Beyond Native Apps: Web Technologies to the Rescue! (Keynote)

Ivano Malavolta
Vrije Universiteit Amsterdam, The Netherlands
i.malavolta@vu.nl

Abstract

As of today, mobile software development teams can follow a number of different development and distribution strategies, ranging from native apps, to mobile web apps, hybrid apps, and the recently emerging progressive web apps.

This talk provides a state-of-the-art overview of the development strategies and technologies for developing mobile apps, each of them with its own advantages and drawbacks. In this context, the use of web technologies is discussed as a promising investment for moving forward one of the most intriguing challenges in the world of mobile apps: its fragmentation with respect to mobile platforms. A discussion of research challenges, and thus opportunities, closes the talk.

Categories and Subject Descriptors H.4.3 [*Information Systems Applications*]: Communications Applications

General Terms Languages, Management

Keywords Mobile applications, Mobile web, Hybrid development frameworks, Progressive web apps

1. Native Mobile Apps

Mobile apps consist of binary executable files that are downloaded directly to the user's device and stored locally [1]. Mobile apps are distributed via dedicated app stores, such as the Google Play Store for Android apps and the Apple app store for iOS apps. Native apps are developed directly atop the services provided by their underlying mobile platform. Those services are exposed via a dedicated Application Programming Interface (API) with methods related to communication and messaging, graphics, location, security, etc. [2]. Programming languages and tools for native mobile apps are platform-specific; for example, Android apps are created in Java via the Eclipse-based Android SDK, whereas Apple iOS apps are developed using either Objective-C or Swift via the XCode tool. Thanks to the platform-specific APIs and tools, developers can create native mobile apps with rich user experiences, heavy advanced graphics, and high performance. However, the use of platform-specific technologies leads to the well-known challenge of mobile platform **fragmentation**, as code written for one mobile platform (e.g., the Java code of an Android app) cannot be used on another (e.g., the Objective-C code of an Apple iOS app) [1]. Fragmentation makes the development and maintenance of native apps for multiple platforms one of the major technical challenges affecting the mobile development community [3]. This results in potentially high development time, high testing and maintenance costs, and low portability.

2. Web-Based Mobile Development Strategies

Standard web technologies like HTML5, CSS3, JavaScript can help in building mobile apps via a common aligned technological stack, thus mitigating the fragmentation problem. As of today, three are the main development strategies for developing apps via web technologies: mobile web apps, web-based hybrid mobile apps, and progressive web apps.

2.1 Mobile Web Apps

Mobile web apps are developed with web technologies, hosted on remote servers, served via standard protocols (e.g., HTTP), accessed via a unique URL. Basically, they are mobile-optimized websites accessed via the browser apps installed on users' mobile devices (e.g., Chrome, Firefox).

Since the code of mobile web apps conforms to standard languages, a single app delivers a uniform experience across platforms [1], offering fast development, simple maintenance, and full application portability. Those advantages are mainly facilitated by the widespread compatibility with the WebKit rendering engine, an open-source project led mainly by Google and Apple providing the most comprehensive HTML5 implementation available today [1]. Even if the browser is getting more and more a fully-fledged software platform (e.g., the HTML5 standard provides APIs for geolocation, accessing the camera, microphone, etc.), as of

today mobile web apps struggle in handling heavy graphics (e.g., in contrast to the ones provided by 3D Unity game engine) and still there no straightforward means for full access to low level features (e.g., background services management). Finally, since mobile web apps are hosted and served like usual websites, they cannot be distributed via app stores.

2.2 Web-Based Hybrid Mobile Apps

Web-based hybrid mobile apps take the best aspects from native and web mobile apps [4, 5]. They are developed via standard web technologies and they can be distributed for any supported mobile platform, like Android, iOS, or Windows Phone [1]. More specifically, a hybrid development framework (e.g., Apache Cordova) allows developers to create a cross-platform web-based mobile app by providing (i) a native wrapper for containing the web-based code, and (ii) a generic JavaScript API that bridges all the service requests from the web-based code to the corresponding platform API.

