Android Application Front-end for an Energy Brokerage Agent

Christos Petsos, Kostas Kalogirou, and Evangelos Bekiaris

***Abstract*—**This paper outlines the development process of an energy management model, the Brokerage Agent Front-End application (BAF) for Android smart phone devices. This application is addressed to end users who consume and produce energy from any type of source such as photovoltaic panels and wind turbines. The application is a front-end illustration of data from smart meters that enables the user to achieve a more efficient energy management. Moreover, this paper analyses the technical issues that were considered in order to reuse the code created for the Brokerage Agent Front-End web application [12] and the additional implementation activities needed for the native Android application.

***Keywords*—** Android native application, energy management system, solar energy, interactive charts.

I. Introduction

THE initial approach was to investigate whether the code from the BAF web application [12] could be used also for the Android version. Although Android technology fully supports RESTful web services and Google protocol buffers, additional classes had to be written in order to be fully functional. Then, the user interface requirements had to be considered during the design process. The limited screen size of the Android smart phone mandated the rendering of graphical charts in a landscape format rather than portrait. The following paragraphs analyze the research that took place in order to investigate the technical requirements, presenting the functionalities provided by the BAF Android native application to the standard prosumer user (is defined from words producer and consumer).

II. Technical Requirements

The paradigm of smartphone applications, like that of the BAF for Android, poses a number of additional challenges to the ones of the initial web-based application. The first thing that had to be evaluated was whether the web-based application was fully functional in the smartphones' browsers. The Android application had to be able to serialize/deserialize Google Protocol Buffer (GPB) messages and send/receive them through a secure RESTful communication channel. Additionally, time-based energy data should support the end-user's timezone as selected in the Android device. The Android devices used for evaluation were HTC Sensation Z710e, Samsung Galaxy S3 and Sony Ericsson XPERIA X10 mini E10i.

A. JSF 2.0 application compatibility

After the research that took place during the design of the BAF web-based application, the JSF 2.0 implementation selected was the open-source Mojarra 2.0.3 provided by Oracle [2]. In addition, the PrimeFaces 3.0.M1 for JSF was used as component suite library [3] for the composite UI elements. The application server used to deploy the web application was Apache Tomcat 6.0. Finally, the chart component used to visualize the smart meter's historical data was Flot 0.7 [1] which is an HTML 5 component controlled with JavaScript through JQuery.

The optimum scenario would be that the web application could run as it is on the smartphones' web browsers. Unfortunately, this was not succeeded. During the evaluation on HTC Sensation Z710e and Samsung Galaxy S3 certain HTML rendering errors occurred along with the inability to select time slot in the Marketplace's chart. Android applications introduce new semantics in human-machine interaction [6] and those were not taken under consideration during the implementation of the web based application. For instance, in the web-based application selecting a time period in the chart's graph was simply a "mouse-down, drag, mouse-up" sequence. However, in smartphones' web browsers this procedure corresponds to the panning of the browser window, so an essential piece of functionality was not available in the smartphones. In the case of Sony Ericsson XPERIA X10 mini E10i the situation was even worse since the browser was complaining with parsing errors and could not launch the application at all.

Following the feedback acquired from the evaluation of the web-based application in these devices, it has been decided that a native Android application should be implemented for the smartphone version of the BAF application.

This work was realized during the European research FP7 funded project NOBEL and the submission of this paper is supported by the Hellenic Institute of Transport of Centre for Research and Technology Hellas (H.I.T./CE.R.T.H.)

K. Kalogirou, software engineer with MSc at Digital Signal Processing, H.I.T./CE.R.T.H., Thessaloniki, 57001, Macedonia, Greece, phone: 0030-2310498461; e-mail: kalogir@ certh.gr.

C. Petsos, MSc Software Engineering, H.I.T./CE.R.T.H., Thessaloniki, 57001, Macedonia, Greece, phone: 0030-2310498431; e-mail: cpetsos@ certh.gr.

E. Bekiaris, Dr. Mechanical Engineer, Research Director in H.I.T./CE.R.T.H., Thessaloniki, 57001, Macedonia, Greece, phone: 0030-211069551; e-mail: abek@ certh.gr.

