SYSTEM FOR RESTRICTING USER ACCESS TO ANDROID OPERATING SYSTEM

Dumitru Bivol, Valeriu Ionescu
University of Pitești,Romania
bivoldumitru@gmail.com, manuelcore@yahoo.com

Keywords: Android, online database, interface customization

Abstract: *Production costs for small businesses in today's IT market conditions make it impossible to develop cheap custom hardware devices that target niche potential customers. The solution is to order standard devices from Original Equipment Manufacturers and customize them accordingly by adding hardware and software features. The system presented in this paper uses a standard Smartphone contained in a custom case that was designed to restrict access to the controls and keep the user inside the application while offering him the possibility to access most important functions of the device via software controls. The user operation and device data are logged and saved in online database in case that device has internet connection.*

1. INTRODUCTION

The main scope of this project is to offer to user of the application the possibility to transform his device from a stock android device to a device with dedicated functions.

Android is a mobile operating system that was introduced in 2007 by Google. The development of the operating system is made by Google but the source code is available under open source license, allowing the developers to investigates alternative implementations, especially for the user interface (UI).

In the past years Android has evolved at a rapid pace, a platform of billion of devices for all types of users, becoming the number one mobile platform. This is also related to the decrease of the hardware implementation costs and the size of the hardware that is capable of running Android with acceptable performance.

As today most of the hardware that runs Android is ARM based and is produced in China and neighboring countries because of low production costs, the developers are faced with the problem for developing software for a limited amount of variations of the same hardware platform. Because of this one of the trends in the market is order standard devices from Original Equipment Manufacturers and customizes them accordingly by adding hardware and software features [1]. Sometimes the user access to the platform is restricted in order to allow only custom interaction via the UI.

This application presents a new interface for the user that allows performing the common required actions while having only access to the graphical user interface.

The applications for this approach are the implementation of custom user interfaces for Android. Very important is the possibility for user to save some data somewhere online where it's safe and he is more than sure that can access that data later. In our case we used online host and a database. Also, in case that user wants to save data but he don't have internet connection data will be saved local in a file.

Another part of our app is working with Google Maps. User has the ability to see his position on map, to read latitude and longitude values and even with a connection to internet he can see the name of the city and country where he is located, very useful for someone who got lost or for the purpose of locating the respective device by the server.

The ability to read hardware data about device that user is using is also a feature of this

application. The user can see things like architecture of CPU, how many MB of RAM are in use or version, required for those who want to install a new firmware to his android device.

Finally another feature implemented in this application is the possibility of dynamically changing the text of the user interface. This is a very important if for example a translation for the user interface is necessary.

2. APPLICATION DESIGN

In figure 1 we have the main block diagram of the application. After the application is installed user must agree with some terms, in our case these are:

- Make sure there is a internet connection
- Make sure GPS is activated

These are necessary to make sure that user is informed about with app requirements for a full functionality. Without a connection to internet the main features of this app will not be visible.

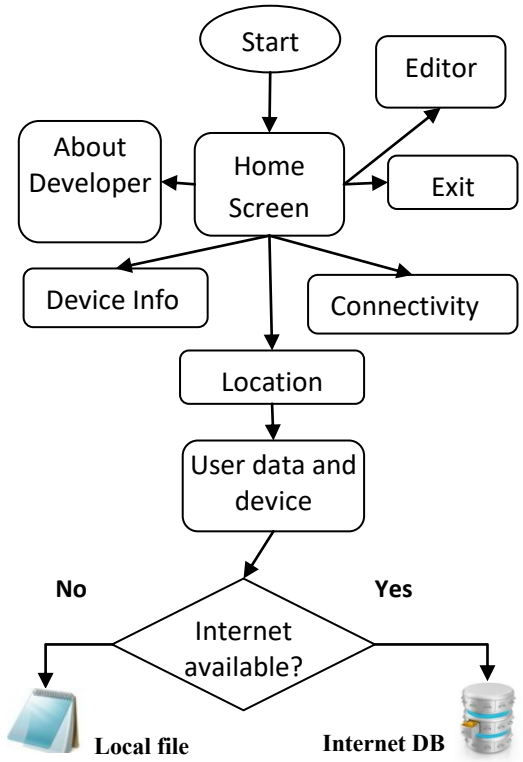

Fig 1 Block diagram of the UI interface

The application was built using Andrtoid Studio [2] and designed as a launcher in order to keep the user inside. After the user has reached home screen (seen in Fig. 2) and agreed with our terms, he will be able navigate to six different activities that offers different functions and show different information.