Thanks to the native wrapper, a hybrid mobile app can be packaged and distributed for any supported platform. Existing knowledge of web developers can be reused also for developing mobile apps, and the development process is simplified, mainly because of the need to maintain a single code base for all platforms. On the negative side, hybrid mobile apps can access the platform APIs only via the JavaScript bridge provided by the hybrid development framework, which considers only a subset of all the possible APIs provided by each platform; moreover, the existence of the JavaScript bridge imposes an additional performance overhead when accessing platform APIs. Finally, the user experience provided by a hybrid app is the same across multiple platforms; if on one side this may be a positive trait in terms of product identity, on the other side developers should take into special consideration how the app integrates with the platform it is running in (e.g., the app should manage the physical back button in Android devices, whereas in iOS devices this functionality is managed on screen).

2.3 Progressive Web Apps

Progressive web apps (PWAs) are special kinds of mobile web apps in which progressive enhancement, support for low or no network connectivity, background processing capabilities, push notifications support, and security are the first-class characteristics . Similarly to mobile web apps, a PWA is served from a remote server via HTTPS (so no update distribution delays like in native or hybrid apps) and can be initially accessed as a standard web app via a browser (i.e., no install is required before using the app). Then, the user can decide to install the PWA in the device, thus *promoting* it to a top-level mobile app, with full-screen support (no browser tabs needed), offline support via a dedicated management of caching, push notifications, etc. Three conditions must hold for considering a mobile web app as a PWA, namely: (i) it is served over HTTPS (this is a requirement for avoiding man-in-the-middle attacks), (ii) it comes with a web app manifest declaring app metadata like its name, icons, base URL, and (iii) it uses service workers , a set of APIs for allowing developers for programmatically caching and preloading assets and data, managing push notifications, etc.

As of today, the features provided by PWAs strongly depend on how each browser supports web standards. For example, PWAs can manage push notifications, access to camera, geolocation in Chrome, but currently they do not fully support access to contacts, calendar, system-level alarms, telephony data (messages, calls, or the user's phone number), access to low-level hardware features and sensors.

3. Research Challenges and Opportunities

Mobile apps development is an ever evolving and dynamic research playground, where many are the research challenges posed by mobile apps developed using web technologies. In the following we discuss some of the most intriguing and inspiring ones. As also highlighted in [6], the collection, measurement, and analysis of *quality* properties (e.g., performance, reliability, security) will be fundamental for advancing the state of the art and practice in web-based mobile app development. This aspect is also related to the need for designing and assessing a *testing* method that works across platforms in a seamless manner. A special focus can also be given to investigate on recurrent design and code anti-patterns impacting the *performance* of (hybrid or progressive) mobile apps, specially across different platforms. Finally, since PWAs have been advertised as performance boosters, network savers, providers of better user experience, etc., it will be interesting to investigate on the price that developers and users may have to pay for those features (e.g., higher *energy consumption*, higher *development and testing* effort, higher *complexity*?).

References

[1] Native, Web or Hybrid Mobile-app Development. *White paper, IBM Corporation*, April 2012. Document Number: WSW14182USEN.

[2] B. Fling. *Mobile design and development: Practical concepts and techniques for creating mobile sites and Web apps*. O'Reilly Media, Inc., 2009.

[3] M. E. Joorabchi, A. Mesbah, and P. Kruchten. Real Challenges in Mobile App Development. In *Empirical Software Engineering and Measurement, 2013*, pages 15–24, 2013.

[4] I. Malavolta, S. Ruberto, T. Soru, and V. Terragni. Hybrid mobile apps in the google play store: An exploratory investigation. In *Mobile Software Engineering and Systems (MOBILESoft), 2nd ACM Intern. Conference on*, pages 56–59. IEEE, 2015.

[5] I. Malavolta, S. Ruberto, T. Soru, and V. Terragni. End Users' Perception of Hybrid Mobile Apps in the Google Play Store. In *Mobile Services (MS), 2015 IEEE International Conference on*, pages 25–32. IEEE, 2015.

[6] A. I. Wasserman. Software Engineering Issues for Mobile Application Development. In *FSE/SDP Workshop on Future of Software Engineering Research*, pages 397–400, 2010.