Android project proposals

Luca Bedogni Marco Di Felice ({lbedogni,difelice}@cs.unibo.it)

May 2, 2014

Introduction

In this document, we describe four possible projects for the exam of the ”Laboratorio di applicazioni mobili” course. Each student can choose a project from the set, or suggest something else based on his/her personal interests. In this latter case, project proposals should be submitted via e-mail to Luca Bedogni (lbedogni@cs.unibo.it) and/or to Marco Di Felice (difelice@cs.unibo.it), with a brief description of the application goals, contents and requirements.

The following project descriptions should be considered as hints. Students are strongly encouraged to expand the tracks, adding new features to the applications, and/or further customizing the contents.

Projects submission

Projects must be submitted through email to `lam-projects@cs.unibo.it`, including all the code, a technical report, and eventual directives required to compile and execute the project. Best projects will be asked to upload on the Wilma-Lab wiki (http://www.cs.unibo.it/projects/wilma-lab/wiki/doku.php?id=start).

1 Budget Tracking

This project proposal consists in the deployment of an Android application that enables budget tracking, and provides facilities to keep personal finances in order. In particular, the application should:

- Allow the tracking of everyday expenses
- Manage planned/periodic expenses
- Allow the browsing of expenses by date, and the generation of reports
- Compute and display useful statistics

In the following, each features is further discussed and analysed in detail.

1.1 Allow the tracking of everyday expenses

The application should allow a user adding information about each current expense, such as: current date, amount, description, category, etc. All the information must be stored locally on an internal database. Moreover, it must be given the possibility to save the location where a given expense has been performed (e.g. shop's location). Optional element:

- Save a picture of an item, acquired through the photocamera.

1.2 Manage and remind planned/periodic expenses

The application should allow a user adding information about periodic expenses (e.g. a loan payment) and about planned expenses in the future (e.g. a phone bill payment). The budget must be updated at the payment date, and periodic reminders should be shown 1 and 2 days before (e.g. through alert dialogs when the application starts, or through notifications in case the application can run in background mode).

1.3 Visualize and browse expenses by date

The application show allow a user visualing and browsing the budget and the list of expenses performed day by day, in the last week or in the last month. It must be given the possibility to save the generated report into a PDF file. In addition, it must be possible to display the locations on the Google Maps.

1.4 Provide weekly and monthly statistics

The application should foresee the possibility to compute and visualize useful statistics about weekly and monthly expenses performed by a user (e.g. total expenses for each category). Charts can be generated to visualize data.

2 Mobile Latex editor

LaTeX is a language to write documents, particularly used for scientific publications and reports. Via a LaTeX compiler, it is possible to generate PDF files from a LaTeX source file. However, the LaTeX compiler requires considerable resources in terms of memory and computation resources, and thus it might not be supported by a mobile device.

The project consist in developing a LaTeX editor for the Android platform, that is able to compile the document on a remote server, and then open the corresponding PDF on the device in use. More in detail, the app should:

- Provide a text editor for managing text files.
- Support the LaTeX syntax.
- Foresee the possibility to compile the LaTeX source code on an external server, and then download the resulting PDF.

2.1 Provide a text editor for managing text files

The application should provide classical functionalities of a text editor, i.e. the possibility to open, edit, save, close a file.

2.2 Support the LaTeX syntax

The application should be able to help the user in writing LaTeX code. Syntax highlighting should be provided. Moreover, the applications should provide facilities to insert LaTeX code (e.g. helping the user to insert math symbols).

2.3 Remote compiling

When the user clicks on the Compile button, the file should be automatically transfered to a remote server, where a PDF compiler is working. Once the PDF has been generated, it should be transferred back to the mobile applications. In this case, an Intent should be generated to open the PDF file. Optional elements:

- Handle the compiler logs (e.g transfer them back to the mobile applications)
- Manage .tex files with images. In this case, the app should be able to compress the folder in which the source files are (i.e. tex files and images), and send it to a remote server. At server-side, the compressed archive should be extracted, compiled, and the corresponding PDF should be returned back to the device. After that, the extracted directory on the remote server should be deleted.

