Introduction to Service Design Final Project
Motivational Coach

29/01/2017
Davide Lissoni Mat.179878, Daniele Dellagiacoma Mat.182251
Department of Information Engineering and Computer Science
University of Trento

1.INTRODUCTION

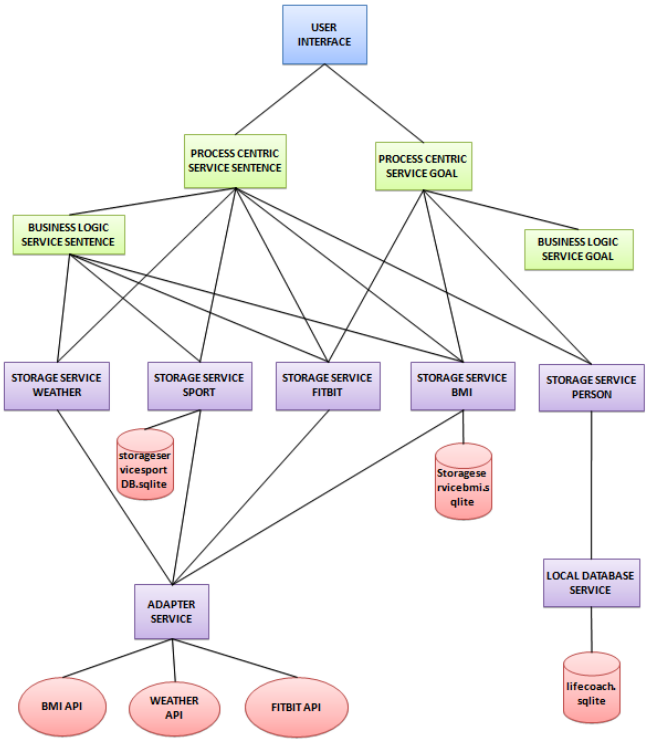
This report is about the final project implemented for Introduction to service Design course.

The project topic was to develop an application composed by a set of web-services which will track the users physical activities and will allow some other operation related to them and the user lifestyle in general.

Since already exist a large amount of web services and applications that allow users to track in details their activity, we move the topic forward thinking on how to persuade the users in order to make more activities, basing our algorithms on the users' activity and lifestyle and some external information.

“Motivational coach” then, generates sentences and advices that would entice the user to do more physical activity. The calculation are based on real time data monitoring.

The application is composed by two different System logics, one used in order to generate the motivational sentences based on users and external conditions while the other one is in charge to change the goals of the users drawing on the progress or the difficulties encountered by him/her.



2.IMPLEMENTATION

In this chapter we are going to explain the architecture of the application and its orchestration.

The project consist in a set of services interfaced each other, but completely reusable separately.

The architecture of the project follows the delivery requirements and can be split in three main parts: Data Source, System Logic and User Interface [Schema above].

2.1 Data Source

This part contains a set of 7 different services responsible for the data storage and some initial and basic data manipulation.

The project make use of both local and external databases and also make use of external api used in order to get some external detailed and accurate information.

2.1.1 LocalDatabaseService:

LocalDatabaseService is a RESTful service that provides CRUD operation on its data. This Service contains the application database and their models are:

Person: idPerson, firstname, lastname, birthdate, email, genre, password, nGoalAchieved, nTotalGoal, idLifeStyle, idLevel

LifeStyle: idLifeStyle, style, description

Level: idLevel, name, description, nGoalNecessary

2.1.2 AdapterService:

this web service provides the communication between the storage services and the external api. In particular this project make use of the following API:

Fitbit API: used in order to track activities, goal and to get some detailed information about sports (for example the Metabolic Equivalent of Task MET). The Fitbit API are used also in order to get some user information such as weight and height.

Weather API: used in order to get the local forecasts (this feature will be useful in order to advice some sport based on the forecast)

Bmi API: used in order to have a descriptive and accurate calculation of the bmi in real time.

