Progressive Web Apps (PWAs)

Michael Hebesberger
Diego Jacobs Tercero
Christoph Koch
David Mischak

Institute of Interactive Systems and Data Science (ISDS),
Graz University of Technology
A-8010 Graz, Austria

14 Dec 2020

Abstract

Progressive Web Apps (PWAs) have been talked about for a few years now amongst web developers, but outside of that circle few have heard or know of it even though they might have interacted with such an app or website. First the definition of a Progressive Web App is described. Then the current technological support of PWAs in the industry leading web-browsers and operating systems is compared. Following that the main characteristics and architecture on which PWAs are based are explained. Another important aspect is the user experience which PWAs provide. Of course, before anyone can start using a PWA it has to be available on the web or in an app store first. The options to achieve this are outlined in this survey, as well as the tools which can be used to create a PWA in the first place or to test an existing website on its adherence to the definition of a PWA.

Contents

List of Figures

List of Tables

List of Listings

Chapter 1

Introduction

A progressive web app (PWA) is a web app with the look and feel of a native mobile app. With it there is the ability to reach any person that has a web browser installed, no matter what device he has. PWAs can be distributed through an URL, which means there is the need to host the PWA in an own server, or upload it to an app store. People say that native apps can have a richer functionality since they are closer to the hardware. So a PWA brings the advantages of mobile apps and web apps. Since it has many of the capabilities of a mobile app that are going to be explain later on and there is also the ability to reach same amount of people as a web app, since it can be self hosted [Richard and LePage 2020]. You can see the comparison in figure 1.1.

1.1 Advantages of Using PWAs

Having a PWA gives many advantages if there are still some doubts on why should a website offer the option to install it as a PWA. First there will not be the need to pay for hosting it in the app stores as it was mentioned before and it is going to be explain with more detail later on, there is the option to host a PWA in an own domain. Another advantage is that it is cheaper to maintain and build PWAs since there will be only one code base for all platforms. They will be responsive to any screen. PWAs are supported for almost all modern browsers. More details about support coming later. Last but not least PWAs can be added to the home screen (A2HS), offer an offline experience, and have push notification like in a mobile app [Toonen 2020].

1.2 Examples of PWAs

There are some sites that already have the option to install them as a PWA, so there is the option to experience how PWAs work [Dom 2020]. For example:

- Financial Times as we can see in figure 1.2.
- Twitter
- Starbucks
- Uber
- Pinterest
- Spotify

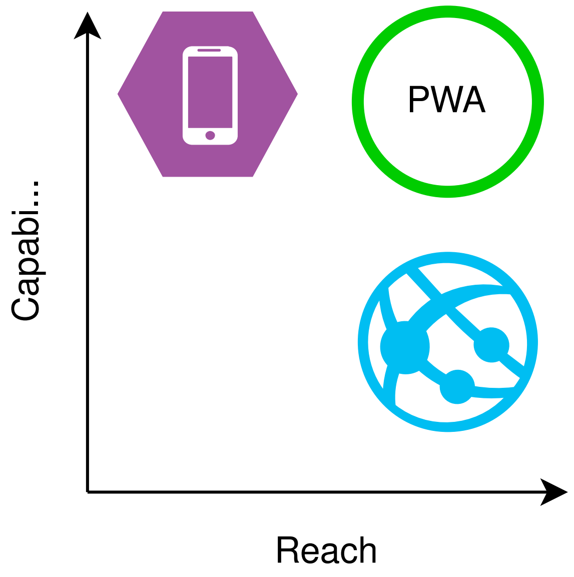
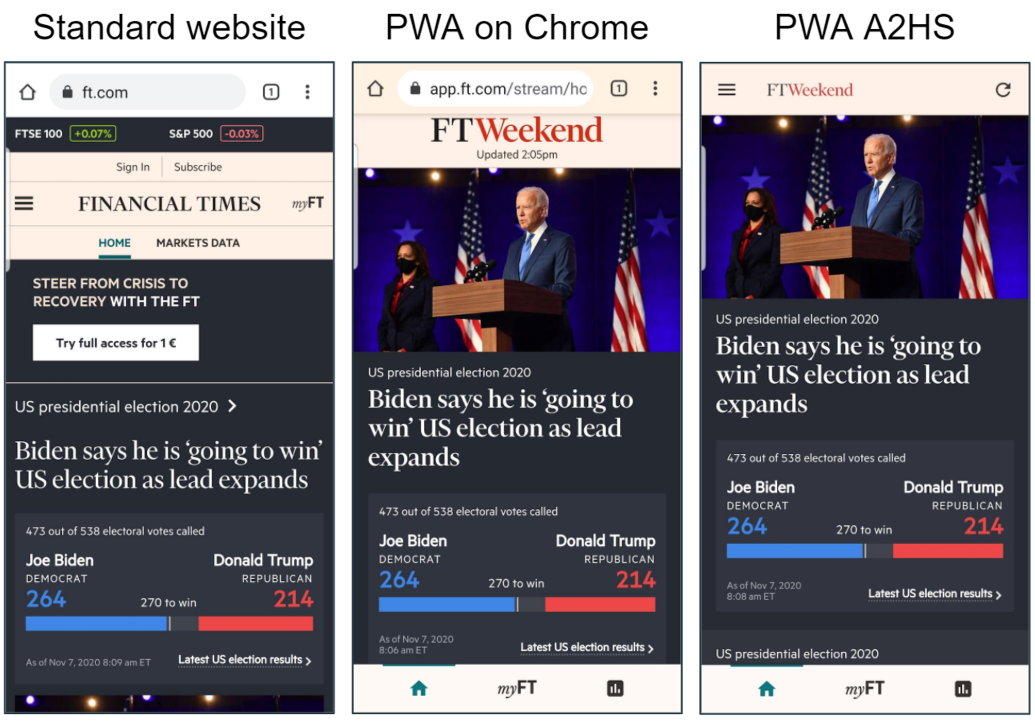


