Hes·so VALAIS WALLIS
Haute Ecole Spécialisée de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences Western Switzerland

Travail de Bachelor 2011

Filière Informatique de gestion

Agent-based Diabetes Monitoring with Android Mobile

Etudiant-e : Ivan Samardziev

Professeur : Michael Schumacher

www.hevs.ch

Preface

This bachelor thesis focuses on the Android application for women affected by Gestational Diabetes Mellitus (GDM) called G-DEMANDE. Gestational diabetes will be presented in order to understand the risks of the disease. All information useful for understanding the project and the Android application will also be shown in detail. The design and implementation of the application and architecture will be presented.

Thanks

I would like to take this opportunity to thank all the people who have contributed in some way to this reports particularly Mr. Michael Schumacher who initiated the project.

My thanks also go to Mr. Stefano Bromuri who supervised and mentored my work, for his attention, and for all the time he awarded to me, his advices were really useful and appreciated.

I also thank Mr. Johannes Krampf who was always ready to help me.

Sworn statement

I declare, by this document, that I have done the bachelor attached alone, without other assistance than those properly reported in the references, and that I have only used the sources explicitly mentioned. I will not give any copy of the report to third parties without the permission of the RF and the teacher responsible for monitoring the bachelor work, including applied research partner with whom I worked, except for those who provided the key information necessary for the writing of this work, and that I quote below:

- Michael Schumacher
- Stefano Bromuri
- Johannes Krampf

Sierre, 10 August 2010

Ivan Samardziev

Summary

Gestational diabetes mellitus (GDM) is a form of diabetes which affects women without previously diagnosed diabetes during their pregnancy. It can lead to delivery complications and death during and after birth. Regular glucose monitoring can lower the risk of bad results.

An Android application is used to collect frequently data entered directly by the patient. A web application allows the caretakers to easily analyze the collected values. This reduces the risk to the patient and saves time for the management of the disease allowing her to maintain a normal life.

The objective of this project was the development of the Android application that we will see in detail in this report. Following is a summary.

The application allows women to enter their different observed values like blood pressure, weight, glucose and symptoms. These data are sent to the pervasive healthcare infrastructure which monitors continuously these patient values and alerts the caretakers about any risks.

The application has a local database containing all the data entered by the patient that can be displayed in charts to be easily understandable by the patient. It also allows to work without an Internet connection and save the state of each value to determine if it was sent to the server or not. The unsent values will be sent at the next connection.

An encryption system has been put in place to ensure the protection of patient data. All data are stored in an encrypted form in the local database in order not to be readable by a hacker.

To ensure user authentication, a connection system by scanning barcode was set up. The patient should scan her barcode to enter the application, which will allow authenticating the user when saving data on the server

The application is also multilingual in order to be easily used by many people. At the moment the application is available in English, French and German but can easily be translated into another language if necessary.

Contents

Table of Figures

1 Introduction

1.1 Personal motivations

Having already done Android development through my studies at the HES-SO Valais, I wanted to deepen this knowledge. The brief description of the project pleased me. I found the idea of developing a medical application interesting.

1.2 Gestational diabetes mellitus

Wikipedia provides a good description of the gestational diabetes mellitus:

"Gestational diabetes mellitus (GDM) is a condition in which women without previously diagnosed diabetes exhibit high blood glucose levels during pregnancy. Gestational diabetes is caused when the body of a pregnant woman does not secrete excess insulin required during pregnancy leading to increased blood sugar levels.

As with diabetes mellitus in pregnancy in general, babies born to mothers with gestational diabetes are typically at increased risk of problems such as being large for gestational age (which may lead to delivery complications), low blood sugar, and jaundice. Gestational diabetes is a treatable condition and women who have adequate control of glucose levels can effectively decrease these risks."

1.3 Web application

The web application provides an interface for caretakers to visualize and analyze data and to introduce new data collected during a patient's visit. Caretakers can visualize treatments for patients as well as any data introduced by patients. An intelligent agent system will analyze the values and raise an alert to the caretakers about any risks.

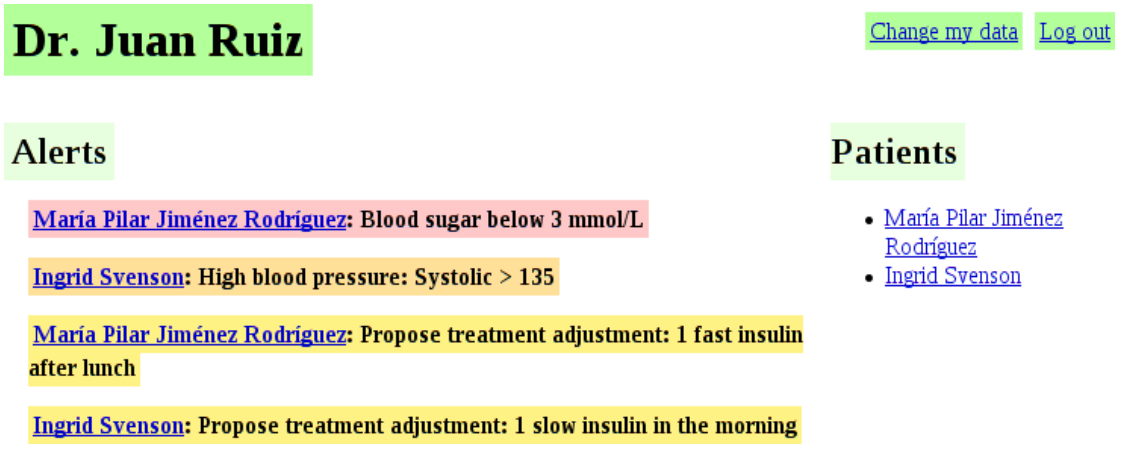


FIGURE 1: WEB APPLICATION

1.4 Android application

The Android application allows women to enter their different observed values like blood pressure, weight, glucose and symptoms. These data are sent to the pervasive healthcare infrastructure which monitors continuously these patient values and alerts the caretakers about any risks.

