bugs.

5.2.4 Sequence of Component/Function Integration

In this section there will be explained how every component is integrated whit the other ones in its same layer and in which order they are integrated. As already explained above, the system will be developed in a bottom-up strategy so the modules will be developed respecting the layer order and in definition order from the lowest to the highest one.

Data Layer
In the graphs below it is shown which are the components to interface with the DBMS. Both graphs start with the simpliest module and end with most difficult one. The order of integration has been chosen to respect the bottom-up development strategy and to ease-up the process to the development team. By following this structure, the system architecture should be more solid and maintainable. The simpliest modules to integrate is: Login.

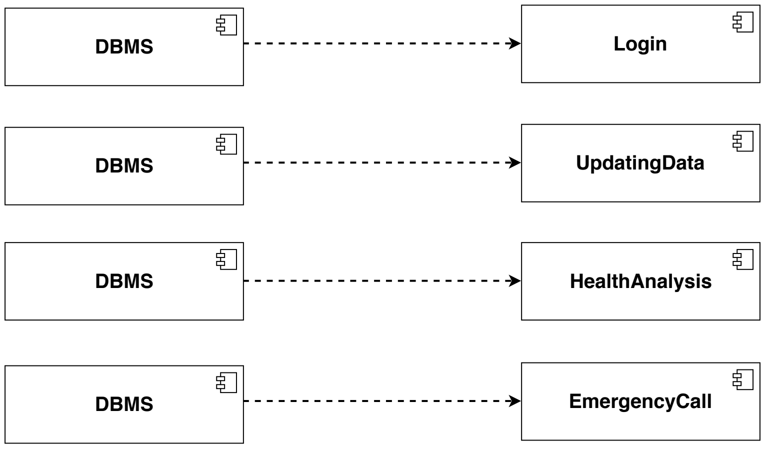


Figure 28: PUServices components integration with DBMS

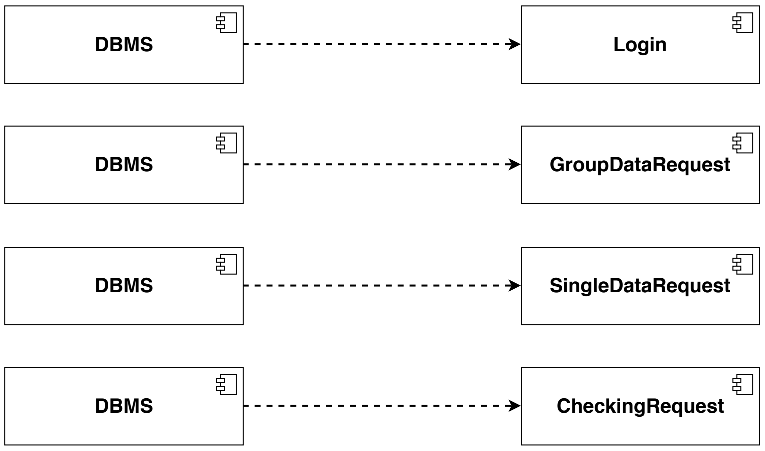


Figure 29: TPUwebServices components integration with DBMS

Application layer

The integration order of this layer is similar to the previous one but with the modules of the Application layer. The order of integration reflects the bottom-up integration strategy starting from the Router and then moving on with more difficult module pairing. At the end of the *PU Services* module the developers must implement the integration between **UpdatingData, HealthAnalysis and EmergencyCall** which, as stated before, constitute one of the most difficult and delicate structure in the system. Similarly, at the end of the *TPU Web Services* module there's left the integration between **SingleDataRequest, CheckingRequest and GroupDataRequest** which are the modules that represent the core of the TPU Web Services functionalities.

