Android project proposals

Luca Bedogni (lbedogni@cs.unibo.it)

April 26, 2017

Introduction

In this document, we describe five 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) , 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.

1 Wireless Hot Spot finder

This project proposal consists in an application that should discover wireless access points at the current user's location, save their data on an internal database and display them on Map. In particular, the application should:

- Discover access points in the current range.
- Visualize discovered access points on a map.
- Color the areas on which wifi coverage might be present

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

1.1 Find access points in range

The application should listen to the 802.11 channels to spot the presence of wifi networks in the current user's range, and save them on a internal database together with:

- Location where the access point has been found
- Signal strength
- Security type (`WEP, WPA, Open` ecc.)
- ... (any further information available on the access point)

1.2 Plot found access points on a map

The application should be able to visualize the discovered access points on a map, identifying them with different icons and colors. In particular, open access points should be displayed as `green`, secured access points should be displayed as `red`. Moreover, developers should foresee different visualization methods (e.g. visualize all the access points, visualize only the access point in a given range, visualize only the open access points, etc), that might be selected by the end-user.

1.3 Color the areas on which wifi coverage might be present

The application, based on its internal database, should color the map where radio coverage might be present. To implement this functionality, triangulation algorithms could be used to determine the coverage area of each access point.

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.

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

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 transferred 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 Environmental monitoring with mobile crowd-sensing

This project proposal consists in the development of a mobile Android application able to run in background, report different measurements coming from all the smartphone's environmental sensors, report connectivity indexes of the network interfaces and communicate them to a central server. Subsequently, for each measurement, the application receives as a reply a temporal and a spatial constraint determining where and when the next update is needed, based on a set of rules. In the following sections each feature is discussed in more detail.

3.1 Measurements

The application should detect which sensors are available on the smartphone (temperature (id:13), humidity (id:12), pressure (id:6), sound (id:100), light (id:5)) and start during the boot of the system. In the same way, it should detect which wireless interfaces are hosted on the phone (WiFi (id:101), cellular (id:102)) and periodically detect their parameters. Afterwards, it should run in background and send the sensed data to the central server, hosted on 130.136.37.15, querying the resource `sensor_send`. The layer 7 protocol used is CoAP, a lightweight REST-based application protocol for constrained device. The data of all calls is structured in JSON records and its format is indicated in a WADL resource tree, available at `https://www.dropbox.com/s/swxudf9muu0a012/wadl.xml?dl=0`

- In the case of WiFi, the field `value` should contain BSSID, SSID and RSSI separated by commas.
- In the case of cellular, the field value should contain technology, RSSI, operator and throughput separated by commas, where the throughput is measured in bps and should be measured by the client sending a burst to the `calc_throughput` resource and measuring it over time.

3.2 Rules

Anytime a sensing participant sends a measurement, if well-formed, a configuration is returned, containing a timer and a zone, identified by a center and a radius, inside which the smartphone is supposed to be located while sending. Such configuration is obtained using a set of rules that characterize each sensor, thus it is released on a per-sensor basis. Upon the reception of such configuration, each participant is supposed to send the next measurement when the timer expires OR when the user falls outside the delimited area. The presence of the user within the delimited area should be implemented using the geo-fencing facility.

3.3 Stakeholders and Campaigns

In our scenario, different stakeholders can be interested in the data provided by the users. They can upload new rules and promote new campaigns, often offering stricter spatial and temporal constraints. The application should offer

to the user to subscribe to one or more stakeholders' campaign through the `update_subscriptions` call and to visualize the list of all the active stakeholders together with the user's subscription through the `get_subscriptions` call.

For major details please refer to `http://ieeexplore.ieee.org/abstract/document/7845471/`

4 Smartphone Sensors Recording

Modern smartphones carry several sensors such as accelerometer, gyroscope, magnetometer and others. This project consists in making a mobile app that is capable of recording the data coming from these sensors on a local .csv file, with different parameters that can be configured by the user.

In particular, the application should:

- Start at the system boot. Plus: check from time to time that it is active, and if not, activate it.
- Present the user with a list of possible sensors, and make the user able to select which ones she/he wants to record. Plus: add also sound level, gps information, call information and others.
- Enable the user to select the frequency at which she/he wants to record the data.
- Record the data in background, even if the user closes the app, on a .csv file. Plus: implement from the app a system to share the recordings.
- From time to time, popup a notification asking to the user what she/he is currently doing, from a list of possible actions.

In the following sections each feature is discussed in more detail.

4.1 Start at the system boot

The application could be either started manually by the user, or start in the background after the boot is completed. Recordings are then saved on the smartphone memory.

As a further step, the application should also check from time to time that the service which records the activities is still active. If it has been stopped/crashed/destroyed, it should be started again.

4.2 Sensor list

The application, when started manually, should present the user with a list of the sensors available in the smartphone, and enable the user to select which sensors she/he wants to record. The minimum requirements are satisfied by including all the sensors available through the Sensor Manager. Additional points can be obtained by adding data such as the noise level, gps speed, call information and such.

4.3 Frequency

Recording everyday activities for the whole day can rapidly deplete the smartphone battery or/and the smartphone storage. Hence, the application should enable the user to select the frequency at which she/he wants to record the data. This can be put in Hz, so the user who selects to register at K Hz will record at most K measurements from each sensor every second.

If the user puts 0 in the frequency preference, it will disable the application, which should not start the service nor record anything in the .csv file.

4.4 Record the data

The main component of the application has to record the data in the background. The data has to be saved in a .csv file in the smartphone memory. The format of the recording has to be the following:

`TIMESTAMP,S1-V1, S1-V2, ... , SN-VN, LABEL`

where `SN-VN` represent the value of the n-th value of the n-th sensor. Each row should represent a separate measurement. Therefore, the columns representing sensors which have not an available value ave to be marked as NA for that specific row.

To gather additional points, it is possible to implement a sharing mechanism for the recordings. From the application, the user can decide to share the recordings available on the smartphone via mail, instant messaging and such. The user should also have the possibility to cancel previous measurements.

4.5 Notification

From time to time, the application should ask to the user what she/he is currently doing, from a list of possible actions, which are:

- Walking
- Running
- Driving
- Being on a Bus
- Bike
- Still
- Being on a Train

The user should then select what she/he is doing at the moment, and the application will label that row with the action of the user, with the syntax presented in the previous section.

5 Bologna OpenData

The Bologna Municipality has made available several data in an open format, through the website `http://dati.comune.bologna.it/`. These include more than 800 different dataset concerning the mobility, municipal buildings, and many more.

The goal of the project is to leverage on the data provided to offer different kind of services to the user. The application should be able to update the data by using the permalink API offered by the municipality, the details of which are published on the specific dataset to be used.

This project does not have specific guidelines, as everything depends on the chosen dataset. Thus, students are advised to check with Dr. Luca Bedogni before starting to develop their own project.

6 Project bonus

While for the exam it is sufficient to implement the requirements presented above, we strongly encourage to enrich the application by adding new original features to the applications.

7 Projects submission

Projects must be submitted through email to lam-projects@cs.unibo.it, including all the code, a technical report, and a presentation (10-15 slides).