E-Health Service
Design Description

Version 2.0

Revision History

Date	Version	Description	Author
2012-11-04	0.1	Initial Draft: Introduction	Gregorio Perego Stefania Pezzetti
2012-11-05	0.2	Defined External Interfaces for Web App	Gregorio Perego Stefania Pezzetti
2012-11-07	0.3	Defined External Interfaces for Mobile App	Gregorio Perego
2012-11-08	0.4	Defined Software Architecture and Detailed Software Design	Marko Kelava
2012-11-08	0.5	Corrected some diagram	Stefania Pezzetti Vedran Šikić
2012-11-09	0.6	Corrected some diagram	Bojan Kosanović Marko Kelava
2012-11-09	1.0	Revised document format	Gregorio Perego Stefania Pezzetti
2012-12-17	1.1	Inserted database description and diagram	Stefania Pezzetti
2012-12-28	1.2	Updated class diagram. Revised the entire document.	Stefania Pezzetti.
2012-01-20	2.0	Revised the entire document.	Stefania Pezzetti

Table of Contents

1. Introduction

1.1 Purpose of this document

This document is a detailed description of system design of E-Health Service project. It needs to be a stable reference, outlining all parts of the software and how they will work: it follows the Requirements Definition document, and should be continuously revised during the development.

1.2 Intended Audience

The intended audiences for this document are:

- *Team members* to use it as a reference during the Implementation phase of the project.
- *Customer* to understand the project scope.
- *Project supervisor* to have an overview of the project work.
- *All people responsible for monitoring the project.*
- *Future developers of the system* that should know the basics of the system before starting upgrading it.

1.3 Scope

Scope of this document is to provide an insight into detailed design of the E-Health Service project: the document is commanded to give a fairly complete description of hardware, software and communication interfaces, software architecture and software designs, so it will be useful to give development team an overall guidance of project architecture. It obviously does not include requirements analysis description, which have been described in the previous Requirements Definition document.

1.4 Definitions and acronyms

1.4.1 Definitions

Keyword	Definitions
E-Health Service	Project Title
Web App	The E-Health Service web application
Mobile App	The E-Health Service web application

1.4.2 Acronyms and abbreviations

Acronym or abbreviation	Definitions
MVC	Model View Controller
CSS	Cascading Style Sheets
GUI	Graphical User Interface
HTML	Hypertext Markup Language
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
JSON	JavaScript Object Notation
MSSQL	Microsoft SQL Server
AJAX	Asynchronous Javascript and XML
IIS	Internet Information Server
REST	Representational State Transfer
URI	Uniform Resource Identifier
SSL	Secure Sockets Layer
3G	3rd generation of standards for mobile telecommunication services
GPS	Global Positioning System

1.5 References

Main references to this project can be found on the DSD course official webpage:
http://www.fer.unizg.hr/rasip/dsd/projects/e-health_service

Previous documentation:

- Project Plan:
  http://www.fer.unizg.hr/_download/repository/Project_Plan%5B5%5D.pdf
- Requirements Document:
  http://www.fer.unizg.hr/_download/repository/Requirements_Definition%5B6%5D.pdf

2. Software architecture

Software architecture is the set of structures needed to reason about the system, which comprise software elements, their properties and relations among them. Documenting software architecture facilitates communication between our stakeholders and defines early decisions about high-level design.

2.1 Conceptual design

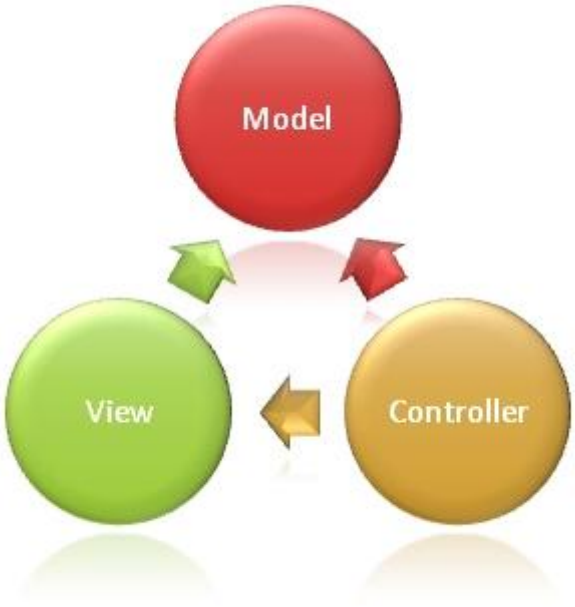
In this section we identify components that constitute our system and external components/systems that surround it.

2.1.1 Client - server architecture

The E-Health Service is a web based service implemented using classic client - server architecture. HTTP is used for client - server communication. The client can use either a mobile app or a web browser to connect to the service.

While the mobile app connects to the service directly and gets back the data in JSON format, the browser must first send a request to the web server, who then connects to the service, does the work, and sends data to the client in HTML. Web server also gets data from service in JSON format. The service consists of a business logic and database.

