Wireless Services in Faculty Information System

Sava Mikalački, Žarko Bodroški, and Srđan Škrbić

Faculty of Science, Trg Dositeja Obradovića 3
21000 Novi Sad, Serbia
sava.mikalacki@dmi.uns.ac.rs
{shkrba, bodroskiz}@uns.ac.rs
http://www.is.pmf.uns.ac.rs

Abstract. The ongoing process of implementation of the information system of the Faculty of Science in Novi Sad has entered a new phase in which wireless services for mobile devices have been designed and implemented. In this paper we briefly sketch the architecture of the information system and describe how wireless mobile services are attached to it. We give the most important pointers on choice of technologies and discuss the most interesting implementation considerations. Key challenges and lessons learned in the design and implementation processes are outlined in the text.

Keywords: information system architecture, J2ME, wireless

1 Introduction

Handheld computers, Web-phones and other mobile devices are becoming important components in the ongoing trend towards a pervasive Web [1]. Complex information systems, such as the information system of the Faculty of Science in Novi Sad, are increasingly required to support the delivery of information to mobile devices. To meet the needs of the many users who might access its services using these devices effective user interfaces and communication methods need to be developed [2]. Handheld browsers have very different interaction characteristics to their desktop counterparts and these have to be considered [3].

In this paper we present the most important problems and lessons learned in the attempt to create wireless service system as part of the existing information system. In the first part of the paper we give some pointers on choices of technologies we made before the implementation process. The second part briefly describes the existing architecture of the information system and presents the way the mobile services are fitted into it.

2 Implementation Considerations

Java 2 Micro Edition (J2ME) is a specification of a subset of the Java platform aimed at providing certified collection of Java APIs for the development of software for small and resource constrained devices like mobile phones, PDAs etc. J2ME devices that implement this specification are based on configurations and profiles.

A configuration is a specification that defines the software environment for a range of devices defined by a set of characteristics such as the type and the amount of the available memory, processor speed etc. Currently there are two configurations: CLDC (Connected Limited Device Configuration) and CDC (Connected Device Configuration). Typical CLDC platform is a mobile phone or PDA with around 512K of free memory. On the other hand, CDC addresses devices with higher performance characteristics such as smart phones, high-end PDAs etc.

Profiles are more specific than configurations. A profile is based on a configuration and provides additional APIs such as user interface, persistent storage etc. Depending on the configuration and the device type we have different profiles. FP (Foundation Profile) is a specification for devices that can support rich networking environments. It does not support user interface, but other profiles can be layered on top of it to add user interface support and other functionalities. PDAP (PDA Profile) for developing applications for PDA devices that meet hardware requirements for this profile. At last, there is MIDP (Mobile Information Device Profile) which is a specification for developing applications for resource constrained devices like mobile phones, pagers etc. [4]

Having all this in mind, the targeted configuration for our Faculty's needs would be CLDC configuration with MIDP profile and these requirements are fulfilled by almost every mobile device on the market today. Current versions of CLDC and MIDP are CLDC 1.1 and MIDP 2.0. Applications written for this platform are called MIDlets, as an extension to Java naming convention like Aplets etc.

As we said earlier, profiles define some set of APIs that can be used by developers. MIDP profile defines a set of APIs that each mobile device vendor must implement, such as HTTP protocol support, persistent storage with RMS (Record Management Store) etc. Other APIs like MMAPI (Multimedia API) or BTAPI (Bluetooth API) are optional and vendors do not have to implement does APIs. In order to use Web Services, mobile device must support WSA (Web Service API). Set of these APIs enables mobile device to communicate with Web Services in a manner an ordinary PC would do. Unfortunately, WSA is defined as optional package by JCP (Java Community Process) and mobile device vendors are not obligatory to implement these APIs.

