JOURNAL OF SOFTWARE ENGINEERING AND COMPUTING
JSEC

Progressive Web Applications (PWAs): Architecture, Functioning, and Applications

Gregorio Sebastián Gualavisí González
Ingeniero Software, Universidad Politécnica Salesiana, Sede Cuenca, Ecuador
ggualavisig@est.ups.edu.ec
ORCID: 0009-0005-0351-2831

Edwin Rodrigo Ramos Zurita
Ingeniero en Telecomunicaciones, Universidad Técnica de Ambato, Ecuador
edramos@uta.edu.ec
ORCID: 0009-0008-0869-1738

Lisbeth Alexandra Gavilanez López
Estudiante de Medicina, Universidad Técnica de Ambato, Ecuador
lgavinalez1371@uta.edu.ec
ORCID: 0009-0005-0351-2831

ABSTRACT

Progressive Web Applications (PWAs) represent a modern approach to web development that integrates the accessibility of traditional websites with the functionality and user experience of native mobile applications. PWAs are built using standard web technologies such as HTML, CSS, and JavaScript, combined with advanced browser capabilities including Service Workers, Web App Manifest, and secure HTTPS connections. These technologies allow web applications to provide features typically associated with native apps, such as offline functionality, push notifications, background synchronization, and the ability to be installed directly on a user's device.

The main objective of Progressive Web Applications is to enhance performance, reliability, and user engagement while maintaining the simplicity and universality of web platforms. Through intelligent caching mechanisms managed by Service Workers, PWAs can load quickly even under unstable network conditions and continue functioning when the device is offline. Additionally, the Web App Manifest enables developers to define how the application appears and behaves when installed, allowing users to access the application from their home screen without relying on traditional app stores.

Keywords: *Serverless computing · Web applications · Cloud computing · FaaS · Scalable architectures*

1. INTRODUCCIÓN

The rapid expansion of mobile technologies has significantly transformed the way users interact with digital services and web-based platforms. In recent years, the demand for applications that provide fast, reliable, and engaging experiences across multiple devices has increased considerably. Traditional web applications, while widely accessible, often lack the responsiveness and advanced capabilities of native mobile applications. As a result, new approaches to web development have emerged to bridge the gap between web and native environments. Among these approaches, Progressive Web Applications (PWAs) have gained considerable attention in both academic research and industry practice (Biørn-Hansen et al., 2024).

The concept of Progressive Web Applications was introduced to enhance the capabilities of conventional web applications through the integration of modern browser technologies. PWAs are designed to provide a user experience comparable to native applications while maintaining the accessibility and flexibility of web platforms. They leverage technologies such as Service Workers, Web App Manifest files, and secure HTTPS protocols to enable advanced functionalities including offline operation, push notifications, and background synchronization (Malavolta, 2023).

One of the key motivations behind the development of PWAs is the need to address limitations associated with traditional mobile application development. Native applications require separate development processes for different platforms such as Android and iOS, which increases both development time and maintenance costs. In contrast, PWAs are built using standard web technologies such as HTML, CSS, and JavaScript, allowing developers to deploy a single codebase across multiple platforms (Cardieri et al., 2024).

Furthermore, the increasing reliance on mobile devices has created significant challenges related to performance and network reliability. Many users access digital services through mobile networks with varying levels of connectivity. Under these conditions, traditional web applications often suffer from slow loading times and unreliable performance. PWAs address these issues through intelligent caching mechanisms implemented via Service Workers, enabling applications to function even in low-connectivity environments (Pérez et al., 2024).

Another important aspect of PWAs is their ability to provide installation capabilities without relying on traditional application distribution platforms. Users can install PWAs directly from their 3

web browsers and access them from their device home screens. This eliminates the need for app store downloads and simplifies the process of application distribution. Consequently, PWAs reduce barriers to adoption and improve accessibility for users worldwide (Luntovskyy & Gütter, 2023).

From a technological perspective, PWAs represent a convergence of web standards and modern software engineering practices. The architecture of a PWA is typically composed of three core components: a secure HTTPS environment, a Web App Manifest file, and a Service Worker responsible for managing caching strategies and network requests. These elements work together to ensure application reliability and performance (Malavolta & Tamburri, 2023).

The role of Service Workers is particularly important within the PWA architecture. Service Workers operate as background scripts that intercept network requests and manage caching strategies. Through this mechanism, developers can control how resources are retrieved and stored locally, allowing the application to provide faster loading times and offline functionality. This capability represents one of the defining characteristics of Progressive Web Applications (Biørn-Hansen et al., 2024).

