École Polytechnique (X)
Research Internship

Documentation
LifeBand : The App

Author : Louis Faucon
Email : lpfaucon@gmail.com
Date : August 1, 2014

Introduction

LifeBand is an app ...

This document contains...

Contents

1 Architecture Overview

1.1 Activities

An `Activity` object implements the functions of the screen the user interacts with. It uses an xml file from `res/layout/` to describe its layout. The diagram (Fig. 1 shows how user can navigate from one to another.

- **MainActivity** is the main activity. You can see the two bands and access other activities.

- **ColorActivity** is the nudges sending activity. You can choose between ten colours and then click on SEND.

- **SettingsActivity** is for setting up user information. You just need to fill in name and login and then click on SAVE.

- **SettingsLocationActivity** is for setting up location information. You need to click on the buttons once when at home and once when at work.

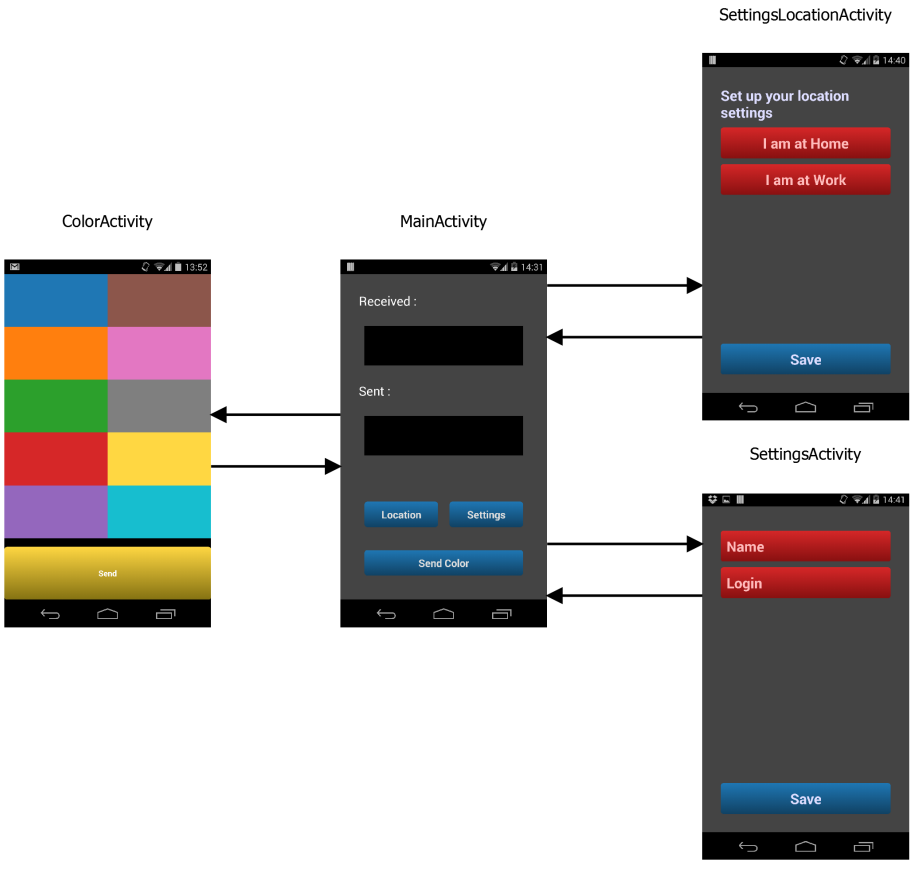

Figure 1: Activities architecture

1.2 Location updates

The following diagram (Fig. 2) shows the passive process of sending locations updates.

