# WEB SYSTEM FOR INTERLOCUTOR'S AVAILABILITY

Ljupcho Antovski
Faculty of Computer science and Engineering

Skopje, Macedonia

Elena Janevska
Faculty of Computer science and Engineering
Skopje, Macedonia

ABSTRACT

It is inevitably noticeable that mobile technologies are increasingly occupying a central place in today's world. Currently, they form the fastest growing technologies and it can easily be concluded that a generation of smart mobile devices has started. Today, there are four billion active users of mobile phones and one billion users are familiar and are using smartphones. Also, for two to three years, we are expecting accession to the Internet from mobile devices to surpass joining the Internet via desktop devices [1]. This situation requires more work and research in this area, so that we can create new and useful mobile applications that hopefully in the future might become necessary in everyday situations.

However, developing mobile applications is not only creating the software that runs on any portable device with low power and with limited operating system's and platform's features. Creating a mobile application means to design and create entire systems with components of different nature that will exist individually and independently but will provide all the data and functions that a mobile application needs. Briefly, mobile phones cannot, and should not carry large processing or large data. Their duty is only in such communication with external systems.

This paper will present a system of multiple components, with special focus on a mobile application that will enable new, so far unknown function for all users of the Android operating system. Namely, it is a system which provides fast information about the availability of interlocutors. It is presented in an intuitive way on the device display, and the same information is taken from the user's calendars. All details will be presented in the following chapters.

## I. INTRODUCTION

### A. *Today's mobile applications*

All unique benefits that are brought by the area of mobile technologies, is a big boost for innovative and challenging achievements through daily creation of new applications for mobile devices. When we say unique benefits, we think of constant connection to all users wherever they are, using the opportunity to access their current location, having the list of the people in their lives through the address book, having the messages, emails etc. Using this data, if the user approves, we can create a lot more useful and not standard applications.

On the other hand, the daily creation of new applications is not a creation of this type. The "Android Market" consists of large -scale applications that it offers to its users. Indeed, the figure is impressive, but from our experience, most of the applications are without any particular useful functionality and applications that will fail the users' expectations.

Coming up with unique and useful idea is the hardest part, and this issue may therefore result in this market's products.

What the world wants and expects, is an invention and application that would be very useful for all regular users of mobile phones.

### B. *Integration of web system and mobile applications*

Besides the need for larger processing by mobile devices, storing larger data is an issue too. So, storing and sharing data is of particular interest in this field also.

Android OS provides more methods to save the mobile applications data. You can store primitive data type key - value, private data on phone memory or external memory for further sharing, storing data in a relational database - SQLite and most important section where we want to proceed: storing data on the web through constant device connection to the internet.

Logically, in these systems, it is necessary to have a database that is installed on some server. Best practice when communicating mobile application and database is creating API, or web service that acts as an intermediary module between them (Figure 1).

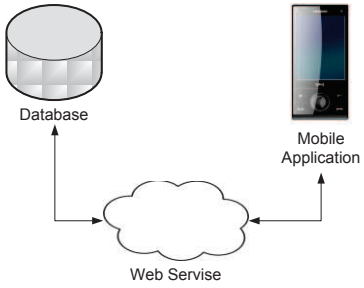


Figure 1: Integrated system infrastructure

Next step while building this system is selecting the appropriate platforms for each component. There are certainly many arguments to be for or against the choice of the most appropriate platforms. The presented system consists of Android application, MS SQL database and .Net web services.

## II. DEFINING THE SYSTEM

The system for intraocular availability consists of a database with all register users, which are all users that installed the application on their mobile phones. When one user calls a person, there will be one of these scenarios:

1. The called user has the application installed
    a. The application is running in background
        i. The user is busy
        ii. The user is free
2. The called user does not have the application installed.

This data is presented to the caller in order to get more information about the callee's availability.

Today one the most commonly used systems for managing private time is Google Calendar. The simple interface and easy calendar synchronization on multiple devices simultaneously allows most people opt for using it. In our system, Google Calendar events will serve us as information about the current availability of the users.

Communication with user's calendars may not be the key problem and the main functionality. Important parts of the work of the mobile application are the two interlocutor's screens. Special importance is given to the caller, because he needs to get information about when the called user will be free. By this, more questions set in devising the system and its operation: how smartphones will interact with each other, how one would know when to take the information from your calendar and send it to another phone etc. In the next chapters we will provide detailed explanation for the system specific issues and functionalities.