3 RSS client

This project proposal consists in the deployment of an RSS[1] client for the Android mobile platform. That client should be able to:

- Add new RSS feeds based on user inputs.
- React on Intents launched by external applications.
- Display the RSS feeds.
- Send a post link to social networks
- Configure some parameters

In the following, each feature is further discussed and analyzed in detail.

3.1 Add new RSS feeds based on user inputs

The application should be able to add new RSS feeds directly typed in by the user. More in detail, the user could enter a web URL corresponding to an RSS feed. After typing it, a service should check the validity of the URL (e.g. `url not found`, `url found but not an RSS`, `RSS found and so on`) and add it to the list of current RSS feeds.

3.2 React on Intents launched by external applications

The application should be able to add new RSS feeds based on Intents launched by other applications. That means that in the `AndroidManifest.xml` the application should declare to be able to handle Intents with action `VIEW` and with data corresponding to an internet URL. Of course, every feed added this way should pass the validity check described in the previous paragraph.

3.3 Display the RSS feeds

The application, in the main Activity, should display the list of all feeds currently registered in the device, providing the number of unread items for each feed too. Clicking on a feed will bring the user to the list of post for that selected feed. By long-clicking on a post, the user should be able to mark the post as unread, if he/she has already read it, or as read, if he/she hasn't already read it, or mark it as favourite. The application should also provide a combined post list, where each post is displayed regardless of the feed which it belongs to, while the action depending on the long click are the same as before. Downloaded posts must be locally saved into an internal database for off-line reading.

3.4 Send a post link to social networks

The application should provide, via a long click listener or something similar, an action to post a link to the post in social networks such as twitter or facebook. Additional text can be (eventually) inserted by the user.

[1]http://www.rssboard.org/rss-specification

3.5 Configure parameters

The application must provide a set of parameters to be configured, such as:

- Update frequency
- Notification types for new posts

The application should use Android's `SharedPreferences` system, saving user preferences across session and reconfiguring application's services based on those preferences.

4 IFTTT Engine

This project proposal consists in the deployment of an Android application that performs automatic actions and triggers when specific conditions are met, on the basis of the popular If-this-then-that programming (IFTTT) paradigm. In particular, the application should:

- Recognize a set of pre-defined contexts (e.g. attending a meeting)
- Capture a set of pre-defined events (e.g. a phone call is incoming)
- Define a list of possible actions (e.g. turn off the ring tones)
- Build an engine that allows a user to specify the default action to perform when a context is recognized and/or an event is generated (e.g. turn off the ring tones when there is a phone call incoming and I'm attending a meeting).

In the following, each features is further discussed and analysed in detail.

4.1 Context Recognition

The application must be able to recognize a predefined set of contexts that characterize the current user's operations. Basic context can be defined in the basis of: (i) temporal information (e.g. any time between 8-9am) and/or (ii) spatial information (e.g. I'm close to the DISI department) and/or (iii) mobility information (e.g. I'm moving at speed higher than 10Km/h). The user must be given the possibility to provide a context's name, and the information that characterize the context. Optionally:

- More fine-grained contexts can be recognized considering the information provided by embedded sensors (e.g. accelerometers), the radio interfaces (e.g. Wifi) or the photocamera/microphone.

4.2 Event Recognition

The application must be able to capture and recognize a list of external events that might occurr on the smartphone. Examples of events: phone call, SMS reception, open WiFi network detected, Bluetooth connection detected, etc

4.3 Action Definition

The application must provide a list of pre-defined actions and notifications that can be executed. Three major categories of actions must be considered:

- Setting manipulation:. these actions modify the smartphone setting (e.g. turn on the vibrations).
- Notifications: these actions recall the user's attention through status-bar notifications.
- Social network: these actions involve operations on social media (e.g. publish a state update on Facebook)

Additional and more specialized categories of actions can be defined.

4.4 IFTTT Rules definition

The application must allow a user specifiying IFTTT rules. An IFTTT rule is composed by an action, a context (optional) and an event (optional). When the context/event is recognized, the corresponding action must be performed. The application must be able to run in background in order to continously monitor the current context and capture incoming events. Optional elements:

- Context/event can be combined through boolean operators AND, OR, NOT.
- Multiple actions can be defined for the same context/event.