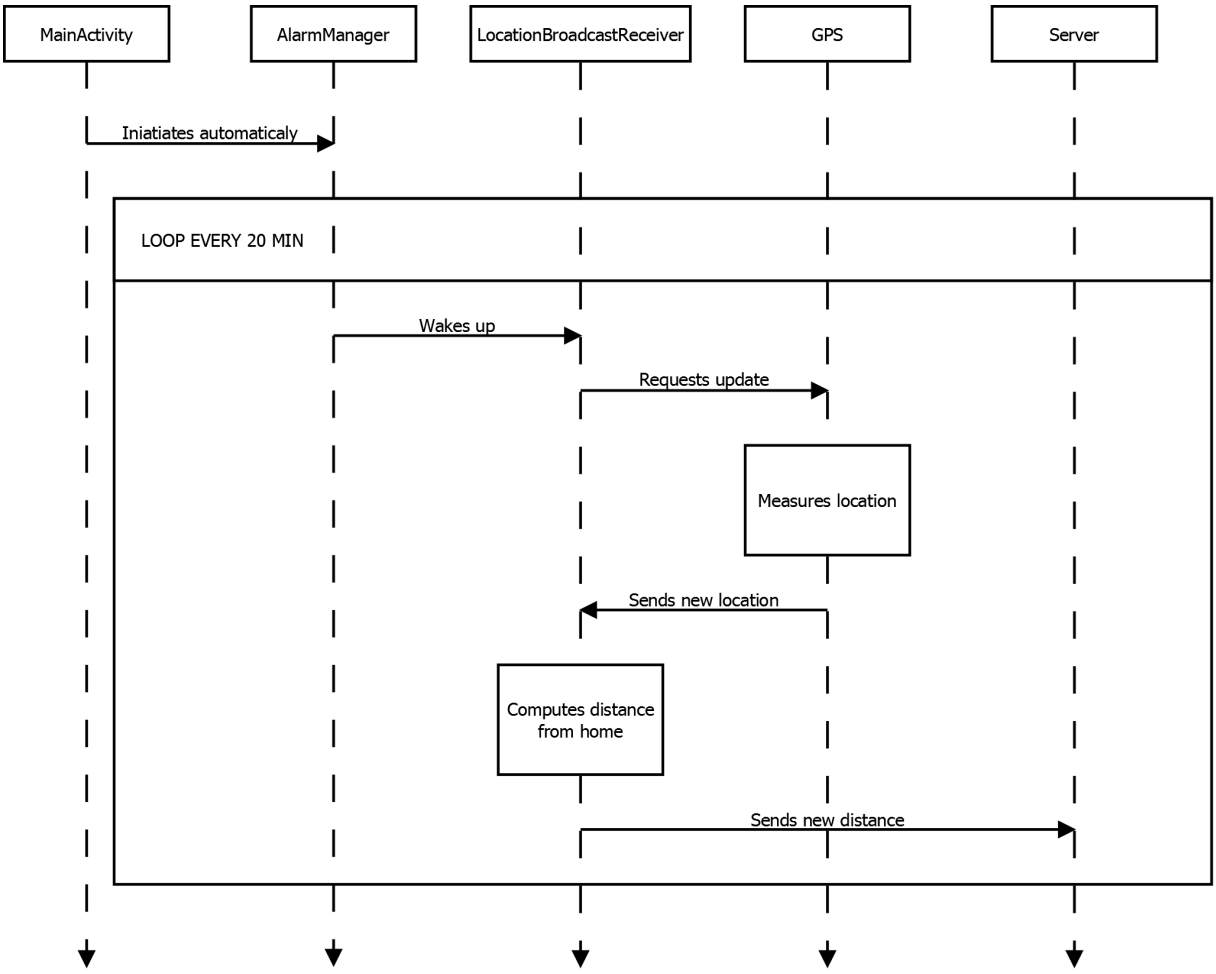

Figure 2: Location updates

1.3 Saving data

1.3.1 Preferences

The `Preferences` are good for saving small static amount of data for the application. The code for using `Preferences` is in the `class SettingsUtils`.

`String name`	User's name
`String login`	Login to link two partners
`String regid`	ID for google cloud messaging
`String bffid`	Partner's ID for google cloud messaging
`float home_lat`	Latitude of home place
`float home_lon`	Longitude of home place
`float work_lat`	Latitude of work place
`float work_lon`	Longitude of work place

1.3.2 SQL database

The `SQL database` is good for saving larger amount of data that often change.

1.4 Messages

1.4.1 Messages from the app

The app sends messages to the server using POST end GET requests. This is done by a static method of `LinkToServerManager`.

type	time	message	from	to
`colortoken`	`long`	an `int` for the color	`String regid`	`String bffid`
`locationtoken`	`long`	an `int` for the distance	`String regid`	`String bffid`

1.4.2 Messages from the server

The server sends messages to the phones using GOOGLE CLOUD MESSAGING. Every time a message is received from the app, the server send two messages, one to the sender with type `sent` and one to his partner with type `received`.

type	time	message
`colortokensent`	`long`	an `int` for the color
`colortokenreceived`	`long`	an `int` for the color
`locationtokensent`	`long`	an `int` for the distance
`locationtokenreceived`	`long`	an `int` for the distance

1.4.3 Special messages for registration

The first message is sent from the phone and contains the `name`, the `login` and the `regid`. Then the server sends two messages : the first one to acknowledge the registration and the second one containing the partner's id.

