White paper

When should you consider using a progressive web app?

May 2022

When should you consider using a progressive web app?

What is a PWA?

A PWA is the extension of a standard web app with functions that are actually reserved for native mobile applications. A PWA should feel like a native app; however, it actually runs in a web browser.

What properties does a PWA have?

Progressive:	Works on all browsers
Adjustable:	Adjusts to all screen types: Desktop, mobile, tablet…
Network-independent:	Can be used offline
Similar to an app:	Behaves just like a mobile app when it comes to interactions, navigation, user interface, etc.
Installable:	Can be installed on the start screen, just like a standard app.

How is a PWA structured?

From a technological perspective, a PWA has the same structure as a standard web app. That means standard technologies such as JavaScript, HTML and CSS as the basis and the frameworks built upon those such as React JS, Angular JS, Vue JS. In addition, PWA-specific configurations are detailed in what is known as a manifest:

- An app icon that appears on the start screen
- How the app is displayed e.g.: in a separate window, with or without a browser UI etc.
- Colour scheme

The following app-specific functions can be used via the PWA:

- The type of data, for how long, offline availability, details on when it is updated
- Local data storage can be used
- A PWA can be published in the Android App Store
- Device-specific functionalities such as geo-localisation, contacts, etc. (see section "Device functionalities")
- Display and control outside the app, e.g. in a widget or as a Siri suggestion

Elegant middle ground between web and native

People nowadays use their smartphones more than any other device to call up websites. This trend is increasing. If you already have a website, you can usually make it available offline, generally smartphone-friendly and installable in just a few steps. This is made possible by a PWA with a little effort and a responsive UI.

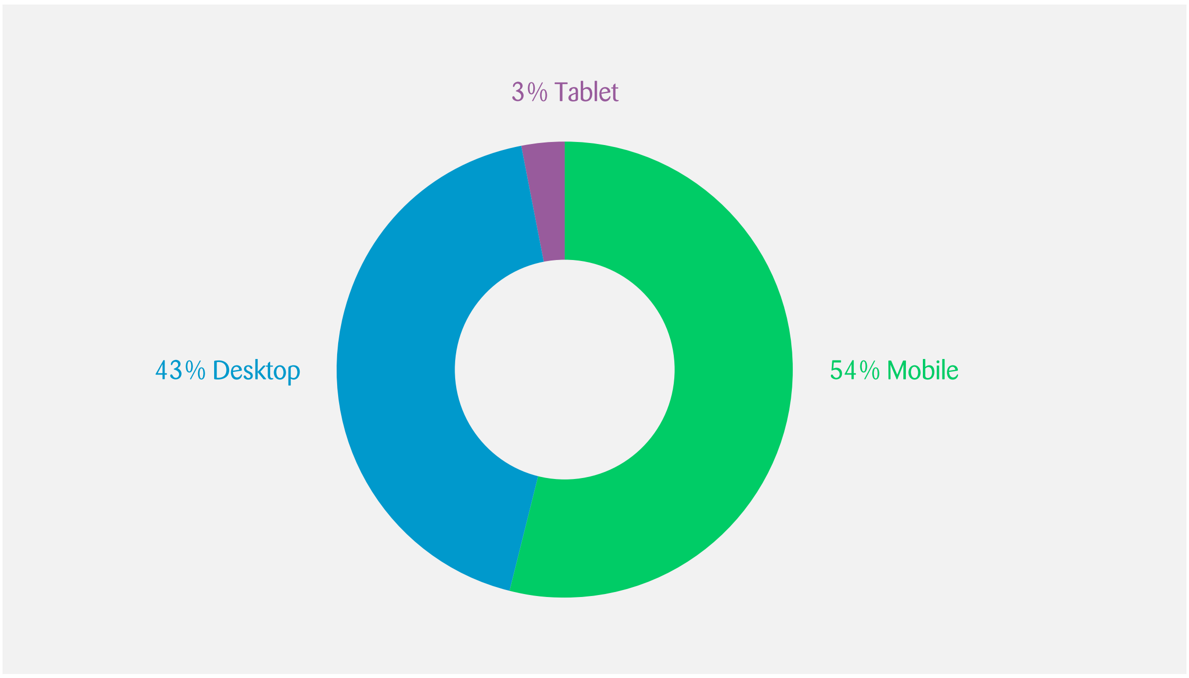


Figure 1: Desktop vs mobile vs tablet market share: https://gs.statcounter.com/platform-market-share/desktop-mobile-tablet