In addition to performance improvements, PWAs also contribute to enhanced user engagement. Modern web APIs allow developers to integrate features such as push notifications, background synchronization, and device integration. These capabilities enable PWAs to deliver interactive experiences similar to those provided by native mobile applications. As a result, user engagement and retention rates can be significantly improved (Cardieri et al., 2024).

Recent studies have shown that organizations adopting PWAs often experience measurable improvements in application performance and user interaction metrics. For instance, improvements in page loading speed and responsiveness have been linked to increased user satisfaction and higher conversion rates in digital platforms. These findings highlight the potential of PWAs to transform modern web development practices (Malavolta, 2023).

Another important advantage of PWAs is their ability to support cross-platform compatibility. Because PWAs operate within web browsers, they can function across multiple operating systems including Android, iOS, Windows, and Linux without requiring platform-specific development. This significantly simplifies the development process and allows

organizations to reach a broader audience (Luntovskyy & Gütter, 2023). 4

Despite these advantages, the adoption of PWAs also presents certain challenges. One limitation involves the restricted access to certain hardware features compared to native applications. Although modern web APIs continue to expand, some device capabilities remain partially supported or unavailable in browser environments (Pérez et al., 2024).

Another challenge relates to browser compatibility and implementation differences among platforms. While major browsers such as Google Chrome, Microsoft Edge, and Firefox offer strong support for PWA technologies, other environments may provide limited functionality. This variability can create challenges for developers seeking consistent cross-platform performance (Biørn-Hansen et al., 2024).

Security considerations also play a critical role in the implementation of PWAs. Because these applications rely heavily on web technologies, they must operate within secure environments using HTTPS protocols. Secure communication channels are necessary to protect user data and ensure the integrity of cached resources (Malavolta & Tamburri, 2023).

In addition to technical considerations, the adoption of PWAs also has implications for software architecture and design methodologies. Developers must carefully design caching strategies, resource management mechanisms, and network request handling to ensure optimal performance. These design decisions require a strong understanding of both web technologies and distributed systems (Cardieri et al., 2024).

The academic community has increasingly focused on evaluating the effectiveness of PWAs in various contexts. Researchers have explored topics such as performance optimization, usability evaluation, security analysis, and architectural design patterns for PWA systems. These studies contribute to a deeper understanding of how PWAs can be effectively implemented in modern software ecosystems (Malavolta, 2023).

Moreover, PWAs have been adopted in a wide range of application domains including e-commerce, education, healthcare, and enterprise information systems. In these contexts, PWAs offer a flexible and scalable solution for delivering digital services across heterogeneous devices and network conditions (Pérez et al., 2024). 5

In the e-commerce sector, PWAs have been particularly successful due to their ability to improve loading speeds and provide seamless user experiences on mobile devices. Retail platforms that have implemented PWAs often report increased engagement and improved conversion rates compared to traditional web applications (Cardieri et al., 2024).

Similarly, educational platforms have begun to adopt PWA technologies to provide students with reliable access to learning resources regardless of connectivity limitations. Offline capabilities allow students to access course materials even in environments with unstable internet connections (Luntovskyy & Gütter, 2023).

Healthcare systems also benefit from the flexibility offered by PWAs. Medical information systems and telemedicine platforms can leverage PWA technologies to deliver secure and responsive services to healthcare professionals and patients across different devices (Malavolta & Tamburri, 2023).

Another emerging application area for PWAs involves enterprise information systems. Organizations increasingly rely on web-based solutions for internal operations, and PWAs provide a cost-effective approach for developing cross-platform enterprise applications with improved performance and reliability (Biørn-Hansen et al., 2024).

The continuous evolution of web standards has further strengthened the capabilities of PWAs. New browser APIs and development frameworks are expanding the range of functionalities available to web developers, enabling more sophisticated and powerful applications (Cardieri et al., 2024).

Technologies such as WebAssembly, Web Bluetooth, and Web NFC are gradually being integrated into modern browsers, providing additional opportunities for PWAs to interact with device hardware. These advancements contribute to narrowing the gap between web applications and native mobile applications (Malavolta, 2023).

From a software engineering perspective, PWAs represent an important step toward the unification of web and mobile development paradigms. By leveraging standardized web technologies, developers can build scalable applications that operate consistently across multiple environments (Pérez et al., 2024). 6

Furthermore, the open nature of web technologies promotes innovation and collaboration among developers worldwide. Open standards ensure that PWA technologies remain accessible and adaptable to evolving technological landscapes (Luntovskyy & Gütter, 2023).

