CASE STUDY OF VEHICLE PARKING MOBILE PAYMENT APPLICATION: DATA STORAGE AND SYNCHRONIZATION SOLUTION

Jekaterina Smirnova, Nikolajs Bumanis, Gatis Vitols
Latvia University of Agriculture
jekaterina.smirnova@llu.lv, nikolajs.bumanis@gmail.com, gatis.vitols@llu.lv

Abstract. Various mobile applications can be used by mobile devices. It is possible to perform various transaction type operations with those applications, for example, make payments. However, possibilities of single application are often limited. New solutions which allow creating mobile applications without any special programming knowledge are being developed in the world now. These solutions give a possibility to develop new application using the existing components. Each component may be individual application by itself, with the ability to perform one simple task, for example, authorization or payment. Other components fulfill the role of dynamic (buttons, textboxes) and static content (logo, information text). Components are being stored in the data storage cloud maintained by a particular company. It is required to monitor the usage of components; therefore, log of activity is being utilized. Problems occur if the mobile device is lost or a new one is purchased. The task is to restore user settings and history of services that were active in the previous device. The mechanism to solve this problem must be created. A possible solution is data storage cloud maintenance within which each user has a profile. This profile will store the necessary information to restore the applications and settings in their current state. This profile serves as an intermediary between the service provider and the mobile device.

Keywords: mobile application development, cloud storage, parking application.

Introduction

The payment for parking places in Latvia can be done in two ways: by using stationary payment machine, including both, cash and the bank card, or by using mobile application to send appropriate *SMS*. The second scenario can be realized using a smartphone or other mobile device. The complexity of payment methods, which use smartphone applications, are rapidly growing. This type of service is implemented into *Mobilly* (developed by *CityCredit*) [1] and *RigaParking* (developed by *Amberphone*) [2] applications. These mobile applications are mainly designed for usage within the Riga city and few districts of Riga. There are no applications which offer more opportunities and can be used throughout the Latvian territory. The existing applications are available only for three cities – Riga, Liepaja and Daugavpils [1].

Materials and methods

Application can be installed by downloading it either from the developer's website or public application market. Both *Mobilly* and *RigaParking* applications are available for downloading from the largest [3] application stores: *AppStore*, *Google Play* and *Windows Phone.* Mobile application store is a software distribution platform, which is often implemented as a part of the device operating system [4]. Using application stores it is possible for the users to view details and reviews of the particular applications, choose products for purchase, download and install software. Application installation is a common procedure and follows the standard installation scenario (see Fig. 1). All proposed parking payment applications in Latvia are platform dependent, which means that application must be downloaded from appropriate application store. For example, *Mobilly* is *iOS* based application, which is available only on *AppStore*.

Applications aimed for payment processing, including payment for parking places, usually require some sort of user identification. Therefore, it is necessary to implement user account creation in the context of the application. In addition to typical user ID and password, the smartphone number can be used. The number is unique; therefore, it allows the merchant to identify the user. In case, where a mobile device is being used by multiple users, application ID can be applied as well, i.e., application has its unique instance.

In most cases, the user profile stores the following data: user name (phone number) and password, vehicle registration number, details of payments, payment history, account recharge history and the amount of the available funds. It is considered unsafe to store the profile data on the mobile device alone, as it may be, for example, hacked or lost [5]. Therefore, a way to store data outside of the

mobile device must be implemented. The possible solution is to use cloud computing services, implemented by trusted organization[6].

