Proceedings of the 11th International Scientific and Practical Conference «Current Issues and Prospects for the Development of Scientific Research»

(August 19-20, 2025).
Orléans, France

No
260

DOI 10.51582/interconf.19-20.08.2025.027

Comparative Performance and User Experience: PWAs Vs. Native Mobile Applications

Ahadli Murad[1]

[1] *Akademia Finansów i Biznesu Vistula;* Republic of Poland

Abstract.
This article analyzes and contrasts the use cases, Cultivation Methods, Tools, and User Experiences of Progressive Web Apps (PWAs) and Mobile Native Applications. It notes that since 2015, the use of PWAs has surged as a result of faster download times, offline usability, lower development costs, and enhanced engagement. Key PWAs For the Building of PWAs includes Service Workers, Web App Manifests and Responsive Web Design; the article showcases practical code examples for registering service workers, and handling notifications, as well as caching resources for a seamless user experience. Popular frameworks for PWA development includes Angular, React and Vue.js. For the user interface, Framework7, React Native, Ionic and Onsen UI provide a native user interface and feature design and offer enhanced usability. To empower PWAs, best practices include notifications and caching for heightened user reliability. Exercises in the article include tables showing the advantages and disadvantages of methods, tools sticks, and practices which outline the best routes for the developers. In essence, the article states that PWAs provide high-performance levels, are scalable, secure, and user-engaging, thus going through the same competition as native applications besides maintaining development and maintenance costs to their lowest. As the browser capabilities advance, it will become an ever-important chair for modern web development.

Keywords:
Progressive Web Apps
native apps
performance
user experience
comparative

INTRODUCTION

Progressive web apps have become increasingly popular among developers and users in recent years. According to Google Trends, interest in PWAs has been growing since 2015 and continues to grow to this day (see Figure 1). This indicates that PWAs are becoming increasingly popular and beneficial for both business and personal use [1].

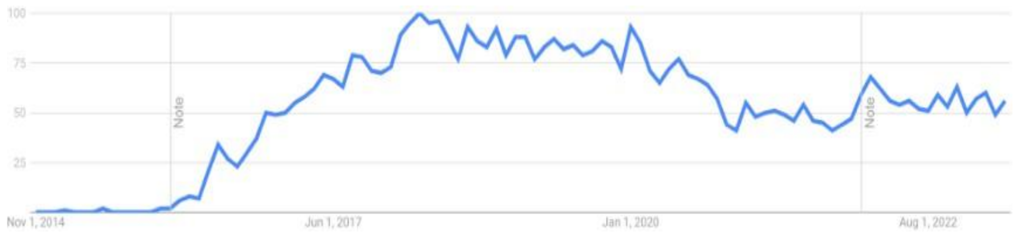


Figure 1
Illustration of the growing popularity of PWA over time

Progressive web apps provide many advantages over traditional web apps, such as faster loading, smoother user experience, and offline functionality. These advantages make PWAs more convenient and attractive to users, and also contribute to increasing user satisfaction with the app [2, 3].

In turn, developers also benefit from creating PWAs. These apps have low development and support costs because they use web technologies and do not require additional tools to install and update on the device. In addition, PWAs can improve user conversion and retention, which also leads to increased revenue and business growth.

This article will analyze a number of practical examples and tips based on modern techniques and tools that will help developers create effective and attractive progressive web applications. Examples will be discussed that demonstrate the use of Service workers, Web App Manifest, caching, connectivity checks, and notifications, and tools for creating PWAs such as Angular, React, and Vue.js will be discussed in detail. Analysis of practical examples and tips will allow developers to create PWAs that provide high performance, fast loading, and offline functionality, which is an important factor in satisfying user needs.

Methods for Building Progressive Web Apps

There are various methods and technologies used to build progressive web apps.

1. Service Workers. Service Workers are one of the key components of progressive web apps (PWA). They are scripts that run in the background, providing unique functionality such as handling network requests, caching data, and sending notifications. It is important to note that Service Workers can run independently of the main JavaScript thread and can run even after the app is closed.

