IAES Journal of Electrical and Electronics Engineering
Vol. 6, No. 2, December 2019, pp. 33~37
ISSN: 2338-7432, DOI: 10.11591/jeee.v6i2.pp33-37

Web and android technologies: architectures, integration approaches, performance, and security challenges

Deny Nugroho S[1], Watra Arsadiando[2], Tri Wahono[2]
[1]Faculty of Industrial Technology, Universitas Ahmad Dahlan, Yogyakarta, Indonesia
[2]Embedded Systems and Power Electronics Research Group, Yogyakarta, Indonesia

Article Info

Article history:

Received Aug 12, 2018
Revised Jan 2, 2019
Accepted Apr 21, 2019

Keywords:

Android
Hybrid applications
Mobile applications
Mobile security
Progressive web apps
REST API
Web technology

ABSTRACT

Web technologies and the Android platform have become closely connected in the delivery of modern digital services. Websites provide broad reach through browsers, while Android applications offer persistent installation, deeper device integration, and access to platform-specific capabilities. Yet developers face recurring decisions about architecture, interoperability, performance, security, maintainability, and user experience when selecting between web, native, and hybrid approaches. This review examines research published up to 2018 on web and Android development, focusing on mobile web applications, WebView-based hybrid systems, progressive web applications, cross-platform frameworks, RESTful services, and Android security. The literature indicates that web-based approaches can reduce duplicated development effort and simplify content distribution, whereas native Android development generally offers more direct access to device functions and platform conventions. Hybrid architectures provide a practical compromise but introduce integration and security risks at the boundary between JavaScript and native code. Progressive web applications extend browser capabilities through service workers, caching, and installation features, although platform support and device access remain uneven. The review concludes that architecture selection should follow application requirements rather than a universal preference, with systematic attention to security, performance, accessibility, and long-term maintenance.

This is an open access article under the CC BY-SA license.

1. INTRODUCTION

The growth of internet access and smartphone adoption has transformed how individuals interact with digital information and services. Websites that were originally designed for desktop computers have increasingly evolved into responsive platforms that accommodate mobile browsers and different screen sizes. At the same time, Android has become an important environment for delivering applications that use device capabilities such as cameras, location services, sensors, and local storage. These developments have created a close relationship between web technologies and mobile application engineering, while introducing important architectural and technical decisions for developers [1].

One central challenge is deciding whether a service should be developed as a website, a native Android application, or a hybrid solution. Web applications can be accessed through browsers without requiring conventional installation, while native applications can integrate more directly with the operating system. Hybrid applications attempt to combine these advantages by embedding web technologies inside a

native application container. This review examines these approaches, their integration mechanisms, and their limitations using research published no later than 2018 [2], [3].

2. REVIEW SCOPE AND RESEARCH PROBLEMS

This review focuses on research concerning web-based application development, Android application architectures, cross-platform development, web service integration, and mobile security. The literature includes peer-reviewed journal articles and conference proceedings that provide identifiable DOI records and were published by the end of 2018. The discussion is organized around application architecture, communication between web and mobile components, performance and resource consumption, and security. This structure provides a path from the main development problems to the technical comparisons and conclusions.

Four related problems guide the review. First, developers must manage platform fragmentation without creating excessive development and maintenance costs. Second, web and Android components must exchange data reliably despite differences in programming environments, network conditions, and application lifecycles. Third, applications must maintain acceptable responsiveness and resource efficiency on mobile devices with limited processing power and battery capacity. Finally, security and privacy must be preserved when applications handle user data, permissions, network communication, and embedded web content [4], [5].

3. WEB AND ANDROID APPLICATION ARCHITECTURES

3.1. Web-based applications

Web applications generally use browser technologies such as HTML, CSS, and JavaScript to present content and support user interaction. Their main advantage is broad accessibility because users can access services through a compatible browser without installing a separate application for every operating system. Developers can also update server-side content and functionality without requiring users to download a new application package. However, browser-based applications may have limited access to device-specific features and can be affected by browser compatibility, network availability, and performance constraints [6].

The development of mobile web applications has made websites more adaptable to smartphone usage. Responsive design allows interfaces to adjust to different screen dimensions, while mobile-oriented interaction patterns improve navigation on touchscreens. Jobe's comparison of native and mobile web applications demonstrated that the suitability of a web approach depends strongly on the application's requirements, particularly when extensive hardware interaction is needed [7]. Therefore, web applications are attractive for information delivery and service access, but they may be less suitable when applications require intensive use of device-specific capabilities.