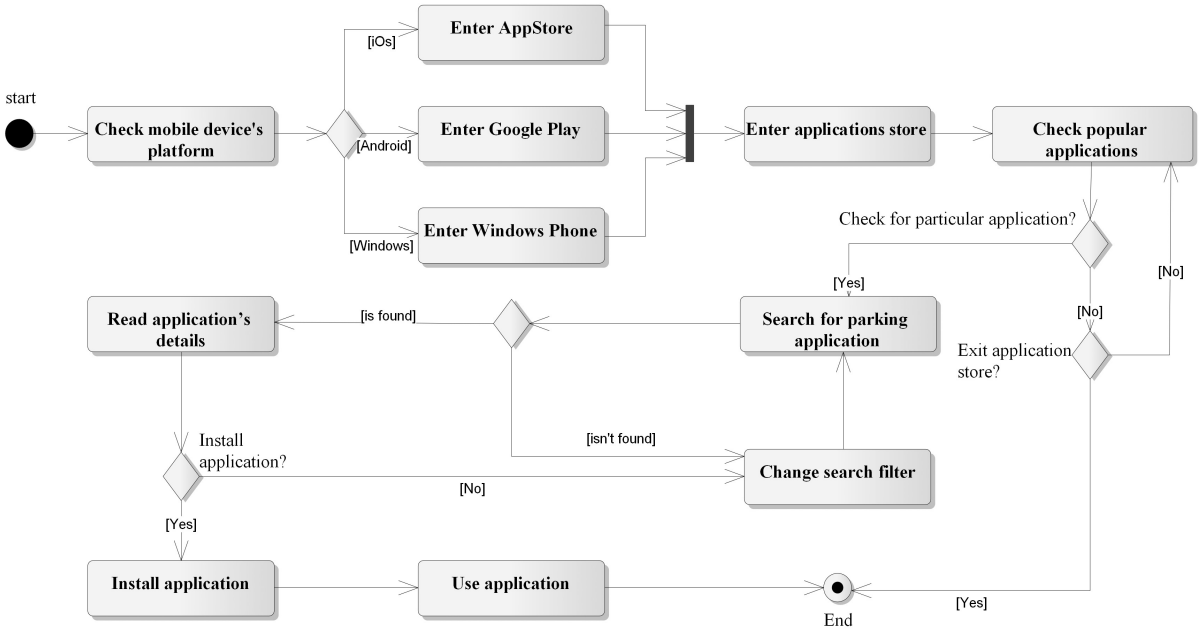

Fig. 1. **Standard installation scenario of mobile application**

For data transfer between mobile devices and cloud storage, the process of synchronization must be implemented. Synchronization can prevent the following data loss problems:

- when the mobile device has been lost or broken;
- when application has been uninstalled before the same application was installed again, using its updated version;
- when the mobile phone number has been changed to a new one, which requires update of link to the existing user's profile.

The essence of the synchronization process is the data exchanged between the client and server. Two types of synchronization are implemented for applications that process the payment for parking places [7]:

- user profile synchronization;
- application synchronization.

Application version synchronization can be initialized either by manually checking the possibility to update the application to the latest version or by receiving a notification of the possible update from application merchant. Or if the update process is automatized, application can generate the message itself.

Two types of profile synchronization are implemented in the application - fast and slow synchronization. Slow synchronization is being initialized when the application is started for the first time: it is necessary to receive all possible profile data from cloud storage, as well as save the mobile device data on it. Slow synchronization is initialized even if an error occurs during fast synchronization. The result of slow synchronization assures that the data provided on both, the mobile device and cloud storage are identical. The process of synchronization begins with data transfer from the mobile device to the cloud storage (see Fig. 2-a), changing to opposite direction (see Fig. 2-b) when the data transfer is complete, providing that both entities have identical data. Even if data transfer from the mobile device to the cloud was unsuccessful, the data transfer from cloud storage to the mobile device is still initialized.

In case of fast synchronization only the users profile changed data are being synchronized. The data concerning the last successful synchronization fact are being stored on both, the mobile device and cloud storage. The process of fast synchronization initializes the same data transfer pattern as it is

being implemented with slow synchronization, by synching the mobile device data with cloud storage at first (see Fig. 3-a) and doing opposite transfer after (see Fig. 3-b).