2.1.2 MVC



Both service and web server will be done using a Model-View-Controller (MVC) architectural pattern, although service will not be using the views. This pattern allows us to separate the application logic from user interface, thus permitting us to develop, test and maintain each separately.

2.1.3 REST

The architectural style of choice is Representational State Transfer (REST). REST is intended to evoke an image of how a well-designed web application behaves: presented with a network of web pages (a virtual state-machine), the user progresses through an application by selecting links (state transitions), resulting in the next page (representing the next state of the application) being transferred to the user and rendered for his use. This is accomplished by assigning each resource (an object, method etc.) a unique URI, and exposing the interface through those URIs.

2.2 System specification

The E-Health Service consists of a web service and a web server hosted on a virtual machine. The MS SQL database management system is also used, and hosted on the same virtual machine.

The technologies used to implement the system are:

- Frontend
  - ASP.NET
  - HTML
  - CSS
  - jQuery
  - AJAX
  - Android SDK
  - Windows Phone 8 SDK
- Backend
  - C#
  - Entity Framework 5
- Database
  - MS SQL Express
- Hosting
  - Microsoft Server
  - IIS

2.3 Error handling

The table below reports possible errors which can emerge during the utilization of our application and related responses from the system.

Error	**Action**
User input error	Cancel the current input. Notify the user.
Data access error	Notify the user.
Service error	Notify the user.
Server error	Notify the user.
Client error	Notify the user.

3. External interfaces

This section provides an overview of interfaces for communication between the product and systems/entities/humans outside the system borders. Hardware, software, communication and graphic user interface have to be defined in order to develop, in implementation phase, a system that enables a simple and intuitive service for users.

The system itself is exposed to the users in two different ways: web application, which offer system functionalities available to all users, and mobile application, which offers a limited set of functionalities to doctors and patients.

3.1 Hardware Interfaces

E-Health Service is currently designed for working without integration with medical sensors, which patients have to use for monitoring their vital parameters.

The system operates similarly on various mobile devices which support Android operating system. As we said in the requirements definition document, system must be able to detect patient's position when he's using mobile application for sending data, so we have to use (without consuming a lot of battery) GPS technology: of course we have to assume that patients' mobile devices are equipped to use this system.

When patient uses web application, he fills form optionally with address information and then, using Google Maps API and JavaScript we should easily display maps in our web pages.

In addition our system uses emails to alert doctors if their patients insert a measurement where vital parameters are out of range. So if in doctors' mobile devices is installed a tool that allows email management (e.g. Gmail for android) doctors can see immediately the alert.

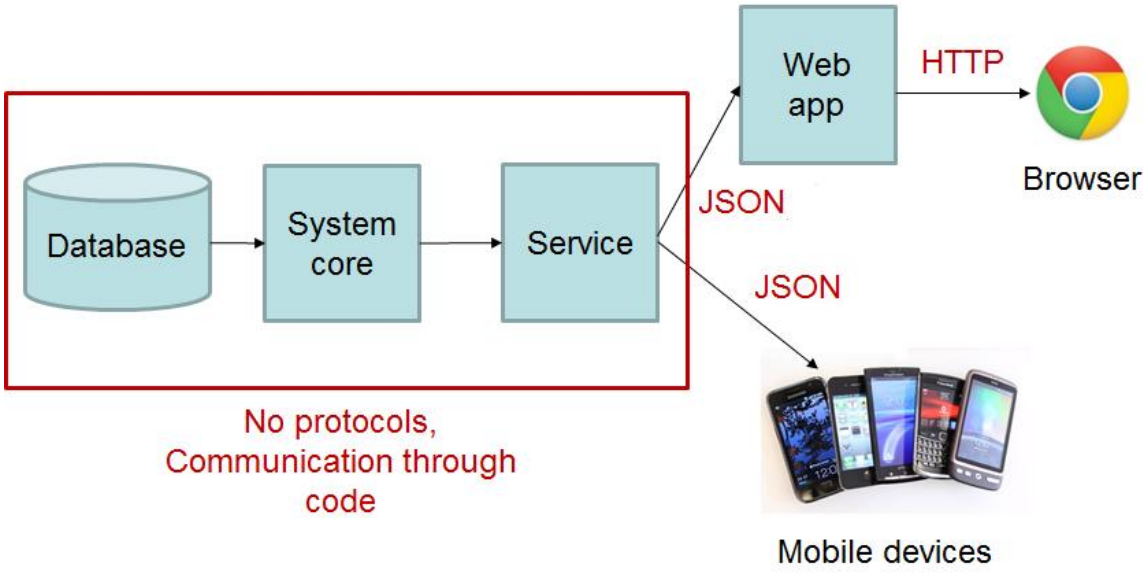
3.2 Software and Communication Interfaces

Access to system functionalities requires Internet connection, therefore doctors and patients using mobile devices need to have 3G connection active.
The communication protocol between mobile devices and service and between service and web app is HTTP, on which data are exchanged using JSON format.

