Progressive Web Apps: the Definite Approach to Cross-Platform Development?

Tim A. Majchrzak
University of Agder,
Kristiansand, Norway
Email: timam@uia.no

Andreas Biørn-Hansen, Tor-Morten Grønli
Faculty of Technology, Westerdals Oslo ACT,
Oslo, Norway
Email: {bioand;tmg}@westerdals.no

Abstract

Although development practices for apps have matured, cross-platform development remains a prominent topic. Typically, apps should always support both Android and iOS devices. They ought to run smoothly on various hardware, and be compatible with a host of platform versions. Additionally, device categories beyond smartphone and tablets have emerged, which makes multi-platform support even trickier. Truly developing an app once and serving the multitude of possible targets remains an issue despite having cross-platform frameworks that are acknowledged by practice and research. The technology unifier remains to be found, but Progressive Web Apps (PWA) might be a step towards it. In this paper, we analyse the foundations of PWAs in cross-platform development and scrutinize the status quo of current possibilities. Based on our observations, we investigate unified development, and discuss open questions. We seek to stimulate interest and narrow the immense gap that has arisen since industry started to embrace PWAs.

1. Introduction

Over 10 years have passed since the introduction of the first iPhone [1]. In this time, the development methods have matured (cf. [2], [3], [4], [5], [6]). Parallel, the basic requirements of developing mobile apps can be said to have become *less complex* and *more complex* at the same time. On the one hand, the number of platforms with relevant market share has shrunken [7], powerful cross-platform development frameworks have emerged [8], and progress has been made in many other regards. On the other hand, device fragmentation remains a problem [8], [9], apps need to support novel device categories such as wearables [8], [10], and the technological progress is still rapid. The technology for unified development has still not been identified [11].

The advent of cross-platform app development frameworks have made it much easier to create apps for multiple platforms. Despite reduced learning effort, typically lower costs, and a quicker time-to-market [12], [13] cross-platform approaches do not prevail in all cases. While there are natural exceptions – such as graphic-intensive games, which ought to be programmed with the native SDKS [14], [15] –, the choice between native apps, cross-platform generated ones, and Webapps can remain tricky [16], [8]. Although many different attempts have been made with regard to *how* cross-platform development frameworks should work, no technology is considered absolutely superior. Unsurprisingly, still new frameworks arise [3] despite PhoneGap [17] (previously a.k.a. Apache Cordova [18]) is widespread in industry.

Progressive Web Apps (PWAs) are a novel way to develop, they promise to combine Web technologies' ease of development with the versatility of native apps [11]. This might be achieved without a (profound) performance penalty [19] but with a dramatic decrease in app size [20], [21]. Due to their novelty, scientific coverage is low [11]. Moreover, so far PWAs have been mainly evaluated in the realm of Android [21], [11]. With this research paper, we set out to narrow the gap between scientific assessment and industry adoption[1]. In particular, we seek to understand whether PWAs can become *that* unifying technology for cross-platform app development, or in fact even a multi-platform approach that overcomes the idiosyncrasies of cross-platform development.

This paper makes several contributions. First, we comprehensively assess competing concepts for developing multi-platform apps. Second, we scrutinize PWAs as a possible technology to overcome shortcomings of existing cross-platform development approaches. Third, we provide lessons learned and name open questions for a research agenda.

The remainder is structured as follows. In Section 2 we introduce the background of our work by shedding light on Webapps and native apps, cross-platform development and PWAs. We then discuss implication of unified development in Section 3. In Section 4 we discuss our findings. Section 5 gives a conclusion and names our next steps.

2. Background

In the following, we sketch the background of our work. First, we sketch native development and Webapps as base-

1. Please note that we have referenced several non-scientific works even including videos. This is mandated for staying up to date with the latest developments. We of course scrutinize such works with particular care. In fact, the literature situation is part of the reason for several of the open questions in Section 4; see specifically Section 4.2 on p. 7.

URI: http://hdl.handle.net/10125/50607
ISBN: 978-0-9981331-1-9

line approaches. We then comprehensively describe cross-platform app development and PWAs, for each investigating foundation, characteristics, and current literature.

2.1. Baseline: Native Development and Webapps

The *natural* choice for developing apps is the native *software development kit* (SDK). It usually is provided by the vendor of the mobile platform [22], [23, p. 225]. The SDK offers a development experience tailored to the platform [24]: There are normally few programming languages available. In the case of iOS, Objective-C and Swift can be leveraged [23]. On Android, the recent addition of the Java-superset language *Kotlin* was welcomed by the otherwise Java-heavy developer community [25]. If developing for the Windows platform, a so-called Universal Windows Platform app, multiple languages can be used, including C++, C#, Visual Basic and JavaScript [26]. Device features can be accessed through the platform's own application programming interfaces (APIs) [27].

The graphical user interface (GUI) is composed of native interface elements [15]. Native look & feel and short reaction times are direct consequences. Apps appear and can be interacted with in the same way as platform-specific apps respectively of the mobile operating system itself; the performance is high at least for carefully designed apps [28]. Obviously, native development was the starting point for app development when platforms were newly released.

Although features introduced to mobile platforms soon find their way into *all* contemporary platforms if they are considered beneficial, platforms remain incompatible [15], [29]. The incompatibility has deep roots. Not only does the GUI look different (when it follows the design guidelines [30], [31]) but the whole development experience is diverging. Typically, the APIs are not designed similarly, the programming languages impose a certain style, and the ecosystems of the platform including tools and paradigms differ [15]. As all of this makes it very expensive to develop the *same* software for native apps from several platforms [32], [33], cross-platform development frameworks have become popular [34], [24]. The same difficulties that humans face make also the creation of such a framework complex, as will be elaborated in the following section.

Webapp by itself is a rather unclear term. We are not aware of a consistent definition. In general, many applications that are provided via the Web and that follow a client-server model are considered Webapps; such applications existed before the advent of the smartphone. In the mobile computing literature Webapp typically denotes a *mobile Webapp*. In this paper we only consider Webapps in the sense of applications for mobile devices that have been built using Web technology such as HTML, CSS and JavaScript.

