Implementation of Android Application for Faculty Employees

Sofija B. Purić, Uroš P. Romić, and Boško D. Nikolić, *Member, IEEE*

***Abstract* — The paper describes the functionality and implementation of applications for mobile phones used in the School of Electrical Engineering at the University of Belgrade in the daily work of faculty employees. The application uses a system's shared data for financial and material accounting, human resources and teaching process. The system was implemented using a REST Web service, Google's model for Android REST client applications and Robospice technologies.**

***Keywords* — faculty Web services, Android application, Robospice, REST Web services.**

I. Introduction

Students, teachers and employees of the School of Electrical Engineering at the University of Belgrade, have been using contemporary information technologies for their daily tasks for the past several years [1]. Review and entry of data is performed by using Web based faculty services [2], within the IIS (Integrated Information System).

Implemented services use mostly the data that is retrieved from two IIS sub-systems: FIS (Faculty information System), used for monitoring and organizing the teaching process at the faculty, and FIMES system, used for personnel records, financial and operational activities of faculty employees. The current version of above information system is based on open source solutions – Java Web technologies based on Spring technology and PostgreSQL database. Over the past several years, the functionality of Web based applications for students and employees was significantly expanded, while the number of active users has been increased several times – the system is currently in use at more than thirty colleges and universities in Belgrade, Novi Sad, Novi Pazar, Banja Luka and other cities in Serbia and the region.

Paper received April 8, 2015; revised May 22, 2015; accepted May 25, 2015. Date of publication July 15, 2015. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Jovan Đorđević.

This paper is a revised and expanded version of the paper presented at the 22th Telecommunications Forum TELFOR 2014.

Sofija Purić – School of Electrical Engineering, University of Belgrade, Bulevar Kralja Aleksandra 73, Belgrade, 11020, Serbia (email: sofija.puric@ etf.bg.ac.rs).

Uroš Romić – School of Electrical Engineering, University of Belgrade, Bulevar Kralja Aleksandra 73, Belgrade, 11020, Serbia (e-mail: uros.romic@ etf.bg.ac.rs).

Boško Nikolić – School of Electrical Engineering, University of Belgrade, Bulevar Kralja Aleksandra 73, Belgrade, 11020, Serbia (e-mail: bosko.nikolic@ etf.bg.ac.rs).

On the other hand, in recent years there has been a significant development and popularization of mobile devices of the third generation, especially smartphones, which are increasingly used as an alternative to desktop and laptop computers to access content on the Internet. This development of technology and the increase in the number of users and their needs have created conditions for implementing the ability to access certain functionalities via mobile devices, within the IIS. Other papers in this field also described the benefits of using Android applications in educational process at faculties [3], [4].

The purpose of this paper is to demonstrate the ability of selected modern technologies to implement a complex mobile application, which relies on previously implemented systems and has the highest level of protection. In the following chapter, software environment of the application is described and the reasons that influenced the choice of the most suitable software solutions for this kind of application are stated. After that, application functionalities and implementation are described. Finally, evaluation results regarding most common application functionalities are given, representing the expected execution acceleration achieved by using Robospice library [5].

II. Software environment of the Android application for employees

Within the IIS, a Web application for employees (called eZaposleni), based on Java and Spring Web technologies, is used [6]. The application communicates with other applications within the system by RPC (remote procedure calls), using Spring HTTP (Hypertext Transfer Protocol) invoker [7].

Considering the fact that the Web applications for employees and students (eZaposleni and eStudent) are externally available and that they already have access to the necessary information to the other IIS sub-systems (FIMES and FIS) by RPC, it was decided that Web services would be created within these applications, by which the data retrieved by RPC would be delivered to the applications for mobile devices. Thus, the existing code in Web applications is utilized, and also the security of the system was not affected, as FIMES and FIS, which are not publicly available, were allowed to remain in the protected area of the network.