JSON (JavaScript Object Notation) is a text-based open standard designed for human-readable data interchange and it is derived from the JavaScript scripting language for representing simple data structures and associative arrays, called objects. Despite its relationship to JavaScript, it is language-independent, with parsers available for many languages (included C# and Java). We'll use JSON as a valid alternative to XML, for serializing and transmitting structured data over the network, offering good performance and scalability.

HTTP functions as a request-response protocol in the client-server computing model. The client submits an HTTP *request* message to the server. The server, which provides *resources*, or performs other functions on behalf of the client, returns a *response* message to the client. The response contains completion status information about the request and may also contain requested content in its message body.

As regards Browsers we'll optimize our web application for working well on Google Chrome.



3.3 User Interfaces

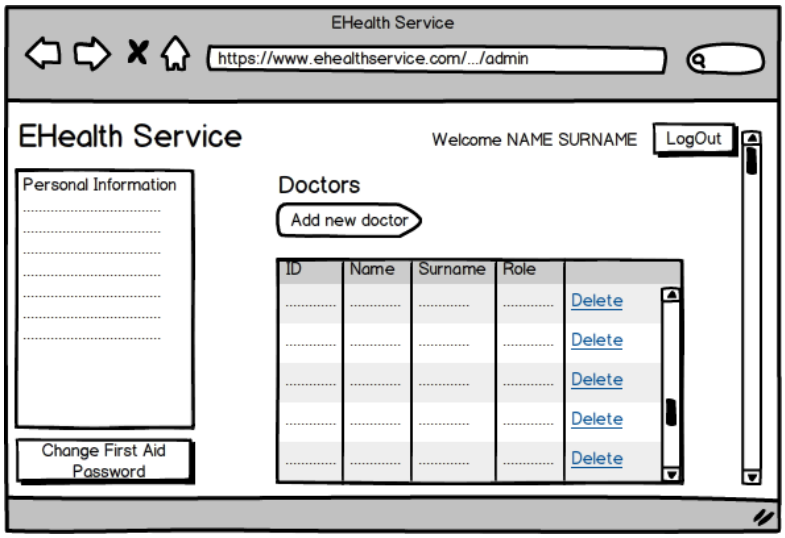
Designing the visual composition and temporal behavior of GUI is for us an important part of software application programming in the area of human-computer interaction: what we want is, especially for mobile, offering to users intuitive and user-friendly interfaces. Of course graphical user interfaces offered by mobile application are not the same offered by web application but, as we said previously and as you can see in the figures reported below, a lot of offered functionalities are common.

3.3.1 Web application user interface

The user web interface of the E-Health system is a user-friendly web application, based on HTML, CSS and ASP.NET. Each user, except outside user, must login with their given username and password for accessing their proper functionalities. Outside users can consult doctors' information through a link in the homepage that lead to a second page where doctors' public information is displayed.

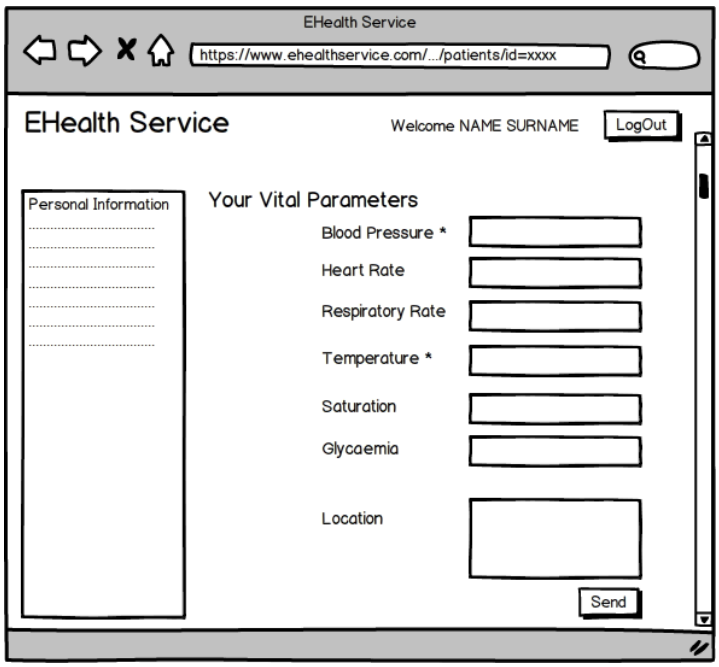
We have four categories of user: administrator, patient, doctor and first aid. Above you can see our first version of some Graphical User Interface, one for each kind of user.

