Journal of Artificial Intelligence General Science (JAIGS)

ISSN: 3006-4023 (Online), Volume 6 , Issue 1, 2024 DOI: 10.60087

Home page https://ojs.boulibrary.com/index.php/JAIGS

Comprehensive Comparative Analysis of Mobile Apps Development Approaches

Maher Gerges[1], Ahmed Elgalb[2]
[1,2]Independence Researcher

Abstract
Growing use of the cell phones and tablets over the computer for humans' daily life has increased the development of mobile apps. Different paradigms have been introduced to develop a mobile app. Up till now, the major paradigms have been introduced are native apps, hybrid apps, web app and the new trend namely progressive web app (PWA). Each methodology has its pros and cons. This paper discusses about native development issues and how web app aimed to solve these problems. The hybrid apps will be discussed as a solution of cross-platform development problem of native apps. In addition, problems of web apps and the gap between web app and native apps will be introduced. PWA is supposed to bridge the gap between native apps and web apps. The main technologies –service worker- will also be discussed.

Keywords:
Mobile App Development, Development Approaches, Comparative Analysis,Cross-Platform Development, Native App Development, Hybrid App Development, Progressive Web Apps (PWAs)

ARTICLE INFO: *Received: 19.10.2024 Accepted: 10.11.2024 Published: 03.12.2024*

I. INTRODUCTION

The use of cell phones has been increased over the last few years, since the smart phone was introduced. The use of the smart phones is no longer limited on calling and texting. The smart phone has become part of our lives and it became easier and faster to use the mobile for many tasks. The use of mobile phones is dramatically increased over the last few years. Figure 1 shows the use of mobile forecast from 2003 till 2021. Today, in the USA, 67% of the time spent on the didital media is spent on mobile devices[3][6].
Mobile apps have the large share of mobile usage. Almost for every life situation and every workflow there is now an app. throughout the last few years, there has been rapid development in mobile apps and mobile webpages. This have increased the importance of developing good user experience, high performance, and resources consumption efficient apps [6]. So that many approaches have been adopted. Mobile apps are broadly categorized into mainly native apps and cross-platform apps. The cross-platform apps are categorized into runtime environment and generative approaches. The runtime environment apps are web apps, hybrid apps and - the new enhanced web app- PWA. While generative approaches are Model Driven Software Development (MDSD) and transpiling [5]. Figure 2 shows the classification of mobile apps development approaches. In this paper we will only focus on the native apps and the runtime environment approaches.

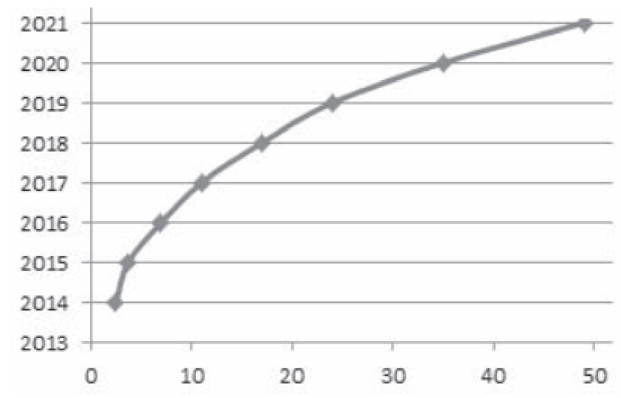

Figure 1 Mobile traffic forecast world-wide (by years 2013-2021) [6]

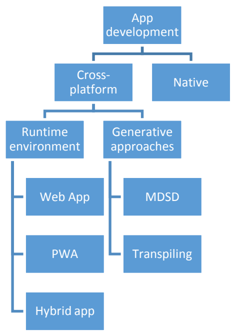

Figure 2 shows the classification of mobile app development approaches

For a long time, the main choice for developing mobile app was native development. In the native development, an application is built for each platform. There are many problems with adopting native development [7]. These problems could be summarized into two main problems -platform dependency and resources consumption. Currently, mobile operating systems are dominated by the duopoly of IOS and Android. In order to develop a native app, at least two teams are required to develop the app to reach the most used mobile OS nowadays –IOS and Android. In order to use the native app, users has to download the app from the play store which consumes many resources –in the core of these resources storage and energy [6] [7]. Native development will be investigated in more details in a separate section.

