Let your friends and mobile phone help you

Konstantinos Karampogias
(10-937-498)

Semester Thesis
Lab: Institute for Computer Engineering and Networks Laboratory (TIK) (www.tik.ee.ethz.ch)
Group: Distributed Computing Group(DISCO) (www.disco.ethz.ch)
Supervisor: Prof. Roger Wattenhofer
Advisor: Johannes Schneider
March 2011 to August 2011

Abstract

The purpose of this semester project was to build an innovative remind/alert/to-do based Android application. In more details, the application offers to the end user the capability to create reminders connected to Wifi/Bluetooth proximity, to an information extracted from a web page, to a route retrieved by the Google maps and finally to the id of an incoming caller. Furthermore, the application enables users to help each other by providing a sharing mechanism and enabling them to exchange alerts.

Contents

Chapter 1

Introduction

The available functionalities/capabilities of the Android platforms have enabled developers to create extremely useful applications for the end user. Nevertheless, there can be an alternative approach on how can someone take advantage of all these capabilities. This semester project proposes and implements some ideas on that direction. Besides, smartphones are mobile phones that offer great connectivity capabilities, in the sense that devices can communicate with other devices in many ways (GSM, 3g, Wifi, Bluetooth, NFC etc). Taking that into consideration, we conclude that more and more people will be constantly connected in some way. The application intents to take advantage also of this connectivity by providing a sharing mechanism. In that way application users, in particular friends, can share their data and creatively cooperate in order to facilitate their life.

1.1 Description - Innovation

To start with, the application is an alert, reminder, to-do based application. It enables users to create reminders or alerts[1] (a small note connected to a particular event). The mobile reminds them the note upon to that specific event. In a more formal way, reminder or alert is a note connected to trigger, where note is a piece of text that is displayed on screen and trigger is a condition that can be either false or true. The approach behind the application is given in details in the **Figure 1.1**.

The concept of the application is quite simple and works as follows: The user creates a note (step 1), connects this note to trigger (step 2), the mobile will constantly check for which of the triggers are active (step 3), and it will remind to the user the note when the relevant trigger is observed (step 4).

In order to clarify the usefulness of the application a trivial example is given: The user creates the note "I need to buy oil", connects this note to the trigger "when I am in the supermarket". The mobile application will exploit the hardware/Android platform capabilities and will recognise in some way when this trigger is valid. Finally, the mobile will remind the note to the user. These types of examples are simple but also very useful in daily life.

The innovative part of this project are the triggers and the way that the mobile application will judge if they are valid or not. For instance, in the previous example, the mobile needs to detect when the user is in a supermarket, there can be various ways with which the device can acquire this information. For instance, in a classic way, by using the GPS of the phone or by asking the user to define the place. Nevertheless, it can be done more creatively by using proximity to a Wifi access point located in that location, by listening to the microphone or by taking a photo and then trying to extract patterns which identifies the location.

The hard and the most time consuming part of this project was the graphic user interface (GUI). It was given emphasis on how it will be practical enough, on how it will demand the minimum input from the user in every case, on how easy will be for the user to understand it.

[1] We will use the terms alert and reminder interchangeably.

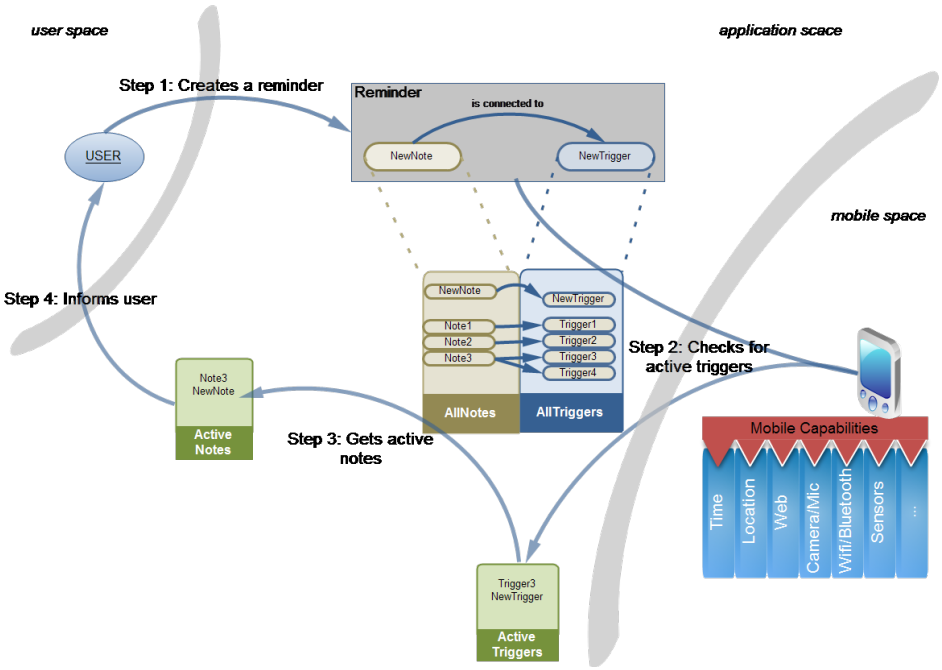


Figure 1.1: Application idea

Apart from the main idea, a simple sharing scheme was also implemented in order to enhance the usefulness of the application. Users of the application are able using their registered email to communicate and share creatively reminders/notes/alerts between them. The idea is depicted in the **Figure 1.2**.