The first Webapps, merely made use of the platforms' built-in browsers. They lacked functionality such as being placed in an smartphones app list and to be distributed using the app stores [35]. Moreover, they had no or little access to device-specific functionality [36], [27] and relied on a generic Web appearance. While Webapps are still developed with the same basic Web technology, the actual programming is enriched by a huge variety of frameworks [37], [3]. Moreover, possibilities and performance have been improved. Due to their reliance on the browser, Webapps are compatible on all platforms that run a sufficiently capable browser [38]. However, as browsers differ in W3C HTML and JavaScript specification implementation and compliance, a Webapp does *not* inherently function as expected across all browsers [39]. Also, disabling JavaScript is a possibility in most modern browsers; Thus, effectively disabling the logic layer and often the user interface layer of modern Webapps results in apps rendering blank pages.

Before cross-platform app development became popular, developers were confronted with choosing between native apps and Webapps [15]. The decision advise has remained the same for "extreme" cases, but shifted for "average" apps (cf. e.g. [8]). Hardware-near, high performance apps require native development [40]. Rather simple, form-based apps are quickly developed as Webapps [15], [29]. With HTML5 and modern JavaScript, better performance has been achieved and a broader range of device features have become available [39]. Frameworks – in particular for GUI and for JavaScript – offer many additional features, e.g. Webapps that mimic a native look & feel [3]. Consequently, Webapps are popular [34], [29], [24].

Native development on the one side and Webapps on the other side remain the benchmark for any contestants (cf. [40]). This is true not only regarding cross-platform development frameworks but also for any other development technology, even if it is not specific to multi-platform development. Therefore, possibilities, ramification, and the quality of native apps and generic Webapps needs to be weighted in when discussing unified development.

2.2. Cross-Platform App Development

Cross-platform app development approaches are not a new emergence; however, technological possibilities have changed over time. We will first introduce the concept before describing the characteristics of developing in a cross-platform way. Eventually, we highlight related literature.

2.2.1. Introduction. The quick proliferation of smartphones (and, with some offset, tablets) is already a historic fact. It also meant, that particularly in the early years before 2010, many different platforms competed. Although now only two major platforms remain [41] , developing apps for multiple platforms at a time is an economic necessity. To reach as many users as possible, all platforms need to be supported that have a sufficiently high user base at the

moment of development (respectively the forecasted user base for the intended time of deployment). Developing native apps means that the effort for developing these apps scales almost linearly with the number of supported platforms [16]. Save for some organisational activities and the platform-independent part of requirements engineering, all other software development activities are required for each platform. Even worse, skilled developers must be hired for each target platform as quality problems are otherwise likely.

While HTML5 is not new anymore, Webapps in early smartphone times did not perform as well as they do today. In particular, a look & feel of a Web page instead of that of an app, and inferior performance were reasons not to build Webapps as an easy option for multi-platform support [42]. Therefore, possibilities for cross-platform app development were sought. The development of cross-platform development frameworks is active until today, although scientific research has already proposed convincing technological possibilities and the industry has widely adopted frameworks.

2.2.2. Characteristics. There is one basic characteristic that is shared by all cross-platform app development frameworks: code is developed *once* while apps can be provided to *several* platforms. Besides this, frameworks differ significantly in basic approach and paradigm, and in scope. The basic paradigmatic possibilities have been summarized in Figure 1. Although the origin of this depiction reach back to 2012 [43, p. 63], it still serves well for its purpose. In fact, only with PWAs an extension might soon be necessary; the reasons for this are discussed later in this paper.

Obviously, the initial choice is to use cross-platform development, or not. From a high level point of view, this leaves only native development when deciding against cross-platform app development. When taking a closer look, the options are a bit more blurred. PWAs might so far not have been considered a cross-platform development approach, although they by concept seek to support multiple platforms from one code base (inherently being Web-based).

Cross-platform development can be divided into two main paradigms: *runtime environments* and *generative* approaches. With the latter, the ultimate aim is to provide a native app for each supported platform. How this native app has been *generated* from a single code base can be further differentiated. *Model-driven approaches* (referred to as MDSD in the figure: model-driven software development [45]) rely on a platform-independent model from which apps are generated. Typically, they employ a – usually textual and, or possible alternatively, graphical – domain-specific language (DSL) to describe an app. *Transpilers* take (usually native) code written for one platform and transform it to native code for another platform.

Approaches that do not generate apps rely on a *runtime environment*. Instead of running code directly on a platform, which requires the code to be native to that platform, the runtime environment bridges between app and platform. There is a number of ways to realize this paradigm technologically. *Webapps*, which we already named a baseline approach, are simply run in the platform-provided Web browser. *Hybrid* apps typically rely on Web technology but provide a native packaging. Traditionally, this has been beneficial e.g. with regard to access to device-specific features but limited these apps to a look & feel that at most approximated native apps. Finally, apps based on a *self-contained runtime* are similar to hybrids yet use native GUI elements.

For examples of approaches and more details regarding the paradigms, please refer to [4], [44], [13].

2.2.3. Literature Coverage. The literature on cross-platform app development is broad. Leaving out papers in which cross-platform considerations are a side topic, there are several particularly notable categories of papers. First, papers present work on the creation or improvement of a specific framework (e.g. [2], [46], [13]). Typically, the frameworks represent one specific approach towards cross-platform development or seek to advance the technological possibilities into a particular direction. Second, papers address one specific framework, often in a case study-like fashion (e.g. [47], [48]). Such work typically seeks to explore certain aspects of a framework or to assess its feasibility in predefined contexts. Third, papers combine work with several frameworks, usually including a comparison (e.g. [3], [14], [49]), focusing on giving advice (e.g. [50]), or looking at particularities (e.g. [51]). These works can help with decision making and they may also contain such contributions to theory that enable an understanding of which approaches excel under which circumstances. Fourth, there are meta studies, which make use of the first two kinds of papers. Especially notable are studies that propose categorisation and comparison frameworks. As part of such, exemplary comparisons have been carried out. Therefore, the fourth kind of papers provides the gateway to most other relevant work.

A number of papers provide a comprehensive overview of cross-platform technology and how existing frameworks can be assessed. The most widely cited paper by Heitkötter et al. is already from 2013 [52]. The authors propose an evaluation framework. They have also exemplarily applied it to several development approaches, including Webapps and native apps as benchmarks. An extension of this original idea has been presented in 2016 [8]. This work not only provides a more extensive catalogue of evaluation criteria but also tries to include weighted evaluation that cater for novel device categories such as wearables. Several authors have conducted work that takes similar aims [53], [54], [55], [27], [56], [57]. All of these papers support a deepened understanding of cross-platform app development as a whole.