1.5 General

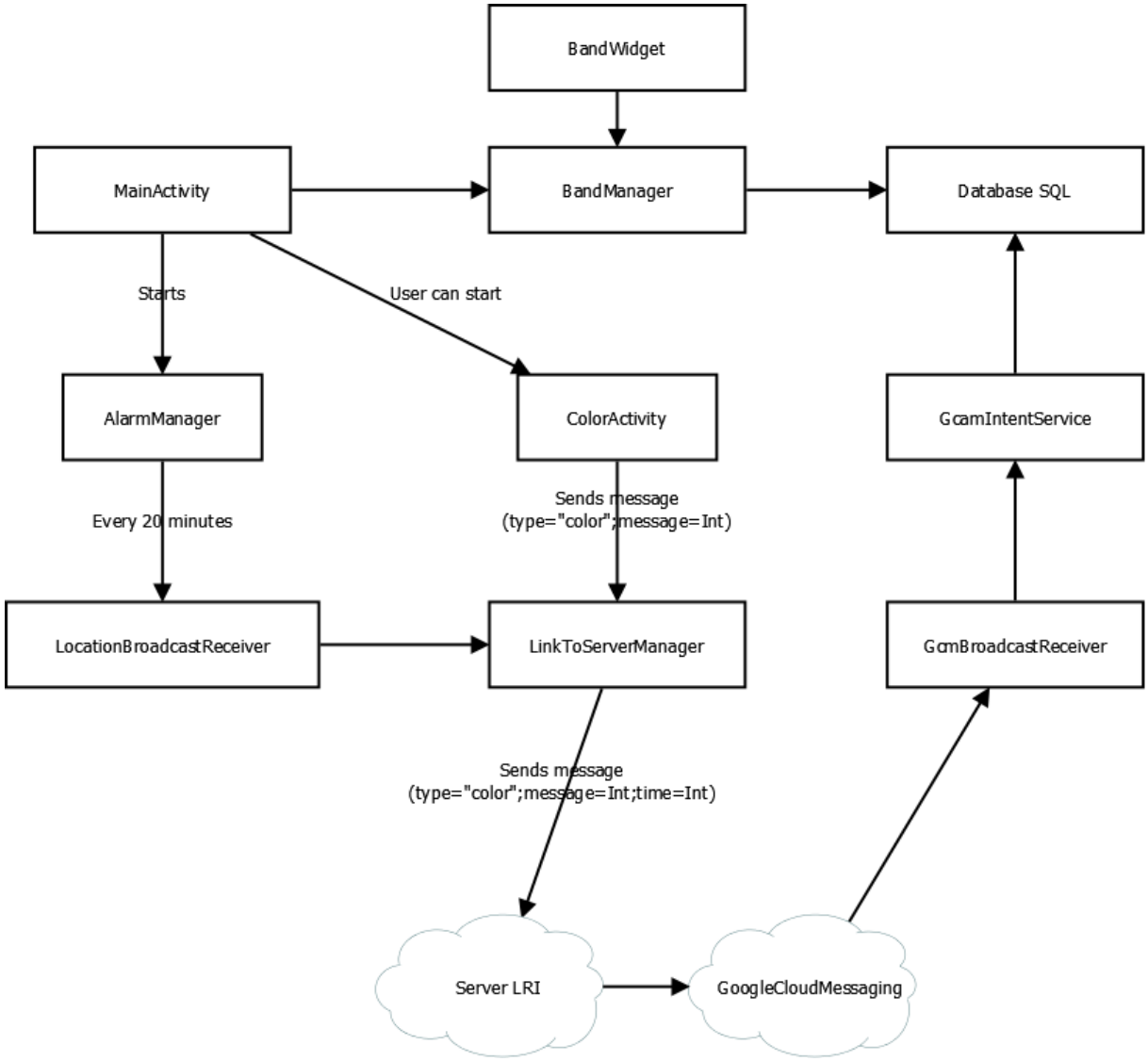

Figure 3: General architecture

2 Complete Documentation

2.1 lri.prototype.cosquare.app.bff

2.1.1 BandManager

The two methods `getBitmapReceived` and `getBitmapSent` give the band bitmap images (See Figure 4). The band can be configuered by changing the values of `LENGTH_TIMELINE`, `LENGTH_UNIT`, `WIDTH_BITMAP` and `HEIGHT_BITMAP`.

