BACHELORARBEIT

Analyse der Auswirkung von Progressive Web Apps auf bestehende Web Apps

durchgeführt am
Studiengang Informationstechnik & System–Management
an der
Fachhochschule Salzburg GmbH

vorgelegt von

Refik Kerimi

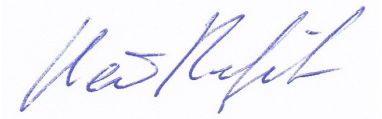
Studiengangsleiter:	FH-Prof. DI Dr. Gerhard Jöchtl
Betreuer:	DI Norbert Egger BSc

Salzburg, September 2018

Eidesstattliche Erklärung

Ich versichere an Eides statt, dass ich die vorliegende Bachelorarbeit ohne unzulässige fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt und alle aus ungedruckten Quellen, gedruckter Literatur oder aus dem Internet im Wortlaut oder im wesentlichen Inhalt übernommenen Formulierungen und Konzepte gemäß den Richtlinien wissenschaftlicher Arbeiten zitiert, bzw. mit genauer Quellenangabe kenntlich gemacht habe. Diese Arbeit wurde in gleicher oder ähnlicher Form weder im In- noch im Ausland in irgendeiner Form als Prüfungsarbeit vorgelegt und stimmt mit der durch die Begutachter beurteilten Arbeit überein.

Salzburg, am 1.09.2018



Refik Kerimi

1410555043

Matrikelnummer

Allgemeine Informationen

Vor- und Zuname:	Refik Kerimi
Institution:	Fachhochschule Salzburg GmbH
Studiengang:	Informationstechnik & System-Management
Titel der Bachelorarbeit:	Analyse der Auswirkung von Progressive Web Apps auf bestehende Web Apps
Schlagwörter:	PWA, Manifest, Service Workers, Push Notification, Cach API
Betreuer an der FH:	DI Norbert Egger BSc

Kurzfassung

Abstract

Danksagung

Danken möchte ich vor allem meinem Betreuer für die Unterstützung bei dieser Bachelorarbeit.

Besonderer Dank gilt auch meiner Familie und Freunden, die uns während des Studiums in allen Belangen immer unterstützt haben.

Inhaltsverzeichnis

Abkürzungsverzeichnis

PWA	Progressive Web Application
SP	Smart Phone
JS	Java Script
SHP	Smart Home Prototypen
NA	Native App
WA	Web App
HTML5	Hypertext Markup Language
CSS	Cascading Style Sheets
HyApp	Hybrid App
SW	Service Worker

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

1 Einleitung

Durch die Markteinführung des Smart Phones(SP) hat sich unser Leben gravierend geändert. Nicht nur unsere Kommunikation, sondern unser Leben im Allgemeinen, ist durch dieses kleine Wundergerät erleichtert worden. Wir haben ständig das SP im Einsatz, zum Organisieren, zum Spielen, zum Musik hören, um unsere Kontakte zu pflegen und ab und zu wird es auch zum Telefonieren verwendet. Das SP hat nicht nur unseren Alltag beeinflusst, sondern auch das Internet und die Entwicklung von Webapplikationen. Kurz nach der Erfindung des smarten Handys kam ein weiterer Markt hinzu, der sich parallel dazu entwickelt hat. Es wurden neue Berufe gegründet wie z.B.: der Native App Entwickler. Native Apps werden speziell an das Betriebssystem angepasst und können somit im Gegensatz zu einer Standard Web Applikation die Ressourcen eines Mobilen Gerätes optimal nutzen. Das Ganze benötigt natürlich eigene Entwickler die sich auf die jeweiligen Plattformen spezialisieren. Dies führt zu höheren Entwicklungskosten, wenn man das Produkt auf verschiedenen Plattformen betreiben will. In den letzten Jahren wurde auch, durch die immer besseren werdenden Browser, die Webapplikation stetig weiter verbessert und durch Technologien wie den Progressive Web Apps sind diese heute schon in der Lage mit den Native Apps zu konkurrieren.

1.1 Motivation

Wie im vorigen Kapitel beschrieben werden native Applikationen für ein bestimmtes Betriebsystem optimiert. Diese haben dann den Vorteil die Hardware des Gerätes nutzen zu können. Dadurch sind komplexe Anwendungen realisierbar. Doch diese sind relativ kostspielig und auf Grund der vielzahl der diversen Apps in den Appstores nicht gerade lukrativ. Durchschnittlich werden (genaue Prozentzahl ermitteln) monatlich pro Nutzer runtergeladen und benötigen viel Speicherplatz auf dem Gerät. Die Progressive Web Application(PWA) vereint die Vorteile von native App und von Webanwendungen und gibt dem Nutzer ein Gefühl, dass man es mit einer auf das System angepassten Anwendung zu tun hat.

