Computer Standards & Interfaces 44 (2016) 205–219

Contents lists available at ScienceDirect

Computer Standards & Interfaces

journal homepage: www.elsevier.com/locate/csi

Towards end-user development of REST client applications on smartphones

Gebremariam Mesfin [a], Tor-Morten Grønli [b], Dida Midekso [a], Gheorghita Ghinea [b,c,*]

[a] Addis Ababa University, Addis Ababa, Ethiopia
[b] Westerdals Oslo ACT, Faculty of Technology, Oslo, Norway
[c] Brunel University, London, UK

A R T I C L E I N F O

Article history:
Received 12 December 2014
Received in revised form 18 May 2015
Accepted 5 August 2015
Available online 4 September 2015

Keywords:
REST service
Usability
Service composition
Smartphone
Cross-platform

A B S T R A C T

HTML5 can be used to develop client applications by composing REST web services within the context of Web 2.0. However, the possibility of implementing cross-platform smartphone applications with REST services needs to be studied. Accordingly, we developed a REST-based cross-platform application with PhoneGap. The application was deployed on the Android, Windows Phone, and iOS platforms; subsequently we evaluated its usability. We observed that REST-based cross-platform smartphone applications can be implemented with HTML5 and PhoneGap, which can be scaled-up into a REST service composition tool. Moreover, the application's usability remains unaffected on the native platforms and adaptation required only minimal effort.

1. Introduction

Smartphones play a very important role in life, being used in education, healthcare, and business, to name but a few application areas. Smartphones may be described as mobile phones with increased capabilities, such as touch screen, intelligence and alertness. For the purpose of this study, we consider smartphones to boast features as described in [5]; hence we define smartphones as mobile phones that are capable of accessing the Internet and of running a variety of mobile operating systems such as Google's Android, Microsoft's Windows Phone, and Apple's iOS.

On top of the operating system, smartphones are equipped with software development kits (SDKs) that enhance the characteristics of smartphone application software and configurations such as reusability, and interoperability. Smartphone applications themselves can broadly be categorized into *native* and *cross-platform* based on the software development environments they are produced from. Native applications belong to one category of smartphone applications that are written and developed for a specific operating system. They have unhindered access to device hardware and support all user interface and interactions available in the respective mobile operating environment [29]. Cross-platform applications, on the other hand, can be dedicated mobile web applications, generic mobile web applications (also called mobile websites), and hybrid applications [29].

In this study, by cross-platform we understand those dedicated mobile web applications which are designed to mimic the native applications of the host operating system but actually execute on a web browser. Such applications are implemented based on a web browser, using fundamental web technologies — HTML5, JavaScript, and Cascading Style Sheets (CSS).

Cross-platform smartphone applications can also be REST-based applications implemented by composing (mashing-up) REST web services. The REST web service belongs to the family of service-oriented architecture (SOA) implementation technologies. It is a client–server architecture designed to make HTTP-based stateless communication between the components of a composite application using URIs [37]. In terms of productivity and time to market, cross-platform smartphone applications are preferred to native ones. However, cross-platform smartphone applications are challenged by limitations in the user experience when deployed on native platforms [29].

Usability is defined in [21] as the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use. Evaluating the usability of REST-based cross-platform smartphone applications is also of paramount importance. Accordingly, in this paper we evaluate the usability of REST-based cross-platform smartphone applications on their respective deployment operating platforms, which is an extension of our previous work [15].

* Corresponding author at: Dept. of Computer Science, Brunel University, Kingston Lane, UB8 3PH, UK.
E-mail addresses: mesfin.assres@gmail.com (G. Mesfin), tmg@westerdals.no (T.-M. Grønli), mideksod@yahoo.com (D. Midekso), george.ghinea@brunel.ac.uk (G. Ghinea).

http://dx.doi.org/10.1016/j.csi.2015.08.004
0920-5489/© 2015 Elsevier B.V. All rights reserved.

The paper is structured as follows. Cross-platform smartphone application development, REST-based application development, and issues related to usability oriented implementations with HTML5 are discussed in Sections 2, 3 and 4 respectively. A concrete comparison of usability of a REST-based cross-platform application on different platforms is presented in Section 5. In Section 6 we discuss our findings; finally, we draw our conclusions in Section 7.

2. Cross-platform smartphone applications

Smartphone operating systems are rich in libraries and built-in features. However, they still have to match and create a consistently high user experience, irrespective of the fact that their basic architecture and support of programming languages varies. Studies such as that of Henning et al. [18] make the point that the proliferation of a fragmented smartphone market with multiple operating platforms makes the development of native mobile applications a challenging and costly endeavor. To improve this, the literature and industry envision cross-platform development approaches.