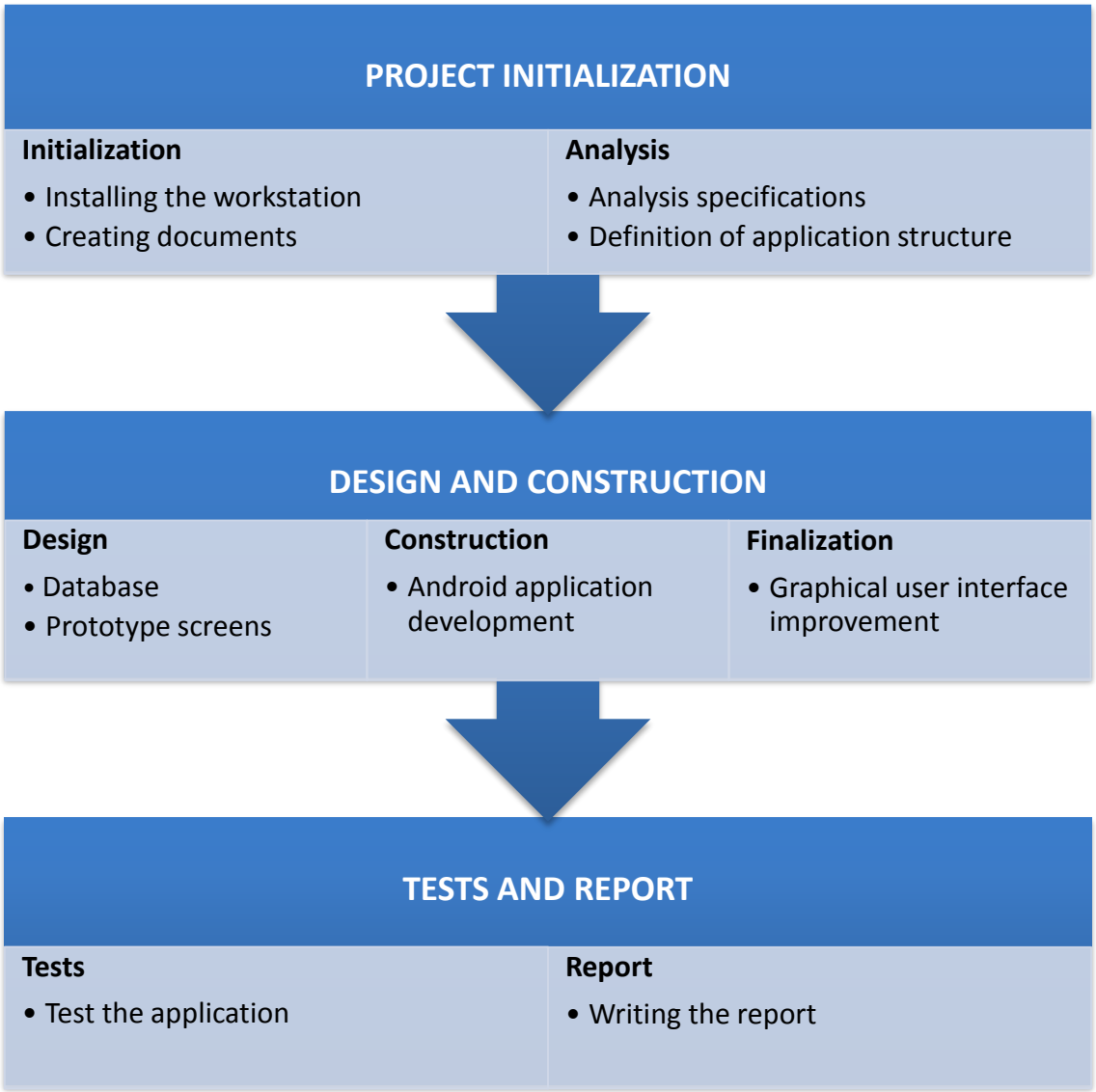
All values entered by the patient are also stored locally on the mobile. This allows the application to work even without an Internet connection. Indeed, the data entered offline will be recorded as not-sent and will be sent at the next connection.

The application also allows the patient to see charts about the data and easily see whether the values are good or not.

FIGURE 2: ANDROID APPLICATION

2 Working method

Here is the structure defined for the project.



This structure has been defined in order to establish an initial plan of the project to get an idea of the workload. It's just an overview of the tasks that can be changed. As with agile methods, some analysis could be made during construction part to modify or provide new specifications.

3 Requirements

Here is the final list of features implemented in the application. The first requirements have been modified after discussions with Michael Schumacher and Stefano Bromuri during the project.

3.1 Functional requirements

A user can:

- Login into the application by scanning a barcode
- Select symptoms in a list
- View the frequency of symptoms
- Enter the glucose
- View the glucose charts
- Enter the blood pressure
- View the blood pressure charts
- Enter the weight
- View the weight charts
- Change the date and time while entering values
- View content of local database in tables
- Send all unsaved data to server

The application must:

- Save all data entered by the user in the embedded database
- Encrypt all patient data to ensure security
- Synchronize the embedded database with the server when connection available

3.2 Non-functional requirements

Here is the list of implicit requirements implemented to ensure the good functioning of the application and a user-friendly interface.

- Design and model an embedded database
- Store the user certificate for authentication
- Improve charts
- Improve graphical user interface

4 Contribution

There was already an existing application but it was just a prototype to see the potential features of the application. Indeed, no feature was implemented, it was just the screens. This allowed seeing the navigation between screens and the potential of an Android application. The charts were static to get a glimpse and no data was stored locally or sent to the server.

FIGURE 3: OLD MENU

FIGURE 4: OLD INSERTING FORMS

The main goal of this project was to save all data entered by the patient on a server to make them accessible to the caretakers. For this, we had to implement the connection to the web service to send the data to the server. We also had to create an embedded database to store all data locally in order to be able to work offline and send them on the next connection.

In addition, we improved the charts displaying and the user interface such as menus or inserting forms in order to make the application more user-friendly. We also implemented an encryption system for all patient data and a connection by scanning a barcode.

5 Background

5.1 Used technologies

Android SDK

Before starting the implementation of the application, we had to choose a development environment. Different options were analyzed for developing a mobile application.

The application has to run on Android, so there were two options to choose: develop an Android application using the Android SDK or develop a web application offline in HTML 5 and CSS 3.

The advantage of the web application is that it can be easily compiled to run on different operating system like Android, iPhone and BlackBerry.

However, as it is not necessary that the application runs on another operating system that Android, we decided to develop an Android application.

The Android operating system runs on many brands of mobile that have different resolutions. It is therefore easier to develop an application adapted to different resolutions with the Android environment. In addition, an Android native application is more efficient than a web application.

FIGURE 5: ANDROID

The Android SDK also provides an easy way to have a multilingual application.

SQLite

A SQLite database is used to store patient data locally. The database is directly integrated into the program.

This is the easiest way to store and read data on an Android mobile.

FIGURE 6: SQLITE

Achartengine API

Two APIs are available for drawing charts: Google Chart API and Achartengine.

Unfortunately, the Google API that offers nicer designs requires an Internet connection. The application must also work offline, so we decided to use Achartengine.

FIGURE 7: GOOGLE CHART API

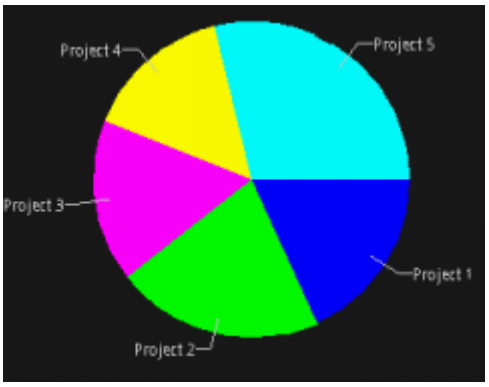


FIGURE 8: ACHARTENGINE

5.2 Used tools

Eclipse EE

Eclipse EE with the Android Development Tools (ADT) was chosen as development environment. The ADT plugin allows setting up new Android projects and creating applications.

FIGURE 9: ADT PLUGIN FOR ECLIPSE

Git

Git is a version control system. It provides a good file management system. It allows to merge different versions of files modified by users and also to restore the application to a previous running state if problems occur.

We used Git to manage application files and versioning.