1.2 Zielsetzung

Das Ziel dieser Arbeit ist es, einen Smart Home Prototypen(SHP) zur demonstration zu entwickeln. Dem Prototypen werden die PWA typischen Features wie das hinzufügen auf dem Startbildschirm, Offline arbeiten, die Pushfunktionen, das Zugreifen auf Gerätefunktionen und das chachen über eine Clientseitig integrierte Datenbank, hinzugefügt. An diesen Features sollen die Vorteile, Nachteile, Entwicklung, Betrieb und User Experience betrachtet werden. Basictechnologien und verwendete Frameworks (z.B.: Java Script(JS),ReactJS, NodeJS, Yarn,...) werden in dieser Arbeit nicht behandelt.

2 Grundlagen

Wie in Kapitel 1 beschrieben, hat der stetige Zuwachs von Smart Phones SPs [1] zum Umdenken bei der Planung und beim Entwickeln von Webapplikationen geführt. Zu beginn jedes Projektes steht die Entscheidung an, welche Technologien und Tools zur Entwicklung verwendet werden sollen um die bestmögliche Ergebnis zu erhalten. Wenn die falschen Methoden gewählt werden, kann das zu gravierenden Fehlern in der Applikation führen, die sich erst mit Fortdauer der produktiven Verwendung ersichtlich machen. Entscheidet man sich für eine Anwendung die auf das Betriebssystem zugeschnitten ist oder doch für eine plattformübergreifende Webanwendung. Beide haben Vorteile und Nachteile und diese werden im Zuge dieser Arbeit betrachtet. Der Kern der Arbeit stellt, aber die von Google entwickelten PWA [2] da. Die PWAs sollen den Spagat zwischen diesen beiden Anwendungen schaffen. Eventuell könnte diese neue Form der Appentwicklung die traditionelle Technologien gar zur Gänze ablösen? Der Trend in den letzten Jahren geht in Richtung der Mobilen Nutzung und da ist das Smart Phone klar wie, in Abbildung 2.1 und 2.2 dargestellt, voran.