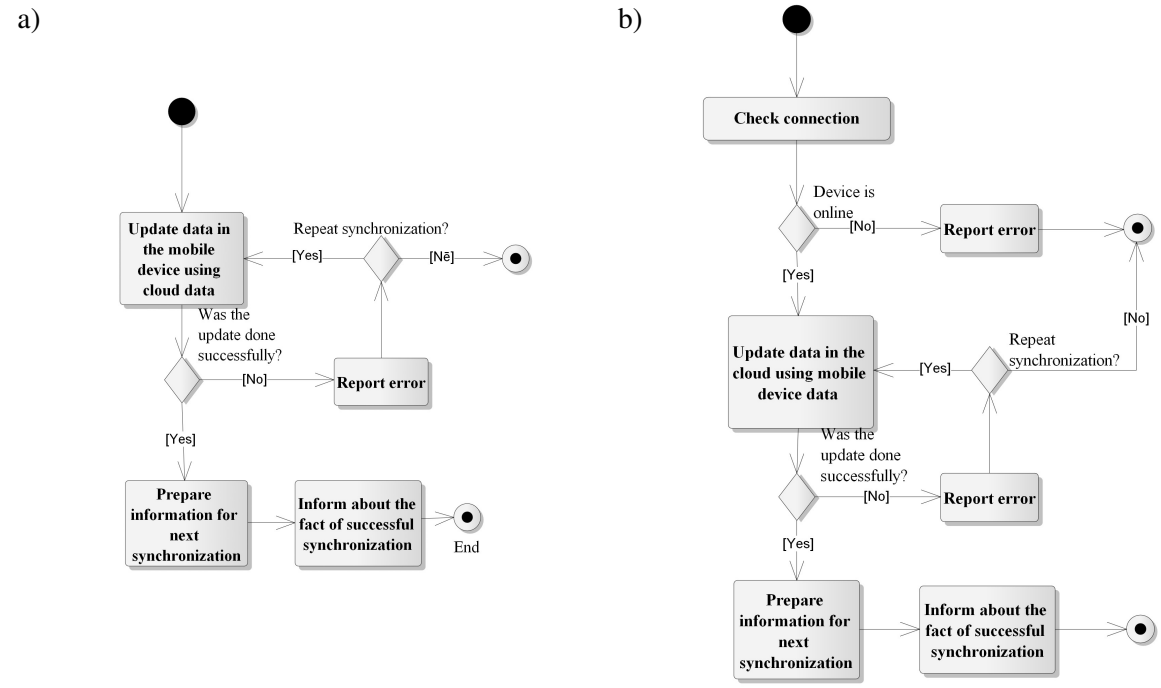

Fig. 2. **Slow synchronization scenario:** a – mobile device-cloud synchronization; b – cloud-mobile device synchronization

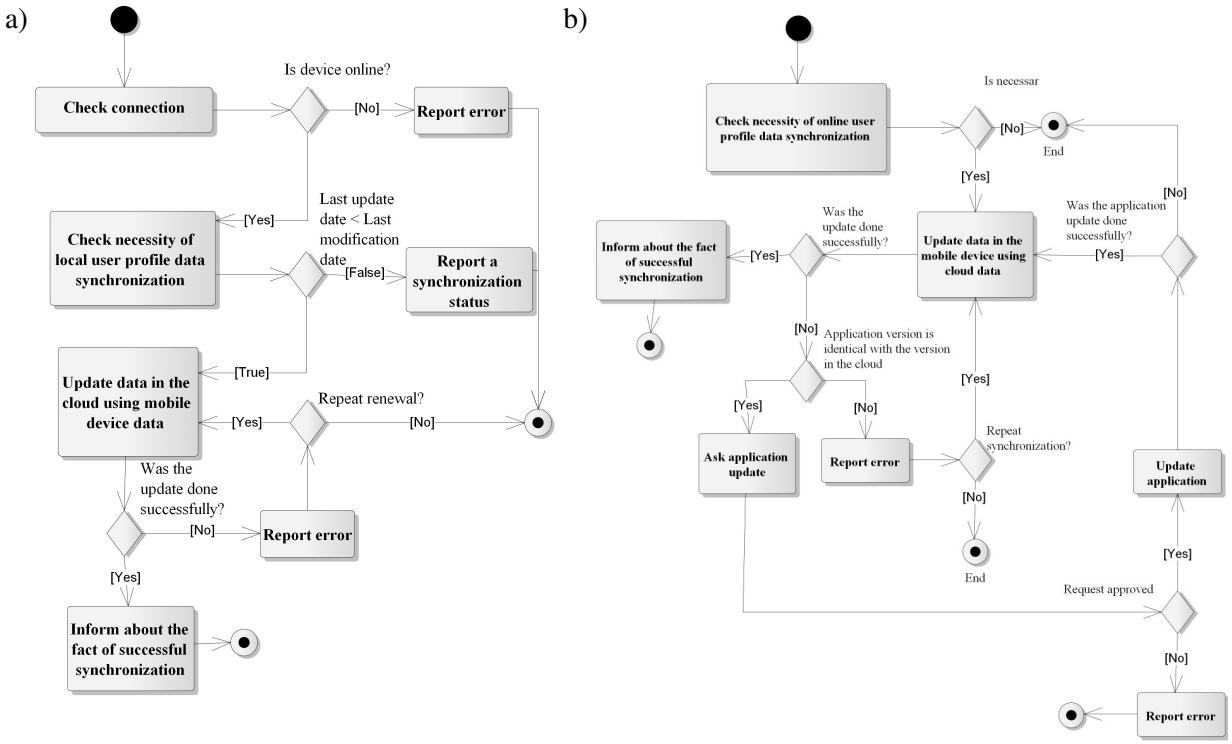

Fig. 3. **Fast synchronization scenario:** a – mobile device-cloud synchronization; 2 – cloud-mobile device synchronization

The necessity of synchronization is being verified by comparing the date of the last synchronization to the date of the last profile data change.