Due to a client-server nature of the communication between the mobile device and the application which exposes the Web service, and also considering the benefits of using the HTTP protocol, as well as the need for minimizing the data transfer, a REST-based

doi:10.5937/telfor1501055P

(Representational State Transfer) Web service has been chosen as the optimal solution [8]. For this Android application [9], the data is delivered in JSON (JavaScript Object Notation) format, and to simplify the client-side deserialization, the client application uses the same Java library as the application which exposes the Web service, containing the same JavaBean classes which are used for data transfer [10]. XML (Extensible Markup Language) format may be used instead of JSON, but it is far more massive, complicated (in terms of parsing) and contains a huge quantity of useless "tag-shaped" data, which represents a significant overhead during client-server data transmission. That is why the JSON was chosen for the described system.

The functionalities that were made available through the Web services are limited primarily to those that require minimal input data on the client side. It is assumed that the input of the large amounts of data or using the time consuming operations (for example entering the points and grades for a large number of students) would be more comfortable for users if they remain within the existing Web based application accessed via browser on desktop computer.

The above REST Web services were implemented using Java JAX-RS (Java API for RESTful Web Services) interface and Jersey referent implementation [11]. For Jersey and Spring technologies integration, Java libraries of jersey-spring module were imported [12]. It should be noted that Jersey implementation version 2 enables the wiring of JAX-RS classes and Spring Beans by annotation only. Considering that all dependencies between Spring beans in eZaposleni application are defined using XML configuration files, Jersey 1 implementation (version 1.17.1) was used.

In order to support conversion of classes into JSON strings within the application, Jersey-json module Java libraries should be imported into project.

Authentication and authorization of the requests to REST resources is implemented using BASIC authentication over HTTPS (HTTP on top of SSL/TSL security protocol). Security configuration parameters are defined within Spring Security model.

Within the application, utility methods are used for conversion of Java objects retrieved by remote calls, into objects specifically designed for this purpose. These are JavaBean classes modified for Web service functionality. It facilitates the deserialization of JSON data received by client application, and objects that are created can be further used for the presentation of data to the user.

III. Review of Employee Android Application Functionalities

Implemented application functionalities are divided into three groups: "employee", "teaching process", and "exams". Within the employee related functionalities, personal record of user and monthly rosters reviews are available. "Teaching process" functionalities include a review of user's course engagement, search of class timetable, and search of students. "Exams" functionalities include a review of exam schedule for a chosen examination term and lists of students registered for exams in courses that a current user is engaged into.

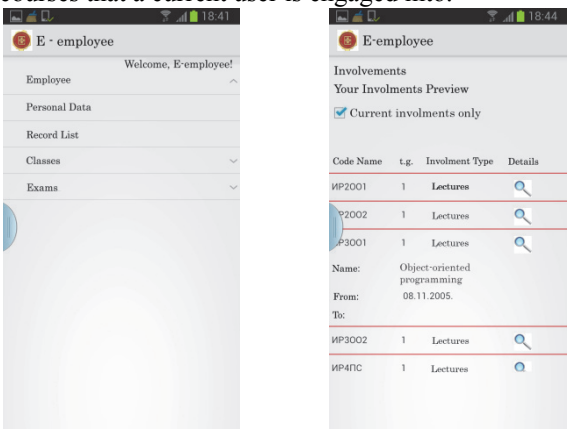

Fig. 1.a) Main menu b) Course engagement.

In the following text, there is a detailed description of each functionality.

After the application is started, the initial page is displayed, requiring user authentication. The application requires a working internet connection. If the connection is lost at any time, the user is notified and further use of application is disabled. If user authentication is successful, the main screen is displayed (Fig. 1 a). Otherwise, user will be shown an authentication error message.

For displaying user information, two tabs are used. The first one displays personal data, and the second one contains user contact information.

For a monthly roster review, the user must first select the year and month, and then the new screen is displayed containing requested information, divided into four tabs: coefficients, hours, extra hours and other data.

After selecting the course engagement option within the "teaching process" category, the screen with the list of user's course engagements is displayed (Fig. 1 b). By clicking on the individual engagement, detailed information about it is displayed, and clicking the "details" image next to the listed engagement, opens a new screen with the information about the selected course (Fig. 2 a). Course information is divided into three tabs, the first one containing information about teaching groups, the second one displays teaching units, and the third one displays the number of classes for the selected course.

