
# From Native to Progressive Web App
# A Proof of Concept

**Gunharth Randolf**
**1810738273**
**Entwicklung & Betrieb Mobiler Informationssysteme**
**Web Communication & Information Systems**
**29. July 2019**

**Abstract**

Lorem ipsum

# Contents

# 1. Introduction

In 2015 the Google Developer Alex Russel first drafted the term Progressive Web Apps (PWAs) and formulated the concept of PWAs as it stands today.

Full Progressive App support will distinguish engaging, immersive experiences on the web from the "legacy web". Progressive App design offers us a way to build better experiences across devices and contexts within a single codebase but it's going to require a deep shift in our understanding and tools. (Russell, 2015)

Since then, notably Google and its Chrome browser focus on the implementation of all necessary technologies in order to fully support PWAs. Nearly every new release offers the Chrome browser adds further support for PWAs. As an example, at the time of writing Chrome 76 shows an install icon to the left of the address bar, if a site meets the Progressive Web App installability criteria[1].

So far, the bedrock for engaging and immersive experiences on mobile devices has been bound to the development of native and/or hybrid apps. However, developing a native application is an expensive solution. Programmers spend years learning to code heavy native apps and app owners usually invest a huge budget finding a development team (Nguyen, 2019). That said, at least two versions need to be developed, maintained and published to the Apple and Google App Stores respectively.

As PWAs are still a rather new concept there are known limitations in terms of unified browser support of APIs[2], missing filesystem access and platform-level features integrations like calendar and contacts (Biørn-Hansen et al., 2018;

[1] https://developers.google.com/web/fundamentals/app-install-banners#criteria

[2] https://whatwebcando.today

Malavolta, 2016). However, there are many applications that do not depend on the before mentioned limitations. On 15th July 2019 Twitter announced its new website, which is a PWA build on one code base feeding all devices and browsers that are accessing the site on the Web. The goal for the new site was to make it easier and faster to develop new features for people worldwide and to provide each person and each device with the right experience (Croom and Baker, 2019).

The purpose for this study is to look at the technical side of PWAs, whereby the following questions will be answered. First, whether it is possible to take an existing mobile app as a prototype and re-build the app as a PWA. Second, what services and programming languages are best fitted in doing so and how well do they support the development of a PWA? Finally, I discuss my findings and any limitations I might have encountered.

# 2. Methods

This study implements three research methods to gain a broader understanding of the possibilities, state of research and technological practices regarding Progressive Web Apps.

## 2.1 Literature Search

Our searches for academic involvement in Progressive Web Apps returned a very limited amount of results as per July 2017 (Biørn-Hansen et al., 2018). The same search performed 2 years later doesn't reveal any further major releases on this topic. Scientific research is still pretty sparse. However, technology related publishers have picked up the topic with a handful of publications in 2017 and 2018 (Hume and Osmani, 2018; Ater, 2017). In 2019 though, there seems to be only one notable publication; a practical guide to PWAs by Liebel written in the German language (Liebel, 2019).

Interestingly, I was not able to find any scientific research on the commercial aspects of PWA. A comparison on budgets and running costs required by native/hybrid apps versus a PWA. Further st

## 2.2 Related Works

The Google Web Fundamentals group acts as the driving force behind the documentation on PWA related topics and the creation of blog posts and tutorials. As the supporting technologies advance with every Chrome browser release,

the Web Fundamentals website[1] is the foundation for upcoming research and studies.

Other than Google-created content, online platforms such as Medium blog posts[2] and Youtube tutorials[3] lay the foundation of technical research for this study. Notably, the technical series of articles about PWA published by Eder Ramírez Hernández[4] on Medium were extremely informative.

