Project Description

1. Introduction

This document is a description of the final project for the Introduction to Service Design and Engineering course. This project implements a virtual life coach system. The system supports people in daily life in order to improve their lifestyle. To support and control the user, the system provides also an interface for the doctor and a family member.
Each user is associated with a doctor, who check and follow the evolution of the health.
The family member covers an important role in the improvement of the health of the user. They can motivate the user improving the commitment.

2. Description

During the initial requirements phase we defined three actors: user, doctor and family. For each actor we analyzed the goals and how to achieve them. See the wiki of the project to see the models generate during the goal analysis. In the wiki you can find also the use case.
In the following paragraphs there is a list functionalities for each actor.

Person

The system provides the following functionalities:

- insert new measures regarding own health and lifestyle (e.g. height, weight, steps, heart rate and other);
- set new target to achieve. For example, reach a certain weight in defined period of time;
- insert a reminder. A reminder are notifications that are sent to the user to remind some action to perform or to inform about particular situations. Examples are: warning when heart rate or blood pressure exceed a certain range, remind to go to the doctor, remind to measure blood pressure every morning, motivate the user with a motivational phrase;
- get information about the weather and weather forecast to permit the user to plan activities during the following days.

Family

The action that can be performed by the family member associated with the user are:

- he/she can receive alarm if the vital signs exceed a specific range;
- visualize data about the user.

Doctor

The doctor can check measurements inserted by the user to determine the health situations. The doctor can send notification to the user. Each user is associated with a doctor.

3. Software Architecture

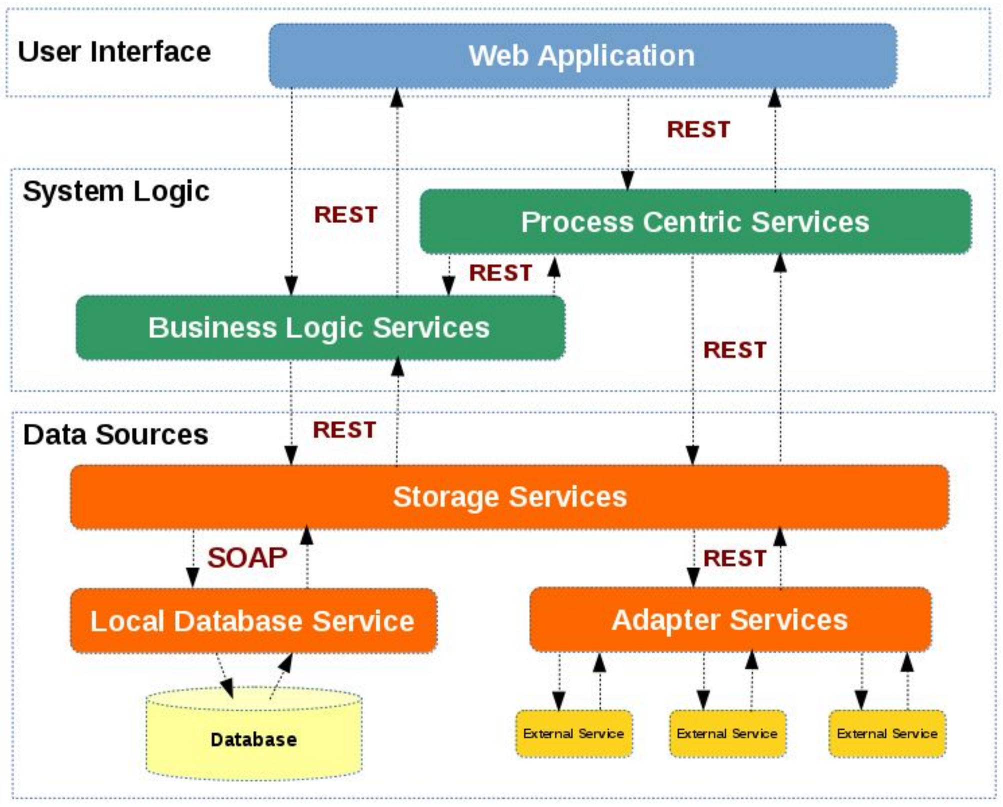
All the services, apart the user interface is implemented with Java. The architecture is divided in three layers. The lowest level is responsible for the management of the data sources and it is composed by three services:

- Local Database Service (LDS), SOAP web service. It is responsible for the management of the database. It provides methods for CRUD operation on the entity and in addition some useful methods;
- Adapter Service (AS), REST web service. It interacts with external data sources, operating on external web services, to retrieve data using external API. We use the following external web services:
    - weather http://openweathermap.org/current and weather forecast http://openweathermap.org/forecast5 (REST web service);
    - motivation phrases http://www.icndb.com/api/ (REST web service).
- Storage Service (SS), REST web service. This service redirects request for the data sources from the upper level to the correct web service: CRUD operation on the entity redirected to the Local Database Service and get weather information and motivational phrases to Adapter Service;