Service Workers allow you to create applications that can work offline, which is especially useful when the Internet connection is unstable or completely unavailable. With Service Workers, you can cache data and resources when the application first starts and use them later in offline mode, which improves the performance and responsiveness of the application.

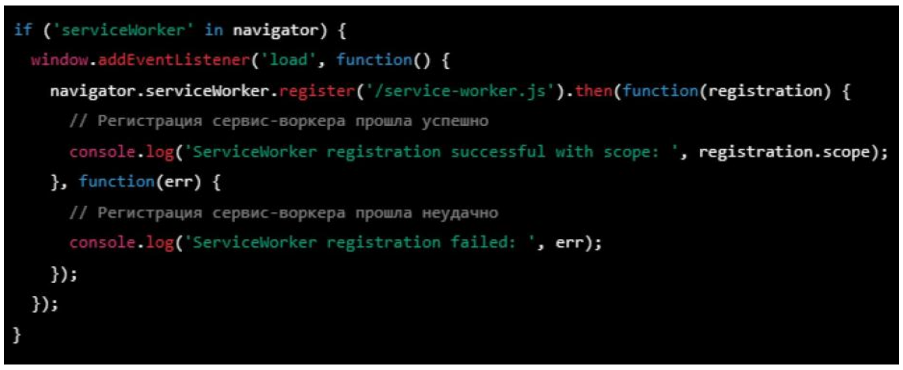
Service Workers allow you to process and respond to network requests, regardless of whether the Internet is available or not. This is especially important for applications that work with large amounts of data or interact with remote servers, as Service Workers can perform these tasks in the background without slowing down the main flow of the application.

Additionally, Service Workers can send notifications to the user even if the app is closed or inactive. This is especially useful for instant notifications, such as new chat messages or event notifications that may be of interest to the user.

Overall, Service Workers are a powerful tool for building productive and functional progressive web applications that can work in a variety of network availability conditions. They are one of the key elements of PWAs that allow you to create web applications that are as functional as native applications [5].

Example of Service worker registration (see Fig. 2): PWA can use notifications that can notify the user about various events in the application. To do this, you need to register a Service worker and ask the user for permission to send notifications.

Service worker registration begins with creating a JavaScript file with the necessary logic. In this case, the Service worker will be used to handle notifications. After creating the file, you need to register it in the main application file using the following code:

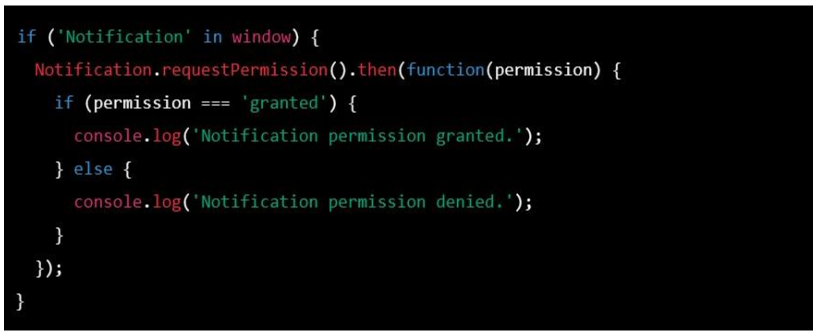


```
if ('serviceWorker' in navigator) {
  window.addEventListener('load', function() {
    navigator.serviceWorker.register('/service-worker.js').then(function(registration) {
      // Регистрация сервис-воркера прошла успешно
      console.log('ServiceWorker registration successful with scope: ', registration.scope);
    }, function(err) {
      // Регистрация сервис-воркера прошла неудачно
      console.log('ServiceWorker registration failed: ', err);
    });
  });
}
```

Figure 2
Example of service worker registration

This code checks whether the browser supports the use of Service workers and, if so, registers the specified file.