Since many mobile devices do not support WS API we had to find another solution. We found it in XML-RPC protocol (XML Remote Procedure Call). XML-RPC is an extremely lightweight protocol that invokes remote procedures over a network by sending XML formatted messages over HTTP [5]. It represents a set of implementations that allow software running on disparate operating systems and different environments to make procedure calls over the Internet.

Using XML-RPC protocol it is possible to make calls to Web Services and its methods over the Internet using mobile devices. XML-RPC uses HTTP protocol to allow communication between the client and the server. Fig. 1. describes this communication.

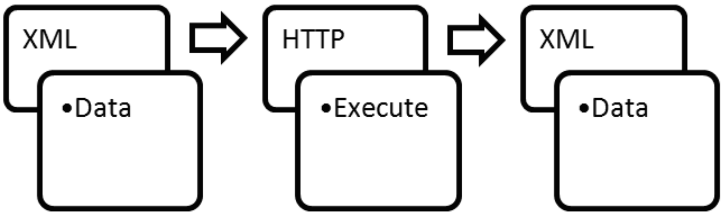

Fig. 1. XML-RPC Protocol.

Since XML-RPC protocol uses HTTP for transmitting XML messages it is possible to use it on mobile devices because MIDP profile defines HTTP protocol support as obligatory. XML messages used in communication are small which is very important when taking in consideration that XML messages will be parsed on devices with relatively slow performance characteristics. Another important thing is the size of XML messages. Since mobile devices use GPRS (General Packet Radio Service) for sending and receiving data over the Internet, users have to pay different prices for using it. Our targeted groups of users are students, so small XML messages are of great importance since their transmission doesn't cost that much. XML-RPC represents simple and effective solution to our problem of invoking Web Services from mobile device.

Another XML protocol for making remote procedure calls is SOAP (Simple Objects Access Protocol). SOAP XML messages consist of three parts: an envelope which defines what is in the message and how to process it; a set of encoding rules for expressing instances of data types and a convention for representing remote procedure calls and responses. SOAP tries to pick up where XML-RPC left off, by adding support for user defined data types, ability to specify the recipient, message specific processing control and other features [5].

REST (Representational State Transfer) is a HTTP protocol based software architecture. It is based on HTTP Put and Get methods and provides support for serialization and caching of objects. Having in mind these capabilities we came to conclusion that REST is an approach for a browser based application. Our application offers some offline services which would not be available if it was a browser based application.

Choosing which protocol to use depends on what kind of functionality do we want in our client server communication. XML-RPC is a simple lightweight protocol for making remote method calls. Basic difference between SOAP and XML-RPC is in the type and the structure of the XML message used to describe client server communication. SOAP XML messages carry more information and thus they are much more complex then XML-RPC messages. Having in mind that our client application is supposed to be running on mobile devices we decided to use XML-RPC. Parsing SOAP XML message is more time consuming then XML-RPC message. Also, SOAP XML messages are larger and therefore their transmission over GPRS is money and time consuming. For the needs of our Faculty, XML-RPC protocol is sufficient because we use only simple data types and we do not need types of information that SOAP provides in its XML messages.

This type of communication between client and the server uses HTTP protocol to transmit XML messages between two parties. J2ME enabled mobile devices do not have XML-RPC implementation as a part of their core API so we had to use some of the open source XML-RPC implementations. Having considered all of the open source implementation, we concluded that kXML implementation meets all of our requirements, thus, it is specifically designed to be used on resource constrained devices such as mobile phones. It has a small XML pull parser and it is fully compatible with our server Apache XML-RPC implementation.

Since security is one of the biggest questions that needs to be answered in any modern and complex information system so it is done in J2ME part of our information system. As we said, kXML uses HTTP protocol to transmit data between client and the server but it does not support HTTPS. Our web application has HTTPS protocol supported but unfortunately, this cannot be used as a security protocol when it comes to J2ME client application. Each mobile device has its own set of root certificates issued by different vendors to the device manufacturer. In order for HTTPS to be used as a security protocol we would have to have a certificate issued by the same certificate vendor in order to sign our MIDlet and thus use the benefits of HTTPS. Procedure for obtaining such certificate is complicated and very expensive so we decided to use other approach to the security question.