Fig 2. Application main screen

Device Info – in this activity the user can see device hardware information (Fig. 3).

Fig. 3 Device Info activity

Editor- once entered in this activity user has the possibility to edit the text of all buttons from the home screen of the app. This makes an application easy to translate in a different language.

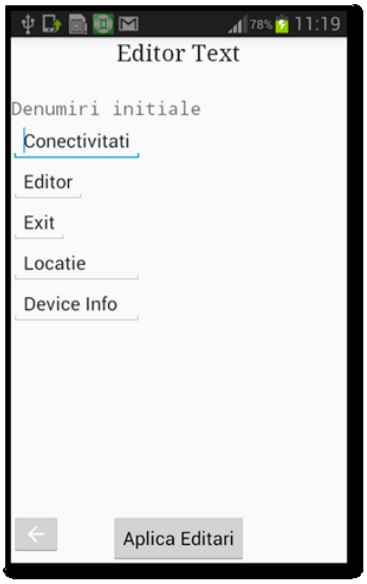

Fig. 4 Editor activity

Connectivity- here the user can activate and deactivate the most important features of Android device. Functions like Wi-Fi, Bluetooth, GPS, Mobile Data, can be easily activated/deactivated in this activity with an On/Off switch. Even if this is normally available on the control panel of any Android device after a swipe-down on the screen, as we wanted to lock the user in our application, this feature is now disabled, effectively locking user intro our app. This behavior is visible in Fig. 5.

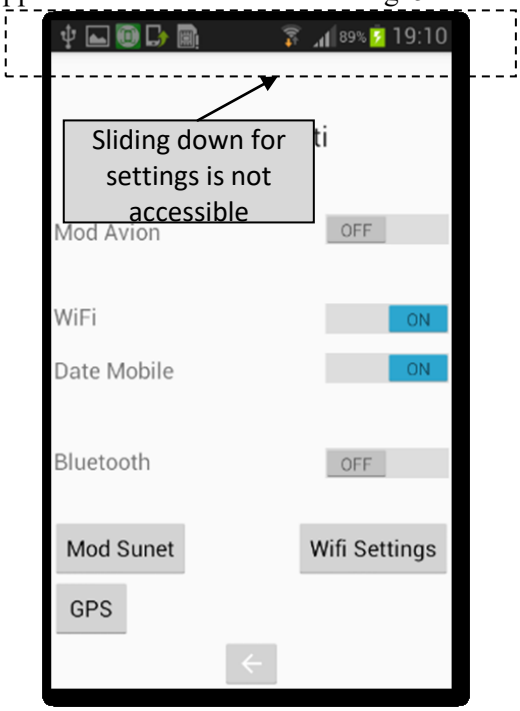

Fig. 5. Connectivity activity

Location- here is the main activity of our app. First, we show to user some hardware information including battery level. GPS location information is only available after a lock is established. The time of response depends very much on the location of the user and if there is any internet connection to device. This will improve not only time of reply but also the accuracy of data that will be sent. After the lock is established, we can proceed and hit the Map button where we can see our location in real time on the Google Map. If we want to see a more accurate version of our location we can hit another button that will show our location in terms of streets.

Another aspect present in many applications is the uploading of user location data to an internet server for the purpose of monitoring. For example many employers use this data to see that the mobile user did not leave the optimal path or exceeded different restrictions such as speed or time between stops. In the **Location** activity the user can save data in a database online in case the user has internet connection or on a local file if this is not the case. Information will be sent (or saved) every 30 seconds to the internet server (or local file). Associated with this data is included location and battery level.

In case that the device gets lost, in online database still remains entries with name of device, battery level, time and location, very useful information in order to locate the device's last known position.

The user interface is visible in Fig. 6. The user can see the current location when the button *Harta* is pressed. This launches an interface visible in Fig. 7.

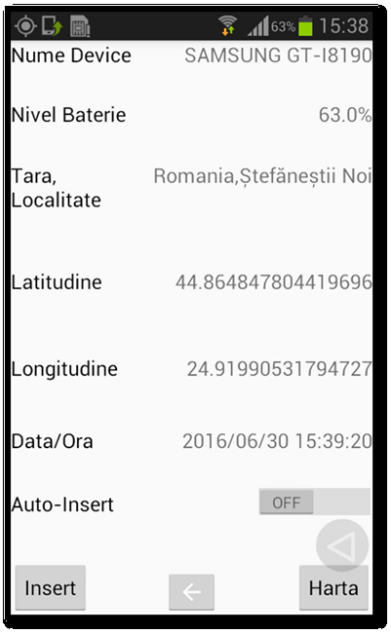

Fig. 6. Location Activity

Fig. 7. Map Activity

The last activity implemented is the **Exit** activity. The main purpose of the project was to keep the user locked inside the application, this last Activity is password protected. A future improvement is to make the access to this activity hidden (now it is a button) and launched for example only if a certain touch sequence is performed.