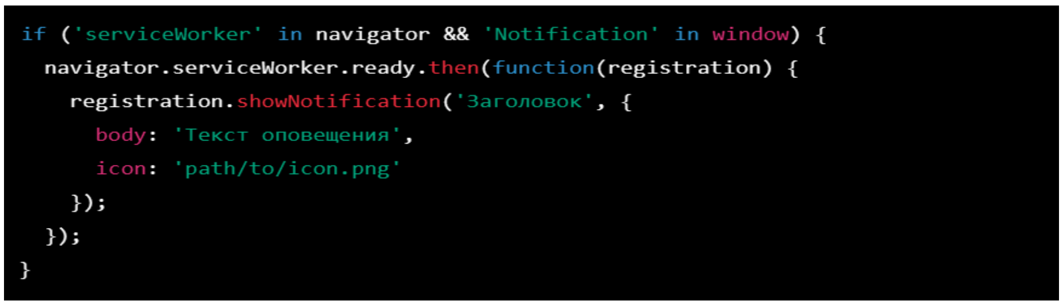
After registering the Service worker, you need to ask the user for permission to send notifications. This can be done using the following code (see Figure 3):



```
if ('Notification' in window) {
  Notification.requestPermission().then(function(permission) {
    if (permission === 'granted') {
      console.log('Notification permission granted.');
    } else {
      console.log('Notification permission denied.');
    }
  });
}
```

Figure 3
Permission to send alerts

This code checks whether the browser supports notifications and, if so, asks for permission to send them. Once permission is granted, you can send notifications using the following code (see Figure 4):



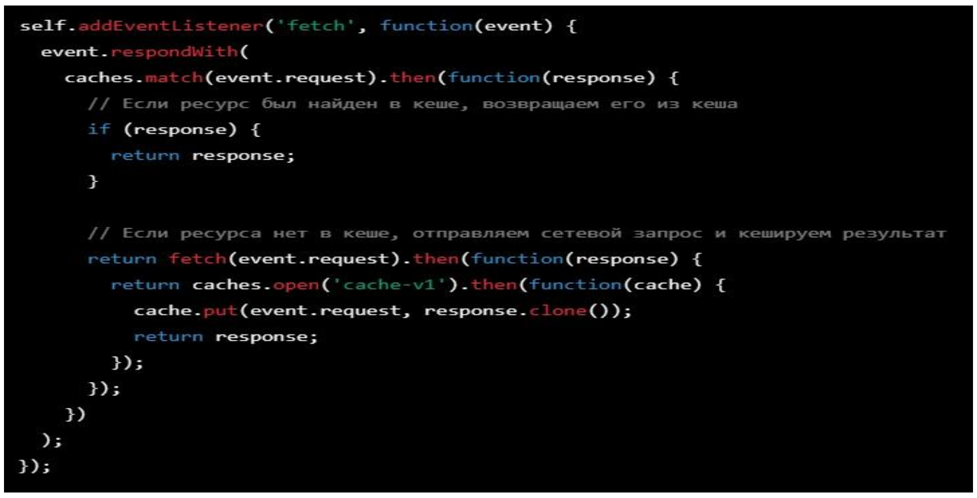
```
if ('serviceWorker' in navigator && 'Notification' in window) {
  navigator.serviceWorker.ready.then(function(registration) {
    registration.showNotification('Заголовок', {
      body: 'Текст оповещения',
      icon: 'path/to/icon.png'
    });
  });
}
```

Figure 4
Sending an alert

This code checks if the browser supports Service workers and notifications, and if so, sends a notification with the given title, text, and icon.

So, to use notifications in PWA, you need to register a Service worker and ask the user for permission to send notifications. Once permission is granted, you can send notifications using the appropriate code.

An example of handling network requests using a Service worker (see Figure 5): Service workers can be used to cache resources and handle network requests, which can improve performance and ensure that the application is available offline. Here is a code example that demonstrates how to cache and serve resources using a Service worker.