User profile data synchronization from the application to the cloud continues with data synchronization from the cloud to the application (Fig. 3-b). Profile data update from the cloud to the application takes a similar scenario. If a decision about synchronization necessity was performed and the user data update was successful, the user receives a notification and the last registration date changes to the current registration date.

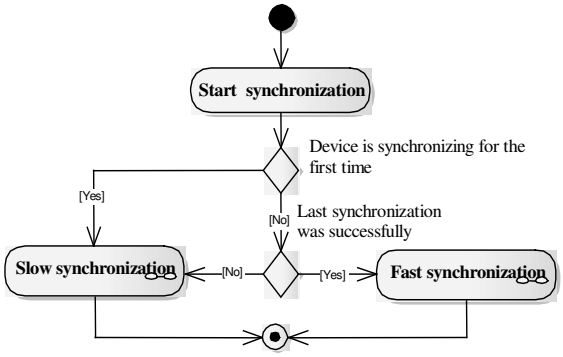

Fig. 4. **Decision of synchronization scenario**

The choice, which type of synchronization to initialize, is being done at the initiation of application (see Fig. 4). In case of the first application initiation slow synchronization is always being initialized. However, slow synchronization is also initialized if fast synchronization was processed unsuccessfully.

Results and discussion

There is no application to process the payment for parking places implemented in Jelgava. After the research was done, the decision to develop application named *J-Parking* was made (Fig. 5). There are two possible options to develop applications similar to *J-Parking*:

- new application development;
- existent application adaptation for Jelgava city needs.

Fig. 5. **J-Parking application interface**

It is possible to create a unified system, which fully conforms to the customers' requirements, by building the application from the scratch. However, building of such system will require additional resources, such as qualified developers. Implementation of the adaptation of the existing system may cause problems with functional integration.

It is necessary to ensure a quick and effective synchronization process for application successful functionality. Optimum option is to provide synchronization through using profile data storage in the cloud, which is maintained by IT Company or IT department of country institutions. For the maintenance of the cloud in Jelgava city the following could respond: IT department of Jelgava City

Council, Jelgava City Council Traffic Department or other company, which has a transferred data cloud maintenance function (Fig. 6).

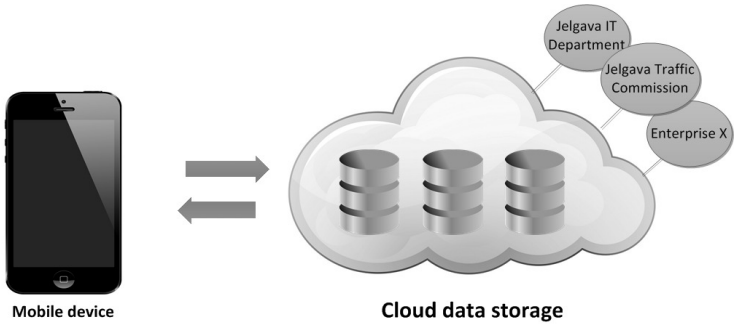

Fig. 6. **J-Parking synchronization scenario**

Synchronization can be activated by the following events:

- during application transmission time (then the same event happened);
- after a defined timestamp (timer counter);
- manually - the user initializes synchronization.

For development of J-Parking, the hybrid application development approach is used. It is easy to adapt hybrid application for other city needs. *Gartner* predicts [8], that due to increase of smartphone popularity, by 2016 50% of the available mobile applications in the market will be hybrid. *Gartner* also recommends [9] companies to develop hybrid mobile applications for business needs. One of the most well-known frameworks for hybrid application development is *Adobe PhoneGap* [10], [11].