Figure 1.1: Graph of Native Apps vs Web Apps vs PWAs. [The image was created by the authors of this paper]



US presidential election 2020

Biden says he is 'going to win' US election as lead expands

473 out of 538 electoral votes called

Joe Biden DEMOCRAT 264 — 270 to win — Donald Trump REPUBLICAN 214

As of Nov 7, 2020 8:09 am ET — Latest US election results

As of Nov 7, 2020 8:06 am ET — Latest US election results

As of Nov 7, 2020 8:08 am ET — Latest US election results

Figure 1.2: Financial Times PWA comparison. [The image was created by the authors of this paper]

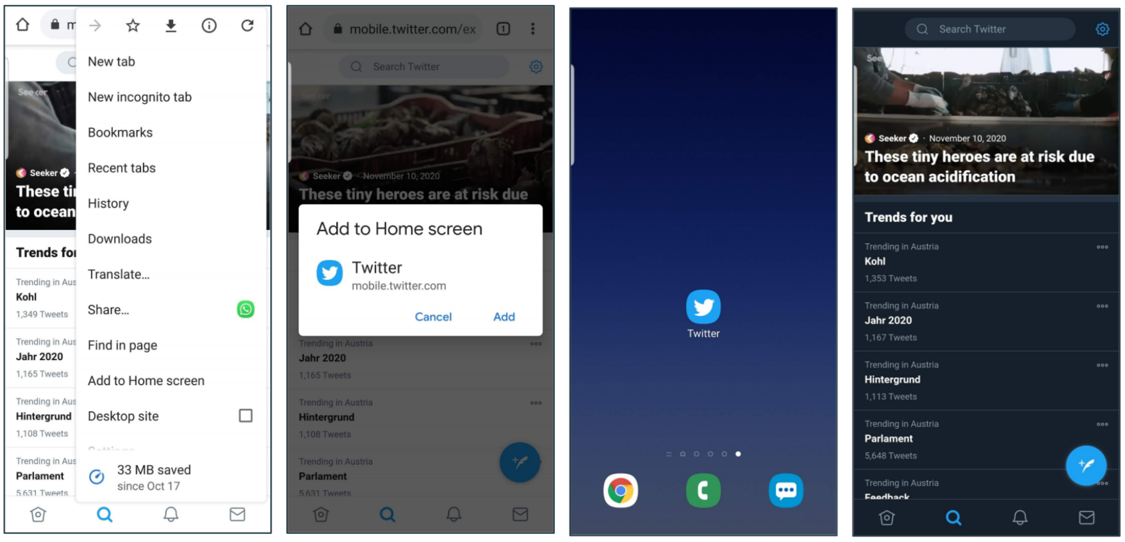


Figure 1.3: Steps to add Twitter to home screen. [The image was created by the authors of this paper]

1.2.1 Add Twitter PWA to Home Screen

To add a PWA in an Android device was use Chrome browser and Twitter to have another example. It works pretty well since you can see the icon in your home screen and get notifications, but it takes sometime for the icon to show up in your home screen see figure 1.3.

Chapter 2

Support

Unfortunately not all browsers on all platforms and operating systems do fully support all the features PWAs have to offer. So in this section you are getting an overview of which features are supported on which platforms.

2.1 Mobile Devices

The support of mobile devices is generally pretty well covered, but a few improvements still can be made. A short overview can be seen in table 2.1

2.1.1 iOS

PWA support on iOS devices is quite limited in terms of browser choice, as there is only Safari supporting PWAs at all, but at least for Safari one does get nearly all available PWA features like service workers and A2HS (add to home screen). Sadly the Push API which is required to send push notifications still is not supported.

If you wanted to use another browser on iOS, like for example Chrome or Firefox, the support is very unpleasing, as they straight up do not support PWAs at all.

2.1.2 Android

On Android however the situation is much better. All main browsers like Chrome, Firefox, Samsung Internet or the UC Browser support all key features of PWAs like A2HS, Service Workers and even push notifications.

	iOS			Android			
	Safari	Firefox	Chrome	Firefox	Chrome	Samsung Internet	UC Browser
Add to home screen (A2HS)	✓	✗	✗	✓	✓	✓	✓
Service Workers (Offline Browsing)	✓	✗	✗	✓	✓	✓	✓
Push Notification	✗	✗	✗	✓	✓	✓	✓

Table 2.1: Support for mobile devices.

2.2 Desktop Devices

Desktop device support is overall satisfying. All three main operating system have browsers that do support all features of PWAs. In table 2.2 is a quick overview of the three main operating systems and their browser support.

2.2.1 Windows

The support on Windows in general is pretty good as the most common browsers like Chrome and Edge fully support all the PWA features and Firefox also supports everything apart from the A2HS functionality.

2.2.2 macOS

macOS is currently in completely the same position as Windows with full PWA support for Chrome and Edge and full support apart from A2HS for Firefox.

2.2.3 Linux

On Linux the selection of browsers supporting PWA's is very weak unfortunately with only Chrome supporting the features fully.

	Windows			macOS				Linux
	Firefox	Chrome	Edge	Firefox	Chrome	Edge	Safari	Chrome
Add to home screen (A2HS)	✗	✓	✓	✗	✓	✓	✗	✓
Service Workers (Offline Browsing)	✓	✓	✓	✓	✓	✓	✓	✓
Push Notification	✓	✓	✓	✓	✓	✓	✗	✓

Table 2.2: Support for desktop devices.

