Project Report - GPS Tracker

Pascal Bruegger
Msc Course - Ubiquitous Computing
University of Fribourg, Switzerland

March 9, 2006

Contents

List of Figures

Note of the author: Before starting the study of this report the reader must have a good understanding of the J2EE and EJB technologies. The concept of the project is presented without explicit specifications of EJB 2.1. If any doubt on J2EE technology, refer to [9, 2].

1 Backgrounds

1.1 Motivations and Goals

Security is very important in some activities. Freeride, mountain walking or climbing, paragliding are those where accidents can be serious or fatal. Having the possibility to follow physically the position of a person on regular basis can be comfortable for family, relatives or others.
The project is meant to propose *a simple and portable solution for people to get traced during a trip*. The application is *web based* and should be available for every people who have the possibility to

1. Run a small Java application on its mobile phone.
2. Has link between a GPS device and its mobile phone.
3. Has Internet access.

This concept is not new and a lot of applications involving GPS are available on the market: nowadays almost every new car is equipped with a GPS on board and help people in city or country side to find their road.
The concept of this project is a bit different. We are not focusing on *"where are we?"* but more on *"where he/she is?"*. For this the idea is to use a cell phone which accepts to receive GPS coordinates and send them to a server able to record them under the account of a register user (tracked person). Then from a web client, the user can be followed on a map in real time mode.

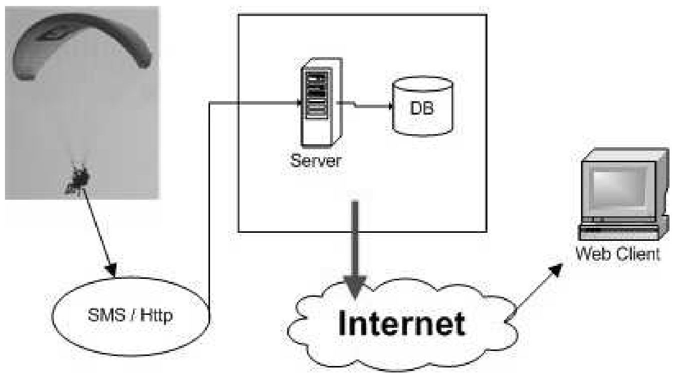


Figure 1: GPS traker: General schema

1.2 State of the art

As mentioned above, there are already a lot of applications which use the geo-positioning. Garmin, one of the famous GPS manufacturer, proposes maps of almost every countries in the world ready to be download into their panel of GPS devices. Different kind of applications for different kind of public: road maps and tracking, topologic maps for technical job like geologist (for instance), flight's map for pilots, etc.
Also it exist, for mountain activities like freeride, hicking, those automatic signalling systems which switch on as soon as the rider get cought by an avalanche for example. It transmits a radio signal to the closest relay and indicate the exact position of the victim. It helps for the search and often save lives. The portable TomTom GPS proposes a full navigation system with vocal indication. The list of geo-positioning applications is huge and a simple search on the Internet gives hundred web sites talking about the topic.

1.3 Use Cases

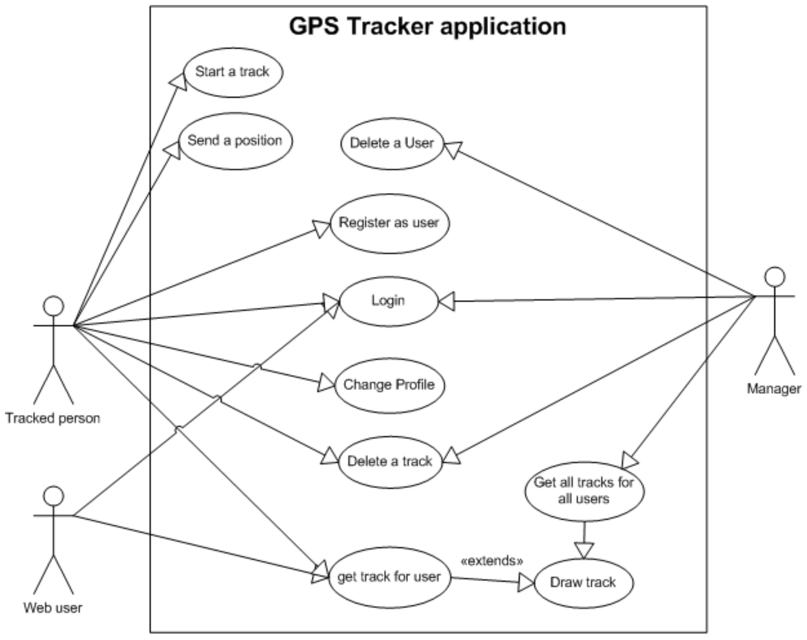


Figure 2: Use case of the application

The diagram (figure 2) shows the use cases for the application. There are 3 actors, 10 use cases which represent for different action what actors can do with the system. The use case *Start a track* is in fact 2 use cases. A track can be started from the web client or from the mobile application (cell phone application). The use case *Send a position* is only available from the mobile application. We will see in the architecture what is done where (mobile vs server application).

2 Functional Description

2.1 Requirements

The application will follow the use cases described in figure 2. All use cases are now described formally.
Within the 3 actors, the manager has a special role and is automatically created with the user Id 1. It can do exactly what a standard user can do but has special rights that the others don't have.
The application will allow to do:

Creation of users. Any user who wants to be followed needs to be register in the system first. The application will offer a web interface to enter the following data:

1. First Name
2. Last Name
3. Street
4. Zip code
5. City
6. Phone
7. Password

Then the user receives a user Id (integer) and is recorded in the database. This part is done on the web client side. The password must be more than 4 characters long. Any fields must be empty otherwise the user is not register.

Modification of the user profile. Once logged, the user can modify its profile using the same data as above and same constraints.

User login This web page is the first one in the system. The user must be identify in order to access any options. The web page contains two fields:

1. User Id
2. Password

This page proposes the link to the registration web page in case the user is not yet register.

Creation of a track for a given user. The user, once registered, can log in the system and create a track. There is two possibilities to do it:

1. From the web client application.
2. From the mobile application (mobile phone).

If a new track is started from the mobile application, the user must provide:

1. User Id
2. Password

Then the server will create a new track and be ready to store the future positions in under this track's id.
For a track started from the web client, the user can (but it is not mandatory) give:

1. North coordinates
2. East coordinates

This will represent the first point of the track. The track has a unique identifier within the system. The time and date when the track is created is taken from the server's clock.

Sending GPS coordinates to a server using mobile data communication. The user get the coordinates (position) from its GPS device and send them to the server with mobile phone. The position must contain:

1. User Id
2. Time
3. North coordinates
4. East coordinates
5. Altitude

Then the server record the position under the user's current track id. If no track is created or the user Id does not exists, the system gives an error message and do not record the position.

Follow a user in real time mode on the Internet. Any person having the right to watch a user's track must log in the system under the user's ID and password. Then he/she can choose the track to be drawn. A list of the user's tracks is available. This is done through the web page. The drawings of the track are done in two different manners: Swiss Topo 2D maps and Google Earth. If the track chosen is the current one and drawn in 2D mode, it is refreshed periodically using a "refresh time" parameter.

Delete a track. The user can delete tracks from a list of tracks. If the track deleted is the current one then the previous one become again the current track and all the future positions sent to the server will be save under this track id.

Manager - Delete users. The web page allow the manager to delete users from a list of register one. This list shows all the available users.