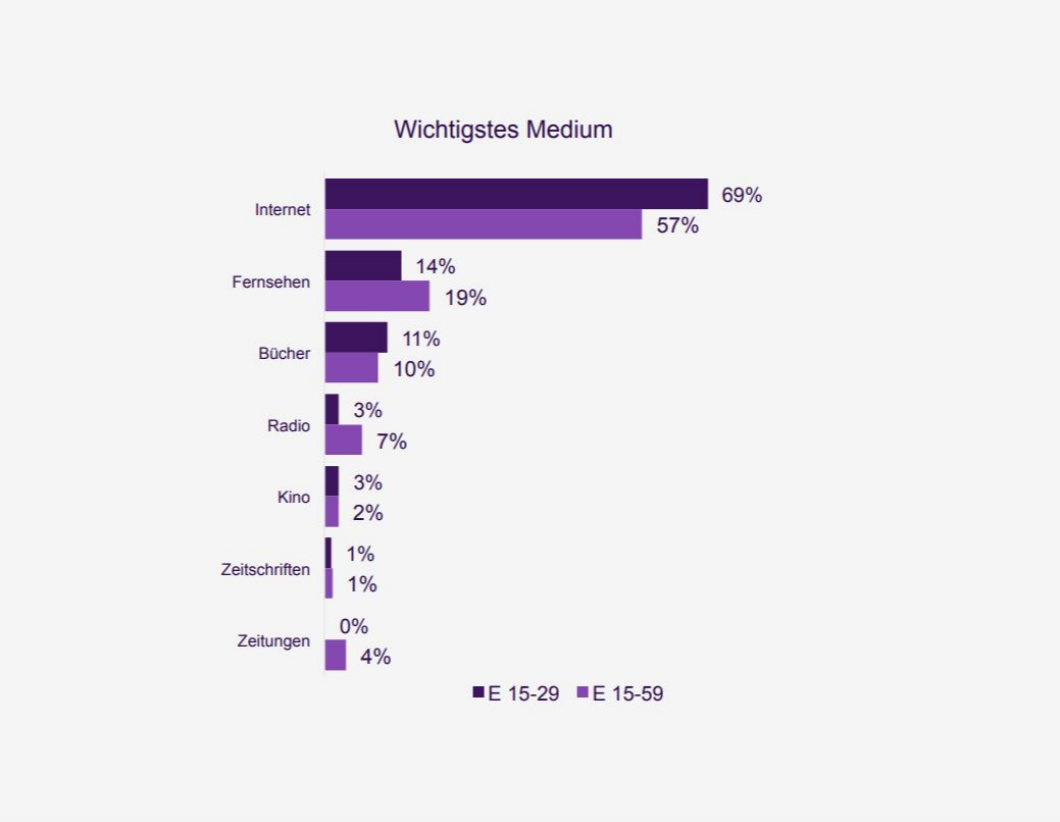


Abbildung 2.1: Internetnutzung [1]

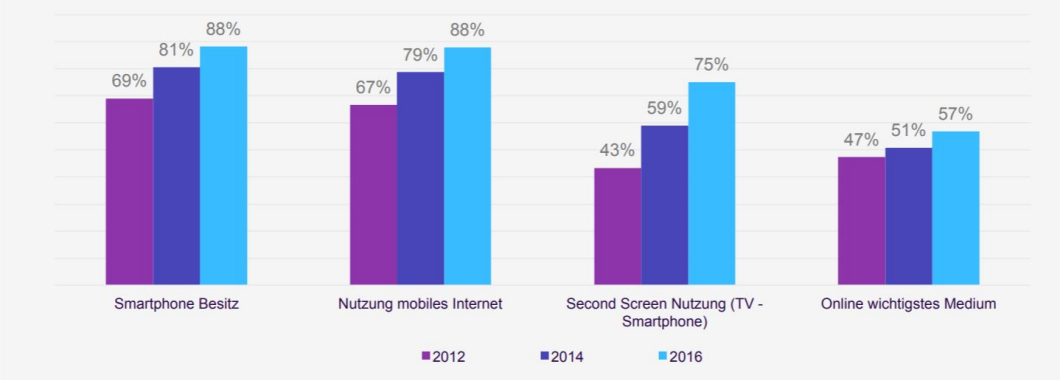


Abbildung 2.2: Smartphonenutzung [1]

2.1 Geschichte Softwareentwicklung

Um die Geschichte der Softwareentwicklung darstellen zu können müssen wir als aller ersten die Frage stellen "Was ist Software?ünd was wie ist eine Software definiert. Diese Frage stellen sich sicherlich alle mal die zum ersten Mal in ihrem Leben mit dieser Technologie in Berührung kommen. Eine genau Definition zu finden ist schwierig da die Software für die Gesamtheit eines Produktes steht . In [3] ist die Softwaretechnik wie folgt definiert:

> *„Zielorientierte Bereitstellung und systematische Verwendung von Prinzipien, Methoden und Werkzeugen für die arbeitsteilige, ingenieurmäßige Entwicklung und Anwendung von umfangreichen Softwaresystemen. Zielorientiert bedeutet die Berücksichtigung z.B. von Kosten, Zeit, Qualität.“([3] Seite 17).*

Im laufe der Jahre wurden verschiedenste Software entwickelt die mehr oder weniger nützlich für unseren Alltag waren. Der Begriff Software wurde 1958 vom US-amerikanischen Statistiker John W. Turkey eingeführt. Zu Beginn bildeten Software und Hardware eine Einheit. Erst nach der Entscheidung durch die US Regierung, dass IBM die Hardware und die Software separat verrechnen werden sollte. Die Software bildet das Gehirn eines eines Computers. Nach der Entscheidung der US-Regierung entstanden erstmals rein Softwareorienttiere Unternehmen wie Microsoft oder SAP [4] [5].

2.2 Mobile Applikationen

Anfang des neues Jahrtausends war die Vorstellung, dass das Mobiltelefon für uns sehr viele alltäglichen Aufgaben erledigt unvorstellbar, doch heute können wir uns das Leben ohne Mobiltelefon kaum vorstellen. Wir organisieren unser Leben damit und steuern unsere Haushaltsgerät, unser Garagentor und verbinden uns mit unserem Auto usw. all diese Möglichkeiten werden durch Apps ermöglicht. Die Apps werden im allgemeinen in 3 Kategorien aufgeteilt Native-, Web- und Hybridapps.