2.3 Notifications

Although Firefox supports the basic notifications there are experimental features which it does not support. For basic notifications that is not a problem, but for more complex ones there is currently no standard. Additionally, there seems to be no up-to-date reliable library, tool or function to trigger notifications after a certain time or at a certain date.

Chapter 3

Technology

This chapter describes the main characteristics of Progressive Web Apps at first. This section encompasses various properties like for example that PWAs are installable to the home screen and discoverable through search engines. Furthermore, other important factors like network independency are mentioned and roughly explained. Secondly, in next section, the architecture of PWAs is debated. That section states what is at least necessary to provide the minimum functionality and what is generally possible.

3.1 Main Characteristics

As described before, Progressive Web Apps are web applications. In addition to standard websites, they use new technological capabilities of modern web browser in order to obtain a native app-like feeling. They use service workers and a manifest, which let them become reliable and installable. They are also armed with a fallback-plan, which means that they will always provide the core experience if the capabilities of modern browsers are not available [Richard and LePage 2020].

3.1.1 Discoverable

The web app can be found through search engines, since it has an URL and is indexable. As a consequence, search engines have access to data in relation with the user behaviour [seobility 2020].

3.1.2 Installable

Progressive Web Apps can be installed to the home screen. From there, they will open in a standalone window. The user will also be able to find them in the device search. They will have a native feeling like they would have been installed to the device like a standard native app.

3.1.3 Linkable

It is possible to share the PWA to other users by just copying and pasting the URL.

3.1.4 Network independent

A PWA will always start up, if there is an internet connection or not. Users will also expect to see their most recent content, even if they are offline. If they are offline and make a server request, they rather want to be told that there is some trouble with the connection instead of having an app crash.

3.1.5 Progressive

The Web App has to work on different modern browsers as well as be backwards-compatible.

3.1.6 Responsive

The website will be displayed correctly on all screens of devices.

3.1.7 Re-engageable

Update-notifications to the desktop/home screen will be available even if the app is closed.

3.1.8 Safe

The web application must be provided in a secure context what is ensured by HTTPS. It means that transactions must be secure. In addition to that, some features like geolocation will not work with an insecure internet connection.

3.2 PWA Architecture

A Progressive Web App consists of two main parts: The Application Shell and the Service Worker, which is a type of Web Worker. The description of these components is following below:

3.2.1 Application Shell

The Application Shell encompasses the minimum functionality of the Progressive Web App. It can be imagined like the 'skeleton' of the App and includes least necessary HTML, CSS and JavaScript in order to work. This core functionalities are able to work without an internet connection (offline). For fetching data from a server, the Application Shell communicates with the Service Worker [Addy 2019].

3.2.2 Service Worker

The Service Worker is a script that runs in the background of the Progressive Web App and manages the communication between client, cache and server. Since it runs in its own thread, it does not interrupt or slow down the Application Shell. Furthermore, it also has no possibilities to change any element in the DOM. The Service Worker manages how HTTPS-responses are stored in the cache. This leads to a good user experience when the user tries to load content while being offline [The Net Ninja 2019]. See section 3.3 for more information about this process.

3.2.3 Manifest

The Manifest file provides all important information about a Web App, in order to make it installable on the home screen of a device and make it usable offline. Also it is required to provide a richer experience like better performance. The Manifest is written in JSON and contains the following attributes [MDC contributors 2020]:

- background_color: This background color appears before the stylesheet is loaded.
- categories: Categories define where this PWA belongs to. E.g. 'Books', 'Education' or 'Medical'.
- description: This string explains, what the application does.
- dir: 'dir' stands for 'direction' and defines, in which direction-capable members of the manifest are shown. E.g. Arabic goes from right to left.
- display: This attribute allows the developer to set a preferred display mode like full screen or minimal UI.

- iarc_rating_id: This string determines the target age of users to visit the website.
- icons: Determines the icon-path, icon-size and icon-type for the PWA.
- lang : This string sets the language of the PWA.
- name: This is the title of the application.
- orientation: This attribute determines if the screen is presented in e.g. portrait or landscape mode.
- prefer_related_applications: This value determines, which platform (see 'related_applications') is preferred.
- related_applications: This array of objects determines, from which platform(s) the PWA is downloadable.
- scope: Defines the navigation scope of the web application.
- screenshots: Showcase pictures of the app. E.g. this can be used in any app store.
- serviceworker: This attribute describes the service worker, which needs to be installed.
- short_name: This string represents a short cut version of the actual app name, if a textfield is not long enough to show the full name.
- shortcuts: A user can type e.g. '/create/event' into the URL in order to create a new event.
- start_url: The start url is used to launch the web app.
- theme_color: This color-attribute defines the default color theme of the PWA.

```
"name": "HackerWeb",
"short_name": "HackerWeb",
"start_url": ".",
"display": "standalone",
"background_color" : "#fff" ,
"description": "Eine einfach lesbare Hacker News App.",
"icons": [{
    "src": "images/touch/homescreen48.png",
    "sizes": "48x48",
    "type": "image/png"
    }, {
    "src": "images/touch/homescreen72.png",
    "sizes": "72x72",
    "type": "image/png"
    }, {
    "src": "images/touch/homescreen96.png",
    "sizes": "96x96",
    "type": "image/png"
    }, {
    "src": "images/touch/homescreen144.png",
    "sizes": "144x144",
    "type": "image/png"
    }, {
    "src": "images/touch/homescreen168.png",
    "sizes": "168x168",
    "type": "image/png"
    }, {
    "src": "images/touch/homescreen192.png",
    "sizes": "192x192",
    "type": "image/png"
}],
"related_applications": [{
    "platform": "Web"
    }, {
    "platform": "play",
    "url": "https://play.google.com/store/apps/details?id=cheeaun.hackerweb"
}]
```

Listing 3.1: A typical example of a manifest.json file which can be found in all PWAs.

