NEW RESOURCES CONSULTING

1000 N. Water St
Milwaukee, WI
newresources.com

CASE STUDY

PROGRESSIVE WEB APPLICATIONS

INDUSTRY

NON-PROFIT

TECHNOLOGIES

PROGRESSIVE WEB APPLICATIONS, IONIC

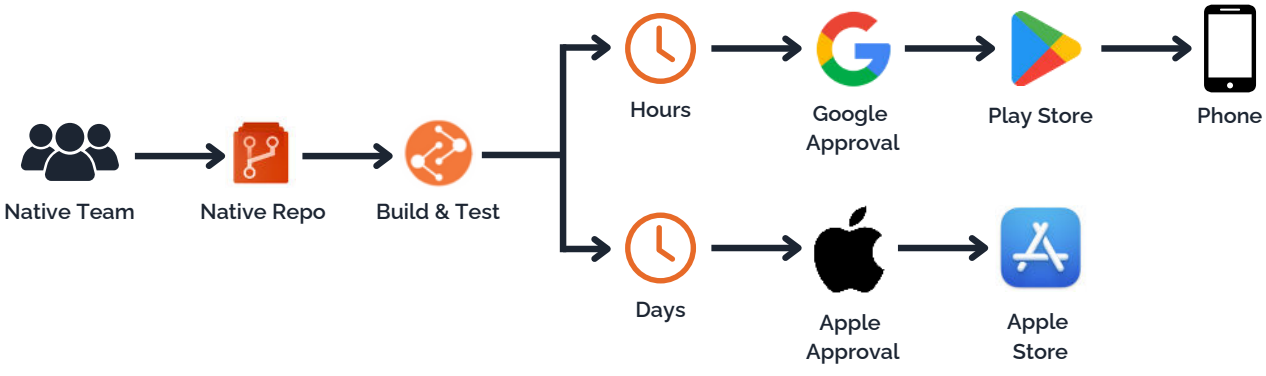
CHALLENGE

In the dynamic landscape of mobile solutions, every SaaS company seeks ways to enhance quality and improve engagement with their end-users. One such industry leader, operating within educational software solutions, embarked on a strategic initiative to improve their mobile app offerings.

Transitioning from reliance on third-party vendors to developing an in-house solution, this company sought to assert greater control over the quality and customization of its mobile applications. Central to this transformation was the selection of a robust mobile technology stack capable of meeting the diverse needs of their customer base across iOS and Android platforms.

After rigorous evaluation and market research, React Native emerged as the frontrunner due to its ability to streamline development with a single code base while maintaining native-like performance. However, this transition posed formidable challenges, including the acquisition of additional expertise in native app development and the management of deployment and maintenance for a multitude of individualized app instances catering to different educational tenants.

Native DevOps Flow



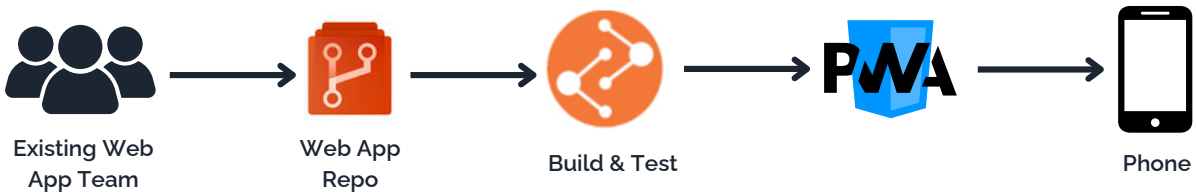
1000 N. Water St
Milwaukee, WI
newresources.com

SOLUTION

The company collaborated with New Resources Consulting (NRC) to spearhead the design and execution of the new mobile application. At first, the company requested resources to help create the mobile application and automate the hundreds of deployments. In line with their methodology, NRC initiated a series of requirement-gathering sessions to understand the project's objectives and constraints independent of the requested solution.

During this investigative phase, NRC considered several solutions for automation and native framework; however, after gaining a deeper understanding of the native features needed for the mobile solution, NRC identified the potential for a novel approach that could leverage the existing team's skill set and eliminate the need to automate hundreds of releases each month. This solution would leverage an established yet emerging technology called Progressive Web Applications (PWA).

PWA DevOps Flow



WHAT IS A PWA?

A Progressive Web Application (PWA) is a web application that utilizes modern web technologies to deliver an app-like experience to users. PWAs are designed to work across all devices and platforms, providing users with a seamless experience similar to native mobile applications.