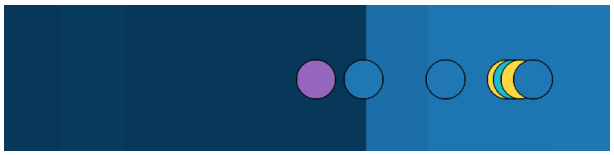

Figure 4: Example of the output bitmap.

```
private static final int LENGTH_TIMELINE = 10 * HOUR;
private static final int LENGTH_UNIT = HOUR;
private static final int WIDTH_BITMAP = 1024;
private static final int HEIGHT_BITMAP = 256;

public static Bitmap getBitmapReceived(Context context){...}
public static Bitmap getBitmapSent(Context context){...}
```

2.1.2 ColorActivity

`ColorActivity` uses the layout `res/layout/activity_color.xml` (See Figure 5). The activity shows 10 buttons whose color are described in the file `res/values/colors.xml`. The method `onClick` selects a color and the method `send` sends it.

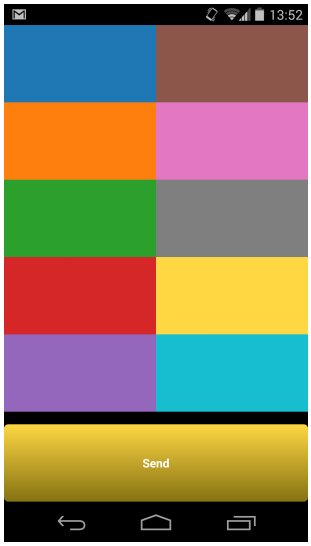

Figure 5: ColorActivity layout

```
public void onClick(View view){...}
public void send(View view){...}
```

2.1.3 ColorManager

This class has only one static method `getColor`. It takes two arguments an `int p` between 0 and 100 and a color and outputs a new color darker or lighter depending on `p`. The output color is the same as the input for `p=50`

Figure 6: Color output for purple and p=0 on the left to p=100 on the right

```
public static int getColor(int p,int color){...}
```

2.2 lri.prototype.cosquare.app.gcm

2.2.1 GcmBroadcastReceiver

This class is an always on receiver that listen to the Google Cloud Messaging. When a message is received the receiver just gives it to `GcmIntentService`.

2.2.2 GcmIntentService

This service handles the messages from the server. The messages contain three fields `type`, `time` and `message`.

2.2.3 GcmManager

This class is just used at the first launch of the app to register an ID with the Google Clood Messagging. The `SENDER_ID` is the ID of the server that sends messages. It is linked with the google account of the developer.

```
String SENDER_ID = "403235248512";
public void register(){...}
```

2.3 lri.prototype.cosquare.app.location

2.3.1 LocationBroadcastReceiver

`LocationBroadcastReceiver` is an always on receiver that receives messages every 20 minutes from an `AlarmManager` and wakes up the GPS to measure location and send it.

2.3.2 LocationUtils

`TIME_BETWEEN_UPDATES` is used to configured the `AlarmManager` that ask for location updates. The two other methods are just useful.

```
static public int TIME_BETWEEN_UPDATES = 20*MINUTE;
static public float distanceFromHomeToWork(Context context){...}
static public float distanceFromHome(Context context,Location location){...}
```

2.4 lri.prototype.cosquare.app.server

2.4.1 LinkToServerManager

This class has static methods to send messages to the server.

```
static final String SERVER_URL = "https://www.lri.fr/~faucon/gcmserver.php?todo=";

public static void sendRegistrationInfo(String name,String login,Context c){...}
public static void sendLocationToken(Context c, final int distance){...}
public static void sendColorToken(Context c, final int color){...}
```

2.5 lri.prototype.cosquare.app.settings

2.5.1 SettingsActivity

`SettingsActivity` uses the layout `res/layout/activity_settings.xml` (See Figure 7). The activity shows 2 `EditTexts` that must be filled in by a name and a login.