Fig. 8. Exit Activity. Password request

Of course even if the user is completely locked in the software, it is still possible to reboot the device and revert it to its previous stated (remove the custom software/launcher). This is why we have built for the device a hardware case that prevents the user to access the buttons and allows only the touch interface and the charging port. This is visible in Fig. 9.

3. APPLICATION IMPLEMENTATION

1. Sending data to database online

This app needs to interact with outside world. There are multiple ways to do this but for this project the choice was to use PHP. To make app work with PHP the **Volley** library was used. It is not included in Android Studio and it is mainly used to connect Android applications to the outside world. From this library the **RequestQueue** class was used and also **StringRequest.** First of all we created a MySql database on a website that offers free web-hosting and then created the connection between database and our PHP code. We need 2 PHP files, one to establish connection and one that contains the function for inserting data. To make

sure that PHP communication works we've used PostMan extension from Google Chrome browser. All we have to do there is to specify the URL to the PHP file with insert function and write some data. After the check was completed we can step up to next level, making connection between our android app and php files.

Now in Java class we need to create objects from the classes *RequestQueue* and *StringRequest.* In the *StringRequest* object we need to specify the method used to send data and in this case is *POST* , also we need to specify the URL to the insert file from our webhosting. After this we used method *getParams()* which will send data from Android app to database.

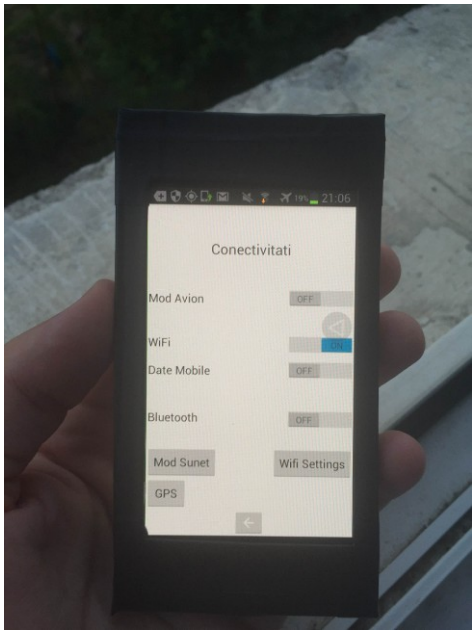

Fig.9 The device with hardware case

2. Getting Data from GPS

The main problem of this was timing, as the user needs to wait depending on his location a few seconds or a few minutes to get a GPS lock. This was implemented by checking for the two events that are triggered: one event for the GPS search start and the other for the GPS locked. The implementation was made in the ***Location*** *class*[3][4]. Then we create a object from this class and called *getLatitude*() and *getLongitude*() methods from this class. GPS gets this request and after a lock is acquired, data is sent to user and they are showed in *Location* activity. Afterwards, this data is used to get Country and Locality name with *getCountry*() method and *getLocality*() wich works instantly.

Fig. 10 presents how things work in the **Location** activity. Some data can be displayed instantly but other data like Latitude and Longitude needs some time before they can be displayed to the user. This activity needs GPS activated and an internet connection because to get latitude and longitude we need GPS, and in order to get the country name and city name to the user we need a internet connection.

Also, when *Insert* button is pressed app checks for an internet connection, if there is any, data is saved in online database; otherwise data is saved in a local file. Same for Auto-Insert switch that send data once in 30 seconds depending on the internet connection availability.

While developing this app multiple problems had to be solved. Some of them are: Hiding control panel, editing text from other activity, getting data from GPS, Updating the interface in real time, locking user into our application. The following section describes these problems in detail.

Android offers possibility to developers to get rid of control panel/status bar by using *Immersive Mode*, but this method will hide status bar and when user swipe down on the upper part of the screen control panel will show up but this was not intended, therefore we used a different method. We allow the device to show the control panel shows after swipe-down action but the layout will be made transparent and the user will not see anything.