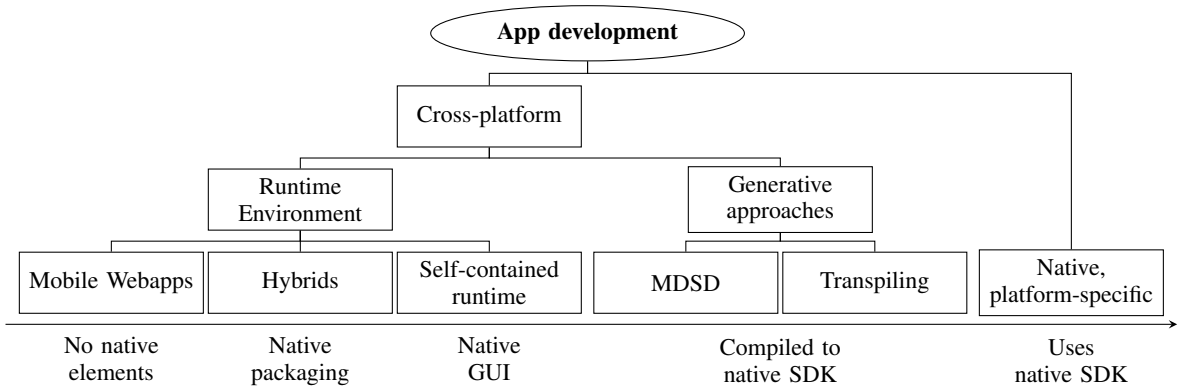

Figure 1. Categorization of Cross-Platform Approaches (adapted from [44], originally from [43, p. 63])

2.3. Progressive Web Apps

PWAs are new in concept, although they rely mostly on existing technological artifacts and concepts. As in the prior subsection, we introduce the idea, give details on characteristics, and summarize related literature.

2.3.1. Introduction. The mobile Web is becoming ever more capable of accessing and handling features previously only available in native and cross-platform apps. With the introduction of Progressive Web Apps (PWAs), regular Web sites can to a larger extent than before act, feel and look as any other installed app – so far particularly on an Android-based mobile device [58]. This is enabled through a set of new concepts and requirements, advocated by Google as well-worth implementation efforts [11]. In short, a PWA is any Web site implementing certain specific technical features, which this chapter will further investigate. Such a Web site can, in PWA-supported browsers, be added to the home-screen of a user's device and used offline. It looks like a regular app although being run inside a stripped-down Chrome browser, which hides all its interface artefacts.

2.3.2. Characteristics. There are certain characteristics that define PWAs, and that differentiate them from regular Web sites and native or cross-platform mobile apps. The main differentiator between a regular Web site and a PWA is the added functionality and User Experience (UX) the latter provides [58]. Where a regular Web site requires the user to open a browser, type in a URL and wait for all content to be downloaded on every visit, effectively preventing an offline experience, a PWA only requires these steps for the first visit. After a home screen installation, all necessary static files, including HTML, CSS, JavaScript, images and fonts for the Web site, are now stored on the user's phone, ready to be used offline [59]. All dynamic data can be cached for offline (or low-connectivity) use, and re-fetched when needed, e.g. when new data is available *and* the phone is on a decent network connection [59].

Where a regular Web site would be wrapped in a browser (e.g. Chrome Android) with visible browser artefacts (such as address bar and menus), a PWA will similarly run in a browser instance, but without those artefacts [58]. Thus, a PWA will look similar to a regular app. If a PWA is styled correctly, following the design guidelines of each mobile platform, telling apart a regular native or cross-platform app and a PWA from the appearance would be challenging.

When comparing a PWA to a regular native or cross-platform mobile app, one of the main characteristics and benefits of the PWA is the minimal footprint it leaves on a user's phone [20], [21], [11]. Examples of this are provided in Section 3.4 using the *Twitter Lite* and Indian ride-sharing app *OLA* as cases. In either case, their PWA would be two order of magnitude smaller than the comparable native apps – without compromising functionality [20].

From a technical perspective, a Progressive Web App is simply put a regular Web site with a JSON-based meta data document (*manifest*) [60], an *Application Shell* and a background script (*Service Worker*) written in JavaScript [11]. The Application Shell is the minimal static GUI and logic needed to render and display the application without connection-dependant dynamic content, i.e. in an offline setting [61]. This GUI and logic may include pieces of the app such as routing and navigation logic and UI elements [62]. Due to the Application Shell being available offline and fetched from the app cache, it renders immediately (measured as *time-to-first-paint*), making the application accessible quicker than a non-PWA online-only regular Web site [61]. Next, the Service Worker script handles and proxies all network connectivity, caching logic and background tasks, thus enabling the offline experience a PWA can provide [63]. The Service Worker can decide if new content should be fetched from a Web service, or if any cached content is still relevant to the user, thus avoid unnecessary network calls, or even to display cached content in an offline setting. Additionally, the Web site must be served over HTTPS [63] and implement responsive design [64], i.e. work satisfactory across devices of different screen sizes and resolutions [59].

From a developer perspective, multiple considerations should be taken into account when developing a Progressive Web App. As an example, Google advocates use of the (experimental) PRPL pattern (abbreviation for Push – Render – Pre-cache – Lazy-load, pronounced *Purple*). The

pattern builds on the idea that the Web is slow on mobile devices, and that the time it takes for the application to become interactive (measured as *time-to-interactive*) is of utmost importance to optimize for. This is done through techniques such as heavy caching of dynamic content, the new HTTP/2 standard to smarter request and receive static files from the Web server, the Application Shell, and smart loading of content [62]. Moreover, Google has released a developer-oriented PWA testing tool named Lighthouse [65]. This tool helps in testing PWA compliance of Web sites, thus easing the developer experience of PWA development.

2.3.3. Literature Coverage. A lack of literature covering Progressive Web Apps has already been identified by a recent position paper [11], which brought the concept of Progressive Web Apps to the academic communities. The paper gives a holistic introduction to PWAs, as well as some initial research findings and thoughts. Research was based on three apps: one hybrid app, an interpreted app, and a PWA. Some initial performance measurements were conducted and the generated app sizes were compared. This paper additionally presented a list of possible further research areas and topics to be explored. While this paper can be seen as a kind of shorter predecessor to our article, it must be ascertained that to a large extent the current body of knowledge is made up of literature published by developers, practitioners, and the industry in general.

Practitioners and the industry continues to put efforts in implementing PWA characteristics into their Web sites, as discussed in detail in Section 3.4. Little progress within academia can so far be recorded. The academic contributions identified are few in number. Malavolta et al. [21] makes an interesting contribution discussing energy efficiency of Progressive Web Apps, and the energy impact of Service Workers. Their research includes measuring energy consumption using different devices and scenarios. For further research, we propose drawing from the results of both [21] and Ciman and Gaggi' research [40] on energy consumption of cross-platform app development frameworks and approaches.