3.3.1.1 Administrator role



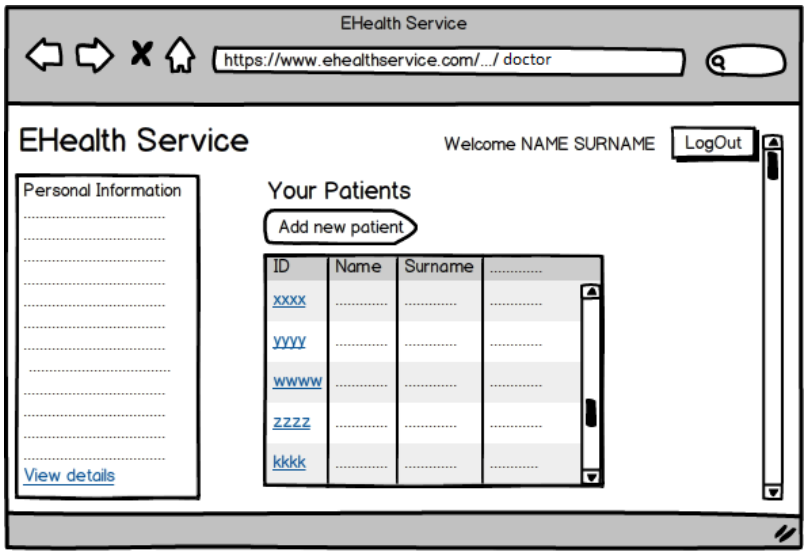
On the top, every user (not only the administrator) can see the main navigation of the web application, which will stay consistent over all pages. On the left side we putted a column containing personal information of the Administrator and a button which lead to the functionality of changing first aid password, while in the body we represented the main part dedicated to doctors' accounts management.

3.3.1.2 Patient role



The main functionality of a patient is sending vital parameters' measurement. In the sketch mandatory fields, determined by doctors for each patient, are marked with *. Initially we thought the patient personal area as you can see in the picture. Then we developed patient's personal area in a slightly different way, decoupling vital parameters insertion from update location. So patient can update his location without insert his vital parameters.

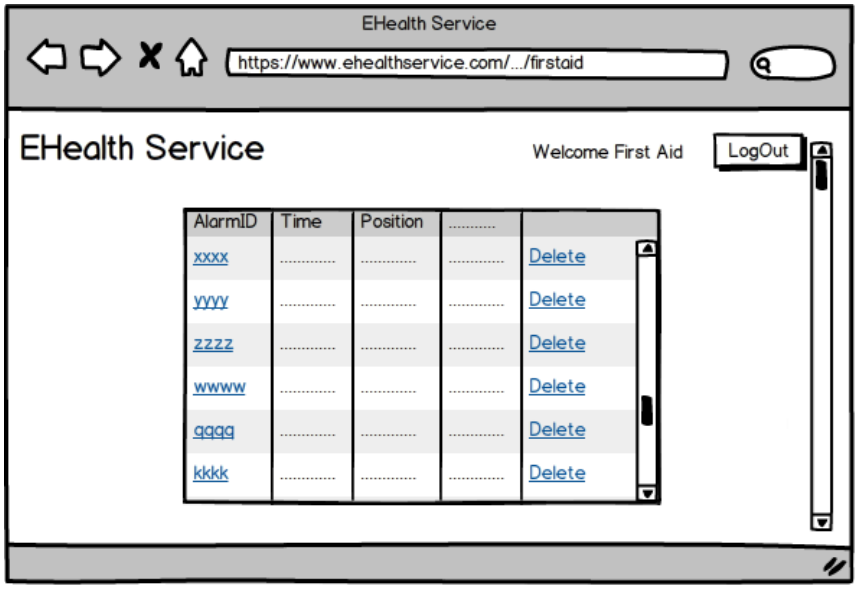
3.3.1.3 Doctor role



On the section dedicated to report doctor's personal information there's a link that can be clicked when doctor

would read and modify his public information details. List of doctor's patient is displayed in the body: selecting a patient, doctor can access to a second page where he has the possibility to view and manage every patient's information, basically all measurements received and medical history.

3.3.1.4 First Aid role

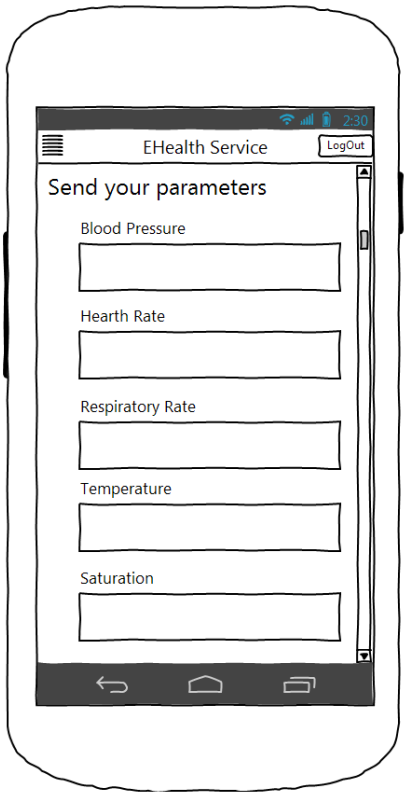


Firs aid uses information coming directly (in real time) from patients and doctors, but also as results of system elaboration, for managing emergency situations and organizing rescue team. With one click first aid can access to a page where details about a particular alarm notification are displayed.