The web app was proposed to overcome native app issues. A web app is a normal website built using web technologies CSS, HTML and JavaScript and runs through the mobile browser. The web app differ from the normal website in terms of user experience; it gives the user the feel and look of a native app. When the user scroll down, the web app render on full screen mode. However the web app seems to solve the platform dependency problem and the resources consumption, it caused other problems. Push notification, quick access, preloading and cashing are the core issues of the web app [7].
The PWA was introduced as a solution to bridge the gap between native apps and web apps. PWA combines the best of both web apps and native apps. The PWA is a special type of web app [5]. While the PWA accessed for the first time through the mobile browser, it provides functionality to be added to home screen. This supports the quick access –like the native apps- through web app Manifest. The so-called service worker – set of APIs supported by the mobile browsers- enables the push notification, preloading and cashing functionality [4]. Service worker will be discussed in more details in a separate section.

II. Native Development

Native apps are platform dependent apps that developed specifically based on the operation system of the mobile phone and uploaded on the store of each platform like apple store and play store. Native application is implemented in a device specific programming language and integrated development environment (IDE). For example, IOS app is usually written in one of two main languages either Swift or Objective-c in the IDE XCode. While Android apps are usually written in Java using the Android Studio IDE. The least used one is the Windows platform, also known as Universal Windows Platform app, written in languages like C++, C#, Visual Basic and JavaScript[9]. Native applications have full and direct access to the native platform functionalities and are capable of delivering the best performance and user experience on each platform. This approach affords minimum code reuse, and is the most expensive option in cost and time [14] [15].
Native apps are generally installed through app stores on mobile phones and have rich access to device hardware through platform specific APIs. For a long time, the main choice for developing mobile app was native software development kit (SDK). The SDK offers a development experience tailored to the platform. Device hardware resources, communication, location, security, graphics, etc. can be used via Application Programming Interface (API) [7].
Since programming languages and tools for used to develop native apps are platform-specific, as code written for one mobile platform cannot be used on another; Which makes the development and maintenance of native apps for multiple platforms one of the major technical challenges affecting the mobile development community **[11].**

A. *Native apps pros*

- Performance: One of the most effective benefits of native apps is performance. Hardware-near and apps and apps with high graphics like games select the native approach because they need low latency levels. This enables processor-intensive apps to be successfully used on a regular basis. Since they're developed specifically for a given operating system, native apps are quicker and more refined than hybrid apps [7].
- Enriches users' functionality: Since a native app works on the operating system of the device, it's able to make use of the capabilities that are available to it. With native apps, you can build the foremost out of the functionalities offered by mobile devices, for example, calendar, camera, even push notifications, microphone, GPS, and others [14].

B. *Native apps cons*

- Platform dependency: as we mentioned above, native apps are platform dependent. It is developed specifically for a certain platform IOS is different from Android which cost time and money.
- Resources consumption: native apps are resources consumers on both levels development level and usage level. It takes more resources and time to develop, maintain and distribute applications for specific platforms. Additionally, for the enterprise to develop an application they need two teams IOS and Android. On the other level, native apps use much device resources like storage, memory, and what is more important is energy. We will talk about energy in more details in a separate section. This is a huge disadvantage for both companies and developers [3].

III. HYBRID APPS

Since native apps are platform dependent and code used for developing native app for android does not work for IOS. As a result, the development and maintenance of native apps for multiple platforms is one of the major technical challenges affecting native apps development. So there has to be a solution for cross-platform development. Code should be written once and work on all platforms. Different approaches has been adopted to make the apps works on all platforms. One of these approaches is the so-called web-based hybrid mobile apps. In this context, web-based hybrid mobile apps allow developers to use normal web technologies such as HTML, CSS, and JavaScript to develop an app and wrap the app in via cross-platform wrappers –frameworks- and tools. In other words, hybrid app is developed using web languages and wrapped as native app [8].