FIGURE 10: GIT

Dropbox

Dropbox is a service for storing and sharing files online. This service is accessible via any web browser but also by using a multi-operating system client that allows you to use Dropbox as a directory of the computer.

We used Dropbox for sharing files such as documents or images.

FIGURE 11: DROPBOX

6 Installation

6.1 On virtual device

Here is the procedure to install the application on an Android Virtual Device. You must have previously installed the Android SDK and the Android ADT plugin for eclipse.

1. Import the project contained in the directory “AndroidClient” in eclipse
2. Right click on the project > “Run as”> “Android Application”
3. The emulator starts. Once it is launched, the application should be installed on the emulator and launch automatically.

If it does not work you must first start the emulator and then run the application.

1. “Window” > “Android SDK and AVD Manager”
2. Create an emulator if not already done
3. Click on "Start" then "Launch"
4. Wait until the emulator is started (home screen)
5. Right-click on the project in eclipse> “Run as”> “Android Application”.

6.2 On real device

Here is the procedure to install the application on an Android phone.

1. Open the folder “AndroidClient / bin”
2. Copy the file “AndroidClient.apk” on the mobile
3. Open the file with a file manager to install it.

7 Architecture

Here is a simple schema of the architecture established for understanding the communication of the Android application with the server.

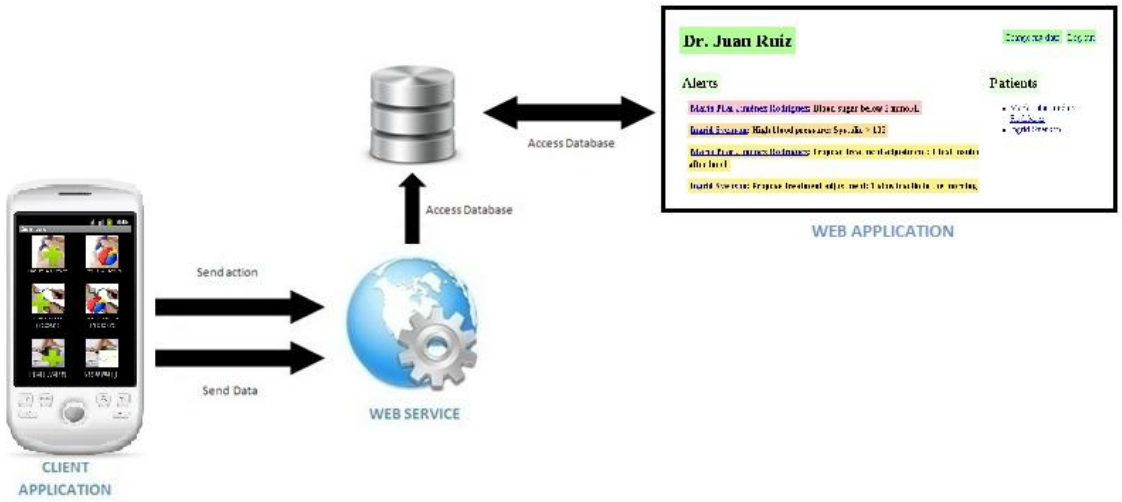


FIGURE 12: ARCHITECTURE

In order to save the data entered by the user on the database, we have implemented a web service with which the client application can communicate.

Once the data saved on the database, the caretakers will be able to view these data and will be alarmed about any risks through the web application.

7.1 Web service

The client application can only send data to the web service and will never receive. All data is stored locally, so it is not necessary to receive any data from the web service. This is also very important for safety issues. Indeed, if a hacker wants to attack the Web service, he will never be able to access the patient data available on the database.

Before saving data, a client has to log in with a username and a password. The client will then get a session id which will be used to authenticate for further requests.

This session id is valid only a limited time. Requests will fail if the session id timed out and return a value indicating that it is necessary to obtain a new session id.

Here is a list of all possible requests with their parameters and response:

Logging in	
Description:	This function allows the user to connect to the web service.
URL:	https://aislab2.hevs.ch/mobile/connect
Parameters:	- login: string - password: string
Response:	- if authentication failed: CONNECTION FAILED - if authentication was successful: the session id

Saving blood pressure	
Description:	This function allows the user to save the blood pressure by entering the different values.
URL:	https://aislab2.hevs.ch/mobile/insert/bloodpressure
Parameters:	- session_id: string - when: long (Unix Time Stamp indicating the observation time) - systolic: float - diastolic: float - pulse: float
Response:	- if session id is incorrect: PLEASE CONNECT - if parameters are missing or not of the correct type: PARAMETER ERROR - if request was successful: OK

Saving glucose	
Description:	This function allows the user to save the glucose by entering the different values.
URL:	https://aislab2.hevs.ch/mobile/insert/glucose
Parameters:	- session_id: string - when: long (Unix Time Stamp indicating the observation time) - glucose: float
Response:	- if session id is incorrect: PLEASE CONNECT - if parameters are missing or not of the correct type: PARAMETER ERROR - if request was successful: OK

Saving weight	
Description:	This function allows the user to save the weight by entering the different values.
URL:	https://aislab2.hevs.ch/mobile/insert/weight
Parameters:	- session_id: string - when: long (Unix Time Stamp indicating the observation time) - weight: float
Response:	- if session id is incorrect: PLEASE CONNECT - if parameters are missing or not of the correct type: PARAMETER ERROR - if request was successful: OK

Saving symptoms	
Description:	This function allows the user to save symptoms by entering the different values.
URL:	https://aislab2.hevs.ch/mobile/symptoms
Parameters:	- session_id: string - when: long (Unix Time Stamp indicating the observation time) - symptoms*: string, multiple values possible
Response:	- if session id is incorrect: PLEASE CONNECT - if parameters are missing or not of the correct type: PARAMETER ERROR - if request was successful: OK
* The only accepted values for symptoms are: epigastric_pain, dyspnea, blurred_vision, chest_pain, severe_headache, oedema.	

For now, the authentication to the web service is still done with a username and password. But that will change and there will be no need for username and password anymore. The 'connect' function and the 'session_id' parameter in the other requests will no longer be used. The user authentication will be done automatically using the certificate stored in the mobile.

8 Implementation

In this section, the implementation of the application will be described. All features may not be visible to the user but are necessary for the proper functioning of the application.

8.1 Local storage

The application has a local database containing all the values entered by the patient. This allows the user to view his data, but also to work offline.

All data must be transmitted to the server through the web service. If the mobile is not connected to the Internet when entering a value, the value is recorded as not sent. When the mobile is reconnected, all data will be sent.

Here is the diagram of the embedded database designed to store the patient values.