Search of courses is accessible by selecting the appropriate option in "teaching process" category. The user must enter at least three characters into a search input text field. After that, the dropdown list is displayed containing all courses which meet search criteria (Fig 2 b). The user can select each of the listed courses, to display additional data about it in a new screen.

A class timetable review contains the following review types: user's own timetable, timetable for a selected student, or for a selected course, teacher, study profile, or classroom. After choosing the review type, selecting the criteria and clicking the submit button, the screen with a class timetable is displayed, for a one-week period. Each term is clickable, displaying detailed information about classes in a selected term (Fig 3 a).

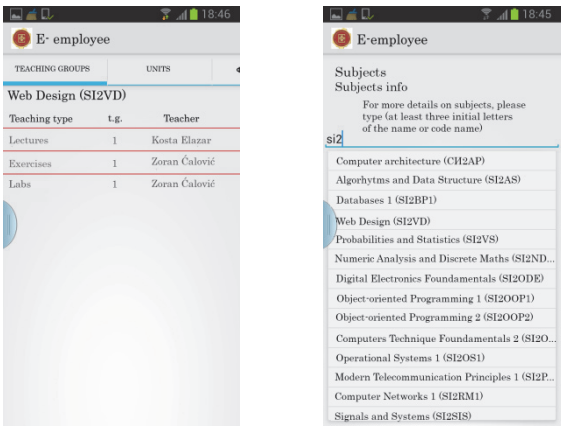

Fig. 2.a) Course information b) Search of courses.

The student search is also available in "teaching process" options category. The student search request can contain one or more of the following criteria: year or index number, name, last name, type of study and whether the student is active or not. By clicking the submit button, the search is performed and the list of the students which meet the criteria is displayed. By clicking on each of the listed students, a new screen is displayed containing detailed information about a selected student (Fig 3 b).

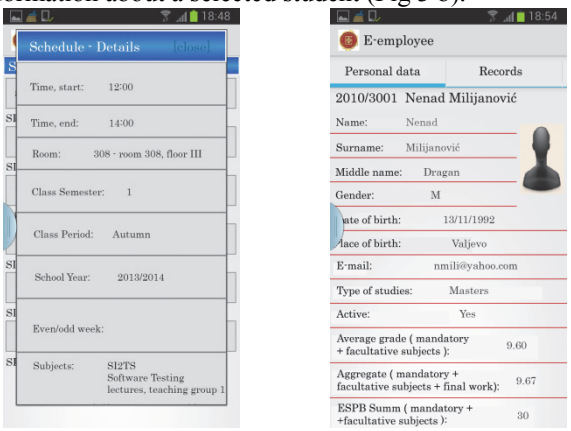

Fig. 3.a) Detail about class b) Detail about student.

Student details are divided into several tabs displaying basic information and data about enrollment, exams passed, exams failed, exams that a student is allowed to apply to, exam applications, course engagement, obligations and the final exam of a selected student.

By choosing the "examination terms" option of the "exams" category, a list of active examination terms is displayed. For each term, "mine" and "all" links are available, displaying lists of exams for courses a user is engaged into, or all exams in a selected term, respectively.

A review of students registered for exams in courses that a user is engaged into is also available within the "exams" category. Upon selecting this option, the list of exams is displayed, and further details about each exam are accessible by clicking on the exam name. The complete list of students registered for exam is displayed by selecting "list" option for each exam. User may click on each student to get detailed information, the same as within the search option.

The information about application itself is also available (by selecting About us option), as well as a logout option that shuts down the application. If no user action is performed for more than 20 minutes, the user is automatically logged out and the notification is shown.

The navigation within the application, in addition to selecting the desired option, is also available via a Back button, which displays the previous screen to the user. If the user returns to the initial screen and clicks the back button once again, a confirmation dialog is displayed, asking the user to confirm or cancel the application shutdown. Thus, there will be no automatic logout by pressing the Back button accidentally.

IV. Android Application Implementation

A. REST

Application eZaposleni is an Android REST client application. Its core functionality is data exchange with Java Web application for employees using REST Web services and presenting the retrieved data to an authenticated user.