Manager - Follow a tracked user in real time mode on the Internet. The manager can choose the track to be drawn. A list of all user's tracks is available. This is done through the web page. The other options are the same as standard users.

3 Operational Instructions

The architecture of project is based on Entreprise JavaBeans technology. The mobile application is developed in Java 2 Micro Edition.

3.1 System Requirements

The system needs different hardware and software components:

- An Application Server J2EE EJB 2.1 compatible.
- A SQL database
- Java compatible mobile phone with Bluetooth
- Internet access

3.2 Software

The OS where all the following were installed is Windows XP sp2:

1. J2EE Application Server: Sun Java System Application Server Platform Edition 8.1 2005Q1
2. Entreprise Java Beans 2.1
3. Java Midlet : SUN J2ME Wireless Tool Kit 2.2
4. PointBase free Version: 5.2 ECF build 294
5. Test Browsers : Fire Fox 1.0.7, Internet Explorer
6. Java VM : J2SE Version 1.5.0 (build 1.5.0 06 b05)

In addition, we have used Google Earth (v3.0) Release Notes - November, 2005 (v3.0.07xx). The Development platform is Eclipse Version: 3.1.1. To built and deploy the application Xdoclet and Ant script file are used.

3.3 Hardware

Only two devices where needed and provided by the PAI group:

1. Mobile Phone Nokia
2. GPS EMTAC Bluetooth - GPS Trine

These two hardware were used by Otto Poveda for his part of the development (communication between GPS and mobile Phone).

3.4 Architecture

The technologies used for the project are client-server and web based. The main platform of development is J2EE. The full description of the language, server version is describe above. We will not explain in detail how J2EE works and assume that the reader knows this technology and how the modules are defined. The general structure of the application is shown in the figure 3.
The application is based on 3-tiers architecture:

1. Client side: mobile application, web browser
2. Server side: business logic, web application (interface)
3. Database

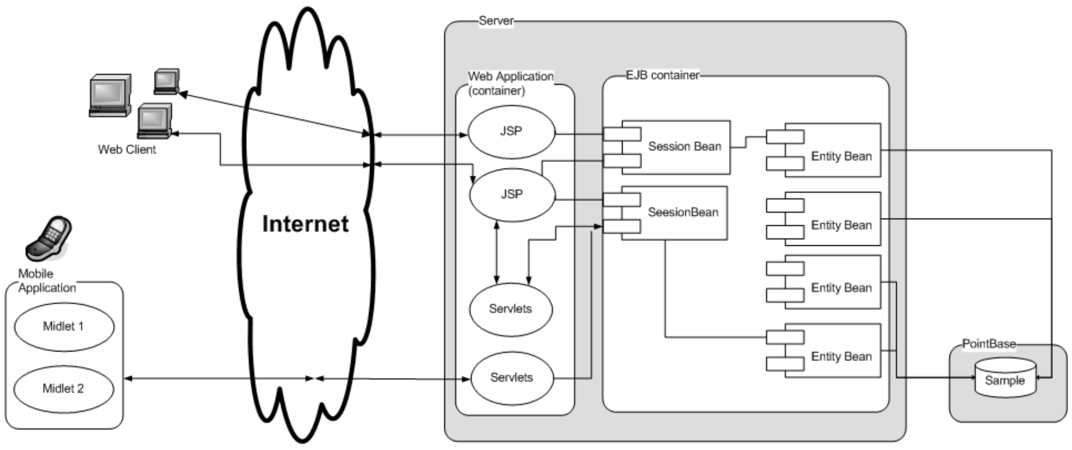


Figure 3: General Architecture of GPS Tracker application

3.4.1 Components

As shown in the figure 3, there are 3 main components. The client side which contains an Java Midlet application (loaded in the mobile phone) and the browsers. The application server which contains the Web container (JSPs and servlets) and the EJB container where the business logic (Session beans and Entity beans) stands. The third tier is the database managing the tables where the entity beans store their values.

3.4.2 Software Modules

Now we enter into the different modules which are contained in the main components. We will treat them by decreasing complexity. Finally we will describe the communication layer between the different modules.

1. EJB container: description of the Session and Entity Bean
2. WEB container: description of the JSPs and Servlets
3. Database: description of the tables and relations
4. Midlet: description of the Java mobile application

3.4.3 EJB container: description of Entity Bean and the Session

The figure 4 shows the definition of the entity beans class and the relation between them.

Primary Key management. The primary key management is implemented with an entity bean. The idea is to avoid for the user to give manually a unique key for each new position, track, address, user, etc. This is simply impossible to do it in a large scale. Also managing the primary key allow the system to be "database vendor independent".
For this we have set up PrimeKeyEntityBean (Figure 4) which is linked with the table `tracker_primekey`. This table has one unique record with fields defined in the section 3.4.5. Each time the entity is created, its id is given by calling getPersonId(), getAddressId, getTrackId, etc. Then we increment the value using setPerson(), setAddress, etc.

Note: The definition of the bean classes includes implicitly the different interfaces needed in the J2EE specifications. Also in a class, the standard ejbCreate(), ejbActivate() methods, etc. are underlying (ref [9, 2]).

The application uses the pattern SessionFacade [1]. The client does not access the entity bean directly but the session bean which provide methods. Two session beans are described in the figure 5. The pattern Service Locator has also been implemented in order to separate the search of the entity bean instance each time the session bean needs it. Finally, the figure 6 describe the complete component (with associations) of the EJB container:

3.4.4 WEB container: description of the JSPs and Servlets

The figure 7 shows the different JSPs and Servlets that the Web container manages [3]. The association represent the navigation between the JSPs. This figure contains the web access JSPs. The JSPs are composed with HTML code for the front end and quite a lot of Java code which call methods in the session beans. We still need to describe the servlets which allow the mobile phone to access the application server.

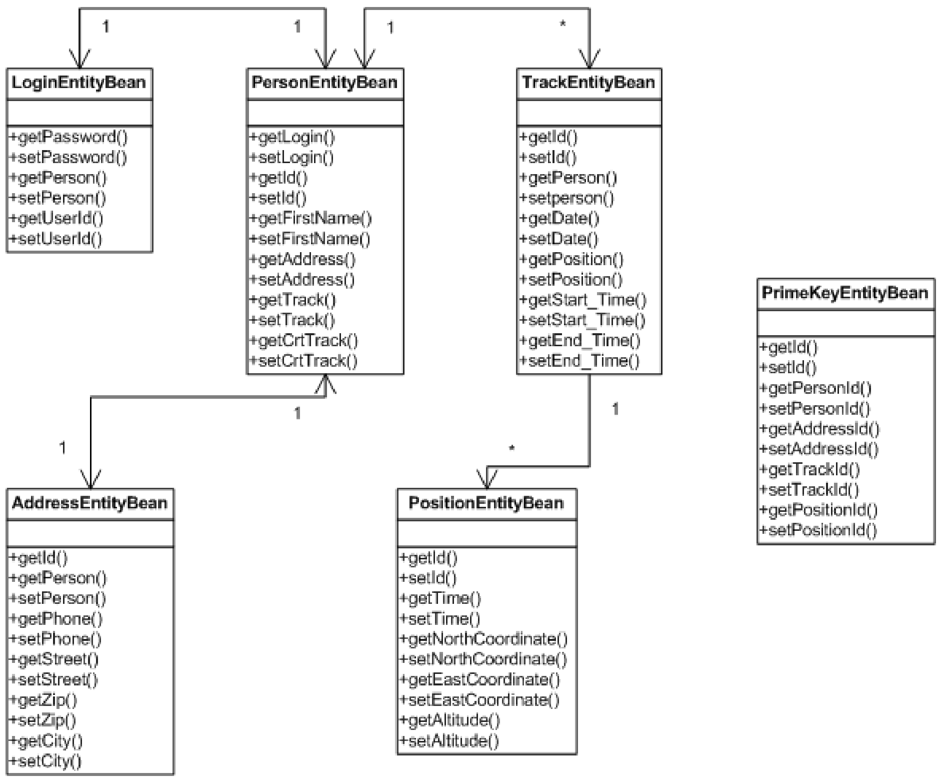


Figure 4: Entity Bean: Class diagram

Servlets. Four servlets are used specifically to interface the EJB container.