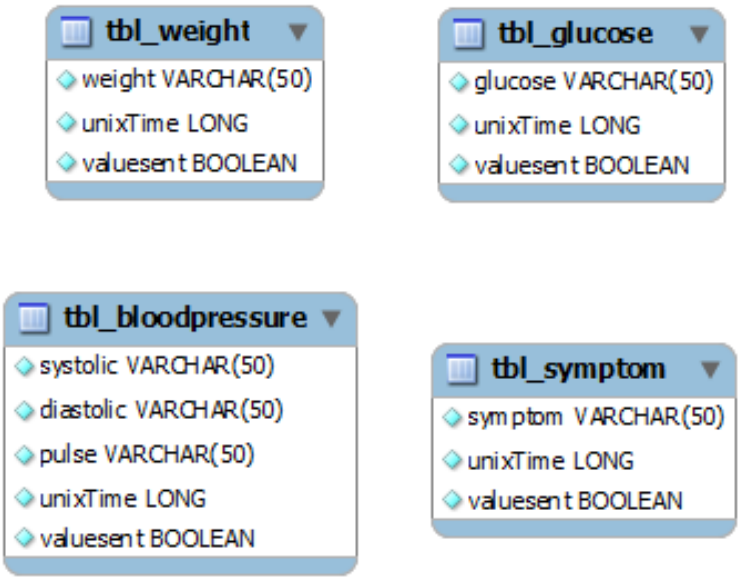


FIGURE 13: DATABASE MODEL

This is a simple database without any connection between tables.

All tables contain a field “valuesent” to see if the value has been sent to the server or not. The field “unixTime” is also present everywhere, it is a Unix Time Stamp indicating the observation time.

It is important to ensure the protection of patient data. All patient values, even numerical, are stored in text fields (VARCHAR) because they are in encrypted form.

8.2 Encryption

The Advanced Encryption Standard (AES) is a symmetric-key encryption standard. That means that there is only one key used for the encryption and decryption.

This encryption algorithm is used by the government of the United States. The encryption and decryption operations are very fast and do not require much memory (RAM). The application will not be slowed because of the encryption.

For more security, we add some random values to the input before storing it in the database. During the decryption, we will delete the random part added.

Take for example the value 4.3 for glucose. Before recording this value in the database, we will add random decimals to get a value like 4.31864. During the decryption of this value, we will take only the number with one decimal.

This way, even if we save several times the number 4.3, it will never be the same encrypted value.

```
4.3  ==>  74E6E9F73EA5106C152A8CEB7E69B0E1  ==>  4.3
4.3  ==>  0E0C673B23DC44B2F92A37B4787BB03D  ==>  4.3
4.3  ==>  12E371D7455CB705A5114F9A3FC8903F  ==>  4.3
4.3  ==>  79B6613B137F402D25A7426326C18A6F  ==>  4.3
4.3  ==>  8D3CBB0C06591EEAE6937D0C5E718296  ==>  4.3
```

FIGURE 14: NUMERIC ENCRYPTION

The text data will be encrypted the same manner. Instead of random decimals, we will add a special character followed by random characters. The special character will allow us to recognize the part to remove during the decryption. For example, the symptom "headache" will be encrypted as "headache/oWslP".

```
headache  ==>  8580D2EC26BE92321F32CDFEE99CF8E1  ==>  headache
headache  ==>  FC084053126AE46E2E5DC1590D210EF3  ==>  headache
headache  ==>  51CAA6A0BE7D3F1CE00BA4D37AAF3B77  ==>  headache
headache  ==>  676647468854FB0415E7DA9316C2851B  ==>  headache
headache  ==>  B78B651E051CDCAD8386498AED7B7B2B  ==>  headache
```

FIGURE 15: TEXT ENCRYPTION

This prevents that a hacker decrypts the patient data. Since the encrypted value is never the same, it is impossible to compare two of them to determine the actual value.

8.3 Login username / password

The authentication by username / password is no longer part of the application. But we are still going to describe it quickly because it was a feature expected at the beginning that has been developed.

The first connection with a new user must be done online to verify the information with the server. Then the Internet connection was no longer required because the username and password were stored in the database of the mobile. There was an automatic login function, thus avoiding the user having to enter his information each time.

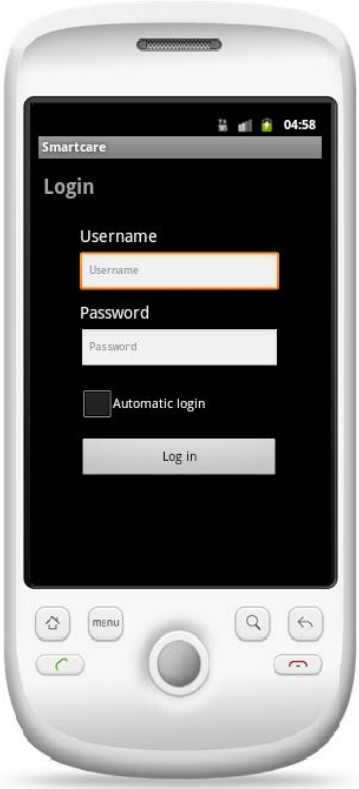


FIGURE 16: USERNAME / PASSWORD LOGIN

User switching was done automatically, without having to manually delete the old data. When connecting of a new user, a confirmation message was displayed before deleting the old user data. This would avoid losing data accidentally.

The problem with this method is the protection of the identity and password of the patient. The username and password being saved in the database to allow the connection offline were easily accessible for a hacker.

This authentication method was changed for a certificate authentication. The username of the patient will no longer be present on the mobile, only his certificate.

For now a username and a password are hard coded in the application because the web service was not modified.

8.4 Certificate authentication

The user authentication is now done through the certificate on the mobile. It's safer to protect the identity of the patient because there is no more registered username.

The certificate is stored on the mobile as a bks file. It is located in the res/raw directory. Each user will have an associated certificate. The certificate is protected by a password that will be needed to make requests to the server.

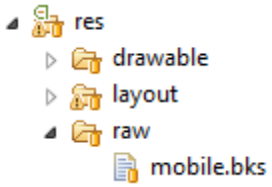


FIGURE 17: USER CERTIFICATE

When switching user, the certificate must be replaced by the new user's keeping the same name "mobile.bks" and the application recompiled.

When the application is reinstalled on the mobile, the old data will be automatically removed.

8.5 QR Code

The QR code (QR for Quick Response) is a type of barcode in two dimensions. We chose this type of barcode because its content can be decoded quickly. The barcode is used to allow the user to log into the application.

FIGURE 18: QR CODE

The generated QR Code contains the password for the certificate and an encryption key. When the barcode is scanned, we check if the password matches the certificate stored in the mobile.

Once authenticated, we store temporarily the password to make future requests to the server. We also store the encryption key that will allow us to encrypt/decrypt the values in the database.

The password and encryption key are stored only the execution time of the application, they will be deleted when we leave the application.

8.6 Barcode scanner

To read the barcodes, we decided to use the open source barcode scanner of Google.

FIGURE 19: BARCODE SCANNER

Other possibilities were examined, including use of libraries for decoding the images of barcodes. These options were not selected because we had to manage ourselves taking pictures with the camera of the mobile and send them for processing. The decoding would probably have been slower because it is hard to define the frequency of sending picture or whether the view angle has changed to send another image. Here we just call the Google application to scan the barcode and get the result in our application.

Using the Google scanner is faster and we can expect to benefit from any updates from Google. As the barcode scanner reads all the barcodes, it avoids having to change our application if we decide to change the type of the barcode.

Also, if the Google barcode scanner is not installed on the mobile, an alert window invites us to download it directly on the Android market.