It is called hybrid because it is developed using web technologies and perform like native. It provide a good user experience just like the native app. There are many excellent hybrid app frameworks nowadays like Cordova, Phonegap, Titanium, Ionic, etc. one of the most commonly used framework is Cordova. Just like most hybrid app frameworks, Cordova uses HTML5, CSS, and JavaScript, so it means that hybrid app is implemented by web technologies. HTML and CSS to implement the user interface, while JavaScript and third-party plugins are used to access the device features (e.g., camera and GPS) by interacting with certain native operation system's API and the API would in turn communicate with the system to realize certain functionality [7]. Figure 3 shows the architecure of cortova framwork.

A. *Hybrid apps pros*

- Cross-platform portability
- Quick access, good performance, and look and feel like native apps
- Reuse of existing knowledge of web developers, simpler and less expensive development processes [8].

B. *Hybrid apps cons*

- The installation time of hybrid app is 22.6% longer than the native as it is larger in size [7].
- Restricted access to hardware features not like native apps.
- High resources consumption just like native apps and even more [7]. More detailed explanation of this is discussed in the next section (hybrid vs native).
- Has to be downloaded from the app store [8].

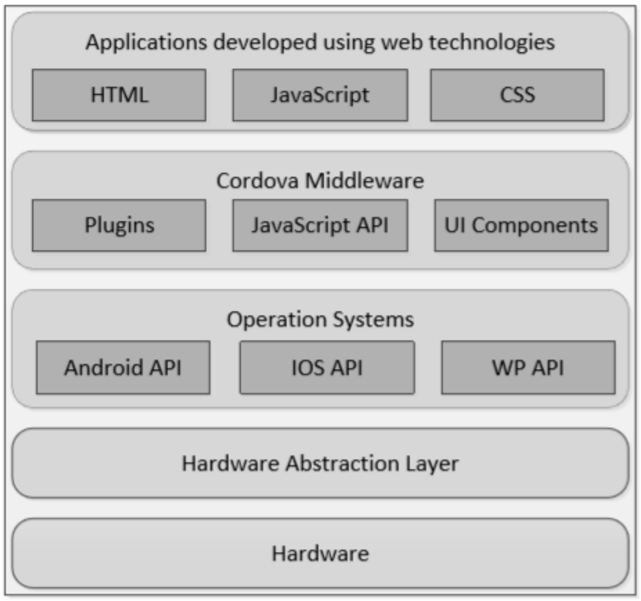

Figure 3 Cordova framework Architecture [7]

IV. HYBRID VS NATIVE

As discussed above, native development requires developers to use different languages and tools to develop the app for certain platform, Java for Android and Object-C for IOS. This raised the need to develop cross-platform app that can be coded once and run over different platforms [6]. Hybrid app is a good choice that meets the cross-platform demand. However hybrid mobile app is a good solution to cross-platform portability which could be developed using simple web technologies; but on the other side it suffers from a number of shortcomings such as restricted access to hardware features, variations on user experience, and decrease in performance [11]. Multiple companies like IBM and large group of developers have adopted the hybrid app development as a solution for platform fragmentation [9].
A study made by [8] on dataset of mobile apps on google play store contains 445 hybrid mobile apps, counting for a 3.73%. The result clearly shows that hybrid mobile apps are significantly uncommon among the top-500 apps within 25 Google Play categories.
The hybrid apps' average end user ratings is 3.75 while for native apps is 3.35. These ratings should not be a surprise; since both native and hybrid apps have the same look and feel. Also, users' ratings are mainly about the service provided by the app. More interestingly, hybrid apps and native apps are performing equally with respect to end users' star-rating across all categories, with ineffective differences [10]. Regardless of some exceptions like

games, apps developed using a hybrid development framework or apps developed natively are almost the same with respect to end users' perception of the app. The performance of the hybrid apps is varied from the native apps in some cases. It is noted that hybrid apps related to Book & References category are even reviewed positively, and there are less reviews signaling bad performance of hybrid apps in the Travel & Local, Communication, and Media & Video categories. But there are more negative reviews with respect to the performance of hybrid apps in the categories in Shopping, Health & Fitness, Transportation, Photography category [11] [12].

