[Introduction 1](#_Toc531944438)

[Architectural design 1](#_Toc531944439)

[Overview 1](#_Toc531944440)

[Component view 3](#_Toc531944441)

[Deployment view 8](#_Toc531944442)

[Runtime view 9](#_Toc531944443)

[Component interfaces 15](#_Toc531944444)

[Selected architectural styles and patterns 15](#_Toc531944445)

[User interface design 15](#_Toc531944446)

[Requirements traceability 16](#_Toc531944447)

[Implementation and test plan 16](#_Toc531944448)

[Effort spent 16](#_Toc531944449)

[References 16](#_Toc531944450)

# Introduction

# Architectural design

## Overview

The physical components that are involved in the Data4Help service are:

* Individual’s cell phone
* Individual’s Smartwatches
* Third party’s Computer
* TrackMe system (which contains databases and server)

For the service Automated SOS, the following service must be added :

* Ambulance service

Communications between components:

* Cell phone ↔ Server (The cell phone sends requests to the server and the server answers)
* Computer ↔ Server (The computer sends requests to the server and the server answers)
* Server ↔ Database (The server makes queries and the database answers)
* Smartwatch ↔ Cell phone (The cell phone sends queries to the smartwatch and the smartwatch sends data to the cellphone via Bluetooth)
* Cell phone → Ambulance service (The cell phone sends SMS to an ambulance service)

The server must send data to the different users (individuals or third parties) only when it’s necessary. Because the users use the app through devices that can be offline, the server must only send data when asked by the users. Otherwise the user could never receive the data if his device is offline. For those reasons, an event-based system would not be adapted

We need to design a system which involves many stakeholders such as individuals, third parties, ambulance services, track systems. Moreover, in all the interactions the system is providing a service to the users so we decided to use a client-server architectural approach.

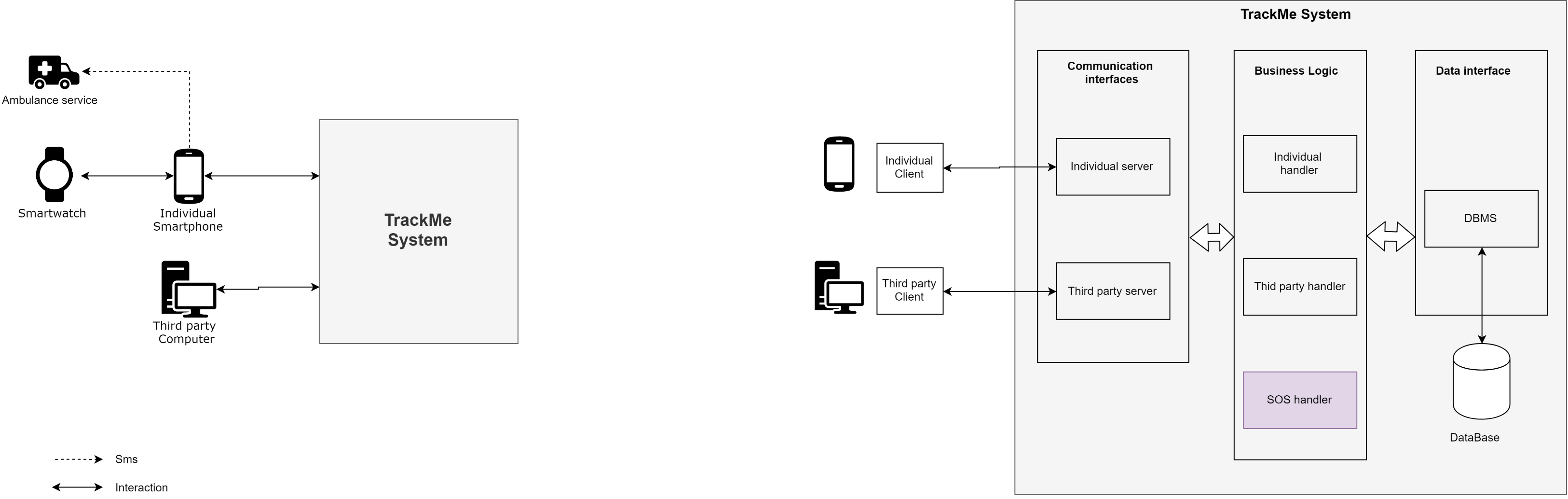


Figure 1 Overview 1

Going deeper in the analysis of the components of the TrackMe system, we are able to identify three different layers :

* **Communication Interfaces :**

This lawyer contains the interfaces components that allow the system to communicate with external agents (individual’s smartphones and third parties’ computers). As TrackMe system interacts with different external agents so it needs to have different communication Interfaces :

* A software module is needed in order to provide functionalities of the system to the individuals on their smartphone. This software module should allow individuals to send requests to the server using an API.
* A software module is needed to provide the functionalities of the system to the third parties. This module includes a website back -end and an API. The website back-end allows third parties to communicate with the server using a website ; the API allows the parties to download the data requested.
* **Business Logic :**

This lawyer focuses on the application logic of the TrackMe system. The business logic manages the individuals data, the third parties requests and their subscriptions ; for each of these functions, several software modules are necessary. Those modules will use the communication interfaces between the end user interface and the database. The business logic module receives from the communication interface orders to do specific actions; then the business logic ask to the data base interface the required data to execute these actions, and finally, returns the result to the communication interface.

* **Data Interface**

This lawyer contains all the modules that allow to store the data produced or retrieved from external resources. These modules allow interaction between the Business Logic modules and the System Databases.

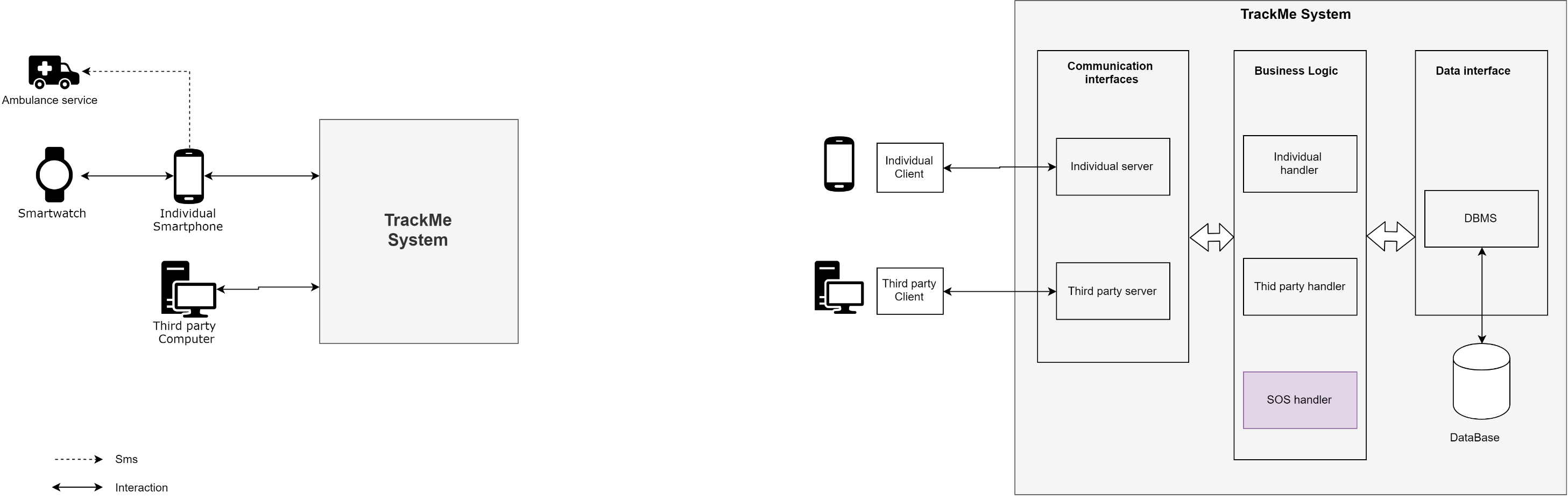


Figure 2 : Overview 2

## Component view

The following diagram, give a high-level representation of the components of the TrackMe system.