3.2. Native android applications

Native Android applications are developed for the Android operating system and can use its application programming interfaces to access hardware and system services. This approach offers direct integration with platform features, including notifications, location services, cameras, sensors, background processing, and local databases. Native development also supports interaction patterns that are familiar to Android users and can provide strong performance for tasks requiring intensive device interaction. Nevertheless, native applications require platform-specific development and maintenance, which can increase costs when similar services must be delivered across multiple operating systems [3], [8].

The choice of native development should therefore be based on functional requirements rather than an assumption that native applications are always superior. Applications that depend on hardware access, intensive processing, or detailed control over operating-system behavior may benefit from native implementation. Conversely, services centered on content presentation, forms, and common online transactions may be implemented effectively through web technologies. The development decision should consider the expected user experience, available resources, maintenance requirements, and target devices [9].

3.3. Hybrid applications and WebView

Hybrid applications combine web technologies with a native application container. Android WebView allows developers to display HTML content and execute JavaScript inside an Android application, creating a bridge between web-based interfaces and native functionality. This approach can reduce duplicated interface development and allow developers to reuse web components across different platforms. Adinugroho *et al.* [10] examined WebView-based development for a learning management system and discussed its potential for combining web technologies with mobile application delivery.

However, hybrid development introduces additional architectural complexity. The application must coordinate JavaScript execution, native code, device permissions, and communication between the WebView component and the Android environment. A poorly designed bridge can expose sensitive functions to untrusted web content or create inconsistent behavior across devices. Hybrid applications therefore require careful control of the content loaded into WebView and the native interfaces exposed to JavaScript [11].

4. PROGRESSIVE WEB APPLICATIONS AND CROSS-PLATFORM DEVELOPMENT

Progressive Web Applications represent an effort to extend browser-based applications with capabilities commonly associated with installed applications. Service workers can intercept network requests and support caching, while web application manifests provide information used for installation and presentation. These mechanisms can improve loading behavior and allow selected functions to remain available during temporary network interruptions. Biørn-Hansen et al. discussed the potential of Progressive Web Apps as a unifying approach for mobile development, while also examining their performance and technical limitations [12].

Further work has explored how progressive web technologies can support background synchronization and application responsiveness. Behl and Raj described an architectural pattern involving service workers, push notifications, and background synchronization to provide a more application-like web experience [13]. Fortunato and Bernardino compared Progressive Web Apps with native mobile applications and discussed the capabilities and limitations of this approach across devices [14]. Despite these developments, browser support and access to platform-specific functions remained important considerations within the 2018 technological landscape.

Cross-platform development frameworks provide another way to reduce the cost of producing applications for multiple operating systems. These frameworks differ in how they translate, interpret, or render application code, and their implementation choices influence performance, user-interface consistency, and access to native functions. Heitkötter et al. evaluated cross-platform approaches and identified trade-offs between development effort and platform-specific capabilities [8]. Latif *et al.* [9] and Biørn-Hansen *et al.* [15] further discussed the approaches, classifications, and research challenges associated with cross-platform mobile development.

5. WEB SERVICE INTEGRATION AND DATA COMMUNICATION

Communication between Android applications and web-based services is a central requirement for many modern digital systems. RESTful services provide an architectural approach in which clients interact with resources through standardized HTTP methods and commonly exchange data using JSON. This arrangement allows an Android application to retrieve or submit information without embedding the complete business logic and data management functions within the mobile client. Oumaziz et al. examined REST API usage in Android applications and identified practical patterns in the way mobile applications communicate with web services [16].

The quality of an API is important because poorly designed interfaces can create compatibility problems and increase the maintenance burden for application developers. Neumann *et al.* [17] analyzed public REST APIs and highlighted issues associated with API design and implementation practices. For Android applications, reliable service integration also requires appropriate handling of network failures, timeouts, authentication, data serialization, and asynchronous operations. A well-designed client should communicate clearly with the server while avoiding unnecessary requests and protecting sensitive information during transmission.

6. PERFORMANCE, RESOURCE CONSUMPTION, AND USER EXPERIENCE

Application performance is influenced by the architecture, implementation framework, network conditions, and hardware characteristics of the target device. Native applications can avoid some browser-rendering and bridging overhead, while hybrid and web applications may benefit from code reuse and centralized updates. Grzmil *et al.* [18] compared the performance of native and cross-platform mobile applications, demonstrating the importance of evaluating actual application behavior rather than relying solely on development convenience. The results of such comparisons should be interpreted in relation to the specific framework, application workload, and test environment.

Energy consumption is another important factor because mobile devices depend on limited battery capacity. Application behavior such as frequent network access, background processing, intensive computation, and unnecessary sensor usage can increase energy demand. Li and Halfond investigated energy-saving programming practices for Android applications and demonstrated that software design

decisions can influence power consumption [19]. Developers should therefore consider resource efficiency alongside response time, interface quality, and functional completeness.