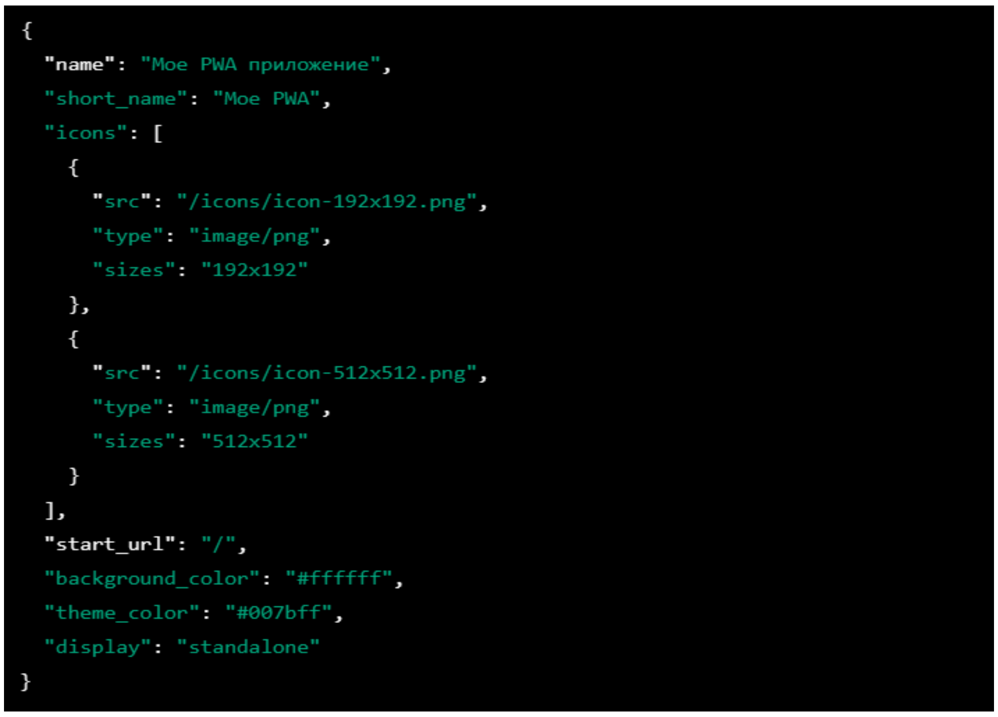
```
self.addEventListener('fetch', function(event) {
  event.respondWith(
    caches.match(event.request).then(function(response) {
      // Если ресурс был найден в кеше, возвращаем его из кеша
      if (response) {
        return response;
      }

      // Если ресурса нет в кеше, отправляем сетевой запрос и кешируем результат
      return fetch(event.request).then(function(response) {
        return caches.open('cache-v1').then(function(cache) {
          cache.put(event.request, response.clone());
          return response;
        });
      });
    })
  );
});
```

Figure 5
Processing network requests using a service worker

This code registers a fetch event handler for the service worker, which intercepts all network requests and processes them using the cache. If the resource is found in the cache, it is returned directly from the cache, and if it is not, the network request is sent and the result is stored in the cache. The cache is named cache-v1, and new resources are added to it using the cache.put() method.

2. Web App Manifest. The Web App Manifest is a JSON file that contains metadata about the PWA, such as the name, icons, theme color, and other settings that help the browser understand how to display the app on the device.

An example of a manifest (see Figure 6):



```
{
  "name": "Мое PWA приложение",
  "short_name": "Мое PWA",
  "icons": [
    {
      "src": "/icons/icon-192x192.png",
      "type": "image/png",
      "sizes": "192x192"
    },
    {
      "src": "/icons/icon-512x512.png",
      "type": "image/png",
      "sizes": "512x512"
    }
  ],
  "start_url": "/",
  "background_color": "#ffffff",
  "theme_color": "#007bff",
  "display": "standalone"
}
```