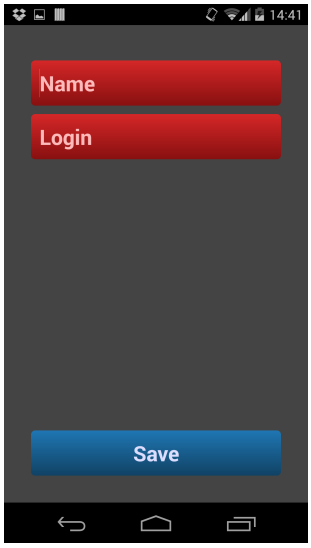

Figure 7: SettingsActivity layout

2.5.2 SettingsLocationActivity

`SettingsLocationActivity` uses the layout `res/layout/activity_settings.xml` (See Figure 8). The activity shows 2 `Buttons` that, once clicked, register in the preferences the present location as Work or Home.

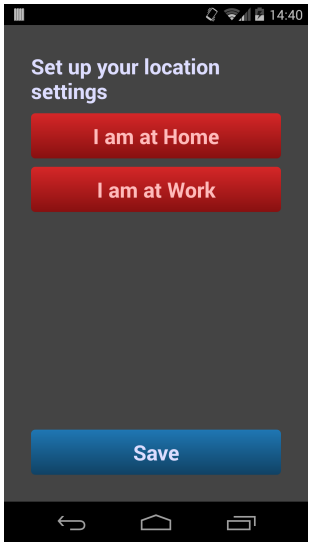

Figure 8: SettingsLocationActivity layout

2.5.3 SettingsLocationUtils

This class has useful static methods to record and retrieve information in the `Preferences` of the application.

```
//GETTERS
public static String getName(Context context){...}
public static String getLogin(Context context){...}
public static String getRegid(Context context){...}
public static String getBffid(Context context){...}
public static Location getHome(Context context){...}
public static Location getWork(Context context){...}

//SETTERS
public static void setUserInfo(String name,String login,Context context){...}
public static void setRegid(String regid,Context context){...}
public static void setBffid(String bffid,Context context){...}
public static void setHome(Context c){...}
public static void setWork(Context c){...}
```

2.6 lri.prototype.cosquare.app.sqldatabase

The `SQL databas` of the application contains four tables : `ColorTokenReceived`, `ColorTokenSent`, `LocationTokenReceived` and `LocationTokenSent`. Each of this class has a method `add` and a method `getAll`. These are used when a message is received or when the app needs to display the band.

`DatabaseUtils` contains easier to use methods to add data in the database.

`FeedReaderDbHelper` is used to create the tables.

2.7 lri.prototype.cosquare.app.widget

2.7.1 BandWidget

The class `BandWidget` describes the behaviour of the widget. Its properties are described in `res/xml/band_widget_info.xml` and its layout in `res/layout/widget_band.xml`.

2.8 MainActivity

`MainActivity` uses the layout `res/layout/activity_main.xml` (See Figure 9). The activity shows the 2 bands and 3 `Buttons` to access the other activities.

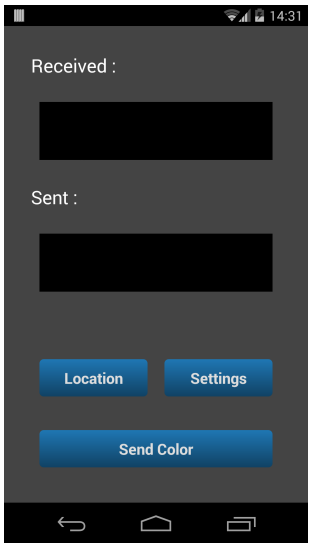

Figure 9: MainActivity layout

3 Server implementation

3.1 gcmserver.php

The server receives GET request with a parameter `todo` that can take the value `regid` for registration, `colortoken` to send a nudge or `locationtoken` to update location. Complementary arguments are sent using POST.

3.2 Database

name :	'redblue'
host :	'sql5.lri.fr'
user :	'lfaucon'
password :	'K4redblueSQL'
phpmyadmin :	'`http://web-int.lri.fr/phpmyadmin/index.php`'

The database contains two tables : `user` and `data`

Table user :

name	name
regid	id for Google cloud messaging
login	login to link the couple
bffid	bff's id for Google cloud messaging

Table data :

regid	id for Google cloud messaging
type	the type of the message : color or location
time	the time the message was sent (in milliseconds)
message	the content of the message

3.3 Data analysis

The file `data.php` is a script to analyse the content of the database.

Références

[1] http://developer.android.com/

[2] http://stackoverflow.com/