## 2.3 Technical Implementation

To gain a better understanding of the possibilities of Progressive Web Apps a PWA was developed. The motivation for the PWA was to take an existing native application as a reference and to re-create the application as a Progressive Web App. For this purpose I picked the *Beer With Me*[5] application, which is available on the Apple App Store as well as on Google Play. As stated by the Swedish developers Antonsson, Knutsson and Tidbeck "Beer With Me is a social application to notify your friends when you are drinking so that they can join".

[1] https://developers.google.com/web/fundamentals
[2] https://medium.com/search?q=Progressive%20Web%20Apps
[3] https://www.youtube.com/results?search_query=Progressive+Web+Apps
[4] https://medium.com/@eder.ramirez87
[5] https://beerwithme.se/

# 3. Concepts & Technical Implementation

This chapter provides a detailed look at the required concepts and the technical implementation of the PWA. The project has been open-sourced to allow verification of the results[1]. Further, the application is available online[2].

## 3.1 Requirements

The following section outlines the concepts and requirements used for the project by looking at the core functionality of the original *Beer with Me* app.

**Progressive Web App**. The functional clone of the *Beer with Me* app must implement all relevant techniques as outlined by the Progressive Web App specifications. Technologies include Service Workers, Application Shell (AppShell), Web App Manifest and serving the demo installation over HTTPS. Further, it is the intention to follow the 10 principal concepts, which lay the foundation for Progressive Web Applications, as closely as possible: Progressive, Responsive, Installable, Connectivity Independent, App-Like, Fresh, Safe, Discoverable, Re-engageable and Linkable (Osmani, 2015).

**Authentication and Authorization**. Users must be able to register for the service with a combination of E-Mail and Password. As an alternative an existing Google account may be used in order to login directly.

**Realtime Database**. Data is stored in a cloud-hosted database and synchronized in realtime to every connected client.

[1] https://github.com/gunharth/bwm
[2] https://bwm.gunicode.com

**Realtime Notifications**. If enabled by the user push-notifications shall be send to the user in realtime, whenever a new user registers for the service or when a new post was published by a user.

**Realtime Map**. If enabled by the user his/her current geolocation is saved and displayed on an online map.

**Single Page App (SPA)**. The app shall be implemented as a Single Page App with one of the main JavaScript frameworks (React, Angular or Vue.js).

Option to add a photo using the device's camera

Frontend, Backend framework/system straight forward deployment process

The following topics were defined as being not part of the requirements, as they do not have an impact on the used technologies of the PWA, nor do they have an effect on the results of this study. Hence, design considerations of the application can be neglected, nor is the intention to copy the interface and design of the original *Beer with Me* app. The resulting PWA focusses on the implementation of the functionality and not on offering a identical clone of the original. Further, all functions are implemented as a proof-of-concept following the MVP spirit (Wikipedia, 2019). Thus, to make it a user centric production ready PWA further development cycles are required.

## 3.2 Technical Implementation of Requirements

As per the actual implementation of the PWA the goal was to base the solution on the least different providers on the one hand, as well as a minimum in terms of coding and programming languages. Hence, the front-end of the application uses the basic building blocks of regular web sites being HTML, CSS and JavaScript in form of the JavaScript framework Vue.js. To satisfy the back-end Firebase was picked as the sole solution provider.

### 3.2.1 Single Page App (SPA) & Progressive Web App

For the development of the frontend the JavaScript framework Vue.js[3] and its ecosystem was used. The Vue Cli[4] offers great tooling support for Vue.js developments - among other features it comes with a preset for installing a Progressive Web App skeleton.

Listing 1: Installation and project creation commands with the Vue Cli

```
1 $ npm install -g @vue/cli
2 $ vue create beerwithme
```

On project creation Vue Cli offers you to manually select the features that will be installed. Next to Progressive Web Application support the Vue Router and Vuex was installed. The Router enables navigation services within a SPA and Vuex is responsible for the management of state within the application.