Responsive Website v. Native App v. Progressive Web App

Capabilities	Responsive Website	Native App	PWA (Progressive Web App)
Mobile Friendly	✓	✓	✓
Installable	—	✓	✓
SEO Indexed	✓	—	✓
Offline Mode	—	✓	✓
Push Notification	—	✓	✓
GPS Enabled	—	✓	✓

1000 N. Water St
Milwaukee, WI
newresources.com

KEY FEATURES OF PWAS

1. Responsiveness: PWAs are built to adapt to different screen sizes and orientations, ensuring a consistent experience across devices.
2. Offline Functionality: PWAs can work offline or in low-network conditions by caching resources and data, enabling users to continue using the app without an internet connection.
3. App-like Interface: PWAs often incorporate native app-like features such as push notifications, full-screen mode, and access to device hardware like cameras and GPS.
4. Progressive Enhancement: PWAs are designed to work on any browser, with progressive enhancement ensuring that users with modern browsers get the best experience while still functioning on older browsers.
5. Installability: Users can add PWAs to their device's home screen, making them easily accessible with a single tap without downloading and installing from an app store.

Overall, PWAs offer the benefits of both web and native applications, providing users with fast, reliable, and engaging experiences while simplifying development and deployment for developers.

PWA + NATIVE CONTAINER

Even though the PWA can be installed through the browser on the home screen, most users have yet to become accustomed to interacting with PWAs in this manner. To bridge this gap between PWAs and user behavior, NRC elected to leverage a 3rd party tool created by Microsoft called PWA Builder. Beyond making PWAs easier to build, PWA Build has a feature that allows you to create a native container app for iOS and Android. This container app wraps the PWA around a native shell, which can be built and distributed to the app stores.

While this requires an initial time investment in publishing the native container app, all subsequent updates to the PWA can circumvent the lengthy iOS app approval process. Instead, content updates and bug fixes are deployed directly to the PWA hosted in AWS using the company's traditional DevOps pipeline. It is still possible that, in some cases, a native shell will need to be updated. However, those instances are limited to cases such as Apple or Google releasing new icon dimensions for support of greater and greater screen resolutions. Simply put, 98% of releases to the PWA will never require complete publication in the app store.

APPROACH

Starting with a Responsive SPA

The initial step involved crafting a responsive single-page application (SPA) utilizing the team's established technology stack, Vue.js, and content APIs powered by PHP. Given the framework-agnostic characteristics of Progressive Web Applications (PWAs), the team embarked on developing a new app within their familiar tech stack without the need to consider the device.

Developing Push Notifications

While the development of the Single-Page Application (SPA) was in full swing, another team member enhanced the app's functionality with push notifications. Rather than using traditional methods, we decided to integrate Firebase's Messaging SDK. This choice was driven by the need for a robust and scalable solution to manage our push notifications. Firebase's Messaging SDK streamlined the interaction with the browser's notification system, checking device opt-ins and delivering notifications efficiently. The SDK simplified our development process and ensured that our push notifications were reliable and capable of scaling as our user base grew.

Creating the Native Wrapper

After thoroughly testing and confirming that our PWA functioned flawlessly as a standalone application in the browser, the next step was to encapsulate it within a native shell.

This time, the team chose Ionic Capacitor over other tools to craft the native containers for both iOS and Android. Ionic Capacitor allowed us to integrate native device functionalities seamlessly, tapping into the full potential of the native platforms while still leveraging our existing web technologies.

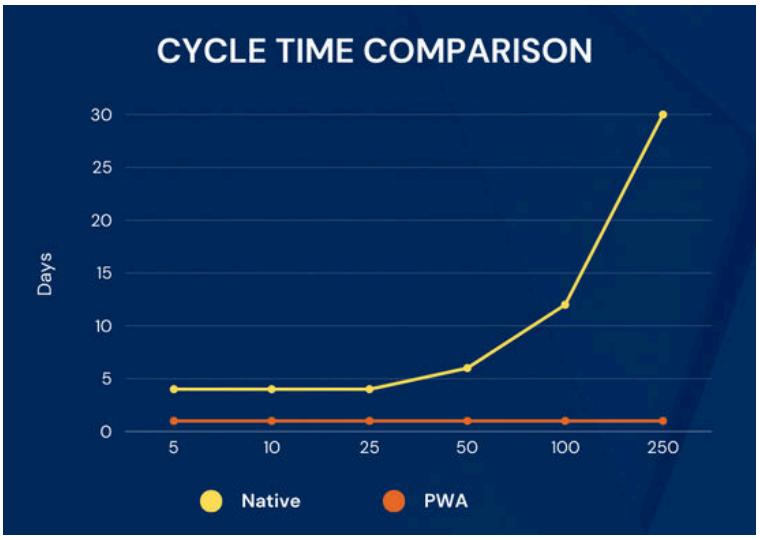
For the iOS version, Capacitor helped generate an Xcode project that we then built and deployed using our Apple Developer account. The process for Android was even smoother; Capacitor enabled the creation of a native application package directly from the PWA, making it ready for immediate distribution through the Google Play Store.