1. StartNewTrack.class
2. PostNewPosition.class
3. DrawTrack.class
4. GetPointList.class

The StartATrack servlet receives HTTP requests from the Midlet and calls the method startRemoteTrack(userId) in the TrackManagementSessionBean. If the operation is done it sends the a string "track *X* has started" (Figure 8). Same principle is used to send a new position from the mobile device. The servlet PostNewPosition receives HTTP request and sends HTTP response using the methods addRemotePosition() in TrackManagementSessionBean. Example of the doGet in the StartNewTrack is shown under. Here we see the power of the Session facade. Once the client method is identified, it is very simple to interface with the entity beans.

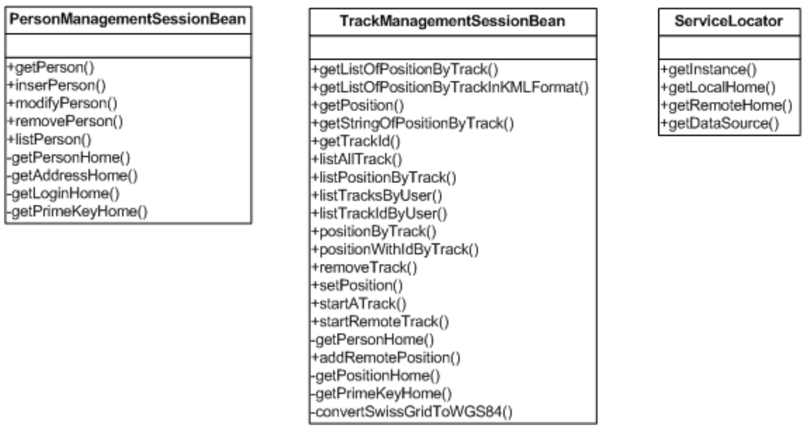


Figure 5: Session Bean and Service Locator: Class diagram

```
public void doGet(HttpServletRequest request, HttpServletResponse response)
                  throws IOException, ServletException {

        response.setContentType("text/plain");
        PrintWriter out = response.getWriter();

        String data = request.getParameter("data");

        res = data;

        if (data != null && data.length() > 0){
            res = trackManagement.startRemoteTrack(data);

        }
        out.print(res);
        try {
            trackManagement.remove();
        } catch (RemoteException e) {
            e.printStackTrace();
        } catch (RemoveException e) {
            e.printStackTrace();
        }
 }
```

GPS Coordinates systems As first and important remark, the system of coordinates used in the project is the Swiss Grid. It means that the north and east coordinates in the form of xxx.yyy (ex: 560.230, 170.040). This system is explain on the web site http://www.swisstopo.ch/.
The applet TrackDrawing (explain under) is using this system to draw the track

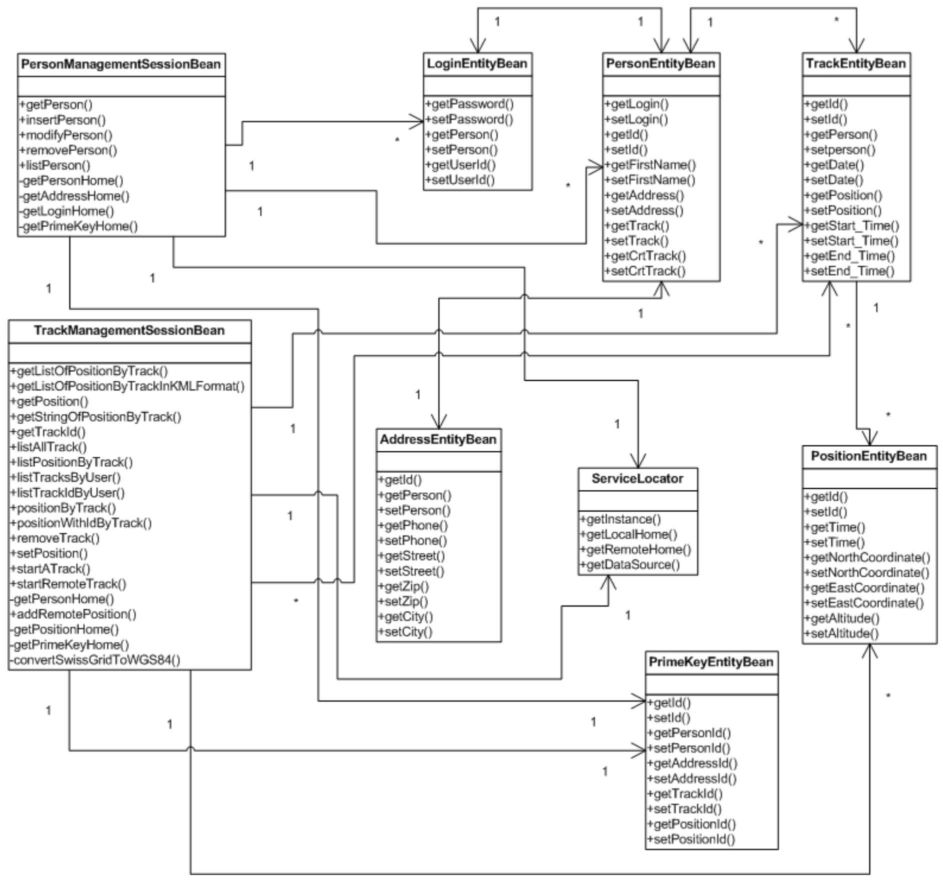


Figure 6: EJB Container - Complete schema of classes with associations

on the Swiss topo map. The position are saved in this format in the table tracker_position.
Google Earth is using the system WGS84 (deg, min, sec.).

Drawing a track To be able to draw a track on a map, the applet Track-Drawing is launched in the JSP Draw.jsp. This applet gets a list of points from the TrackManagementSessionBean by calling the servlet GetPointList. Then the servlet calls the method getStringOfPositionByTrack(track). The applet refresh the drawing with the elapsed time given as parameter (Figure 7). To get access to this method the applet must call the servlet otherwise the ejb container does not allow to access its session bean without special trust parameters. The servlet being in the same application server (same virtual machine) as the session bean, it is not consider as ”foreigner” and can get access to the EJBs. Finally, DrawTrack is the servlet launched in the viewTrack.jsp (”Draw Track with Google Earth” button). It converts the list of points of the given track