Another way to secure our client-server communication is using digital signatures as a security technique. The XML digital signature is a W3C specification. The sender can choose to digitally sign the entire document or only part of it. Digital signatures, digests, and public keys are formatted into XML elements. Those extra elements of security information can envelop the entire original XML message, or alternatively they can be embedded into the original message.

Bouncy Castle is an open source, lightweight cryptography package for the Java platform that is also optimized for J2ME Platform. Its supports a large number of cryptography algorithms and it is the only complete cryptography package that runs on the J2ME platform. Bearing in mind resource constrained environment on mobile devices, the RSA algorithm offers the best performance on wireless devices. Using it, server generates and then encodes the digest, signature and key parameters into ASCII text form and embeds the text in XML digital signature format. The verification MIDP application parses the digest, key parameters, and signature out of the XML document, reconstructs the public key, and validates the signature [6].

Since there are a variety of mobile devices on the market today with different characteristics (screen size for example) we want our client application to be able to run on all devices regardless of their characteristics. Having considered these requirements one question arises: how do we achieve write once run anywhere goal? Using an application called J2ME Polish we can achieve this goal. J2ME Polish provides a powerful framework for developing rich applications for different mobile devices. Developers can use CSS to create advanced GUI applications and therefore separate graphical design of the application from its business functionality. It also provides support for XML-RPC and many other options. It has a comprehensive mobile device database describing each device features and characteristics [7].

Client application is compiled and packaged with all third party libraries in a JAR (Java Archive) file using J2ME Polish. It also generates a JAD (Java Application Descriptor) file which contains information about MIDlet and its properties. These files are required when users want to install client application on their mobile device. Now, the question of application deployment arises.

There are several ways to deploy the application to the clients:

- *OTA* - Over The Air deployment; clients point their web browsers on their mobile devices to the JAD file of the application and they install it on the device;
- *Bluetooth* - users can download the application on their computers and then using Bluetooth install it on the devices;
- *USB cable* - same as Bluetooth, except communication between mobile device and computer is realized through data cable.

3 Architecture of Web Services in the Information System

The Faculty of Science, one of the largest faculties of the University of Novi Sad employs over 400 staff and has substantial IT requirements. Moreover, exchange of data in electronic form with other faculties, the Rectorate and Ministry offices has become a must. The IT environment at the Faculty has heterogeneous application platforms and associated servers, mainly implemented in obsolete technologies. To address the IT needs, the Faculty is developing and deploying an open source information system [8,9].

Information system of the Faculty of Science in Novi Sad has several subsystems. One of them is the student affairs subsystem that provides students with many services regarding their studies. For the purpose of this paper, we will concentrate on this segment of our information system with closer insight on mobile Web Service.

Our information system is based on a three tier architecture (Fig. 2) that consists of a database management system, an application server and a client interface.

Application server is divided into two layers:

- XML-RPC Server, and
- Enterprise Java Beans 3.0 layer.

Enterprise Java Beans 3.0 layer is responsible for providing communication with database. Applications written using EJB 3 technology are reliable, scalable and highly secure. This is of prime importance for developing complex systems that must preserve their functionalities and maintain consistent despite the large number of users at the same point of time. EJB 3.0 technology provides object-relational mapping, which means that users are not working with the database directly. Instead, they are working with Java object representation of the database, and when all the changes are made, only then they are stored in the database [8,10].

All the methods communicating with the database in our system are stored in session beans. Thus, it is only necessary to call the particular method from the session bean and to return the result directly to the client [9]. In order for J2ME client application to be able to communicate with the EJB 3.0 layer an adequate communication protocol is required.

Using XML-RPC protocol, we provide flexible and easy to use protocol for data transmission.