Except from the previously discussed literature, little else has been identified directly on the topic of PWAs within academia. Outside of academia, in the fast-paced world of JavaScript and the (mobile) Web, articles and discussions on PWA proliferate. As discussed in Section 3.4, the Google I/O 2017 conference featured seven PWA-related talks, some highly technical, some rather business-related. In 2016, Google also hosted the first *Progressive Web App Dev Summit* [66], as an effort to further advocate the concept to developers and the industry. While practitioners' enthusiasm must of cause be weighted carefully as Google seems to rally for PWAs, the spread of interest is nonetheless remarkable.

Multiple books on PWAs either have been published or are in the process of being written, e.g. Hume's early-access book [67], a technical step-by-step textbook for building PWAs. Published on O'Reilly also as an early-access book, Ater [68] aims to teach various technical aspects of PWAs from a mobile *native* point of view, suggesting he is *Bringing the Power of Native to the Browser*.

The literature situation suggests numerous research possibilities, as discussed in Section 4.2. Quite notable is the lack of discussion of PWA development in the realm of iOS, owed to the current lack of support. While missing support by Apple might be a major hindrance for the spread of PWAs, practitioners began to question whether Apple can retain its position [69].

2.4. Further Approaches

The boundaries between cross-platform apps, PWAs, and even Webapps are blurry. This can be explained with the lack of definitions and with a disjunctive distribution of *some* technological aspects. Additionally, some approaches do not serve multi-platform development as the primary goal but still contribute to it. An extensive study of this phenomenon is out of scope. Two approaches are noteworthy, though.

The concept of offline-enabling Web sites on mobile phones is not new. For some time, the Application Cache (*AppCache*) was the solution for achieving exactly that [70], [71]. However, with the introduction of Service Workers, the AppCache is in the process of being discontinued [72] in Service Workers' and thus PWA's favour.

Another alternative approach is advocated by Microsoft [73], named *Hosted Web Apps* (HWA). The concept fits in-between web apps and native apps, similar to Progressive Web Apps. The main benefit of using HWA over PWA is its tight integration with the Windows platform. This enables features such as invoking native Windows platform APIs from the HWA JavaScript codebase [73]. An HWA will also run cross-Windows, including devices such as PCs, Windows Phone devices and Xbox. [73] This is indeed an interesting type of *hybrid* solution, as it blends platform API calls into a web setting. This approach can seemingly be compared to Cordova for hybrid mobile app development, although limited to the Windows platform.

3. Unified Development

Based on an understanding of the background, we draw the vision of unified mobile development. Therefore, we first propose requirements for a unified approach before looking how the transition from cross-platform frameworks to PWA follow these. We then scrutinize the possibilities and limitations of the available technology. Eventually, we investigate into the status quo of PWA adoption.

3.1. Requirements

Requirements for unified development can be discussed from two perspectives. First, requirements to design apps

in a multi-platform manner can be scrutinized, e.g. regarding GUI and performance—the conceptual view. Second, technical requirements can be addressed, e.g. regarding preconditions and bridging elements—the technological view.

We deem the following requirements essential when seamlessly building apps for several platforms at a time:

- A user interface (UI) that can be built from the typical elements, and that provides different layouts,
- the possibility to provide a control flow and various forms of interaction in an app,
- a look that aligns with design and usability guidelines for the respective platforms,
- the possibility to define data types and provide basic create, retrieve, update, and delete, (CRUD) operations, both on the device and in a client-server model,
- mapping of form fields to a data model, including the validation of input and the possibility to persist data,
- means to react to input, events and state changes, and
- the possibility to access at least the most common hardware features.

Moreover, a framework should have a steep learning curve, support major platforms, and provide timely results. As should be apparent from the discussion of PWAs, a sufficient long-term feasibility is a strict requirement [8].

This assessment roughly follows the prerequisites for business apps, as proposed by [13], [74]. It has been amended with insights from the earlier discussed related work as well as from our considerations presented in this article so far.

Several technological prerequisites can be identified for the creation of apps that span several platforms. What we propose again follows general suggestions from the specific literature (particularly [8]) in conjunction with the thoughts presented in this article:

- Adequate development support by a development environment and tools, as well as good testability of apps,
- a sufficient degree of scalability,
- a near-native performance,
- a good level of maintainability and extensibility of apps,
- interfaces for business functionality, such as support of backend systems, and
- at least a basic level of framework-integrated security.

3.2. From Cross-Platform Technology to PWAs

At the time of writing, certain technologies required for Progressive Web Apps to work are still not completely (if at all) implemented in some major browsers. According to *Is ServiceWorker Ready* [75] per mid-2017, Chrome, Firefox and Samsung Internet do all fully support the Service Worker specification. Microsoft's Edge, the successor of Internet Explorer, states that the Service Worker specification is currently being implemented. Regrettably, Apple's Safari browser does not implement the specification, but it is listed as a feature under consideration [76]. However, representatives from Apple are present at face-to-face W3C specification meetings regarding Service Workers [77]. Thus, hopefully Safari will at some point in the future implement the specification, which would allow PWAs to function correctly on iOS devices. Due to the current lack of Service Worker support in Safari, no iOS devices are able to take full advantage of PWAs, thus limiting them from becoming a true cross-platform development approach or alternative.

It was previously speculated that Safari's lack of Service Worker support might be due to the impact PWAs can have on the Apple App Store and the income it generates [11] (cf. also [69]). If users can download and install apps through their browsers without going through a general app store, Apple's app ecosystem may suffer.

PWAs enable cross-platform development outside of just the mobile sphere. Google Chrome OS users will be able to install PWAs on their machines, also greatly expanding the ecosystem of the operating system [78]. The Windows 10 Store will crawl the Web for PWAs and list them in their store, making PWA a *first-class citizen* of their app ecosystem [78], [79]. As a side note, it will be interesting to see whether or not Microsoft has a business plan for revenue generation from PWAs marketed through their store.

3.3. Technological Prospects and Limitations

A substantial developer-oriented effort was presented during Google I/O 2017 [80], where Osmani showcased a technical baseline for testing JavaScript frameworks against PWA criteria. The baseline, named HNPWA [81], aims at helping developers choose a JavaScript front-end framework for building PWAs. Numerous frameworks are included in the test, such as React, Preact, Svelte, Vue.js and Angular. All the tests' code-bases are open-source, and tests using new frameworks and approaches can be added to the HNPWA Web site for others to learn from.