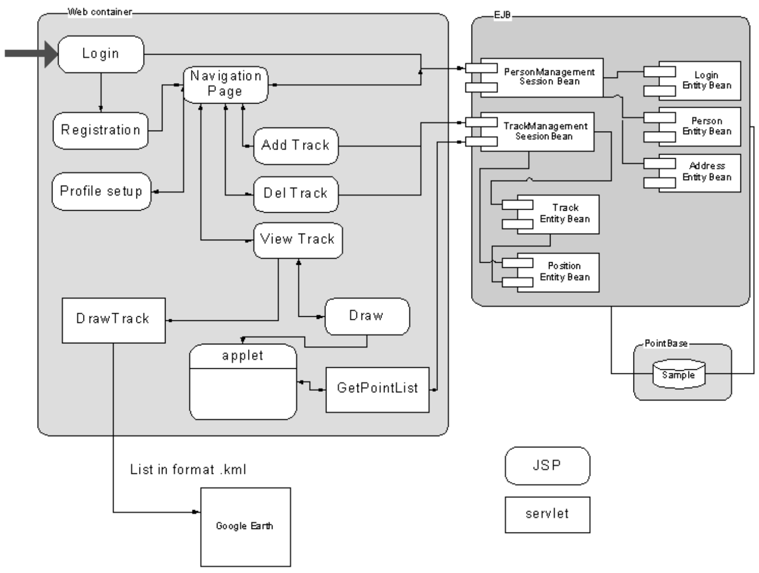


Figure 7: Web Container - Schema of JSPs and Servlets with associations for web browser access

into KML format (Figure 7). The response is in the form of:

```
response.setContentType("application/keyhole");

        out.println("<?xml version=\"1.0\" encoding=\"UTF-8\"?>" );
        out.println("<kml xmlns=\"http://earth.google.com/kml/2.0\">");
        out.println("<Placemark>");
        out.println("<description>"+ comment +"</description>");
        out.println("<name> Track "+ track +"</name>");
        out.println("<LookAt>");
        out.println("<longitude>"+ longitude +"</longitude>");
        out.println("<latitude>"+ latitude +"</latitude>");
        ...
        ...
```

and using the MIME type to launch Goople Earth. The KML format is a XML based format used by Google Earth to describe a track (for instance). The complete syntax is explain in [4]. Before drawing the track on Google Earth, we have to translate the Swiss grid format into WGS84 format. This is done by getListOfPositionByTrackInKMLFormat(track). This method calls the private method convertSwissGridToWGS84(String[] coordinates) which convert each points into WGS84 format.

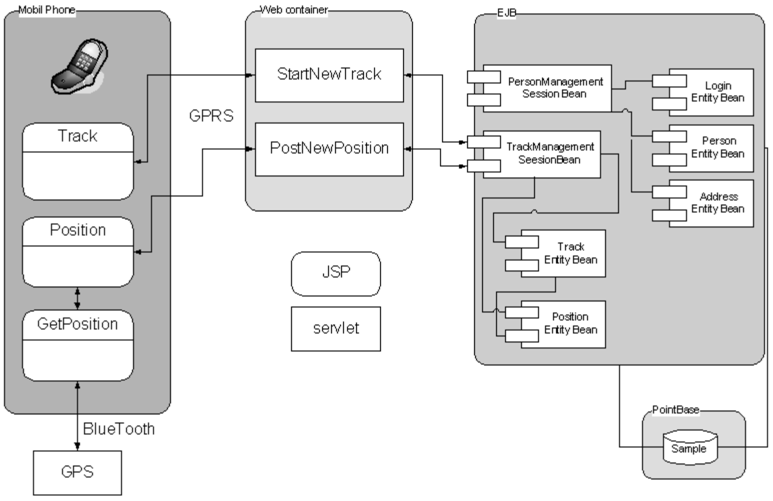


Figure 8: Servlets - Access from a mobile device to the EJB using the servlets

Applet TrackDrawing. The applet TrackDrawing is taking as input the list of points to be drawn and a map in jpg format. At first it calculate the maximum and minimum of the map coordinates (North and East) given as parameter and find the ratio *coordinate's range/# of pixels of the graphic window* (ex: (580-560)/800). The graphic dimension are given by the input parameters *resolutionX, resolutionY*. The string of points is "tokenized" and all the points are drawn in their position according to the map calibration. If a point is out range, it is taken in consideration but not drawn (virtually out of the map). The points are linked with simple lines.
The maps are jpg files and must be calibrated. As example: a map covering a region from North 560.000 to 580.000 and East 150.000 to 170.000 will have 560.000/150.000 as 0.0/0.0. The maps are saved in ./src/web (cf section 3.4.9).

3.4.5 Database: description of the tables and relations

Each Entity Bean represents the storage medium and must correspond to a table in the database. We going to describe in pseudo SQL command the fields of the six tables contained in the database SAMPLE.

```
TABLE TRACKER_PERSON (
   ID INTEGER NOT NULL CONSTRAINT PERSON_PK PRIMARY KEY,
   FIRSTNAME VARCHAR(24) NOT NULL,
   LASTNAME VARCHAR(24) NOT NULL,
   ADDRESS_ID INTEGER NOT NULL,
   CRT_TRACK INTEGER NOT NULL,
   USER_ID INTEGER);
```

```
TABLE TRACKER_ADDRESS (
   ID INTEGER NOT NULL CONSTRAINT ADDRESS_PK PRIMARY KEY,
   STREET VARCHAR(24) NOT NULL,
   ZIP INTEGER NOT NULL,
   CITY VARCHAR(24) NOT NULL,
   PHONE VARCHAR(12) NOT NULL);

TABLE TRACKER_PERSON ADD CONSTRAINT ADDRESS_FK
FOREIGN KEY (ADDRESS_ID) REFERENCES TRACKER_ADDRESS(ID);

TABLE TRACKER_TRACK (
   ID INTEGER CONSTRAINT TRACKER_PK PRIMARY KEY,
   DATE DATE NOT NULL,
   START_TIME TIME NOT NULL,
   END_TIME TIME ,
   PERSON_ID INTEGER NOT NULL);

TABLE TRACKER_TRACK ADD CONSTRAINT PERSON_FK
FOREIGN KEY (PERSON_ID) REFERENCES TRACKER_PERSON(ID);

TABLE TRACKER_POSITION (
   ID INTEGER CONSTRAINT POSITION_PK PRIMARY KEY,
   EAST VARCHAR(8) NOT NULL,
   NORTH  VARCHAR(8) NOT NULL,
   ALTITUDE INTEGER NOT NULL,
   TIME TIME NOT NULL,
   TRACK_ID INTEGER NOT NULL);

TABLE TRACKER_POSITION ADD CONSTRAINT TRACKER_FK
FOREIGN KEY (TRACK_ID) REFERENCES TRACKER_TRACK(ID);

TABLE TRACKER_PRIMEKEY (
   ID INTEGER CONSTRAINT PRIMEKEY_PK PRIMARY KEY,
   PERSON_ID INTEGER NOT NULL,
   ADDRESS_ID INTEGER NOT NULL,
   TRACK_ID INTEGER NOT NULL,
   POSITION_ID INTEGER NOT NULL);

TABLE TRACKER_LOGIN (
   USERID INTEGER CONSTRAINT USERID_PK PRIMARY KEY,
   PASSWORD VARCHAR(15) NOT NULL,
   PERSON_ID INTEGER NOT NULL);

TABLE TRACKER_LOGIN ADD CONSTRAINT PERSON_FK
FOREIGN KEY (PERSON_ID) REFERENCES TRACKER_PERSON(ID);
TABLE TRACKER_PERSON ADD CONSTRAINT USERID_FK
```

```
FOREIGN KEY (USER_ID) REFERENCES TRACKER_LOGIN(USERID);
```

3.4.6 Midlet: description of the Java mobile application

The mobile application is developed in Java using J2ME 2.2. The general structure is shown in the figure 8. Now we describe more formally the 2 midlets and the helping class.

Note: It is supposed that the reader has the necessary knowledge in Java Midlet application. Otherwise refer to [5, 6, 7, 8].