B. Reuse of existing code

Having to re-implement the whole application for the Android platform was initially a large frustration. However, a rather pleasant surprise came up very early in the design phase of the Android application. It seems that the use of JSF, and inherently the Java platform, bridged a large percentage of the gap between the web-based and the native Android application, since Android development means Java development. The design of the initial web-based application was as follows: the JSF front-end was communicating with the JSF back-end using the specific JSF semantics and the JSF back-end was consuming a secure RESTful web service that provided user authentication and energy data using a Java client, specifically the Jersey open source JAX-RS (JSR 311) reference implementation [7]. GPB messages were transmitted over this communication channel. So, for the Android version of the application, the only thing that needed to be replaced was the JSF front/back-end with that of the native Android application front-end, including Intents, Layouts, Dialogs and graphical components. The whole design stack and the reusable elements are presented in the following figure.

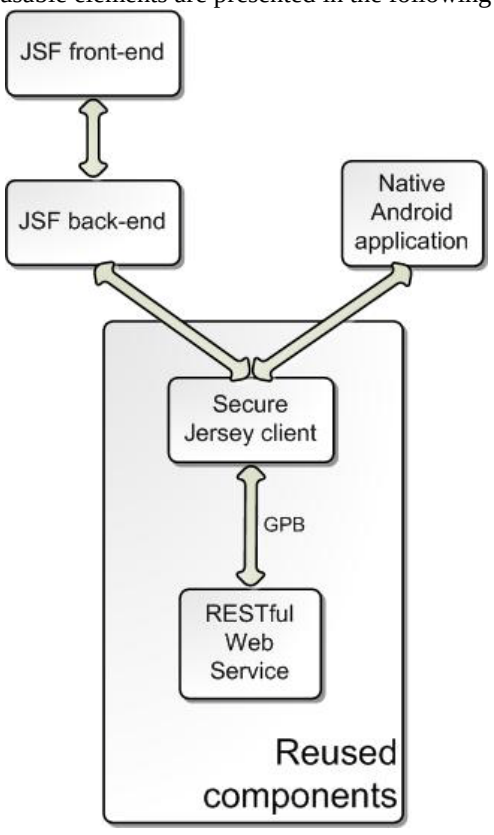

Fig. 1 BAF application design stack and reused components

C. Reusable components integration

Re-using the parts of code that used the Jersey client to consume the RESTful web services was a big advantage in the design and implementation of the Android BAF application. However even following this path obstacles were faced. During the first tests with the Jersey client, a number of "NullPointerException" errors occurred during run-time. Jersey client was designed for desktop and server applications, so an official Android port did not yet exist. Jersey has to read a file located at "META-INF/services/" location in order to operate correctly. However, the Android platform compiles all application files to Dalvik Virtual Machine specific bytecode and there is no way to reference internal application files by location. Implementing a Jersey client version for Android was out of scope for the NOBEL European funded research project [5]. Hopefully, a less time-consuming solution could be employed by providing a custom ServiceIteratorProvider instance to the client's ServiceFinder [8].

Serialization and deserialization of GPB messages operated with no additional effort for the Android application. Additionally, timezone support mechanisms incorporated in the web-based application were also used intact by feeding them the timezone that the user has selected in the Android platform.

III. User Interface Requirements

Apart from the technical challenges that are introduced when porting a web-based application to a smartphone platform, another important point arises; that of user interface design and interaction paradigm shift.

A. Smartphone device use case

The first notable difference when comparing an application targeting desktop machines and one that is designed for smartphones is that of the screen size difference. Small screens on smartphones enforce a total redesign of the Human Machine Interaction (HMI) protocol. This was very easily observed when evaluating the web-based application on the HTC Sensation Z710e device and Samsung S3. Both smartphones have a rather large screen for a handheld device, 4.3 inches. Even in that size, interacting with the web application using the touch screen was rather cumbersome. Text size was too small and interaction cues, such as buttons and drop-down menus, were difficult to be clicked. When the user zoomed in order to easily read text and press buttons, he/she had to constantly pan to all directions to follow the application flow. It looks like a web application designed for desktop screens, without any provision for small-screen devices, is impossible to be used in a smartphone. Considering that, it can be argued that the only components that could be reused in the Android application were the ones that were actually reused. The JSF front-end should have been redesigned for small-screen devices and the back-end should have to be altered.

Another important thing in Android applications is that of the orientation support. Most smartphones' screen aspect ratio demands for different components layout in portrait and landscape view. Orientation support has been implemented in the BAF Android application by using the built-in mechanisms provided by the Android SDK.