For PWAs, access to device and platform APIs is still constrained to those APIs supported by the users' browsers. This is one of the major limitations of the PWA approach when compared to native or cross-platform development where all, or most, open device APIs are exposed to the developer through the native SDKs or similar abstractions. Somewhat reassuring, the Google Chrome team added 215 new APIs in the last year alone [20], so progress to bridge that gap is undoubtedly improving. Nevertheless, with the inherent future introduction of new mobile platform features, PWAs' access to them is still limited until an HTML/JavaScript specification has been formalized by W3C and (or at least) is implemented by browser vendors. Fragmentation of platforms, operating systems versions and browser- and device capabilities do not help the Web move towards unification.

Interestingly, even concerning programming languages the Web might see further fragmentation. An notable techno-

logy, slowly gaining traction in the industry, is WebAssembly (*wasm*), a format and enabler for writing Web applications using other languages than JavaScript. This may open the porting of software, specifically of games, which traditionally only ran as native desktop applications [82]. For Webapps and PWAs, support for multiple languages might introduce new developers to the platform, which could lead to further adoption and proliferation of such apps.

3.4. Status Quo of PWA adoption

During Google's developer conference, Google I/O 2017, a number of Progressive Web Apps initiatives were presented both on-stage and as mentions. Large companies and key players in the mobile-Web space have already started converting their existing Web apps to PWAs with great success. This includes the before mentioned Twitter and OLA, both leading companies within their fields. The companies also revealed statistics about app sizes and reported increased usage after PWA adoption [20].

The native Twitter apps for Android and iOS are respectively $>$ 23MiB and $>$ 100MiB, while their PWA weighs in at 0.6MiB (600KiBs) still providing most of the expected features. They also see more than a million home-screen launches daily, and found their PWA, *Twitter Lite*, to be of high importance for emergent markets.[20]. Similarly, OLA, India's largest ride-hailing app, found their PWA to weigh 0.20MiB (200 KiBs), considerably smaller than their native Android and iOS apps, respectively at 60MiB and 100MiB. OLA categorizes their customers into *tiers* based on their location, where tier 3 represents very low-connectivity areas in India, and tier 1 represents areas with good or decent connectivity. What OLA experienced was that in tier 3 areas, a 68% increase of mobile traffic had been observed since the launch of their PWA, and had a 30% higher conversion rate than their native app. In tier 2 areas, the conversion rate from their PWA was at the same rate as their native app. This illustrates the business potential and importance of the PWA for a company such as OLA that operates in different kinds of areas, dealing with low-connectivity situations [20].

Other notable companies working on PWAs include Forbes, Financial Times, Lyft, Expedia, AliExpress, Tinder, Flipkart, and Housing.com. These companies and their products represents different niches and markets, illustrating how PWAs can be a fit anywhere. Again, when mentioning these names a component of advertisement must be kept in mind; they notwithstanding illustrate notable spread.

Also during the conference, seven talks were directed towards the development of- and enthusiasm for this possible next generation of the mobile Web. PWAs were discussed in the context of mobile User Experience (UX) [83], support by technical frameworks [84] [80], performance testing [85] and migration [86]. Google is obviously pushing PWAs as an effort to improve the user experience of the mobile Web.

4. Discussion

In the following section, we sketch a vision of emergent developments. We then elaborate open research questions, aiming at a better understanding of which of them might be answered soon and which might remain temporarily open or even unresolved. Finally, we take steps towards a research agenda for the unification of mobile app development.

4.1. Emergent Development

Mobile development, whether being native, cross-platform or Web-based, moves fast with steady streams of new technical frameworks, approaches and techniques. The result is a fragmentation of developer communities, development approaches, and thoughts on how apps *should* be developed. In this section, we outline our balanced considerations on where this type of development is going, focusing on two main technologies and concepts.

First, with the introduction of Kotlin as a first-class programming language on the Android platform, more developers can arguably get into Android development. However, additionally interesting is a new effort by Jetbrains, the creators of Kotlin, named *Kotlin/Native* [87]. This LLVM-based backend will aid developers in running Kotlin on non-(J)VM platforms such as iOS. Thus, the Kotlin language might be suitable for cross-platform development in the near future, with default support on Android. The cross-platform support will not be limited to Android and iOS; platforms such as MacOS, Ubuntu and Raspberry Pi are also said to be supported [88]. As Kotlin can also transpile to JavaScript, similar to other solutions such as CoffeeScript and TypeScript, developers can now use Kotlin to develop software across multiple platforms.

Second, we are witnessing proliferation of *mobile-first* technical frameworks and development approaches. Where previous research often cite cross-platform app frameworks such as PhoneGap, Titanium Appcelerator, DragonRad and Rhodes [89], [90], [23], [12], a plethora of newer and more technically updated frameworks have been released lately. Although very recent papers cover frameworks such as React Native, Ionic and Fuse [3], but other solutions including NativeScript [91], Quasar [92] and Apache Weex [93] are yet to be scrutinized at the same level as e.g. PhoneGap in an academic context. As PWAs are increasing in popularity, perhaps we will see another direction of such frameworks, from development of native(-like) apps rather to target Web without any native abstractions such as Cordova.

4.2. Open Questions: Towards a Research Agenda

We have taken an approach in this paper that combines several methodological aspects: it joins a literature study, a theory-contribution, and a position paper. While this

synthesis allowed us to answer many questions, answering the underlying *research* question was only possible to some degree. Consequently, open questions have risen.

PWA adoption might fall with a lack of iOS support. Although there might be technological pressure on Apple, it is very unlikely that PWAs will be a unifier if iOS-based devices are not supported. Apple has shown in the past that they can take very consequent positions if they are not convinced of a technology, as demonstrated with Adobe Flash [94].

Connected to this unclarity is the problem of predicting industry adoption. The technological progress is still so rapid that even educated estimates can easily turn wrong. Indicators such as technological superiority or current market power are not reliable, as has been demonstrated by the history of cross-platform app development thus far. It also remains to be seen if disruptive technological progress can be made in the area of smartphones and tablets. A stronger focus on voice control, virtual reality, augumented reality, and even technology that still has a science fiction appeal (such as neural interfaces) may be game changers – *or not*.