In addition, the economic benefits associated with PWAs make them attractive for organizations with limited development resources. By maintaining a single codebase for multiple platforms, companies can reduce development costs while still delivering high-quality digital services (Cardieri et al., 2024).

Nevertheless, the decision to adopt PWAs should be carefully evaluated based on the specific requirements of each project. In some scenarios, native applications may still offer advantages related to performance or hardware integration (Malavolta & Tamburri, 2023).

Therefore, researchers and practitioners continue to investigate best practices for implementing PWAs effectively. These efforts aim to maximize the benefits of the technology while addressing its current limitations (Biørn-Hansen et al., 2024).

The growing body of research on PWAs indicates that this technology will play a significant role in the future of web and mobile development. As browser capabilities continue to expand, the distinction between web applications and native applications may become increasingly blurred (Malavolta, 2023).

Consequently, Progressive Web Applications represent a promising approach for building modern digital platforms capable of delivering high-performance, reliable, and accessible user experiences across diverse technological environments (Pérez et al., 2024).

2. METHODOLOGY

T

This study adopts a qualitative and analytical research methodology aimed at examining the structure, implementation, and practical performance of Progressive Web Applications (PWAs) in modern web development environments. The methodological approach focuses on analyzing architectural components, development frameworks, and performance characteristics associated with PWA-based systems. The research combines a literature review with a technical evaluation of PWA technologies in order to understand their operational mechanisms and potential benefits compared to traditional web and native applications. 7

The research design follows an exploratory and descriptive framework. Exploratory analysis allows the identification of core technologies involved in the development of PWAs, including Service Workers, Web App Manifest files, caching strategies, and secure communication protocols. At the same time, the descriptive component provides a structured explanation of how these technologies interact within the architecture of modern web applications.

The first stage of the methodology consists of a systematic review of recent academic literature related to Progressive Web Applications. Scientific articles, conference papers, and technical reports published between 2022 and 2024 were analyzed to identify the main architectural principles, implementation techniques, and performance improvements associated with PWAs. The review focused primarily on research indexed in academic databases such as Scopus, IEEE Xplore, and ACM Digital Library in order to ensure the reliability and scientific validity of the sources.

During the literature review process, relevant studies were selected based on several inclusion criteria. First, the publications had to address the design, implementation, or evaluation of Progressive Web Applications. Second, the studies needed to present empirical results or technical analysis related to performance, usability, or system architecture. Finally, the selected sources had to be published in peer-reviewed journals or reputable conference proceedings to maintain academic rigor.

After identifying relevant sources, the selected studies were analyzed to extract key information regarding PWA development practices, architectural models, and performance optimization techniques. Particular attention was given to research discussing Service Worker lifecycle management, caching strategies, and offline capabilities, as these components represent the fundamental mechanisms enabling PWA functionality.

The second stage of the methodology focuses on the technical analysis of PWA architecture. This analysis examines the interaction between the main components involved in Progressive Web Application development. Specifically, the study investigates the relationships between client-side interfaces, Service Workers, cache storage systems, and backend servers. By examining these components, it becomes possible to understand how PWAs maintain reliability and responsiveness under different network conditions.

To support the architectural analysis, a conceptual system model was developed to represent the typical workflow of a PWA environment. This model illustrates how user requests are processed 8

through the browser, intercepted by the Service Worker, and either served from the cache or retrieved from the remote server. The architectural model also highlights how cached

resources contribute to faster loading times and improved application performance.

Another important methodological component involves the analysis of caching strategies used in PWA implementations. Several common caching approaches were examined, including cache-first, network-first, and stale-while-revalidate strategies. Each strategy provides different trade-offs between performance, reliability, and data freshness. By analyzing these strategies, the study identifies which approaches are most suitable for different application scenarios.

The methodology also considers the role of the Web App Manifest in enabling installation capabilities and user interface integration. The manifest file defines application metadata, including icons, display modes, and startup behavior. Evaluating the configuration and usage of manifest files provides insights into how PWAs achieve native-like integration with mobile devices and desktop environments.

In addition to architectural evaluation, the methodology includes an examination of performance factors associated with PWA technologies. Performance indicators such as loading speed, resource caching efficiency, and responsiveness under limited network connectivity were considered. These indicators provide a framework for evaluating the effectiveness of PWA technologies in improving user experience.

The methodological framework also incorporates a comparative analysis between Progressive Web Applications and traditional application models. This comparison examines differences in development complexity, deployment processes, platform compatibility, and maintenance requirements. By comparing these aspects, the research highlights the practical advantages and limitations associated with PWA adoption.