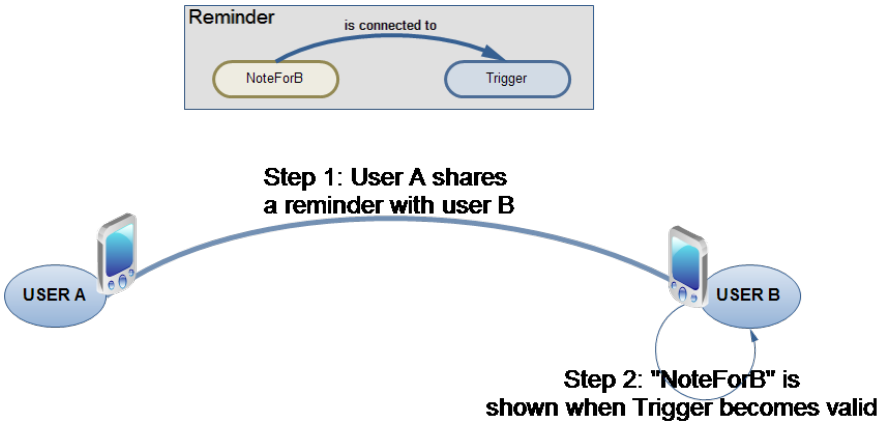


Figure 1.2: Sharing idea

1.2 Organization

The Chapter 2 is devoted to thoroughly explain and present what kind of triggers have been implemented and provide some intuitive examples how these triggers can be useful. Furthermore, in that chapter a comprehensive presentation of graphical interface is given along with some important comments. Chapter 3 holds the technical information of the project, and how the ideas were implemented, what tools were used, what were the restrictions and describes some solutions that were introduced. Finally, the last chapter refers some potential future triggers/extensions of the application.

Chapter 2

Application

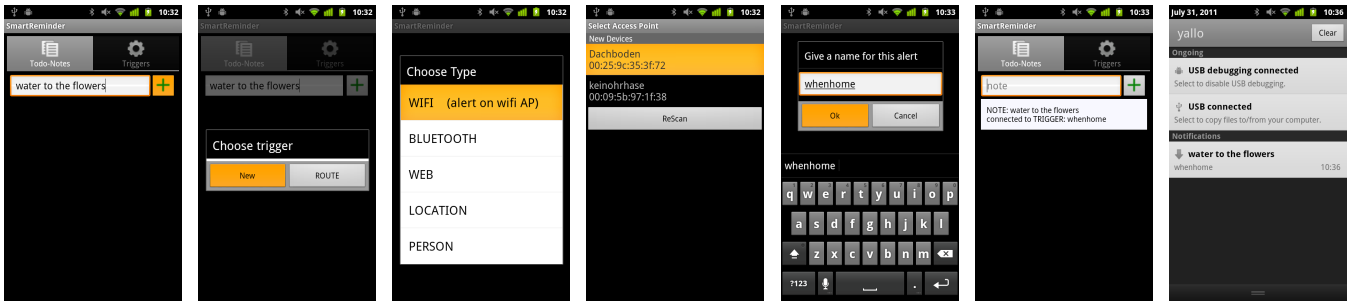
2.1 Let Your Mobile Phone Help You

The first section is devoted entirely on the features which take advantage of the mobile device. The modern mobile devices offer a variety of tools/possibilities (sensors, wifi, Bluetooth, GPS etc) that can be used in a different way that their original purpose and offer some new perspectives to the user experience and to the developer's potentials.

2.1.1 Feature 1: Proximity Alerts

The first feature of the application is the proximity based alerts. A user can create a note connected to a known network device, and be reminded to that note when his mobile is in proximity with this specific device.

The idea for that feature started from the fact that mobile phones are without doubt wifi and Bluetooth enabled, and the user has the ability to connect to a wifi access point as well as to connect its mobile with another Bluetooth device. This is the classical use, but there is an alternative approach. We can use this functionality in order to get a sense of proximity to a specific device, and consequently acquire information which might be useful for the user. Getting the proximity is simple and straightforward, wifi access points, as well as discoverable Bluetooth devices, broadcast an EM signal on the air, the mobile can receive this signal and can do a proximity estimation. Therefore, the mobile phone can provide a proximity alert efficiently, since sensing the air is not power consuming task and should work flawless because it requires no active connection. In the **Figure 2.1** we present step by step the procedure, which the user should follow in order to create this type of alert.



(a) (b) (c) (d) (e) (f) (g)

Figure 2.1: Proximity based alerts

In order to indicate the usefulness of this feature we provide some more examples:
1. Proximity on a MAC address of a wifi access point: when the user will be at home/office, remind him to do something 2. Proximity on a wifi essid: when I am in proximity with a known wifi network with ESSID "Starbucks", remind user to buy a coffee. 3. Proximity on a static Bluetooth device: when user is in proximity with his laptop, remind him to search the internet for a topic. 4. Proximity on a mobile Bluetooth device: when user is in proximity with his friend Nick, remind him to tell him something.
Nevertheless there are some topics that should strongly be under consideration:

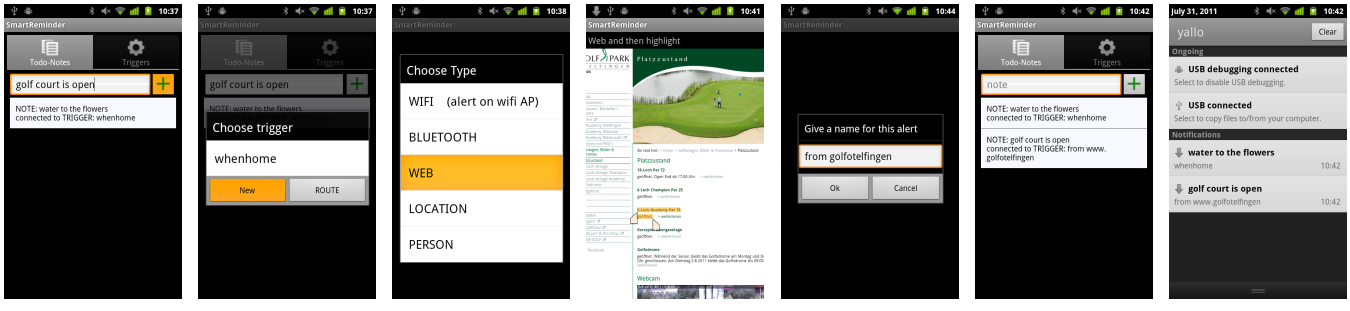
- Security issues: mac addresses and essid names are easily spoofed.
- Estimating exactly the proximity is a hard task because needs mapping the receiver level of signal to distance. In order to do that, many variables should be taken into account like the propagation model, the receiver/transmitter signal power, sensitivity etc. In that application a simple decision whether the device is in communication range[1] is made without further analysis.
- Bluetooth capabilities require that the Bluetooth adapter of the device are in discoverable mode and enabled. This usually is not the case and also it might drain the device battery faster.
- Scanning the air periodically might hide some difficulties in the implementation and might create malfunction to the mobile. Of course the efficiency depends on the code implementation, on the mobile and on the operating system.