Wearables and other novel mobile devices likely will gain even more momentum in their development – and market presence. It is currently unclear how well PWAs are suited for wearables. Even more questionable is whether they make sense for Internet-of-Things (IoT) devices, which might come without a graphical user interface. Future research needs to explore this direction. In particular, it would be important to learn about gaps that PWAs (and other cross-platform approaches) pose regarding IoT.

Considering all the technological aspects, the profound question is how well PWAs will do when the next technological steps come. The same question of course need to be asked for any possibly unifying technology. For sketching a research agenda, we believe that scientific research must seek to keep pace with the developments. We must not sacrifice rigour for this, and especially we must not engage in speculation as would easily be possible as underlined with the open questions we named. However, we deem work on the cutting edge vital, if this work manages to build on the increasingly solid base we have in the understanding of the modern notion of mobile computing.

Therefore, we propose the following steps:

1) Extend the base of experimental work with PWAs, and with other novel cross-platform frameworks.
2) Seek case-study-based work with a practical flavour.
3) If technological underpinnings have been fully understood, conduct a large-scale quantitative study with practitioners.
4) In the meantime, scrutinize the latest developments.

Moreover, we propose joining the activities from research on mobile apps and that of IoT at yet-to-be-defined interfaces. Both streams of research have much in common, and much to contribute to each other.

5. Conclusion and Future Work

In this paper, we have presented work on several topics from modern mobile computing. We have proposed a research question: Can Progressive Web Apps (PWAs) be the definite approach to cross-platform development? To answer the question, we have introduced several technologies and their underpinnings. Based on the background, we have scrutinized PWAs regarding their possibilities in being an unifying technology for mobile app development. Whether we can assert that PWA fulfil many requirements for unified multi-platform development already, is too early to say and whether they will be able to replace existing cross-platform development approaches. Moreover, other noteworthy technologies are likely to be seen soon.

However, the massive interest by practitioners mandates further research. At least, PWA can contribute to a richer development experience, and – eventually – *better* apps. We have therefore sketched future developments and suggested a research agenda. We suggested a balanced approach of experimental and qualitative work.

While we hope to encourage other researchers as well, our interest in the topic undoubtedly has been sparked. We will continue investigating, aiming at closing some of the tasks we have set out in this paper. Our own work will not only continue technologically. If the understanding has matured further, we hope to be able to set a research agenda, based on the milestones to be achieved until the definite approach for cross-platform development has been found.

References

[1] M. Macedonia, "iPhones Target the Tech Elite," *Computer*, vol. 40, pp. 94–95, 2007.

[2] M. Usman, M. Z. Iqbal, and M. U. Khan, "A product-line model-driven engineering approach for generating feature-based mobile applications," *JSS*, vol. 123, pp. 1 – 32, 2017.

[3] T. A. Majchrzak, A. Biørn-Hansen, and T.-M. Grønli, "Comprehensive analysis of innovative cross-platform app development frameworks," in *Proc. 49th HICSS*. IEEE CS, 2017.

[4] T. A. Majchrzak, J. C. Dageförde, J. Ernsting, C. Rieger, and T. Reischmann, "How Cross-Platform Technology Can Facilitate Easier Creation of Business Apps," in *Apps Management and E-Commerce Transactions in Real-Time*, S. Rezaei, Ed. IGI Global, 2016.

[5] J. C. Dageförde, T. Reischmann, T. A. Majchrzak, and J. Ernsting, "Generating app product lines in a model-driven cross-platform development approach," in *Proc. 49th HICSS*. IEEE CS, 2016, pp. 5803–5812.

[6] M. Ciman, O. Gaggi, and N. Gonzo, "Cross-platform mobile development: A study on apps with animations," in *Proc. 29th Annual ACM SAC*. ACM, 2014, pp. 757–759.

[7] L. Goasduff and A. A. Forni, "Gartner says worldwide sales of smartphones grew 7 percent in the fourth quarter of 2016," 2017, http://www.gartner.com/newsroom/id/3609817.

[8] C. Rieger and T. A. Majchrzak, "Weighted evaluation framework for cross-platform app development approaches," in *Proc. 9th EuroSymposium*, ser. LNBIP, vol. 232. Springer, 2016, pp. 18–39.

[9] S. Evers, J. Ernsting, and T. A. Majchrzak, "Towards a reference architecture for model-driven business apps," in *Proc. 49th HICSS*. IEEE CS, 2016, pp. 5731–5740.

[10] C. Rieger and T. A. Majchrzak, "Conquering the Mobile Device Jungle: Towards a Taxonomy for App-Enabled Devices," in *Proc. 13th WEBIST*. SciTePress, 2017, pp. 332–339.

[11] A. Biørn-Hansen, T. A. Majchrzak, and T.-M. Grønli, "Progressive web apps: The possible web-native unifier for smobile development," in *Proc. 13thWEBIST*. SciTePress, 2017, pp. 344–351.

[12] A. Ribeiro and A. R. da Silva, "Survey on cross-platforms and languages for mobile apps," in *Proc. 8th QUATIC*. IEEE Computer Society, 2012, pp. 255–260.

[13] H. Heitkötter, T. A. Majchrzak, and H. Kuchen, "Cross-platform model-driven development of mobile applications with MD^2," in *Proc. SAC '13*. ACM, 2013, pp. 526–533.

[14] M. Palmieri, I. Singh, and A. Cicchetti, "Comparison of cross-platform mobile development tools," in *16th ICIN*, 2012, pp. 179–186.

[15] A. Charland and B. LeRoux, "Mobile application development: Web vs. native," *Queueing Syst.*, vol. 9, no. 4, 2011.

[16] H. Heitkötter, S. Hanschke, and T. A. Majchrzak, "Evaluating cross-platform development approaches for mobile applications," in *LNBIP*. Springer, 2013, vol. 140, pp. 120–138.

[17] "PhoneGap," 2017, http://phonegap.com/.

[18] "Apache Cordova," 2017, http://cordova.apache.org/.

[19] J. Archibald, "Instant loading: Building offline-first progressive web apps," 2016, https://www.youtube.com/watch?v=cmGr0RszHc8.

[20] R. Roy-Chowdhury, "The mobile web: State of the union," 2017, https://www.youtube.com/watch?v=_ssDaecATCM.

[21] I. Malavolta, G. Procaccianti, P. Noorland, and P. Vukmirović, "Assessing the impact of service workers on the energy efficiency of progressive web apps," in *Proc. 4th MOBILESoft*. IEEE Press, 2017, pp. 35–45.