2.2.1 Native Apps

Native Apps (NA) (deutsch; angepasste Anwendung) ist speziell für eine Plattform angepasste Anwendung. Diese werden speziell für ein bestimmtes Betriebssystem konzipiert und haben in der Regel zugriff auf alle Ressourcen eines Gerätes [6]. Hauptsächlich werden zur Programmierung für Mobile Geräte die Hochsprachen Java (Android) und Swift(IOS) verwendet. Native Apps können in App Stores heruntergeladen werden, die bekanntesten sind Apple Store und Google Play [7].

2.2.2 Webapplikationen

Im Gegensatz zu den NAs sind Web App (WA) speziell Programmierte Webseiten [7]. WA funktionieren nach dem Server-Client Prinzip und werden vom Browser aufgerufen. In der Regel werden WAs auf der Basis von JS, CSS und HTML5 entwickelt. Die Verarbeitung erfolgt auf dem Webserver oder auf der Cloud. Client seitig werden die Ergebnisse der Datenverarbeitung angezeigt. Der größte Vorteil ist sicherlich der unkomplizierte Zugang im Gegensatz zu den NAs [8]. Durch die Einführung von Responsive Frameworks wie z.B.: Bootstrap, SemantikUI oder Foundation um nur die bekanntesten zu nennen, wurde die Webentwicklung vielseitiger in der Verwendung. Durch diese Technologien können viele Gerätegrößen mit wenig Aufwand abgedeckt werden [9].

2.2.3 Hybridapplikationen

Hybrid Apps (HyApp) verbinden die Eigenschaften den in Kapitel 2.2.1 und 2.2.2 genannten Technologien. Zum einen verwenden Sie die Webbasierende Client-Server Technologie zum anderen kann man mit einer HyApp auf Gerätefunktioonen wie Kamera und Kalender zugreifen [10].

2.2.4 Progressive Web Apps

Progressive Web Application sind im Grunde eine Weiterentwicklung von einer WA. Diese Technologie der Webentwicklung wird durch die immer schneller Wachsende Welt der Webapplikationen immer wichtiger. Dem User wird das Gefühl gegeben er arbeitet mit einer NA. Das Herausragende dabei ist es,im Gegensatz zu einer HyApp, dass jede bestehende WA in eine PWA umgebaut werden kann. Durch hinzufügen eines Manifest Files und eines Service Worker (SW) werden Features hinzugefügt, die es ermöglichen Offline zu arbeiten oder das Icon der App auf den Desktop oder Home-Bildschirm zu speichern [2]. Google definiert die PWA wie folgt:

- **Progressive** - Funktioniert für alle User unabhängig vom Browser
- **Responsive** - Passt sich jedem Gerät an
- **Verbindungsunabhängig** - funktioniert auch bei schlechtem oder gar keinem Internetzugang
- **App-like** - fühlt sich an wie eine NA
- **Akteull** - Durch die Wartung des SW immer auf dem aktuellsten Stand
- **Sicher** - wird nur über HTTPS bereitgestellt
- **Erkennbar** - erkennbar dank das W3C Manifest durch Suchmaschinen
- **Wiedereinschaltbar** - wird durch die Funktion Push Notfication
- **Installierbar** - Ermöglicht das hinzufügen auf dem Startbildschirm
- **Verteilbar** - Einfache Freigabe über URL [11]

.

3 Basistechnologien

3.1 Aufbau PWA

IN diesem Kapitel werden die Vorteile/Nachteile im Vergleich zu den Native Apps aufgelistet und der Aufbau einer PWA erklärt.

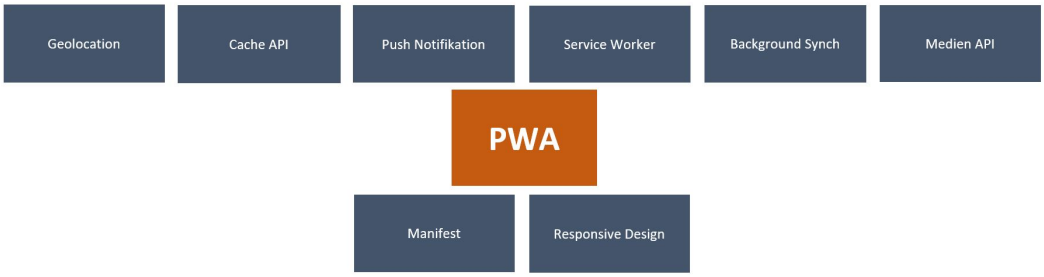


Abbildung 3.1: PWA Komponenten

3.1.1 Tabelle

3.2 Web App Manifest

Das App Manifest ist ein JSON File verrät dem Browser wie sich die WA, bei der Installation auf dem Startbildschirm, verhält. Im Manifest wird der Name,der Kurzname, die Größe, Aussehen der Icons und weitere Eigenschaften definiert diese im Kapitel **??** näher erklärt werden.