REST is a client-server architecture in which a client creates and sends a request to the server to retrieve or update a single resource. The server responds by passing the resource representation to the client [13]. Different HTTP methods are used for different CRUD (create, read, update and delete) operations: GET represents a read operation, PUT is used to change the state of a resource or to update it, POST is used to create a resource and DELETE operation is used to remove resources. REST is stateless, i.e. the client context is not stored on the server side, but all state representations are passed from the server to the client and vice versa.

B. ROBOSPICE

Robospice is an Android open source library which simplifies writing of an asynchronous task with prolonged execution, and complies with the Google's principles for the proper implementation of Android REST client applications.

For executing REST requests and communication to Web services, Spring Android module is used, which is supported by Robospice library. Communication itself is realized by using POJO (Plain Old Java Object). POJO is a term used to describe any Java class that doesn't extend any other class nor implements any interface. These are utility classes supporting the application logic.

Robospice library and Spring Android REST module enable the execution of the request to the Web server, conversion of resulting JSON strings in POJO objects, caching them, controlling the expiry of the data in the cache and notify applications when the result is ready. To use Robospice within the application the following steps should be taken:

1) Creating Spice Service – this step should be performed once for all request, to enable their execution. If Spring Android library is used for parsing JSON results, converting them into POJO, and for caching the resulting POJO object by using JSON, a subclass of JacksonSpringAndroidSpiceService must be defined and declared in AndroidManifest.xml configuration.

The relevant parts of the source code are presented in Fig 4. Basic authentication and gzip compression is added

to Spice Service. Application class is a singleton which contains the connection properties.

```
public class
EzaposleniJacksonSpringAndroidSpiceService extends
JacksonSpringAndroidSpiceService {
   @Override
   public CacheManager createCacheManager(Application
   application) {
      CacheManager cacheManager = new CacheManager();
      JacksonObjectPersisterFactory
      jacksonObjectPersisterFactory = null;

try {
        jacksonObjectPersisterFactory = new
        JacksonObjectPersisterFactory(
        application);
      } catch (CacheCreationException e) {
        e.printStackTrace();
      }
      cacheManager.addPersister(
      jacksonObjectPersisterFactory);
      return cacheManager;
   }
   @Override
   public RestTemplate createRestTemplate() {...}
}
```

Fig. 4. Spice Service creation.

2) Core class BaseSpiceActivity should be defined. Other classes that interact with the server extend this class.
3) Any activity that interacts with the server may have one or more methods. The example of a method which is executed after a specific user action is given in Fig 5. Within the method, the request to the server is created. The cache key for that request is also created, which represents the unique value used to identify the request and retrieve it from the cache. After that, the execute method of the cache manager object, retrieved from base class (BaseSpiceActivity), is called which is shown in Fig 6.:

```
private void performRequestLicniPodaci() {
     KorisnikLicniPodaciRequest request = new
     KorisnikLicniPodaciRequest();
     lastRequestCacheKeyLicniPodaci =
     request.createCacheKey();
     getSpiceManager().execute(request,
     lastRequestCacheKeyLicniPodaci,
     10000, new KorisnikLicniPodaciRequestListener());
}
```

Fig. 5. Request executing method.

This method exectutes the request which is passed as the first argument of the method. Before the *SpiceRequest.loadDataFromNetwork()* method call, the cache is checked, and if a result is cached by the key same as the requestCacheKey (the second method argument), Robospice checks the third argument – cacheExpiryDuration (its value is given in miliseconds) to determine if the cached data is valid or not. If the cached data hasn't expired, it will be retrieved form cache. Otherwise, *SpiceRequest.loadDataFromNetwork()* is called, and the resulting data will be put into cache using requestCacheKey The last parameter of the execute method is a listener which should be notified after the request is completed.

```
public <T> void execute
           (SpiceRequest<T> request,
            Object requestCacheKey,
            long cacheExpiryDuration,
            RequestListener<T> requestListener)
```

Fig. 6. Execute method.

```
public class BaseSpiceRequest<T> extends
SpringAndroidSpiceRequest<T> {
   public BaseSpiceRequest(Class<T> clazz) {
      super(clazz);
   }
   public String getBaseUrl() {
      String baseUrl = "http://"
           + ApplicationEzaposleni.getInstance().getUrl()
           + ":" +
           ApplicationEzaposleni.getInstance().getPort();
      return baseUrl;
   }
   @Override
   public T loadDataFromNetwork() throws Exception {
      return null;
   }
   public String createCacheKey() {
      return null;
   }
}
```

Fig. 7. Base class BaseSpiceRequest.