By contrast, a native app would mean much more time spent on development and would need to undergo an approval process for publication and subsequent placement in a store such as Google Play or Apple Store. Depending on the functionality and requirements of the application, this time and effort can be avoided by using a PWA.

A PWA represents a good middle ground between web and native. Alongside the various benefits, however, there are some restrictions and limitations where a PWA is concerned. We would like to highlight these in this article and also offer some advice on when you might consider using a PWA and when it's best to avoid them.

Benefits & features

Development starts faster

There is little difference between the development of a PWA and that of normal web apps. For many popular frameworks such as Angular, ReactJS or Vuejs, there are already templates available for a PWA that make it very easy to start the app. But you don't even need to go looking for these. All a PWA needs is a manifest, a service worker and the app's home page. There is no need for a complicated pipeline with MacOS and you don't have to manage any accounts with Apple or Google.

Better value for simple apps

For simpler apps, the effort involved in developing a normal app is not worth it, be it with cross-platform technologies such as ReactNative or Flutter. A PWA, by contrast, can be kept very lean. Due to the low requirements, a lean app can be rolled out and feedback can be collected in next to no time.

"Live" immediately

Rolling out a PWA is very similar to rolling out a normal website. Once the artefacts of the app are rolled out, each browser will download the latest version and automatically install it on the user's device. No approval from Apple or Google is necessary.

Website and app combined

The developed PWA can also show off most of its advantages in website form. For example, all static data is temporarily stored in the user's browser so that only dynamic data is loaded. A PWA can therefore also display the same website on the desktop. Reactive design allows you to display the same content, albeit with a different arrangement, on a large, wide screen. Furthermore, you can also convert a modern single-page application into a PWA. As a result, the maintenance effort is much lower compared to a normal mobile app.

Access to device features

A PWA can access some mobile-specific functionalities, the best example being the exact location of the device. Notifications can also be sent to the device, with the exception of iPhones or iPads. Another feature of a PWA is that it can be registered as a payment handler. This means that a user can use a PWA to pay for things, similar to ApplePay or GooglePay. Once a dealer accepts the PWA's payment method, transactions can take place between users and dealers. The PWA can of course also perform background updates, for example, in order to save user data on the device so that the user can use the PWA completely offline.

Drawbacks & impediments

Integration for devices and browsers

There are now multiple different browsers available running on numerous different devices. Of course, not all devices have the same functionalities. On an iOS device, for example, push notifications do not work from a simple PWA. Different browser versions, too, can be limiting. For example, the payment handler does not yet work on all versions of Firefox. The limitations of the various browser versions and devices must be considered during development. A native app is controlled by an app store and can only be installed for certain devices. By contrast, a PWA can be opened from any web browser running on any device.

Installing a PWA

Apps are usually installed from one of the well-known app stores (Google Play or iOS App Store). PWAs are not usually installed via an app store. They are installed using a menu button in the browser with the option “Add to start screen”. Many users are unaware that this option exists.

A custom installation user flow can be added so that users understand what installation options are available to them. This option is only available on Android.