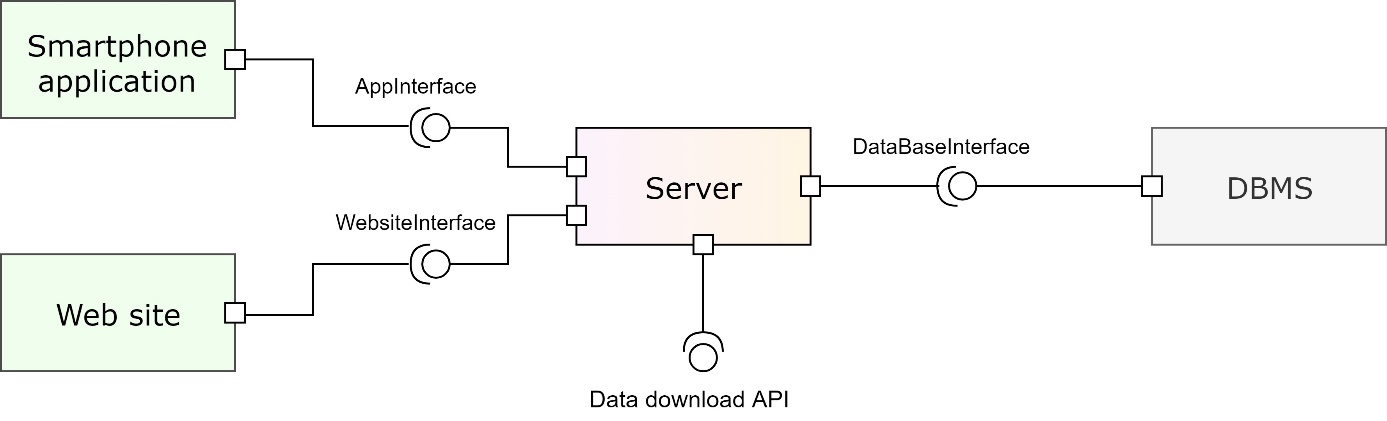


Figure 3: High level representation TrackMe system

As we can see, the system is composed of four main parts : A smartphone application, a Web site, a server and a DBMS. The smartphone application is intended for individuals and the Web site is intended for third parties. The server contains the business logic and make the connection between the others components and the DBMS.

The next component diagram contains a more low-level description of the server component :

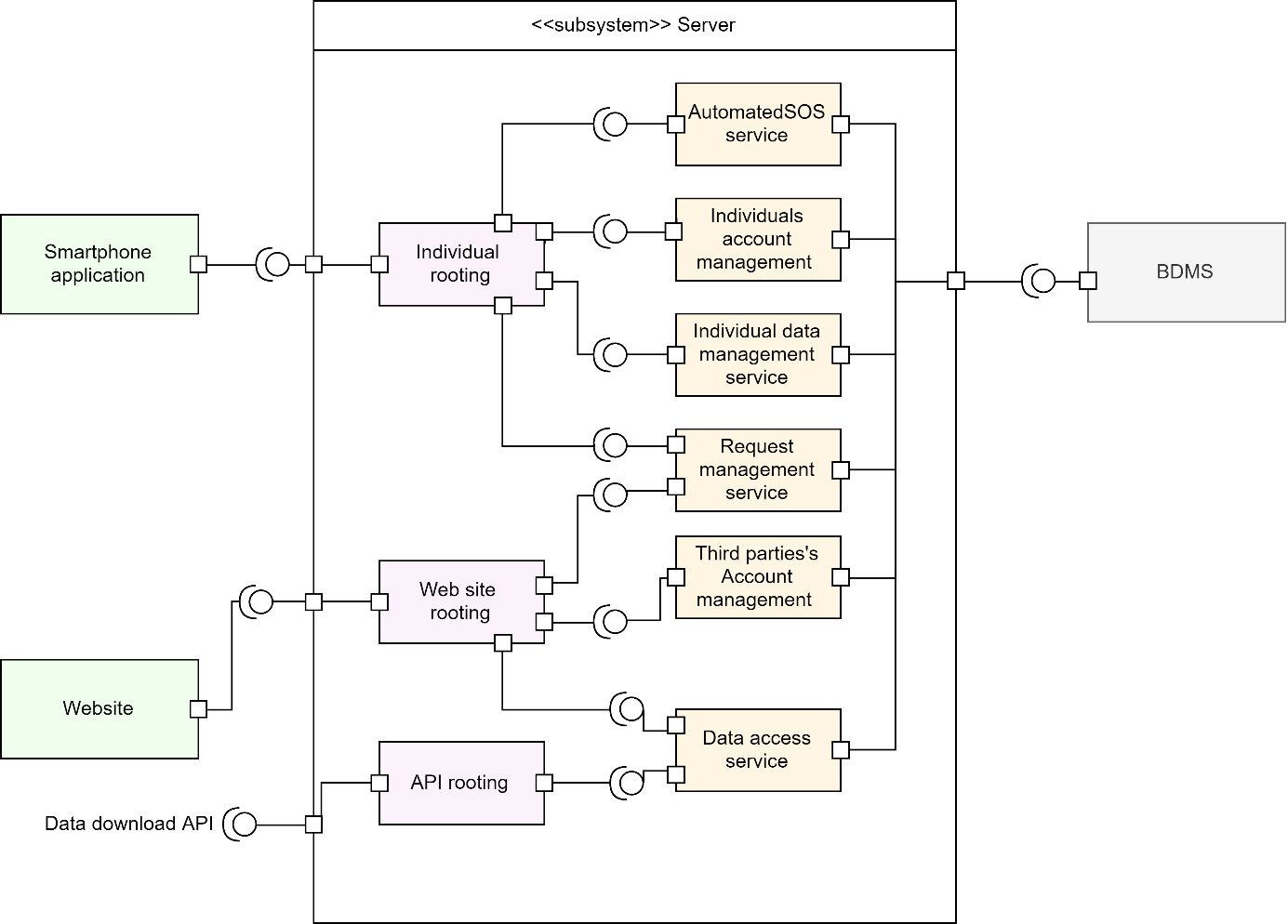


Figure 4: Low level representation trackMe system

As we can see, the server is composed of three main parts : Individual rooting, Web site rooting and API rooting. Those components determine whether a request that is received is valid or not and, if it is, to dispatch it to the relevant service component.

* **Individual rooting.**

This component manages all the functions the smartphone application has to provide to the individuals.

Individual rooting redirects the requests from the application to the corresponding component:

* *AutomatedSOS service* : manages all the functionalities of the AutomatedSOS service
* *Individuals account management* : manages the functionalities that allow individuals to register, login etc..
* *Individuals data management service :* manage the logic of the health and position data of individuals
* *Request management service :* manage the logic of the requests (for individuals : accepting or refusing request)
* **Web site rooting and API rooting.**

As explained in the RASD, the third parties can access all the functions of the Data4Help system through the Website and have also the possibility to download the data from accepted requests through an API. The API is aimed at accelerating the data acquisition process for third parties who need to acquire data regularly.

Web site rooting redirect the requests from the website to the corresponding component:

* *Third parties account management:* manages the functionalities that allow third parties to register, login etc..
* *Data access service:* manages the logic of formatting data from the Database to the third party.
* *Request management service:* manage the logic of the requests (for third parties : submitting individual or anonymized requests, subscribing to accepted requests)

The two following diagrams describe more precisely the business logic components, by showing their interactions with the data model. It is assumed that each component is able to invoke operations on the database