2.1.2 Feature 2: Web Alerts

This feature is called web type alert and according to our knowledge it is unique. In short, the user is able to surf on a web page, mark a text, and connect a note to that information. The smartphone checks in the background whether this text continues to exist or not in the web page and alerts the user accordingly. To be more specific, we provide some examples where this feature can be useful to the user:

- A user wants to go swimming, but the swimming pool sometimes is closed due to bad weather or to another event. By using the application, the user can go to swimming pool's web page and mark the point where refers whether the pool is open or not, and then the mobile will inform him for a potential change.

- A second indicative example is when someone is a regular buyer in the Ebay. He can surf to the web page and mark the current value of the product. The mobile automatically will fetch this page, will check the value and will inform accordingly the user. In the **Figure 2.2**, some screenshots are given explaining how the user can create this type of alert in 5 steps.



(a) (b) (c) (d) (e) (f) (g)

Figure 2.2: Web based alerts

Nevertheless there are some topics that should be taken into account:

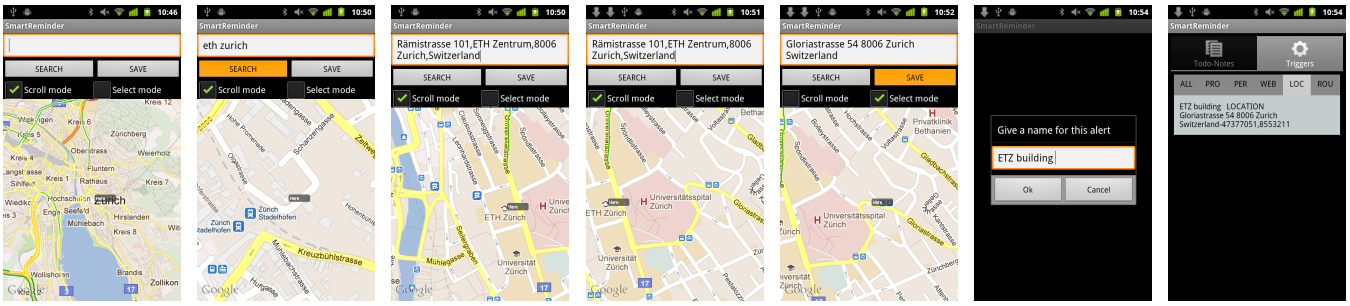
[1] given signal reception and checking to a threshold.

- User should use efficiently an embedded browser to visit fast a web page.
- How the application will understand the user input. In our case we used the copy and paste feature of the Android operating system. Another idea will be by acquiring a screenshot of the web page along with the zoom level and then give the user the ability to mark an area over that image[2]. The first way is quite simple and straightforward but it may not work if the information is given in a way[3] that copy paste is disabled. The second way is by far more complicated but independent on how the information is presented.
- How the application will check whether the search text exists in the web page. This step is tightly connected on which format the user's marked text is saved. If we have saved the user input in clear text then we can compare this text in the HTML source by using regular expression (as it was implemented) or follow a complicated XML manipulation technique. If images are used then some image comparing techniques should be used. The first case is restrictive only to classic web pages that are served in a readable html code, whereas in the second case we are not restricted, it can be applied to all web technologies (flash, HTML5 etc). An alternative interesting idea will be to use OCR (or catscha cracker in extreme cases) in order to create a hybrid solution.

2.1.3 Feature 3: Location Alerts

Another feature, which we call location based alert, is when a user connects a note to specific location point and the mobile by using its location based capabilities (GPS, coarse location etc) informs him accordingly. This feature is a not innovative but greatly completes the application purpose. An example can be the one referred in the introductory chapter where the location "Supermarket" is pin pointed by its coordinates. Another example, is to use the application as site seeing helper, by connecting information(notes) to specific sites(triggers). The effectiveness of this feature is how easily the user can search for an address. We proposed the following interface and we give an example on how it works (**Figure 2.3**). For instance, the user wants to find the ETZ building but he does not remember where it is, only where it is located in comparison with the main building of ETH.

1. the application opens an embedded map (**Figure 2.3a**)
2. the user searches in the search field the keyword "ETH Zurich" (**Figure 2.3b**),
3. this search returns and centers the map to ETH HB (**Figure 2.3b**),
4. then navigates/zooms the map to the ETHZ building (**Figure 2.3d**),
5. enables the touch screen search (**Figure 2.3e**)
6. gives a personal name to that location (**Figure 2.3f**),
7. the location is saved with the right address and coordinates (**Figure 2.3g**)



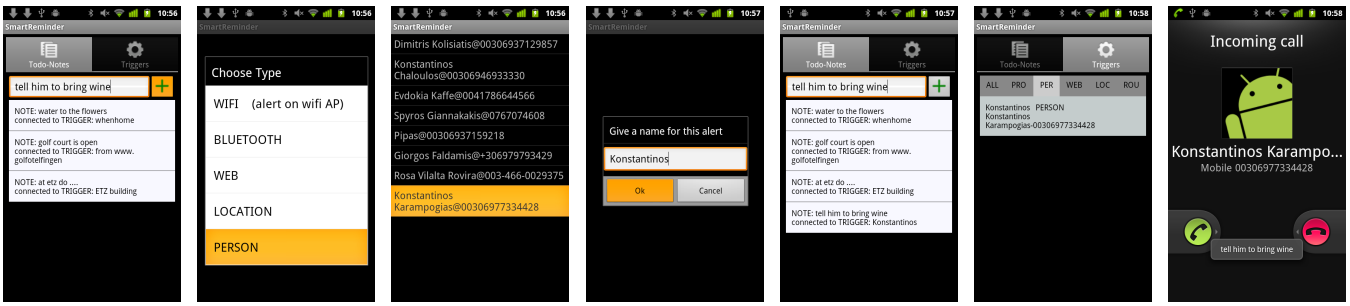
(a) (b) (c) (d) (e) (f) (g)

Figure 2.3: Location based alerts

[2] in the same that some scanners enabling to crop a specific part.
[3] the information might be hidden in an image, not in a text

2.1.4 Feature 4: Incoming Call Alerts

Another small but useful feature that was implemented in the application was the incoming call alerts. The user has the capability to create a note, and connect this note to a contact person's name (or number). The application is responsible, when this person call, to make the note appear in the screen. The usefulness of that feature is the fact that in everyday life we have many times wanted to say to another person something, but which we had finally forgotten. The efficiency of this feature is how the message will appear (e.g. when the phone starts ringing, when the user answers the call, how long etc).