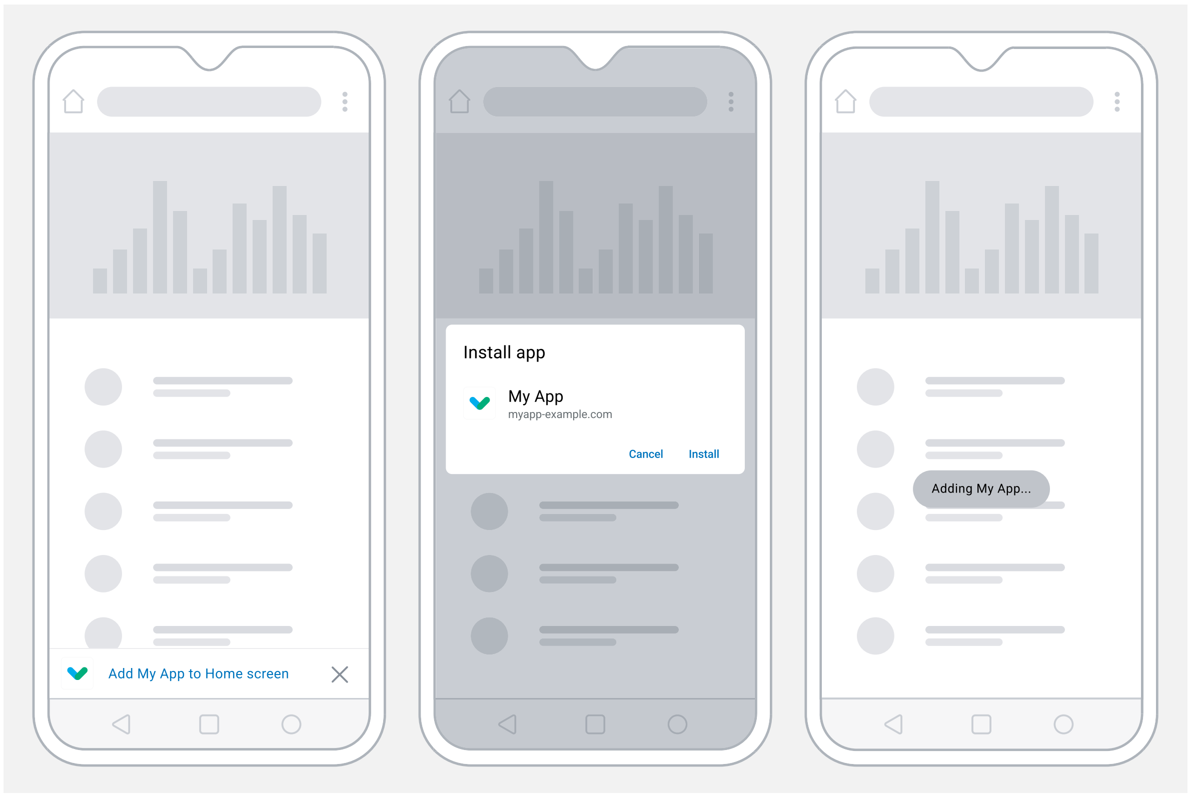
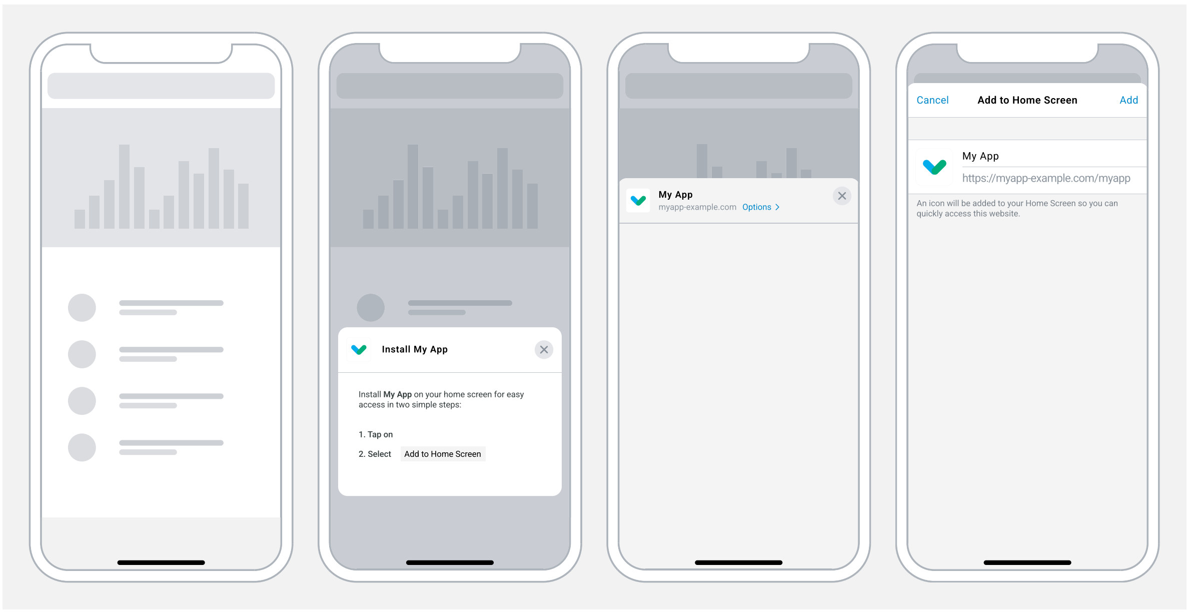