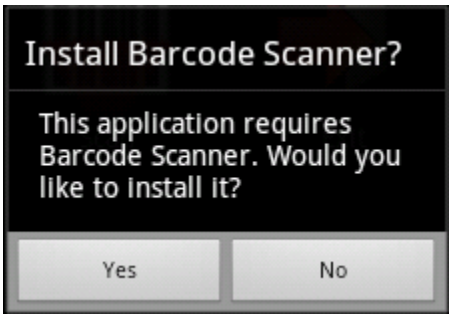


FIGURE 20: INSTALL BARCODE SCANNER

8.7 Multilingual application

Android allows you to easily develop multilingual applications. We only need to create an xml file containing variables of all the texts of the application in the default language.

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    [...]
    <string name="insert_glucose">Insert Glucose</string>
    <string name="show_glucose">Show Glucose</string>
    <string name="insert_blood_pressure">Insert Blood Pressure</string>
    <string name="show_blood_pressure">Show Blood Pressure</string>
    <string name="insert_weight">Insert Weight</string>
    <string name="show_weight">Show Weight</string>
    <string name="insert_symptoms">Insert Symptoms</string>
    <string name="show_symptoms">Show Symptoms</string>
    <string name="table">Tables of values</string>
    <string name="about">About</string>
    <string name="logout">Log out</string>
    [...]
</resources>
```

FIGURE 21: ENGLISH VALUES

We can then translate this file in the languages desired.

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    [...]
    <string name="insert_glucose">Insérer glucose</string>
    <string name="show_glucose">Voir glucose</string>
    <string name="insert_blood_pressure">Insérer pression sanguine</string>
    <string name="show_blood_pressure">Voir pression sanguine</string>
    <string name="insert_weight">Insérer poids</string>
    <string name="show_weight">Voir poids</string>
    <string name="insert_symptoms">Insérer symptômes</string>
    <string name="show_symptoms">Voir symptômes</string>
    <string name="table">Tables des valeurs</string>
    <string name="about">A propos</string>
    <string name="logout">Se déconnecter</string>
    [...]
</resources>
```

FIGURE 22: FRENCH VALUES

The language files will be loaded according to the language settings of the phone. If the application has not been translated into the language of the mobile, it will be displayed in default language. The variables not translated will also be displayed in the default language.

To add a new language to the application, simply translate the file "strings.xml" and place it in the directory of the corresponding language.

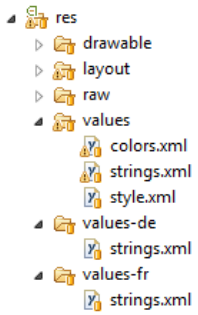


FIGURE 23: LANGUAGE DIRECTORIES

9 User interface

In this section, we describe all the things done for the user interface to be more user-friendly and easy to understand.

9.1 Menus

The menus are composed of images and texts. The images are representative of the text. The button layout of the main menu was chosen to be nice and logical. For each line, there is the insert function and the viewing function for a value.

FIGURE 25: SCANNER MENU

FIGURE 24: MAIN MENU

The images were chosen to be easily understandable. That way, even if the patient does not understand the language or cannot read the text because it is too small, he understands what the use of the button is.

9.2 Forms of values insertion

The forms of values insertion are simple and easy to use. The font size is sufficiently large to be easily readable.

FIGURE 27: INSERT SYMPTOMS

FIGURE 26: INSERT BLOOD PRESSURE

By default, the time of insertion is defined as the current time but it is possible to change it to an earlier time. This allows the user to save all observed values even if he did not have his mobile on him.

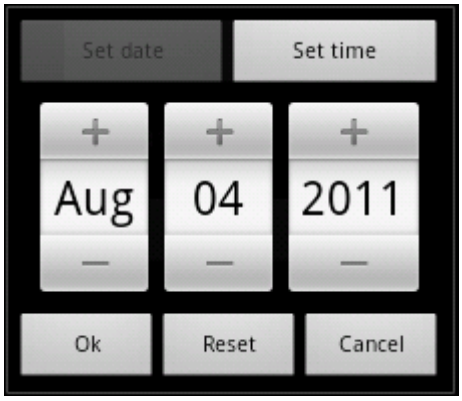


FIGURE 29: SET DATE

FIGURE 28: SET TIME

The data are validated before being saved to avoid empty fields and typing errors, such as entering a weight of 800 kg instead of 80 kg or glucose of 53 instead of 5.3. The data are verified with minimum and maximum values. A message will be displayed to warn the user.

9.3 Charts

For blood pressure and glucose, we decided to draw pie charts. The values are separated into three categories: too low, normal, too high. It is easily understandable by everyone by having a look.

For the weight, we cannot determine if the values are too low or too high, so we draw a line chart. This chart allows seeing weight changes week by week. The axis of weight is from 50 to 120 by default, but it will automatically be adjusted if there are lower or higher values. We can zoom in and out and scroll horizontally and vertically to see the values in more detail.

We tried to show the frequency of symptoms in a bar chart but it was unreadable because the names of symptoms are too long. Instead, we decided to simply write a list of symptoms with their frequency if it is greater than zero.

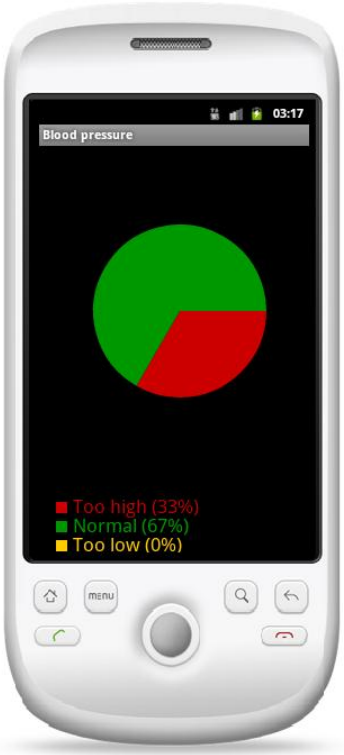


FIGURE 32: SHOW BLOOD PRESSURE

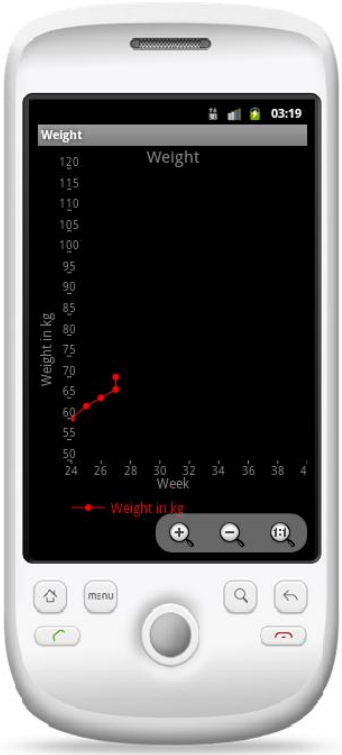


FIGURE 30: SHOW WEIGHT

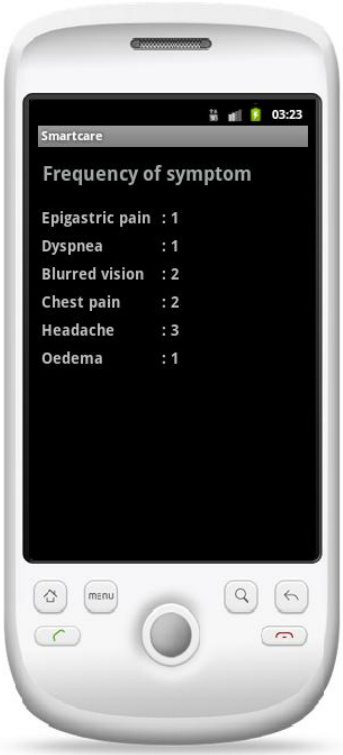


FIGURE 31: SHOW SYMPTOMS

9.4 Tables of values

We have created tables containing all the values of the local database. This allows the user to see the values in more detail than in the charts. All available information on the value are displayed, date of insertion and also if it has been sent to the server or not.

