POLITECNICO DI MILANO

Facoltà di Ingegneria dell'Informazione

Corso di Laurea in Ingegneria Informatica

Technical Report

The MPower Presentation Layer

Matteo Ferroni

ferroni@elet.polimi.it

Version 0.1 - Last update: N/A

NECST Laboratory

mpower.necst.it

Contents

List of Figures

List of Tables

List of Listings

List of Abbreviations

TTL Time-To-Live

Chapter 1

Presentation layer and User Interaction

In order to describe the presentation layer, we start analyzing the implemented user interaction flows, then presenting every activity implementation.

Figure 1.1 describes the navigation flow through the application. Every block represent an Activity class, while the arrows indicates the possibility to go from an activity to another. These arrows have just a single direction, since the Android framework handles the foreground activities using a stack data structure: a new activity is pushed on top of that stack as soon as the user ask to launch it, covering the one previously shown. The user can come back to the previous activity pressing the "back" button of the device: the last opened activity is then closed and popped out of the stack.

The interaction flow is really simple and will be discussed in the next subsections along with a comprehensive description of every activity.

AuthActivity

The first interaction with the user is an authentication phase, handled by the AUTHACTIVITY. Our goal is to uniquely identify the device and its owner, in order to

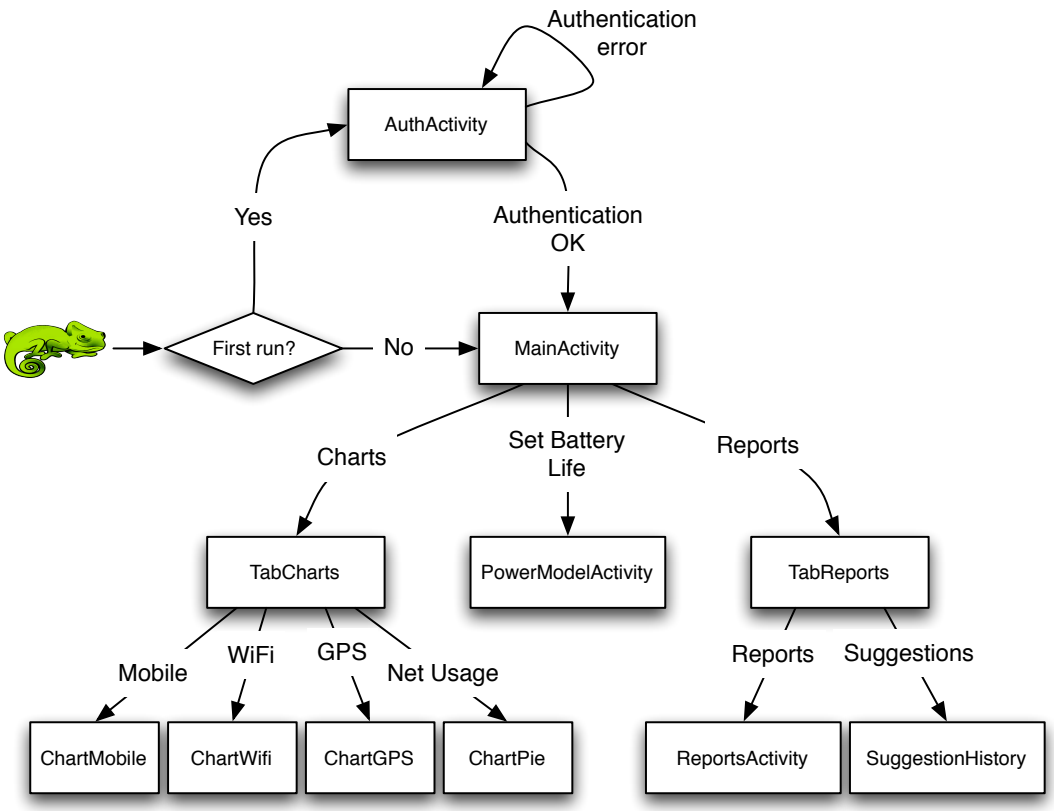

Figure 1.1: User interaction flow between the different activities

register these information as a single entity on our server infrastructure and prevent abuse of the data sending. We also consider the case of a user that owns multiple devices, such as a smartphone and a tablet.

The device identification policy is defined and implemented by the *Sense Libraries*, starting from its unique hardware features, as discussed in Section **??**, while an authentication mechanism is needed for the user identification.

We used *Pseudo-authentication* [**?**] for this purpose, that differs from standard authentication methods because the web application directly requests a limited access token from Google OAuth on behalf of the user, and the user grants us the permission to access his/her information. We then use the granted access to discover the personal

information of the user, so we do not directly authenticate the user, but instead we lean against a third party provider to do this for us, and we get the response from them.

We decided to implement the pseudo-authentication using Google OAuth2 Login only to verify that the email inserted by the user is a legitimate Google email address. Google is not the only third-party OAuth 2 service provider, but it is the best choice for our purposes since every Android user needs a Google account to access all the functionalities of an Android device.