4) One SpiceRequest class should be created for each request that is sent to the server. These classes differ by the type of the return object. Therefore, one base class is created which will be extended by all other SpiceRequest classes. This base class is shown if Fig. 7.

An example of the request class that shoud be created for each type of request is shown in Fig.8. In this example, the class that will be returned is FimesOsobaWeb, one of the above mentioned POJO. The request class returns the POJO converted into JSON string, while the conversion itself is controlled by framework. The method loadDataFromNetwork() targets the specific URL which has a REST service defined and which returns the resource that was requested. The createCacheKey() method returns the unique key for each resource which will be used to retrieve the resource from cache when needed.

```
public class KorisnikLicniPodaciRequest extends
BaseSpiceRequest<FimesOsobaWeb> {
   public KorisnikLicniPodaciRequest() {
      super(FimesOsobaWeb.class);
   }
   @Override
   public FimesOsobaWeb loadDataFromNetwork() throws
   Exception {
      String urlProba = super.getBaseUrl()
           + "/rest/korisnikPodaci/getKorisnik";
      return getRestTemplate().getForObject(urlProba,
      FimesOsobaWeb.class);
   }
   @Override
   public String createCacheKey() {
      return "getKorisnikLicniPodaci";
   }
}
```

Fig. 8. Extended Request class for each request type.

5) One or more inner classes that will represent Request Listeners should be created within the activities that interact with the server. The service will return the results of completed requess from the server to these classes, and the listener classes will update the user interface inside the Main thread which executes the application. One example of Request Listener class is shown in Fig. 9.

C. Structure

In this chapter the most important UML (Unified Modeling Language) diagrams will be shown in order to completely clear up the way the system operates. Here, we will show the state diagram (Fig. 10), the sequence diagram for a typical activity in the system (Fig. 11) and the POJO class diagram that are used in order to get a response from the server (Fig. 12).

```
private     class     KorisnikLicniPodaciRequestListener
implements RequestListener<FimesOsobaWeb> {
     @Override
     public void onRequestFailure(SpiceException arg0) {
        Toast.makeText(MainActivity.this,
             "Неуспешно дохватање личних података!",
             Toast.LENGTH_LONG).show();
     }


     @Override
     public void onRequestSuccess(FimesOsobaWeb osoba) {
        Intent ourIntent = new
        Intent(getApplicationContext(),
        PersonalActivity.class);
        ourIntent.putExtra("Osoba", osoba);
        startActivity(ourIntent);
     }
}
```

Fig. 9. Private class Request Listener.

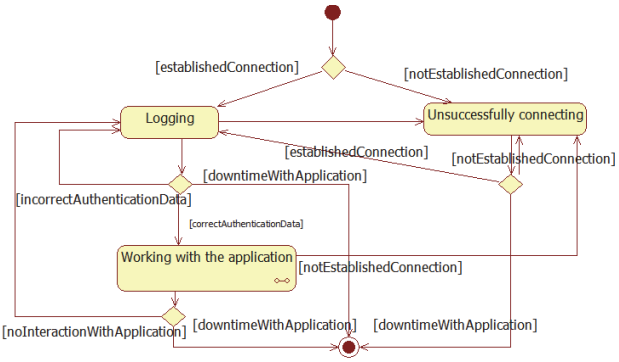

Fig. 10. State diagram.