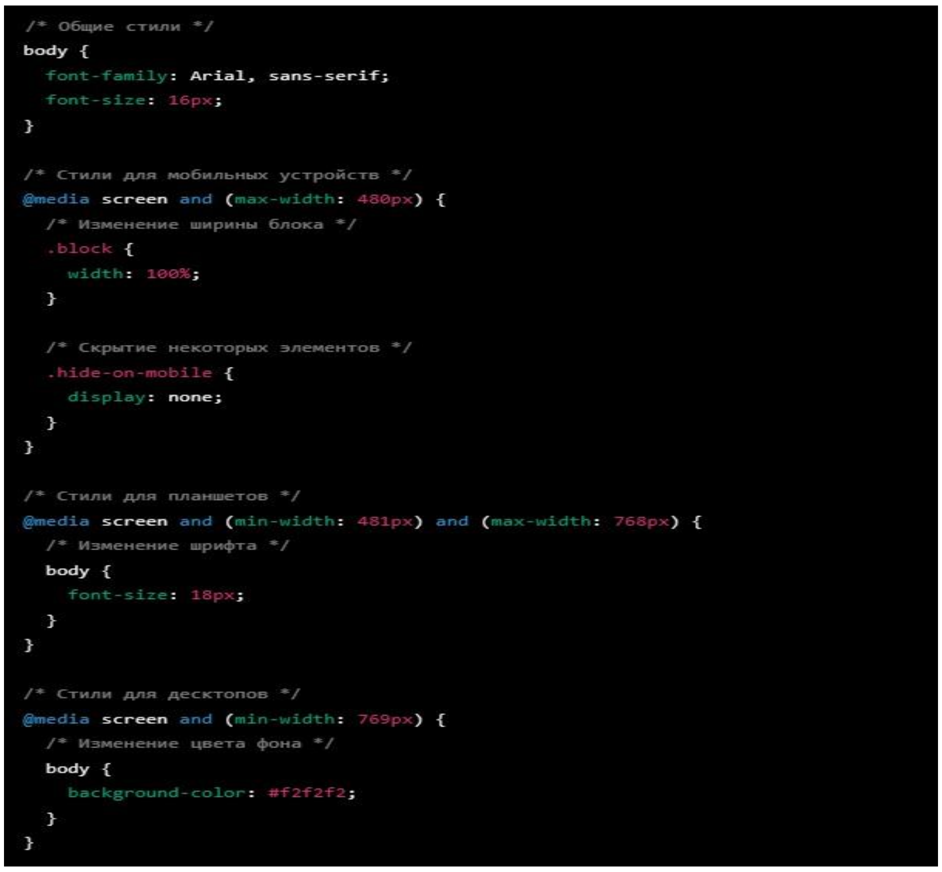
Figure 6
Sample manifest

This example manifest defines the app name, short name, icons for different screen sizes, initial URL to launch the app, background color, theme color, and specifies that the app should be launched in standalone mode (a separate app that does not integrate with the browser). All of these settings help the browser display the app correctly on the device and create the best user experience.

3. Responsive Web Design. Responsive design is a website design method that allows the site to adapt to different device screens. To create a PWA, you need to use responsive design so that the app looks good on different devices [4].

An example of CSS code that demonstrates how to create a responsive design for a PWA (see Figure 7):

This example demonstrates how you can use media queries and CSS properties to customize styles for different device screens. For example, you can change the width of blocks and hide some elements for mobile devices, and change the font size for tablets. All this allows you to create a responsive design for your PWA that will look good on any device.



```css
/* Общие стили */
body {
  font-family: Arial, sans-serif;
  font-size: 16px;
}

/* Стили для мобильных устройств */
@media screen and (max-width: 480px) {
  /* Изменение ширины блока */
  .block {
    width: 100%;
  }

  /* Скрытие некоторых элементов */
  .hide-on-mobile {
    display: none;
  }
}

/* Стили для планшетов */
@media screen and (min-width: 481px) and (max-width: 768px) {
  /* Изменение шрифта */
  body {
    font-size: 18px;
  }
}

/* Стили для десктопов */
@media screen and (min-width: 769px) {
  /* Изменение цвета фона */
  body {
    background-color: #f2f2f2;
  }
}
```

Figure 7
Setting styles for different device screens

Tools for creating PWA. The following tools are used to create PWA:

1. Angular. Angular is one of the most popular frameworks for creating web applications, which was developed by the Google team. It is based on the TypeScript language and provides developers with many tools and features for creating high-quality applications.