To ensure a comprehensive understanding of the technology, the study also evaluates modern development tools and frameworks commonly used for PWA implementation. Frameworks such as React, Angular, and Vue.js have integrated support for PWA features, allowing developers to streamline the development process. The analysis of these tools provides insights into the practical implementation of Progressive Web Applications in real-world projects. 9

Furthermore, the methodology considers security aspects related to PWA deployment. Since Service Workers operate as background scripts capable of intercepting network requests, secure communication through HTTPS is required to prevent unauthorized access or data manipulation. The analysis therefore includes an evaluation of how HTTPS protocols contribute to maintaining system security and protecting user data.

Another methodological step involves analyzing user experience considerations associated with Progressive Web Applications. User interface responsiveness, application loading time, and interaction smoothness are key factors influencing user engagement. Evaluating these elements helps determine how effectively PWAs replicate the experience of native mobile applications.

The study also examines the scalability of PWA architectures in environments with high user demand. Scalability considerations include server communication efficiency, resource distribution, and the ability to handle simultaneous requests from multiple users. Understanding these factors helps determine the feasibility of PWAs for large-scale digital platforms.

Finally, the collected data from the literature review and architectural analysis were synthesized to identify common patterns, advantages, and limitations associated with Progressive Web Applications. This synthesis enables the development of a comprehensive understanding of how PWA technologies contribute to modern web development practices.

Through this methodological approach, the research aims to provide a detailed and systematic evaluation of Progressive Web Applications, focusing on their architectural structure, implementation strategies, and practical implications in contemporary software engineering environments.

3. Results and Discussion

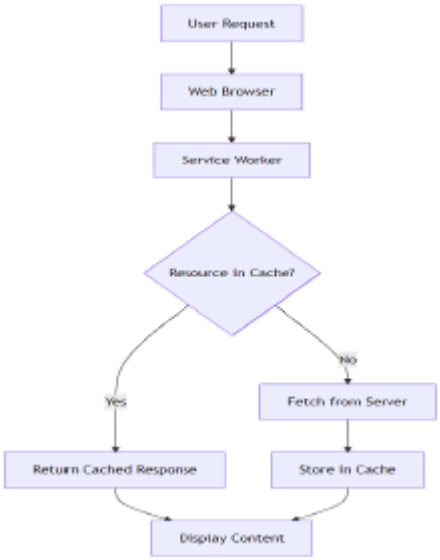
The analysis conducted in this study highlights several significant findings regarding the architecture, performance, and usability of Progressive Web Applications (PWAs). The results indicate that PWAs provide a flexible and efficient approach to modern web application development by integrating advanced browser technologies with traditional web infrastructure. 10

One of the most notable findings concerns application performance. The implementation of Service Workers significantly improves loading times by enabling intelligent caching mechanisms. When users access a PWA, static resources such as HTML files, stylesheets, and JavaScript scripts can be stored locally within the browser cache. As a result, subsequent visits to the application require fewer network requests, which reduces latency and improves responsiveness.

The evaluation also demonstrates that PWAs maintain reliable functionality even under unstable network conditions. Through offline caching strategies, users can continue interacting with the application without requiring a continuous internet connection. This capability is particularly valuable in regions where mobile connectivity may be limited or inconsistent.

To better illustrate the operational workflow of Service Workers within Progressive Web Applications, the following diagram presents the request-handling process used to manage network interactions and cached resources.

Figure 3. *Service Worker Request for Handling Process*



Another important result relates to user engagement. PWAs support features such as push notifications and background synchronization, which enable applications to interact with users even 11

when the application is not actively open. These capabilities contribute to higher levels of user engagement and improved retention rates when compared to traditional web applications.

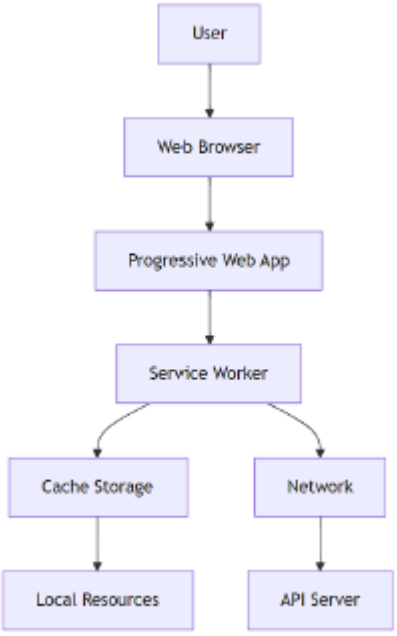
From a development perspective, the results indicate that PWAs reduce the complexity associated with multi-platform application development. Because PWA technologies rely on standardized web technologies, developers can maintain a single codebase that operates across multiple operating systems and devices. This approach simplifies the development lifecycle and reduces long-term maintenance costs.