## III. Integration of Google Calendar

Android devices usually come with a set of pre-installed applications like an e-mail client, client management for SMS, calendar and contacts list, WebKit- based web browser, gallery and more. Because it's Android, often there are existing Google applications on the phone: Google Maps as Maps app, Gmail as e-mail client, Google Talk like IM client and in our case Google Calendar as the official calendar for the use of the devices.

These applications, as well as the calendar, are doing their own functions and store data directly on the device. Calendar synchronization is also provided as a function of the application. What's in our interest is to find the appropriate API and to communicate with the calendar by our application. Google provides a common API for working with data from calendar online in the form of web services which of course can be accessed through our application of the standard way as accessing any web service [3]. But here is the key question raised: why should we care about the connection, response time, battery life, authentication, etc., when we already have the data on our phone.

Working with Google Calendar web services was the practice for most application developers working with Google Calendar. However, the standing version: Ice Cream Sandwich 4.0, in October 2011, came with a new public API for working with data from calendar [4]. While this was indeed a great help for building powerful applications, we were encountering a problem that occurs with any new technology or skill to be mastered : the documentation was not yet carved, rare were the answers to the problems on the forums, there were few examples that to find and use this API.

In order to access the user's calendar, you must add permission under the manifest of our application. Because at this point we want only to perform reading, we need a permission of this type:

```
<uses-permission
android:name="android.permission.READ_CALENDAR"/>
```

For our application is sufficient to ensure that you can access to any listed event, its name, the time it begins and the time it ends. Later you can easily manipulate the data, or select only the current event, if any. For this purpose we will show code that reads directly from the database and is pulling calendar event name, start and end of the event. This code can then be part of any activity as a core component of Android (Activity) or as a part of any other component in the application.

```
// Creating the database cursor

Cursor mCursor = null;

final String[] COLS = new
String[]{CalendarContract.Events.DTSTART,
CalendarContract.Events.DTEND,
CalendarContract.Events.TITLE};

mCursor =
getContentResolver().query(CalendarContract.Events.CO
NTENT_URI, COLS, null, null, null);

mCursor.moveToFirst();



// Defining stra, finish and name variables for an event

long start = 0;

long finish = 0;

String name = "";

while(!mCursor.isAfterLast())

{

    start = Long.parseLong(mCursor.getString(0));

    finish = Long.parseLong(mCursor.getString(1));

    name = mCursor.getString(1);

    mCursor.moveToNext();

}
```

## IV. CHANGING USER STATUSES

Once we read the current event for a user, the next thing is to timely proceed that information to the database server . One way to do it immediately is to get the event of the user who is calling on a short specific time intervals, to call the function of the web service and to change the field for current event. This functionality should be running constantly in the background of the phone. If any other activity on the user's mobile device, it should not be obstructed. The solution to this concept is provided by one of the basic components when developing Android applications, and it is an Android service [7] .

Android services are intended to ensure that operations are performed in the background for a long time without any interaction with the user. Services have no user interface and they can start and stop the application as required by other components. Services receive much higher priority then the activities and they are unlikely to be killed by the system. To create the service you need to make a new class that will inherit from the class Service. If necessary, it could be amended or supplemented with the functions: onCreate(), onStart() and onDestroy () [8] .

For our solution, when we start the service, we need to know the user's identification and call the function onCreate() in order to write a code that will read data from the calendar and send it to the web service. It is acceptable for this action to be repeated every ten minutes. This means that every ten minutes we'll be reading the calendar and sending the latest data. Such counting is performed using a timer and Java classes. The following code shows the concept of this work:

```
@Override
public void onCreate() {
    super.onCreate();
    Timer timer = new Timer();
    timer.scheduleAtFixedRate(new TimerTask() {
    public void run()
     {
        // some code for reading the calendar
        // some code for calling the web service
      }, 1000, 600000);
  }
```

## V. INCOMING AND OUTGOING CALLS

Perhaps the most important part is the moment when establishing a call with our interlocutor. Here, we will first explain the calls that we create, the case when a user establishes a call. At this point it is necessary to write to the call screen following information: "The user you are looking for is at *"the name of the meeting"*. In case you do not get an answer, try calling at: *"look at the time when the meeting ends"*".