3.3.2 Mobile application user interface

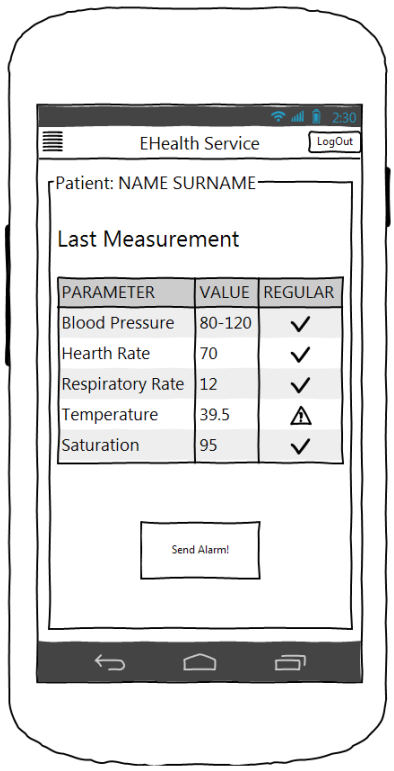
As we said before, GUI for the mobile application should be intuitive and user-friendly and designed for being used only by doctors and patients: the following figures illustrate a couple of GUI (offered to these actors).

3.3.2.1 Patient role



After logging into the system, patient can use mobile application for sending measurement of his vital parameters to his doctor. Basically he has to fill some form and "submit" values, without manually indicate his position (differently from web application), which could be automatically revealed by GPS.

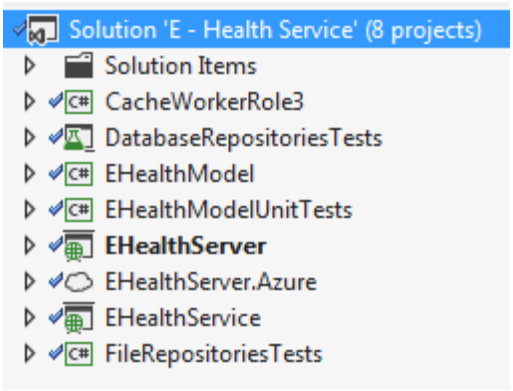
3.3.2.2 Doctor role



As we said in requirement definition document, doctor receives an email whenever a value of some parameter inputted in a measurement sent by one of his patients is out of "regular" range. When doctor decides taking a look to that particular measurement he has to login into the system and access to last measurement received. If he uses a mobile device he finds a GUI similar to that reported above. As you can see, he has the possibility, in addition to reading values, to prepare alarm message for the first aid.

4. Detailed software design

This section provides details about product structure and intended implementation. Many of diagrams reported below are in UML (Undefined Model Language), a standardized general-purpose modeling language in the field of object-oriented software engineering.



4.1 Implementation modules / components

The project will be divided into following modules:

- DatabaseRepositoriesTest - contains database tests
- EHealthModel - contains model classes
- EHealthModelUnitTests - contains JUnit test for Model
- EHealthServer - contains web server classes
- EHealthService - contains web service classes
- FileRepositoriesTest - contains tests for file repositories
- CacheWorkerRole3 and EHealth Server.Azure are two folders needed for the deployment