More precisely we have the description in figure 9 of the Midlets architecture. Again, only the methods which are needed to request or send data are shown. The "not described" methods are usual ones in a Midlet (Menu definition, button, action, etc.). The method addName() request the user Id in order to create a track or send a position. invokeServlet does the http request to the servlet and wait for the answer.

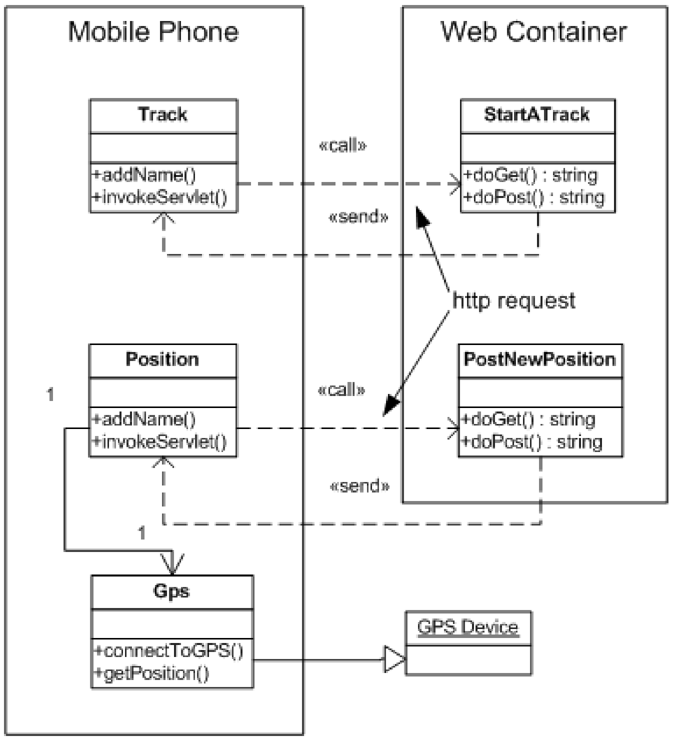


Figure 9: Midlets - complete architecture of the mobile application

3.4.7 Implementation status

Every module which have been described above are implemented. The only part of the project not fully implemented is the Midlets and especially the communication with the GPS device. The communication Bluetooth between the GPS and the mobile phone has been developed partly (Otto Poveda) but not integrate into the present implementation. The actual situation is that a class PositionList is simulating the GPS class and a pre-defined list of gps coordinates is used to simulate the positions (Figure 10). It is important to realise that the actual project is only a prototype of what could be the application. A lot of implementation details and choices (name of methods, security roles, etc) are not satisfactory or not implemented.

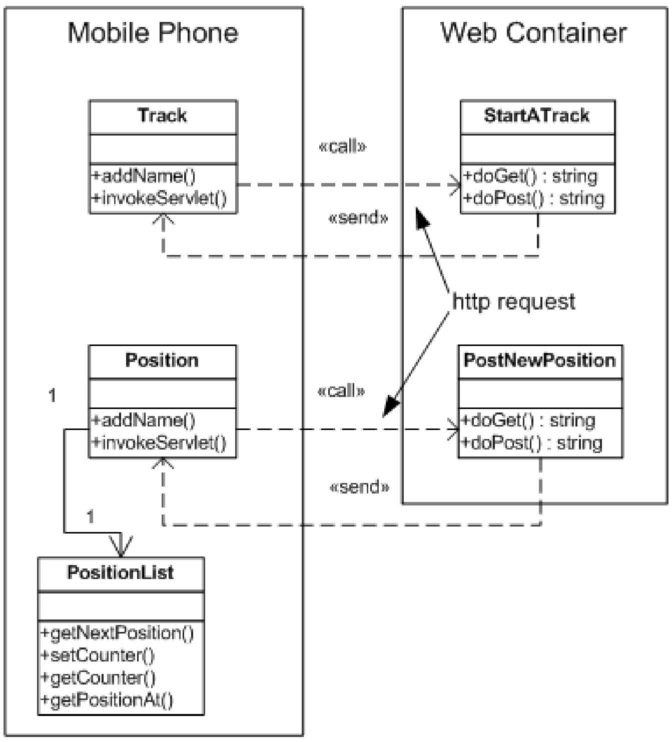


Figure 10: Midlets - architecture of the mobile application using the Position list

3.4.8 Communication between modules

The communication layer is describe in the figure 11. For the project, Http request between mobile phone and server (using GPRS) is implemented. The idea is to develop the SMS communication in the future.

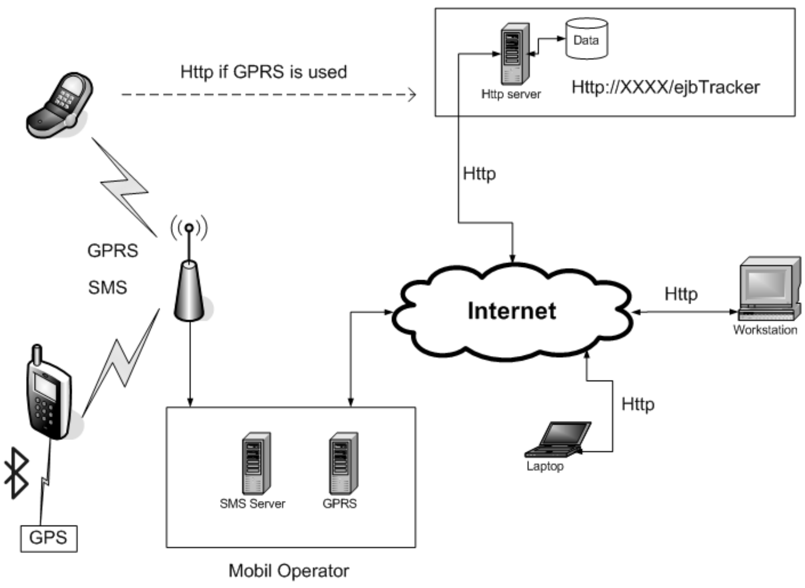


Figure 11: Communication between the different modules of the application

3.4.9 Package's structure

Server application. The application is divided into several packages. The packages listed below contain the following source files:

- tracker.ejb.person
  - AddressEntityBean.java
  - PersonEntityBean.java
  - LoginEntityBean.java
  - PersonManagementSessionBean.java
- tracker.ejb.track
  - PositionEntityBean.java
  - TrackEntityBean.java
  - TrackManagementSessionBean.java
- tracker.ejb.util
  - PrimeKeyEntityBean.java
- tracker.util
  - ServiceLocator.java
  - ServiceLocatorBase.java

    - ServiceLocatorException.java
    - ServiceLocatorSUNRemote.java

- tracker.ejb.webapp.applets
    - TrackDrawing.java
    - TrackDrawingClient.java
- tracker.ejb.webapp.servlets
    - DrawTrack.java
    - GetPointsList.java
    - PostNewPosition.java
    - StartNewTrack.java

The servlets and JSPs are saved in the standard (J2EE structure) src/web/ directory.

Mobile application. The source file of the Midlets are saved in the CD-ROM under ../java_Sources_eclipse-Project/WTK2.2-Apps and the mobilTracker.jad is under /WTK2.2-apps/MobilTracker.

4 User's guide

The application can run on a local machine very easily. The machine must have the Java SDK 1.5, SUN application server with PointBase, Java Wireless Tool Kit 2.2 installed.