The essence of cross-platform environments is a subset of the software development environments aiming at building platform independent applications. Cross-platform application development environments work based on the general principle of "*write once, and run everywhere*". In the smartphone application development, Dalmasso et al. [42] described the general architecture available to cross-platform mobile application development tools. However, as pointed out by Henning et al. [18], the diverse hardware and software platforms inevitably make portability a hassle for mobile application developers. Portability primarily depends on runtime support and the feasibility of achieving identical look-and-feel and functionality across platforms.

There are several attempts of implementations of cross-platform smartphone application development environments. For example, Java ME supports cross-platform development through configurations and profiles. Damianos and Economou [14] describe a configuration as the minimum Java VM features and library set for devices with similar processing and memory limitations, user interface requirements, and connection capabilities, while a profile comprises libraries specialized in the unique characteristics of a particular device class.

In related work, Grønli et al. [40] investigated the strengths and weaknesses of the mobile application development ecosystem and pointed out that developer support has improved the performance of developer tools which provide a higher level abstraction of performance-critical third party libraries. However, cross-platform development environments are being challenged by the different implementations, immature platform support, as well as by the variety of devices and browsers; in contrast, platform-specific ones like Windows Phone, iOS, and Android benefit from being tightly integrated with their respective operating system. Moreover, the work of Grønli et al. [40] showed that there is better integration between the development environment and deployment devices on the platform-specific ones than that of the cross-platform environment. This indicates that the cross-platform application development is in its early stages.

Studies [6,14,18] showed that cross-platform development tools are flourishing; they aim at addressing user experience, stability of framework, ease of updating, cost of development for multiple platforms, and the time to market of an application. When realized, the interests of many developers would be satisfied in terms of releasing applications for major mobile platforms and provide a consistent user experience across the platforms with minimal or no change to the original code. PhoneGap, Rhomobile, JQuery Mobile, and Xamarin are some of the cross-platform mobile application development tools available.

In the work described in this paper, we employ PhoneGap because of its popularity [6,35]. PhoneGap is an open source cross-platform smartphone application development tool developed by Adobe Systems Inc. under the Apache license. It provides a toolbox for building native mobile applications using only HTML5, JavaScript and CSS [6,35]. PhoneGap is quite popular among users mainly because of its flexibility, straightforward architecture and ease of use. Its architecture is mainly composed of the Web application, PhoneGap, and the operating system, along with native Application Programming Interfaces (APIs — Fig. 1).

PhoneGap is a "wrapper" that allows developers to enclose applications written in known programming languages into native applications [35]. That is, applications developed using PhoneGap are neither purely web-based nor purely native and thus some layout rendering is done via web-view instead of the native language of the operating system; consequently there is a lack of support of HTML in some functions. PhoneGap does not provide its own IDE (integrated development environment) to develop applications, but developers have to write the source code with an IDE and port their code into other IDEs such as the Eclipse for Android and Xcode for iOS. Thus far, PhoneGap permits the creation of applications for Windows Phone, Android, iOS, Bada, Symbian, and the WebOS operating systems.

In general, PhoneGap and other cross-platform development tools leverage device capabilities with the help of JavaScript APIs and generate the HTML code for presentation. However, the resulting code needs to be ported into specific operating systems like the Windows Phone, Android, and iOS so that they behave like native applications. In the following section, we provide an overview of these operating systems and their corresponding integrated development environments.

2.1. Windows Phone

Windows Phone is the smartphone operating systems advocated by Microsoft. In the latest versions of Windows Phone, smartphone applications are written in managed code by frameworks that support multiple languages such as C# from the Microsoft.NET environment. Windows Phone is primarily built with the Windows Phone SDK together with Silverlight and XNA add-ons on Visual Studio. XNA is employed for 2D and 3D games development while Silverlight is used to develop powerful and engaging interfaces. Programs created for Windows Phone are encapsulated into XAP files, which are packaged Silverlight applications [40].

2.2. Android

Android is based on the Linux kernel and developed as an open source system platform. In addition to the operating system, Android provides a development environment to write managed code with Google's Java libraries, and the Dalvik Virtual Machine for the smartphone applications to run on [9].

The development environment enables rich multimedia, the use of 2D and 3D graphic libraries, a customized SQL engine for persistent storage, and 3G, 4G and WLAN network capabilities [40]. Eclipse and IntelliJ IDEA are two main vendors of software development tools for Android.

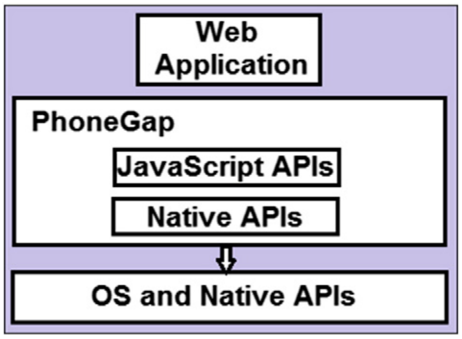

Fig. 1. Interfacing layers of the PhoneGap architecture.