In order for our application to always responds when calling a callee, we must implement the so-called Broadcast Receiver as a third component. Its purpose will be only listening for outgoing calls [9]. Now we will give a small description of two key components for this issue: Intent and Broadcast Receiver. Intent is used as a mechanism for sending messages between components in the application itself, and also between other different applications. They can launch a new service or a new activity under the name of the class or the name of the activity [10] .

In our application we will use intent for each of these needs, but the part that needs to be clarified here is that you can use intent to send messages around the system. Applications may have registered Broadcast Receivers that have the task of receiving already sent Broadcast Intents[11]. When a phone number is called, this intent will reference our application ad will show us the number.

This component that is waiting for an intent is implemented in the framework of the Android service. Why? Because the service is activated when the user runs application usage and before that will so not need call listening. Code for this part looks like this:

```
BroadcastReceiver br = new BroadcastReceiver(){
@Override
public void onReceive
(final Context context, final Intent intent)
{
if(intent.getAction().equals("android.intent.action.NEW
_OUTGOING_CALL")) {
    Timer timer = new Timer();
    timer.schedule(new TimerTask() {
    public void run() {
    final String tel;
    tel =
intent.getExtras().getString("android.intent.extra.PHON
E_NUMBER");
Intent i = new Intent(context,
OutgoingCallActivity.class);
i.putExtra("tel",tel);
i.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
i.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
    context.startActivity(i);}}, 5000);}} };
this.registerReceiver(br,new
IntentFilter("android.intent.action.NEW_OUTGOING_CA
LL"));
```

When it comes to calls receiving, the problem is solved another way. Again we use Broadcast Receiver, but working with the component Telephony Manager also. With its help we have access to the phone's API and we can get a variety of information such as: phone features, network, SIM details etc. We will work with the phone states [12] [13]. In our interest are three situations:

• **Ringing** - a condition in which is the mobile phone only while rings;

• **Off Hook** - state when the conversation is done no matter if someone called us or we have called him. It is important that the conversation is on-going. When we wait for a response, the call's found in this condition too.

• **Idle** - condition when the phone is not active in terms of calls. After this condition, Ringing and Off Hook may occur.

To get the time when receiving a call, we should find ourselves in the "Ringing" situation. It detecting was easy, but there was a problem at the moment when a call ends and getting the duration of this activity. In fact, with its completion we need to stop the information we display. So, after "Ringing" the phones state goes into "Off Hook" and we should remove our information off the screen. But the state "Off Hook" can be activated in another way as mentioned above and we may request termination of information that we never offered (because the condition was not caused by the ending of "Ringing"). Therefore, as a solution we set static variable that will always keep a record of whether the phone was ringing before.

## VI. Displaying Screen Messages

We have already concluded that the by monitoring changes in conditions and identifying the condition of interest, we show a new activity that carries information to the call screen. Each activity represents a single screen application that is presentend to the user. The larger applications you create , the more activities you will have. Typically, this application must include at least one primary user interface for the current screen that is defined in the appropriate xml file. The logic is built into the framework of the activities which interact easily with components of the xml file (user interface files). To create a new activity in the application, create a new class that inherits from the predefined class system - Activity class [14].

All these activities are managed as a stack of activities . When a new activity is started by running, system adds it on top of the stack and it becomes the main activity. Activities can have several conditions, including:

• The activity is in the foreground of the screen and running;

• The activity has lost focus but is still visible, which means that an invisible activity is above it on the screen . This means that the activity is paused. This condition will be particularly important when building screen calls;

• The activity is stopped. Another activity is started, but the old activity still exists in the background with saved state information. It is no longer visible to the user and will soon be killed by the system if needed for memory.

What we should do is calling our activity in the foreground during a call. As part of this activity we call a web service and receive results which we show on the simplest TextView which is part of the UI activity. Proper xml file for this activity is as follows:

```
<TextView
xmlns:android="http://schemas.android.com/apk/res/andr
oid"
android:id="@+id/text"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:gravity="center_vertical|center_horizontal"
android:windowBackground="@android:color/transparent
"
android:windowIsTranslucent="true"
android:windowAnimationStyle="@android:style/Animatio
n.Translucent"android:textColor="@android:color/white">
</TextView>
```

## VII. Free Callee Functionalities

The end product is an Android mobile application Free Callee whose functionalities are presented within this section. The app is in English and after installation on the device, it is requiring the user to register the mobile device and to provide a regular internet access.