4.1 Installation procedures

Application Server. At first the SUN application server must be installed completely with PointBase on the machine and correctly configured. The server is available in http://java.sun.com/j2ee/1.4/download.html#sdk or on the CD-ROM delivered with the application.
Once installed and running, some setup have to be made in order to use the correct PointBase database:

1. Open the server admin console (http://localhost:4848/admingui/TopFrameset)
2. Go to Resources, JDBC, Connection Pools, PointBasePool.
3. Go to Properties and check that:
   DatabaseName = jdbc:pointbase:server://localhost:9092/sample
   Password = pbpublic
   User = pbpublic

Usually the default domain used is Domain1 and all the files extracted from .ear will be saved under ../domain1/...

Database. Before the deployment of the file ejbTracker.ear, the database Sample must be filled with the tables tracker_xxxxx.
To do it :

1. Enter in the PointBase console startConsole.bat (under Windows) in install_dir/AppServer/pointbase/tools/serveroption.
2. Execute the file create.sql available in the CD-ROM delivered with the documentation and sources.
3. Deploy the ejbTracker.ear on the server with 2 possibilities:

Deploy the application.

1. Copy the file ejbTracker.ear under install_dir/AppServer/domains/domain1/autodeploy
2. Use the server admin console go to Applications, Entreprise Applications, Deploy... and specify the file .../ejbTracker.ear.

4.2 Download and files locations

The CD-ROM given with the documentation contains (directory and files):

- /report-presentation: the project's report and the presentation in PDF.
- /ejbTracker_project : the ejbTracker.ear, classes, create.sql, built.xml.
- /util : SUN server installer, WTK 2.2, Google Earth.
- /reference-documentation: all the PDF files mentioned in reference.
- /java_Sources_eclipse-project: The complete project in Eclipse structure (/MITS for the EJB project and /WTK2.2-Apps for the mobile application)
- /GPS-tracker_javadoc: javadoc.

4.3 Tutorial

The application is very simple to use and very intuitive. The user interfaces are:

1. Web client
2. Midlet

4.3.1 Web Client

The web client can allow 2 types of users to be log in the system: 1) the manager 2) the user. The manager is a special user and is identified in the system by the user Id number "1". **It is in fact the first user to be register**.
The figure 12 shows the web site map for a normal user. We will see what are the additional rights that the manager has later.
To access the application from the web the URL

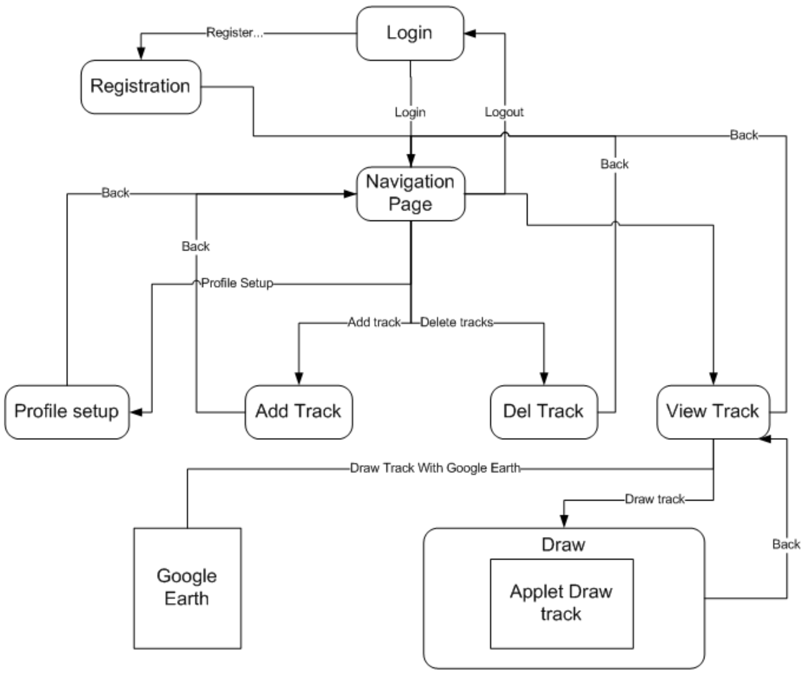
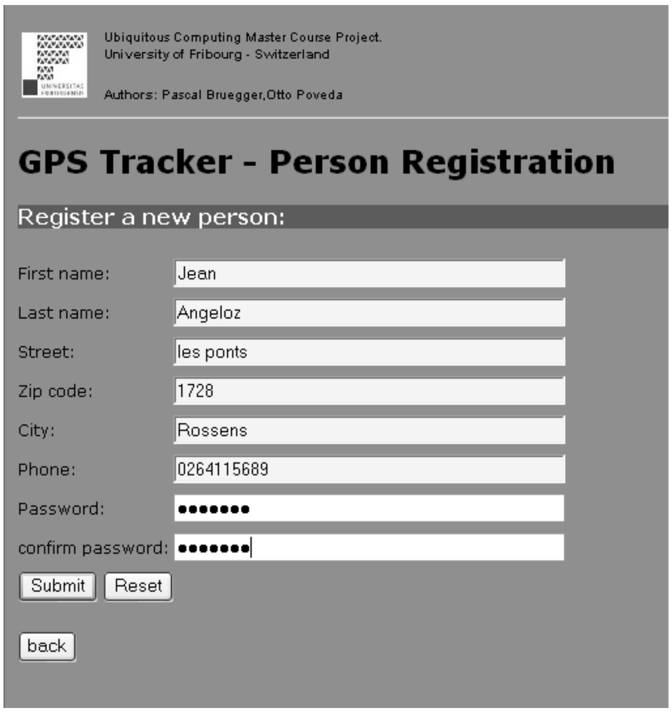


Figure 12: User navigation : site map

`http://localhost:8080/ejbtracker/` must be entered in the browser. This is valid of course if the server is local. The main page asks for a Login and if the user is not yet register he/she must register first. The user receives a user Id and will need it every time he/she enter in the system.
Once logged in, the navigation page allow the user to change its profile, add a new track, delete a track, view its different tracks, logout the system.
The different buttons are:

- Profile setup : allow the user to change its profile and password.
- Add track : allow the user to add a track. The field coordinates and altitude are optional. Once the track created, all the positions send to the server will be saved under this track Id.


Ubiquitous Computing Master Course Project.
University of Fribourg - Switzerland

Authors: Pascal Bruegger,Otto Poveda

GPS Tracker - Person Registration

Register a new person:

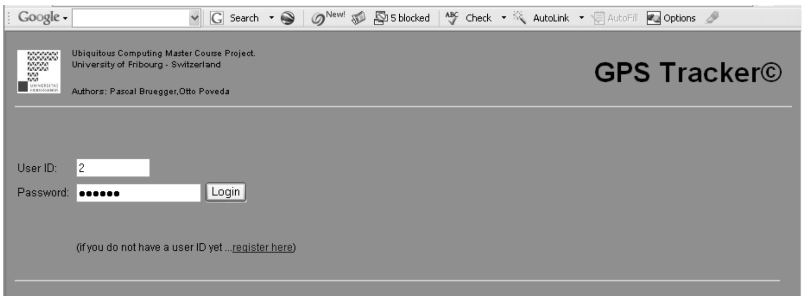
First name:	Jean
Last name:	Angeloz
Street:	les ponts
Zip code:	1728
City:	Rossens
Phone:	0264115689
Password:	•••••••
confirm password:	•••••••

Figure 13: Web Client: registration

- Delete : Delete a selected track from a list. If the track is the current one, the previous track become the current one.
- View Track: The user can view the list of track and select one from a list. Once the track selected, click *Get List of points* and then only the track can drawn on maps. Two options are available:
  1. Draw on a Swiss topo Map
  2. Draw the track using Google Earth application (which must be installed!)