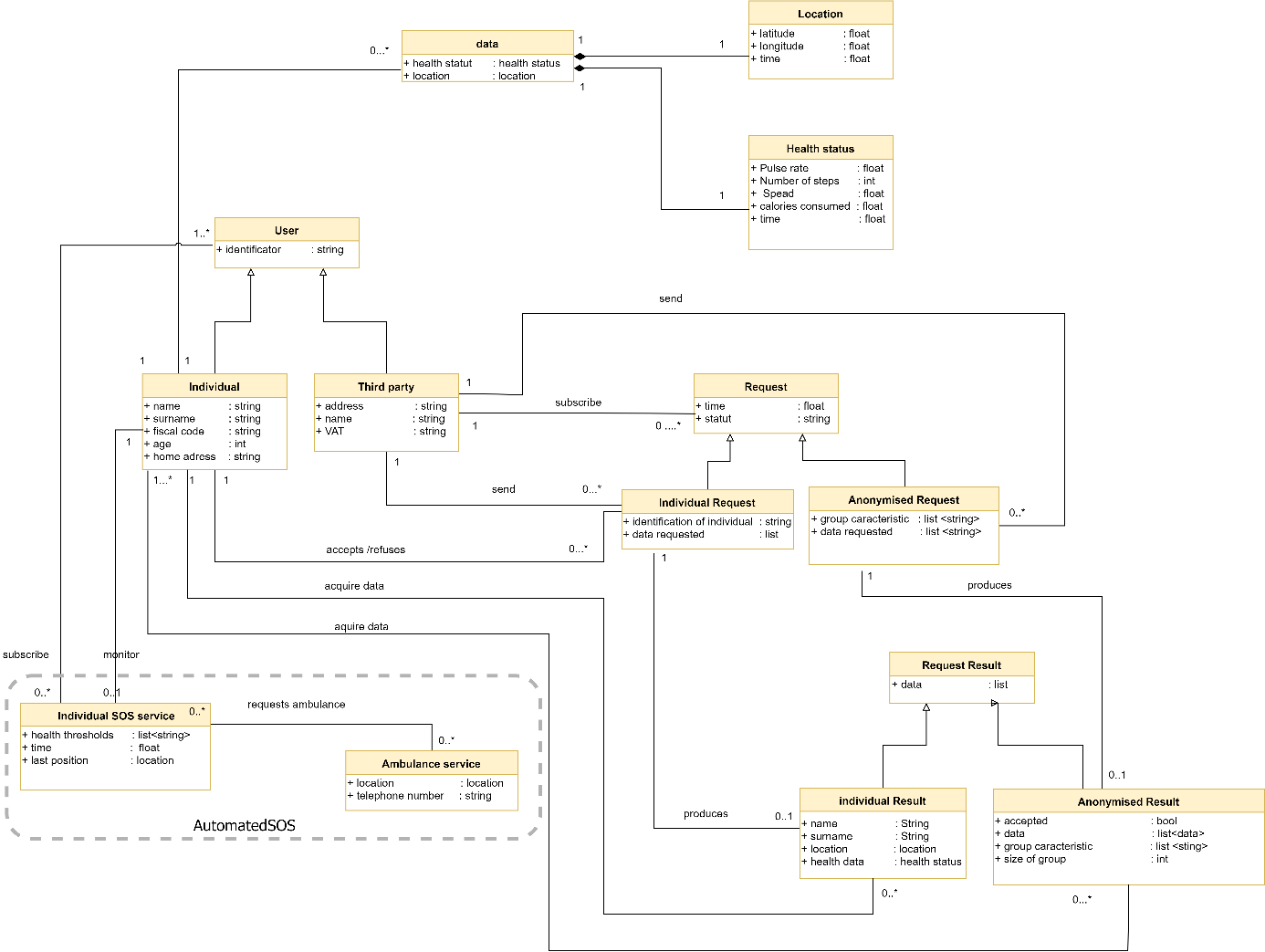


Figure 5: Class diagram of AutomatedSOS system

Some business components contain functions that requires the need to save data in the database. Those functions return a Boolean value : True if the data has been successfully saved, False otherwise. When a function is of this type, the return value is “bool\*”. The symbol \* make the distinction between functions of this type and other Boolean functions.

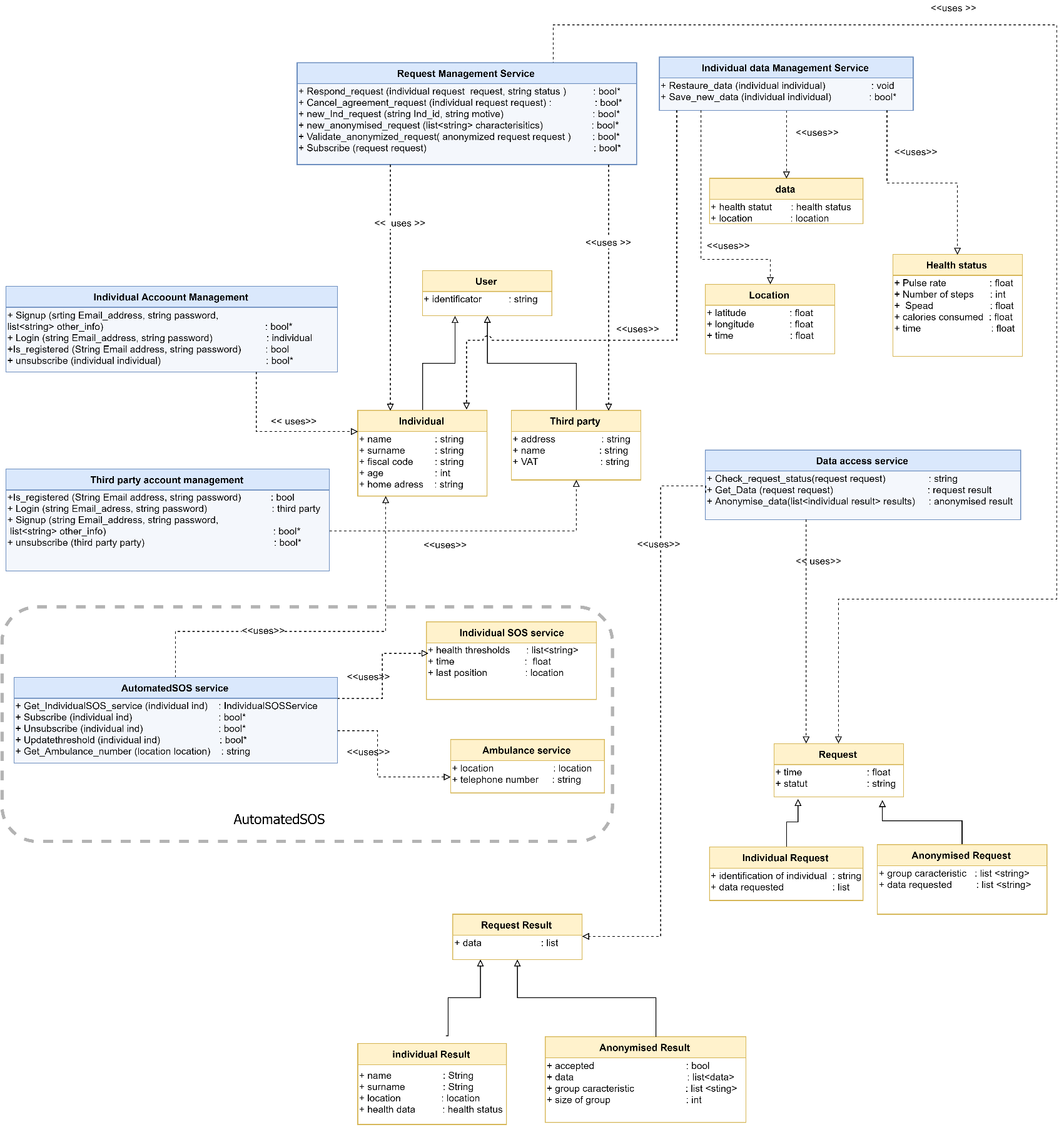


Figure 6: Description of logic components

The following diagram (figure XX) describes in further details the components of the individual’s smartphone. A smartphone contains a Database which stores the last health data of individuals. When an internet connection is available, the collected data is send to the TrackMe server.

Moreover, the thresholds and the phone number of the SOS services are also stored in the smartphone database and updated when an internet connection is available. This allows the app AutomatedSOS to send an SOS (using SMS) even without having an internet connection.