3.2.1 Bereitstellung des Web App Manifest

das App Manifest.json file wird in die gleiche Ebene wie die Index.html Datei in das Projekt eingepflegt und und übre den folgenden Link-Tag in der Index Datei bereitgestellt:

```
1 <link rel="manifest" href="/<Dateinname>">
```

Listing 3.1: Manifest.json

Der Aufbau der Manifest ist wie in Listening (Nr) gezeigt aufgebaut:

[**?**]

3.3 Service Worker

Der Service Worker (SW) ist ein Script das der Browser im Hintergrund ausführt [**?**]. Mit der Hilfe des SW ist es möglich die WA offline zu betreiben, Push Notifikation zu erhalten, gecachte Daten abzurufen. SW verhalten sich wie Proxy-Server, welche in einer Zwischenschicht vom Browser und den Netzwerk sitzen. Ein SW wird von einem Worker-Kontext [12] ausgeführt, hat keinen DOM Zugriff und wird als Haupt-Java Script Thread verwendet [13].

3.3.1 Basis Architektur

Der Cyclus eines SWs ist von der Webseite getrennt. In der Installationsphase werden benötigte statische Datein zwischengespeichert erst danach ist der SW installiert. Die Installation erfolgt über die JavaScript-Funktion:

```
1 navigator.serviceWorker.register
```

Listing 3.2: Service Worker Navigator

Danach folgt die Aktivierungsphase, in dieser Phase werden alte Cache-Inhalte verwaltet und Aktualisiert.

Um die neuen Seiten zu steuern muss der SW erneut geladen werden. In der Abbildung 3.3 ist eine vereinfachte Erstinstallation zu sehen:

3.3.2 Registrierung Service Worker

Um den SW zu registrieren muss folgender JS-Code in das Projekt im (genauen Pfad rausfinden) integriert werden.

```
1  if ('serviceWorker' in navigator) {
2    window.addEventListener('load', function() {
3      navigator.serviceWorker.register('/sw.js').then(function(
           registration) {
4        // Registration was successful
5        console.log('ServiceWorker registration successful with
             scope: ', registration.scope);
6      }, function(err) {
7        // registration failed :(
8        console.log('ServiceWorker registration failed: ', err);
9      });
10   });
```

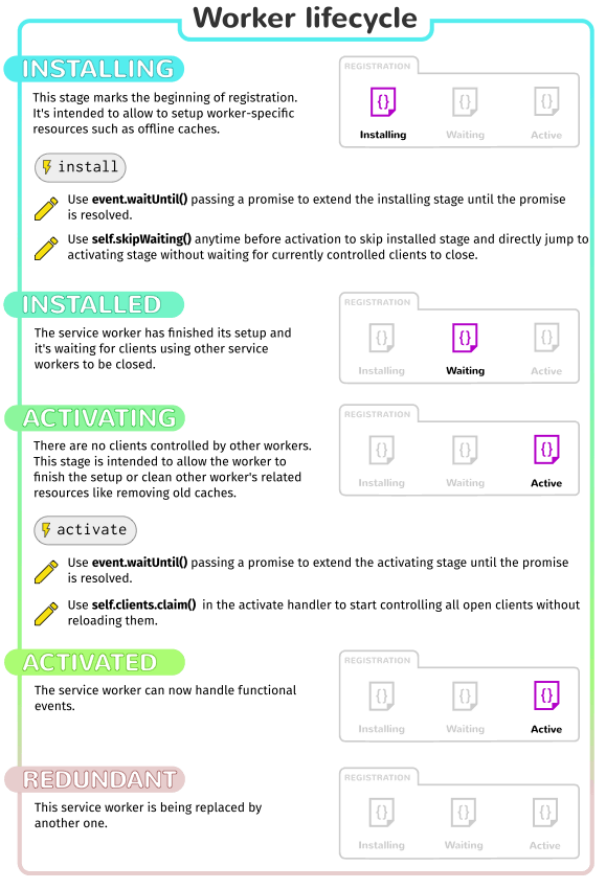


Abbildung 3.2: Basis Architektur Service Worker
[?]

```
11 }
```

Listing 3.3: Service Worker Register

Hier wird die Unterstützung durch den Browser geprüft.