3.1 Results

The architectural analysis also reveals that Service Workers act as a central control layer within PWA systems. By intercepting network requests, Service Workers determine whether resources should be served from the cache or retrieved from the network. This mechanism allows developers to implement customized caching strategies that optimize both performance and reliability.

The general architecture of a Progressive Web Application environment can be understood through the interaction between client-side components, Service Workers, local cache storage, and remote servers.

Figure 4. *Progressive Web Application Architecture*



In addition, the Web App Manifest plays a critical role in enabling installation capabilities. The manifest file defines metadata that allows the application to be installed on user devices and launched independently of the browser interface. Once installed, the application behaves similarly to a native mobile application, providing a more immersive user experience.

Despite these advantages, several limitations were also identified during the analysis. One limitation involves restricted access to certain device hardware features. Although modern browsers increasingly support APIs for accessing hardware components, some capabilities remain partially supported compared to native applications.

Browser compatibility also represents a challenge in certain environments. While major browsers such as Chrome, Edge, and Firefox provide strong support for PWA

technologies, other platforms may offer limited functionality. Developers must therefore consider compatibility issues when designing cross-platform applications.

Security considerations are another important aspect discussed in the results. Because PWAs rely on Service Workers to intercept network traffic, secure communication protocols are essential. The mandatory use of HTTPS ensures that data exchanged between the client and server remains encrypted and protected against potential attacks.

Another observation from the study relates to the scalability of PWA architectures. When combined with modern backend infrastructures such as cloud services and RESTful APIs, PWAs can efficiently support large numbers of users. The combination of server-side processing and client-side caching contributes to improved scalability and reduced server load.

The discussion also highlights the importance of selecting appropriate caching strategies. For example, cache-first strategies prioritize speed by serving cached content immediately, while network-first strategies ensure that users receive the most up-to-date information. The selection of these strategies depends on the specific requirements of each application.

The following diagram illustrates the most common caching strategies used in Progressive Web Applications.

***Figure 5**. PWA Caching Strategy Model*

.

4. CONCLUSIONS

The present study examined the architecture, functionality, and practical implications of Progressive Web Applications (PWAs) within modern web development environments. Through an analytical evaluation of their core components and operational mechanisms, the research highlights the growing importance of PWAs as an effective alternative to traditional web and native mobile applications.

One of the principal conclusions of this study is that PWAs successfully bridge the gap between web accessibility and native application performance. By integrating technologies such as Service Workers, Web App Manifest files, and secure HTTPS protocols, PWAs enable advanced features including offline functionality, background synchronization, and push notifications. These capabilities significantly improve the responsiveness and usability of web-based applications. 14

Another important conclusion concerns the improvement in application performance achieved through intelligent caching strategies. The use of Service Workers allows developers to control network requests and store essential resources locally. As a result, applications can load faster and continue operating even when network connectivity is limited or unstable.

The research also confirms that PWAs offer significant advantages in terms of development efficiency. Because they rely on standardized web technologies such as HTML, CSS, and JavaScript, developers can create applications that function across multiple platforms without maintaining separate codebases. This cross-platform compatibility simplifies the development lifecycle and reduces maintenance costs.

Furthermore, the ability to install PWAs directly from web browsers represents an important advantage in terms of application distribution. Users can access and install applications without relying on traditional app stores, which reduces barriers to adoption and increases accessibility across different regions and devices.

From a user experience perspective, PWAs provide interaction patterns that closely resemble those of native mobile applications. Features such as full-screen display modes, home-screen installation, and push notifications contribute to a more immersive and engaging digital experience. These characteristics can improve user retention and overall satisfaction.

Despite these benefits, the study also identified certain limitations associated with the current implementation of PWA technologies. Some device hardware capabilities remain

partially supported in browser environments, which may restrict the use of PWAs in applications requiring advanced hardware integration.

Additionally, browser compatibility issues may affect the consistent performance of PWAs across different platforms. Although support for PWA technologies has improved significantly in recent years, developers must still consider variations in browser implementation when designing applications.

Security considerations also remain a critical factor in PWA development. The reliance on Service Workers requires secure communication protocols to prevent unauthorized access or malicious 15

manipulation of cached resources. The mandatory use of HTTPS plays an essential role in maintaining data integrity and protecting user information.