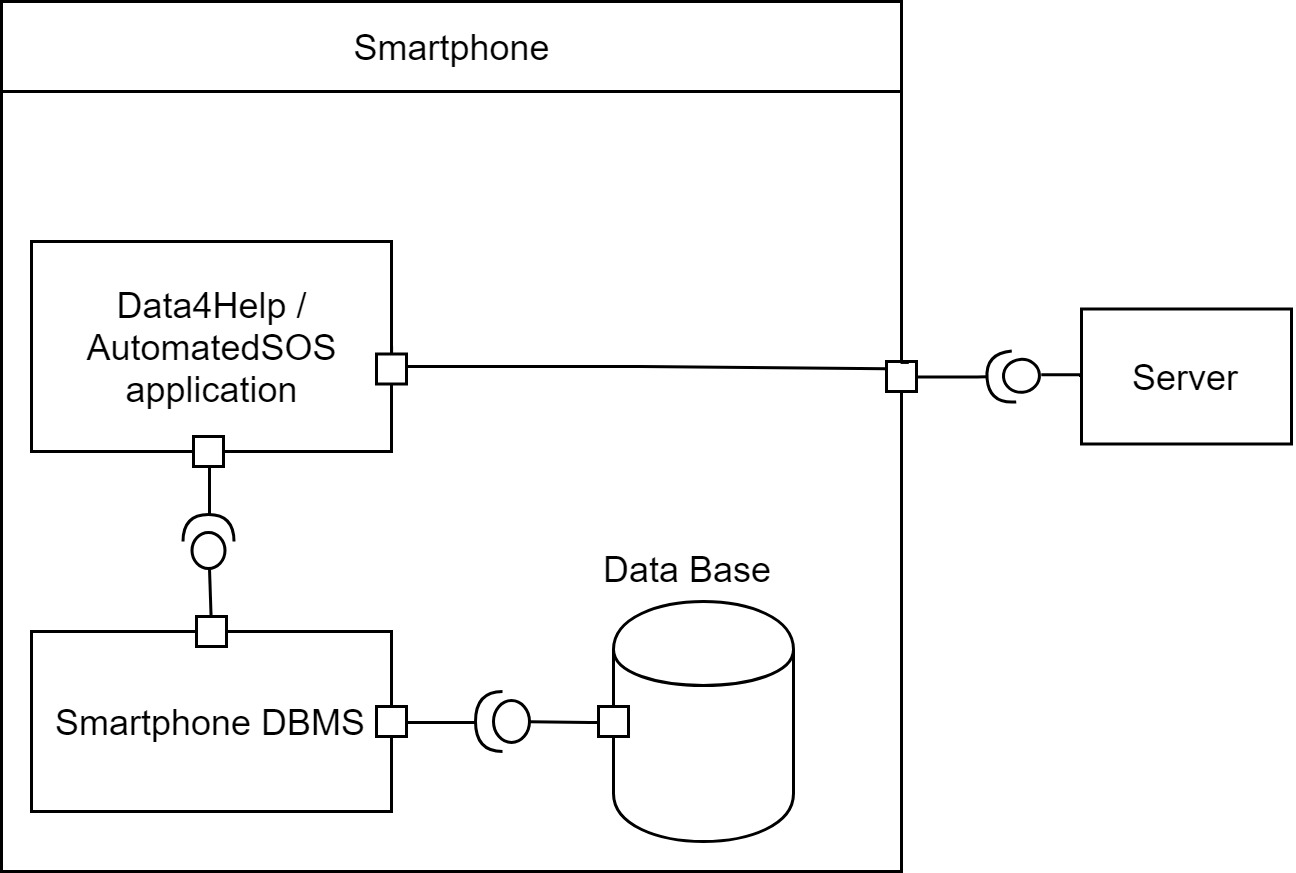


Figure 7: Component view system smartphone

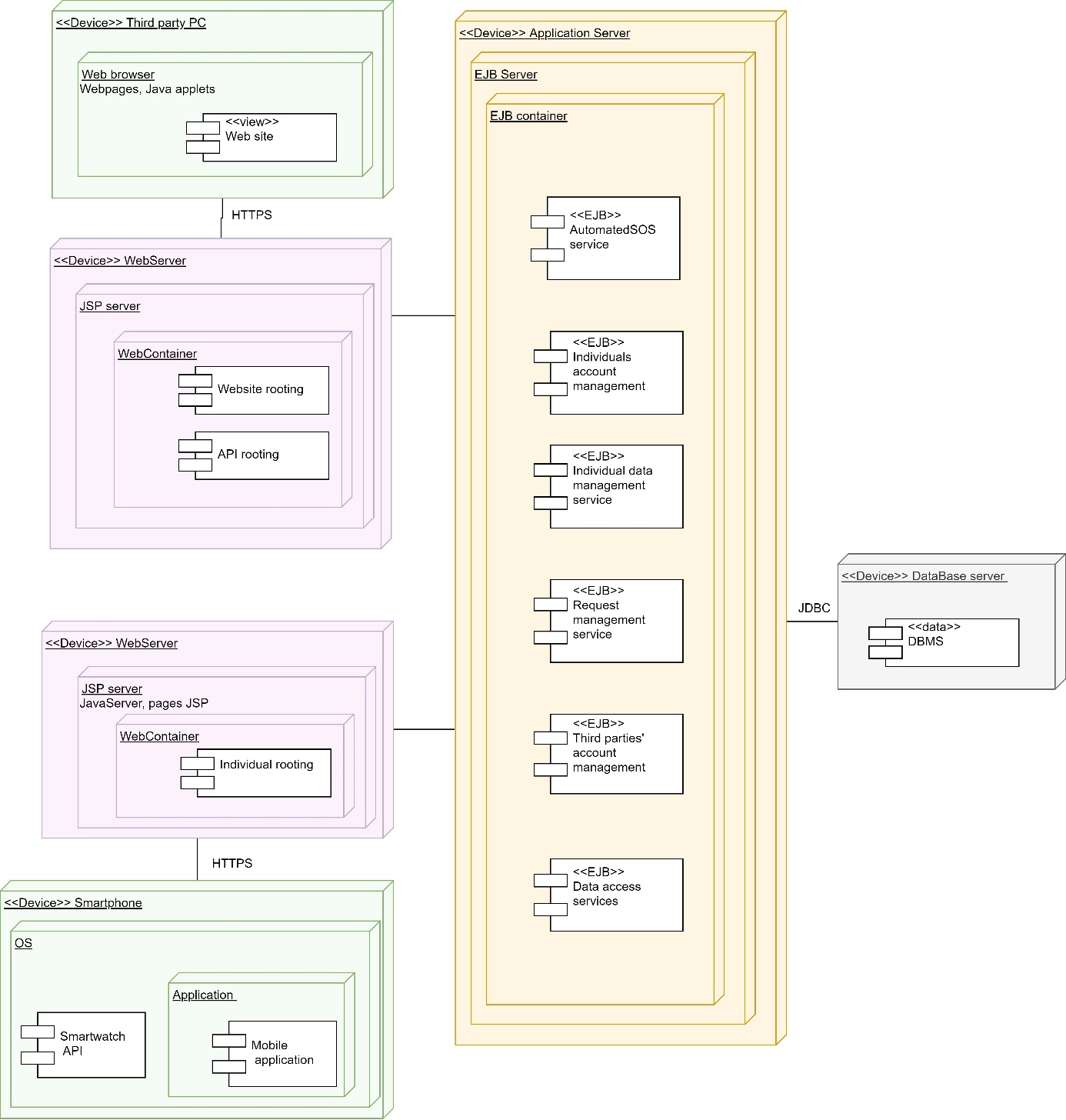
## Deployment view

The following deployment diagram represents the system's use of the physical infrastructure and how the system components are distributed and how they relate to each other.

The application Data4Help and AutomatedSOS require the following devices: A smartphone (for individuals) and a computer (for third parties). Those devices send information to a Webserver. There are two distinct Webservers, one that is in charge of the requests from the smartphone and one that is in charge of the requests from the computer.

The Webserver component communicate with the application server. The application server contains the business logic and communicates with the Webservers and the database server.

The database server stores all the data that is necessary for the Data4Help and AutomatedSOS services. Concerning the data of individuals, the database stores the account information, the history of their health and location data and their thresholds (if they subscribed to AutomatedSOS). Concerning third parties, the database stores their account information, their request and their subscriptions. The database also stores information on ambulance services (phone number and location).



## Runtime view

In this section we represent some runtime views of the interactions between the components of the Data4Help and AutomatedSOS systems.

### A third party makes an anonymized request

Third parties can make individual or anonymized requests. In the case where a third party submit an anonymized request, the validation (or non-validation) of the request can be done immediately as the choice does not involve a response for the individuals concerns.

The third party submits the anonymized request though the website interface. Then, the website rooting service transfers the request to the Request management service that carries out the validation.

The validation function implies an SQL request to the DBMS in order to obtain the number of individuals whose data satisfy the request. Then the status of the requests (validate or refused) is updated in the database and the response is communicated to the third party.