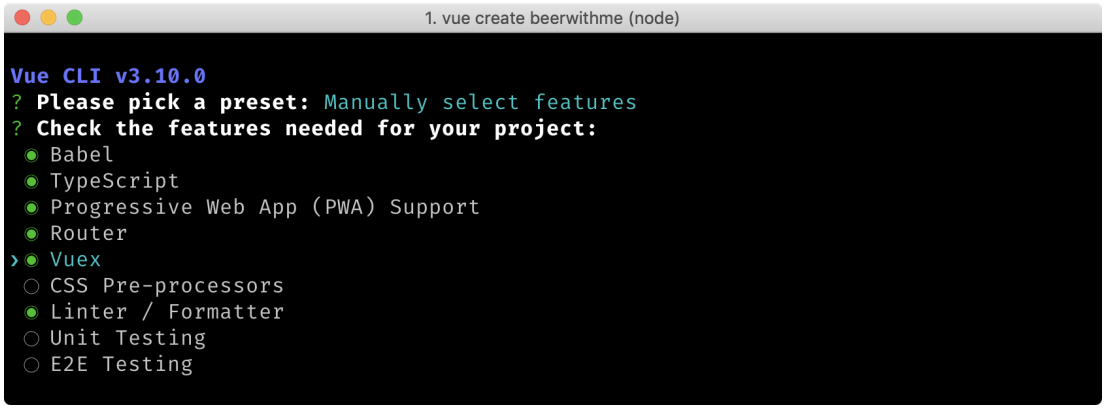



Figure 1: Installation options for a Vue.js project

The Vue Cli PWA support comes with Workbox[5] installed - a library developed by Google for adding offline support to web applications. Among other features it mainly helps with the automatic creation and management of application pre-cache and the App Shell .

Listing 2: Service Worker with Workbox and Firebase specific initiation (firebase-messaging-sw.js)

```
1 importScripts("https://www.gstatic.com/firebasejs
    /5.6.0/firebase-app.js");
```

[3] https://vuejs.org
[4] https://cli.vuejs.org
[5] https://developers.google.com/web/tools/workbox

```
2  importScripts("https://www.gstatic.com/firebasejs
     /5.6.0/firebase-messaging.js");
3
4  self.__precacheManifest = [].concat(self.
     __precacheManifest || []);
5  workbox.precaching.suppressWarnings();
6  workbox.precaching.precacheAndRoute(self.
     __precacheManifest, {});
7
8  workbox.routing.registerRoute(
9    new RegExp(
10     "https://firebasestorage.googleapis.com/v0/b/
         bmwgunicode.appspot.com/.*"
11   ),
12   workbox.strategies.staleWhileRevalidate()
13 );
14
15 firebase.initializeApp({
16   messagingSenderId: "ID"
17 });
```

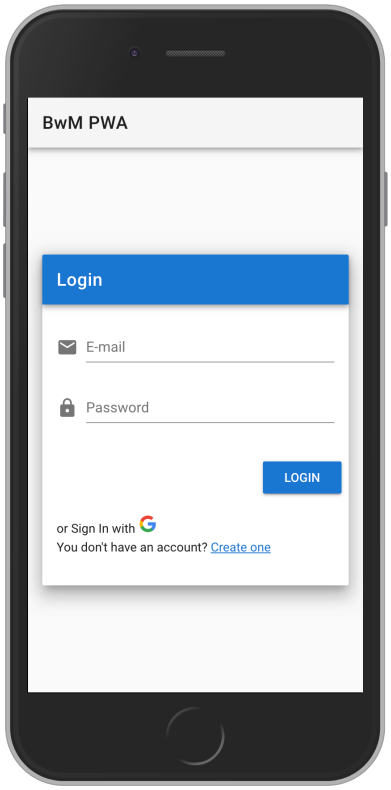
For the front-end design of the app Vuetify[6] was used. Vuetify is a responsive design component framework for Vue.js based on the material design[7] guidelines developed by Google.