The values are in tables that can not only scroll vertically but also horizontally. The column size is not fixed because there are many different screen resolutions for Android phones and with horizontal scrolling we can display data in full.

If necessary, we can hold the phone in landscape mode in order to better view the information.

FIGURE 33: TABLE WEIGHT

FIGURE 34: TABLE BLOOD PRESSURE

The tables can be sorted by all columns. To do this, simply click on the title of the column we want to sort.

A small arrow is displayed next to the title of the column that the sorting is done. The sorting can be done in ascending or descending order, the direction of the arrow indicates the sorting order.

9.5 Overview of unsent data

A screen containing an overview of the number of unsent values was established.

It allows the user to quickly see how many values were not sent and to send them simply by clicking on the button.

If all data are sent, there will be a message saying there is no data to send instead.

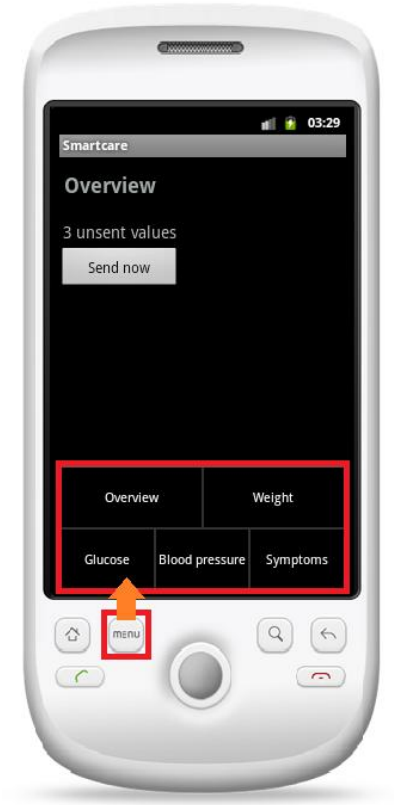


FIGURE 35: OVERVIEW UNSENT VALUES

The options menu that allows viewing the different tables of values is automatically displayed when we arrive on this screen. Otherwise, it is possible that a user will never know that an options menu exists. The user can then click the menu button to switch between different tables and the overview.

9.6 Help the user

We know that some patients are not familiar with using an Android phone, or even a smartphone. After viewing a chart, for example, a user could click twice on the back button to return to the menu and exit the application by mistake. He should then scan the barcode again to restart the application. This can be annoying quickly.

So, to help users and prevent them from leaving the application in error, we display a confirmation window before closing the application.

FIGURE 36: EXIT CONFIRMATION

Other measures have been taken to help the user, such as displaying messages to explain what is happening or progress dialogs while sending data to server.

FIGURE 37: MESSAGES

10 Possible improvements

10.1 Change data on charts

It would have been nice to be able to select the data on which to display the charts. For example, let the user choose if he wants to see the data from the last day / last week / last month or all data. This would allow the patient to quickly see the evolution of values, for example to see if the glucose level is stabilizing.

This feature has not been developed because the API Achartengine does not allow adding buttons or options menu on the charts. Achartengine takes as parameters the data to be processed and returns the screen to display.

10.2 Possibility to change values

Another option that might be interesting is to allow the user to change a value. For example, if he made a typing error when entering or wishes to change the date, he may modify or delete the value.

To do this, we should give an identifier to each value and change the web service.

10.3 Automatic change of user

It would be nice not having to change the certificate manually and having to recompile the application on each user switching. For example, we might have a function on the application where we should enter the new username and it would install the appropriate certificate and will also delete the data from the old user after confirmation.

After discussion with Johannes Krampf, which handles the web service, we have decided not to develop this feature because at the moment there are only a few patients so the operation should only be done a dozen times.

This feature will be necessary if we deploy the application on the Android market or if we have many patients.

11 Problems encountered

11.1 Tables of values

I had some problems for displaying the tables. I wanted to fix the header containing the title so it does not scroll vertically with the rest of the table but I also wanted the table to scroll horizontally in order to display all data.

Unfortunately I was unable to combine these two functions so I chose to keep the horizontal scrolling that I find more important considering the many different resolutions of Android phones.

11.2 Barcode scanner

It took me a long time to include the barcode reader because I was trying to change it to get a standalone version with the minimum of features, but it was too complex. I was not able to include it in the application. I finally found a way to call the application of Google and offer to install it if it is not.

I then had another problem because the barcode scanner keeps a history of all scans by default. Fortunately there is a way to use the application by specifying not to save the scan.

11.3 Writing in English

The hardest part was writing this report in English because I have not a much evolved vocabulary. This is the first time I write something in English and I did not think it would be so difficult.

However, this was an interesting and rewarding experience that I do not regret.

12 Project management

The Bachelor's thesis began 16 May 2011 after the delivery of topics and ends 16 August 2011, when the work must be delivered. During this time, 360 hours must be made, representing eight weeks of full-time job. Each student is free to allocate his working hours as he wish.

An initial schedule was made to get an idea of the workload to achieve but it was not fixed. Some features could be modified, deleted or added during the project following a discussion with Michael Schumacher and Stefano Bromuri. We held weekly meetings to discuss project progress and features to be developed.

12.1 Planning

Here is a comparison between the time originally planned and the time actually done.

As we can see, the part for the caretakers has been abandoned during the project because it was no longer necessary. Instead, we developed a connection by scanning a barcode and a system for data encryption.

Bachelor plan	**40 jours**
Analysis	**7 jours**
Reunions and tools installation	1 jour
Understand current application	1 jour
Analysis & Writing requirements, Define functions to develop	4 jours
Design DB and architecture for synchronization with server	1 jour
Development	**23 jours**
Implement synchronization with server	**8 jours**
Database conception & local storage	3 jours
Test web service communication	1 jour
Login and connection to server	2 jours
Upload data to server	2 jours
Charts improvement	**8 jours**
Learn charts design	2 jours
Modify existing charts	3 jours
New charts (recap sent data)	3 jours
Caretakers part	**4 jours**
Download data from server	1 jour
Screens and charts for caretakers	3 jours
Ergonomy improvement	**3 jours**
Improve GUI, fields verification, notification, confirmation...	3 jours
Final report	**10 jours**
Write final report	10 jours

FIGURE 38: TIME PLANNED

Bachelor plan	**40 jours**
Analysis	**6 jours**
Reunions and tools installation	1 jour
Understand current application	1 jour
Analysis & Writing requirements, Define functions to develop	3 jours
Design DB and architecture for synchronization with server	1 jour
Development	**24 jours**
Implement synchronization with server	**6 jours**
Database conception & local storage	3 jours
Login and connection to server	2 jours
Upload data to server	1 jour
Charts improvement	**5 jours**
Learn charts design	0.5 jour
Modify existing charts	0.5 jour
Tables of values	4 jours
Start caretakers part (abandonned)	2 jours
Login by scanning QR Code	**4 jours**
Include QR Code reader	3 jours
Certificate authentication	1 jour
Encryption of data	**3 jours**
Implement encryption/decryption	3 jours
Ergonomy improvement	**4 jours**
Improve GUI, fields verification, notification, confirmation...	4 jours
Final report	**10 jours**
Write final report	10 jours

FIGURE 39: TIME PERFORMED

All features defined have been implemented. We even added some features that were not initially specified in the requirements list.