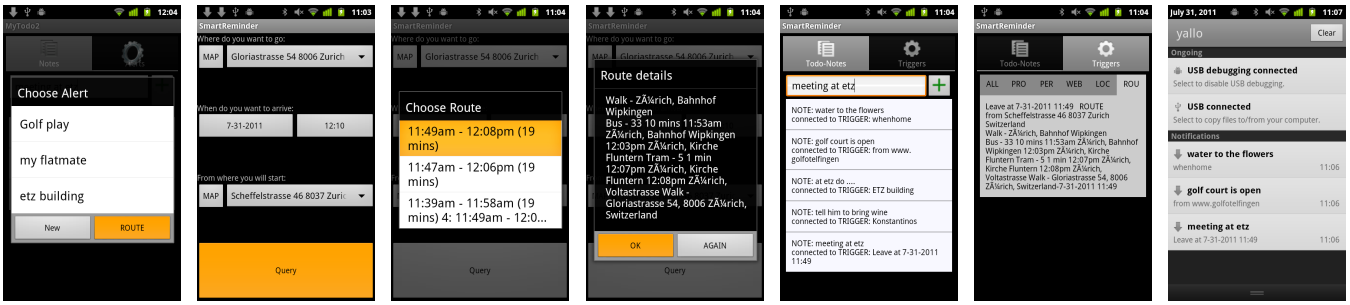
(a) (b) (c) (d) (e) (f) (g)

Figure 2.4: Incoming call based alerts

2.1.5 Feature 5: Route Alerts

This feature is called route based alert and the application enables a user to retrieve and save fast how he will reach a desired destination point at a specific time. The route that user should take, along with the exact time that he should leave is saved. Furthermore, the mobile reminds effectively the user when he should leave, by bringing an alert on the mobile screen one hour before the leaving time.

Key decision is how the application will extract the desired route. Unfortunately, there is no official API from SBB or from Google maps[4]. The effectiveness of the feature depends on how practical the GUI will be, how the information will be saved and how the user will be informed. In the **Figure 2.5**, the graphical interface of the feature and the procedure to create this type of alerts is introduced.



(a) (b) (c) (d) (e) (f) (g)