The following code-snippet shows an example of a Manifest file [MDC contributors 2020].

3.3 PWA Online/Offline Mode

The following figure demonstrates how a network connection of a PWA works. It basically works like browsing on a standard website. The user requests a resource on the server by clicking on an element like a button or a hyperlink. Thereon, the browser sends a HTTPS-request to the web server and obtains a HTTP-response, which includes the requested resources. Before the new content is displayed on the browser, the Service Worker puts parts of the HTTP-response into the cache, in order to make the experience faster for HTTP-requests in the future [The Net Ninja 2019].

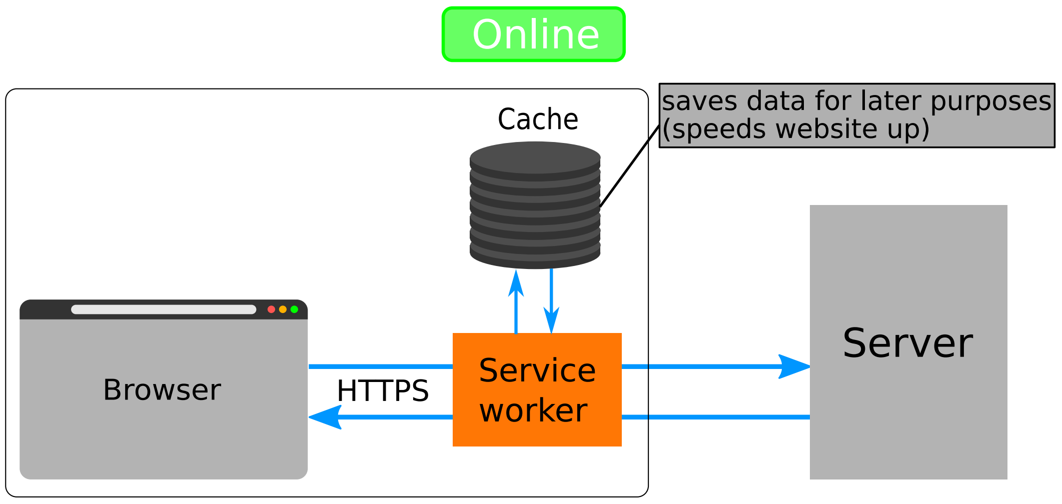


Figure 3.1: PWA is connected to the internet. [The figure was drawn by the author but inspired by [The Net Ninja 2019]]

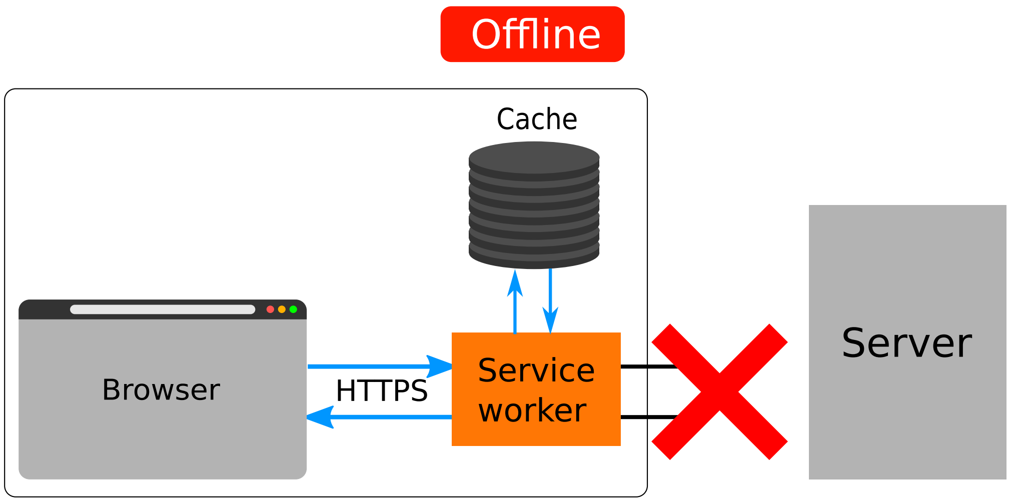


Figure 3.2: PWA is cut off from the internet. [The figure was drawn by the author but inspired by [The Net Ninja 2019]]

Figure 3.2 shows what happens, if the connection to the server was cut off. In this case, the browser sends a HTTPS-request as usual, but the request is redirected to the cache by the Service Worker instead. It is not possible to fetch any new content, but the user experience remains as if the user were online.[The Net Ninja 2019]

Chapter 4

Experience

In this chapter the experiences between regular Web Apps and progressive Web Apps are debated. This encompasses a rough definition of each technology and a summary of compared main features and benefits.

4.1 Regular Web App VS Progressive Web App

A regular Web App is built by a combination of different web technologies like HTML, CSS and JavaScript and runs in web browsers. The architecture allows the website to be accessible on many different devices and screens. Depending on the browser, they either provide native-features or not [Khurana 2019].

Basically, a Progressive Web App is a typical Web App, that additionally provides the user with an enhanced user experience. It is a perfect combination of a native and standard website experience [Khurana 2019].

4.2 Native Experience

Even though it runs 'only' in the web browser, it feels like it would be installed like a native app on the device. It also delivers features like push notifications to the home screen or desktop, when the Web App is currently opened or not [Khurana 2019].

4.3 Ease of Access

Usually when users want to download an app, they need to do multiple steps in a row, like opening the App Store, installing the app and so on. In the case of a PWA it is enough to obtain the URL [Khurana 2019].