13 Conclusion

13.1 Android application

All the features described in the specifications were done respecting the constraints. We implemented the communication with the web service to save the patient values to the server. We created a local database containing all data in order to be able to work offline and send them on the next connection. We improved the charts displaying and the user interface such as menus or inserting forms in order to make the application more user-friendly. We also implemented an encryption system for all patient data and a connection by scanning a barcode to authenticate the user to the web service with his certificate.

The application was tested with three people, one using an Android phone daily, one using another smartphone and one that never used smartphones. Following their remarks and problems, the application has been modified and is now ready to be used by patients.

Of course, it would still be necessary to have a feedback of the first uses to fully meet the expectations of users.

13.2 Personal perspective

Through this project, I had the opportunity to deepen my knowledge in the Android development which is an important asset, as the demand for this platform is increasing. This experience is very rewarding and could be important for my future career.

14 Sources

14.1 Bibliography

Mark Murphy. *L'art du développement Android*. Pearson, 2009.

14.2 Webliography

Analysis

http://www.phonegap.com/home (23.05.2011)

http://www.html5-css3.fr/html5/tutoriel-application-web-offline-html5-cache-manifest (23.05.2011)

http://www.whatwg.org/specs/web-apps/current-work/multipage/offline.html (24.05.2011)

http://developer.android.com/guide/index.html (24.05.2011)

Android development

http://www.tutomobile.fr/comment-utiliser-sqlite-sous-android-tutoriel-android-n%C2%B019/19/10/2010/ (24.05.2011)

http://huuah.com/using-tablelayout-on-android/ (14.06.2011)

http://sdroid.blogspot.com/2011/01/fixed-header-in-tablelayout.html (14.06.2011)

http://www.coderanch.com/t/384000/java/java/Convert-Unix-TimeStamp-Date-Format (22.06.2011)

http://leepoint.net/notes-java/other/10time/30calendar.html (23.06.2011)

http://blog.xebia.fr/2010/10/21/comment-integrer-des-graphiques-dans-une-application-android/ (04.07.2011)

http://www.achartengine.org/ (04.07.2011)

http://developer.android.com/guide/topics/resources/localization.html (07.07.2011)

http://linuxfr.org/forums/programmationjava/posts/parser-une-string-en-xml (11.07.2011)

http://www.java-tips.org/java-se-tips/javax.xml.parsers/how-to-read-xml-file-in-java.html (11.07.2011)

http://www.javabeat.net/tips/182-how-to-query-xml-using-xpath.html (11.07.2011)

http://www.androidsnippets.com/encryptdecrypt-strings (13.07.2011)

http://code.google.com/p/zxing/ (14.07.2011)

http://www.helloandroid.com/tutorials/store-imagesfiles-database (18.07.2011)

http://oudomvilla.wordpress.com/2010/08/23/create-icon-with-text-using-gridview-and-layout-inflater/ (19.07.2011)

http://developer.android.com/guide/topics/ui/dialogs.html (21.07.2011)

http://www.brighthub.com/mobile/google-android/articles/82805.aspx (21.07.2011)

http://stackoverflow.com/questions/693997/how-to-set-httpresponse-timeout-for-android-in-java (21.07.2011)

http://java.developpez.com/faq/java/?page=langage_donnees#trierCollection (22.07.2011)

http://developer.android.com/guide/market/publishing/multiple-apks.html (22.07.2011)

http://developer.android.com/resources/tutorials/testing/helloandroid_test.html (22.07.2011)

http://mobile.tutsplus.com/tutorials/android/android-sdk-junit-testing/ (22.07.2011)

http://developer.android.com/resources/tutorials/views/hello-timepicker.html (25.07.2011)

http://developer.android.com/resources/tutorials/views/hello-datepicker.html (25.07.2011)

http://fr.wikipedia.org/wiki/ISO_9126 (26.07.2011)

Report writing

http://en.wikipedia.org/wiki/Gestational_diabetes (08.08.2011)

15 Annexes

- Initial requirements
- Weekly hours
- CD Rom containing:
    - The Android Project
    - The source code documentation
    - This report

Requirements

Contents

1 Introduction

This document aims to present the different requirements of the G-DEMANDE project.

1.1 Context

This project is realized for my Bachelor thesis. There is already a demo version of the application G-DEMANDE that we must improve to make it usable for end users.

1.2 Existing application

The existing application allows the user to enter various measures (glucose, blood pressure, weight) and display charts concerning them.

1.3 Goals

For now, the data entered by the user are stored only in memory of the current session.

The main goal of this project is to transfer this data on a server. The application must also have a local database to store the data on the mobile so that when we do not have internet access we can transfer the data to the server later or at the next login.

We also need to retrieve this data to show to the caretakers the monitored values.

We will improve the charts displaying to be easily understandable by the user and maybe change the graphical user interface to be more user-friendly.

At the end of the project, the goal is to have a functional application that can be deployed. We will also ensure that the application can easily be updated.

1.4 Author contact

Ivan Samardziev
Route de la plaine 10
3960 Sierre
Tel. +41 79 964 20 70
Email prof. ivan.samardziev@students.hevs.ch

2 Architecture

The client can send requests and data to save to the server through web service that will contact the database and save the data. It may also request data to show to the caretakers the monitored values.

3 Requirements

Here we will list all requirements, whether functional or not.

3.1 Functional requirements

Here is the list of user functions. Some of them must be created, others must be improved.

A user can:

- Login into the application
- Select symptoms in a list
- Enter the glucose
- View the glucose charts
- Enter the blood pressure
- View the blood pressure charts
- Enter the weight
- View the weight charts
- View his current location

A caretaker can:

- View the alerts (data too low or too high for a patient)
- View the list of patients
- View the different charts for each patient

The application must:

- Save all data entered by the user in the embedded database
- Synchronize the embedded database with the server when connection available

This list will not be changed in this document but may well be extended during the project by Ivan Samardziev, Michael Schumacher and Stefano Bromuri. We will decide the importance of each function and the time to develop it together at meetings.

3.2 Non-functional requirements:

Here is the list of implicit requirements that must be implemented to ensure the good functioning of the application and a user-friendly interface.

- Design and model an embedded database
- Improve charts
- Improve graphical user interface

4 Working method

4.1 Analysis

In this step we will study the various requirements and constraints and on this basis, we will define a structure for the application.