Listing 3: Command to add Vuetify to the Vue.js project

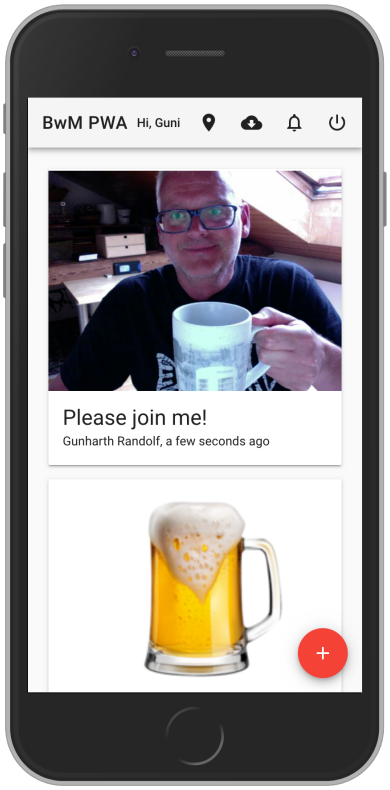
```
1  $ vue add vuetify
```

The map functionality was implemented by using the npm package vue2-leaflet[8], which is a Vue wrapper library for the open-source JavaScript library Leaflet[9]. Map tiles are served from OpenStreetMap[10].

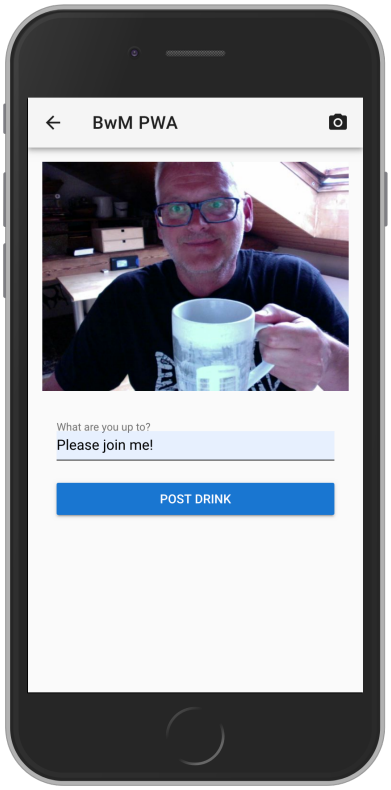
[6] https://vuetifyjs.com
[7] https://material.io
[8] https://www.npmjs.com/package/vue2-leaflet
[9] https://leafletjs.com
[10] https://www.openstreetmap.org



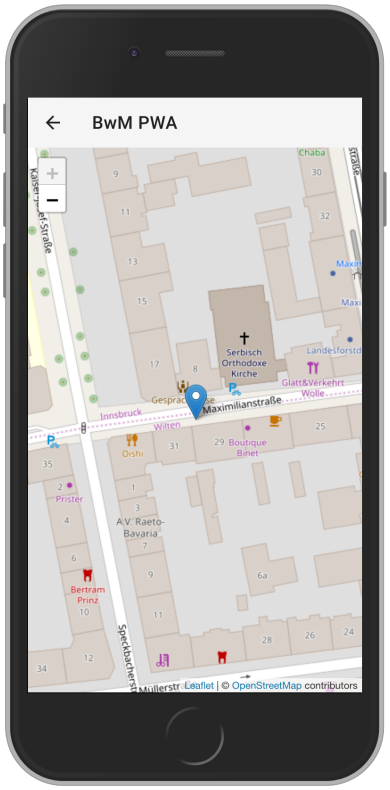

(a) Login screen




(b) Main app screen




(c) New post screen with photo option




(d) The map

Figure 2: Screenshots showing Vuetifys' material design components and the map

Figure 3 shows the Lighthouse[11] Progressive Web App report for the *Beer with Me* PWA.