4.4 Faster Services

Since PWAs can use the cache, it can deliver content faster even if other parts of the content have not loaded yet [Khurana 2019].

4.5 Updated Real-Time Data Access

Mobile developers can provide to be always updated. This occurs because the Web App needs only to be updated on the web server [Khurana 2019].

4.6 Lower Cost

Progressive Web Apps do not need to be installed like native apps, thus they do not need to be hosted on a platform. This leads to lower development costs. In addition, this leads to a higher return of investment [Khurana 2019].

Chapter 5

Hosting and App Stores

Just like any other website PWAs have to be hosted somewhere in order to be accessible. There are all the same choices one would suspect for a regular website, but there is also the option to place a PWA into an app store. This chapter describes the options a developer or publisher has to deploy a PWA onto a server or app store.

5.1 Hosting a PWA

Hosting a PWA is no different than hosting a regular website, because a PWA is first and foremost just a regular website. Naturally, this means there are no extra costs or requirements to host a PWA. Nevertheless, the basic options for hosting a website or PWA are listed in the following sections.

5.1.1 Self-Hosting

The easiest way to get access to a website is to have it locally accessible. This is also used during the development process albeit usually in a developer mode where certain features are deactivated or restricted in their behaviour. The developer can work on the site and see the results of the local files in their browser. Taking this concept further any computer can become a server if the data is accessible from other computers. Following the website running locally on the developers computer can be shared to the world.

Unfortunately, deploring this method comes with security issues which the developer has to have in mind and address, or otherwise malicious actors might cause damage to the site, the developer's computer and third parties. This reason alone does not make it very beginner friendly. Additionally, the developer has to know how to configure his server, set up certificates, probably get a domain name and many other technical aspects.

5.1.2 Paid Services

The far easier and safer option is to use one of the countless paid services to host the website or PWA. The downside of course is that these services cost a monthly fee in order to use them. Furthermore, the developer still has to buy a domain name, but there are also services who include a domain name in their offers, but they might only be free for a limited time.

There are different kind of server hosting services which are offered as described by O'Brien [O'Brien 2020]. The cheapest option is usually to rent a space on a shared server. As well as being the cheapest option it does only allow a limited amount of traffic before the shared server space can not handle the demand anymore. The next step is cloud hosting. Here the amount of traffic is a bit more manageable, as the cloud consists of multiple servers and in case of a demand spike more resources can be provided on short notice in reasonable margins. Then there are virtual private servers which are similar to a shared

```
pages:
    stage: deploy
    script:
        - npm install
        - npm run build
        - mkdir .public
        - cp -r dist/* .public
        - mv .public public
    artifacts:
        paths:
            -public
    only:
        - master
```

Listing 5.1: Simple extract of a Continuous Integration (CI) script used in a repository hosted on GitLab. In this case npm is used to deploy a node.js based PWA project.

server, but there is a dedicated section of the server reserved for your website opposite to a shared server. And finally, the most expensive option is a dedicated server, which is only really necessary for huge websites with a large number of visits and resources.

5.1.3 GitHub and GitLab Pages

Both GitHub [GitHub 2020] and GitLab [GitLab 2020] offer a service called Pages which can be used to deploy the code of static websites hosted on GitHub or GitLab to a dedicated space on their servers. GitHub and GitLab handle the deployment differently. While GitHub has a very basic, but easy system GitLab deploys the website by using Continuous Integration through their pipelines. This offers more control, but at the same time makes deploying the website a bit harder than in GitHub. Fortunately, the GitLab documentation concerning this topic and in general is very comprehensive and small basic scripts like depicted in listing 5.1 are usually enough to get a basic deploy.

5.2 Placing PWAs in App Stores

Additionally to hosting PWAs like as a website they can also be submitted to app stores [Vu 2020a]. This offers more use cases and a much wider audience reach. At the same time this also blurs the lines between website and native app even further. As the PWA can act in some cases just like a native app the question is then if it makes sense to have a PWA instead of an actual native app, which might be easier to get into certain stores and might be better integrated into the operating system. The decision on the sensibility obliges the developer or publisher in the end. There is no definitive answer for all PWAs if adding them to a store is a sensible decision, just a waste of time or if the PWA should have been a native app in the first place.

5.2.1 Apple App Store

The Apple App Store is known for having a high standard and a rigorous review process [Vu 2020b]. Naturally, getting PWAs to the stores is difficult. The PWA has to act and react just like a native app. This is a problem, because as described in chapter 2 some PWA features are yet to be properly integrated in the current version of their mobile operation system iOS. Due to this developers have to use certain tools and libraries to trick the store into accepting PWAs.

Currently the easiest way to get a PWA into the Apple App Store is to use Microsoft's PWA Builder as is describes in section 6.1.2 in chapter 6. This tool can convert the PWA into an XCode Project by simply supplying the URL of the website to the tool. Unfortunately, to get the full functionality out of the XCode Project the developer needs to be on macOS which means either the developer has a machine with macOS as operating system or a service which emulates macOS is required. An example for such a service would be MacinCloud which starts at a price of $20 per month.

5.2.2 Google Play Store, Samsung Galaxy Store & Huawei AppGallery

Publishing a PWA to the Google Play Store, the Samsung Galaxy Store or the Huawei AppGallery is supported and does not even require significant additional effort to publish to all three in one go [Caricofe 2020; Samsung 2020a; Zubair 2020]. All Android-based stores offer the ability to publish PWA by utilising Google's Trusted Web Activities (TWA) technology. TWA basically consists of a bare Chrome tab which runs the PWA independent of Chrome itself. Converting a PWA into an app for the Android-based stores can be done by simply using Android Studio like with any other native app. Another option is to use a service like PWA2APK. This service just streamlines the configuration which would normally be necessary in Android Studio and thus produces the same result as Android Studio in the end.