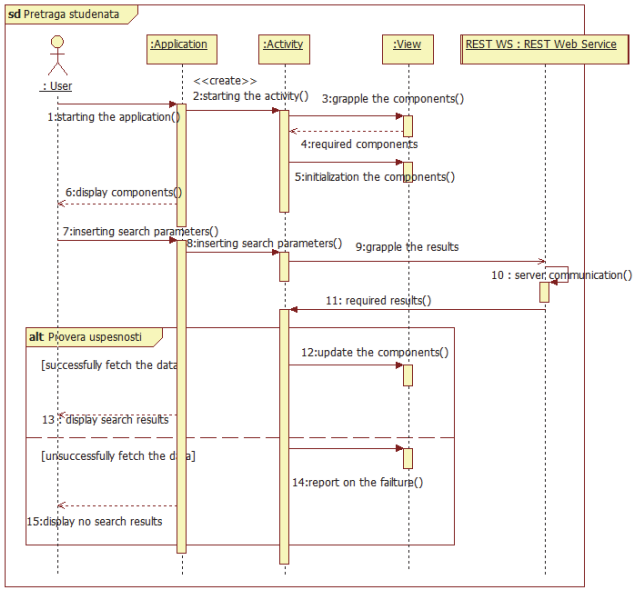

Fig. 11. Sequence diagram.

V. Evaluation

In this chapter the evaluation results will be shown represented in a table where we shall see time duration in milliseconds for the executions of various functionalities of the Application when the data are obtained from the server (column – execution time 1), when the data are obtained from the cache memory for the first time (column – execution time 2) and when the data are obtained from the cache memory for the second time (column – execution time 3) (Table 1). The application testing was conducted at cell phone Samsung, model S III Galaxy. The operating system was Android 4.0.3, 16GB of internal memory, 1GB of RAM memory, CPU: Quad-core 1.4GHz Cortex-A9.

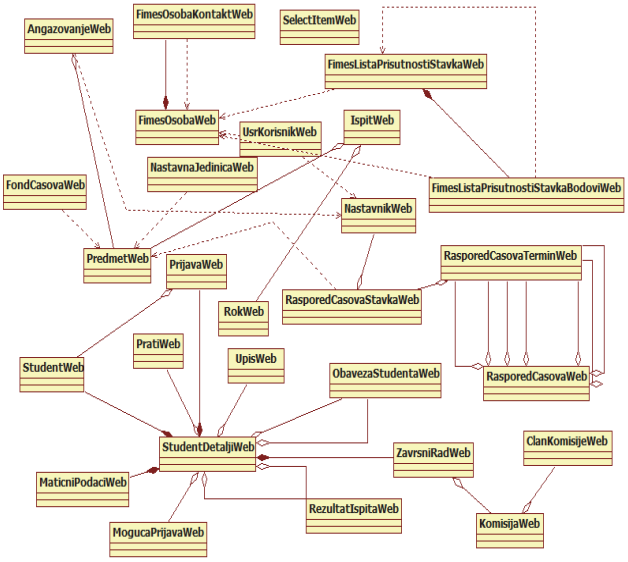

Fig. 12. POJO class diagram.

TABLE 1: TABLE WITH EVALUATION RESULTS.

Functionality	Execution time 1[ms]	Execution time 2[ms]	Execution time 3[ms]
Personal data	589	193	42
Record lists	1146	459	74
Engagement	1446	339	45
Getting the subject list	2630	2394	1637
Getting the subject	2773	320	279
Getting the schedule	1491	549	84
Getting the students' list	3817	879	822
Getting the student	2417	796	45
Signed up students	3763	460	137
The list of signed up students	8752	857	635

Besides this table, we will show the chart (Fig. 13) which shows the acceleration of application execution. It can be seen that the execution time (in milliseconds) shown for different functionalities is significantly shorter when the data is retrieved repeatedly. This acceleration was achieved by using Robospice library, which allowed the data caching and getting the data from the cache memory (execution time represented by red and blue lines) once it was retrieved from the server (represented by a green line).

Based on the analysis of the obtained results, it can be concluded that the application meets the requirements in terms of performance. The effect of applied technologies is particularly noticeable when a large amount of data is requested (for example, when the list of signed up students is retrieved).

VI. Conclusion

The objective of this paper is to demonstrate the abilities of chosen contemporary technologies in the implementation of a complex mobile application, which relies on the existing systems and has the highest level of security. The described application is a part of integrated information system which is currently in use at more than thirty colleges and universities in Belgrade, Novi Sad, Novi Pazar, Banja Luka and other cities in Serbia and the region. Application is implemented using modern technologies, such as REST Web services, Google's Android REST client model, and Robospice libraries.