Figure 2.5: Route based alerts

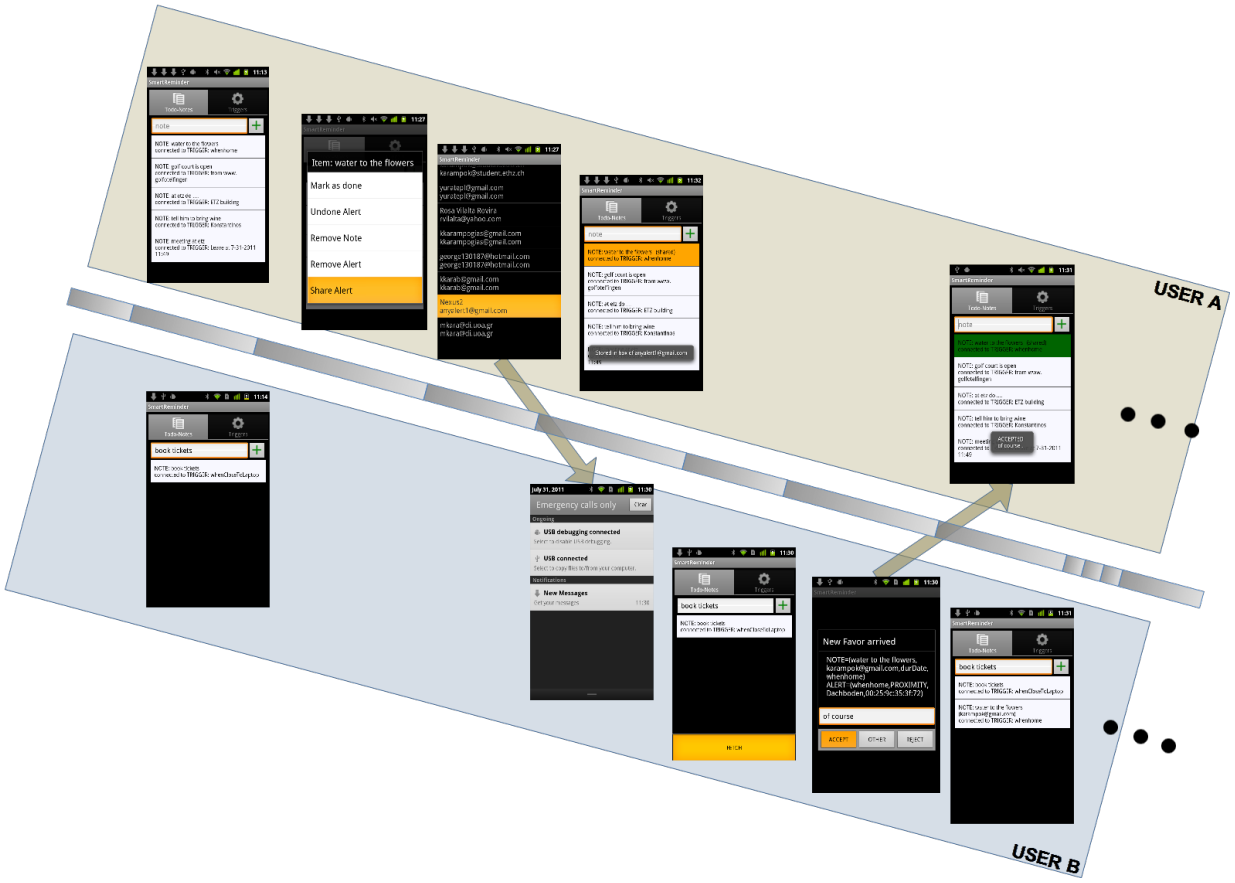
[4]which supports public transport

2.2 Let Your Friends Help You

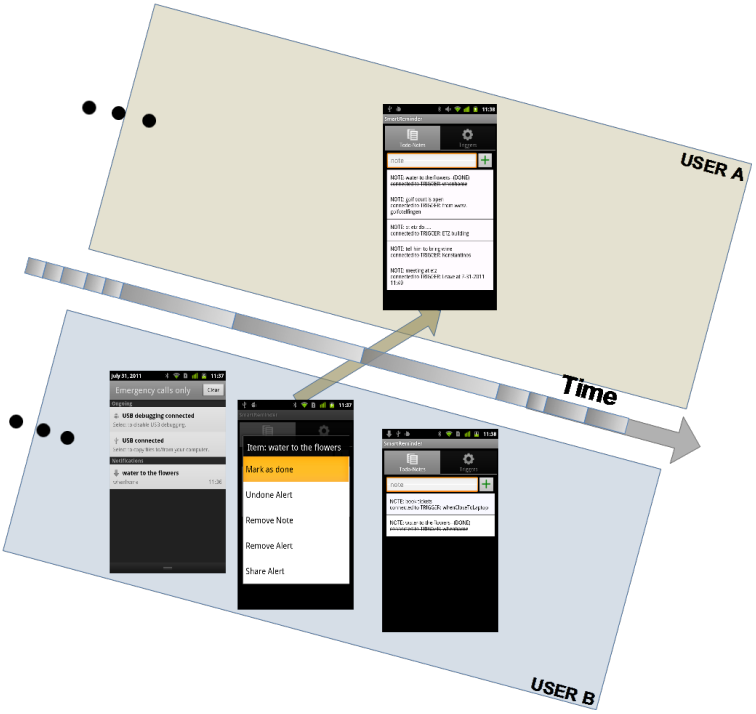
2.2.1 Simple Sharing Feature

The application provides a sharing mechanism. Thus, users can create an alert and share it with a friend. It is easy to understand how the sharing contributes in every one of the above feature. To that end, some examples are given:
"When a user A is in holidays, he wants someone to check his house. Thus he can share a note to friend and when the friend visits the house, his mobile will remind him what to check"."Flatmates can easily share a to-buy list". "A small group of people who like to play basketball, but the court opening times are not stable and are available to a web site. A sharing web type alert might be useful to inform the group accordingly". "User A can effectively remind user B to discuss a topic with a third person when the last one calls user B".

In the following **Figure 2.6** the exact steps, that a user A and user B should do in order to share an alert , are presented. The screenshots have been aligned in a time basis.



(a)



(b)

Figure 2.6: Simple sharing flow

Chapter 3

Implementation

3.1 Design

When creating the application, it was taken into account that the features were not totally fixed. Thus, the application should be able to dynamically be extended with new features. For that reason, special effort was given to have a modular and layering design. The application was primarily built over four layers

- The XML base: How the user data will be saved.
- Top level GUI: How the application will interact with the user.
- The core of the application: How the activities interact.
- Background service: How the application checks the triggers.

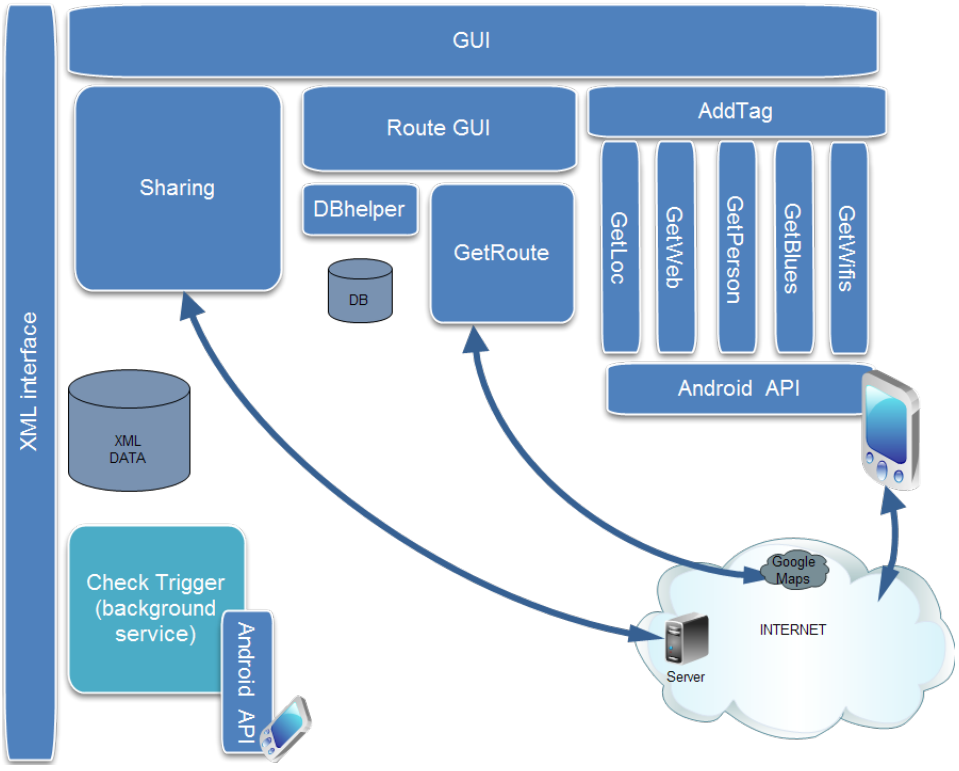


Figure 3.1: Design of the application

3.1.1 The XML Base

When starting an application, a first important decision is how the user data will be saved. There are many options, for example saving the data in a database format[1], saving the data in a binary format[2] or saving the data in custom text files. The first approach is usually preferred because developers are familiar with databases and the API offered by Android covers the most of the functions. The second approach is simple but the fact that it is complicated how the classes are directly stored in connection to fact that data are not readable might create issues. In this application the third approach was preferred. The user data was saved in two XML files (trigger.xml and notes.xml). The reasons for that decision were the followings:

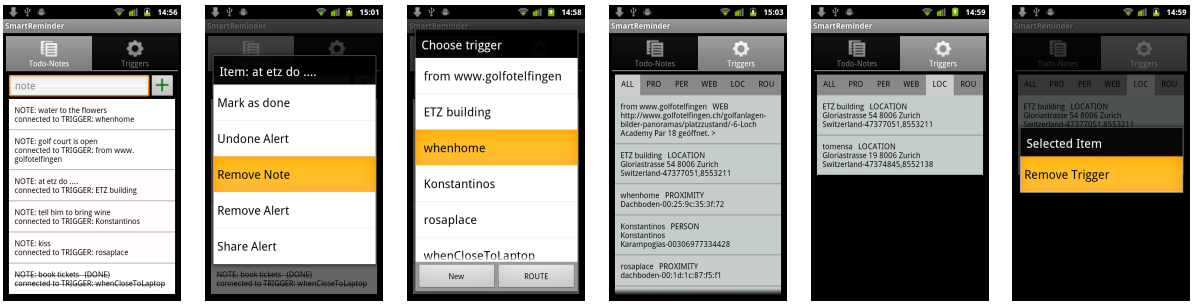
1. XML processing is a common task and there is also an extensive API.
2. the files are simple and can be easily be extended[3].
3. XML files can be by far more generic and cooperation with different platforms[4] can be possible.
4. when transferring messages through the network, XML format is preferred.
5. the data can be accessed by a third applications independently, e.g. if the files are put in the SD card, we can create a third application that can read them and use them to do a brand new task (to synchronize with Google tasks) without a lot of effort.
6. XML files can easily be used in a backup because the files which should be a saved are strictly defined[5].

However, working with XML files was not an easy task and created a lot of work.

3.1.2 Top level GUI

A nice and practical graphical interface is extremely important for a good application. It should minimize the user's input, should predict as possible the input and should be easy enough for the user to understand it without effort. In all points where the user interacts with the application we tried to do have a practical GUI. The most of the interfaces were presented and commented in the Chapter 2, in this section we introduce also the home screen of the application.

The home screen should always be simple and completely intuitive. After studying numerous to-do based application, we concluded in using a minimalistic two tabs environment. Some indicative screenshots are given to **Figure 3.2**. In the first tab, the user can see the available notes **(3.2a)**, delete a note **(3.2b)** or add fast a new alert (a note with an existing or new trigger) **(3.2c)**. In the second tab, the user can see all the saved triggers **(3.2c)**, or triggers of a specific type **(3.2d & 3.2e)** and delete a trigger **(3.2f)**.



(a) Note tab **(b)** Del note **(c)** Add note **(d)** Trigger Tab **(e)** Select trig **(f)** Del trigger

Figure 3.2: Top level GUI

[1] Android provides an SQLite database.
[2] directly serializing object to the hard disk
[3] e.g. in an XML file we can just add a new attribute.
[4] e.g. if the application is can be ported easier in IPhone or in a regular PC etc
[5] if a database was used, the developer should know how the backup is made (which files should be saved, conflicts between different versions etc). Furthermore, along with useful information, some not relevant might be saved.

3.1.3 Core

The routing based alert was built over the submodules. The first module holds the graphical interface of the feature, the second module holds the useful data for the feature (in that case points and addresses) and the third module holds the essence of the feature which is the route retrieval. As it was reported earlier the application uses a crafted html query to Google Maps in order to receive the route. If in the future, SBB releases an official free API, by changing this module only.

3.1.4 Background Check Of Triggers

The efficiency of the application depends without doubt on the part which is responsible for monitoring the active triggers. "The faster it detects a valid triggers, the more the user will like it " and "By avoiding false positive the user will trust it". Before proceeding to the implementation, the main issues that were taken consideration were the following two: Firstly, energy issues, no user wants his battery be drained fast and secondly, CPU consideration, heavy tasks in the background will result in the mobile less responsive. Both factors can decrease user experience dramatically.

We followed the regular approach where the application runs in the background and checks periodically for the active triggers. In more details, the application includes a service which wakes up every X sec[6] using the alarm manager offered by Android API. Then, it executes three steps: in the first steps gathers the available Wifi networks, then gathers the available Bluetooth devices and finally checks for the rest of triggers. After finishing these steps, the service maps the active triggers to their corresponding notes and presented to the user. Finally, the service is destroyed enabling the phone to sleep. Our approach is depicted in the following **Figure 3.3**:

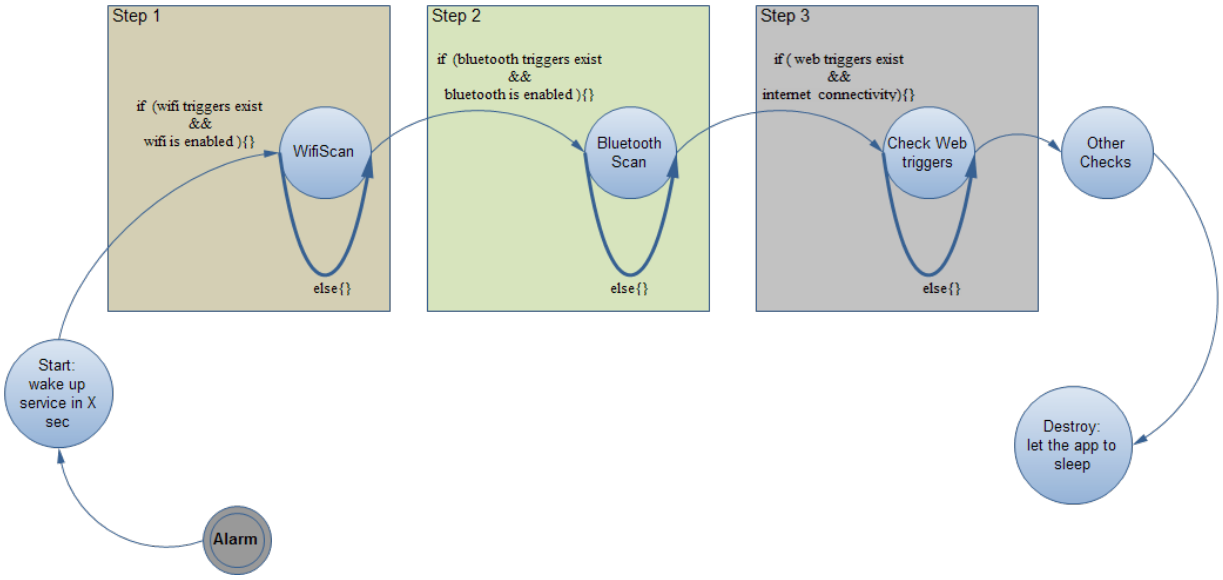


Figure 3.3: Background check of triggers

What have we succeeded by that design: the device is keeping the mobile awake for the minimum possible time, since it avoids any not necessary check. According to our observation the application keeps the device awake for 20 seconds every X seconds. Furthermore, the checks take place sequentially and hence minimize the CPU load[7]. The drawback is that it requires thoroughly testing in order to avoid a break between steps which will prevent the device from sleeping.