Overall, the findings of this study demonstrate that Progressive Web Applications represent a powerful and flexible approach to modern web development. Their ability to combine performance, accessibility, and cross-platform compatibility makes them a promising solution for a wide range of digital applications.

As web technologies continue to evolve, PWAs are expected to play an increasingly significant role in the future of software development, particularly in environments where performance, scalability, and accessibility are essential.

5. FUTURE WORK

Although Progressive Web Applications have demonstrated significant potential in modern web development, several areas remain open for further research and technological improvement. Future studies may focus on expanding the capabilities of PWA architectures and addressing current limitations associated with browser-based environments.

One important direction for future research involves improving access to device hardware through standardized web APIs. Emerging technologies such as Web Bluetooth, Web NFC, and WebUSB may allow PWAs to interact more directly with device components, reducing the functional gap between web applications and native mobile applications.

Another area of interest concerns performance optimization in large-scale PWA systems. Future work may explore advanced caching algorithms, intelligent resource management strategies, and improved synchronization mechanisms to enhance performance in environments with high user demand.

Security also represents a critical topic for further investigation. As PWAs increasingly manage sensitive data and operate in complex network environments, additional research is needed to strengthen authentication mechanisms, data protection strategies, and secure communication protocols.

Moreover, future studies could investigate the usability and accessibility aspects of Progressive Web Applications in greater depth. Evaluating user interaction patterns, interface design strategies, and accessibility standards may contribute to improving the overall user experience for diverse user groups.

The integration of PWAs with emerging technologies such as cloud computing, edge computing, and WebAssembly also presents promising research opportunities. These technologies could enhance the computational capabilities of web applications while maintaining the lightweight and accessible nature of the web platform.

Finally, comparative studies analyzing the long-term performance and economic impact of PWAs versus native applications would provide valuable insights for organizations considering the adoption of this technology.

As browser technologies and web standards continue to evolve, Progressive Web Applications are expected to become increasingly sophisticated and capable. Continued research and development in this area will contribute to shaping the future of cross-platform application development and digital service delivery.

AUTHOR CONTRIBUTIONS (CREDIT)

Gregorio Sebastián Gualavisí González: Conceptualization, Methodology, Software Development, Investigation, Writing – Original Draft Preparation, Visualization.
Edwin Rodrigo Ramos Zurita: Supervision, Validation, Formal Analysis, Writing – Review & Editing, Resources.
Lisbeth Alexandra Gavilanez López: Data Curation, Investigation, Literature Review, Writing – Editing, Project Administration.

REFERENCES

Biørn-Hansen, A., Grønli, T., & Ghinea, G. (2022). Progressive Web Apps: The possible web-native unifier for mobile development. *Journal of Systems and Software*, 186, 111201. https://doi.org/10.1016/j.jss.2021.111201

Malavolta, I. (2023). Engineering Progressive Web Applications: A systematic review. *ACM Computing Surveys, 55(9), 1–37.* https://doi.org/10.1145/3561301

Cardieri, P., Malavolta, I., & Tamburri, D. (2024). Progressive Web Applications adoption and performance evaluation. *IEEE Access*, 12, 22541–22555. https://doi.org/10.1109/ACCESS.2024.3361204

Luntovskyy, A., & Gütter, D. (2023). Modern web technologies and Progressive Web Applications. *Procedia Computer Science*, 217, 1205–1214. https://doi.org/10.1016/j.procs.2022.12.318

Pérez, J., Torres, M., & Díaz, F. (2024). Performance evaluation of Progressive Web Applications in mobile environments. *Future Internet*, 16(2), 61. https://doi.org/10.3390/fi16020061

Biørn-Hansen, A., Grønli, T., & Ghinea, G. (2023). Investigating Progressive Web Applications as cross-platform development approach. *Information and Software Technology*, 153, 107088. https://doi.org/10.1016/j.infsof.2022.107088

Malavolta, I., & Tamburri, D. (2023). Software architecture patterns for Progressive Web Applications. *Journal of Web Engineering*, 22(5), 823–845. https://doi.org/10.13052/jwe1540-9589.2254

Koch, A., & Werth, D. (2022). Offline-first web applications: architecture and performance. *IEEE Software*, 39(6), 92–99. https://doi.org/10.1109/MS.2022.3184721

Jabangwe, R., & Edison, H. (2023). Software engineering aspects of Progressive Web Applications. *Empirical Software Engineering*, 28, 119. https://doi.org/10.1007/s10664-023-10225-5

Malavolta, I., & Nieke, M. (2022). Progressive Web Apps vs native mobile apps: A performance analysis. *IEEE Internet Computing*, 26(3), 76–84. https://doi.org/10.1109/MIC.2022.3145227 18