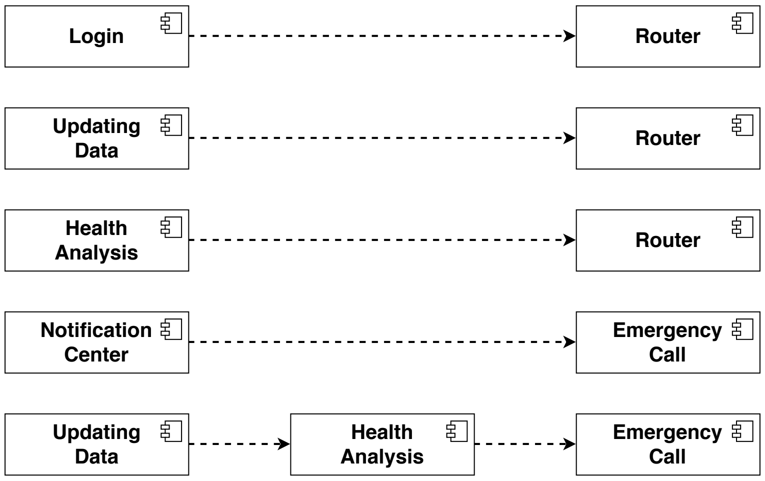


Figure 30: Application level components integration for PU Services

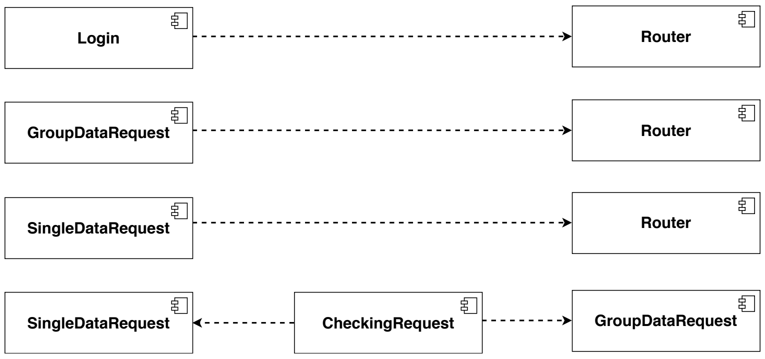


Figure 31: Application level components integration for TPU Web Services

Presentation Layer

When the core of the Data4Help system is completely developed and integrated, the developers can continue to develop the Presentation layer. This layer has been purposely left for last because it expects integration with high level components such as *Mobile Application, Web Application and EmergencyCall*. Since this are complex components they may require some time to integrate correctly with the system which must be almost ready to deploy to integrate with them. The integration order is similar to the previous layer and it is aimed to ease the developers work.

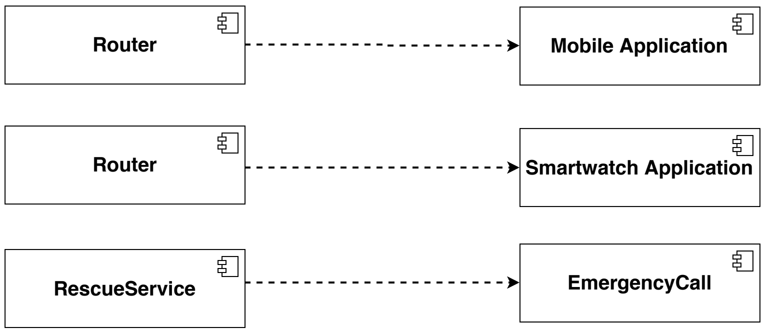


Figure 32: Presentation level components integration for PU Services

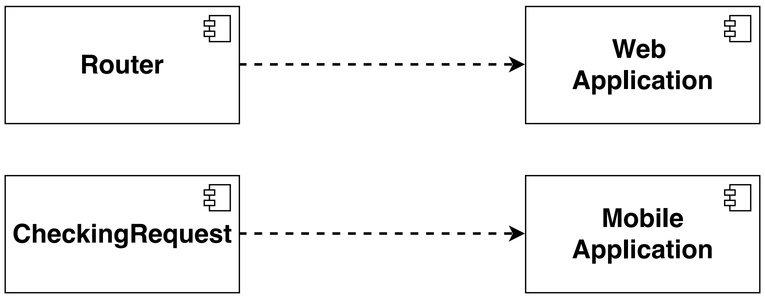


Figure 33: Presentation level components integration for TPU Web Services

6 Effort spent

6.1 Davide Damato

Date	Task	Hours
27/11/2018	High Level Architecture - Diagram	2
28/11/2018	High Level Diagram	1
30/11/2018	Component View Diagram	2
30/11/2018	Deployment View Diagram	3
03/12/2018	Component Interfaces Diagram and fix	3
04/12/2018	Runtime View Diagram	3.5
05/12/2018	Runtime View Diagram fix - Design Pattern - Other Design decisions	3.5
06/12/2018	Component interfaces description	2
06/12/2018	Requirements	1,5
07/12/2018	Implementation, integration and test plan	6
10/12/2018	Document Review	3
	Total	30.5

Table 1: Davide Damato work hours detail

6.2 Luciano Franchin

Date	Task	Hours
27/11/2018	High Level Architecture	2
30/11/2018	Scope - Overview	2
03/12/2018	Overview - High level architecture	1.5
03/12/2018	Text writing: ”High Level component view” and ”Component view”	3
04/12/2018	Runtime View Diagram	3.5
05/12/2018	Scope - Runtime View Descriptions	3.5
06/12/2018	Implementation plan	1
07/12/2018	Implementation, integration and test plan	6
09/12/2018	General document review	1
09/12/2018	High component view update	2
10/12/2018	Document review	5
	Total	30.5

Table 2: Luciano Franchin work hours detail

7 References

7.1 Used tools

- `overleaf.com`, Overleaf © 2018.
- `draw.io`, a trading name of JGraph Ltd.
- `websequencediagrams.com`, by Hanov Solutions Inc., of Waterloo, Ontario, Canada.