[6]This a variable and was chosen every 2 minutes.
[7]From our tests, the web checking where we apply a regular expressing pattern matching is of significant CPU load.

3.2 Sharing Feature

Before proceeding in the implementation of the sharing feature there are many critical points that should be taken into account.

Firstly, there is a need for a unique ID per user which should be known to his friends[8]. What will be the unique ID? This decision is critical for the application since Android Market has showed that applications which used an efficient unique ID have great success. Our options here are various: the phone number, the email, the Facebook id, the Skype ID, the name etc. We decided to use the registered email address of the device, because it is highly probable that in a smartphone, all contacts will have been saved including an email address.

Secondly, how the communication will take place? Will it be using client - server model or AdHoc? Will we use pooling or not? How many persons will take part in a communication? For simplicity reason and since an effective transport infrastructure is out of the scope of the project we adapted the use of a central server which will deliver the messages to the destinations. Furthermore, we use pooling[9] in order to avoid networking complexities (e.g. 3g networks can run behind a NAT). What is more, in the application, only one-to-one communication is enabled since implementing multicast/broadcast messages requires a lot of not innovative programming code and a more sophisticated transport infrastructure. Finally, we point out that the server is transparent to the end device, since it does not modify at all the content of the messages, just forward them to the destinations.

The sharing infrastructure is presented in the following **Figure 3.4**

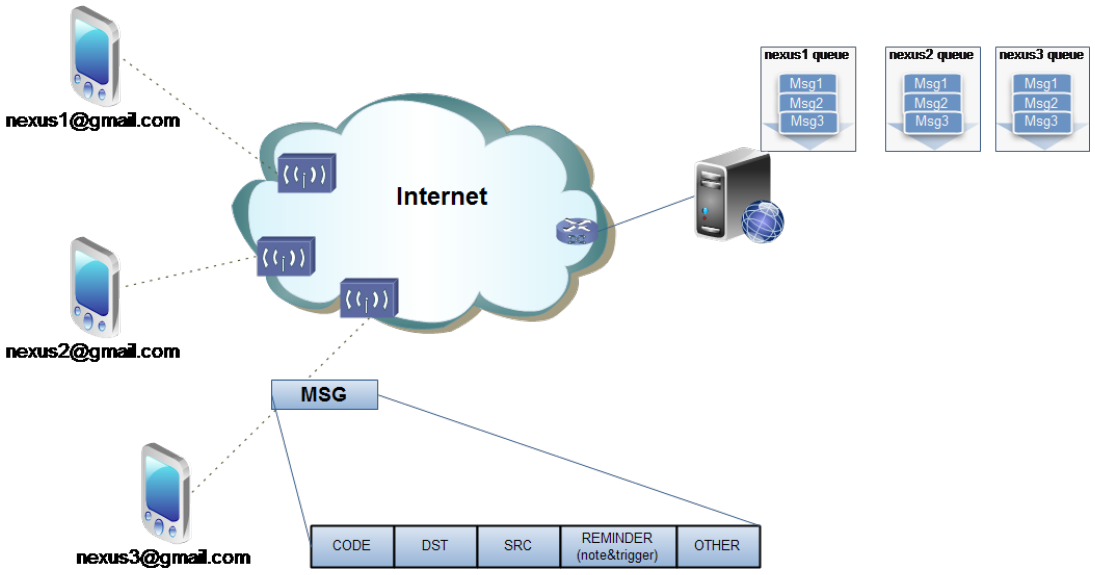


Figure 3.4: The sharing infrastructure

The essence of the sharing feature is that changes some information connected to a reminder (a note connected to a trigger) according to the messages that the application receives. In other words, there are some states associated with a reminder, a reminder is changing its state only when a local/remote action has taken place. The **Figure 3.5** gives all the possible states, as well as all, the possible transitions during sharing action.

[8] the IMEI code of the device is not useful because it is not known.

[9] every device is responsible to fetch their own messages from the server by explicitly asking him