RESULTS

Now that we've outlined the approach let's delve into the results by examining two key metrics that highlight the advantages of PWA.

Cycle Time Comparison

Firstly, let's explore the concept of cycle time for change. This metric denotes the duration for a modification, whether a bug fix, feature enhancement, or any other update, to traverse all internal quality checks and reach the end user. With native applications, there is an inherent overhead associated with publishing updates to various app stores. This overhead significantly extends the time required to implement changes effectively. According to Apple's statistics, integrating changes into native apps typically adds two days to the existing cycle time within your DevOps process. Moreover, due to the unique requirement of individualized apps for each tenant or user segment, this cycle time further elongates as more tenants are added to the ecosystem.

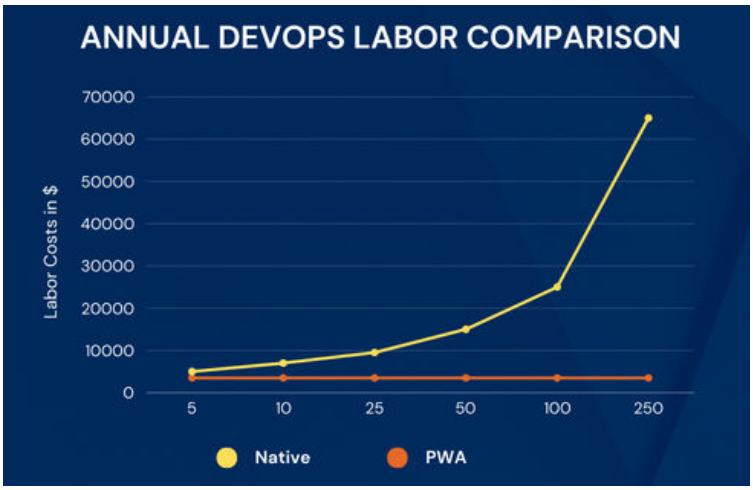


In the chart above, we accounted for a standard DevOps cycle of 1 day, with an additional cycle time of two days allocated for native applications. These extra two days were included in the standard cycle time because native applications necessitated internal DevOps controls and external approval processes. Additionally, we considered that a developer could handle up to 25 app submissions per cycle, considering the time required for building, testing, and submission. In a single-tenant distribution model (as with native apps), bottlenecks become apparent when dealing with more than two dozen apps. In contrast, for PWAs, the cycle time remained unaffected as the number of tenants increased. This is because a PWA is no different from a single-instance multi-tenant web application with a single DevOps pipeline and a single deployment artifact.

RESULTS CONTINUED

DevOps Labor Comparison

The second factor influencing the decision to adopt a PWA was the consideration of labor costs associated with application maintenance. In our cost analysis, we considered a monthly release cadence for each tenant and an additional half-hour of labor per tenant per month. This labor includes building, testing, and submitting each mobile app. It's worth noting that this analysis represented a best-case scenario; however, it still effectively demonstrates how costs scale out of control when managing hundreds of mobile application instances. When juxtaposed with a multi-tenant web application, the advantage of a PWA becomes evident, as there is no need to allocate additional labor for each additional tenant.



CONCLUSION

The transition from a white-label mobile application to a Progressive Web Application (PWA) afforded a strategic solution that enhanced quality control and directly addressed the challenges of scaling and maintenance.

By embracing PWAs, the company could efficiently deliver personalized white-label apps to each tenant without the overhead of managing multiple native applications. This approach optimized development and deployment and resulted in significant cost savings over traditional native apps.

The success of this implementation underscores the value of PWAs in modern mobile strategies. They offer fast, reliable, and engaging experiences to users while simplifying development and maintenance for businesses. This company's journey highlights the transformative potential of PWAs in addressing the evolving demands of mobile application development and underscores the importance of strategic partnerships in navigating technological transitions effectively.