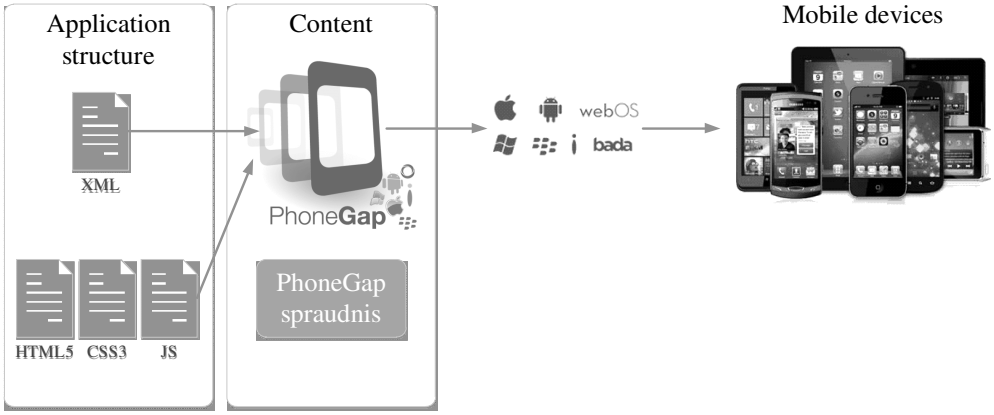

Fig. 7. **Hybrid application development using PhoneGap framework**

Using the *PhoneGap* framework [12], applications are generated to *Android*, *iOS*, *Windows* platforms (Fig. 7). To test the application functionality, the cloud technology is implemented in a local server.

Conclusions

1. To ensure application availability, it is necessary to publish it at least in *AppStore*, *Google Play* and *Windows Phone* application distribution stores, and generate application for at least three platforms: *iOS*, *Android* and *Windows*. Using the hybrid application development approach, the application development costs reduce and the opportunity to adjust application for other cities increases, because there is no need to invest in a professional programmer, which can program in platform native programming language.
2. *J-Parking* application was prototyped as a typical vehicle parking payment application. Therefore, it can be used as a base for development of applications with similar functionality.
3. In situation, when the mobile device is lost or broken, it is necessary to restore all active data. It can be done by using the user profile data synchronization.

Acknowledgement

This research is part of the project ”Competence centre of information and communication Technologies” run by IT competence centre, contract No. L-KC-11-0003, co-financed by European Regional Development Fund.

References

1. Mobilly home page. [online][25.03.2014]. Available at: https://www.mobilly.lv
2. RigaParking Application in Google Play. [online][15.03.2014]. Available at: https://play.google.com/store/apps/details?id=lv.amberphone.rigaparking
3. Pachal P. Apple vs. All the Rest: Every Mobile App Store Compared, 2011. [online][26.03.2014]. Available at: http://www.pcmag.com/article2/0,2817,2382944,00.asp
4. Becker A., Mladenowa A., Kryvinska N., Strauss C. Evolving Taxonomy of Business Models for Mobile Service Delivery Platform. Procedia Computer Science, vol. 10, 2012, pp. 650-657.
5. Thilakanathan D., Chen S., Nepal S., Calvo R., Alem L. A platform for secure monitoring and sharing of generic health data in the Cloud. Future Generation Computer Systems, vol. 35, 2014, pp. 102-113.
6. Flores H., Srirama N. Mobile Cloud Middleware. Journal of Systems and Software, 2013.
7. The OMA SyncML Common Specifications. [online][07.03.1014.]. Available at: http://technical.openmobilealliance.org/Technical/release_program/SyncML_v1_2_2.aspx
8. Rivera J., van der Meulen R. Gartner Says by 2016, More Than 50 Percent of Mobile Apps Deployed Will be Hybrid. 2013. [online][18.03.1014.]. Available at: http://www.gartner.com/newsroom/id/2324917
9. van der Meulen R., Rivera J. Gartner Recommends a Hybrid Approach for Business-to-Employee Mobile Apps. 2013. [online][22.03.1014.]. Available at: http://www.gartner.com/newsroom/id/2429815
10. Heitkötter H., Hanschke S., Majchrzak T. Evaluating Cross-Platform Development Approaches for Mobile Applications. Web Information Systems and Technologies: 8th International Conference, WEBIST 2012, Porto, Portugal, April 18-21, 2012, Part II. Porto: Springer Berlin Heidelberg, 2013, pp. 120-138.
11. Vitols G., Smits I., Bogdanov O. Cross Platform Solution for Integration of Websites into Mobile Applications. Proceedings of the 15th International Conference on Enterprise Information Systems ICEIS 2013, Angers, France, vol. 2, July 4-5, 2013, pp. 324-328.
12. Vitols G., Smits I., Bogdanov O. Cross-platform solution for development of mobile applications. ICEIS 2013 - Proceedings of the 15th International Conference on Enterprise Information Systems, July 4-5, 2013, Angers, France, vol. 2, pp. 273-277.