When publishing to multiple Android-based stores it is important that the app receives a unique package name for every store. This prevents problems like pushing the update on the wrong store and other similar problems. Other than that TWA-based apps are store ready and can be uploaded to the Android stores like any other app.

5.2.3 Microsoft Store

Microsoft is very inclusive regarding PWAs [MS Edge Team 2020]. They offer a variety of different ways to get a PWA into their store. One way to do it is to use their IDE Visual Studio and submit it that way. PWAs created through Microsoft's PWA Builder can be submitted to the store by using the Windows Dev Center. Additionally, to the tools provided to get the PWA into a store there is also the option of having the PWA automatically submitted to the Microsoft Store. For this to have a chance of happening the PWA has to follow a small list of criteria set by Microsoft. If that is the case the Microsoft's Bing indexing service may package the PWA and add it to the store automatically. The only downside is that Microsoft does not specify what sites it selects to include into the store. The only information given is that not all sites which fulfil the criteria will be selected as they are still rolling out this feature.

5.2.4 App Store Fees

Publishing to the app stores is not free. Stores the most prominent ones have a publishing or membership fee and every store takes a commission cut from the revenue earned by the app.

The two biggest stores which do not have a publishing or membership fee are Samsung's Galaxy Store [Samsung 2020b] and Huawei's AppGallery [Huawei 2020]. Google [Google 2020d] and Microsoft [Microsoft 2020c] have reasonable entry fees for individuals at $25 and $19 respectively. In case the publishing entity is registered as a company a additional $99 yearly is added to the cost for using Google's Play Store. In the Microsoft Store the entry fee just increases from the initial $19 to $99. The most expensive app store to publish in is Apple's App Store. There an individual has to pay $99 yearly and a company $299 yearly [Apple 2020].

The commission the stores take are fairly similar, the only exception being the Microsoft Store which has a fixed rate of 15% [MS Store Team 2019], presumably to attract developers and publishers. All the other stores have a rate of 15% to 30% depending on who publishes what. The vast majority will pay the 30% though.

Chapter 6

Tools

There are already quite a few tools available for generating PWAs out of existing websites, for building PWAs from scratch or for testing PWAs. Fortunately due to the rising popularity and overall support of PWAs a lot of popular frameworks have added PWA building and testing support to their products and also big companies like Google or Microsoft have developed new and intuitive ways for developing PWAs.

6.1 Developing PWAs

In this section not only some of the most popular ways to build PWAs are explained, but also some quite new frameworks which are still very interesting and enjoy high popularity even though they may still be in a beta status and not fully production ready yet.

6.1.1 Workbox

Workbox [Google 2020e] is developed by Google Chrome and is a collection of JavaScript libraries which simplify writing service workers, be it for implementing pre or runtime caching, routing or many more useful features. Since service workers are one of the fundamental technologies of a PWA, Workbox is very popular and there is a lot of material online on how to setup and work with it.

6.1.2 Microsoft PWA Builder

Microsoft's PWA Builder [Microsoft 2020b] offers two ways to get started in building PWAs. The first method which is to enter a URL to an existing website on the PWA Builder website and turn that one into a progressive web app. The second way is to start from scratch with their PWA Starter project.

6.1.2.1 Converting a website

After submitting the URL in the form the website is evaluated based on three criteria. It checks if we have a manifest and if it is configured correctly, it is looking for a service worker and it is looking at the security aspect of the website. On the same page PWA Builder offers the user to make a PWA out of the website. To do that a ZIP file can be downloaded contains a few preconfigured resources to cover the basics of a PWA.

Inside the zip there are a "manifest.json" and two folders which contain a documentation for the manifest and the service workers respectively. To integrate the manifest into our website we have to follow the "web_next_steps.md" document in the "web" folder. The other folder in our ZIP contains the scripts for the service worker. To include this in our project we have to follow the instructions in the comment of the file "pwabuilder-sw-register.js" and then include the "pwabuilder-sw.js" script in our root and add it to the header of our HTML file.

This provides a good start to make the website into a progressive web app. There are also features and demos on the PWA Builder website which be easily included into the new PWA.

6.1.2.2 PWA Starter

The PWA Starter project [Microsoft 2020a] can either be downloaded as a ZIP file or be cloned from GitHub via Git. In the project included is a ReadMe which explains the most important information about the project and its setup.

The recommended editor for PWA Builder projects is VSCode, which in connection with TypeScript offers helpful functionality in auto-completing, code hinting and so on. Furthermore, there are two extensions for VSCode which offer further assistance in the development.

The basis of the framework is lit-element which provides the base web components and encompasses the creation of new custom elements. Pre-made components like date pickers, drop-downs, buttons, etc. can be used from the FAST Components library. Additionally, there is a library for routing called Vaadin Routing included. The service workers come from Google's Workbox. Furthermore, two PWABuilder components are already integrated too. The install component adds the feature to install the application to our desktop or home screen if on mobile. The update component notifies the user if the PWA has changed so that the user can refresh the content.

In the project there is already a manifest which contains everything we need and can be modified to the developer's wishes. The code is separated by the pages and the components which are used in the PWA. The index page script includes the routing in the "firstUpdated" function which is part of the lit-element library. The classes for the pages extend LitElement which achieves that each page has its own style function which includes the CSS for the page and a render function which contains the HTML with the components and TypeScript.

6.1.3 Flutter

Flutter [Google 2020b] is a UI toolkit built by Google to build Android, iOS as well as web apps in a single codebase. Flutter uses the Dart language for their framework and expands it with C++ and JavaScript for developing web apps.