The future steps in the development of system include the implementation of the same functionalities for other operating systems for mobile devices, such as Microsoft 8 and iOS. The integration of the implemented system with software tools for learning is also planned.

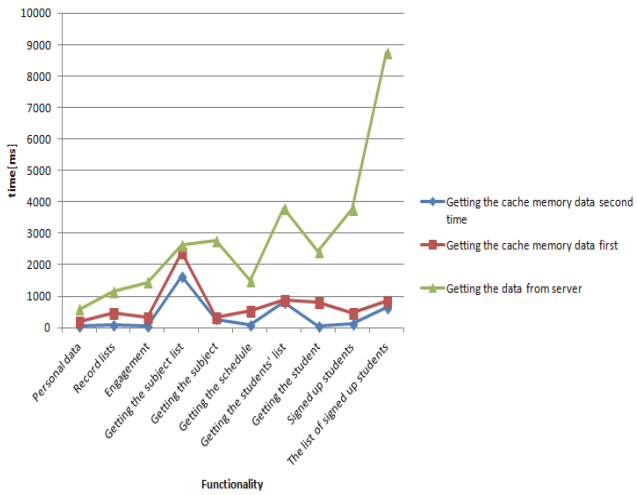

Fig. 13. Chart with evaluation results.

References

[1] Računski centar Elektrotehničkog fakulteta, Fakultetski servisi (2014), http://rc.etf.bg.acrs/?p=web_fakultetski_servisi

[2] Romić U., Manić I., Using Spring-based Tools in Development of Java Web Application for Faculty Employees, *Proceedings of the 17th Telecommunications Forum TELFOR 2009*., pp. 1295-1298, Telecommunications Society - Belgrade, Belgrade, Serbia, Nov, 2009.

[3] H. Ahuja, R. Johari, "Optimization of the issues in the migration from Android Native to Hybrid Application: Case study of Student's Portal application", *Advances in Computing, Communications and Informatics (ICACCI, 2014 International Conference on)*, 2014, pp. 245-249, DOI: 10.1109/ICACCI.2014.6968365

[4] Z. Ali, R. Ismail, "Design and development of Android mobile application for students of engineering education in Saudi Arabia", *Information Society (i-Society) , 2013 International Conference on*, 2013, pp 228 -233

[5] Repo of the Open Source Android library : RoboSpice, JavaDoc Available: http://stephanenicolas.github.io/robospice/site/latest/apidocs/com/octo/android/robospice/SpiceManager.html, [Dec, 2013]

[6] U. Romić, I. Manić, I. Pantelić, "Contemporary Java Web Technologies as a Service for the University Employees*", Journal Of Information Technology And Applications (JITA)*, vol. 2, no. 2, pp. 88-94, Dec, 2012.

[7] R. Johnson, J. Hoeller, K. Donald, Others, "Remoting and web services using Spring." in *Spring Framework Ref. Documentation*, ch. 19, Available: http://docs.spring.io/spring/docs/3.0.x/spring-framework-reference/html/remoting.html, 2010. [Apr, 2014].

[8] F. Aijaz, M. Chaudhary, B. Walke, "Performance comparison of a soap and rest mobile web server", *Proc. of the Third International Conference on Open-Source Systems and Technologies (ICOSST 2009)*, 2009.

[9] Shih-Hao Hung, Yong-Wei Chen, Jeng-Peng Shieh, "Creating Pervasive, Dynamic, Scalable Android Applications", *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2013 Seventh International Conference on*, 2013, pp. 43 -50, DOI: 10.1109/IMIS.2013.17

[10] Android Developers, Application Fundamentals, Available: http://developer.android.com/guide/components/fundamentals.html, [Nov, 2013]

[11] M. Hadley and P. Sandoz, "JAX-RS: Java API for RESTful Web Services", Java Specification Request (JSR), vol. 311, 2009.

[12] "Jersey 1.18 User Guide", Available: https://jersey.java.net/documentation/1.18/user-guide.html, Mar, 2014. [Apr, 2014]

[13] "RESTful Web services: The basics", Available: http://www.ibm.com/developerworks/webservices/library/ws-restful, [Mar, 2014]