Figure 2: Android with a direct link to the installation window

A link that calls up the installation window directly cannot be created on iOS. Installation can only be carried out via the browser menu. You can use graphic instructions to explain to iOS users how they can simply install the app on the main screen of their device.



Install **My App** on your home screen for easy access in two simple steps:

1. Tap on
2. Select Add to Home Screen

An icon will be added to your Home Screen so you can quickly access this website.

Figure 3: iOS with installation guide and additional steps

The PWA can also be published in Google Play if desired. This option is not available for iOS.

Device functionalities

A PWA is heavily dependent on the browser and its capacities. Certain device-internal functionalities are not available in a browser and are therefore not available in a PWA either, e.g. face/fingerprint ID, altimeter sensor. Push notifications, Bluetooth and access to battery information are not yet available on iOS either.

	iOS	Android	Desktop
Camera recording	✓	✓	✓
Geo-localisation	✓	✓	✓
Push notifications	✕	✓	✓
Single sign-on	✓	✓	✓
Access to contacts	✕	✓	✕
Bluetooth	✕	✓	✓
Payment	✓	✓	✓
Sensor technology	✓	✓	✕
Network information	✕	✓	✓
Widgets	✕	✕	–
App clips, instant apps	✕	✕	–
Smartwatch	✕	✕	–

Testing

When it comes to testing, in principle you can proceed as you normally would in a browser application developed with standard web technologies. The various browser versions on mobile devices must also be considered for PWAs. In addition, you should not underestimate reactive design tests for devices with different screen sizes. Certain browsers such as Google Chrome offer excellent tools for analysing and testing PWA-specific technologies, e.g. data caching.

Migrating to a PWA

(Existing web platform for PWA)

Short loading times

Short loading times are a fundamental property of every PWA. The quality of the user's internet connection should play a minimal role when loading the web app. Data can of course be loaded from a back end. Business logic or the framework of the front end can be installed in the browser cache. In conclusion, all static data such as images, fonts and front-end code should be cached in the browser and only downloaded to the device when a new version is deployed.

Responsiveness

Responsiveness is another feature of a good web app. A PWA should feel “app-like” and must offer a good user experience on all platforms including your smartphone, tablet and desktop. The web app should respond quickly to user input and give feedback on what’s going on. A responsive web app has been proven to promote user engagement with the app and as a result increase satisfaction with the product.