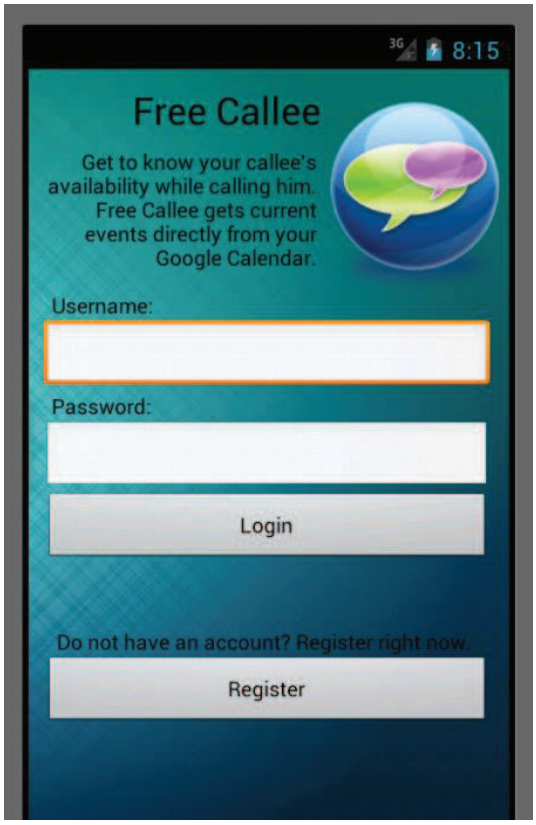


Figure 2 - Free Callee welcome screen

Once the user is successfully authenticated, its name appears on the home screen application that has none particular function except to give information to the user that the operation of the application is activated (Figure 3). It means that at that point calling services keep working in the background. The next thing the user needs to do is to click the button to continue the operation of the application. Then the application will lead you to the home screen and he can resume normal use of his mobile device. The second button on the screen fills possibility to exclude the operation of this application.

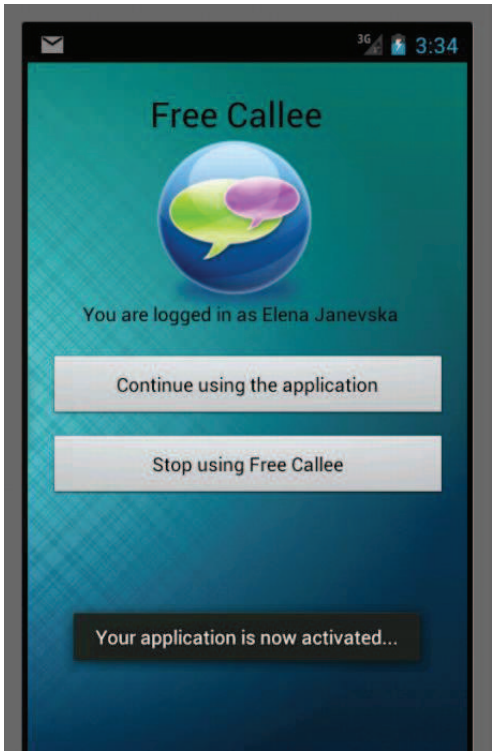


Figure 3 - Free Callee activation screen

The information that should presented when calling somebody, references a number of activities depending on the user who owns the number you are calling. The user may be a registered user of the mobile application, but also does not have to. In the first case, if the user uses the application, and he is regularly changing field where we keep his involvement in our system, the display of the caller will write standard availability information as it is shown in Figure 4.

If the user is free, which means that it is read from the calendar that this time there is no event, we will write the information that the user who is called is free at the moment The other two cases can occur are: the person has never used the application and is not a registered user or he just not currently use the application.

In a case of an incoming call, again we have four different scenarios. The main difference is that, if the user who is calling you is busy, you get to see only where he is, but not the time he will be free.