[22] N. Bermingham and M. Prendergast, "Bespoke mobile application development," in *Handbook of Research on Mobile Devices and Applications in Higher Education Settings*, L. Briz-Ponce, J. A. Juanes-Méndez, and F. J. García-Peñalvo, Eds. IGI Global, 2016, ch. 10.

[23] M. Latif, Y. Lakhrissi, E. H. Nfaoui, and N. Es-Sbai, "Cross platform approach for mobile application development: A survey," in *Proc. IT4OD*. IEEE, 2016, pp. 1–5.

[24] M. Ali and A. Mesbah, "Mining and characterizing hybrid apps," in *Proc. Int. Workshop on App Market Analytics*, ser. WAMA 2016. ACM, 2016, pp. 50–56.

[25] M. Shafirov, "Kotlin on android. now official," 2017, https://blog.jetbrains.com/kotlin/2017/05/kotlin-on-android-now-official/.

[26] M. LeBlanc, A. Koren, and M. Satran, "Choosing a programming language," 2017, https://docs.microsoft.com/en-us/windows/uwp/porting/getting-started-choosing-a-programming-language.

[27] S. Xanthopoulos and S. Xinogalos, "A comparative analysis of cross-platform development approaches for mobile applications," in *Proc. 6th BCI*. ACM, 2013, pp. 213–220.

[28] I. T. Mercado, N. Munaiah, and A. Meneely, "The impact of cross-platform development approaches for mobile applications from the user's perspective," in *Proc. WAMA*. ACM, 2016, pp. 43–49.

[29] I. Malavolta, S. Ruberto, T. Soru, and V. Terragni, "End users' perception of hybrid mobile apps in the google play store," in *Proc. IEEE MS 2016*. IEEE, 2015, pp. 25–32.

[30] N. Mitrovic, C. Bobedy, and E. Menay, "A review of user interface description languages for mobile applications," in *Proc. 10th UBICOMM*. IARIA, 2016, pp. 96–101.

[31] A. I. Wasserman, "Software engineering issues for mobile application development," in *Proc. FSE/SDP*, ser. FoSER '10. New York, NY, USA: ACM, 2010, pp. 397–400.

[32] L. Delia, N. Galdamez, P. Thomas, L. Corbalan, and P. Pesado, "Multi-platform mobile application development analysis," in *Proc. IEEE 9th RCIS*, 2015, pp. 181–186.

[33] M. Willocx, J. Vossaert, and V. Naessens, "A quantitative assessment of performance in mobile app development tools," in *Proc. IEEE MS*. IEEE, 2015, pp. 454–461.

[34] N. Viennot, E. Garcia, and J. Nieh, "A measurement study of google play," in *Proc. ACM SIGMETRICS*, vol. 42. ACM, 2014, pp. 221–233.

[35] R. Mullins, "Apple's iphone open to software developers," 2007, http://www.infoworld.com/article/2662919/.

[36] A. Holzinger, P. Treitler, and W. Slany, "Making apps useable on multiple different mobile platforms: On interoperability for business application development on smartphones," in *Proc. Int. Conf. on ARES*. Springer, 2012, pp. 176–189.

[37] H. Heitkötter, T. A. Majchrzak, B. Ruland, and T. Weber, "Comparison of Mobile Web Frameworks," in *Revised Selected Papers WEBIST 2013*, ser. LNBIP, K. Krempels and A. Stocker, Eds., vol. 189. Springer, 2014, pp. 119–137.

[38] A. Juntunen, E. Jalonen, and S. Luukkainen, "Html 5 in mobile devices–drivers and restraints," in *Proc. 46th HICSS*. IEEE, 2013, pp. 1053–1062.

[39] M. Firtman, "Mobile HTML5 compatibility," https://mobilehtml5.org/.

[40] M. Ciman and O. Gaggi, "An empirical analysis of energy consumption of cross-platform frameworks for mobile development," *Pervasive and Mobile Computing*, 2016.

[41] Gartner, "Gartner – worldwide sales of smartphones," 2017, http://www.gartner.com/newsroom/id/3609817.

[42] T. A. Majchrzak and H. Heitkötter, "Status Quo and Best Practices of App Development in Regional Companies," in *Revised Selected Papers WEBIST 2013*, ser. LNBIP, K. Krempels and A. Stocker, Eds., vol. 189. Springer, 2014, pp. 189–206.

[43] H. Heitkötter, T. A. Majchrzak, U. Wolffgang, and H. Kuchen, *Business Apps: Grundlagen und Status quo*, ser. Working Papers. Förderkreis der Angewandten Informatik an der WWU Münster e.V., 2012, no. 4.

[44] T. A. Majchrzak, J. Ernsting, and H. Kuchen, "Achieving business practicability of model-driven cross-platform apps," *OJIS*, vol. 2, no. 2, pp. 3–14, 2015.

[45] T. Stahl, M. Voelter, and K. Czarnecki, *Model-Driven Software Development*. Wiley, 2006.

[46] V. M. Ionescu, "Using cross platform development libraries. telerik mobile," in *2016 15th RoEduNet Conference: Networking in Education and Research*, 2016, pp. 1–6.

[47] LiTian, HuaichangDu, LongTang, and YeXu, "The discussion of cross-platform mobile application based on phonegap," in *4th ICSESS*, 2013, pp. 652–655.

[48] B. R. Mahesh, M. B. Kumar, R. Manoharan, M. Somasundaram, and S. P. Karthikeyan, "Portability of mobile applications using phonegap: A case study," in *Proc. ICSEMA*, 2012, pp. 1–6.

[49] J. Ohrt and V. Turau, "Cross-platform development tools for smartphone applications," *Computer*, vol. 45, no. 9, pp. 72–79, 2012.