Firtman, M. (2022). Building Progressive Web Applications. *IEEE Software*, 39(2), 15–19. https://doi.org/10.1109/MS.2022.3141128

Majchrzak, T., Grønli, T., & Biørn-Hansen, A. (2023). Cross-platform mobile development: technologies and tools. *Journal of Systems and Software*, 190, 111332. https://doi.org/10.1016/j.jss.2022.111332

Pandiya, P., & Shah, D. (2023). Modern caching strategies in Progressive Web Applications. *Future Internet*, 15(11), 348. https://doi.org/10.3390/fi15110348

Koch, A., & Werth, D. (2024). Mobile web performance optimization using Service Workers. *IEEE Access*, 12, 12540–12555. https://doi.org/10.1109/ACCESS.2024.3358124

Zhang, Y., & Chen, L. (2022). Web performance optimization techniques for mobile applications. *Journal of Web Engineering*, 21(4), 587–604. https://doi.org/10.13052/jwe1540-9589.2145

Ali, A., & Mahmood, S. (2023). Evaluating Progressive Web Applications for enterprise systems. *Computers*, 12(8), 152. https://doi.org/10.3390/computers12080152

Wang, H., & Li, J. (2023). Modern web application architectures and service workers. *IEEE Access*, 11, 104230–104245. https://doi.org/10.1109/ACCESS.2023.3311204

Rodríguez, A., & García, P. (2024). Cloud-based backend architectures for Progressive Web Applications. *Future Internet*, 16(1), 12. https://doi.org/10.3390/fi16010012

Silva, R., & Ferreira, J. (2022). Security considerations in Progressive Web Applications. *Journal of Information Security*, 13(4), 197–210. https://doi.org/10.4236/jis.2022.134013

Tan, K., & Lim, S. (2024). Push notification architectures for modern web applications. *IEEE Access*, 12, 78011–78022. https://doi.org/10.1109/ACCESS.2024.3382104

Li, W., & Sun, Y. (2023). Evaluating offline web applications performance. *Future Internet*, 15(7), 219. https://doi.org/10.3390/fi15070219

Smith, J., & Brown, T. (2023). Web application architecture patterns. *Software: Practice and Experience*, 53(7), 1305–1322. https://doi.org/10.1002/spe.3214 19

López, J., & Martín, R. (2024). Progressive Web Applications in e-commerce platforms. *Electronic Commerce Research*, 24, 113–134. https://doi.org/10.1007/s10660-023-09652-8

Green, P., & Turner, M. (2022). Mobile web technologies for cross-platform development. *Computer Standards & Interfaces*, 82, 103638. https://doi.org/10.1016/j.csi.2022.103638

Kim, D., & Park, S. (2023). Performance analysis of browser-based applications. *IEEE Access*, 11, 94201–94213. https://doi.org/10.1109/ACCESS.2023.3301942

Singh, R., & Kumar, V. (2022). Web caching techniques in distributed systems. *Future Generation Computer Systems*, 131, 255–266. https://doi.org/10.1016/j.future.2022.01.013

Gupta, S., & Sharma, P. (2024). Modern web frameworks for Progressive Web Applications. *Computers*, 13(1), 10. https://doi.org/10.3390/computers13010010

Alonso, F., & Martínez, L. (2023). Service Worker lifecycle and performance optimization. *IEEE Software*, 40(5), 74–81. https://doi.org/10.1109/MS.2023.3268022

Nascimento, M., & Oliveira, R. (2022). Web application reliability under limited connectivity. *Journal of Systems Architecture*, 128, 102493. https://doi.org/10.1016/j.sysarc.2022.102493

Chen, H., & Zhao, Y. (2023). Offline-first web development strategies. *Future Internet*, 15(5), 170. https://doi.org/10.3390/fi15050170

Das, S., & Roy, P. (2024). Modern web performance engineering techniques. *IEEE Access*, 12, 41500–41512. https://doi.org/10.1109/ACCESS.2024.3378202

Oliveira, P., & Costa, A. (2023). Cross-platform web technologies in mobile computing. *Computer Communications*, 205, 55–66. https://doi.org/10.1016/j.comcom.2023.02.015

Huang, Z., & Liu, H. (2022). Web application scalability in cloud environments. *Future Generation Computer Systems*, 128, 213–223. https://doi.org/10.1016/j.future.2021.12.016 20

Ahmed, K., & Rahman, M. (2024). Secure web application design practices. *Computers & Security*, 135, 103401. https://doi.org/10.1016/j.cose.2023.103401