Another challenge was related to changing the interface text from Textviews from another activity. We have used *SharedPreferences* class which allows us to save data to a object and then write it on Textview that we need.

The last problem is related to GPS locking. We must be sure we will not send multiple requests to not put the user in a position to wait too long. When Location Activity opens, the application sends a request to the GPS module to receive latitude and longitude data. The module will respond differently depending on the location of the device and internet connection status. An internet connection or mobile data greatly improves response time data received is very accurate.

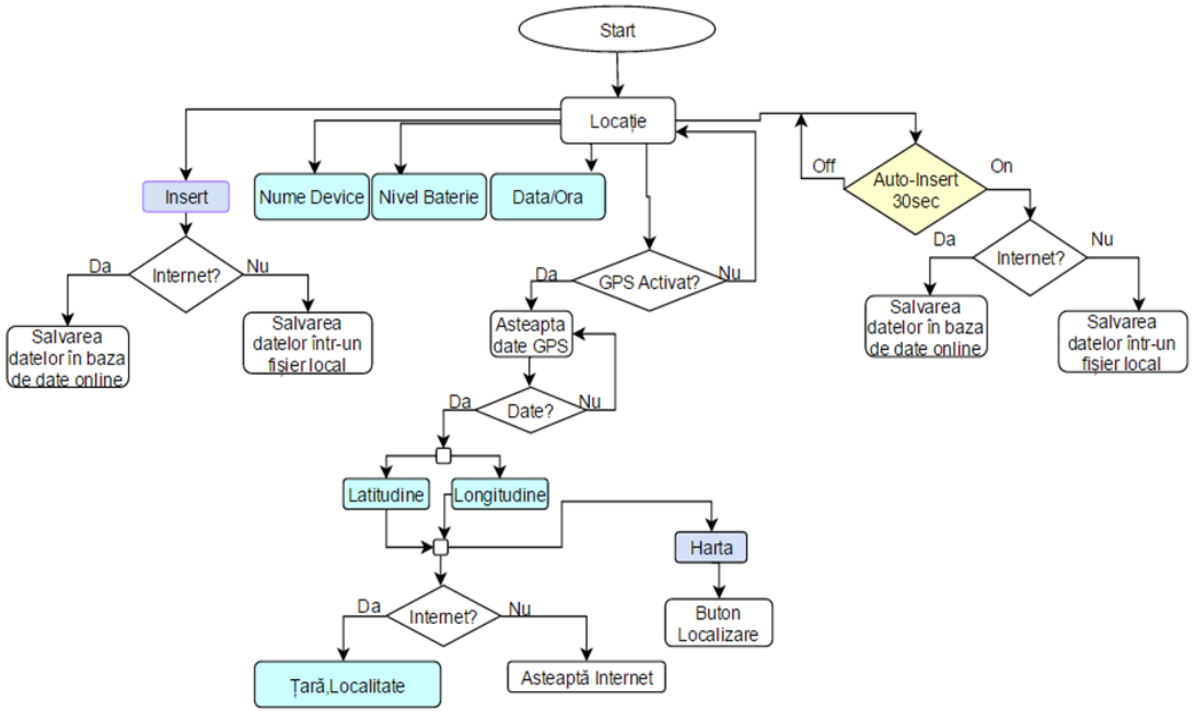

Fig. 10 Flow Diagram for Location Activity

5. CONCLUSIONS

This application is a sample to the creation of closed custom devices based on OEM hardware. The purpose was to lock the user in a software container (launcher) and allow the user to perform only the actions we allow. The hardware was closed in a physical container in order to prevent the user to reset the device to its original state.

The presented hardware and software solution was able to lock the device and run inside our own software interface.

There are a lot of things in this app that can be improved, one of them and most important I think is design of the app The current application was designed with the purpose of keeping the user inside the designed interface. As a development, a mode modular approach should be taken that allows developing new models with the same design goals.

1. REFERENCES

[1]. Hewlett-Packard, Build Your Brand on HP: HP OEM Partnershipard Web. March 2014, Available:http://h22168.www2.hpe.com/docs/oem/4AA4-7081ENW-314_jeedits.pdf

[2]. Google, About Android Studio, Web, Available:*https://developer.android.com/studio/intro/index.html*

[3]. Google, About Geocoder Class, Web. Available:.*https://developer.android.com/reference/android/location/Geocoder.html*

[4]. Google, About Location Class, Web. Available: *https://developer.android.com/reference/android/location/Location.html*.