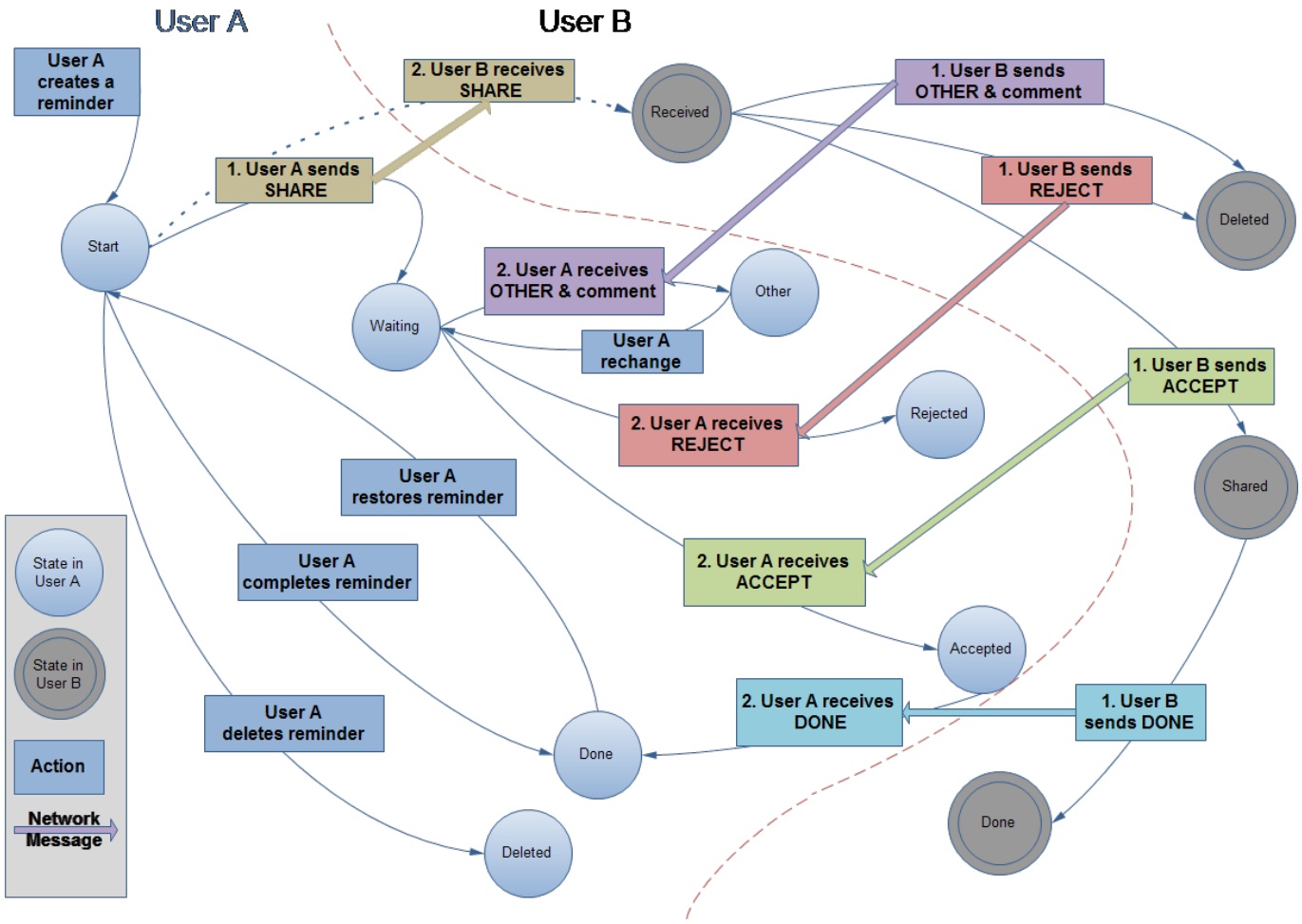


Figure 3.5: The states and the transitions

Chapter 4

Future Work - Extensions

4.1 Main Features

There are numerous minors or majors optimizations that can be proposed in order to improve the application. Nevertheless, implementing all these propositions requires significant time in writing the code and even more time in testing that the application is working properly. In this section, a number of extensions are suggested and proposed as future work.

- Implement multiple triggers per note: For example, we want to create the alert "buy milk when in supermarket", this note should be appeared to the user when he is close to a supermarket but only when the supermarket is open. This problem can be easily solved by connected the note to two trigger (trigger 1: when in supermarket, trigger 2: when time is between 17:00 to 20:00).
- Flexible triggers: Different triggers should be able to identify something in common, for example a place can be identified by coordinates, by the local wifi access point, by a static Bluetooth device etc. If the application can support this flexibility then it raises its credibility to correctly validate triggers.
- Proximity levels: In general proximity features should be accompanied with levels, for example the user might be informed when he is in the street far from your place and this is a drawback.
- More triggers interpretations: This application says when a trigger is valid notify a note. Nevertheless, in general, it can be very useful if the application supports other interpretations (like negative meaning, when the trigger is off).
- Time/alarm/importance support: The application is a to-do based application but lacks some basic features like reminding something to a specific time or giving importance to a note. Implementing this feature will greatly complete it.
- Sensor triggers: The mobile phone has a number of sensors (accelerometer, gyroscope etc). Finding triggers that are useful and that take advantage of the sensors would be a nice extension. A quite common idea is to implement an altimeter and sensing elevation (e.g. find in which floor is the user).
- Efficiency in energy consumption: The user is disturbed by having its battery drained, thus it should be special care to find a way[1] not to do that.
- GUI & Notification optimization : There are many optimizations for the GUI also, for example the routes are not presented in a readable way. Further, how the user will be informed is not also trivial.

[1] an example to that direction is by checking for triggers only when the screen goes from off to on. This might not be in general efficient but for the battery and for user experience might be perfect (example situation, if you know that you are in supermarket, you just want to see what to buy and you can do that by just looking the screen.)

There are many issues like how (with sound, vibration etc), what information will be given, how can deactivate them.

- Google task: There are many users that use Google task for organizing their tasks, thus having the application creatively integrated to that service will be useful.
- Actions instead of notes: In the application the action of an active trigger will be a note appearing on the screen. A promising alternative approach is to include many different actions like sending an SMS, make a call, twitte a message etc.

4.2 Extended Sharing

As far as the sharing mechanism is concerned, the possible extensions are also numerous.

- Choosing your friends: We can select friends from the Skype, from Facebook, from contact list. The application currently is restricted only to persons that are in the contact list and which have email.
- Multicast/broadcast messages: Sending and sharing messages to more than one person is always useful. In our application infrastructure, this can be easily done by fixing the python script in the server. In general, it is not an easy task.
- Creating/maintaining friend lists: Given that there are multicast messages, the application should be able to create groups (a family, close friends, basketball team ... etc) and should easily enable user choose with whom he will share a reminder/favour.
- Priority tasks: Notes/tasks should have different priority and the sharing mechanism should support also this. For example, we can implement the important alerts to be delivered instantly using SMS.
- Decentralized server/ overlay network: The sharing mechanism is centralized which is always a drawback when the application users are increased. Transferring the infrastructure in an Adhoc mode is too difficult and not efficient. One great extension in order to support a lot of users is to use a distributed system(overlay network) to provide the service of the server.
- Security: Without doubt, when an application is on the market, users expect, the messages that are sending over the network, not to be in plain text.
- More interaction: virtual money
- Reminders: Users might neglect a message, thus a mechanism to remind/resend messages will be useful.
- Flexible sharing: the user should be able to share a note, a trigger or an alert. For example, a user wants a note connected to the trigger "supermarket", he does not want to share the trigger (where is the supermarket).