User experience also depends on consistency, accessibility, and the suitability of interaction patterns for mobile devices. A web interface embedded in an Android application may not always behave like a native interface, particularly when users expect platform-specific navigation, gestures, or notifications. Ahti *et al.* [20] evaluated cross-platform development tools and highlighted the importance of usability and interface quality when comparing development approaches. These findings support the need for user-centered testing across actual devices rather than relying exclusively on code reuse or development speed as measures of success.

7. SECURITY AND PRIVACY CHALLENGES

Security is a fundamental requirement for web and Android applications because both environments process information that may be sensitive or personally identifiable. Android uses a permission-based security model to restrict application access to protected resources, but permission management can be difficult for both developers and users. Fang et al. reviewed issues associated with Android permission mechanisms, including excessive permissions and weaknesses in permission administration [21]. Bagheri et al. subsequently examined the Android permission protocol through formal modeling and demonstrated the value of systematic analysis for identifying potential security flaws [22].

Hybrid applications introduce additional risks because web content can interact with native application components through WebView interfaces. Luo et al. investigated attacks against WebView in Android and showed that the interaction between web content and native application functionality can create security vulnerabilities [23]. Developers should restrict access to trusted content, validate data received from web pages, avoid exposing unnecessary native methods, and maintain secure communication channels. Security testing should cover both the web application and the Android components because weaknesses in either environment can affect the complete system.

8. DISCUSSION AND FUTURE RESEARCH DIRECTIONS

The reviewed literature shows that no single architecture is appropriate for every web and Android application. Web applications provide accessibility and centralized maintenance, while native Android development offers deeper device integration and direct access to platform capabilities. Hybrid systems can reduce duplicated development effort, but they require careful management of the boundary between web and native components. Progressive Web Apps extend web functionality, although their suitability depends on browser support and the application's need for device-level features.

Future research should emphasize empirical comparisons using consistent evaluation criteria. Performance, energy consumption, security, accessibility, maintainability, and user satisfaction should be measured under realistic usage conditions. Further work is also needed to improve the security of WebView communication, strengthen API integration practices, and support reliable application behavior under intermittent network connectivity. These directions can help developers select architectures that balance technical quality with practical user needs.

9. CONCLUSION

Web and Android technologies provide complementary approaches to delivering modern digital services. Web applications offer broad accessibility and centralized maintenance, whereas native Android applications provide closer integration with device hardware and operating-system services. Hybrid applications and Progressive Web Apps attempt to bridge these approaches, but their benefits depend on implementation quality, platform capabilities, and application requirements. RESTful services provide an effective communication mechanism between mobile clients and web-based systems, while security and resource efficiency remain essential engineering concerns. The reviewed research indicates that architecture selection should be guided by functional requirements, user experience, performance, security, and long-term maintenance rather than by a universal preference for web or native development. A balanced and carefully evaluated architecture is therefore the most practical foundation for reliable web and Android applications.

REFERENCES

[1] A. I. Wasserman, “Software engineering issues for mobile application development,” in *Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research*, 2010, pp. 397–400. doi: 10.1145/1882362.1882443.