The manager has the same Profile setup and Logout as the standard user. The differences are:

- Users... : allow the manager to delete a user.
- Tracks...: show all active tracks in the system (all users) and can draw them with the *View track* of the standard user.
- GPS points : The manager can modify a specific position by selecting a track then the particular point of this track.



Ubiquitous Computing Master Course Project.
University of Fribourg - Switzerland

Authors: Pascal Bruegger,Otto Poveda

GPS Tracker©

User ID: 2
Password: ●●●●●●

(if you do not have a user ID yet ...register here)

Figure 14: Web Client: login

4.3.2 Midlet

At that level of development, the midlets are available only with the Java Simulator. KtoolBar of the WTK must be launched. You must open the project MobilTracker and run it.
The Mobile Application offers two menus which allow the user to:

1. Create a new track

2. Send a position to the server

The option *create a track* will send a request to server after the user provides its Id as shown in the Figure 18. When a track has started, you begin to send your positions (GPS points). The option *Start Position Tracking* propose the same interface than the *Start new Track* but when the user Id as been sent, the tracking starts and every 5 seconds a new point is sent to server (Figure 19).

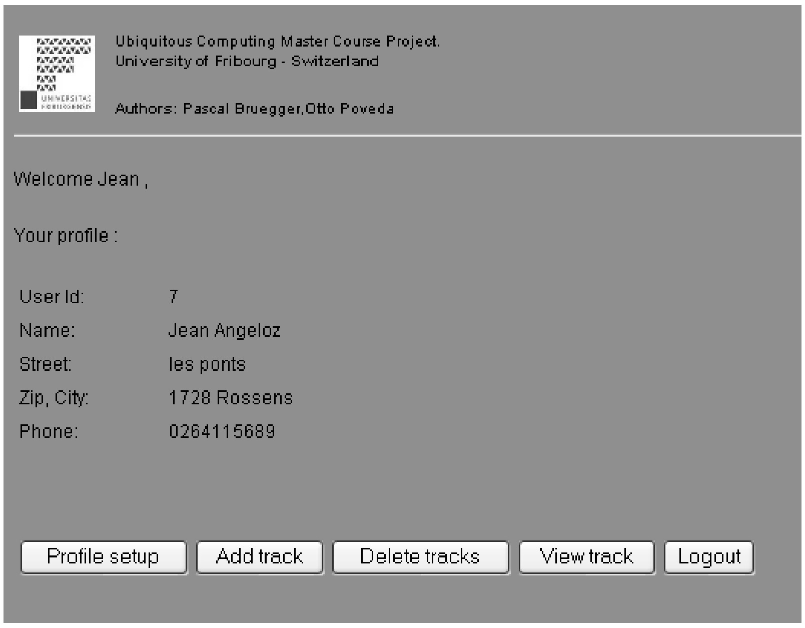
5 Evaluation

5.1 Adherence with the specification

Most of the part where developed following the specifications fixed at the beginning of this document. The server side is based on a simple but robust architecture. It offers already a relatively stable structure and the entity managed by the EJB container are what we need at that stage of the project.
The Web client side is 50 per cent sufficient. It is enough for a proof of concept but that's all.
The Midlet is enough for a proof of concept as well but need a serious work on it.

5.2 Tests

The test of user and tracks creation, delete, sending positions, drawing tracks were done several times and in very bad conditions: low memory capacity,


Ubiquitous Computing Master Course Project.
University of Fribourg - Switzerland

Authors: Pascal Bruegger,Otto Poveda

Welcome Jean ,

Your profile :

User Id:	7
Name:	Jean Angeloz
Street:	les ponts
Zip, City:	1728 Rossens
Phone:	0264115689

Figure 15: Web Client: Navigation page

machine unstable with to many open applications, server unstable, and so on. Basically, the tests were done in a pure messy developer environment without any proper setup. The tests on the application itself were quite successful. The major problems came from the SUN application server.

5.3 Known Problems

As mentioned above we had some problems but not as many as expected. Here is the list of most severe problems:

- We had one serious database corruption due to external access and manual unlock of tables. The database had to be recreated (Pointbase is only an evaluation version!)
- At least two times, the SUN application Server crashed and had to be re-installed. Again the version used is only a free version.
- The Http session are not well managed. The login is a real ”secure Login”. When a user is in the system, an Http session is created and its context is available during the session. But there is a bug when no activities where done for a long period: the server raises an exception (which is good in term of security) when the client try to refresh or access the system. I was not able to catch this exception and redirect the request to the login page

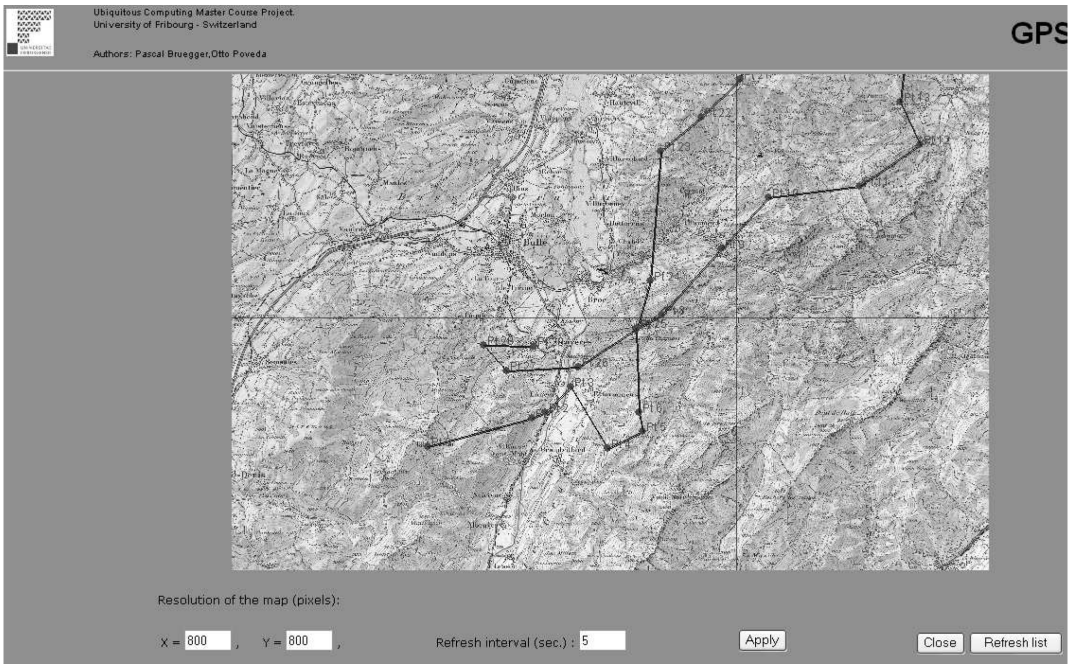

Ubiquitous Computing Master Course Project.
University of Fribourg - Switzerland
Authors: Pascal Bruegger,Otto Poveda
GPS
Resolution of the map (pixels):
X = 800 , Y = 800 ,
Refresh interval (sec.) : 5
Apply
Close
Refresh list

Figure 16: Web Client: Drawing the track on the Swiss Topo map

indicating to the user that the session is over and he must authenticate himself again.

- We have not succeeded in properly saving the applet trackDrawing in the package tracker.ejb.webApp and use it from here. We still need to save it in the src/web where all the JSPs stand and compile it before.
- No real and efficient security is setup and the Midlets can send only the user ID to start a track for instance (unacceptable in a real usage).