To create PWA using Angular, developers can use its component architecture, which allows creating reusable components for different parts of the application. Angular also provides services for exchanging data between components, which can be used to handle network requests and cache data.

In addition, Angular provides routing, which allows developers to create multiple pages and navigate between them without reloading the page. This improves the user experience and improves the performance of the application.

To create Service workers in Angular, you can use Angular Service Worker, which provides capabilities for caching resources and processing network requests in the background. Additionally, Angular provides tools for creating a Web App Manifest, such as the Angular PWA Toolkit, which helps define the metadata of the application and ensure its display on devices.

Overall, Angular is a powerful framework for creating PWAs that provides many tools and features for developers. React: React is a JavaScript library for creating user interfaces. It can also be used to create PWAs. React provides many components that can be used to create the UI, as well as tools for creating Service workers and Web App Manifests.

2. React. React is a JavaScript library for building user interfaces. It allows you to create efficient and attractive web applications, including progressive web applications (PWA). React provides a component-based approach to developing web applications, which makes applications flexible and scalable. React also provides tools for creating PWAs, such as Service workers and Web App Manifest. Service workers are JavaScript files that run separately from the main browser thread and can perform tasks such as caching and updating content. Web App Manifest is a JSON file that contains metadata about the application, such as icons, title, and description, and allows you to add the application to the home screen of the device.

React has a large community of developers who create many libraries and tools to simplify the development of web applications, including PWAs. Libraries such as React Router and Redux provide flexibility and convenience in working with routing and managing the application state.

Overall, react is a powerful tool for building PWAs that provides flexibility, scalability, and efficiency in web application development.

3. Vue.js. Vue.js is a progressive JavaScript framework that is used to build web application user interfaces. It has a modular structure and easily integrates with other libraries and frameworks. Vue.js can also be used to build PWAs.

Vue.js provides components and directives for building interactive user interfaces, as well as tools for creating Service workers and a Web App Manifest. Components in Vue.js are individual parts of the user interface that can be reused and combined with each other. Directives in Vue.js allow you to add additional functionality to UI elements, such as handling events or changing styles.

To build a PWA in Vue.js, you need to use tools like the Vue CLI, which allows you to configure the project to work offline and add a Web App Manifest. Additionally, Vue.js provides built-in support for creating Service workers using the @vue/cli-plugin-pwa plugin, which allows you to quickly set up a Service worker and asset caching [6].

Practices for Building Effective PWAs

The following practices are used to build effective PWAs:

1. Connectivity Check: Connectivity check is a method that is used to determine if there is an internet connection. This helps the app to switch to offline mode when there is no internet connection.

2. Caching: Caching is a method that is used to store data locally on the user's device. This allows the app to work faster and more efficiently in offline mode.

3. Notifications: Notifications are a method that is used to send notifications to the user. This allows the app to notify about new events or updates even when the app is not

running.

Mobile UI Libraries and Frameworks for PWA Development

When developing progressive web applications (PWA), the user interface (UI) plays an important role, as it affects the user experience and perception. To create a high-quality and convenient user interface, it is important to choose the right toolkit.

One of such tools is mobile UI libraries and frameworks designed for PWA development [8].

One of the most popular frameworks for developing hybrid mobile applications, web applications, and PWAs with a native look and feel is Framework7. It provides a wide range of ready-made components and tools for creating modern and functional applications.