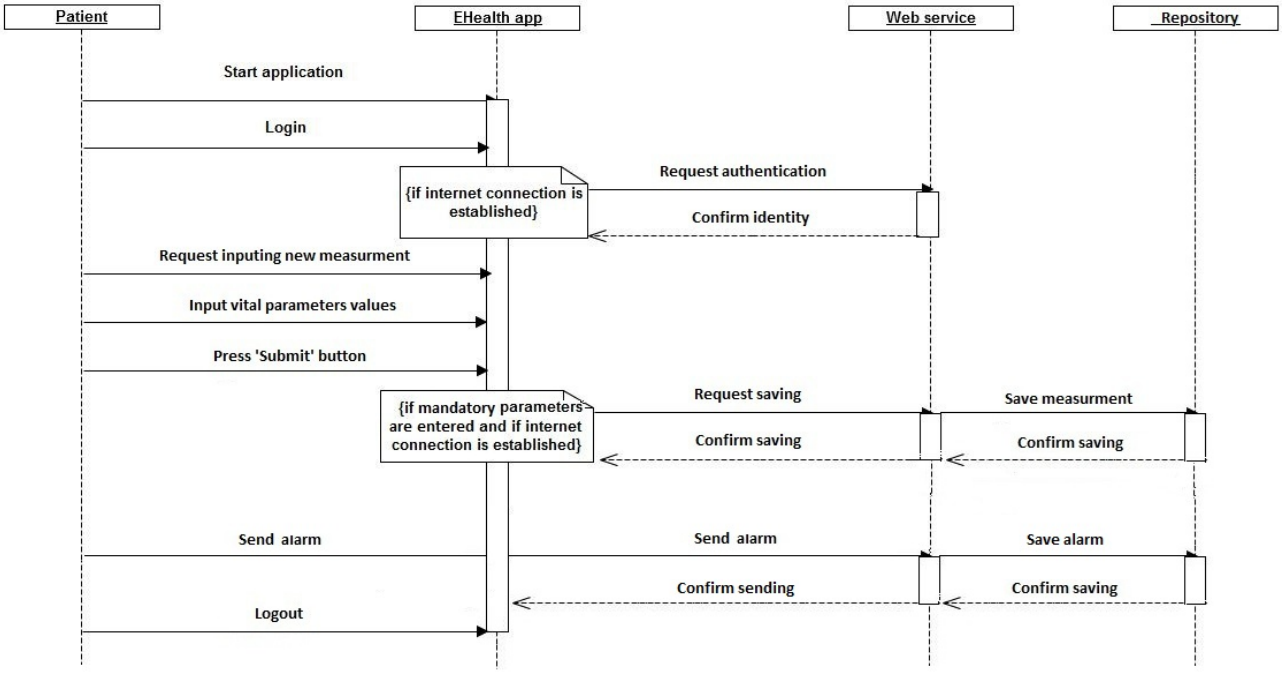
In addition there are two folders. The first one contains the Android code, the second one the Windows phone code:

- EHealthService – Android - contains Android code
- EHealthService WP - contains Windows Phone code

4.2 Sequence diagram

A sequence diagram, related to Use Cases previously defined, describes the dataflow and interaction between system entities. It shows different processes or objects that live simultaneously, and messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a useful graphical manner.

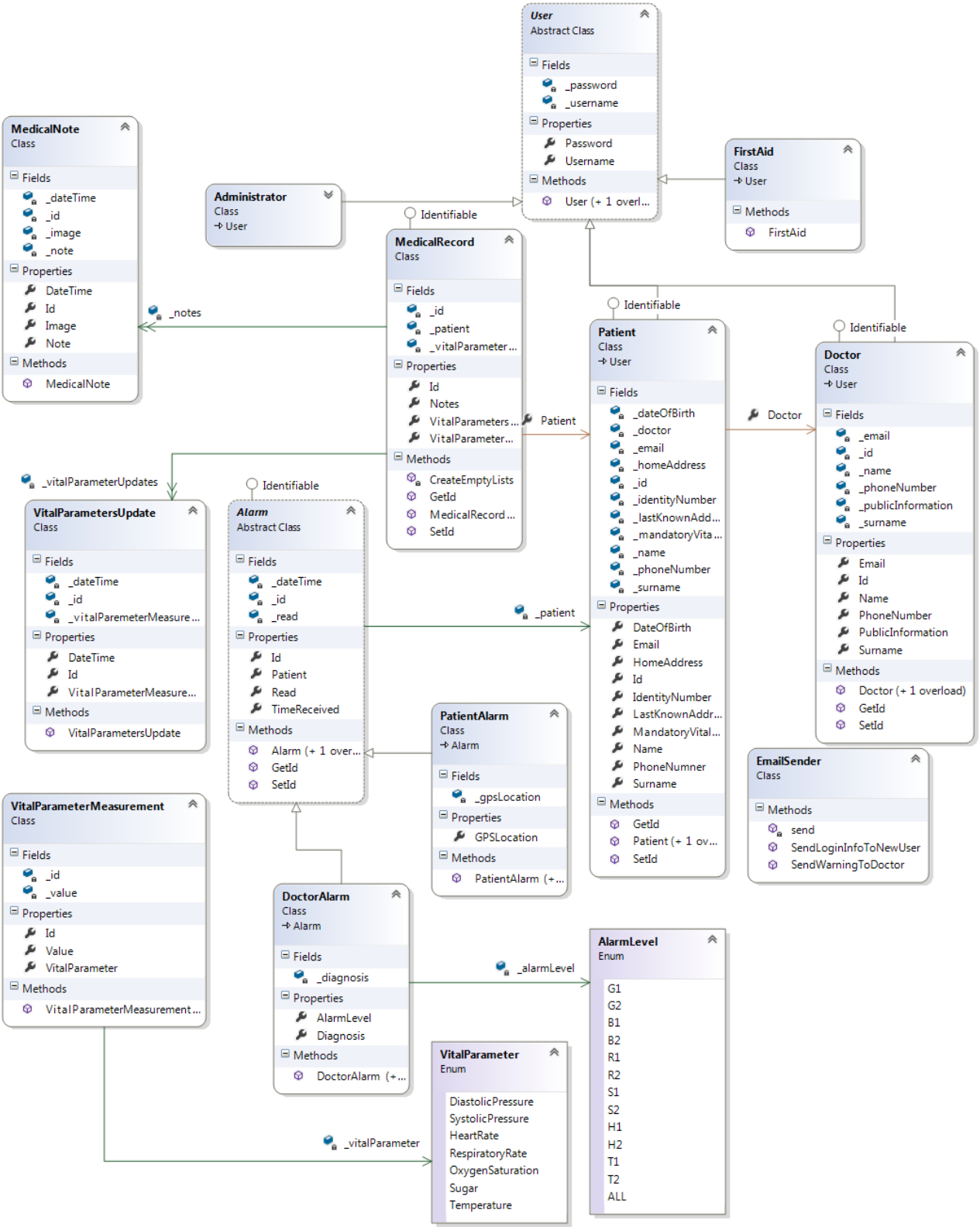
Below there is a diagram, as example to see how our system works, of a patient, firstly inputting a new vital parameter measurement, and then sending an alarm. Uses Cases involved are PATIENT1 and PATIENT3, specified in Requirements Definition Document.