Der SW kann nach der Übernahme der Steuerung zwei Zustände übernehmen, entweder dieser wird beendet oder er übernimmt die Verwaltung der Netzwerkanfragen und der Nachrichten [?].

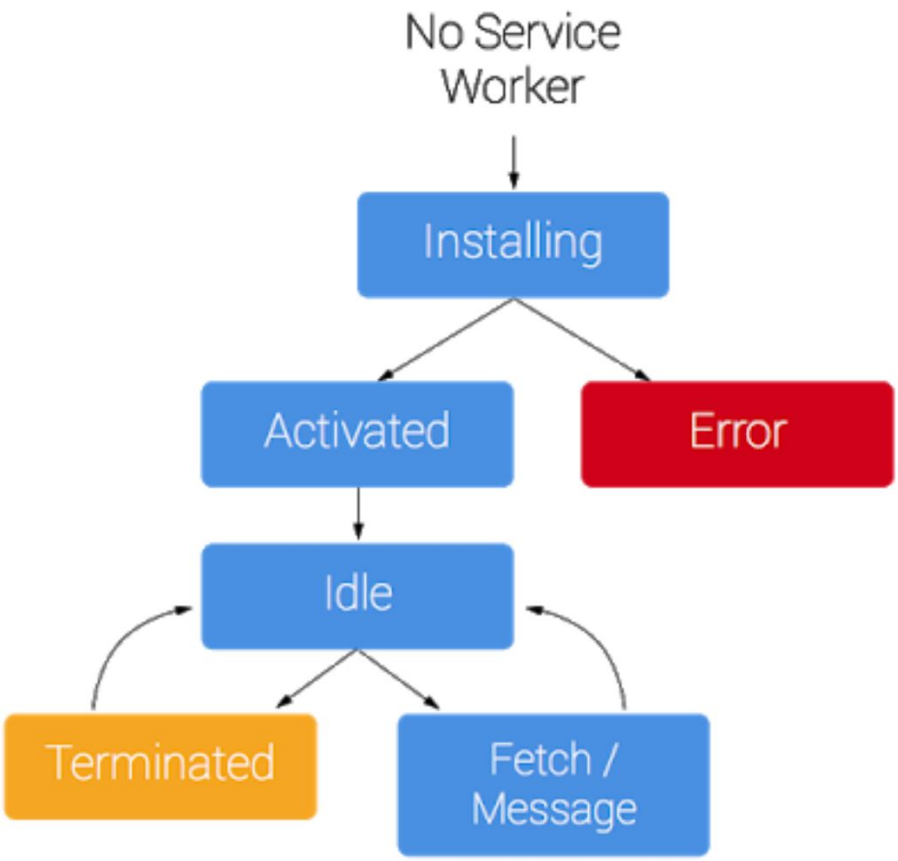


Abbildung 3.3: Erstinstallation Service Worker
[?]

3.4 Push Notifikation

Um dem User bei einer PWA das Gefühl einer Native App aufkommen zu lassen ist die Push Funktion unablässig. Erst durch diese Funktion in Kombination mit dem SW gibt der Web App die persönliche Nähe zum User [14].

3.4.1 Registrierung Push Notifikation

Um die Push Funktion zu integrieren muss die Registerfunktion des SW wie folgt erweitert werden:

```
1
2 if ('serviceWorker' in navigator && 'PushManager' in window) {
3   console.log('Service Worker and Push is supported');
4
5   navigator.serviceWorker.register('sw.js')
```

```
6     .then(function(swReg) {
7       console.log('Service Worker is registered', swReg);
8
9       swRegistration = swReg;
10    })
11    .catch(function(error) {
12      console.error('Service Worker Error', error);
13    });
14  } else {
15    console.warn('Push messaging is not supported');
16    pushButton.textContent = 'Push Not Supported';
17  }
```

Listing 3.4: Push Notifications

Hier wird der Support der Pushfunktionen durch den Browser überprüft wie in Kapitel **??** die Browserunterstützung vomSW und die Push Benachrichtigung. Bei fehlerlosen Durchlauf wird die SW.js Datei registriert [14].