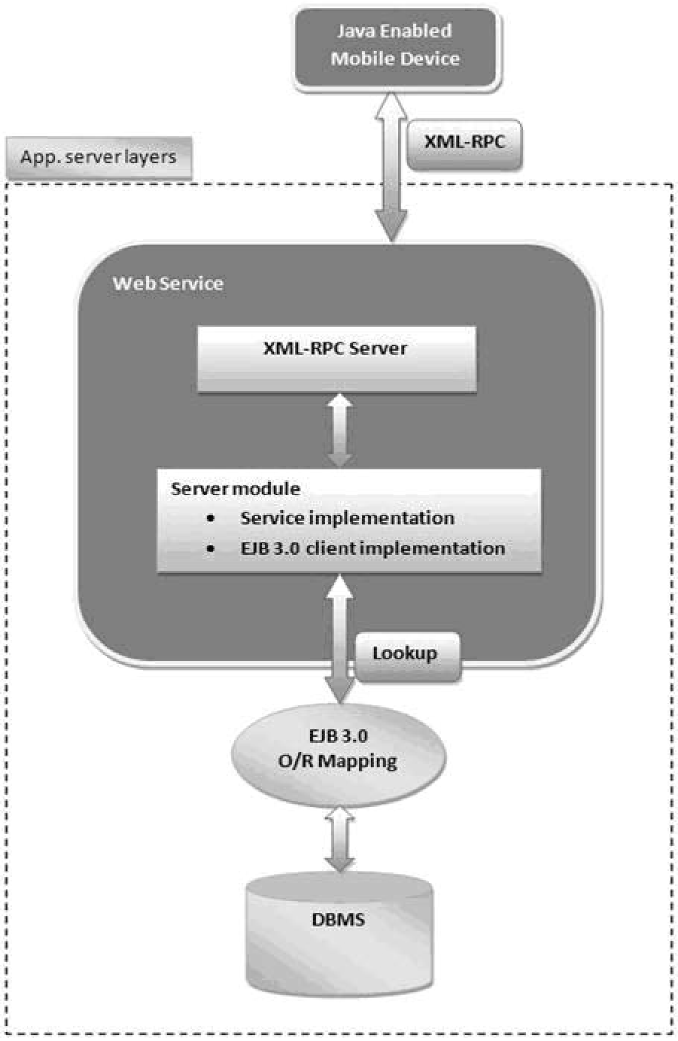

Fig. 2. Information system of the Faculty with J2ME client.

XML-RPC server is a part of our application server layer. Using Apache XML-RPC implementation, we create a servlet that acts as a service provider to our J2ME client, allowing it to invoke session bean methods. We provide specific session beans that contain methods required by are J2ME client to the XML-RPC servlet class which is then responsible for client server communication.

This communication is being realized in four steps:

1. J2ME client sends XML-RPC formatted request to the XML-RPC servlet for a specific method;
2. XML-RPC servlet parses the request, locates requested method within registered session beans and using provided parameters invokes it;

3. After executing the requested method, XML-RPC servlet sends an XML-RPC formatted response to the client with the result of the method execution;
4. J2ME client parses the response and displays results on the display.

Practical example of this collaboration would be our student service for signing up for exams (Fig. 3). At first, client retrieves available sessions for taking the exams. After selecting one, client sends this information to the server and retrieves exams available in the selected session. After exam selection, client retrieves the list of professors who will perform the testing of students.

This is the final step and after combining collected information in one unique structure it is being sent to the server and the process of exam sign up is finished.