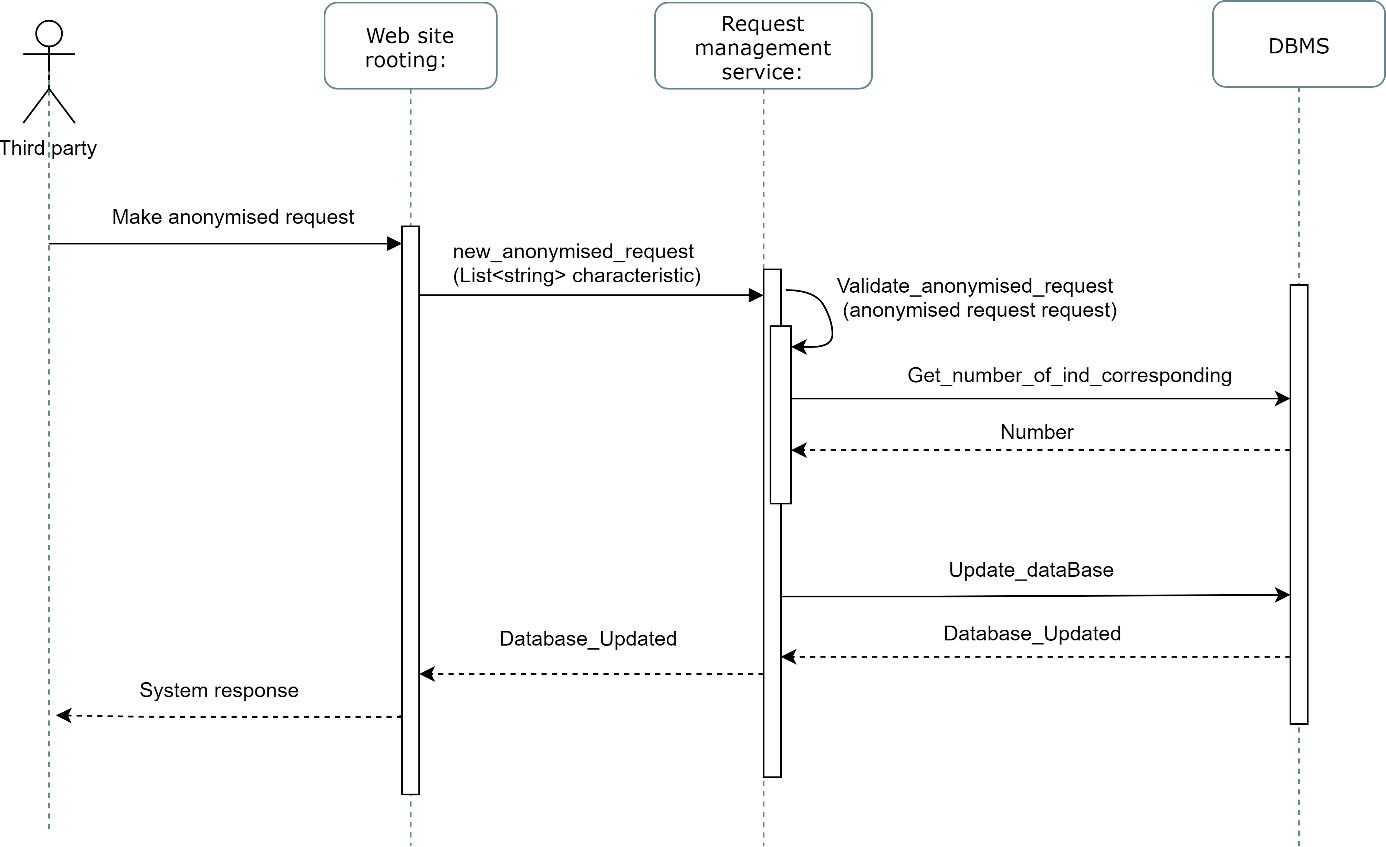


Figure 8: A third party makes an anonymized request

### A third party makes an individual request

Third parties can make individual or anonymized requests. In the case of an individual request the validation of the request is based on the response of the individual concerned.

The third party submits the anonymized request though the website interface. Then, the website rooting service transfers the request to the Request management service that saves it in the database with the status “waiting for validation”. Then the third party receives a confirmation message that his request has been saved.

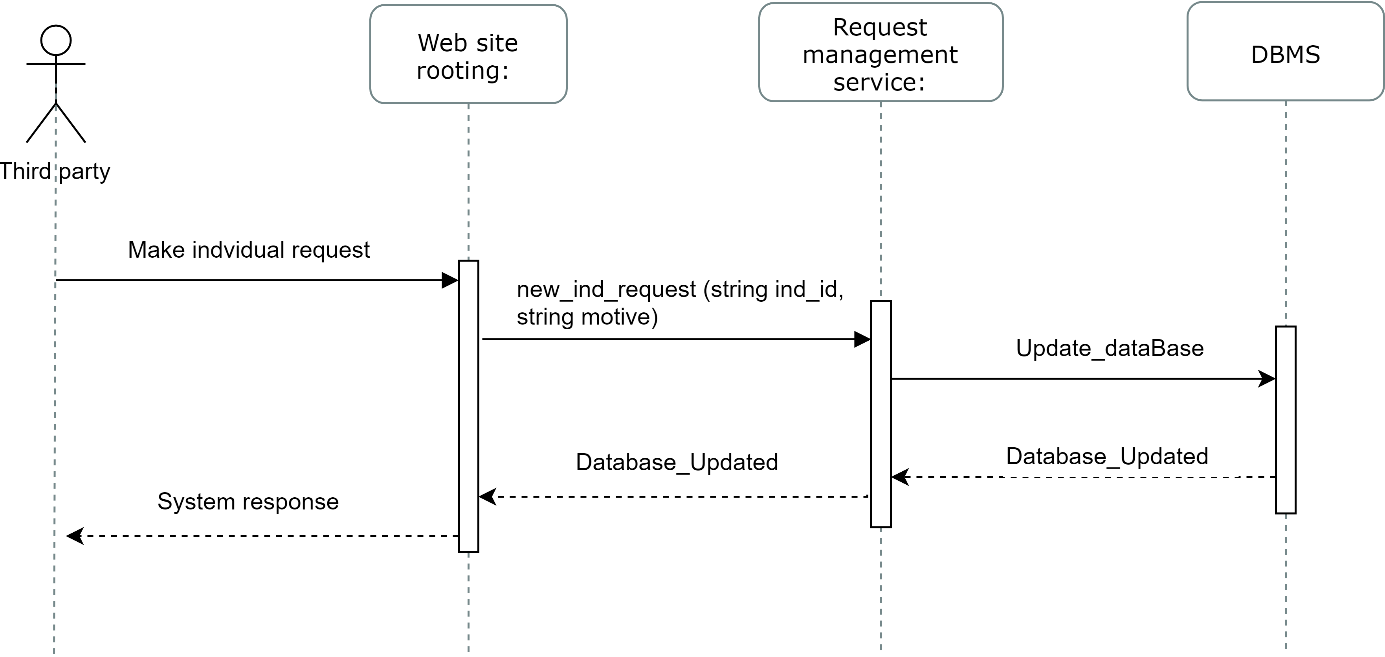


Figure 9: A third party makes an individual request

### An individual responds to a request

Individuals may be the subject of requests from third parties. Whenever an individual logs in on the smartphone application, the application rooting service make a request to the DBMS asking for the unanswered requests. Then the application displays on the screen of the user’s smartphone all the requests that are waiting for a response.

For each unanswered request, the individual can make the choice between accepting or refusing it by pressing a button. The individual’s response is then transferred to the application rooting to the request management service that transfer the modification of the request’s status to the database. Then, a confirmation message is printed to the user’s smartphone screen if the request’s new status is successfully saved in the database