3.4.2 Cache API

3.4.3 Geolocation

https://appdevelopermagazine.com/5877/2018/3/1/progressive-web-apps-vs-native-apps:-showdown-in-2018/

3.4.4 Camera API

3.4.5 Browser

4 Entwurf

In diesem Kapitel wird das Muster im Allgemeinen und die Anforderungen bzw. die Umsetzung der Progressive Web Applications betrachtet.

4.1 Model Native

4.2 Model PWA

4.3 Anforderungsanalyse

5 Implementierung

5.1 Umsetzung der Anforderungen

5.2 Ausgewählte Programmiersprache und IDE

5.3 Manifest

5.3.1 Aufbau

5.3.2 Implementierung

5.4 Aufbau Service Worker

5.4.1 Aufbau

5.4.2 Implementierung

5.5 Zugriff Cache API

5.5.1 Aufbau

5.5.2 Implementierung

5.6 Zugriff Device API

5.6.1 Aufbau

5.6.2 Implementierung

5.7 Fetch API

5.7.1 Aufbau

5.7.2 Implementierung

5.8 IndexDB

5.8.1 Aufbau

5.8.2 Implementierung

6 Funktionstest/Validierung

6.1 Ausgangsbedingung und Ausgrenzung

6.2 Komponententest

6.2.1 Add to Homescreen

6.2.2 Funktion Service Worker

6.2.3 Cache

6.2.4 Kamera

6.2.5 Geolocation

7 Fazit

Literaturverzeichnis

[1] Mindshare, *Über welche der folgenden Geräte nutzen Sie das Internet?*, https://de-statista-com.ezproxy.fh-salzburg.ac.at/statistik/daten/studie/742449/umfrage/umfrage-zur-internetnutzung-nach-geraetetyp-in-oesterreich-nach-alter/ (2018).

[2] Google Developers, *Progressive Web Apps*, https://developers.google.com/web/progressive-web-apps/ (28.06.2018).

[3] H. Balzert, *Lehrbuch der Softwaretechnik: Softwaremanagement*, 2. Aufl. Heidelberg, Neckar: Springer Spektrum, 2008.

[4] Microsoft Corporation, *Microsoft Fast Facts*, https://news.microsoft.com/de-de/fast-facts/ (2018).

[5] SAP SE, *SAP: 46 Jahre Innovation*, https://www.sap.com/corporate/de/company/history.html (2018).

[6] App Entwickler Verzeichnis, *Native Apps vs. Web Apps - Unterschiede und Vorteile*, https://app-entwickler-verzeichnis.de/faq-app-entwicklung/11-definitionen/586-unterschiede-und-vergleich-native-apps-vs-web-apps-2 (2018).

[7] Margaret Rouse, Alexander Gillis, *DEFINITION native App*, https://searchsoftwarequality.techtarget.com/definition/native-application-native-app (2013).

[8] Stephan Augsten, *Defintion „Webanwendung“ Was ist eine Web App?*, https://www.dev-insider.de/was-ist-eine-web-app-a-596814/ (20.04.2017).

[9] Anton Shaleynikov, *Top 5 Most Popular CSS Frameworks that You Should Pay Attention to in 2017*, https://hackernoon.com/top-5-most-popular-css-frameworks-that-you-should-pay-attention-to-in-2017-344a8b67fba1 (2018).

[10] Beratung FLYACTS, *Hybrid-Apps – Definition, Eigenschaften, Einsatzorte, Vorteile und Beispiele*, https://www.flyacts.com/hybrid-apps-definition-eigenschaften-einsatzorte-vorteile-und-beispiele (03.12.2013).

[11] Google Developers, *Your First Progressive Web App*, https://codelabs.developers.google.com/codelabs/your-first-pwapp/index.html#0 (2018).

[12] Dennis Sterzenbach, *Service Worker API*, https://developer.mozilla.org/de/docs/Web/API/Worker (21.12.2017).

[13] Bitbruder, TobiDo, Heniz, *Service Worker API*, https://developer.mozilla.org/de/docs/Web/API/Service_Worker_API (30.01.2018).

[14] Google Developers, *4.3.1 Einführung in Push-Benachrichtigungen im Web und Benachrichtigungen*, https://support.google.com/partners/answer/7336533?hl=de&ref_topic=7327985 (2018).