[50] S. Charkaoui, Z. Adraoui, and E. H. Benlahmar, "Cross-

platform mobile development approaches," in *Third CIST*, 2014, pp. 188–191.
[51] M. Martinez and S. Lecomte, "Towards the quality improvement of cross-platform mobile applications," in *Proc. 4th MOBILESoft*. IEEE Press, 2017, pp. 184–188.
[52] H. Heitkötter, S. Hanschke, and T. A. Majchrzak, "Evaluating Cross-Platform Development Approaches for Mobile Applications," in *Revised Selected Papers WEBIST 2012*, ser. LNBIP, vol. 140. Springer, 2013, pp. 120–138.
[53] T. F. Bernardes and M. Y. Miyake, "Cross-platform mobile development approaches: A systematic review," *IEEE Latin America Trans.*, vol. 14, no. 4, pp. 1892–1898, 2016.
[54] A. Hudli, S. Hudli, and R. Hudli, "An evaluation framework for selection of mobile app development platform," in *Proc. 3rd MobileDeLi*, 2015.
[55] S. Dhillon and Q. H. Mahmoud, "An evaluation framework for cross-platform mobile application development tools," *Software – Prac. and Exp.*, vol. 45, no. 10, pp. 1331–1357, 2015.
[56] A. Sommer and S. Krusche, "Evaluation of cross-platform frameworks for mobile applications," *LNI*, vol. P-215, 2013.
[57] I. Dalmasso, S. K. Datta, C. Bonnet, and N. Nikaein, "Survey, comparison and evaluation of cross platform mobile application development tools," in *Proc. 9th IWCMC*, 2013.
[58] P. LePage, "Your first progressive web app," 2017, https://developers.google.com/web/fundamentals/getting-started/codelabs/your-first-pwapp/.
[59] Google Developers, "Progressive web app checklist," 2017, https://developers.google.com/web/progressive-web-apps/checklist.
[60] M. Guant and P. Kinlan, "The web app manifest," 2017, https://developers.google.com/web/fundamentals/engage-and-retain/web-app-manifest/.
[61] A. Osmani and M. Gaunt, "Instant loading web apps with an application shell architecture," 2017, https://developers.google.com/web/updates/2015/11/app-shell.
[62] A. Osmani, "The PRPL pattern," 2017, https://developers.google.com/web/fundamentals/performance/prpl-pattern/.
[63] M. Gaunt, "Service workers: an introduction," 2017, https://developers.google.com/web/fundamentals/getting-started/primers/service-workers.
[64] M. Nebeling and M. C. Norrie, "Responsive design and development: Methods, technologies and current issues," in *Proc. ICWE*. Springer, 2013, pp. 510–513.
[65] "Lighthouse," 2017, https://developers.google.com/web/tools/lighthouse/.
[66] Google, "Pwa dev summit 2016," 2016, https://events.withgoogle.com/progressive-web-app-dev-summit/.
[67] D. A. Hume, *Progressive Web Apps*. Manning, 2017, https://www.manning.com/books/progressive-web-apps.
[68] T. Ater, *Building Progressive Web Apps: Bringing the Power of Native to the Browser*. O'Reilly Media, 2017, http://shop.oreilly.com/product/0636920052067.do.
[69] M. Asay, "Apple could lose billions on progressive web apps, but it has no choice," 2017, http://www.techrepublic.com/article/apple-could-lose-billions-on-progressive-web-apps-but-it-has-no-choice/.
[70] T. A. Majchrzak and T. Hillmann, "Offline-Provisioning and Synchronization of Content for Mobile Webapps," in *Proc. 11th WEBIST*. SciTePress, 2015, pp. 601–612.
[71] S. J. Vaughan-Nichols, "Will HTML 5 restandardize the web?" *Computer*, vol. 43, no. 4, pp. 13–15, 2010.
[72] Mozilla Developer Network, "Using the application cache," 2017, https://developer.mozilla.org/en-US/docs/Web/HTML/Using_the_application_cache.
[73] Microsoft, "Hosted web apps," https://developer.microsoft.com/en-us/windows/bridges/hosted-web-apps.
[74] H. Heitkötter, H. Kuchen, and T. A. Majchrzak, "Extending a Model-Driven Cross-Platform Development Approach for Business Apps," *Science of Computer Programming (SCP)*, vol. 97, Part 1, pp. 31–36, 2015.
[75] J. Archibald, "is Serviceworker ready?" 2017, https://jakearchibald.github.io/isserviceworkerready/.
[76] "WebKit Feature Status: Service Workers," 2017, https://webkit.org/status//#specification-service-workers.
[77] S. István, "Serviceworker f2f – 03 apr 2017," 2017, https://www.w3.org/2017/04/03-serviceworkers-minutes.html.
[78] S. Birch and A. Russell, "Progressive web apps: Great experiences everywhere," USA, 2017, https://www.youtube.com/watch?v=m-sCdS0sQO8.
[79] J. Rossi, "The progress of web apps - microsoft edge dev blog," 2016, https://blogs.windows.com/msedgedev/2016/07/08/the-progress-of-web-apps/.
[80] A. Osmani, "Production progressive web apps with JavaScript frameworks," USA, 2017, https://www.youtube.com/watch?v=aCMbSyngXB4.
[81] "Hacker news readers as progressive web apps," 2017, https://www.hnpwa.com.
[82] Callahan and Dan, "Firefox 52: Introducing web assembly. . . ," 2017, https://hacks.mozilla.org/2017/03/firefox-52-introducing-web-assembly-css-grid-and-the-grid-inspector/.
[83] O. Campbell-Moore, "Creating UX that "just feels right" with progressive web apps," USA, 2017, https://www.youtube.com/watch?v=mmq-KVeO-uU.
[84] A. Fluin and S. Rickabaugh, "Great progressive web app experiences with angular," USA, 2017, https://www.youtube.com/watch?v=C8KcW1Nj3Mw.
[85] B. Bidelman and E. Kenny, "Staying off the rocks: Using lighthouse to build seaworthy progressive web apps," USA, 2017, https://www.youtube.com/watch?v=NoRYn6gOtVo.
[86] J. Gasperowicz and E. Posnic, "WomenTechmakers.com—A progressive web app migration," USA, 2017, https://www.youtube.com/watch?v=fyi7auD5MzU.
[87] "Jetbrains/kotlin-native," 2017, https://github.com/JetBrains/kotlin-native.
[88] A. Breslav, "Kotlin/Native tech preview: Kotlin without a VM," 2017, https://blog.jetbrains.com/kotlin/2017/04/kotlinnative-tech-preview-kotlin-without-a-vm/.
[89] P. Smutný, "Mobile development tools and cross-platform solutions," in *Proc. 13th ICCC*. IEEE, 2012, pp. 653–656.
[90] I. Malavolta, S. Ruberto, T. Soru, and V. Terragni, "Hybrid mobile apps in the google play store: An exploratory investigation," in *Proc. 2nd MOBILESoft*. IEEE Press, 2015, pp. 56–59.
[91] "Nativescript," 2017, https://www.nativescript.org/.
[92] R. Stoenescu, "Quasar framework," 2017, http://www.quasar-framework.org/.
[93] "Weex," 2017, https://weex.apache.org/.
[94] S. Jobs, "Thoughts on flash," 2017, https://www.apple.com/hotnews/thoughts-on-flash/.