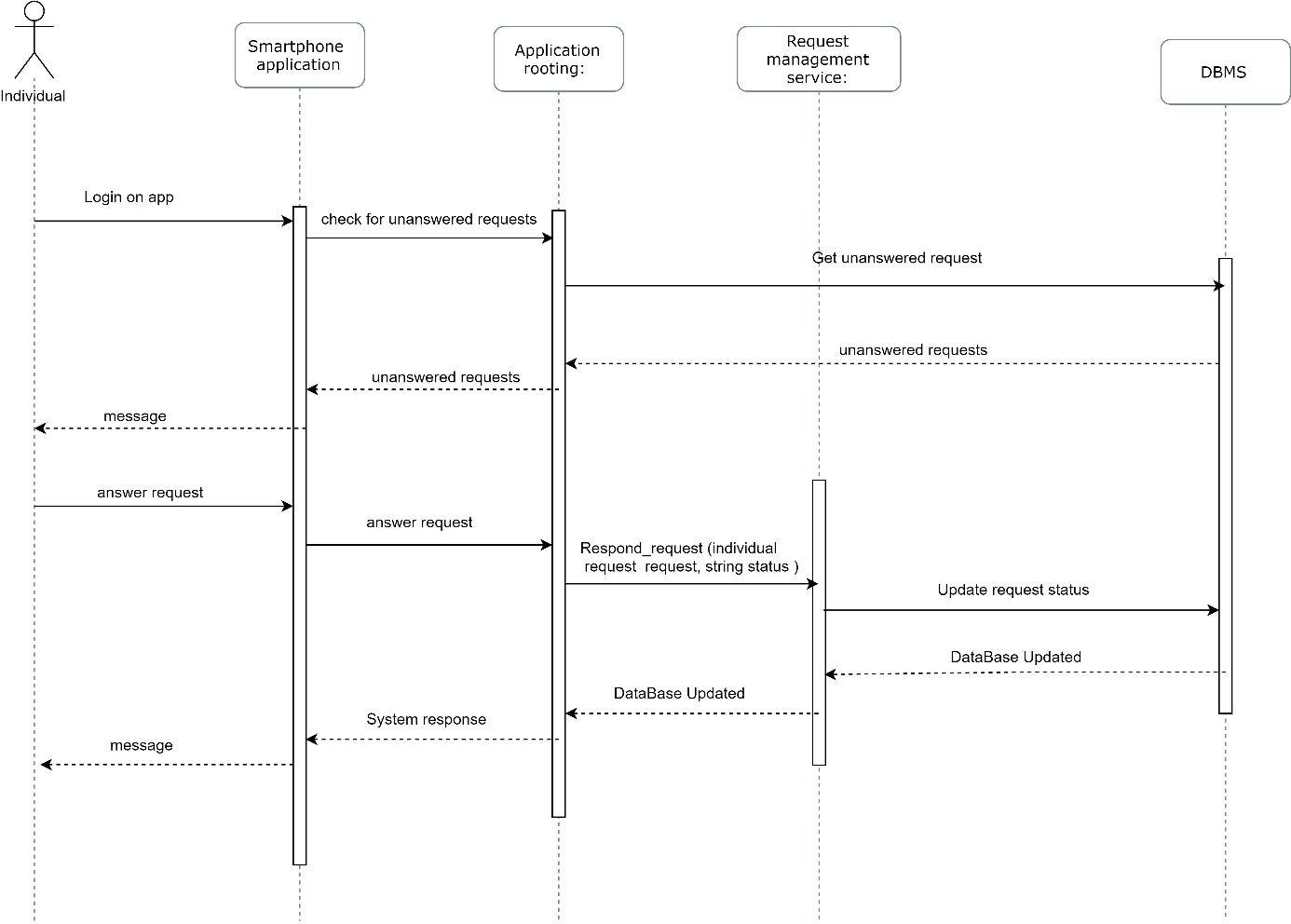


Figure 10: An individual responds to a request

### A third party subscribes to a request

Third parties have the possibility to subscribe to their accepted requests. By subscribing to a request, a third party has the possibility to download the newly available data corresponding to the request without going trough the validation process. When a third party makes request and when that request is accepted, the third party has access to the data until the date on which the request was made. When a third party subscribe to a request, he has access to the up to date data as long as his request stay valid.

When a third party logs in the Data4Help website, if he has an accepted request, he has the possibility to subscribes to it (if a request is not accepted, the “subscribe button” does not appear). When the third party clicks on subscribe, the website rooting transfers the request to the Request management service which transfers it to the data access service that asks to the DBMS the status of the requests.

If the status is accepted, the status of the request is updated in the database and the third party receives a confirmation message.

This double check increases the security and avoid third party being able to access data from unwilling individuals.

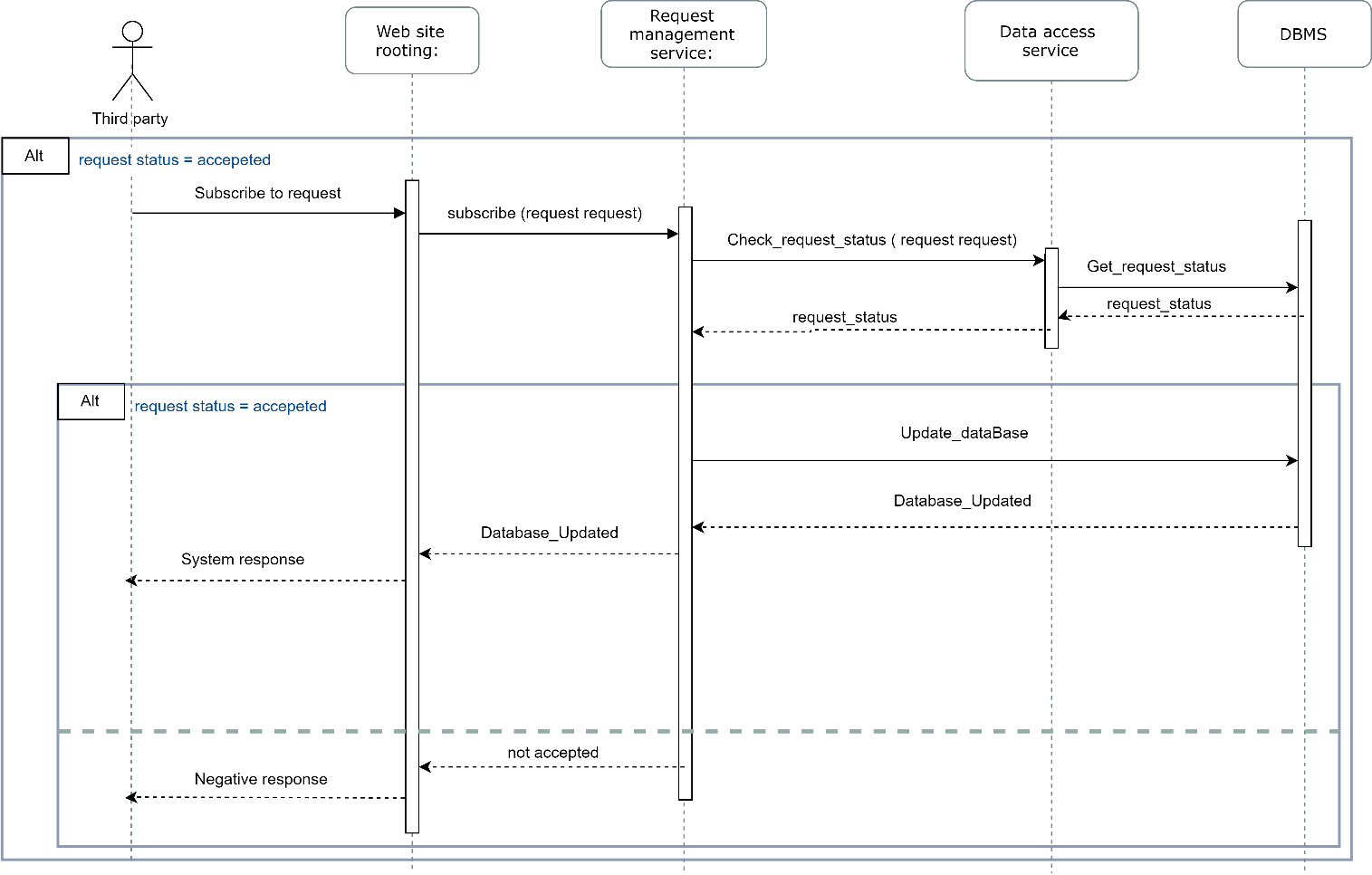


Figure 11: A third party subscribes to a request

### A third party download data from an anonymized request

When a third party logs in the Data4Help website, a request is made to the Website rooting service in order to know what needs to be displayed on the third party’s computer screen. If the third has some accepted requests, the download button will appear on his screen.

When the third party clicks on the download button, the Website rooting service requests to the Request management service to validate the anonymized request. This requires calling the DBMS to count the number of individuals concerned.

Without this double check, if a third party never refreshed the webpage since the moment his request was accepted (so since the moment the download data button was available) and if during this time the number of individuals concerned by the request drop under 1000, a third party would be able to acquire unauthorized data.

If the request is anonymizable, the Website rooting service asks the DBMS to get the needed data. Then the data access service anonymizes the data and sends the result to the third party.