4.2 Development

In this phase, we will develop the application according to the requirements and constraints analysis.

We will use Java to make this application.

4.3 Working place

This project will be conducted in the premises of Techno Ark for Monday, August 15, 2011.

Project meetings with Michael Schumacher and Stefano Bromuri will be planned during the project.

Ivan Samardziev

Bachelor thesis

Filière informatique de gestion/
Studiengang Wirtschaftsinformatik

Agent-based Diabetes Monitoring with Android Mobile

Ivan Samardziev

Week : *20*

Weekly hours

Task	Hours
Project presentation and meetings for starting project	4
Installing tools	6
Analyzing & writing requirements	6
Reading & understanding existing code	4
Modifying code for better readability (indentation & comments)	4

Problems - Solutions Found

Old code not commented, no indentation – difficult to understand	Modification of code, comment, refactoring

Bachelor thesis

Filière informatique de gestion/
Studiengang Wirtschaftsinformatik

Agent-based Diabetes Monitoring with Android Mobile

Ivan Samardziev

Week : *21*

Weekly hours

Task	Hours
Meeting, discussion about an iPhone version	2
Research HTML 5 / CSS 3 / web application offline	4
Research Phonegap / local storage	4
Compare the pros and cons, choosing for an android application	4
Documentation about SQLite (embedded database)	4

Bachelor thesis

Filière informatique de gestion/
Studiengang Wirtschaftsinformatik

Agent-based Diabetes Monitoring with Android Mobile

Ivan Samardziev

Week : *22*

Weekly hours

Task	Hours
Creation database, tables and methods for access (insert, get…)	10
Modification of the code for saving/reading data in/from local database	10
Meeting	1

Bachelor thesis

Filière informatique de gestion/
Studiengang Wirtschaftsinformatik

Agent-based Diabetes Monitoring with Android Mobile

Ivan Samardziev

Week : *23*

Weekly hours

Task	Hours
Start page with login	4
Change layout (relative, scroll), use of styles, fields validation (check numbers)	10
Meeting	1

Bachelor thesis

Filière informatique de gestion/
Studiengang Wirtschaftsinformatik

Agent-based Diabetes Monitoring with Android Mobile

Ivan Samardziev

Week : *24*

Weekly hours

Task	Hours
Meeting	1
Saving all data (symptoms added) in local database	5
Learning gridView for database content (abandoned)	4
Learning tableLayout for database content	3

Bachelor thesis

Filière informatique de gestion/
Studiengang Wirtschaftsinformatik

Agent-based Diabetes Monitoring with Android Mobile

Ivan Samardziev

Week : *25*

Weekly hours

Task	Hours
Save unsent data	8
Connection + auto login	5
Verification change of user + confirmation + delete old data	2
Simple table (tableLayout) with database content, vertical and horizontal scroll	4
Convert unix timestamp to date	6

Diverse

Problems converting the unix timestamp to date because of java.util.Date class (functions to convert gives wrong result), finally used the java.util.Calendar class

Bachelor thesis

Filière informatique de gestion/
Studiengang Wirtschaftsinformatik

Agent-based Diabetes Monitoring with Android Mobile

Ivan Samardziev

Week : *27*

Weekly hours

Task	Hours
Choosing & learning chart engine	3
Drawing pie charts for glucose & blood pressure	4
Database content overview	3
Database tables : improve display, try to fix header, try to set clickable header	8
Meeting	1
Change login & saving data in online/offline mode	3
Database tables : sort data on header click, inverse if click again on same	8
Write percentage in pie chart	2
Search for multilingual application	2
Prepare application structure for multilingual	8
Translate the application in French and German	2

Problems - Solutions Found

Tables – impossible to combine fix header (that don't scroll vertically) with horizontal scroll	Choose horizontal scroll (important because different screen resolution) - To fix header need fix column size, can't fix because unknown resolution

Bachelor thesis

Filière informatique de gestion/
Studiengang Wirtschaftsinformatik

Agent-based Diabetes Monitoring with Android Mobile

Ivan Samardziev

Week : *28*

Weekly hours

Task	Hours
Define screens for doctor interface	1
Store/send http cookie, get it on authentication then send it for others requests	10
Web service return html, parse the html and retrieve the desired data	8
Meeting (doctor part abandoned) – Authentication with certificate + QR Code	2
Improve tables of values (display arrow up and down to show current sort)	3
Change weight chart display	2
Check if there is data to display, if not show a message	2
Learn AES encryption / decryption	2
Include a QR Code reader in the application with test on real device	20
Implementation of encryption / decryption system	7

Diverse

It took me a long time to include the bar code reader because I was trying to change it to get a standalone version with the minimum of features, but it was too complex. I finally found a way to call the application of Google and offer to install it if it is not.

Bachelor thesis

Filière informatique de gestion/
Studiengang Wirtschaftsinformatik

Agent-based Diabetes Monitoring with Android Mobile

Ivan Samardziev

Week : *29*

Weekly hours

Task	Hours
Store certificate file on phone, read the certificate and check password	6
Improve GUI : Change main menu (list of actions), gridview with image and text	10
Test on real device, modification text & image size	4
Change login menu (home page with scanner), gridview with image and text	2
Create symptom chart to show frequency (Bar chart), to be modified, not readable	3
Try to add option on charts to have last day/week/month/all data (abandoned)	2
Display a waiting dialog while saving data on server	6
Set timeout to prevent the application to crash if server not responding or too long	3
Change symptoms charts by table with symptom name : frequency	2
Change sort methods in tables because encrypted data in database	4
Add pulse data in blood pressure	3
Change tables display (add images for true/false…)	2
Test on real device	2
Look for future improvements (multiple APK installation for different mobile android version and different screen resolution, not yet implemented by Google)	2
Look for unit test (not implemented because method to save data don't return anything)	2

Diverse

Try to let the user choose on which data to display the charts (day/week/month/all) but impossible because using a chart API that return the screen to be displayed, cannot add an options menu or buttons to get chosen data

Bachelor thesis

Filière informatique de gestion/
Studiengang Wirtschaftsinformatik

Agent-based Diabetes Monitoring with Android Mobile

Ivan Samardziev

Week : *30*

Weekly hours

Task	Hours
Meeting	1
Change the time when inserting values with validation (no future or before 9 month)	6
Confirmation to exit the application (avoid exit by error on back pressed)	2
Validation of the data inserted (minimum and maximum values)	2
Weight chart modification – adapt the axes to minimum and maximum values	1
Final test	2

Bachelor thesis

Filière informatique de gestion/
Studiengang Wirtschaftsinformatik

Agent-based Diabetes Monitoring with Android Mobile

Ivan Samardziev

Week : *31*

Weekly hours

Task	Hours
Defines structure for the final report and writes some notes	10
Print screens of the application, draw schemas, description of schema and web service	20

Bachelor thesis

Filière informatique de gestion/
Studiengang Wirtschaftsinformatik

Agent-based Diabetes Monitoring with Android Mobile

Ivan Samardziev

Week : *32*

Weekly hours

Task	Hours
Writing report, reading, corrections	50

Diverse

It's really hard to write in English and it takes much more time than expected.