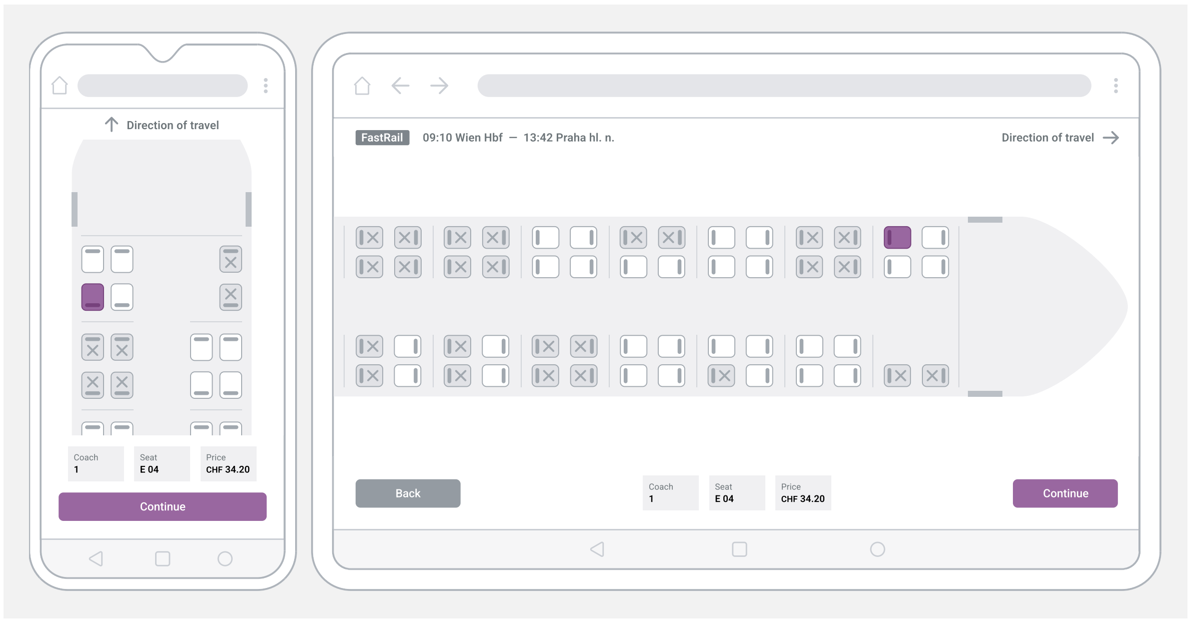
Offline-enabled

A PWA should also work regardless of the quality of the internet connection. As mentioned earlier, a PWA offers good cache mechanisms. Furthermore, in some use cases it can be a good idea to cache more than just static data. More and more users are using their mobile devices for most if not all such interactions. The PWA must also be able handle scenarios in which the internet connection is poor or absent. For a web app displaying flight tickets, for example, it is especially important that the ticket can also be viewed in offline mode.

Enhancements that help convert the web app into a PWA are available for every modern JavaScript framework. These help make the above points feasible, for example, in order to cache static data while loading dynamic data (user-specific data) from the back end.

Experiences with PWA

App for reserving seats – mobile, tablet and browser app in one



As part of a market research project, the goal was to develop a seat reservation app that would allow train passengers to reserve a seat with additional services in advance. We opted for a PWA in this project.

We wanted an app that could be developed quickly with just a few developers. A PWA mainly uses web front-end technologies with no native Android or iOS knowledge required. The app requirements were straightforward and only a few device-specific functions were necessary. The application also needed to be accessible on the desktop via the chosen browser. For these reasons, a PWA was seen as the ideal fit for the given requirements.