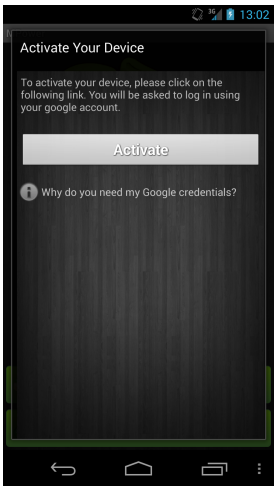

Figure 1.2: AuthActivity layout

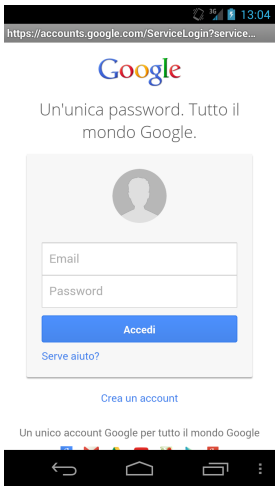

Figure 1.3: AuthActivity WebView

The whole Oauth 2 handshaking process is handled server-side: the AuthActivity just makes use of a WebView component to show a Web page on the Meth server, as soon as the "Activate" button is pressed (see Figure 1.2). The Oauth 2 "dance" is then started, as shown in Figure 1.3. At the end of the authentication, the Meth server obtains the name, surname and email address of the user from Google and can send back to the MPower application a cookie containing an *authorization token*. The WebView is then in charge of storing the received token using the Android Shared Preferences mechanism: this token will then be used to secure the communication between the MPower app and the Meth server. On the server side, a new user/device entity is stored in the database, identified by the retrieved information, the hashed token and

the device ID.

1.0.1 MainActivity

Once the authentication phase is completed correctly, the MainActivity is shown: this is the home MPower app screen. The estimated Time-To-Live (TTL) for the current device state and a "Set battery life" button are shown at its center, if the device has already collected enough data and a power model has already be retrieved from the server (see Figure 1.5). This information is obtained by querying the DatabaseHelper object. If pressed, the button will bring the user to the PowerModelActivity screen. If no model is still available for the current device, the application is in its learning phase and an information message is shown, as in Figure 1.4.

Figure 1.4: MainActivity layout during the learning phase

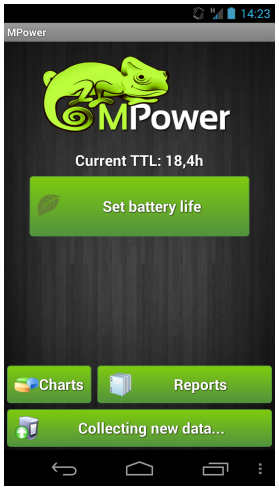

Figure 1.5: MainActivity layout when the model has been learned

The "Charts" and the "Reports" buttons allow the user to reach the TabsCharts and the TabsReports activities. Another button is placed under these two: if there are no log data ready to be sent to the server, the caption of this button is "Collecting new data", otherwise it becomes "Force data sending". In this last case, the user can launch the DataSendingTask and instantly send the data retrieved by MPower up to now.

At its first run, the MainActivity is also in charge of launching the LogService and the already described AuthActivity, if no token is found in the application Shared Preferences.

1.0.2 PowerModelActivity

The PowerModelActivity presents to the user a list of configurations in which the device can be set, with TTL information for each one of these.

The Meth server is in charge of estimating the TTL of the device for every observed configuration and for every battery percentage: a TTLs lookup table is then sent to the device. In this way, given a certain configuration and a certain battery level, the TTL estimation of the current configuration requires no computation on the device, since it is a simple fetch from the TTLs table. This table is handled and accessed using the methods exposed by the DatabaseHelper components.

Given a certain battery percentage, different TTLs can be achieved depending on the actual device configuration. The maximum and minimum TTLs are shown as the extremes of a progress bar, as shown in Figure 1.6: the user can then set the desired TTL between these two boundaries and the configurations that meet that constraint (i.e., the ones that guarantee a TTL greater or equal to the one specified by the user) are shown.

More constraints can be set: for instance, the user may want to specify that he/she needs the 3G, Wi-Fi and/or the Bluetooth module to be enable by pressing the logos corresponding to the chosen functionality. The configurations shown under the progress bar will then be filtered by the lightened logos. For instance, Figure 1.6 shows the configurations that guarantees a TTL greater than the one specified by the user in the TTLs bar (about 15 hours), with the 3G module left on: the related logo under the bar is lightened as the user selected it. The first configuration that meets these constraints has the 3G and the Bluetooth module on, the Wi-Fi module off, the screen