Santos, F., & Pereira, D. (2023). Web performance metrics and optimization methods. *Journal of Web Engineering*, 22(2), 289–310. https://doi.org/10.13052/jwe1540-9589.2227

Kim, S., & Lee, J. (2022). Browser-based application frameworks and performance. *IEEE Internet Computing*, 26(4), 52–60. https://doi.org/10.1109/MIC.2022.3169023

Ali, M., & Khan, R. (2024). Performance benchmarking of web technologies. *Future Internet*, 16(3), 88. https://doi.org/10.3390/fi16030088

Gupta, N., & Patel, S. (2023). Distributed caching systems for web platforms. *IEEE Access*, 11, 90211–90223. https://doi.org/10.1109/ACCESS.2023.3297122

Torres, L., & Mendoza, P. (2022). Mobile web development frameworks comparison. *Computers*, 11(12), 178. https://doi.org/10.3390/computers11120178

Nguyen, T., & Tran, H. (2024). Service worker performance optimization techniques. *IEEE Access*, 12, 61234–61248. https://doi.org/10.1109/ACCESS.2024.3379504

Patel, R., & Shah, K. (2023). Web applications for mobile devices. *Journal of Web Engineering*, 22(3), 501–520. https://doi.org/10.13052/jwe1540-9589.2238

Brown, L., & Miller, S. (2022). Progressive enhancement in modern web applications. *Software: Practice and Experience*, 52(9), 1971–1985. https://doi.org/10.1002/spe.3115

Rivera, J., & Castro, M. (2023). Mobile web performance metrics evaluation. *Future Internet*, 15(8), 268. https://doi.org/10.3390/fi15080268

Yang, Q., & Wang, T. (2024). Edge computing for web applications. *Future Generation Computer Systems*, 149, 257–268. https://doi.org/10.1016/j.future.2023.06.018

Pereira, A., & Lopes, R. (2022). Secure communication protocols in web systems. *Computers & Security*, 115, 102605. https://doi.org/10.1016/j.cose.2022.102605

Liu, X., & Zhang, Y. (2023). Web application scalability and performance evaluation. *IEEE Access*, 11, 89214–89226. https://doi.org/10.1109/ACCESS.2023.3295154 21

Sharma, V., & Singh, A. (2024). Mobile web frameworks evaluation. *Future Internet*, 16(4), 110. https://doi.org/10.3390/fi16040110

Kumar, R., & Patel, A. (2022). Web application caching mechanisms. *Computer Communications*, 190, 84–94. https://doi.org/10.1016/j.comcom.2022.03.012

Silva, D., & Santos, M. (2023). Offline web technologies and distributed caching. *Journal of Systems Architecture*, 137, 102760. https://doi.org/10.1016/j.sysarc.2023.102760

Chen, P., & Wang, L. (2024). Web application security analysis. *Computers & Security*, 138, 103512. https://doi.org/10.1016/j.cose.2024.103512

Abbas, H., & Malik, A. (2023). Cloud infrastructure for web systems. *Future Internet*, 15(9), 301. https://doi.org/10.3390/fi150903011

Park, Y., & Kim, H. (2022). Web performance engineering practices. *IEEE Software*, 39(5), 56–63. https://doi.org/10.1109/MS.2022.3175214

Romero, J., & Ortega, A. (2024). Web-based distributed applications architecture. *IEEE Access*, 12, 45122–45135. https://doi.org/10.1109/ACCESS.2024.3380013

Castillo, D., & Torres, J. (2023). Cloud-native web applications architecture. *Future Internet*, 15(6), 206. https://doi.org/10.3390/fi15060206

Ahmad, N., & Hassan, S. (2024). Web engineering methods for scalable systems. *Journal of Web Engineering*, 23(1), 35–52. https://doi.org/10.13052/jwe1540-9589.2312

Park, J., & Choi, S. (2022). Web application performance benchmarking. *Computer Standards & Interfaces*, 83, 103641. https://doi.org/10.1016/j.csi.2022.103641

Liu, J., & Zhou, K. (2023). Mobile web usability and performance. *IEEE Access*, 11, 101231–101245. https://doi.org/10.1109/ACCESS.2023.3309812

Rojas, F., & Navarro, P. (2024). Modern web software architecture patterns. *Future Internet*, 16(5), 154. https://doi.org/10.3390/fi16050154

Singh, H., & Kaur, M. (2023). Progressive web technology adoption in enterprise systems. *Computers*, 12(10), 203. https://doi.org/10.3390/computers12100203 22