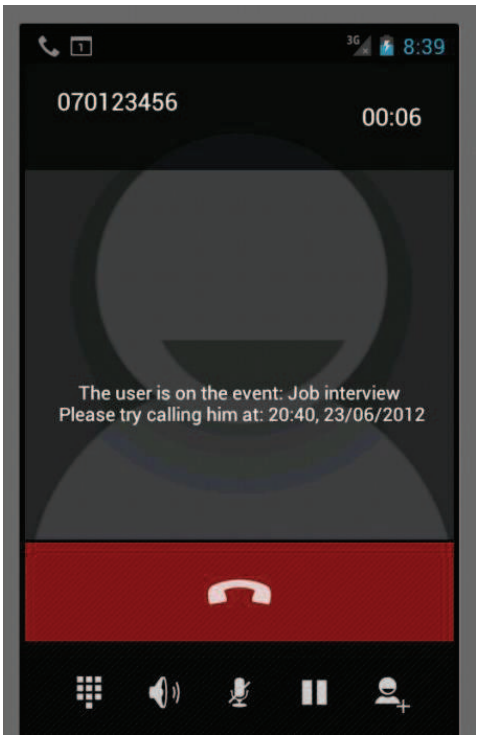


Figure 4 - Free Callee busy user screen

## VIII. APPLICATION BENEFITS

The idea for such a system came from thinking about the possible benefits of mobile options which were not yet enabled and used. In our case the main benefit is fast and timely information about availability of the callee when necessary, i.e. when establishing the call. Other benefits are:

- Another reason for regular planning their time with the help of Google Calendar;
- Obtain information on current events for the needed user;
- Obtaining information about the time when the user will be available again.
- A kind of modern answering machine that always gives precise information about why the called number may not respond;
- Possibility to learn and user location via the event name that currently attends;
- Running of the application in the background while the user uses the phone for any of its needs without special settings and input data into the application;

- Especially useful application within employees in a company, but also in everyday life as a way to get information on availability of friends.

## IX. Conclusion

Finding a solution to this prospect was a big challenge. Finishing the whole system brings new efficiency and something new that could be offered to the market and will open a new horizon for future work. Using the mobile application without problem can be limited to a small circle of people that is important to share their availability, but you can take it globally and to create a network of users who's information will be a daily service.

For future work we are considerin other relevant information that could be taken by another application or calculated by some user data. The new information could pose addition to the work already created of be a reason reason to create a whole new system and Android application.

## References

[1] Digitalbuzzblog. „ Infographic: Mobile Statistics, Stats & Facts 2011“ http://www.digitalbuzzblog.com
[2] API Google Calendar: https://developers.google.com
[3] Tim Bray. „New Public APIs in ICS“
[4] Reto Meier. „Databases and contetnt providers - Using the calendar content provider “ Professional Android Application Development, Updated for Android 4. Indianapolis: John Wiley & Sons, Inc, 2012, 325-330
[5] Android developers. „Calendar Provider“, API Guides.
[6] Reto Meier. „Working in the Background “ in Professional Android 4 Application Development, Updated for Android 4. Indianapolis: John Wiley & Sons, Inc, 2012, 331-345
[7] Android developers. „Android Services“, API Guides.: http://developer.android.com, 12.07.2012.
[8] Satua Komatineti и Dave MacLean. „Broadcast recievers and long-running services“ in Pro Android 4, Android 4 platform SDK techniques for developinf smartphone and tablet app . New York: Apress, 2012, 599-641
[9] Satua Komatineti и Dave MacLean. „Understanding Intents“ in Pro Android 4, Android 4 platform SDK techniques for developinf smartphone and tablet app . New York: Apress, 2012, 113-135
[10] Satua Komatineti и Dave MacLean. „Broadcast recievers and long-running services“ in Pro Android 4, Android 4 platform SDK techniques for developinf smartphone and tablet app . New York: Apress, 2012, 599-641
[11] Reto Meier. „ Accessing Telephony Properties and Phone State “ in Professional Android 4 Application Development, Updated for Android 4. Indianapolis: John Wiley & Sons, Inc, 2012, 705-707
[12] Satua Komatineti и Dave MacLean. „Using the telephony APIs“ in Pro Android 4, Android 4 platform SDK techniques for developinf smartphone and tablet app . New York: Apress, 2012, 599-641
[13] Android developers. „Activities“ http://developer.android.com, 12.07.2012.
[14] Grasshopper.iics. „Calling ASP.NET Webservice from an Android application, the simplest way“. http://www.codeproject.com/, 22.12.2011.
[15] Chuck Hudson. „Incorporating Web Serves in Mobile Applications”: http://www.slideshare.net, 03.04.2009.