Limitations in screen's real estate enforce a new way of thinking the user input too. Navigation and action cues are

replaced by means of haptic input [9, 10]. Since there is not much screen to place many buttons, links, sliders etc, user input is captured in gestures and special screen touches. In the BAF application for Android a back gesture has been implemented for navigating to the previous screen, apart from using the home button. In addition, long clicks, dragging and sliding have been used for interacting with the graph charts. These means of user input may puzzle someone used to traditional desktop applications at first. In a small period though they become intuitive and in some cases more user-friendly than the traditional ones.

B. *Android Charting tool*

The specifications of the BAF application contain the ability to draw several types of graphs in order to present information to the user about energy values, orders etc. These requirements are not less than those of the web-application. The charting tools evaluated were the following:

1) Google Chart Tools
2) AchartEngine
3) ChartDroid
4) AndroidPlot
5) aiCharts
6) GraphView
7) RChart
8) TeeChart

None of the evaluated tools supported interactive charts. An essential element that needed to be supported was that of selecting time periods in the Marketplace. Such a feature was not available in any of the tools. Finally, considering the BAF application requirements and based on a cost/value basis, the RChart library for Android [11] has been selected. This library supported most of the functionality that the BAF application needs for displaying charts. It had some infrastructure available for selecting time-periods but did not support panning and zooming. Hopefully, the source code of this tool was open, so it has been extended rather easily in order to support pan/zoom and selecting time periods.

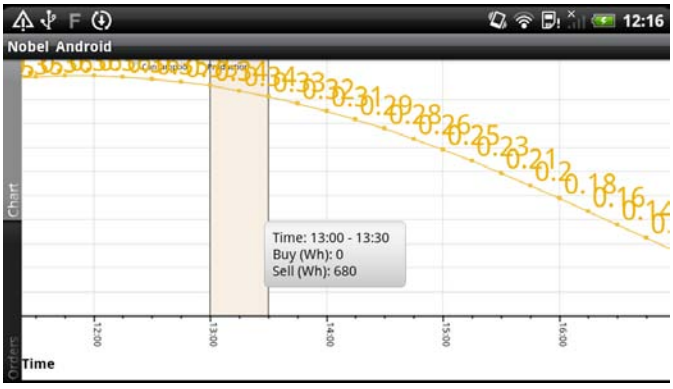

Fig. 1 pan/zoom and selected time period

IV. SUPPORTED FUNCTIONALITY

The current version of the BAF application for Android supports the following pieces of functionality:

1) Monitoring of historical and statistical data about consumption/production.
2) Prediction of future consumption/production of a single smart meter.
3) User profile and device profile management.
4) Marketplace where the user can sell or buy excess of energy.
5) User authentication/authorization.
6) Internationalization.
7) User input validation.

Initially, the user is presented with the login screen.

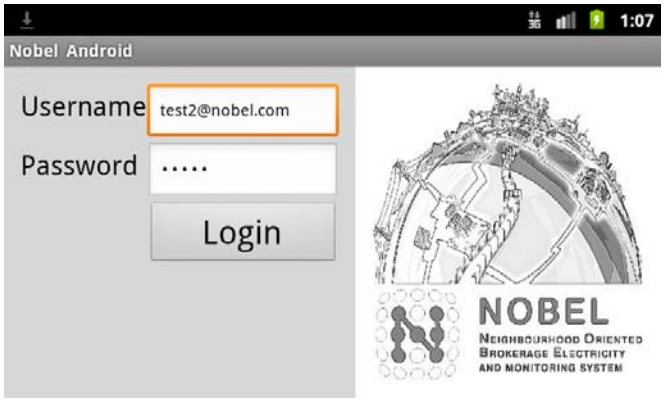

Fig. 2 login screen

Upon successful login the main application menu is loaded.

Fig. 3 main menu

Here the user can select the various operations provided by the BAF application. Clicking the “Monitoring” button leads to the screen where user selects device and time period to monitor.