4.3 Class Diagram

A class diagram describes the structure of a system by showing system's classes, their attributes, methods and relationships among classes.

The following class diagram shows the server and service side of our system: note that not all methods are included.

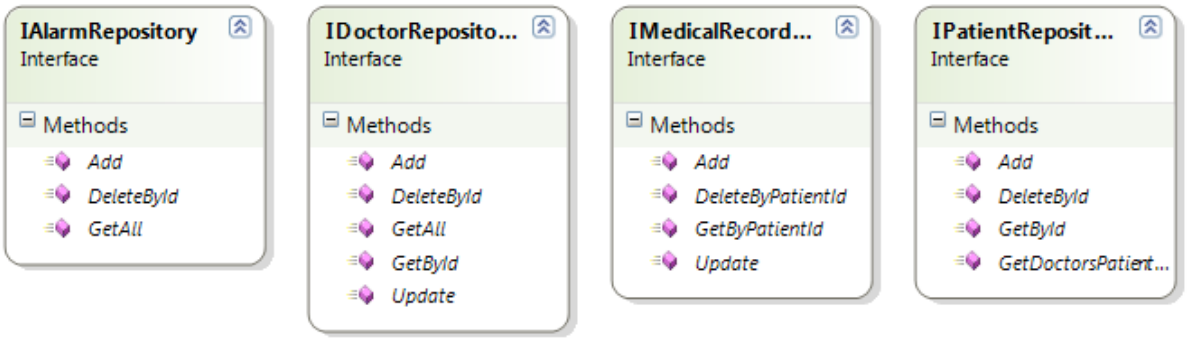


4.4 Database

MS SQL Server 2008 will be used as a database engine. Entity Framework will be in used for object-relational mapping. Initially, we did not intend to use database: at first we used simple data repositories and when we were certain our model was correct and sufficient, we created a database and connected the service to it. This made also testing easier, but that was why at the beginning we did not have a database diagram.