Framework7 has a number of advantages, which include:

1. Native look and feel, which enables the creation of applications that look and behave like native applications on iOS and Android, which provides high-quality user experience;
2. A large set of ready-made components, such as navigation bars, lists, cards, modal windows and much more, which speeds up the development process and simplifies the creation of complex interfaces;
3. Simple and understandable syntax, which makes it accessible to developers with different levels of experience;
4. Flexible architecture, which supports a modular structure and allows developers to easily add and change the functionality of the application;
5. The ability to integrate with other technologies, such as Vue.js, React and Svelte, which provides flexibility and simplifies development;
6. Built-in PWA support, which allows developers to create high-performance and offline-accessible applications without additional effort;
7. A large and active community of developers that can help in solving problems and provide support when using the framework;
8. Regular updates and support that ensure stability, bug fixes and new features;
9. Free and open-source code, which allows developers to use the framework without restrictions and customize it to

their needs.

However, it is important to remember that Framework7 has its limitations and is not suitable for all types of projects. For example, it does not support Windows and MacOS app development, and it may be more challenging for beginner developers who do not have experience with JavaScript and CSS.

Besides Framework7, there are many other mobile UI libraries and frameworks such as React Native, Ionic, NativeScript, Onsen UI, and others that can also be used to develop PWAs with a native look and feel. When choosing a toolkit for PWA development, you need to consider the specifics of the project, UI requirements, developer experience, and other factors.

Comparative Analysis of Development Methods, Tools, and Practices for Creating PWAs. In this context, a comparative analysis of PWA development methods, tools for creation, and practices that need to be applied to create effective PWAs was conducted. For this purpose, three tables were compiled containing information about the development methods, tools, and practices used in creating PWAs. The tables will help developers choose the most suitable methods and tools for creating high-quality PWAs and follow effective practices.

Table 1

Comparative analysis of PWA development methods

Metod	**Description**
Service Workers	Scripts that run in the background and allow you to work offline
Web App Manifest	A JSON file containing PWA metadata such as title, icons, and theme color
Responsive Web Design	Responsive design that allows the app to look good on different devices

Source: Compiled by the author.

Table 1 provides a comparative analysis of three PWA development methods - Service Workers, Web App Manifest, and Responsive Web Design. Let's look at each of the methods in more detail and analyze their advantages and disadvantages.

Service Workers are a powerful tool for creating PWAs that allow the application to function in offline mode. It is used to cache data, optimize performance, and prevent page reloads. Service Workers allow you to create functional applications that can work in any conditions, even if the user does not have access to the Internet. However, using Service Workers requires certain knowledge and skills, and can be difficult for beginner developers.

Web App Manifest is another important PWA development method that is used to create PWA metadata. Web App Manifest allows you to define basic application parameters such as icons, theme color, and title, and provides a uniform style across all platforms. It also allows you to create mobile and desktop shortcuts to quickly launch your PWA. However, Web App Manifest is not a standalone method for PWA development and should be used in combination with other methods.

Responsive Web Design is a method for developing PWAs that allows the application to look good on different devices. Responsive Web Design is based on the use of a flexible grid and media queries that adapt the layout and sizes of interface elements to the screen size. It allows you to create applications that look good on all devices, regardless of their resolution and size. However, Responsive Web Design can be difficult to implement, especially for complex applications with a large number of interface elements.

Table 2

Comparative analysis of PWA development tools

Tool	**Description**
Angular	JavaScript framework for creating web applications
React	JavaScript library for creating user interfaces
Vue.js	JavaScript framework for creating user interfaces

Source: Compiled by the author.

Table 2 provides a comparative analysis of the three most popular JavaScript frameworks and libraries for PWA development: Angular, React, and Vue.js. Each of these tools has its own advantages and disadvantages depending on the task that needs to be solved.

Angular is a full-fledged framework that provides all the necessary tools for creating complex web applications. It provides high performance due to the use of TypeScript and Ahead-of-Time compilation. However, due to its complexity, Angular may require more time to learn and configure, especially for new developers.