ISBN: 978-1-61804-221-7

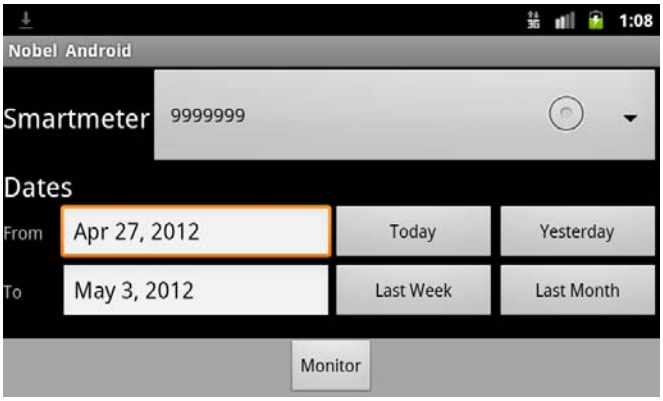

Fig. 4 monitoring options

Selecting "Monitor" presents the chart graph with the energy production and consumption data fetched over the RESTful Web Service.

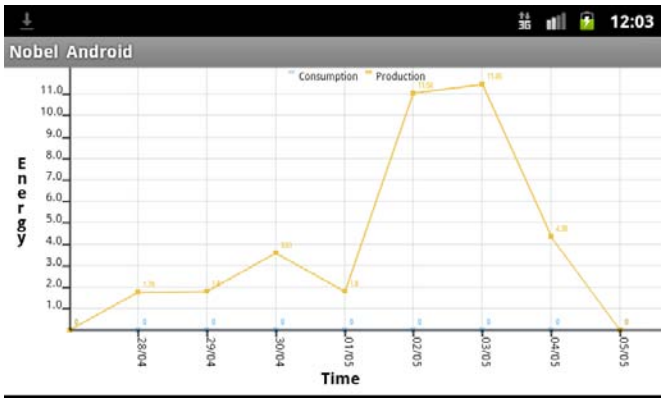

Fig. 5 Monitoring data for last week

Monitoring is allowed only for past dates since it fetches historical data from real devices. On the other hand, the prediction service is only allowed for future dates since it provides an estimation of the energy consumption and production for a specific date in the future.

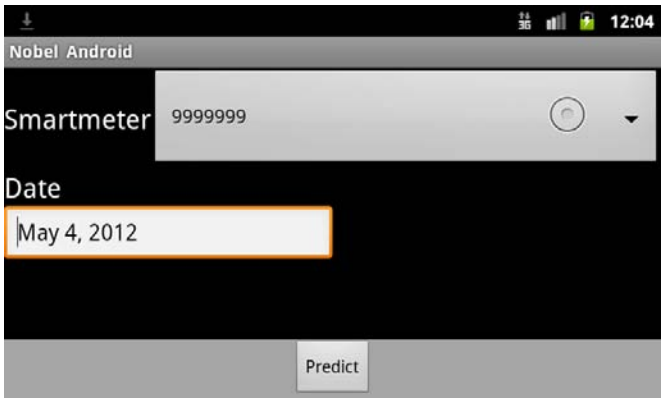

Fig. 6 prediction options

After specifying device and date, clicking "Predict" fetches prediction data and draws the prediction graph for the selected date.

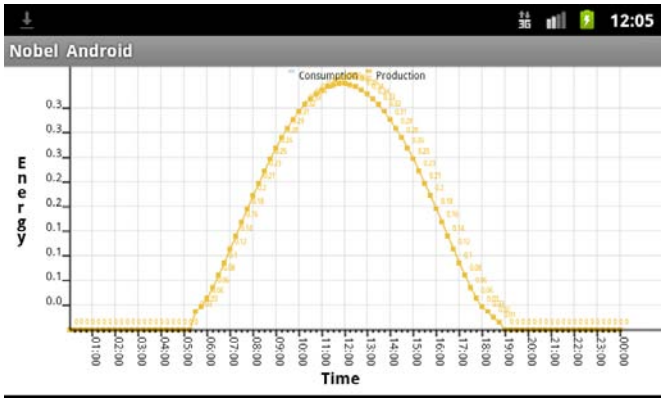

Fig. 7 prediction graph

In the Marketplace scenario the application flow starts exactly as the Prediction service; user fetches prediction data for a future date. However, it continues providing two additional steps of interaction. The user can select a time period and view the amount of energy that he/she can buy/sell. For instance, in the following figure the user has selected time from 12:00 to 14:00 and can observe that he/she has an excess of 2744 Wh of energy which he/she can sell.