6 Future Works

6.1 To do

First of all, the project at that stage is only a prototype. There is a long list of possible extensions of such project. But the development of a real program usable on the Internet should be built with professional tools like JBoss Application server, Studio Creator for the interfaces, Pointbase or Oracle full licence, etc.
Now coming back to this version of the project, the part which has not been developed as wanted is the communication between GPS and Mobile phone. That is the main remaining work for this prototype. We have to create a class

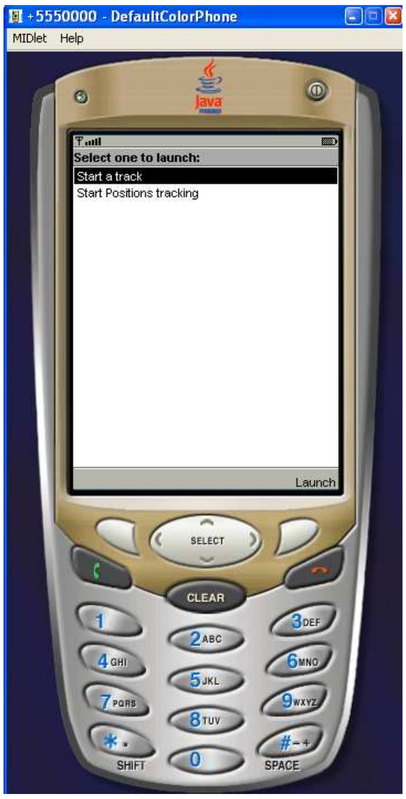


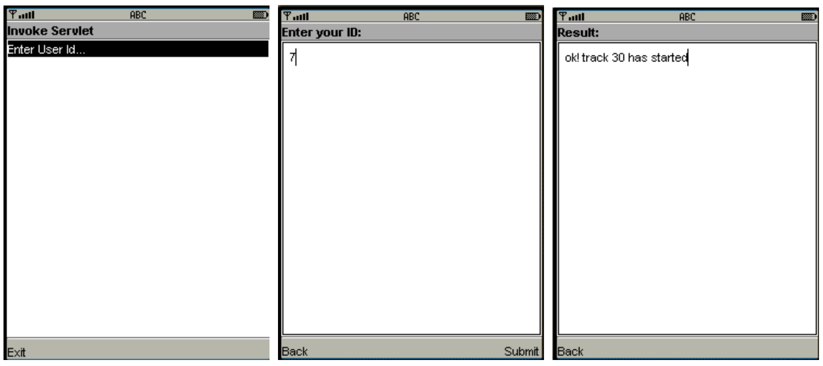
Figure 17: Midlet: Main menu

GPS which open a Bluetooth session with the GPS and collect the positions and send it to the class Position in an appropriate format. The class Position must also be able in case of no communication with the server to store the positions in a list and then when possible send them all. The communication using SMS should be develop and an automatic switch between GPRS and SMS must be implemented.

6.2 Possible extensions

Here is a list of possible extension for such project.

- Server implementation
  - A Map Entity bean in order to choose the appropriate map according to the range of coordinate of the track.
  - A dynamic selection of coordinate system (Swiss Grid, WGS 84, etc.)
  - Definition of the Roles within the application.


Invoke Servlet

Enter User Id...

Enter your ID:

7

Result:

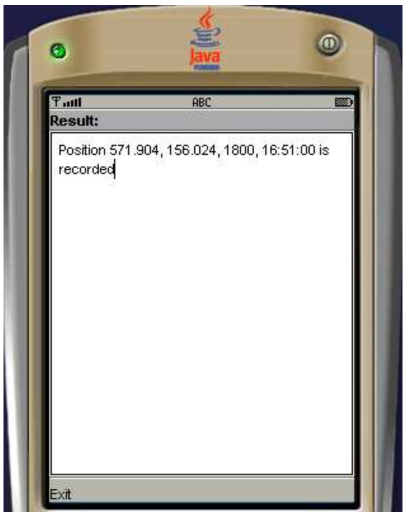
ok! track 30 has started

Figure 18: Midlet: Create a track

  - Re-definition of the Position Entity bean vs simple records in the database.
  - Client interfaces.
  - Web pages to be re-looked.

- Graphics
  - Dynamically change the view on a map (zoom in-out).
  - Interface with Google Earth real time!

- Mobile communication
  - Finalise the GPS-Mobile Phone communication.
  - Abstract communication layer : automatic switch between different type of communication (GPRS, SMS, WIFI, etc.).
  - SMS communication.
  - Mobile graphic interface to follow an other user.



Position 571.904, 156.024, 1800, 16:51:00 is recorded

Figure 19: Midlet: Send position and server response

7 Project Management

The project was supposed to be developed in a team of 2 students.

7.1 Team composition

The team is composed by:

- Pascal Bruegger
- Otto Poveda

7.2 Individual job description

1. Pascal Bruegger:
   (a) Analysis and implementation of the entity (persistence) and session beans
   (b) Analysis and implementation of the Web client (JSPs)
   (c) Implementation of the interface for the Mobile application (servlets).
2. Otto Poveda:
   (a) Analysis and implementation of the Bluetooth communication between the GPS device and the mobile phone.
   (b) Analysis and implementation of the Midlet requesting positions to the GPS.

(c) Implementation of the Midlet starting a track and sending formatted position to server (using the servlets).

7.3 Effective individual contributions

1. Pascal Bruegger:
   (a) Analysis and implementation of the entity (persistence)and session beans.
   (b) Analysis and implementation of the Web client (JSPs).
   (c) Interface with Google Earth (not defined at the beginning).
   (d) Implementation of the interface for the Mobile application.
   (e) Implementation of the Midlet starting a track and send formatted position to server (using the servlets).
2. Otto Poveda:
   (a) Implementation of the Bluetooth communication between the GPS device and the mobile phone.
   (b) Implementation of the Midlet requesting the GPS partly done.

The work done for the project presented in this document and in the class room represent about 250 to 300 hours of work for the author. This part of the project has been entirely developed by Pascal Bruegger and the part including the GPS communication with the mobile phone will be delivered by Otto Poveda (source and documentation) separately.

References

[1] John Crupi Deepak Alur and Malks. *Core J2EE Patterns - 2nd Edition.* SUN microsystem, 2004.

[2] J. Ball E. Armstrong and S. Bodoff. *The J2EE 1.4 Tutorial.* www.sun.com, 2005.

[3] Crawford Farley and Flanagan. *Java Enterprise, 2nd Edition.* O'Reilly, 2003.

[4] Google. *Google Earth KML 2.0*
. http://www.keyhole.com/kml/docs/Google_Earth_KML.pdf, 2004.

[5] Qusay H. Mahmoud. *MIDP Event Handling*
. http://developers.sun.com/techtopics/mobility/midp/articles/event/, 2004.

[6] Qusay H. Mahmoud. *MIDP Inter-Communication with CGI and Servlets*
. http://developers.sun.com/techtopics/mobility/midp/articles/event/, 2004.

[7] Qusay H. Mahmoud. *Wireless Application Programming: MIDP Programming and Packaging Basics.* http://developers.sun.com/techtopics/mobility/midp/articles/event/, 2004.

[8] Sun Microsystem. *User's Guide - Wireless ToolKit version 1.0.4.* www.sun.com, 2004.

[9] Richard Monson-Haefel. *Entreprise JavaBeans 4th Edition.* O'Reilly, 2004.