Table 1 shows that the installation time hybrid app is 22.6% longer than the native; since its package size is bigger than native app. And regarding the load of plugins hybrid app is 42.3% slower than native. Instead of accessing system features directly in native apps, hybrid app has to transfer the JavaScript code to native request so it costs more CPU resources (106% higher), memory space (73.0% higher) and power (temperature 20.7% higher). While the network flow of both native and hybrid is almost equal because there is no difference between the procedures of both apps' net request [7].

Performance parameter	***Hybrid app***	***Native app***
Installation Consuming Time (Second)	9.37	7.64
Start-up Consuming Time (Second)	1.58	1.11
CPU Occupancy Ratio (%)	11.76	5.54
Memory Occupancy (MB)	101.66	58.77
Battery Temperature (ć)	45.32	37.54
Network Flow (KB)	330.56	323.89

Table 1 hybrid apps vs native apps [7]

V. WEB APP

A mobile web app is built using known web technologies like HTML5, CSS3 and JavaScript; but instead of installing them on mobile phone like native apps they are hosted and served from remote web servers and displayed with web browsers installed on these devices like websites. It is simply served through the HTTP protocol like website, and accessed by end users via a unique URL. Since these apps are written with standard languages for browsers, it can work in different platforms. With HTML5 APIs there is some support of accessing mobile hardware, e.g., camera and GPS, but it is not at the same level as for native. In other words, mobile web apps are mobile-optimized websites accessed via the browser apps installed on end users' mobile devices. When scrolled down, It supports the full screen mode just like the native app. Web apps gives the user the feel and look of the native apps [14] [9].

A. Web apps pros

- Platform independent: Web apps are platform independent websites which in many ways look and feel like native. Web apps are run by a browser and typically written in HTML5.
- Due to the improvement of the browsers, the performance of the web apps have been improved.
- Efficient resources consumption: not like the native apps, web apps are served from a remote host server which save the device resources and storage. In addition, only need basics of web technologies to develop a web apps which save resources like time and money.
- No installation: accessed through mobile browser using URL.

B. Web apps cons

- Device hardware access: Native apps have full access to the device hardware while web apps still lack some of the access.

- Since browsers differ in W3C HTML and JavaScript specification implementation and compliance, a Web app does not perform as expected across all browsers.
- No quick access: not like native apps which have an icon to enable quick access; web apps only accessed using URL.
- Web apps are behind native apps in terms of performance, reliability and engagement. Businesses and developers often see the need to develop native mobile applications to overcome the limitations that the web as a platform imposes on mobile devices.
- Internet reliance: web apps cannot work in offline mode or low network connection.

VI. Progressive web app (PWA)

Since both of web apps and native apps have shortcomings, so there was a need for an app which can combines the capabilities and advantages of native apps that served through web like web apps. In other words, PWAs are special web apps that is developed to overcome web apps disadvantages and add the advantages of the native apps. It combines the best of the web and the best of the native apps. It was first introduced in the Google I/O developer conference in May 2016 in San Francisco [4]. Just like the mobile web apps, PWAs are served from a remote server and accessed for the first time from the mobile browser using normal URL. But when the user visit the PWA for the first time, it gives the user the permission to be added to the home screen for quick access later on. PWAs use https, Service Workers, Web App Manifest and Push notification to give the user the experience of native apps. As PWA is a web app, it can be used in all platforms, reducing the cost and resources and increasing reachability. The implementation details of PWAs can be summarized in three main steps [5].

- Use HTTPS
- Service worker
- Web app manifest

The use of https over http is basically security constraints.