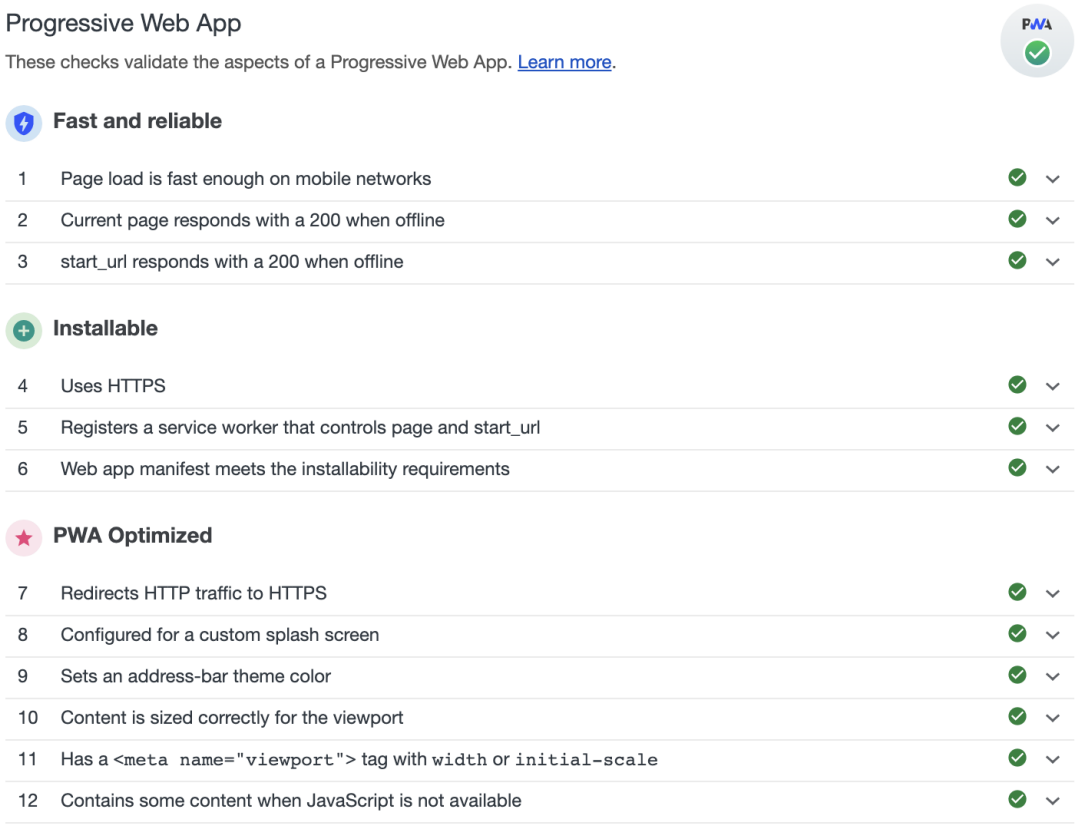



Figure 3: Lighthouse report

### 3.2.2 Cloud Provider and Realtime Functionalities

To cater for the back-end requirements I picked Firebase[12] as the provider. Around the same time in 2015, when the term Progressive Web Apps was born, Google acquired this San Francisco start-up. Since then Google has expanded upon the original service and integrated it closely into their other Google cloud ambitions - whereby the Firebase service is dedicated to providing a specialized Platform as a Service (PaaS) for mobile and web applications.

The following section demonstrates the Firebase services used for the *Beer with Me* PWA.

[11] https://developers.google.com/web/tools/lighthouse
[12] https://firebase.google.com

To use Firebase in a project the following steps need to be performed:

1. Register with Firebase
2. Create a project

Authentication and Authorization

Firebase Auth offers multiple methods to authenticate, including email and password and third-party providers like Google.