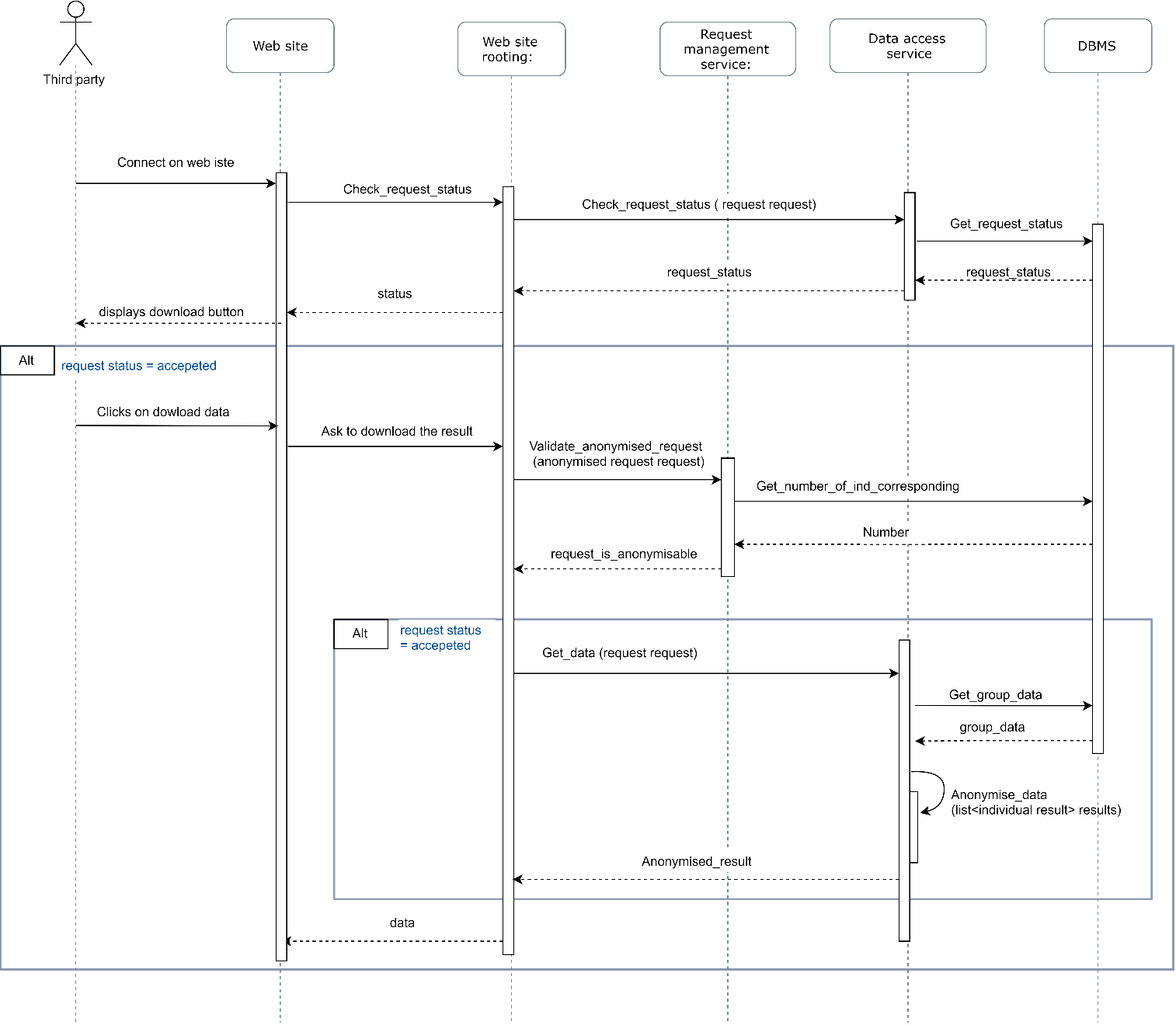


Figure 12: A third party download data from an anonymized request

### A third party download data from an individual request

When a third party logs in the Data4Help website, a request is made to the Website rooting service in order to know what needs to be displayed on the third party’s computer screen. If the third has some accepted requests, the download button will appear on his screen.

When the third party clicks on the download button, the Website rooting service requests to the Data access service to get the required data. To do so, the data access service checks the request status and if the request is accepted, ask the needed data to the DBMS. Then the data is sent to the third party.

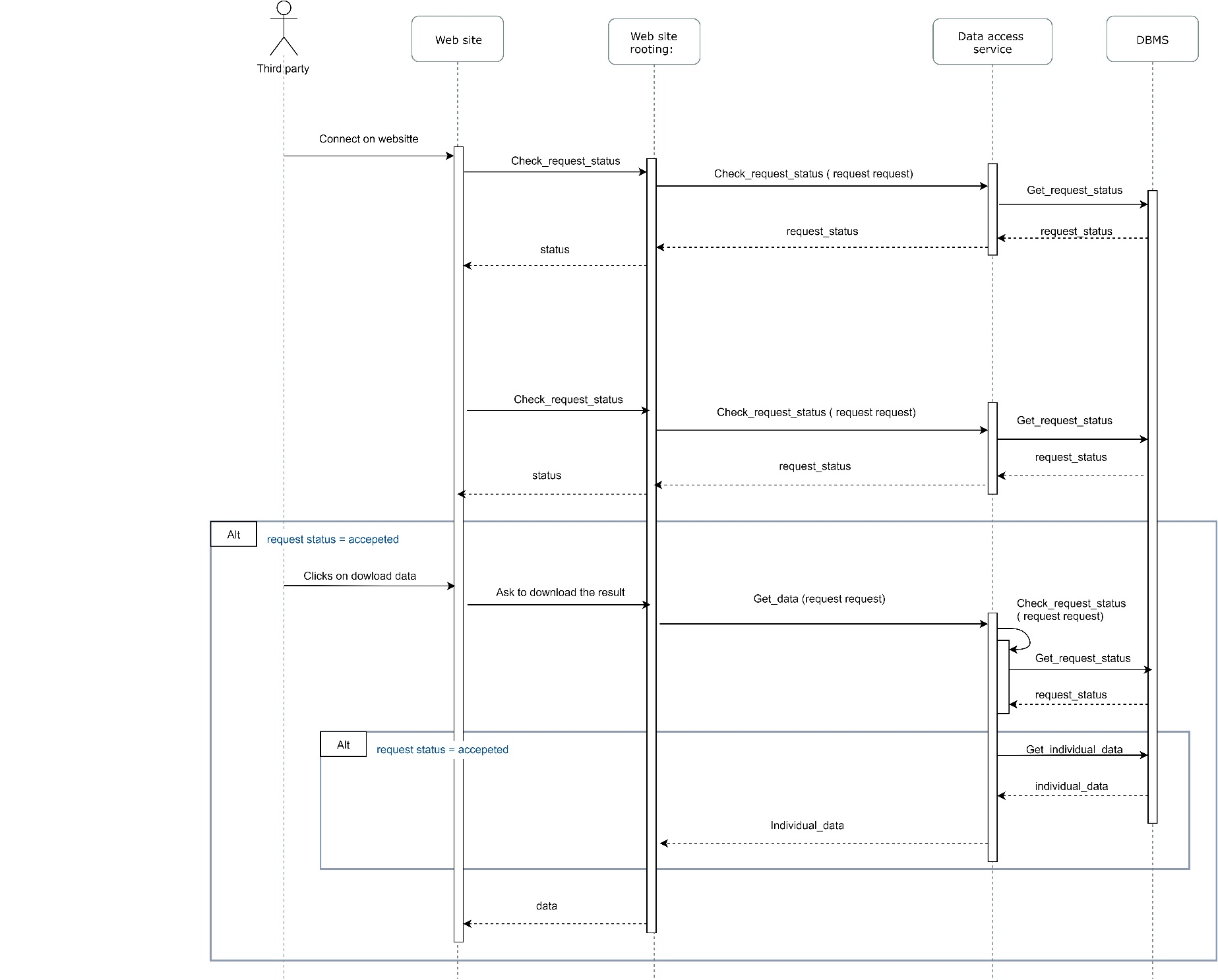


Figure 13: A third party download data from an individual request

### AutomatedSOS

The service AutomatedSOS must send an SOS to an ambulance service when the health parameters of the user are below certain thresholds. The service must therefore be functional even without an internet connection.

To do so, the emergency data (the thresholds and the phone number of the closest emergency service) must be regularly updated and stored in the smartphone internal database when an internet connection is available (every five minutes). This is what is shown on the first sequence diagram (figure XXX).