PWAs are becoming popular due to their native App-like experience and are powered by Web Manifest and Service Worker. PWAs reduce usage of bandwidth and improve response time. Service workers in mobiles enables the app to work offline or on low networks connectivity. A webpage is considered as Progressive Web app if it satisfies the mentioned conditions. It originates from a secure origin, loads while offline using service workers, gives reference to a web manifest and has an icon [2].

A. *Service worker*

At the core of the PWA is the concept of service worker which allows developers to programmatically caching, preloading and managing push notifications. Service workers are a group of application programming interfaces (APIs) that are used for caching and preloading assets and data, managing push notifications, and other services [1]. A service worker is a special kind of web worker that is implemented in a dedicated JavaScript file and runs in a separate thread with respect to the main JavaScript thread. Technically, a service worker is a JavaScript module that runs on a separate thread and provides event-driven background processing (e.g. reaction to the receiving of a push notification). A service worker can listen to events dispatched from the main thread [3]. The push event is raised when a push notification is received from a remote server. Service worker could be considered as client-side proxies between a web page, browser and, if available, a network. The service worker do not have the access to directly read or do any changes to the document object model (DOM). Instead a typical usage for it is to listen for events triggers from the page it is registered on [2][3].

Service Workers are the technical way for enabling background syncs, push notifications, the offline approach features. Background sync lets you defer actions until the user has stable connectivity. This ensures whatever user wants to send is actually sent when connectivity improves. Through the caching mechanism provided by the service worker, the user can experience the offline browsing capability. Service workers are also enable push notification [13].

Service Worker is a script that works on browser background without. Also, it acts as an intermediate or proxy that works on the user side [1]. It controls how network requests from your page are handled by intercepting them. Figure 4 shows how a service worker works. When a service worker intercepts a request made from a webpage, it triggers a fetch event on the service worker. This either returns a response directly from the network or it can be retrieved from the local cache. [13]

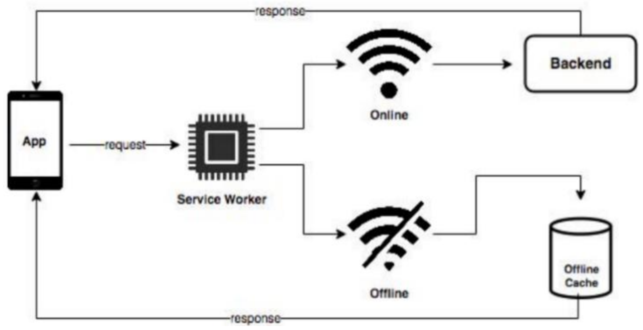

Figure 4 how service workers works [13]

VII. DISCUSSIONS

The native apps are the best in terms of performance, user experience and feel and look. But it is has problems like high cost in development and maintenance. Hybrid apps are a good solution for cross-platform problem. But hybrid apps still have to be downloaded from app stores and they consumes device resources. Web apps are normal websites that gives the feel and look of native apps; when user scroll down it is rendered on the full screen mode. Web apps suffers from the bad performance and does not support quick access, preloading, cashing and does not work on offline or low network connectivity. PWA is a promising technology that aims to combine pros of both native and web apps. PWA is calling the end user to use its service rather than installing the app. User has the choice to install the app after the first access through the browser or not. Table 2 shows a comparison between the four types of apps.