Listing 4: Firebase Auth initiation using VueJS (main.js)

```
// Init Firebase auth before Vue inits the App
firebase.auth.onAuthStateChanged(firebaseUser => {
  if (firebaseUser) {
    store.dispatch("autoSignIn", firebaseUser);
  }
  if (!app) {
    app = new Vue({
      router,
      store,
      render: h => h(App)
    }).$mount("#app");
  }
});
```

Realtime Database feeding the feed of posts on the main application page. Map positions JSON NoSQL

Listing 5: Realtime query for new posts (Home.vue)

```
firebase.db
  .collection("drinks")
  .orderBy("created_at", "desc")
  .onSnapshot(snapShot => {
    this.drinks = [];
    snapShot.forEach(drink => {
      this.drinks.push({
        id: drink.id,
        url: drink.data().url,
        comment: drink.data().comment,
```

```
11          author: drink.data().author,
12          created_at: drink.data().created_at,
13        });
14      });
15    });
```

Realtime Push Notifications

**Firebase**. Firebase is a service that Google aquired in 2015 ties in hand in hand with their goal on pushing on

# 4. Results

What answer was found to the research question; what did the study find? Was the tested hypothesis true?

Is it possible to take an existing mobile app and convert it to a PWA?

In summary,

What services and programming languages are a best fit in doing so?

How do those services and programming languages support the creation of a PWA?

What are the technical limitations, if any?

In conclusion, PWAs might not replace native apps completely within the next few years. However, the techniques involved have leveled up the standard of modern websites. Rapid enhancements of browser technologies promise further integration with devices and enduser behaviors.

One will have to wait and see when Apple and its Safari browser opens up. It may be assumed that with the pressure of Google

On the other hand, business wise? costs?

One a side note, it is puzzling that PWA as a technology has not yet arrived in the world of business and finance. Fact is, building and maintaining native mobile apps is expensive. Developing PWAs

# 5. Conclusion

Limitations, next steps , ...

What might the answer imply and why does it matter? How does it fit in with what other researchers have found? What are the perspectives for future research?

# References


Ater, T. (2017). *Building Progressive Web Apps: Bringing the Power of Native to the Browser*. O'Reilly Media, Sebastopol, CA, first edition edition. OCLC: ocn956340441.

Biørn-Hansen, A., Majchrzak, T. A., and Grønli, T.-M. (2018). Progressive Web Apps for the Unified Development of Mobile Applications. In Majchrzak, T. A., Traverso, P., Krempels, K.-H., and Monfort, V., editors, *Web Information Systems and Technologies*, volume 322, pages 64–86. Springer International Publishing, Cham.

Croom, C. and Baker, G. (2019). Building the new Twitter.com. https://blog.twitter.com/engineering/en_us/topics/infrastructure/2019/buildingthenewtwitter.html. Retrieved 2019-08-02T09:20:08Z.

Hume, D. A. and Osmani, A. (2018). *Progressive Web Apps*. Manning Publications, Shelter Island, New York. OCLC: on1017992176.

Liebel, C. (2019). *Progressive Web Apps: das Praxisbuch*. Rheinwerk Computing. Rheinwerk Verlag, Bonn, 1 edition.

Malavolta, I. (2016). Beyond native apps: Web technologies to the rescue! (keynote). In *Proceedings of the 1st International Workshop on Mobile Development - Mobile! 2016*, pages 1–2, Amsterdam, Netherlands. ACM Press.

Nguyen, P. H. (2019). Progressive Web App in Enhancing App Experience: Life Care Insurance Claim Application.

Osmani, A. (2015). Getting started with Progressive Web Apps. https://addyosmani.com/blog/getting-started-with-progressive-web-apps/. Retrieved 2019-06-08.

Russell, A. (2015). Progressive Web Apps: Escaping Tabs Without Losing Our Soul – Infrequently Noted. https://infrequently.org/2015/

06/progressive-apps-escaping-tabs-without-losing-our-soul/. Retrieved 2019-08-02.

Wikipedia (2019). Minimum viable product. https://en.wikipedia.org/w/index.php?title=Minimum_viable_product&oldid=906835119. Retrieved 2019-08-07.