The middle level, the system logic, is composed by two services:

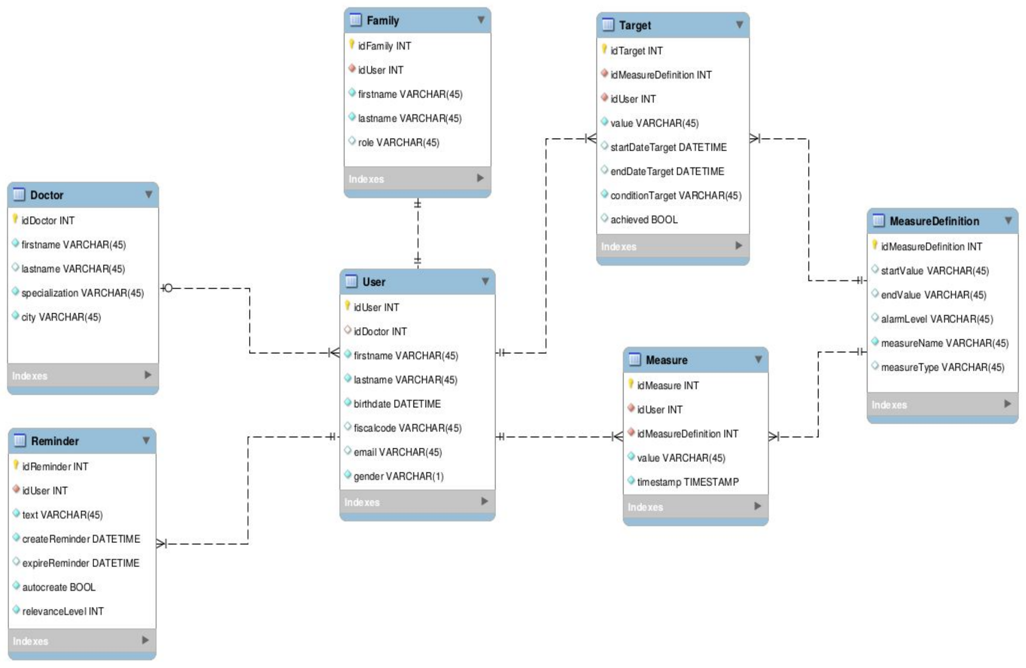
- Process Centric Services (PCS), REST web service. This layer receives requests from User Interface and redirects them to BLS or to SS. This service includes functionalities that require integration of various methods of BLS and SS.
- Business Logic Services (BLS), REST web service. It performs manipulations of the data and decisions. This layer receives requests from User Interface and from the PCS and gets the data from the SS to send results back.

The highest level is the User Interface, that allow the users to visualize, insert, modify data. We use the Telegram Bot platform. Bot are special Telegram accounts managed by software. The bot handle messages automatically. The user send command messages in chants and the bot perform various action in order to respond. Our bot is implemented using Ruby and with the help of the library telegram-bot-ruby. The bot uses HTTPS request to communicate with Telegram and the API of Business Logic Service and Process Centric Service to interact with our system.



4. Data Structure

In the following model you can see the ER structure of the database of LDS. For a better view the image is available [here]().


Family
idFamily INT
idUser INT
firstname VARCHAR(45)
lastname VARCHAR(45)
role VARCHAR(45)
Indexes
Target
idTarget INT
idMeasureDefinition INT
idUser INT
value VARCHAR(45)
startDateTarget DATETIME
endDateTarget DATETIME
conditionTarget VARCHAR(45)
achieved BOOL
Indexes
Doctor
idDoctor INT
firstname VARCHAR(45)
lastname VARCHAR(45)
specialization VARCHAR(45)
city VARCHAR(45)
Indexes
User
idUser INT
idDoctor INT
firstname VARCHAR(45)
lastname VARCHAR(45)
birthdate DATETIME
fiscalcode VARCHAR(45)
email VARCHAR(45)
gender VARCHAR(1)
Indexes
MeasureDefinition
idMeasureDefinition INT
startValue VARCHAR(45)
endValue VARCHAR(45)
alarmLevel VARCHAR(45)
measureName VARCHAR(45)
measureType VARCHAR(45)
Indexes
Measure
idMeasure INT
idUser INT
idMeasureDefinition INT
value VARCHAR(45)
timestamp TIMESTAMP
Indexes
Reminder
idReminder INT
idUser INT
text VARCHAR(45)
createReminder DATETIME
expireReminder DATETIME
autocreate BOOL
relevanceLevel INT
Indexes