Feature	Web app	PWA	Hybrid	Native
Installable (Quick access)	No	Yes	Yes	Yes
Offline capable	No	Yes	Yes	Yes
Require app store	No	No	Yes	Yes
Push notifications	No	Yes	Yes	Yes
Cross-platform	Yes	yes	Yes	No
Device Hardware and resources access	Very limited	Fine access	Good access	Great access
Background syn-chronisation	No	Yes	Yes	Yes
Development and maintenance cost	Low	fine	high	highest
Size	Low	Low	Very high	high
Performance	Bad	Fine	Great	Best

Table 2 PWA vs Hybrid vs native vs web app

VIII. CONCLUSION

Many different approaches have been developed for creating mobile apps. The best approach is determined based on the type of the app and stakeholders requirements. This paper investigates the different types of mobile apps – native, hybrid, web app, and the newly introduced PWA. The technologies, pros, and cons of each approach has been discussed. Despite the high cost of native approach –developing, testing and maintenance- it is still followed for the high performance especially hardware-near apps. Hybrid apps approach is a good solution for cross-development. But it is still required to be downloaded from the play store and it has high resources consumption. PWA seems to be a promising approach that bridges the gap between native and web apps.

IX. References

[1] I. Malavolta, P. Giuseppe and P. Noorl, "Assessing the Impact of Service Workers on the Energy Efficiency of Progressive Web Apps," Amsterdam, 2017.

[2] N. Pande and A. Somani , "Enhanced Web Application and Browsing performance through Service-worker Infusion Framework," in *IEEE International Conference on Web Services*, Bangalore, 2018.

[3] M. Tim , B.-H. Andreas and G. Tor-Morten , "ProgressiveWebApps:theDefiniteApproachtoCross-PlatformDevelopment?," in *51st Hawaii International Conference on System Sciences*, Hawaii, 2018.

[4] E. N. Lukito and G. . H. Andreas , "Development of Monitoring System for Smart Farming Using Progressive Web App," in *9th International Conference on Information Technology and Electrical Engineering (ICITEE)*, Phuket, 2017.

[5] B.-H. Andreas , . M. A. Tim and G. Tor-Morten , "Progressive Web Apps: The Possible Web-native Unifier for Mobile Development," in *13th International Conference on Web Information Systems and Technologie*, 2017.

[6] A. Luntovskyy, "Advanced Software-Technological Approaches for Mobile Apps Development," in *TCSET*, Lviv-Slavske, 2018.

[7] Q. Peixin , G. Xiao and Z. Maokun , "A Comprehensive Comparison Between Hybrid and Native App Paradigms," in *8th International Conference on Computational Intelligence and Communication Networks*, Beijing, 2016.

[8] M. Ivano , "Web-based Hybrid Mobile Apps:State of the Practice and Research Opportunities," in *IEEE/ACM International Conference on Mobile Software Engineering and Systems*, L'Aquila, 2016.

[9] R. Nunkesser, "Beyond Web/Native/Hybrid: A New Taxonomy for Mobile App Development," in *ACM/IEEE 5th International Conference on Mobile Software Engineering and Systems*, Hamm, 2018.

[10] M. Ivano , R. Stefano , S. Tommaso and T. Valerio , "Hybrid Mobile Apps in the Google Play Store: An Exploratory Investigation," in *2nd ACM International Conference on Mobile Software Engineering and Systems*, 2015.

[11] M. Ivano , R. Stefano , S. Tommaso and T. Valerio , "End Users' Perception of Hybrid Mobile Apps in the Google Play Store," in *IEEE International Conference on Mobile Services*, 2015.

[12] F. Wenping and J. Yang , "Design and Implementation of Cross-platform Friends-Positioning Mobile APP Based on Phonegap and HTML5," in *2nd IEEE International Conference on Computational Intelligence and Applications*, Jinan, 2017.

[13] A. Gambhir and G. Raj , "Analysis of Cache in Service Worker and Performance Scoring of Progressive Web Application," in *International Conference on Advances in Computing and Communication Engineering (ICACCE-2018)* , Paris, 2018.

[14] Y. Ma, X. Liu, Y. Liu and G. Huang, "A Tale of Two Fashions: An Empirical Study on the Performance of Native Apps and Web Apps on Android," in *IEEE TRANSACTIONS ON MOBILE COMPUTING*, 2017.

[15] J. Xiaoping , "A Performance Evaluation of Cross-Platform Mobile Application Development Approaches," in *ACM/IEEE 5th International Conference on Mobile Software Engineering and Systems*, 2018.