A big advantage of Flutter is, that you only need to develop one codebase and you get native apps for Android, iOS and web out of it and if PWAs supported features are not enough for your apps, you can simply add native code for whatever app you need for truly native features, that can not be achieved by PWAs, such as push notifications for iOS.

The downside to all those possibilities is, that Flutter, because it is a very young technology, has no production ready web support as of now. That means one can still develop PWAs in Flutter via downloading Flutters beta version, but things may brake as Google still builds and updates Flutters web components.

With that being said, Flutter still is a great choice for building PWAs if one wants to avoid restrictions when it comes to uploading apps to the different stores and furthermore it is quite easy to develop even for complete beginners, because of the very good documentation Flutter provides and also the already big community it has grown.

6.1.4 Ionic

Ionic [Ionic 2020] is a framework for developing hybrid apps and PWAs based on HTML5, CSS, Sass and JavaScript or Typescript. It can be used standalone or in combination with other framworks such as Vue.js and Angular and it provides libraries for components that can be used easily to develop interactive Apps.

6.1.5 Vue.js

Vue.js [You 2020] is a JavaScript framework for developing single-page applications with very little overhead. Together with Vue.js based frameworks like Quasar or Nuxt.js, PWAs can be developed quite easily and with, as already mentioned above, very little overhead.

6.1.6 Angular

Angular [Google 2020a] is a TypeScript based framework used for creating single-page applications. It is one of the biggest frameworks with the most overhead and therefore a perfect fit for creating more complex websites that need more than just a few basic features. Its huge documentation collection helps you develop everything you need in a very clearly arranged style, with many tutorials and also downloadable examples.

6.1.7 Webpack

Webpack [*Webpack* 2020] is an open-source module packer, which allows one to compile JavaScript, sass, css and many more different files into bundled files for browser usage. It renders dependencies into a diagram, which allows it to be modular, which is ideal for creating PWAs.

6.2 Testing PWAs

There are a few options that evaluate and test your PWAs features. Some of them are paid services and therefor offer more functionality, but the most important features can also be tested with free services.

6.2.1 Lighthouse

Lighthouse [Google 2020c] is a free testing tool developed by Google and is integrated into the Google Chrome Dev Tools, but can also be used as a Chrome extension or via npm as a command line tool. Lighthouse runs a few tests against the page and then reports on how well it did in certain categories. From this report on one can easily find and fix flaws of their PWAs.

6.2.2 Lambda Test

Lambda Test [LambdaTest 2020] is a cross browser testing cloud, which lets one run automated Selenium test scripts for all different browsers and operating systems, as well as live testing PWAs in one of the many browsers via the cloud, taking full-paged screenshots, testing the responsiveness of the website. Even though it is not a tool for specifically testing PWAs, cross browser testing is very important for PWAs, because of the different browsers often not supporting all features.

However all of this useful functionality is not free of charge. Lambda Test subscription plans start from $15 per month, but some features can be tested for free for as long as one hour.

6.2.3 PWA Testing Tool

The PWA Testing Tool from SEO Review Tools [SEO Review Tools 2020] is a free testing tool which allows one to test an existing website via the url and then scores its features and shows where one can still improve their PWA. It tests Googles PWA requirements as well as some additional fields like loading speed.

6.2.4 Speedcurve

Speedcurve [Speedcurve 2020] is a paid testing tool, which evaluates nearly all facets a website has to evaluate. Starting from loading times to cpu times, Lighthouse scores and much more it is really a

powerful testing suite for regular websites as well as PWAs, as it has an own improvement section for PWAs.
As already mentioned this tool is very comprehensive and all these features contribute to the higher price of Speedcurve's subscription plans. Starting from $114/month it surely is not affordable and sensible for small PWAs, but if one is planning to make an extensive PWA, Speedcurve might be the way to go. Also there is a free testing option for existing websites via their url.

Chapter 7

Concluding Remarks

After all the information gathered in this survey and having developed a PWA we have seen that it has many advantages if you start to develop your PWA and offer it to your users. PWAs are still not completely supported by all devices so some of your users will have some difficulties and a bad experience like iOS users sometimes since it is not fully supported unless you use Safari.

For developing and testing there are many tools but not all have a considerable community and good documentation about how to use them. Our experience with PWA Builder was good but we were not able to schedule push notifications and we were not able to find a standard way to do the push notifications, so there is still some best practices missing, which will give you some problems when you try to do more complex things. While most of the test tools are paid if you want to use a complete suite, Lighthouse from Google is free and quite good, even though not quite as comprehensive as other tools.

We will definitely recommend to start developing a PWA for your site since it offers you many advantages even though there are still some flaws, but you will have an advantage over your competitors when PWAs become more popular and the development tools are more mature and have more documentation and practices on how to develop.

Bibliography

Addy, Osmani [2019]. *The App Shell Model*. 14 May 2019. `https://developers.google.com/web/fundamentals/architecture/app-shell` (cited on page 8).

Apple [2020]. *Enrollment*. Support page. 11 Dec 2020. `https://developer.apple.com/support/enrollment/` (cited on page 17).

Caricofe, Lori [2020]. *How Apple is Supporting Progressive Web Apps (PWA)?* Article. 07 Mar 2020. `https://medium.com/datadriveninvestor/how-apple-is-supporting-progressive-web-apps-pwa-6cdbb2f844e6` (cited on page 17).

Dom [2020]. *12 Best Examples of Progressive Web Apps (PWAs) in 2020*. 13 Mar 2020. `https://simicart.com/blog/progressive-web-apps-examples/` (cited on page 1).