2.1.3 StorageService:

Every storage service has been implemented as SOAP service and are used in order to perform all the methods allowed by the API listed above in order to get all the information necessary and perform some basic data manipulation on it. Some example in order to better understand the storage service functionalities:

- calculate and save the bmi ;
- get all the personal information;
- calculate the changing of weight, bmi value and bmi status of a person in a given period of time;
- calculate calories, steps and distance missing in order to achieve the user daily goal;
- get a list of appropriate sports in a certain weather condition

We created a different storage service for each different api that are used (i.e. StorageServiceFitibit is connected with the AdapterService and will make use of the Fitbit API, while the StorageServicePerson is connected to the LoacalDatabaseService and so on).

Storage service data sources:

StorageServiceBmi contains its own internal database used in order to save the current Bmi values and the historical bmi status of a user. All the bmi information are taken by the Bmi api.

Model:

Bmi: idBmi, idPerson, prime, risk, status, value.

BmiHistory: idBmiHistory, Date, idPerson, prime, risk, status, value.

StorageServiceSport: this database is used in order to save sport and activities information and to correlate sports with forecast weathers.

Model:

Sport: idSport, name, perfectWeather;

Activity: this table is used just in order to speed up the service by avoiding, whenever possible, to make use of the external API. The table contains the Fitbit activities information.

2.2 System Logic

This layer contains “higher level” services used in order to call the lower layer services for data accessing (Process centric services) and data manipulation (business logic services).

As already said in the Introduction chapter we implemented two different system logic in order to achieve our two goals:

2.1.1SystemLogicSentences:

This system logic is in charge to generate motivational sentences and advices based on the real time data collected using all the data source services listed above. For example:

- for a sunny day we will advice an outside sport and vice-versa;
- if the user just starts his daily goal the sentence generated will be something like: “come on you just start” while if he finished it the sentence will be “congratulation you're super”;
- if the user's goal is to lose weight but in the last days he is actually getting weight the sentence generated will be something like: “be careful you're not improving your weight goal!” etc..

Furthermore the system check once a day, how many goal the user has achieved in order to calculate and assign a motivational level of the person. The levels vary from “unwatchable” to “superman” and generate a sentence also based on it.

This system logic is composed by the BusinessLogicServiceSentences (SOAP service)and the ProcessCentricServiceSentences (SOAP service).

2.1.2 SystemLogicGoals

This system logic is in charge to check and change the goals of the users drawing on the progress or the difficulties encountered by him/her. A basic example is that if a user usually don't reach his steps daily goal, probably it is because the goal is too difficult for him in that particular period of time, and, maybe it's better to decrease the difficulty of the goal (reach a goal is inspiring), while if the

user always reach his goal, the goal pales into insignificance because it is too easy to reach, so the solution is to increase its difficulty.

This system logic is composed by the BusinessLogicServiceGoals (SOAP service)and the ProcessCentricServiceGoal (SOAP service).

2.1.3System Logic life-cycle

Every system logic has the same architecture and act in the same way, their life-cycle could be summarized as follows:

Process Centric Service:

1. get data from storage services;
2. pass the data to the business logic services;
3. get the business logic response;
4. return the response.

Business Logic Service:

1. manipulate the data passed as parameter;
2. return a string.

2.3 User interface

Since Fitbit API requires a web log in, the user interface consist in a web-application.

The application allow the following operations:

- to log in;
- to log in into Fitbit account;
- to register a new user;
- to log out.

The application perform the following operation:

- get the local position of the users;
- get and print all the real time user information;
- every 10 seconds call a random methods of a random System logic and print out the sentence or the result generated;
- once a day check the goal how many goal the user and increase its field regard to the response.

3 DEPLOYMENT AND NOTES

The user interface is already deployed on heroku on the following address:

https://userinterfacesde.herokuapp.com/

Note that the web application require a Fitbit account, make also sure to have enabled the localization on your browser.