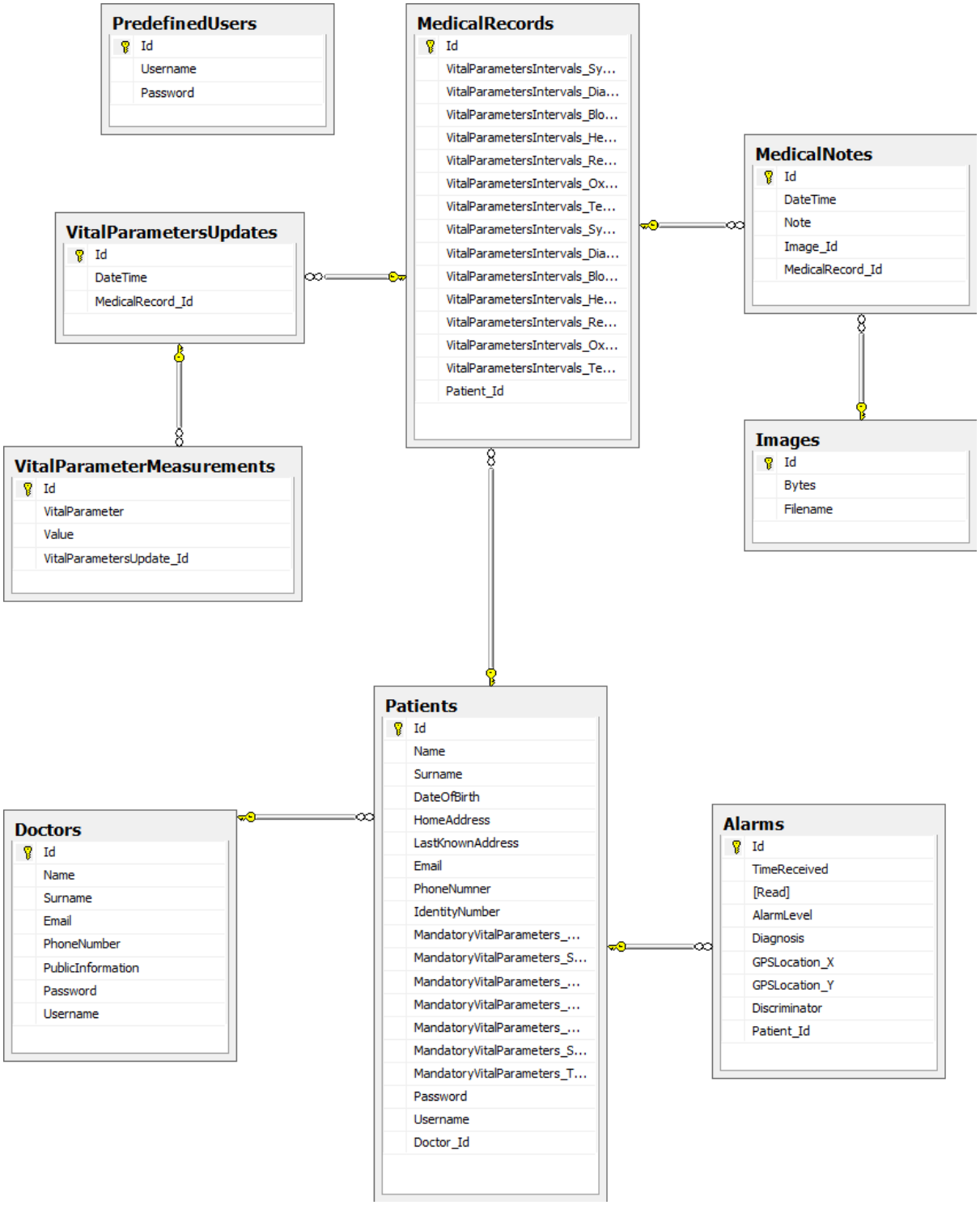
4.5 Repository diagram

The picture below shows the diagram of our data repositories. Data will be stored in XML files.



4.6 Database Diagram

Between Alpha prototype presentation and Beta prototype presentation we decided to switch from simple data repositories to a real Database. The schema below represents the database model diagram.
As the image shows in our database we have these entities: Patients, Doctors, Medical Records, Medical Notes, Alarms, Images, Vital Parameters Updates, Vital Parameters Measurements and Predefined Users.
Each doctor has some personal information, he has also public information that can be seen by every users. Each patient, like doctors, has personal information, information regarding mandatory vital parameters (defined by his doctor), his home address and last known address and obviously a doctor.
Each alarm is composed by Time Received, Alarm Level and Diagnosis (setted by doctor), GPS coordinates, useful in identifying the location of emergency, patient Id to determine which patient is asking help, a flag to indicate if the first aid has read or not the alarm and a discriminator to understand who sent the alarm (doctor, patient or system).
Each Medical Record contains information about patient Medical Notes (written by doctor), vital parameters range (decided by doctor specifically for each patient) and Vital Parameters Updates (inserted by doctor or patient).
Each Medical Note has a text field that explains note content, a date and eventually and image or in general a file.
Finally each Vital Parameters Updates has a date and a time and a value for each vital parameter measurement.


PredefinedUsers
Id
Username
Password
MedicalRecords
Id
VitalParametersIntervals_Sy...
VitalParametersIntervals_Dia...
VitalParametersIntervals_Blo...
VitalParametersIntervals_He...
VitalParametersIntervals_Re...
VitalParametersIntervals_Ox...
VitalParametersIntervals_Te...
VitalParametersIntervals_Sy...
VitalParametersIntervals_Dia...
VitalParametersIntervals_Blo...
VitalParametersIntervals_He...
VitalParametersIntervals_Re...
VitalParametersIntervals_Ox...
VitalParametersIntervals_Te...
Patient_Id
MedicalNotes
Id
DateTime
Note
Image_Id
MedicalRecord_Id
VitalParametersUpdates
Id
DateTime
MedicalRecord_Id
Images
Id
Bytes
Filename
VitalParameterMeasurements
Id
VitalParameter
Value
VitalParametersUpdate_Id
Patients
Id
Name
Surname
DateOfBirth
HomeAddress
LastKnownAddress
Email
PhoneNumner
IdentityNumber
MandatoryVitalParameters_...
MandatoryVitalParameters_S...
MandatoryVitalParameters_...
MandatoryVitalParameters_...
MandatoryVitalParameters_...
MandatoryVitalParameters_S...
MandatoryVitalParameters_T...
Password
Username
Doctor_Id
Doctors
Id
Name
Surname
Email
PhoneNumber
PublicInformation
Password
Username
Alarms
Id
TimeReceived
[Read]
AlarmLevel
Diagnosis
GPSLocation_X
GPSLocation_Y
Discriminator
Patient_Id