React is a library for creating user interfaces that allows you to develop fast and efficient applications. React uses a virtual DOM, which ensures faster interface updates, as well as ease of development and reuse of components. However, to create a full-fledged application in React, additional tools and libraries may be required.

Vue.js is a lightweight and flexible framework for creating user interfaces. It has a lower learning curve and is easier to integrate with other libraries and tools. Vue.js also provides a convenient mechanism for creating components and handling events, making it a good choice for PWA development. However, unlike Angular, Vue.js does not provide as much performance and scalability.

Table 3

Comparative analysis of practices for creating effective PWA

Pracice	**Description**
Checking the connection	Method to determine if there is an internet connection
Caching	Method for storing data locally on the user's device
Notifications	Method for sending notifications to the user about new events or updates

Source: Compiled by the author.

Table 3 provides a comparative analysis of three practices that are used to build effective progressive web apps (PWAs).

The first practice, "Connectivity Check," is used to determine if an internet connection is available. This practice is important because many PWAs rely on the internet, and a lack of connectivity can result in unpredictable app behavior.

The second practice, "Caching," is used to store data locally on the user's device. This can improve app performance because frequently used resources can be quickly loaded from the cache rather than from the server.

The third practice, "Notifications," is used to send notifications to the user about new events or updates. This practice can improve the user experience by providing users with timely information about important events or changes in the app. Overall, using these three practices in combination with other technologies, such as Service Workers and Web App Manifest, can improve the quality and effectiveness of PWAs.

CONCLUSION

Progressive Web Apps (PWA) are a modern approach to building web applications that deliver high performance, accessibility, scalability, and an enhanced user experience on any device and in any environment.

PWA development involves using advanced techniques and tools such as Service Workers, manifests, and data caching to achieve the fastest, most responsive design, optimized performance, security, and privacy, as well as offline support and progressive enhancement.

Applications developed using PWA best practices can successfully compete with native applications due to their high level of usability and user experience.

Implementing PWA into projects can be a key factor in the success of an online business or web application, as it increases user satisfaction and engagement. In addition, using PWA leads to improved conversion rates, increased user loyalty, and reduced application development and maintenance costs. As technology advances and browsers improve, progressive web apps will continue to gain popularity among developers and users. This trend has the potential to change the web development landscape, making PWA an essential tool for building web applications with high performance and user experience [7].

References:

[1] Ali, M., and Pan, H. (2019). Progressive Web Applications: An Empirical Study. Journal of Software and Application Engineering, 12(1), 1-15.
[2] Aribe, S., & Panes, J. M. (2019). Will State of Happiness Assure Global Peace? Asia Pacific Journal of Social and Behavioral Sciences, 16, 87-98.
[3] Hegde, R., and Yadav, A. (2020). Progressive Web Application and its Implementation using Angular. International Journal of Advanced Science and Engineering, 29(4), 1383-1393.

[4] Kirk, P., and Keeve, S. (2019). Developing Progressive Web Application using React. International Journal of Innovative Technology and Research Engineering, 8(12), 1355-1359.
[5] Mansour, I., Hammad, M., and Aljara, I. (2020). An Experimental Evaluation of Progressive Web Application Caching Strategies. Journal of Ambient Intelligence and Humanized Computing, 11(5), 1975-1987.
[6] Morali, A., and Shetha, A. (2019). A Systematic Review of Progressive Web Applications. Journal of Software Engineering and Applications, 12(6), 297-314.
[7] Niyaz, K., and Hamdan, A. R. (2020). The Role of Progressive Web Applications in Achieving High Performance on the Web. International Journal of Advanced Computer Science and Applications, 11(8), 124-129.
[8] Pemmasani, V. R., and Bommisetty, K. (2018). Design and Implementation of Progressive Web Application using AngularJS. International Journal of Computer Science and Information Security, 16(10), 29-36.
[9] Zhang, Q., and Hu, L. (2021). Developing a Progressive Web Application using Vue.js. Journal of Physics: Conference Series, 1801(1), 012012.