Therefore the AutomatedSOS service will be able to detect when the individual is bellow threshold and to send the emergency SMS using the data collected when an internet connection was available.

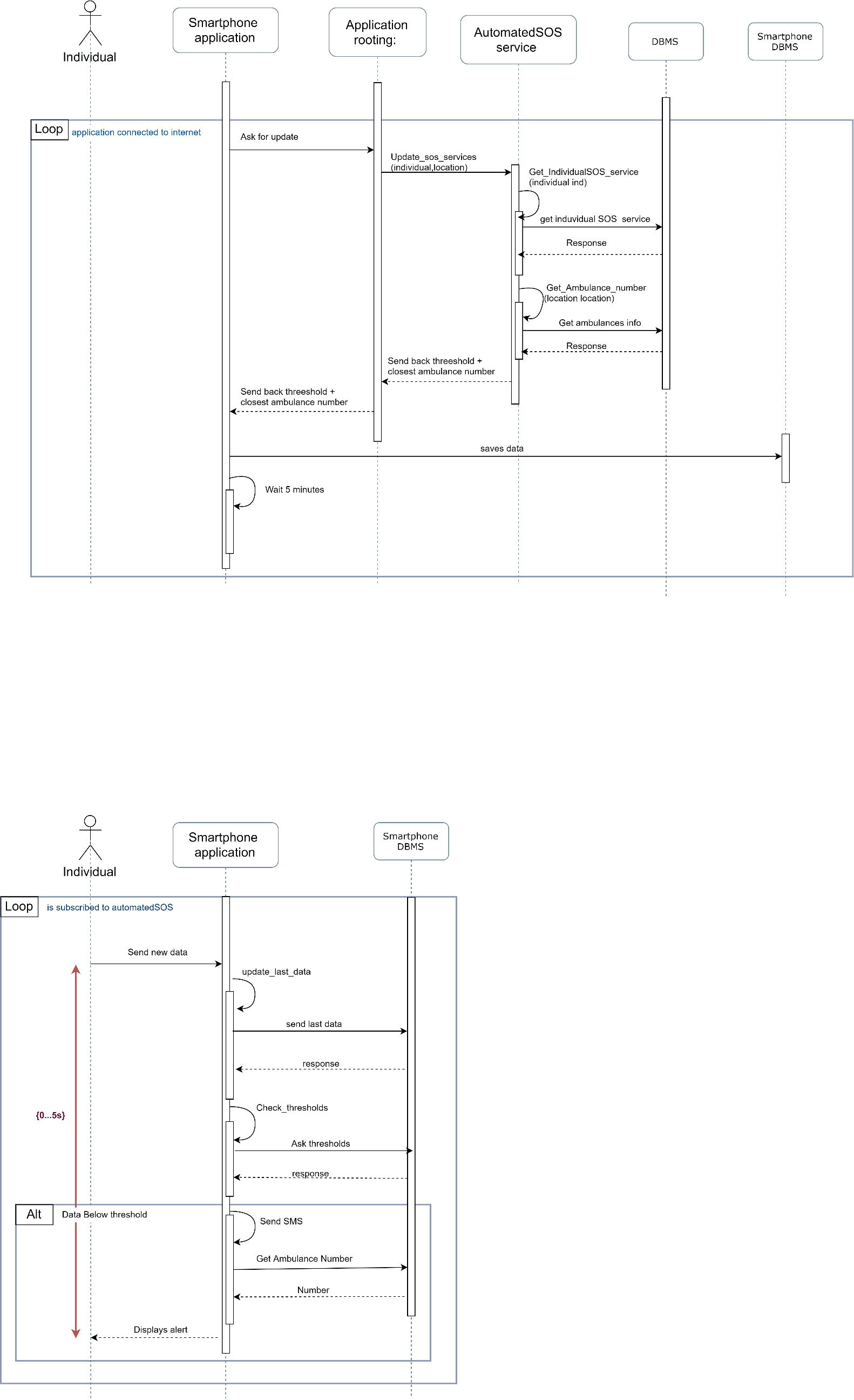


Figure 14: Updating emergency information when online

The following diagram shows the process of sending an emergency SMS. This process does not need any internet connection has it uses only the data stores in the smartphone database.

The smartphone application acquires new data from the users, saves the data in the internal database and check if the thresholds are exceeded. If they are, the smartphone sends an SMS to the closest emergency service. The process from the acquisition of new data to the sent of the SMS must last less than 5 seconds.

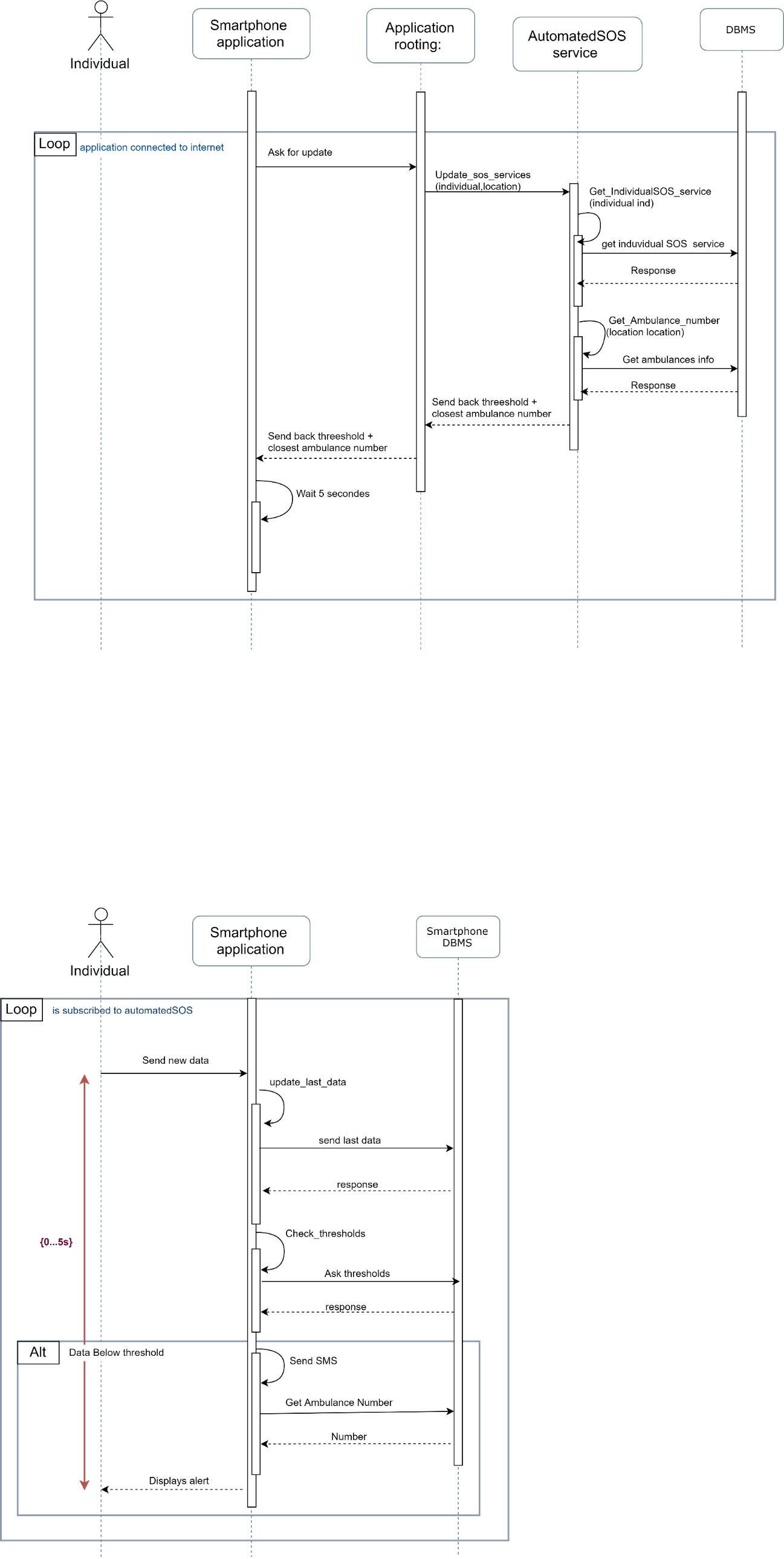


Figure : Sending SOS

## Component interfaces

## Selected architectural styles and patterns

# User interface design

# Requirements traceability

# Implementation and test plan

# Effort spent

# References