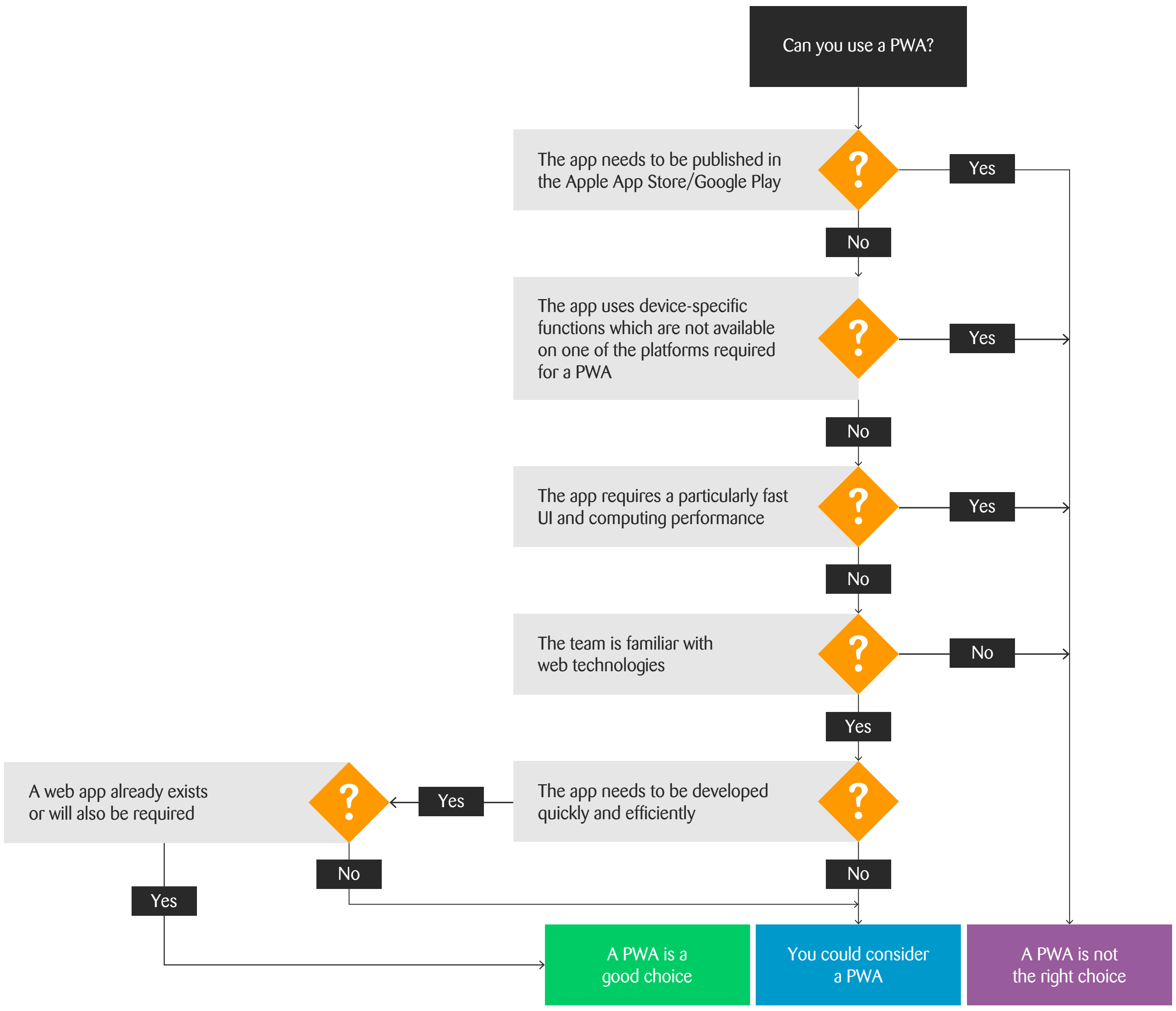
With a mobile-first approach, it is important to be able to load the app as quickly as possible, even when the network connection is poor. We opted for the JavaScript framework VueJS which can be kept very small. The PWA met all of the requirements and its development phase was short.

App for booking appointments

A prototype was built for an insurance company in order to determine the potential of an app to be used by customers for booking and managing appointments. This prototype was used within the customer's company in order to carry out usability tests and demonstrations. A PWA was chosen to ensure a quick and straightforward project launch. Even though the team of developers already had experience in developing apps, a PWA was seen as the more affordable and, more importantly, simpler alternative. The purpose of the application was to enable users to book an appointment with a service provider in their vicinity. With a PWA, they could simply request the device location and display any appointments booked. A mobile-first approach was used in the development phase, meaning that the application was also desktop-ready. It was also important to be able to view the appointments in the app in cases where the internet connection was poor or absent.

PWA guide

The following diagram can be used as an aid for deciding whether a PWA should be considered:



Can you use a PWA?
The app needs to be published in the Apple App Store/Google Play
Yes
No
The app uses device-specific functions which are not available on one of the platforms required for a PWA
Yes
No
The app requires a particularly fast UI and computing performance
Yes
No
The team is familiar with web technologies
No
Yes
The app needs to be developed quickly and efficiently
Yes
No
A web app already exists or will also be required
Yes
No
A PWA is a good choice
You could consider a PWA
A PWA is not the right choice

Contact

Doris Rogger
Head of Mobile
doris.rogger@zuehlke.com
+41 31 561 39 35