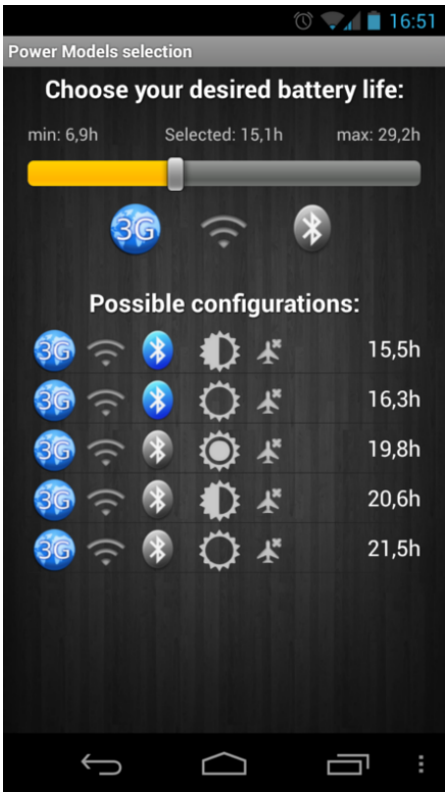

Figure 1.6: PowerModelActivity layout

brightness set at medium level and airplane mode off: in this configuration, we estimated that the device battery will last at least 15,5 hours. The other configurations guarantee an higher TTL and have different screen brightness levels and hardware modules states.

The device can be set in a particular configuration as soon as the user select it: actuation is finally performed using the Actuators Libraries.

1.0.3 TabsCharts and TabsReports activities

TabsCharts and TabsReports activities are composed of different sub-activities, organized in tabs.

TabsCharts contains graphs showing the battery discharging curve with respect to the GPS module operations (ChartGPS activity, see Figure 1.7), the amount of data

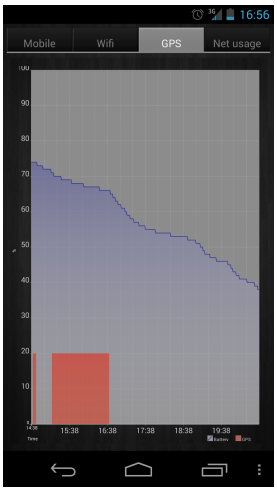

Figure 1.7: ChartGPS activity

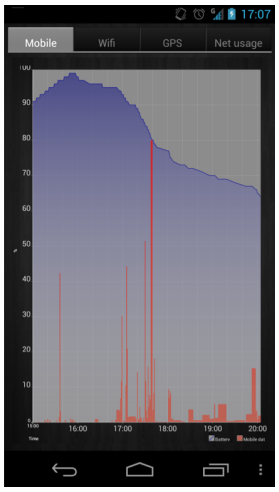

Figure 1.8: ChartMobile activity

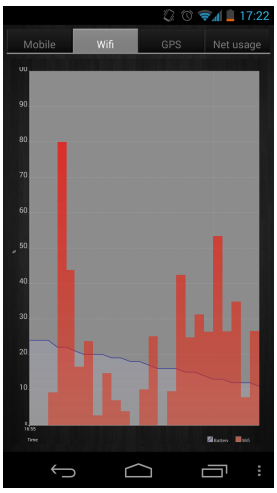

Figure 1.9: ChartWifi activity

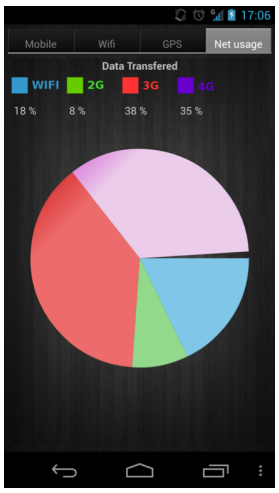

Figure 1.10: ChartPie activity

transferred on the mobile network interface (ChartMobile activity, see Figure 1.8) or the Wi-Fi network interface (ChartWiFi activity, see Figure 1.9), with a final tab showing the percentage of data transferred using a certain mobile network technology (ChartPie activity, see Figure 1.10).

TabsReports contains a tab with general information about the device behavior, like the last time it was fully charged or when log data has been last sent (ReportsActivity, see Figure 1.11), and a tab with the last suggestions reported to the user (SuggestionsHistory, see Figure 1.12).

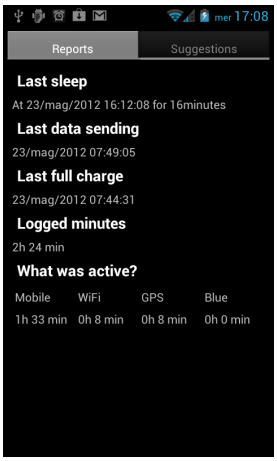

Figure 1.11: ReportsActivity activity

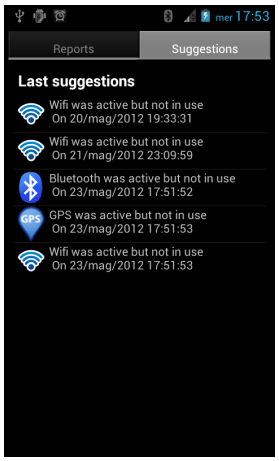

Figure 1.12: SuggestionsHistory activity

These activities rely on data stored on the application internal SQLite database by the LogService and the EventHandler components, that will be presented in Section **??**. These data are accessed through the DatabaseHelper component, that provides an abstraction layer to the underlying table structure.