[2] S. Xanthopoulos and S. Xinogalos, "A comparative analysis of cross-platform development approaches for mobile applications," in *Proceedings of the 6th Balkan Conference in Informatics*, 2013. doi: 10.1145/2490257.2490292.
[3] H. Heitkötter, S. Hanschke, and T. A. Majchrzak, "Evaluating cross-platform development approaches for mobile applications," in *Web Information Systems and Technologies*, Lecture Notes in Business Information Processing, vol. 140, pp. 120–138, 2013. doi: 10.1007/978-3-642-36608-6_8.
[4] M. Latif, Y. Lakhrissi, E. H. Nfaoui, and N. Es-Sbai, "Cross platform approach for mobile application development: A survey," in *Proceedings of the International Conference on Information Technology for Organizations Development*, 2016. doi: 10.1109/IT4OD.2016.7479278.
[5] A. Biørn-Hansen, T.-M. Grønli, and G. Ghinea, "A survey and taxonomy of core concepts and research challenges in cross-platform mobile development," *ACM Computing Surveys*, vol. 51, no. 5, 2018. doi: 10.1145/3241739.
[6] F. Dalisay, "Issues and challenges of mobile web applications: A systematic literature review," *Southeast Asian Journal of Science and Technology*, vol. 3, no. 1, pp. 153–163, 2018. doi: Not assigned.
[7] W. Jobe, "Native apps vs. mobile web apps," *International Journal of Interactive Mobile Technologies*, vol. 7, no. 4, pp. 27–32, 2013. doi: 10.3991/ijim.v7i4.3226.
[8] H. Heitkötter, T. A. Majchrzak, and H. Kuchen, "Cross-platform model-driven development of mobile applications with MD²," in *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, 2013, pp. 526–533. doi: 10.1145/2480362.2480464.
[9] M. Latif, Y. Lakhrissi, E. H. Nfaoui, and N. Es-Sbai, "Cross platform approach for mobile application development: A survey," in *Proceedings of the International Conference on Information Technology for Organizations Development*, 2016. doi: 10.1109/IT4OD.2016.7479278.
[10] T. Y. Adinugroho, Reina, and J. B. Gautama, "Review of multi-platform mobile application development using WebView: Learning management system on mobile platform," *Procedia Computer Science*, vol. 59, pp. 291–297, 2015. doi: 10.1016/j.procs.2015.07.568.
[11] T. Luo, H. Hao, W. Du, Y. Wang, and H. Yin, "Attacks on WebView in the Android system," in *Proceedings of the 27th Annual Computer Security Applications Conference*, 2011, pp. 343–352. doi: 10.1145/2076732.2076781.
[12] A. Biørn-Hansen, T. A. Majchrzak, and T.-M. Grønli, "Progressive Web Apps: The possible web-native unifier for mobile development," in *Proceedings of the 13th International Conference on Web Information Systems and Technologies*, 2017, pp. 344–351. doi: 10.5220/0006353703440351.
[13] K. Behl and G. Raj, "Architectural pattern of progressive web and background synchronization," in *Proceedings of the 2018 International Conference on Advances in Computing and Communication Engineering*, 2018, pp. 366–371. doi: 10.1109/ICACCE.2018.8441701.
[14] D. Fortunato and J. Bernardino, "Progressive web apps: An alternative to the native mobile apps," in *Proceedings of the 13th Iberian Conference on Information Systems and Technologies*, 2018. doi: 10.23919/CISTI.2018.8399228.
[15] A. Biørn-Hansen, T. A. Majchrzak, and T.-M. Grønli, "Progressive web apps for the unified development of mobile applications," in *Web Information Systems and Technologies*, Lecture Notes in Business Information Processing, 2018. doi: 10.1007/978-3-319-93527-0_4.
[16] M. A. Oumaziz, A. Belkhir, T. Vacher, E. Beaudry, X. Blanc, J. R. Falleri, and N. Moha, "Empirical study on REST APIs usage in Android mobile applications," in *Service-Oriented Computing*, Lecture Notes in Computer Science, vol. 10601, pp. 614–622, 2017. doi: 10.1007/978-3-319-69035-3_45.
[17] A. Neumann, N. Laranjeiro, and J. Bernardino, "An analysis of public REST web service APIs," *IEEE Transactions on Services Computing*, 2018. doi: 10.1109/TSC.2018.2847344.
[18] P. Grzmil, M. Skublewska-Paszkowska, E. Łukasik, and J. Smołka, "Performance analysis of native and cross-platform mobile applications," *Informatyka, Automatyka, Pomiary w Gospodarce i Ochronie Środowiska*, vol. 7, no. 2, 2017. doi: 10.5604/01.3001.0010.4838.
[19] D. Li and W. G. J. Halfond, "An investigation into energy-saving programming practices for Android smartphone app development," in *Proceedings of the 3rd International Workshop on Green and Sustainable Software*, 2014. doi: 10.1145/2593743.2593750.
[20] V. Ahti, S. Hyrynsalmi, and O. Nevalainen, "An evaluation framework for cross-platform mobile app development tools: A case analysis of Adobe PhoneGap framework," in *Proceedings of the 17th International Conference on Computer Systems and Technologies*, 2016, pp. 41–48. doi: 10.1145/2983468.2983484.
[21] Z. Fang, W. Han, and Y. Li, "Permission based Android security: Issues and countermeasures," *Computers & Security*, vol. 43, pp. 205–218, 2014. doi: 10.1016/j.cose.2014.02.007.
[22] H. Bagheri, E. Kang, S. Malek, and D. Jackson, "A formal approach for detection of security flaws in the Android permission system," *Formal Aspects of Computing*, vol. 30, no. 5, pp. 525–544, 2018. doi: 10.1007/s00165-017-0445-z.
[23] M. A. Neumann, N. Laranjeiro, and J. Bernardino, "An analysis of public REST web service APIs," *IEEE Transactions on Services Computing*, 2018. doi: 10.1109/TSC.2018.2847344.