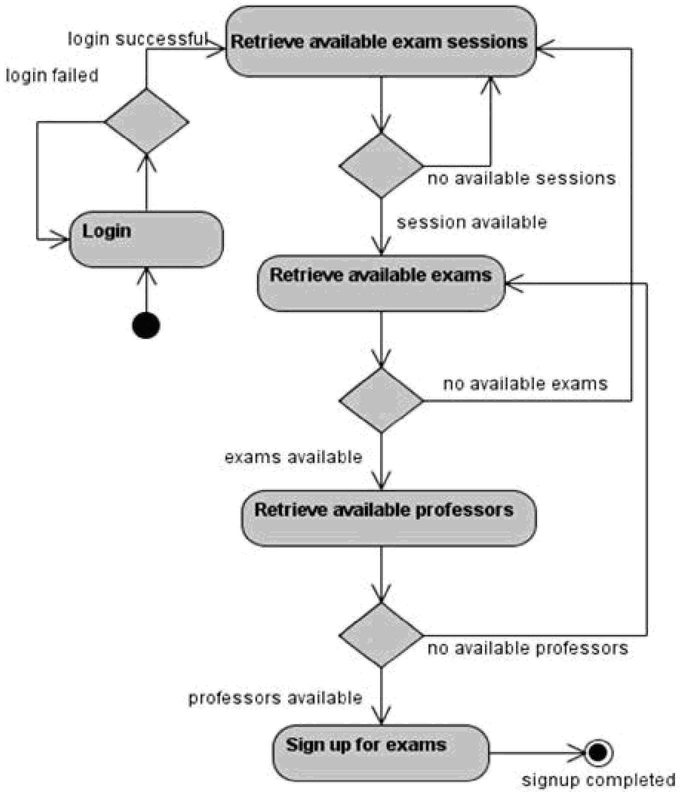

Fig. 3. Activity diagram.

4 Conclusion

With the proliferation of wireless broadband technologies and the development of advanced mobile devices, the software industry is well poised to leverage the advantages of connectivity to produce more reliable, efficient and responsive services. In this paper, we explored a mobile software architecture and implementation methods that allow wireless exchange of information with the existing information system. We identified key challenges in developing the mobile infrastructure and report on lessons learnt in implementing a wireless service system as part of the Faculty information system using J2ME platform. From testing results, we conclude that today software technologies fully cover the possibilities to develop advanced applications for mobile devices and connect them securely to existing enterprise information systems. Future work will include research and implementations for various mobile device platforms such as Android, Symbian and Windows Mobile.

Acknowledgements. Authors would like to acknowledge the support of the Serbian Ministry of Science, project “Abstract Methods and Applications in Computer Science” No. 144017A.

References

1. Jungum, N.V., Doomun, R.M., Ghurbhurrun, S.D.: Collaborative Driving Support System in Mobile Pervasive Environments. In: 4th International Conference on Wireless and Mobile Communications, pp. 358–363. Athens (2008)
2. HinzeA., Buchanan, G.: The Challenge of Creating Cooperating Mobile Services: experiences and lessons learned. In: 29th Australasian Computer Science Conference, pp.207–215. Hobart (2006)
3. Buchanan, G., Jones, M.: Search Interfaces for Handheld Web Browsers. In: Poster Proceedings of the 9th World Wide Web Conference, pp. 86–87. Amsterdam (2000)
4. Krogh, J.: J2ME: The complete reference. McGraw-Hill, Berkeley (2003)
5. Taylor, M.: Strategies for J2ME MIDP/J2EE Integration Over HTTP. Development Consulting Limited, Ripon (2002)
6. Yuan, M.J.: Enterprise J2ME: Developing Mobile Java Applications. Prentice Hall PTR, Upper Saddle River (2003)
7. Virkus, R.: Pro J2ME Polish. Apress, Berkeley (2005)
8. Škrbić, S., Bodroški, Ž., Pupovac, B., Racković, M.: Faculty Information System Based on Open Source Technologies. Novi Sad J. Math. 37(2), 181–192 (2007)
9. Rakić, M., Bodroški, Ž., Škrbić, S.: An Application of the GWT in Faculty Information System. In: 52nd Conference For Electronics, Telecommunications, Computers, Automatic Control And Nuclear Engineering, RT4.4-1-3. Palić (2008)
10. Panda, D., Rahman, R., Lane, D.: EJB 3 in Action. Manning, Greenwich (2007)