GitHub [2020]. *Getting Started with GitHub Pages*. Article. 24 Jul 2020. `https://guides.github.com/features/pages/` (cited on page 16).

GitLab [2020]. *GitLab Pages*. Documentation. 11 Dec 2020. `https://docs.gitlab.com/ee/user/project/pages/` (cited on page 16).

Google [2020a]. *Angular*. 07 Dec 2020. `https://angular.io` (cited on page 21).

Google [2020b]. *Flutter*. 07 Dec 2020. `https://flutter.dev/web` (cited on page 20).

Google [2020c]. *Lighthouse*. 07 Dec 2020. `https://developers.google.com/web/ilt/pwa/lighthouse-pwa-analysis-tool` (cited on page 21).

Google [2020d]. *Service fees*. Support page. 11 Dec 2020. `https://support.google.com/googleplay/android-developer/answer/112622?hl=en` (cited on page 17).

Google [2020e]. *Workbox*. 07 Dec 2020. `https://developers.google.com/web/tools/workbox` (cited on page 19).

Huawei [2020]. *HUAWEI AppGallery Connect Distribution Service Agreement For Paid Apps*. Documentation. 11 Dec 2020. `https://developer.huawei.com/consumer/en/doc/distribution/app/30203` (cited on page 17).

Ionic [2020]. *Ionic*. 07 Dec 2020. `https://ionicframework.com/pwa` (cited on page 20).

Khurana, Vijeta [2019]. *How is Progressive Web App different from Regular Web App?* 03 Jan 2019. `https://websitepulse.com/blog/progressive-web-app-different-from-regular-web-app` (cited on pages 13–14).

LambdaTest [2020]. *LambdaTest*. 07 Dec 2020. `https://lambdatest.com/` (cited on page 21).

MDC contributors [2020]. *Web app manifests*. 28 Nov 2020. `https://developer.mozilla.org/en-US/docs/Web/Manifest` (cited on pages 8, 10).

Microsoft [2020a]. *pwa-starter*. GitHub Repository. 11 Dec 2020. `https://github.com/pwa-builder/pwa-starter` (cited on page 20).

Microsoft [2020b]. *PWABuilder*. Website. 11 Dec 2020. https://www.pwabuilder.com/ (cited on page 19).

Microsoft [2020c]. *Regsiter as an app developer*. Website. 11 Dec 2020. https://developer.microsoft.com/en-gb/store/register/ (cited on page 17).

MS Edge Team [2020]. *Progressive Web Apps in the Microsoft Store*. Microsoft Edge Team. 05 Mar 2020. https://docs.microsoft.com/en-us/microsoft-edge/progressive-web-apps-edgehtml/microsoft-store (cited on page 17).

MS Store Team [2019]. *Updated Microsoft Store App Developer Agreement: New Revenue Share*. Microsoft Store Team. 06 Mar 2019. https://blogs.windows.com/windowsdeveloper/2019/03/06/updated-microsoft-store-app-developer-agreement-new-revenue-share/ (cited on page 17).

O'Brien, Fred [2020]. *How to Host a Website: the Complete Beginner's Guide*. Article. 25 Nov 2020. https://websitebuilderexpert.com/hosting-websites/%5C#section-4 (cited on page 15).

Richard, Sam and Pete LePage [2020]. *What are Progressive Web Apps?* 24 Feb 2020. https://web.dev/what-are-pwas/ (cited on pages 1, 7).

Samsung [2020a]. *Get Started in Galaxy Store*. Documentation. 11 Dec 2020. https://developer.samsung.com/galaxy-games/get-started-in-galaxy-store.html (cited on page 17).

Samsung [2020b]. *Terms and Conditions*. 13 Aug 2020. https://seller.samsungapps.com/help/termsAndConditions.as (cited on page 17).

SEO Review Tools [2020]. *PWA Testing Tool*. 07 Dec 2020. https://seoreviewtools.com/pwa-testing-tool/ (cited on page 21).

seobility [2020]. *Progressive Web Apps*. https://seobility.net/de/wiki/Progressive_Web_Apps%5C#Discoverable (cited on page 7).

Speedcurve [2020]. *Speedcurve*. 07 Dec 2020. https://speedcurve.com/ (cited on page 21).

The Net Ninja [2019]. *PWA Tutorial for Beginners #12 - Fetch Events*. 17 May 2019. https://youtube.com/watch?v=C0vh57N7vM4%5C&list=PL4cUxeGkcC9gTxqJBcDmoi5Q2pzDusSL7%5C&index=12 (cited on pages 8, 10–11).

Toonen, Edwin [2020]. *What is a progressive web app (PWA)? Why would you want one?* 06 May 2020. https://yoast.com/what-is-a-progressive-web-app-pwa/ (cited on page 1).

Vu, Luke [2020a]. *Publishing PWAs to Major App Stores: The Whys and Hows*. Article. 13 Feb 2020. https://www.simicart.com/blog/pwa-app-stores/ (cited on page 16).

Vu, Luke [2020b]. *Publishing PWAs to Major App Stores: The Whys and Hows - Apple App Store*. Article. 13 Feb 2020. https://simicart.com/blog/pwa-app-stores/%5C#Apple_App_Store (cited on page 16).

Webpack [2020]. 07 Dec 2020. https://webpack.js.org/ (cited on page 21).

You, Evan [2020]. *Vue.js*. 07 Dec 2020. https://vuejs.org/ (cited on page 21).

Zubair, Zorain [2020]. *How to publish an App on Huawei AppGallery*. Article. 01 Jan 2020. https://techlapse.com/global/how-to-publish-an-app-on-huawei-appgallery/%5C#How_to_publish_an_app_on_AppGallery_in_3_steps (cited on page 17).