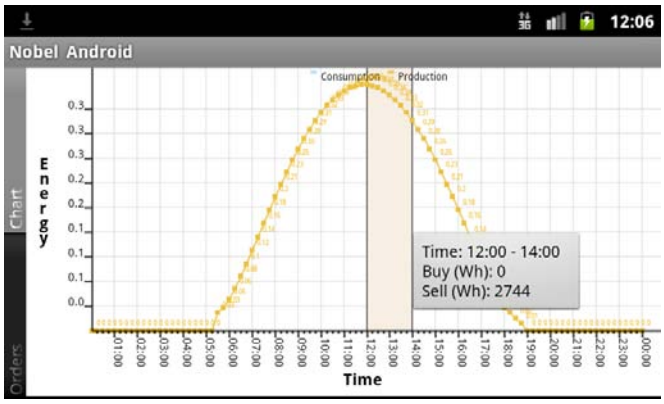

Fig. 8 marketplace time period selection

By clicking the "Buy/Sell" button the user is presented with a set of options that he/she can set in order to complete his/her order.

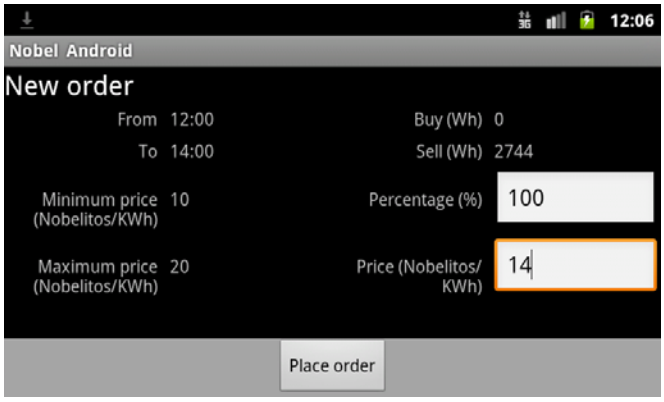

Fig. 9 marketplace order options

ISBN: 978-1-61804-221-7

Specifically, the user can select the percentage of energy that he/she wishes to buy/sell and the price to which he/she will place the bid in the marketplace. By clicking "Place Order" the user is prompted to confirm his/her order prior to communicating with the RESTful web service.

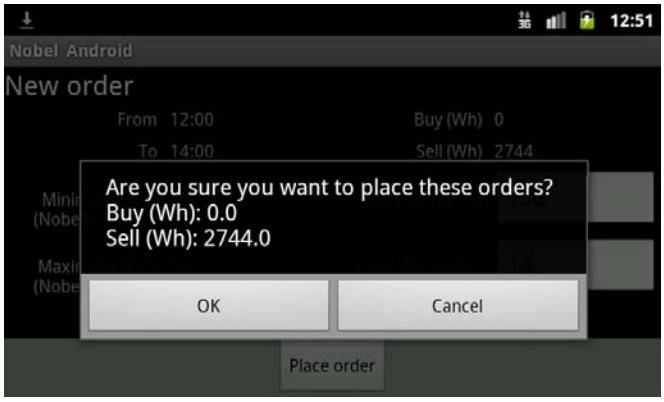

Fig. 10 order confirmation

In the "Management" screen the user is able to view his/her user and device profile.

Fig. 11 customer profile

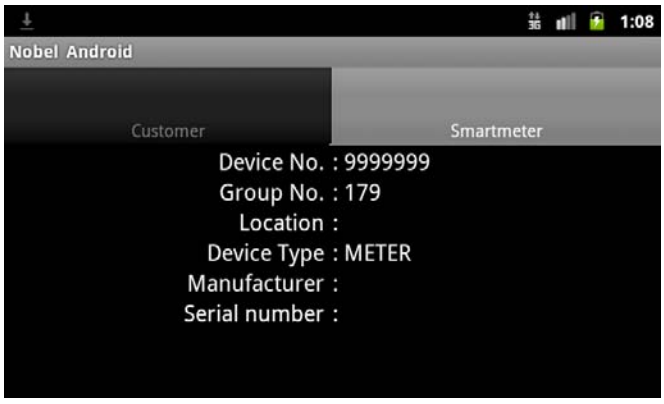

Fig. 12 smart meter profile

Another feature that has been implemented is the support of multiple languages. The following languages are supported:

1) English
2) Spanish
3) Spanish (Valencian)
4) Greek
5) German
6) Swedish

Below, the login screen of the application in the Spanish language is illustrated.

Fig 13 login screen in Spanish language

Finally, the built-in mechanisms of the Android platform were used for validating user input.

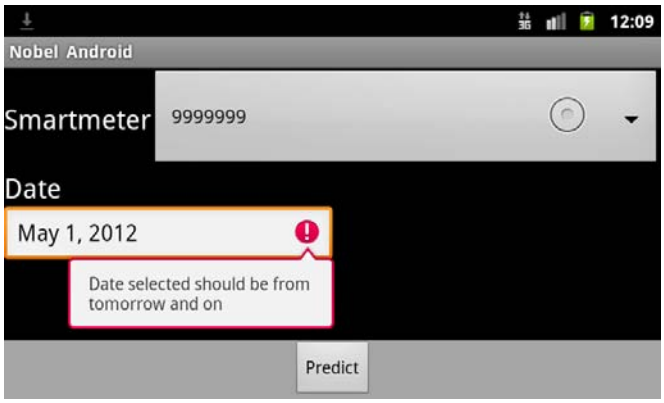

Fig. 14 user input validation

V. Future Work

An important issue that arose during the design and implementation of the BAF application for Android platform was how a web-application design can target to smartphone devices. The combination of JSF design for desktop application and native Android application may be challenging under the umbrella of Java, however a better solution would be to implement a smartphone-ready web application by simply redesigning the JSF front-end only.

Currently, a certain feature is missing from the Android application, compared to the web-based version; that of displaying the user's orders in a tabular view. Presenting information in tables is inappropriate for smartphone screens. Certain questions that come up are: how can one present such dense information in a small screen? How are multiple charts combined in such screens?

ISBN: 978-1-61804-221-7

VI. Conclusion

This paper shows an energy management application on the Android OS platform. The user can take advantage of such application for power quality monitor in order to reduce cost and emissions. Furthermore, a robust energy management system on both software (application) and hardware (smart meter) layers may provide additional valuable and consistent historical data for future investments and savings in specific worldwide geographic areas.

Appendix

The work described in this paper is also available on Android Tablets 4.1.x. The Android tablet version was evaluated on Samsung Galaxy Tab 2 10'' device [14].

Acknowledgment

The Brokerage Agent Front native Android application was developed and funded by the NOBEL FP7 (Grant agreement no FP7- 247926 - ICT-2009-4) European research project [5]. The technical research, design and implementation of the applications have taken place within the NOBEL project's lifecycle. NOBEL is featured together with some selected projects as best practices in a European Commission booklet [15].

References

[1] Flot Homepage - http://code.google.com/p/flot/

[2] Oracle Mojarra JSF Homepage - http://javaserverfaces.java.net/

[3] Primefaces Homepage - http://www.primefaces.org/

[4] E.Bekiaris, C.Petsos, and K.Kalogirou, "Energy management application: brokerage agent front end application", ICPES 2012, April 2012, Hong Kong

[5] http://www.ict-nobel.eu/ , NOBEL EU project FP7 – 247926-ICT-2009-4, 2009

[6] M.Song, H.Song and X.Fu , "Methodology of user interfaces design based on Android", Multimedia Technology (ICMT), 2011 International Conference, 26-28 July 2011

[7] Jersey Homepage - http://jersey.java.net/

[8] Jersey Community Forum - http://jersey.576304.n2.nabble.com/java-lang-NullPointerException-on-Android-td4212447.html

[9] Y.Li, "Beyond Pinch and Flick: Enriching Mobile Gesture Interaction," Computer, vol. 42, no. 12, pp. 87-89, Dec. 2009

[10] K.Wolf, C.Dicke, and R.Grasset, "Touching the void: gestures for auditory interfaces", In Proceedings of the fifth international conference on Tangible, embedded, and embodied interaction (TEI '11). ACM, New York, NY, USA, 305-308, 2010

[11] RChart for Android Homepage - http://www.java4less.com/charts/chart.php?info=android

[12] Nobel web application - https://www.nobel-baf.eu/testChartOffline

[13] Nobel Android application - https://play.google.com/store/apps/details?id=com.certh.nobel.android&hl=en

[14] Nobel Android Tablet application- http://www.hit-projects.gr/TabletAndroidNobelOffline.apk

[15] European Union, "ICT for Societal Challenges: Digital Agenda for EU (DAE)", http://ec.europa.eu/information_society/newsroom/cf/dae/document.cfm?doc_id=1944 , p. 27, 2013